Кравець Валерій Олексійович –
проректор з науково-педагогічної роботи Національного технічного університету «Харківський політехнічний інститут».
Завідувач кафедри «Системи інформації». Кандидат технічних наук, професор.
Має більше 100 наукових публікацій.
Область наукових досліджень: методи обробки цифрової інформації.

Сокол Євгеній Іванович –
проректор з науково-педагогічної роботи Національного технічного університету «Харківський політехнічний інститут».
Завідувач кафедри «Промислова та біомедична електроніка». Доктор технічних наук, професор.
Має більше 100 наукових публікацій.
Область наукових досліджень: мікропроцесорне управління в силовій електроніці.

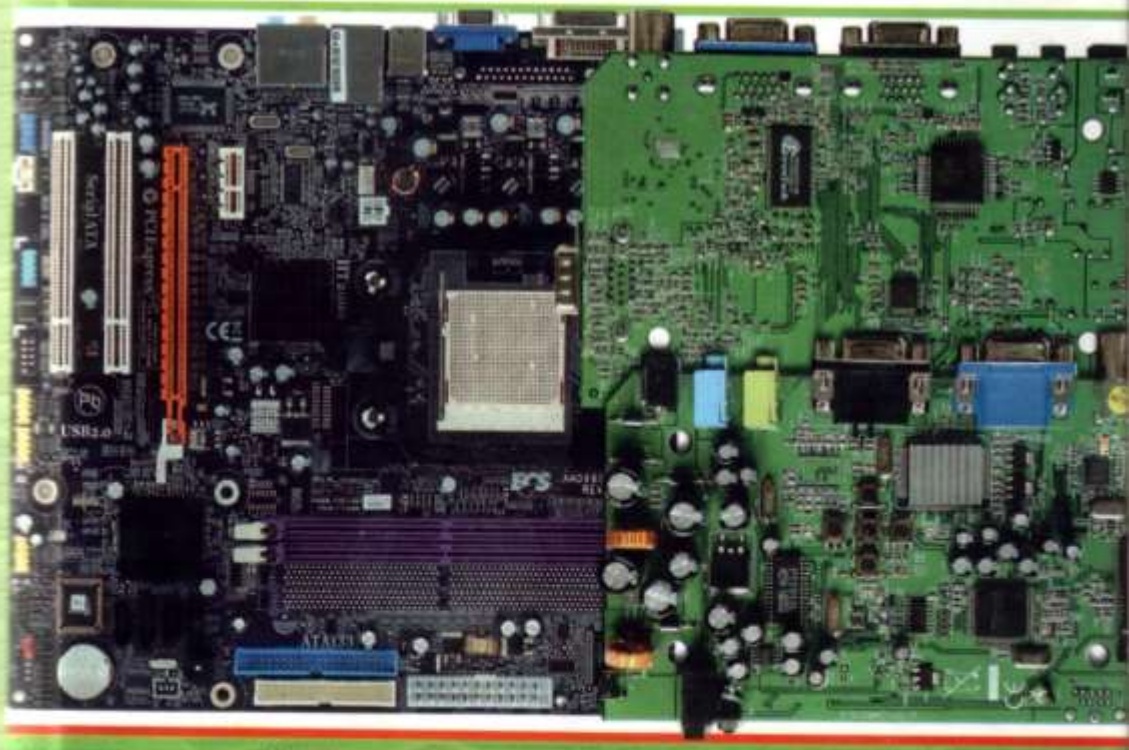
Рисований Олександр Миколайович –
доцент кафедри обчислювальної техніки та програмування Національного технічного університету «Харківський політехнічний інститут» . Кандидат технічних наук, доцент.
Має більше 100 наукових публікацій.
Область наукових досліджень: розвиток теорії нелінійного сигнатурного аналізу для підвищення ефективності діагностування цифрових систем.

ISBN 978-966-593-548-3
9 789665 935483

Друкарня НТУ «ХПІ»

В.О. Кравець, Є.І. Сокол, О.М. Рисований

КОМП'ЮТЕРНА СХЕМОТЕХНІКА



Підручник

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

В.О. Кравець, Є.І. Сокол, О.М. Рисований

КОМП’ЮТЕРНА СХЕМОТЕХНІКА

Рекомендовано Міністерством освіти і науки України
як підручний для студентів
вищих навчальних закладів напрямку
6.0915 “Комп’ютерна інженерія”

Харків НТУ “ХПІ” 2007

ББК 32.973
К78
УДК 004.31

Рецензенти: *І.О. Фурман,* д-р техн. наук, проф., академік АН Вищої школи України, ХНТУСГ;
Г.І. Загарій – д-р техн.. наук, проф., УДАЗЖ;
І.І. Обод – д-р техн. наук, проф., ХУПС ім. Кожедуба.

Гриф надано Міністерством освіти і науки України, лист № 1.4/18-Г-902 від 11.06.07

К78 Кравець В.О., Сокол Є.І., Рисований О.М. Комп'ютерна схемотехніка. Підручник. – Х.: НТУ "ХПІ", 2007. – 480 с.

ISBN 966-593-548-3

В підручнику наведені форми зображення інформації, логічні основи схемотехніки ЕОМ, схемотехніка цифрових елементів, схемотехніка комбінаційних вузлів, схемотехніка цифрових вузлів, схемотехніка обслуговуючих елементів, схемотехніка аналогових та комбінаторних вузлів, схемотехніка мікроконтролерів та запам'ятовуючих пристроїв.

Підручник призначено для студентів, яким викладається навчальна дисципліна "Комп'ютерна схемотехніка", а також може бути корисним усім, хто займається розробкою та використанням сучасної обчислювальної техніки.

Іл.. 336. Табл. 90. Бібліогр. 24 назв.

ББК 32.973

ISBN 966-593-548-3

ЗМІСТ

ВСТУП

Напівпровідникова електроніка бере свій початок з 1948 р., коли групою розробників фірми Bell був створений перший транзистор. Через 11 років інженерами фірми Texas Instruments була розроблена перша мікросхема, яка складалася всього з шести транзисторів, а в 1971 р. фірма Intel розробила перший 4-розрядний мікропроцесор 4004, що містив більше 2000 транзисторів. Надалі мікромініатюризація електронних компонентів набула таких темпів, що це стало приводом для образного порівняння на сторінках журналу Sientific American (1982 р.), у якому висловлювалося припущення, що якби авіапромисловість в останні 25 років розвивалася так же стрімко, як і промисловість засобів обчислювальної техніки, то "Боїнг-767" на 1982 р. коштував би 500 доларів і здійснював би обліт земної кулі за 20 хвилин, витрачаючи при цьому 5 галонів палива. Вражаючі результати, досягнуті в мікроелектроніці, стали можливими завдяки не тільки новітнім напівпровідниковим технологіям, але й величезному багажу схемотехнічних рішень, накопиченому протягом десятиліть багатомільйонною армією розробників. Не зважаючи на величезну кількість транзисторів, зібраних на крихітних напівпровідникових кристалах, слід все-таки пам'ятати, що вони є наборами з простих елементів, розгляду яких і присвячений даний підручник.

Підручник складається з розділів, зміст яких відповідає вимогам бакалаврату "Комп'ютерні науки" з навчальної дисципліни "Комп'ютерна схемотехніка". У підручнику наведені також завдання до самостійних досліджень.

1. ФОРМИ ЗОБРАЖЕННЯ ІНФОРМАЦІЇ

1.1. Характеристика електричних сигналів. Способи електричного відображення двійкових цифр та чисел

Сигнал – це фізичний процес, який відображає повідомлення. У технічних системах найчастіше використовуються електричні сигнали. Вони, як правило, є функціями часу.

1.1.1. Класифікація сигналів

Сигнали можна класифікувати за різними ознаками:

1. ***Безперервні*** (аналогові) – сигнали, які описуються безперервними функціями часу, тобто приймають безперервну безліч значень на інтервалі визначення. ***Дискретні –*** сигнали, які описуються дискретними функціями часу, тобто приймають кінцеву безліч значень на інтервалі визначення.

2. ***Детерміновані –*** сигнали, які описуються детермінованими функціями часу, тобто значення яких визначені у будь-який момент часу. ***Випадкові*** – сигнали, які описуються випадковими функціями часу, тобто значення яких у будь-який момент часу є випадковою величиною. Випадкові процеси (СП) можна класифікувати на стаціонарні, нестаціонарні, ергодичні і неергодичні, а також гауссові, марківські і т.д.

3. ***Періодичні –*** сигнали, значення яких повторюються через інтервал, що дорівнює періоду $x(t)= x(t + nT)$, де $n = 1,2 ,..., \infty$; T – період.

4. ***Каузальні –*** сигнали, що мають початок у часі.

5. ***Фініт ні –*** сигнали кінцевої тривалості, які дорівнюють нулю поза інтервалом визначення.

6. ***Когерент ні –*** сигнали, які збігаються в усіх точках визначення.

7. ***Орт огональні –*** сигнали, протилежні когерентним.

1.1.2. Форми подання інформації

Інформація завжди має вигляд повідомлення, яке передається певним фізичним середовищем. Носієм інформації може бути будь-яке середовище, яке змінює стан залежно від передаваної інформації. Це може бути папір, на якому інформація зображається знаками або спеціальними позначками (наприклад, перфорація); магнітний матеріал (стрічка, диск і т.і.), стан якого змінюється за допомогою магніту; електричний сигнал, у якого змінюється будь-який параметр (частота, амплітуда).

Відповідно до особливостей структури часового зображення всі радіотехнічні сигнали можна розділити на *аналогові, дискретні* і *цифрові* (рис. 1.1).

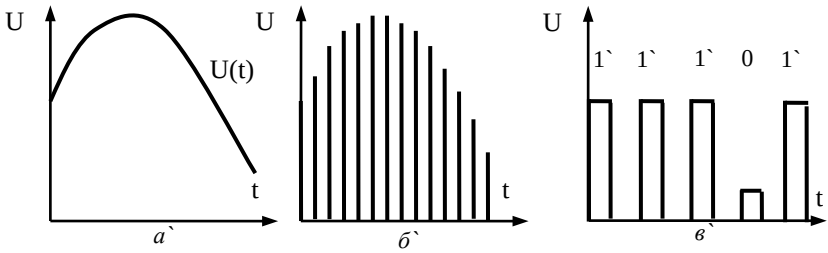


Рис. 1.1. Форми радіотехнічних сигналів:
а –аналоговий; *б* – дискретний; *в* – цифровий

За виглядом електричних сигналів на діючих виходах логічних елементів у різних дискретних пристроях застосовуються три способи подання інформації електричними сигналами:

- потенційний або статичний (рис. 1.2, *а*);
- імпульсний (рис. 1.2, *б*);
- динамічний або потенціально-імпульсний (рис. 1.2, *в*).

При **потенційному** способі подання інформації двом значенням логічної величини “0” і “1” відпові-

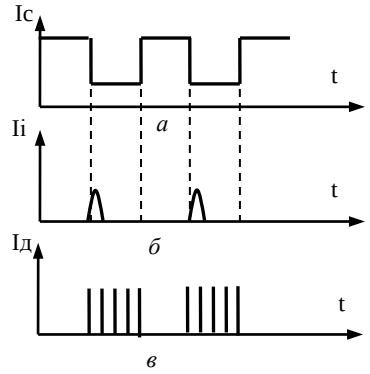


Рис. 1.2. Форми подання інформації

дають низький і високий рівні потенціалу (напруги). Такий сигнал зберігає незмінний рівень (нульовий або одиничний) протягом деякого відрізка часу (тобто деякої кількості тактів синхронізації), коли значення двійкової змінної не змінюється. У перехідні проміжки часу значення сигналу залишається невизначеним. Таким чином, у логічних схемах потенційного типу сигнали є напругами певної величини, які існують весь час, поки відповідні їм логічні змінні зберігають своє значення.

При **імпульсному** способі подання інформації одиничному і нульовому значенням двійкової змінної відповідають наявність (відсутність) електричного (рис. 1.2, *б*), позитивного (наприклад, "1") чи негативного імпульсів (наприклад, "0"). У ідеальному випадку імпульси повинні мати прямокутну форму і з'являтися в тактові моменти часу дискретного часу. Насправді імпульси відрізняються від прямокутних і зсуваються відносно тактового моменту на деякий час затримки.

При **потенційно-імпульсному** (динамічному) способі подання інформації двом можливим значенням змінної відповідає наявність або відсутність періодичної серії сигналів, наприклад, імпульсних, які заповнюють весь інтервал подання логічної змінної або змінних (рис. 1.2, *в*).

Часто при реалізації різних логічних схем з декількома входами і виходами (особливо схем, тригерів) зустрічається так зване парафазне подання сигналу, яке є одним з видів потенційного способу подання інформації. На рис. 1.3 показаний спосіб завдання парафазного сигналу для входів x_1 та x_2.

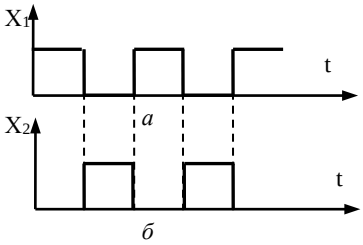


Рис. 1.3. Парафазний сигнал для двох входів

На практиці допустимими є значні зміни рівнів одиничного і нульового сигналів, при яких не виникає небезпеки помилкової оцінки значень змінної.

У зв'язку з цим залежно від кодування сигналів розрізняють позитивну і негативну логіку. При позитивній логіці високому рівню напруг ставлять у відповідність "1", при негативній – навпаки.

Вибір логіки при кодуванні логічної інформації має велике значення, оскільки це приводить до різного тлумачення логічної функції, яку може реалізувати логічний елемент.

Можливість передачі повідомлення за допомогою електричного сигналу реалізується за допомогою каналу зв'язку, який сполучає джерело і приймач інформації (рис. 1.4). Щоб передати інформацію, необхідно її заздалегідь перетворити.

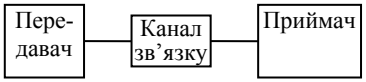


Рис. 1.4. Інформаційна модель каналу зв'язку

Кодування – це процес перетворення повідомлення у форму, зручну для передачі по даному каналу. Як простий приклад можна привести передачу повідомлення у вигляді телеграми. Всі символи кодуються за допомогою телеграфного коду.

Декодування – операція відновлення прийнятого повідомлення. У систему зв'язку необхідно ввести пристрої для кодування і декодування інформації. Теоретичне обґрунтування таких систем дав у своїх роботах К. Шеннон. Він показав ефективність введення кодуючих і декодуючих пристроїв, призначення яких полягає в узгодженні властивостей джерела інформації з властивостями каналу зв'язку. Одне з них (що кодує пристрій, або кодер) повинне забезпечити таке кодування, при якому шляхом усунення надмірності інформації істотно знижується середня кількість символів, що доводиться на одиницю повідомлення. За відсутності перешкод це безпосередньо дає виграш в часі передачі або в об'ємі пристрою, що запам'ятовує. Таке кодування називають ефективним (або оптимальним), оскільки воно підвищує ефективність системи. За наявності перешкод у каналі передачі воно дозволяє перетворити вхідну інформацію в послідовність символів, що найкращим чином відповідає завданням подальшого перетворення. Інший кодуючий пристрій (кодер каналу) забезпечує задану достовірність при передачі або зберіганні інформації шляхом введення додаткової надмірності інформації. Таке кодування називають надмірним або перешкодостійким. ***Перешкодостійкість*** досягається завдяки урахуванню інтенсивності перешкод та їх статистичних закономірностей.

Параметри імпульсів. Вимірювання параметрів імпульсів є процесом визначення числових значень ряду параметрів – показників. Для різних форм імпульсів характерні певні параметри. На рис. 1.5 наведено імпульс прямокутної форми. На рис. 1.6 наведено імпульс зі спадом у області вершини.

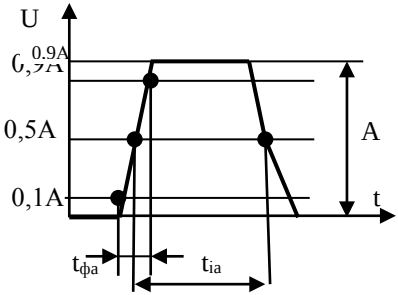


Рис. 1.5. Імпульс прямокутної форми

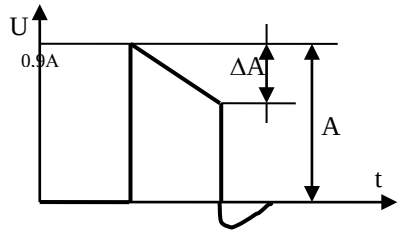


Рис. 1.6. Імпульс зі спадом в області вершини

На рис. 1.7 наведено імпульс дзвіноподібної форми. На рис. 1.8 наведено імпульс з коливаннями в області вершини. Схема визначення періоду імпульсів наведена на рис. 1.9.

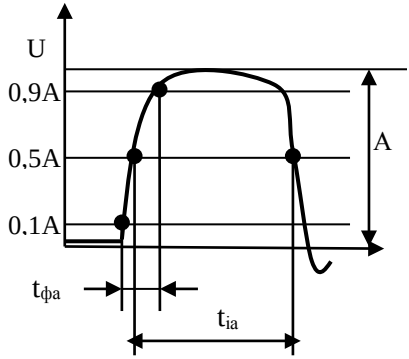


Рис. 1.7. Імпульс дзвіноподібної форми

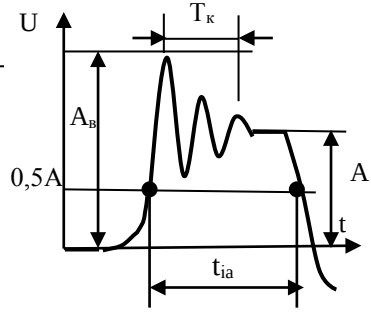


Рис. 1.8. Імпульс з коливаннями у області вершини

Для визначення параметрів імпульсів застосовуються такі позначення:

A – амплітуда (найбільше значення напруги або струму);

$t_{іа}$ – активна тривалість імпульсу (тривалість, вимірювана на рівні половини амплітуди);

$t_{фа}$ – активна тривалість фронту (частина тривалості фронту, вимірювана між рівнями 0,1 і 0,9 амплітуди імпульсу);

ΔA – абсолютне зниження вершини (різниця між амплітудою імпульсу і величиною напруги (струму) у області початку заднього фронту);

$T_к$ – період коливань у області вершини;

$A_в$ – амплітуда викиду (найбільше значення амплітуди імпульсу);

T – період проходження імпульсів (час, відлічений між моментами появи фронтів або максимальних значень);

n – кількість коливань у області вершини (кількість коливань, відлічена з моменту, поки амплітуда коливань знизиться удвічі).

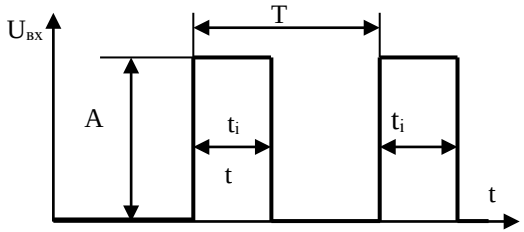


Рис. 1.9. Визначення періоду імпульсів

Деякі параметри можна обчислити:

γ_c – відносне зниження величини ($\gamma_c = \Delta A / A$);

f – частота проходження імпульсів ($f = 1 / T$);

Θ – шпаруватість (скважность – рос.) імпульсного процесу ($\Theta = T / t_{ia}$);

$f_к$ – частота коливань у області вершини ($f_к = 1 / T_к$);

$\gamma_в$ – відносна величина викиду ($\gamma_в = (A_В – A) / A$).

1.2. Розділові, диференціюючі та інтегруючі кола

Інтегруючі та диференціюючі кола широко застосовуються в імпульсній техніці для таких цілей:

Інтегруюче коло використовується:

– для отримання сигналів, пропорційних інтегралу від вхідних сигналів (з певною похибкою);

– для фільтрування високочастотних складових;

– для подовження фронтів імпульсів;

– для збільшення тривалості імпульсів;

– для формування напруг, що лінійно змінюються.

Диференціююче коло використовується:

– для отримання сигналів, пропорційних похідній від вхідних сигналів з певною похибкою;

– для усунення постійної складової сигналу;

– для зменшення тривалості імпульсу;

– для отримання двополярних імпульсів з однополярних.

Складні еквівалентні схеми реальних імпульсних пристроїв після спрощення імпульсних пристроїв часто зводяться до диференціюючих і інтегруючих кіл.

Опис властивостей таких кіл найчастіше використовують у вигляді перехідних, частотних характеристик і реакції на вхідний сигнал, що лінійно змінюється.

1.2.1. Дія імпульсу прямокутної форми на RC-коло

На рис. 1.10 зображені *RC*-коло та епюри напруг u_c на конденсаторі C і u_R на резисторі R при дії на її вхід імпульсу прямокутної форми з амплітудою U і тривалістю t_i [1].

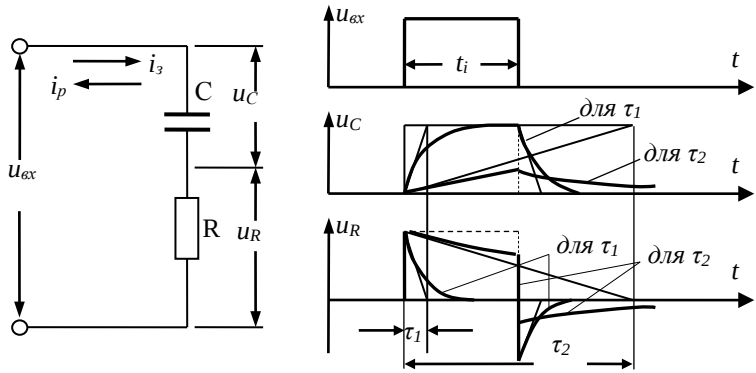


Рис. 1.10. *RC*-коло та епюри напруг на RC-колі

Ці епюри побудовані для двох значень постійної часу кола $\tau = RC$: $\tau_1 < t_i$ і $\tau_2 > t_i$ і виходять з розгляду процесів заряду і розряду ємності через активний опір, які починаються відповідно при дії фронту і спаду імпульсу.

Постійна часу τ має розмірність часу і визначає швидкість

експоненціальних процесів, що протікають при заряді і розряді ємності або індуктивності через активний опір. Постійну часу τ можна визначити як інтервал часу, протягом якого ємність заряджається до 63% постійної напруги джерела або розряджається до 37% початкової напруги на ній.

Теоретично експоненціальні процеси тривають нескінченно довго, але на практиці можна приблизно прийняти їх тривалість рівною 3τ. ***Графічно*** значення τ дорівнює довжині дотичній, яка відповідає початковій точці експоненціальної кривої (см. рис. 1.10).

Якщо вихідну напругу кола знімати з конденсатора C, то вона матиме форму імпульсу з розтягнутими експоненціальними фронтом і спадом. Тривалість цього імпульсу виходить тим більшою, а амплітуда тим меншою, чим більше постійна часу кола.

Виведемо рівняння для RC-кола з **ємнісним** виходом. Зі схеми з'єднання RC елементів (рис. 1.10) випливає:

$$u_{вх} = u_R + u_C ,$$

де $u_R = IR$; $u_C = u_{вих}$; $I = C du_C/dt$.

Підставляємо ці значення у $u_{вх}$:

$$u_{вх} = C\, du_{вих}/dt\, R + u_{вих}.$$

Тому що $\tau = RC$, то

$$u_{вх} = \tau\, du_{вих}/dt + u_{вих}. \qquad (1.1)$$

За умови, що $du_{вих}/dt >> u_{вих}$ (1.2)

отримаємо:

$$\tau\, du_{вих}/dt \approx u_{вх}.$$

З формули (1.2) отримаємо

$$u_{вих} \approx 1/\tau \int u_{вх}\, dt, \qquad (1.3)$$

тобто вихідна напруга є пропорційною інтегралу від вхідного. Тому RC-коло з ємнісним виходом називають **інтегруючим**.

З умови (1.2) виходить, що інтеграція відбувається тим точніше, чим швидше змінюється $u_{вих}$ (чим більше похідна $du_{вих}/dt$) і чим більшою є постійна часу $\tau = RC$. Проте, як випливає з (1.3), при збільшенні τ зменшується величина $u_{вих}$. Отже, чим точнішим є ефект інтеграції, тим він має менше значення. З цієї причини інтегруючий RC-коло майже завжди використовується на практиці не для математичної інтеграції вхідних напруг, а для інших цілей, пов'язаних з наближеною

інтегруючою дією кола.

Оскільки напруга $u_{вих} = u_c$ при дії вхідного сигналу збільшується поступово, то з цього витікає тимчасова умова наближеної інтеграції:

$$\tau >> t, \tag{1.4}$$

де t – тривалість процесу інтеграції.

Зокрема, умовою наближеної інтеграції імпульсу тривалістю t_i є

$$\tau >> t_i . \tag{1.5}$$

Це підтверджується епюрами на рис. 1.9, з яких виходить, що навіть при $\tau = \tau_2 > t_i$ вже відбувається груба інтеграція прямокутного імпульсу за час його дії.

Якщо вихідну напругу кола знімати з **резистора** R, то при $\tau = \tau_2 < t_i$ в моменти дії фронту і спаду вхідного імпульсу на виході виникають два укорочені експоненціальні імпульси різної полярності; при $\tau = \tau_2 > t_i$ на виході вийде майже прямокутний імпульс з тією ж тривалістю t_i, але з вершиною, що знижується за експоненціальним законом, і експоненціальним "хвостом" протилежної полярності.

Виведемо значення $u_{вих}$.

Зі схеми з'єднання RC елементів (рис. 1.10) випливає:

$$u_{вх} = u_R + u_C ,$$

де $u_R = u_{вих}$.

Підставимо $u_R = u_{вих}$ у формулу визначення $u_{вх}$ та отримаємо з отриманого виразу диференціал:

$$du_{вх}/dt = du_R/dt + du_C/dt \tag{1.7}$$

З формули $I = C\, du_C/dt$ виходить, що $du_C/dt = I/C$.

Також відомо, що $I = u_R/R$, тому замінивши $u_R = u_{вих}$, підставимо отримані значення у (1.7):

$$du_{вх}/dt = du_{вих}/dt + 1/r\, u_{вих}.$$

Якщо

$$\frac{du_{вих}}{dt} << \frac{1}{\tau} u_{вих} ,$$

$$\text{то } u_{вих} \approx \tau\, du_{вх}/dt, \tag{1.8}$$

тобто RC-коло, якщо значення $u_{вих}$ знімати з резистора, – є **диференціюючим**.

З умови (1.7) виходить, що диференціювання відбувається тим точніше, чим повільніше змінюється $u_{вих}$ (чим менше похідна $du_{вих}/dt$) і чим менше постійна часу τ. Проте, як виходить з (1.8), при зменшенні τ зменшується величина $u_{вих}$.. Отже, чим точніший ефект диференціювання, тим він менше. З цієї причини диференціююче RC-коло використовується на практиці не для математичного диференціювання вхід-

них напруг, а лише для скорочення імпульсів. Тому, враховуючи, що напруга $u_{вих} = U_R$ при дії фронту вхідного сигналу спадає поступово, тимчасова умова наближеного диференціювання може бути записана у вигляді

$$\tau << t, \tag{1.9}$$

де t – тривалість процесу диференціювання.

Зокрема, умовою наближеного диференціювання імпульсу тривалістю t_i є

$$\tau << t_i. \tag{1.10}$$

Це підтверджується епюрами на рис. 1.1, з яких виходить, що навіть при $\tau = \tau_i < t_i$ вже здійснюється грубе диференціювання прямокутного імпульсу. За умови ж

$$\tau >> t_i \tag{1.11}$$

на підставі (1.6) одержимо $u_{вих} \approx u_{вх}$, тобто дане коло передає вхідний імпульс майже без спотворень (див. епюру на рис. 1.9 для $\tau = \tau_2 > t_i$). При цьому із-за наявності конденсатора C через коло не передається постійна напруга. Тому *RC*-коло при виконанні умови (1.11) називається перехідним або розділовим і застосовується для передачі з виходу одного каскаду на вхід іншого при розділенні цих каскадів за постійною складовою.

1.2.2. Дія імпульсу прямокут ної форми на RL-коло

На рис. 1.11 зображені *RL*-коло і епюри напруг при дії на її вхід імпульсу прямокутної форми. Ці епюри також побудовані для двох значень постійного часу кола $\tau = L/R$ ($\tau_1 < t_i$ і $\tau_2 > t_i$) і виходять з розгляду процесів заряду і розряду індуктивності L через активний резистор R.

Якщо вихідну напругу *RL*-кола знімати з резистора R (рис. 1.12, *в*), то епюри напруги $u_{вих} = u_R$ в цьому випадку відповідають епюрам напруги $u_{вих} = u_C$ для *RC*-кола (див. рис. 1.11).

Тому таке *RL*-коло є інтегруючим, причому умовою наближеної інтеграції імпульсу з тривалістю t_i є те ж співвідношення (1.5) $\tau >> t_i$, що і для інтегруючого *RC*-кола.

Якщо ж вихідну напругу *RL*-кола знімати з індуктивності L (рис. 1.12, *г*), то епюри напруги $u_{вих} = u_C$ в цьому випадку відповідають епюрам напруги $u_{вих} = u_R$ для *RC*-кола (рис. 1.11, *в*).

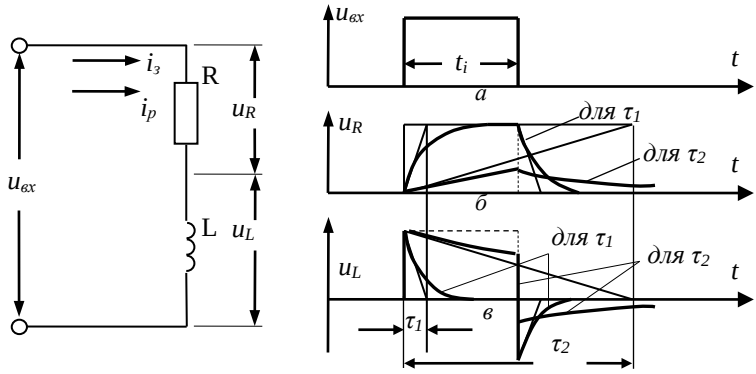


Рис. 1.11. Дія прямокутного імпульсу на *RL*-коло ($\tau_1 < t_i$, $\tau_2 > t_i$)

Тому таке *RL*- коло є диференціюючим, причому умовою наближеного диференціювання імпульсу з тривалістю t_i є те ж співвідношення (1.10) $\tau_i << t_i$, що і для диференціюючого *RC*-кола.

1.3. ПРАКТИЧНЕ ЗАСТОСУВАННЯ ІНТЕГРУЮЧИХ *RC*-КІЛ

1.3.1. Отримання напруги, що лінійно змінюється в інтегруючих RC-колах

Єдине застосування інтегруючого *RC*-кола (рис. 1.12, *а*) полягає у отриманні напруг, що лінійно змінюються в часі.

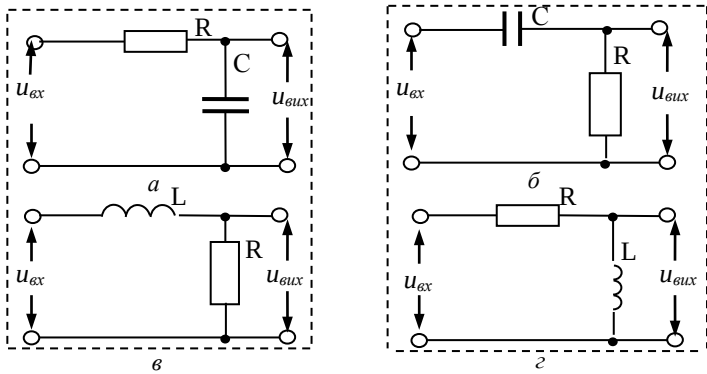


Рис. 1.12. Основні види простих кіл:
а – інтегруюче *RC*-коло; *б* – диференційне або перехідне *RC*-коло;
в – інтегруюче *RL*-коло; *г* – диференційне *RL*-коло

Для отримання напруги, що лінійно змінюється, потрібно точно проінтегрувати постійну напругу $u_{вх} = E =$ const:

$$u_{вих} = \frac{1}{\tau}\int_0^t E dt = \frac{E}{\tau} t . \qquad (1,12)$$

Реально напруга на конденсаторі $u_C = u_{вих}$ при його заряді через резистор R від джерела постійної напруги E змінюється за експоненціальним законом:

$$u_C = E(1 - e^{-t/\tau}). \qquad (1.13)$$

На рис. 1.13 видно, що лише початкова ділянка вихідної напруги може використовуватися як лінійно змінювана, причому лінійність цієї ділянки тим краще, чим вона коротше в порівнянні з постійною часу кола τ.

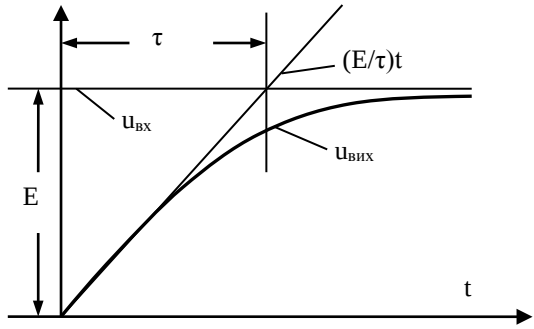


Рис. 1.13. Залежність $u_{вих}$ від часу змінювання сигналу

Але при цьому зменшується максимальна величина вихідної напруги: $u_{вих} << E$.

1.3.2. Збільшення фронт ів імпульсу в інт егруючих RC-колах

Форма імпульсу, що виникає на виході інтегруючого *RC*-кола при дії прямокутного імпульсу, показана на рис. 1.10.

За час дії імпульсу t_i відбувається заряд конденсатора С через резистор *R,* а після закінчення імпульсу – повний розряд конденсатора С через цей же резистор. Тому фронт вихідного імпульсу описується виразом

$$u_ф = u_{вх}(1 - e^{-t/\tau}), \qquad (1.15)$$

а його спад – виразом

$$u_{сп} = u_{вих} e^{-((t-ti)/\tau)}, \quad (1.16)$$

де

$$u_{вих} = u_{вх}(1 - e^{-(ti/\tau)}) \quad (1.17)$$

є амплітудою вихідного імпульсу.

Таким чином, відбувається збільшення і фронту і спаду імпульсу. За рахунок розтягування спаду тривалість вихідного імпульсу завжди перевищує тривалість вхідного імпульсу ($t_{i\ вих} > t_i$). Тому інтегруюче *RC*-коло, що працює в описаному режимі, зазвичай називається колом, що розтягує.

Тривалість розтягнутого фронту вихідного імпульсу визначається часом повної зарядки конденсатора $t_ф \approx 3\tau$, якщо тривалість вхідного імпульсу достатньо велика ($t_i > 3\tau$); якщо ж $t_i < 3\tau$, то зарядка конденсатора за час дії імпульсу не встигає закінчитися і $t_ф = t_i$. **Тривалість збільшеного спаду вихідного імпульсу відповідає часу повної розрядки конденсатора $t_{сп} \approx 3\tau$.**

1.3.3. Згладжувальні фільтри в інтегруючих RC-колах. Виділення постійної складової

Інтегруючі *RC*-кола часто застосовуються як фільтри, що згладжують паразитні пульсації або флуктуації постійних напруг. Властивості інтегруючого *RC*-кола щодо фільтрації визначають її амплітудно-частотну характеристику (АЧХ):

$$K(\omega) = U(\omega)_{вих}/U_{вх} = 1/\sqrt{1+\ \omega^2 C^2 R^2} = 1/\sqrt{1+\omega^2\tau^2}\ , \quad (1.18)$$

де $U(\omega)_{вих}$, $U_{вх}$ – амплітуди гармонійної напруги на вході і виході кола; ω – частота гармонійної напруги.

Ця характеристика (рис. 1.14) показує, що **при проходженні через це коло гармонійної напруги відбувається зменшення амплітуди при збільшенні частоти.**

Дійсно, для постійної напруги (ω = 0) конденсатор, включений паралельно вихідним клемам кола, є нескінченно великим опором $Xc = 1/\omega C = \infty$.

Тому постійна напруга повністю відтворюється на виході кола. Зі збільшенням ω опір конденсатора зменшується, внаслідок чого амплітуда вихідної напруги падає.

Якщо $\omega \to \infty$, то $Xc \to 0$, тобто вихід кола замикається накоротко і вихідна напруга стає практично рівною нулю. Тому АЧХ прямує до нуля.

Таким чином, **інтегруюче *RC*-коло є фільтром нижніх частот (коло пропускає частоти $\omega < \omega_в$**, де $\omega_в$ – верхня гранична частота смуги пропускання). Значення смуги пропускання визначається на підставі (1.18), звідки $\omega_в = 1/\tau$.

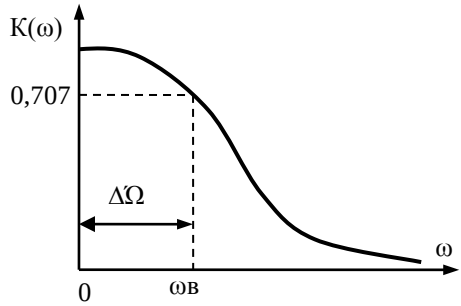


Рис. 1.14. Амплітудно-частотна характеристика *RC*-кола

Коли на вході кола діють імпульси, спектр яких цілком лежить за межами його смуги пропускання, то на виході кола ці імпульси будуть придушені, тобто відбудеться згладжування імпульсної напруги.

1.3.4. Укорочення (загост рення) імпульсів у диференціюючих RC-колах

Основне застосування диференціюючих *RC*-кіл (рис. 1.12, *б)* – полягає в укороченні *(*загостренні) імпульсів для формування короткочасних імпульсів запуску, зриву і синхронізації імпульсних генераторів й інших цілей. Тому диференціюючі *RC*-кола називаються також укорочуваними. Укорочувана дія *RC*-кола пояснюється на рис. 1.10. При виконанні умови $\tau << t_i$ на виході кола в моменти дії фронту і спаду прямокутного вхідного імпульсу виникають два короткі експоненціальні імпульси напруги протилежної полярності з однаковою амплітудою:

$$U_{вих\ 1} = u_{вх} e^{-t/\tau};\ U_{вих\ 2} = -u_{вх} e^{-t/\tau}. \qquad (1.19)$$

Слід зазначити, що стрибки напруги, відповідні фронту і спаду вхідного імпульсу, завжди повністю передаються на вихід кола *($u_{вих} = u_{вх}$).* Це твердження справедливо для будь-якого кола, вхідні і вихідні клеми якого розділені ємністю *C,* і є слідством неможливості виникнення стрибків напруги на самій ємності.

При звичайному способі підрахування t_i на рівні $0{,}5u_{вих}$ дорівнює

$$t_{i\ 0,5} = \tau\ In\ 2 = 0{,}7\tau. \qquad (1.20)$$

1.3.5. Вплив паразитних параметрів у диференціюючих RC-колах

Розглянемо вплив на диференціююче *RC*-коло внутрішнього опору джерела вхідних сигналів (вихідного опору попереднього каскаду) $R_г$, вхідної ємності $C_{вх}$ і резистора $R_{вх}$ наступного каскаду (навантаження). Еквівалентна схема *RC*-кола з урахуванням цих параметрів зображена на рис. 1.15, *а*.

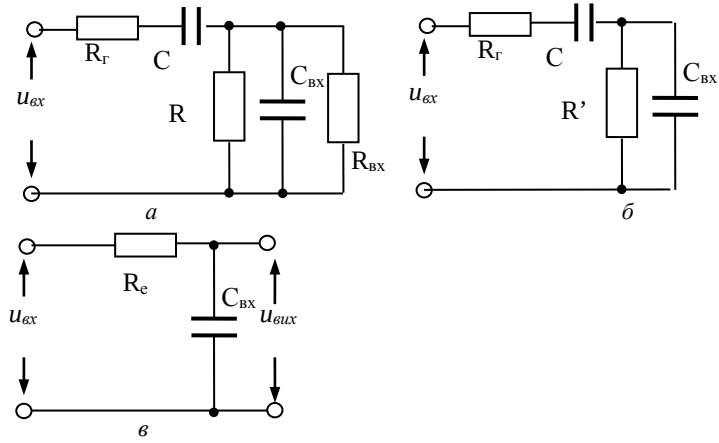


Рис. 1.15. Еквівалентна схема *RC*-кола з урахуванням паразитних параметрів

Ця схема легко приводиться до схеми рис. 1.15, *б,* де

$$R' = RR_{вх} /(R + R_{вх}) < R.$$

Шляхом подальших перетворень ця схема перетворюється на схему рис. 1.15, *в.* Ця схема являє собою інтегруюче *RC*-коло, в якому фронт імпульсу розтягується у зв'язку з процесом зарядки **паразитної ємності** $C_{вх}$ з постійною часу $\tau_e = R_e C_{вх}$.

Реальна форма вихідного імпульсу показана на рис. 1.16.

Таким чином, вплив паразитних параметрів виявляється в зменшенні амплітуди вихідних імпульсів і додатковому розтягуванні їх фронту і спаду. Зазвичай вибирають $C > (4 – 5)\ C_{вх}$. При цьому фронт вихідного імпульсу виходить ще досить коротким у порівнянні з його загальною тривалістю, яка визначається тривалістю спаду.

Проте надмірне збільшення C для зменшення впливу $C_{вх}$ недоцільно, оскільки при заданій тривалості вихідного імпульсу це викликало

б зменшення опору резистора *R.* Але останнє призвело б до зменшення вихідної напруги кола із-за впливу опору $R_г$.

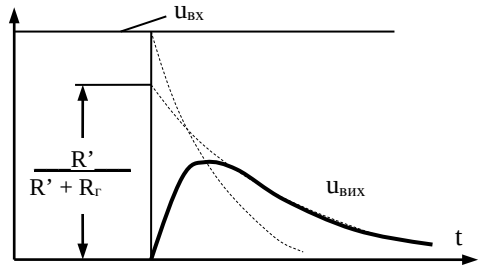


Рис. 1.16. Спотворення імпульсу на виході диференціюючого *RC*-кола через вплив паразитних параметрів

Крім того, при $R < R_г$ тривалість вихідного імпульсу буде залежати від опору $R(R')$.

1.3.6. Дільник напруг. Високочастотна корекція

На рис. 1.17, *а* зображена схема дільника напруги R_1, R_2 з коефіцієнтом передачі $K = u_{вих}/u_{вх} = R_2/(R_1 + R_2)$.

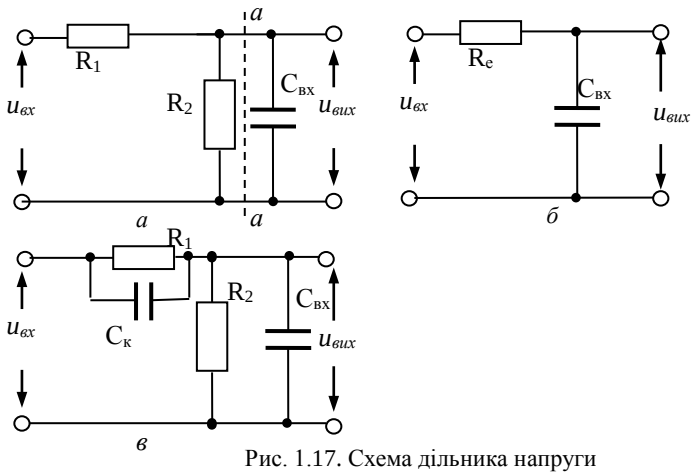


Рис. 1.17. Схема дільника напруги

При передачі через такий дільник імпульсної напруги істотний вплив робить вхідна ємність навантаження (наступного каскаду) $C_{вх}$. Оскільки ця ємність шунтує вихід дільника і її опір зменшується зі зростанням частоти, вона викликає високочастотні спотворення форми імпульсу – розтягування його фронтів. Застосовуючи теорему про еквівалентний генератор щодо точок *аа,* перейдемо до схеми інтегруючого кола на рис. 1.17, *б*, де

$$R_e = R_1R_2/(R_1 + R_2).$$

Таким чином, **форма фронту вихідного імпульсу визначається зарядом ємності $C_{вх}$ з постійною часу $\tau_e = R_eC_{вх}$.**

Для корекції дільника в області високих частот паралельно резистору R_1 включається коректуючий конденсатор $C_к$ (рис. 1.17, *в*). Шунтуючий резистор R_1 на високих частотах, конденсатор $C_к$ додають колу диференціюючих властивостей, збільшуючи для цих частот коефіцієнт передачі дільника (піднімає спад амплітудно-частотної характеристики (АЧХ) в області високих частот).

Ємність конденсатора $C_к$ вибирається на основі наступних міркувань.

На **низьких частотах** можна знехтувати впливом ємностей $C_{вх}$ і $C_к$ ($1/\omega C_{вх} >> R_2$, $1/\omega C_к >> R_1$), а коефіцієнт передачі дільника на низьких частотах $K_н$ буде дорівнювати

$$K_н = u_{вих}/u_{вх} = R_2/(R_1 + R_2).$$

На **високих частотах** можна, навпаки, нехтувати впливом резисторів R_1 і R_2 ($1/\omega C_{вх} << R_2$, $1/\omega C_к << R_1$). При цьому утворюється ємнісний дільник напруги $C_к$, $C_{вх}$ з коефіцієнтом передачі дільника на високих частотах:

$$K_в = u_{вих}/u_{вх} = C_к/(C_к + C_{вх}).$$

Таким чином, умовою неспотвореної передачі імпульсів повинна бути рівність $K_н = K_в$, тобто

$$R_2/(R_1 + R_2) = C_к/(C_к + C_{вх}).$$

Звідки

$$C_{вх}R_2 = C_кR_1. \qquad (1.21)$$

При виконанні цієї умови дільник з коректуючим конденсатором $C_к$ називається таким, що компенсується. Такий дільник передає без спотворень форму імпульсів $u_{вх}$.

Проте **через вплив внутрішнього опору генератора імпульсів $R_г$ спотворюється форма самих імпульсів $u_{вх}$** у порівнянні з формою

імпульсів генератора $u_г$. Спотворені імпульси $u_{вх}$ при виконанні умови (1/20) передаються на вихід дільника вже без додаткових спотворень. Таким чином, спотворення імпульсів на виході дільника є неминучими.

1.3.7. Завдання до самостійних досліджень RC-кіл

1. *Мета*

Закріплення теоретичного матеріалу, набуття навиків створення і моделювання схем аналогових пристроїв, робота з різними вимірювальними приладами.

2. *Теми для попереднього опрацювання*

2.1. Характеристика електричних сигналів. Способи електричного відображення двійкових цифр та чисел.

2.2. Розділові, диференціюючі та інтегровуючі кола

3. *Завдання*

3.1. Дослідження дільника напруг.

3.1.1. Дослідження простого дільника напруг за схемою рис. 1.18, який виконано з використанням двох постійних резисторів R_1 і R_2.

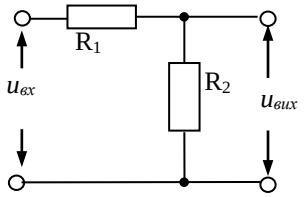


Рис. 1.18. Схема дільника напруг (варіант 1)

Простий дільник напруги – це схема, яка для даної напруги на вході створює на виході напругу, яка є деякою частиною вхідної. У послідовно сполучених резисторах струм визначається таким чином:

$$I = U_{вх}/(R_1 + R_2).$$

Тоді для R_2

$$U_{вих} = IR_2 = U_{вх}R_2/(R_1 + R_2).$$

З останньої формули видно, що $U_{вих} \le U_{вх}$. З цієї формули згідно з варіантом до лабораторної роботи необхідно визначити номінал резистора, який є відсутнім в табл. 1.1.

Зробити заміри U_{R1}, U_{R2} і $I_д$ (де $I_д$ – струм дільника, що протікає через резистори R_1 і R_2).

Завдання 3.1 виконується згідно з варіантом із табл. 1.1.

3.1.2. Зібрати і дослідити дільник напруг за схемою рис. 1.19, що складається з двох постійних резисторів R_1 і R_2 (один із яких визначено

в першому експерименті) та резистора навантаження $R_н$.

Зробити заміри амплітудних значень $U_{вх}$, $U_{вих}$ і I_{R1}, I_{R2} і $I_{Rн}$ (де I_{R1}, I_{R2} і $I_{Rн}$ – струми, що протікають через відповідні резистори).

3.2. Дослідження диференціюючих *RC*-кіл

Для рис. 1.20 виконати дослідження диференціюючого *RC*-кола. Варіанти завдання наведені в табл. 1.2.

Розрахувати постійні часу $\tau = RC$. Одержати і зобразити осцилограми з виходів елементів *R* та *C*.

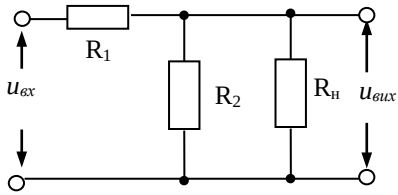


Рис. 1.19. Схема дільника напруг (варіант 2)

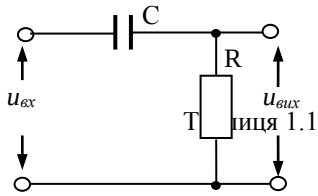


Рис. 1.20. Схема диференціюючого *RC*-кола

Таблиця 1.1

№ з/п	$U_{вх}$, U	$U_{вих}$, U	R_1, Ом	R_2, Ом	$R_н$, Ом	№ з/п	$U_{вх}$, U	$U_{вих}$, U	R_1, Ом	R_2, Ом	$R_н$, Ом
1	1	0,2	100	-	500	9	25	10	200	-	100
2	2	0,3	-	100	$1\ 10^3$	10	30	15	-	200	400
3	5	1	400	-	600	11	35	24	300	-	600
4	10	2	-	400	$1\ 10^3$	12	40	20	-	300	900
5	12	2	$1\ 10^3$	-	$2\ 10^3$	13	45	25	$10\ 10^3$	-	$1\ 10^3$
6	15	5	-	$1\ 10^3$	$1\ 10^3$	14	50	20	-	$10\ 10^3$	$5\ 10^3$
7	17	7	$2\ 10^3$	-	$1\ 10^3$	15	110	20	$20\ 10^3$	-	$10\ 10^3$
8	20	5	-	$2\ 10^3$	$4\ 10^3$	16	220	25	-	$20\ 10^3$	$1\ 10^3$

При виконанні досліджень необхідно пам’ятати величини ємності:

1 Ф = 10^3 мФ ;

1 Ф = 10^6 мкФ;

1 Ф = 10^9 нФ;

1 Ф = 10^{12} пФ.

Таблиця 1.2

№ з/п	Частота f, Гц	Амп. $U_{вх}$	C, F	R, Ом	№ з/п	Частота f, Гц	Амп. $U_{вх}$	C, F	R, Ом
1	$1\ 10^3$	5	$10\ 10^{-9}$	-	9	$0{,}5\ 10^3$	10	-	200
2	$0{,}5\ 10^3$	10	$10\ 10^{-10}$	-	10	$5\ 10^3$	15	-	300
3	$10\ 10^3$	15	$10\ 10^{-10}$	-	11	$10\ 10^3$	20	-	500
4	$20\ 10^3$	20	$10\ 10^{-9}$	-	12	$30\ 10^3$	25	-	$1\ 10^3$
5	$50\ 10^3$	25	$50\ 10^{-8}$	-	13	$50\ 10^3$	30	-	$5\ 10^3$
6	$4\ 10^4$	30	$150\ 10^{-9}$	-	14	$1\ 10^4$	35	-	$10\ 10^3$
7	$5\ 10^4$	40	$75\ 10^{-10}$	-	15	$10\ 10^4$	40	-	750
8	200	10	$10\ 10^{-10}$	-	16	$10\ 10^5$	20	-	150

Наприклад, необхідно розрахувати ***диференціююче*** *RC*-коло при таких даних: $f = 10$кГц, $C = ?$, $R = 500$ Ом. Для цього необхідно:

а) визначити період проходження імпульсів:

$$T = 1/f = 1/10 \times 10^3 = 0{,}1 \times 10^{-3}\ \text{Гц};$$

б) визначити величину, яка є зворотною шпаруватості $Q = T/t_i$. Ця величина потрібна для установки в програмі EWB з позначкою на генераторі сигналів Duty cycle. Обчислювана величина тільки в цій програмі обчислюється в процентах.

$$1/Q = t_i/T = 50\ \%$$

З цього випливає:

$$t_i = (50\%/100\%) \times T = 0{,}5\times\ 0{,}1\times 10^{-3}\ = 0{,}05 \times\ 10^{-3}\, C.$$

При $T = 2 \times t_i$ (що еквівалентно величині $1/Q = 50\%$) такий прямокутний сигнал називається меандром;

в) визначаємо постійну часу досліджуваного *RC*-кола:

$$\tau_i = RC < t_i\,.$$

Зробимо підстановку в цей вираз:

$$500 \times C < 0{,}05 \times 10^{-3}.$$

З останньої формули визначаємо C, яка й була необхідна для дослідження *RC*-кола:

$$C < 0{,}05 \times 10^{-3}/\ 5 \times 10^2 = 0{,}05 \times 10^{-5}.$$

Таким чином, величина ємності С повинна бути набагато меншою, ніж $0{,}05 \times 10^{-5}\ = 500 \times 10^{-9}$.

У нашому прикладі можна вибрати

$$C = 100 \times 10^{-9} = 100 \text{ нФ}.$$

3.3. Дослідження інтегруючих *RC*-кіл

Розрахунок ***інтегруючого*** *RC*-кола виконується аналогічно, за винятком того, що необхідно враховувати умову

$$\tau_i = RC > t_i$$

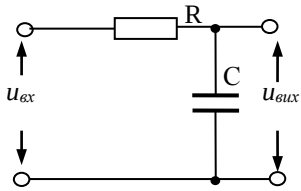


Рис. 1.21. Схема інтегруючого *RC*-кола

Для рис. 1.21 виконати дослідження інтегруючого *RC*-кола. Варіанти завдання наведені в табл. 1.3.

Розрахувати постійні часу $\tau = RC$. Одержати і зобразити осцилограми з виходів елементів *R* та *C* у даних кіл.

Таблиця 1.3

№ з/п	Частота *f*, Гц	Амп. $U_{вх}$	*C*, *F*	R, Ом	№ з/п	Частота *f*, Гц	Амп. $U_{вх}$	*C*, *F*	R, Ом
1	50	2	$200\ 10^{-9}$	-	9	50	40	-	200
2	100	5	$100\ 10^{-9}$	-	10	100	50	-	300
3	200	10	$300\ 10^{-9}$	-	11	200	60	-	400
4	300	20	$300\ 10^{-9}$	-	12	300	70	-	500
5	500	25	$100\ 10^{-9}$	-	13	500	1	-	500
6	$1\ 10^3$	30	$50\ 10^{-9}$	-	14	$1\ 10^3$	5	-	600
7	$10\ 10^3$	35	$10\ 10^{-9}$	-	15	$10\ 10^3$	10	-	700
8	$50\ 10^3$	40	$10\ 10^{-9}$	-	16	$50\ 10^3$	20	-	500

4. *Порядок виконання робот и*

Інструментальні засоби пакета EWB дозволяють проводити побудову електронних схем, дослідження (моделювання) їх роботи як у статичних так і динамічних режимах із застосуванням вимірювальних приладів (таких, як: універсальний генератор сигналів різної форми, осцилограф, вимірник частотних характеристик (плоттер), мультиметр, вольтметр, амперметр), а також джерел напруги і струму, зміни статичних і динамічних параметрів, частотних і часових характеристик досліджуваних схем (таких, як амплітудні значення струмів і напруг у різ-

них точках досліджуваних схем, частота і період проходження вхідних і вихідних сигналів, активна тривалість фронтів і тривалості імпульсів, шпаруватість, амплітудно-частотні характеристики (АЧХ) і фазочастотні характеристики (ФЧХ)).

4.1. Порядок проведення роботи для розробки принципової електричної схеми.

4.1.1. Запустіть Electronics Workbench.

4.1.2. Підготуйте новий файл для роботи. Для цього необхідно виконати такі операції з меню: File/New і File/Save as. При виконанні операції Save as буде необхідно вказати ім'я файлу і каталог, в якому буде зберігатися схема. Рекомендується називати схему на прізвище виконавця.

4.1.3. Перенесіть необхідні елементи з заданої викладачем схеми на робочу область Electronics Workbench. Для цього необхідно вибрати розділ на панелі інструментів (Sources, Basic, Diodes, Transistors, Analog Ics, Mixed Ics, Digital Ics, Logic Gates, Digital, Indicators, Controls, Miscellaneous, Instruments), у якому знаходиться потрібний вам елемент, потім перенести його на робочу область.

4.1.4. З'єднайте контакти елементів і розташуйте елементи в робочій області для одержання необхідної вам схеми. Для з'єднання двох контактів необхідно натиснути на один із контактів кнопкою мишки і, не відпускаючи клавішу, довести курсор до другого контакту. У разі потреби можна використовувати додаткові вузли (розгалуження). Натисканням на елементі правою кнопкою мишки можна одержати швидкий доступ до найпростіших операцій, таких, як обертання (rotate), розворот (flip), копіювання/вирізання (copy/cut), вставка (paste).

4.1.5. Проставте необхідні номінали і властивості кожному елементу схеми. Для цього потрібно двічі натиснути мишкою на зображення елемента. *Перед зміною парамет рів слід відключат и дж ерела ж ивлення схеми, інакше мож на от римат и невірні результ ат и.*

4.1.6. Коли схема зібрана і готова до запуску, натисніть кнопку ввімкнення живлення на панелі інструментів. У разі серйозної помилки в схемі (замикання елемента живлення накоротко, відсутність нульового потенціалу в схемі) буде видане попередження.

4.1.7. Зробіть аналіз схеми, використовуючи інструменти індикації. Виведення термінала здійснюється подвійним натисканням клавіші мишки на зображення елемента. У разі потреби можна скористатися кнопкою Pause.

4.1.8. При необхідності зробіть доступні аналізи в розділі меню Analysis.

4.2. Порядок побудови дільника напруг

4.2.1. Виберіть на панелі інструментів у нижньому рядку піктограму з накресленням елемента живлення. Якщо до неї підвести мишку, то відобразиться надпис Sources. При натисканні на цей елемент піктограма розкриється. Після чого виберіть елемент живлення та перенесіть його на поле програми.

4.2.2. Виберіть на панелі інструмента піктограму із зображення резистора

та аналогічно діям пункту 4.2.1 перенесіть два рази зображення цього елемента на робоче поле.

4.2.3. Виберіть із піктограми Instrument два Multimeter та також перенесіть їх на поле монтажу елементів та з'єднайте їх таким чином, як наведено на рис. 1.22.

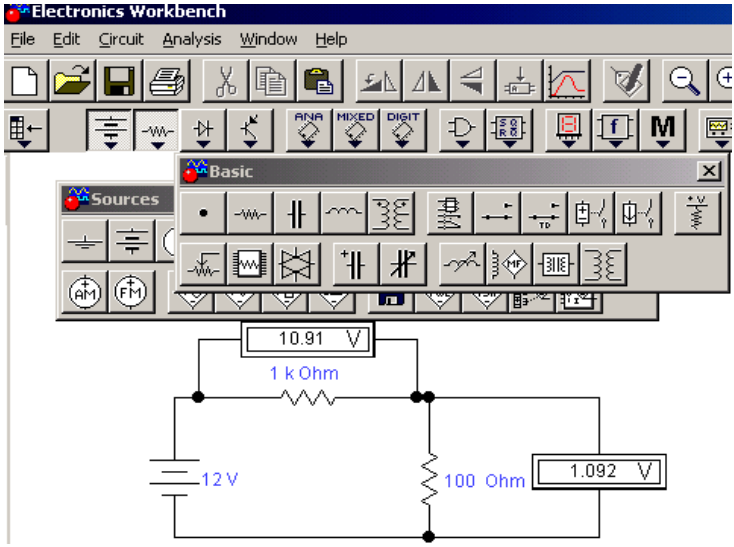


Рис. 1.22. Схема дослідження дільника напруг

4.2.4. Для зображення підписів елементів треба виділити елемент (натиснути ліву клавішу мишки). Виділений елемент повинен стати червоного кольору. Навести стрілку курсора на елемент та натиснути

праву клавішу мишки. Вибрати підменю Component Properties. У ньому вибрати піктограму з назвою Lebel. Ввести назву елемента.

4.3. Порядок побудови *RC*-кіл

4.3.1. При виконанні цього дослідження необхідно вміти користуватися осцилографом та функціональним генератором. Знак функціональних пристроїв вибирається з піктограми, яка розміщується також в останньому рядку віконця програми EWB за кнопкою з зображенням  .

4.3.1.1. Осцилограф (Oscilloscope)

Осцилограф має два канали (CHANNEL) A та B з роздільним регулюванням чутливості в діапазоні від 10 мкВ/діл (mV/Div) до 5 кВ/діл (kV/Div) і регулюванням зсуву по вертикалі (Y POS) (рис. 1.23).

Вибір режиму по входу здійснюється натисненням кнопок  . Режим AC призначений для спостереження сигналів тільки змінного струму (його називають ще режимом “закритого входу”, оскільки в цьому режимі на вході підсилювача включається розділовий конденсатор, який не пропускає постійної складової напруги).

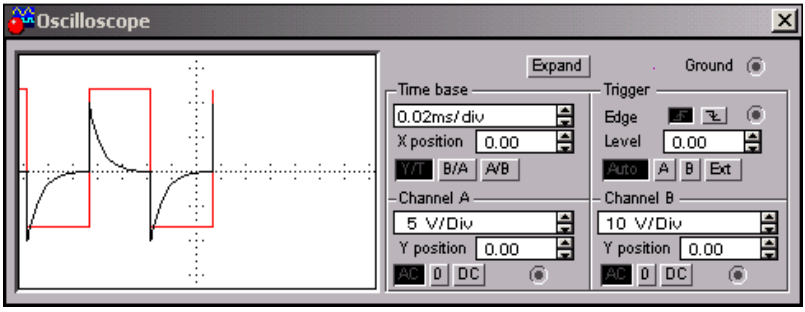
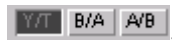


Рис. 1.23. Лицева панель осцилографа

У режимі “0” вхідний сигнал замикається на землю. У режимі DC (включений за замовченням) можна проводити осцилографічні вимірювання як постійного, так і змінного струму. Цей режим ще називають режимом “відкритого входу”, оскільки вхідний сигнал надходить на вхід вертикального підсилювача безпосередньо. З правого боку від кнопки DC розташований вхідний затиск.

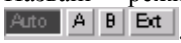
Режим розгортки вибирається кнопками  . У режимі

Y/T (звичайний режим, включений за замовчанням) реалізуються такі режими розгортки: по вертикалі – напруга сигналу; по горизонталі – час;

– у режимі B/ A: по вертикалі – сигнал каналу B; по горизонталі – сигнал каналу A;

– у режимі A/ B: по вертикалі – сигнал каналу A; по горизонталі – сигнал каналу B.

У режимі Y/T тривалість розгортки (TIME BASE) може бути задана в діапазоні від 0,1 нс/діл (ns/div) до 1 с/діл (s/div) з можливістю установки зсуву в тих же одиницях по горизонталі, тобто по осі X (X POS).

У режимі Y/T передбачений також режим очікування (TRIGGER) із запуском розгортки (EDGE) по передньому або задньому фронту сигналу, що запускає (вибирається натисканням кнопок ) при регульованому рівні (LEVEL) запуску, а також в режимі AUTO (від каналу A або B), від каналу A, від каналу B або від зовнішнього джерела (EXT), що підключається до контакту в блоці управління TRIGGER. Названі режими запуску розгортки вибираються кнопками 

.

Заземлення осцилографа здійснюється за допомогою клеми GROUNI, яка розташована в правому верхньому кутку приладу.

Якщо при достатньо великій частоті функціонального генератора не вдається на осцилографі зупинити розгортку, або коли вона дуже мерехтить, необхідно натиснути кнопку ZOOM (рис. 1.24).

При натисканні на кнопку ZOOM лицева панель осцилографа істотно міняється – збільшується розмір екрана, з'являється можливість "*прокрут ки"* зображення по горизонталі і його сканування з допомогою вертикальних візирних ліній (синього і червоного кольору), які за трикутні "вушка" (вони позначені цифрами 1 і 2) можуть бути встановлені курсором у будь-яке місце екрана. При цьому в індикаторних віконцях під екраном наводяться результати вимірювання напруги, тимчасових інтервалів і їх приростів (між візирними лініями).

Зображення можна інвертувати натисканням кнопки REVERSE і записати дані у файл натисканням кнопки SAVE.

Повернення до початкового стану осцилографа проводиться натисканням кнопки REDUCE.

4.3.1.2. Функціональний генератор (Function Generator)

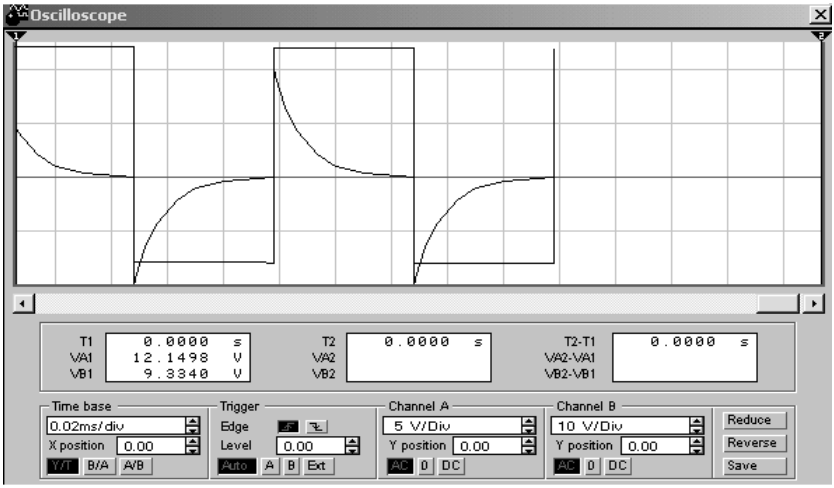


Рис. 1.24. Панель осцилографа в режимі ZOOM

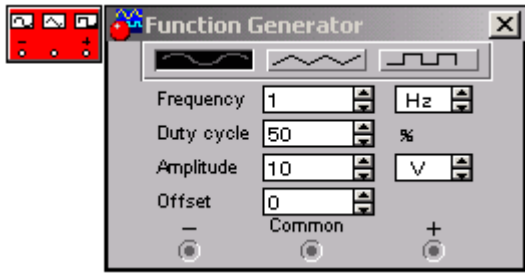


Рис. 1.25. Зовнішній вигляд та лицева панель функціонального генератора

Управління генератором здійснюється за допомогою таких органів управління:

–  – вибір форми вихідного сигналу: синусоїдальної (вибір за умовчанням), трикутної і прямокутної;

– FREQUENCY – установка частоти вихідного сигналу;

– DUTY CYCLE – установка коефіцієнта заповнення в %: для імпульсних сигналів – це відношення тривалості імпульсу до періоду повторення (величина, зворотна шпаруватості), для трикутних сигналів – співвідношення між тривалістю переднього і заднього фронтів;

– AMPLITUDE – установка амплітуди вихідного сигналу;

– OFFSET – установка зсуву (постійної складової) вихідного сигналу;

– 

– вихідні затиски; при заземленні клеми COM (загальний) на клемах "–" і "+" одержуємо парафазний сигнал.

4.3.2. Порядок з'єднання елементів для дослідження *RC*-кіл є аналогічним діям, виконаним у підпункті 4.2.

Один з можливих варіантів з'єднання і дослідження кола наведений на рис. 1.26 та рис. 1.27.

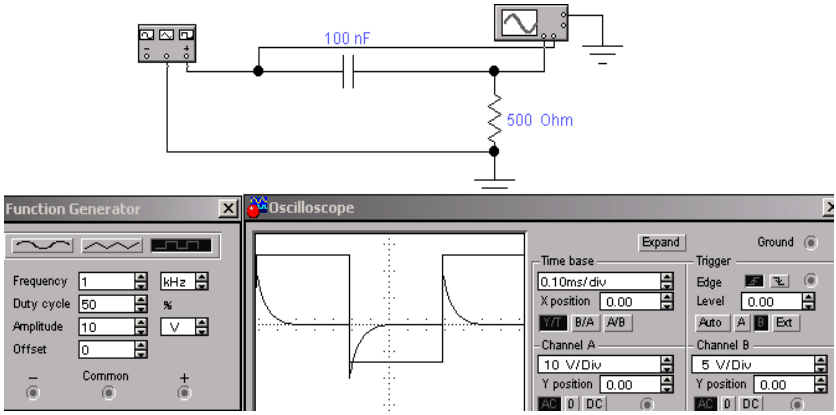


Рис. 1.26. Вікно програми EWB 5.12 зі схемою дослідження диференціюючого *RC*-кола

4.4. Порядок передачі зображення в текстовий редактор WORD:

– увійти в меню Edit та натиснути кнопку Copy;

– натиснути на клавіатурі кнопку Print Screen.

Для запам'ятовування в буфері зображення активної сторінки в WORD необхідно натиснути одночасно дві клавіші Alt та Print Screen.

Для видалення непотрібних частин зображення слід скористатися програмою Paint.

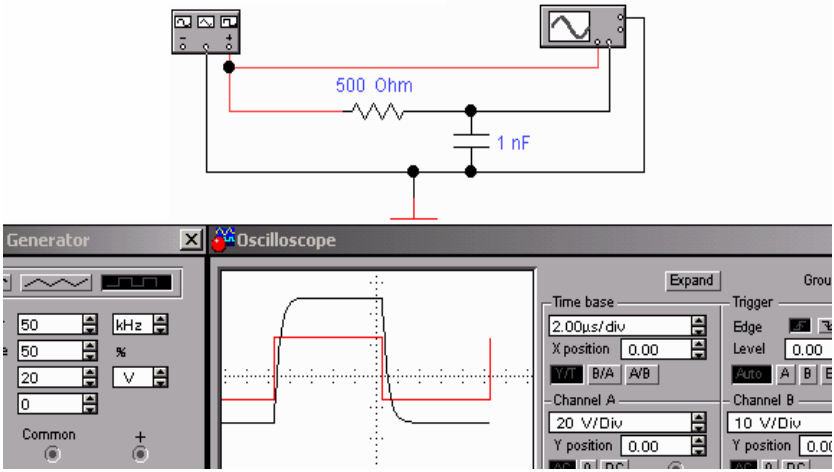


Рис. 1.27. Вікно програми EWB 5.12 зі схемою дослідження інтегруючого RC-кола

5. Питання для самоперевірки

5.1. Як визначається аналітично і геометрично постійна часу кола і який її фізичний сенс?

5.2. При виконанні якої умови на *RC*-колі можливо отримання двох коротких імпульсів?

5.3. Визначте області застосування диференціюючих і інтегруючих кіл?

5.4. За яким законом змінюється величина напруги на конденсаторі в *RC*-колі?

5.5. Який вплив на вихідну напругу здійснюють паразитні параметри кола?

2. ЛОГІЧНІ ОСНОВИ СХЕМОТЕХНІКИ ЕОМ

2.1. Алгебра логіки при аналізі та синтезі логічних функцій

2.1.1. Поняття про цифровий автомат

Автомат – це пристрій, який виконує певний процес без прямої участі людини. Слово “автомат” походить від грецького automatos – самодіючий.

Цифрові автомати (ЦА) – це математична модель реального (технічного) пристрою, який служить для перетворення цифрової інформації.

У загальному випадку ЦА має n входів x_1, x_2,..., x_n і m виходів y_1, y_2,..., y_m (рис. 2.1), кожний з яких може набувати певну кінцеву кількість дискретних значень.

Рис. 2.1. Умовне позначення цифрового автомата

Кількість входів і виходів цифрового автомата обмежена. Входи $x_1,x_2,...,x_n$ утворюють множину входів цифрового автомата:

$$X = \{x_1,x_2,...,x_n\}.$$

Виходи y_1, y_2,..., y_m утворюють множину виходів автомата:

$$Y = \{y_1, y,..., y_m\}.$$

У цифрових автоматах можуть бути елементи, які змінюючи свій стан під дією вхідних сигналів або народжуваних ними керуючих дій, після припинення дії цих сигналів у вихідний стан не повертаються. Вдруге змінити свій стан або повернутись у початковий ці елементи можуть після дії інших вхідних сигналів або повторної дії тих самих вхідних сигналів, або сигналів, породжуваних ними. Такі елементи прийнято називати ***елементами пам’яті*** цифрового автомата. Типовими представниками елементів пам’яті є тригерні пристрої різних типів, які широко

застосовуються на практиці. Наявність у цифровому автоматі елементів пам'яті значно розширює можливості перетворення інформації. При цьому можуть запам'ятовуватись значення вхідних сигналів, які надходять у попередні моменти часу, і використовуватись при формуванні вихідних сигналів.

При розв'язанні задач аналізу й синтезу цифрових автоматів елементи пам'яті прийнято виділяти в окремий блок, який називається блоком пам'яті або просто пам'яттю цифрового автомата (рис. 2.2). Частина схеми автомата, яка не має елементів пам'яті, називається логічним перетворювачем або комбінаційним блоком автомата. ЦА, наведений на рис. 2.2, одержав назву канонічного автомата або автомата з канонічною структурою. Структура може бути декомпозована на дві частини за іншою ознакою: управляючий автомат (ЦА) та операційний автомат (рис. 2.3).

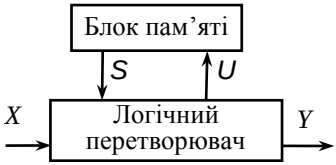


Рис. 2.2. Структурна схема цифрового автомата з пам'яттю

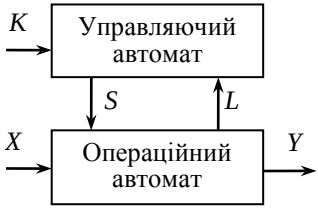


Рис. 2.3. Структурна схема автомата Глушкова

Перший з них згідно з командами K виробляє послідовність мікрокоманд S, які визначають алгоритм переробки інформації операційним автоматом. Ця послідовність коригується залежно від сигналів логічних умов L. ЦА з такою структурою отримали назву автоматів Глушкова.

Віктор Михайлович Глушков (1923-1982) – видатний вчений, засновник всесвітньо відомої наукової школи, директор інституту кібернетики Національної Академії наук України.

У загальному випадку блок пам'яті канонічного ЦА може вміщати *k* елементів пам'яті. Елементи S_1, S_2,..., $S_к$ утворюють множину елементів пам'яті автомата:

$$S = \{ S_1, S_2,..., S_к\}.$$

Кожний елемент пам'яті автомата може мати один або декілька управляючих входів($\ell \geq k$).

Цифрові автомати з двома і більше внутрішніми станами назива-

ють **цифровими автоматами з пам'яттю** або послідовними машинами. Принциповою характерною ознакою цифрових автоматів з пам'яттю є те, що значення вихідних сигналів (набори значень вихідних сигналів) у них визначаються не тільки значеннями вхідних сигналів (вхідними наборами), які надійшли у цей самий момент часу, але й попередніми значеннями, а також порядком їх надходження на входи автомата.

Цифровим автоматом без пам'яті називається автомат, в схемі якого відсутня пам'ять. Вважається, що такий автомат має один внутрішній стан, який повністю визначається складом і схемою з'єднання його елементів. Цифрові автомати цього типу називають також комбінаційними автоматами або логічними перетворювачами. Робота таких автоматів полягає в тому, що вони зіставляють кожний вихідний набір з деякими вхідним набором, значення якого повністю встановлюється значеннями вхідних сигналів, які діють у даний момент часу і не залежать від попередніх значень станів входів. Таким чином, цифровий автомат без пам'яті можна розглядати як окремий випадок цифрового автомата з пам'яттю, у якого кількість внутрішніх станів зведена до одного, а елементи пам'яті відсутні. З іншого боку, цифровий автомат з пам'яттю являє собою сукупність цифрового автомата без пам'яті та елементів пам'яті, об'єднаних у блок пам'яті автомата.

Можлива класифікація за деякими ознаками наведена на рис. 2.4.

Пристрої управління, які часто називаються **контролерами**, є складовою частиною всіх складних цифрових систем і знаходять широке самостійне застосування. Їх функції полягають у формуванні управляючих сигналів для різних об'єктів управління.

Перетворення інформації з аналогової форми в цифрову і навпаки, введення інформації з датчиків, клавіатури, зв'язок між стандартними пристроями і сполучення з ними, обмін по каналах зв'язку – це функції інтерфейсних пристроїв, які називаються також **адаптерами.**

Під функціональною гнучкістю будемо розуміти властивості ЦА змінювати реалізовані функції в процесі настроювання. ЦА, які не мають таких властивостей, називаються пристроями з жорсткою логікою, а ЦА з можливістю одноразового або багаторазового настроювання називаються автоматами (пристроями) з гнучкою логікою.

З метою аналізу і синтезу логічний перетворювач цифрового автомата з пам'яттю зручно подавати у вигляді двох окремих блоків: блока формування вихідних сигналів і блока управління пам'яттю (рис. 2.5, *а*).

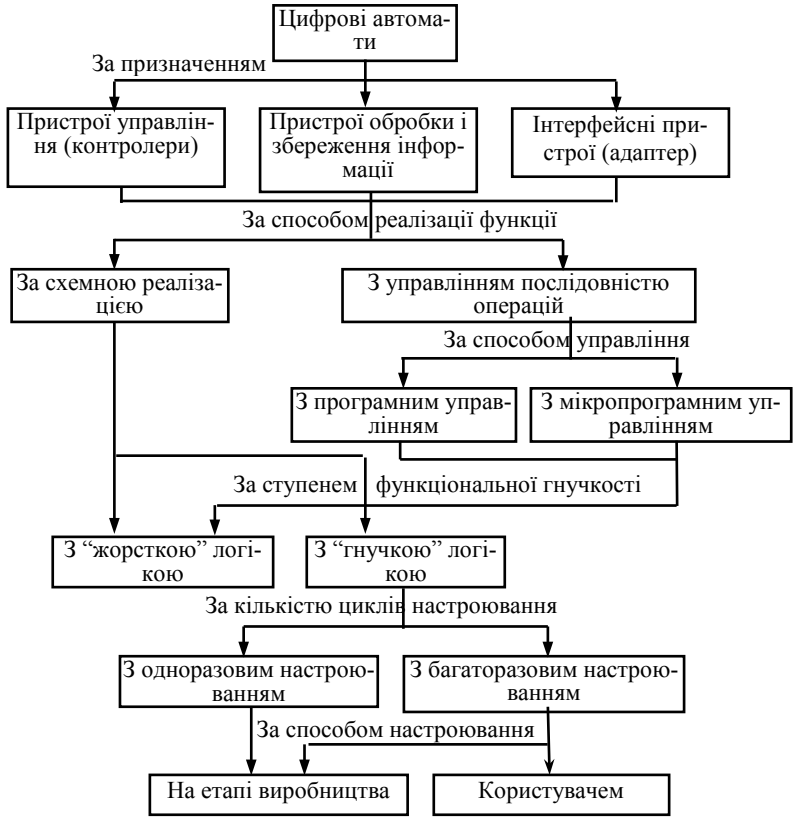


Рис. 2.4. Класифікація цифрових автоматів

Визначимо, що наведений поділ логічного перетворювача на два блоки є умовним і здійснений за функціональною ознакою. В реальних цифрових автоматах, як правило, не вдається виділити окремо блок формування вихідних сигналів і блок управління пам'яттю, оскільки зазначені блоки можуть мати загальні елементи, що спрощує конструкцію автомата в цілому.

Вхідні сигнали цифрового автомата в деяких випадках подаються тільки на входи блока управління (рис. 2.5, *б*).

Характерною властивістю таких автоматів є те, що значення вихі-

дних сигналів повністю визначаються станом пам'яті, а їх тривалість дорівнює часу находження автомата у відповідному стані. Залежно від факторів, що визначають тривалість проміжку часу між двома сусідніми (за часом) станами, цифрові автомати підрозділяються на синхронні та асинхронні.

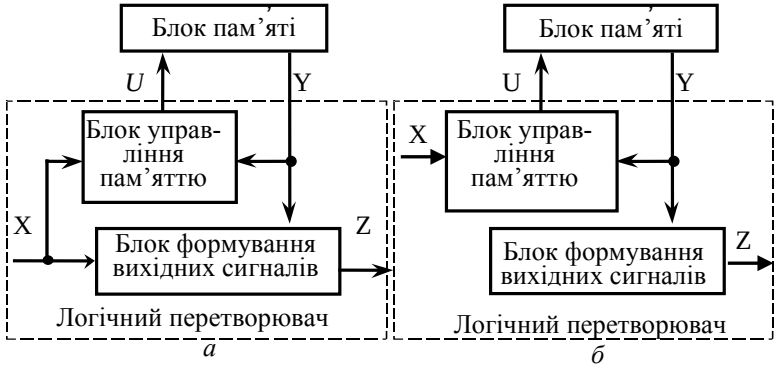


Рис. 2.5. Структурні схеми цифрових автоматів Мілі (*а*) та Мура (*б*)

У синхронному автоматі зміна його станів визначається спеціальним сигналом, який надходить від генератора синхронізуючих імпульсів (ГСІ). Структурна схема синхронного цифрового автомата подана на рис. 2.6.

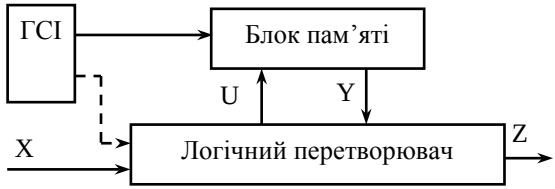


Рис. 2.6. Структурна схема синхронного цифрового автомата

Характерною особливістю таких автоматів є те, що зміни станів автоматів, а в деяких випадках і видача вихідних сигналів, відбуваються в суворо фіксовані моменти часу, які визначаються частотою надходження імпульсів від генератора.

В асинхронних автоматах моменти переходів із одного стану в ін-

ші завчасно не визначені. В таких автоматах немає генератора синхронізуючих імпульсів, і перехід від одного внутрішнього стану в інші здійснюється безпосередньо під впливом вхідних сигналів.

Стан пам'яті автомата в довільний момент часу t + 1 визначається станом входів і станом пам'яті автомата в момент часу t і описується виразом

$$Y(t+1) = \delta[x(t), y(t)].$$

Розглянута функціональна залежність δ називається **функцією переходів автомата.**

Вихідний сигнал автомата в довільний момент часу визначається станом входів і станом пам'яті в цей самий момент часу

$$Z(t) = \lambda[x(t),y(t)], \qquad (2.1)$$

або тільки станом пам'яті

$$Z(t) = \lambda[Y(t)]. \qquad (2.2)$$

Ці функціональні залежності λ називаються **функціями виходів автомата**. Функція виходів (2.1) описує функціонування автомата Мілі, а функція виходів (2.2) – функціонування автомата Мура. Ці автомати набули найбільшого поширення на практиці і названі за іменами вчених, які вперше досліджували їх моделі. Аналіз схем цифрових автоматів показує, що функція переходів описує сумісне функціонування блоків пам'яті і управління пам'яттю, а функція виходів – роботу блока формування вихідних сигналів автомата.

2.1.2. Логічні функції та способи їх завдання

При дослідженні і проектування цифрових автоматів виникають потреби визначення функцій складних перетворювачів за заданими функціями елементарних перетворювачів.

Для вирішення вказаних завдань необхідна наявність математичного апарата, який дозволяли б виражати функції складних змінних через функції простих змінних. Виявилось, що функції елементарних перетворювачів, реалізованих на реле, діодах, транзисторах й інших перемикачах найпростіше описуються за допомогою операцій булевої алгебри.

Назва "булева" привласнено алгебрі на честь англійського математика **Джона Буля** (1815 – 1864 рр.), який зробив великий внесок в її створення.

Спочатку булева алгебра в основному застосовувалася для дослідження зв'язків між логічними висловами. У подальшому сфера застосування булевої алгебри істотно розширилася. Булева алгебра знайшла застосування в теорії множин, теорії вірогідності, в алгоритмах і програмах обчислювального процесу.

Алгебра логіки – одна з складових математичної логіки. Широке використання алгебри логіки як теоретичної основи побудови елементів і вузлів обчислювальної техніки пояснюється тим, що її початкові посилки, які зводяться до подвійного представлення значень використовуваних понять, добре узгоджуються з початками двійково-кодованих систем числення і основними принципами побудови обчислювальної техніки.

Основне поняття алгебри логіки – ***висловлювання***. Висловлювання – деяке припущення, в якому можна стверджувати, що воно істинне або помилкове.

Будь-яке висловлювання можна позначити символом x та вважати, що $x = 1$, коли висловлювання істинне, а при $x = 0$ – коли висловлювання помилкове.

Логічна (булева) змінна – така величина x, яка може набувати тільки два значення $x = \{0, 1\}$. Якщо область значень логічних змінних містить два значення, то такі логічні змінні називають двійковими або булевими. Іншими словами: функція f, яка залежна від змінних x_1, x_2, … , x_n, називається **булевою** або логічною, якщо функція f та будь-який з її аргументів $x_i \in \overline{1,n}$ приймають значення тільки з множини {0,1}.

Логічною функцією називають аналітичний запис залежності між вихідними і вхідними кодовими словами у цифровій схемі, що відображає фізичний процес роботи цієї схеми. Відповідно під ***логічними пристроями*** надалі будемо розуміти пристрої, призначені для формування (реалізації) функцій алгебри логіки.

Функції n змінних, котрі фактично залежать від усіх змінних, називаються **визначеними (невиродженими) функціями** від n змінних.

Функції n змінних, які фактично залежать від меншої кількості змінних, називають **не повністю визначеними (виродженими) функціями.**

Булева функція може задаватись одним з трьох способів: ***матричним (табличним), геометричним і аналітичним.***

При **матричному** способі булева функція $f(x_1, \ldots, x_n)$ задається таблицею істинності (табл. 2.1), в лівій частині якої представлені всі можливі двійкові набори довжини n, а в правій указується значення

функції на цих наборах.

Під двійковим набором $y = \langle y_1, y_2, \ldots, y_n \rangle$, $y \in \{0, 1\}$ розуміється сукупність значень аргументів x_1, x_2, x_n булевої функції f. Двійковий набір має довжину *n,* якщо він представлений *n* цифрами з множини {0,1}. У табл. 2.1 подані всі двійкові набори, які відповідають довжині в чотири розряди.

Таблиця 2.1

X_1	X_2	X_3	X_4	**f**	X_1	X_2	X_3	X_4	**f**
0	0	0	0	**1**	0	1	0	1	**1**
0	0	0	1	**0**	0	1	1	0	**0**
0	0	1	0	**1**	0	1	1	1	**0**
0	0	1	1	**0**	1	0	0	0	**1**
0	1	0	0	**1**					

Зручним способом визначення логічних функцій є використання номерів одиничних, нульових або невизначених наборів. Для цієї мети будь-який набір будемо розглядати як подання цілого невід'ємного числа в двійковій системі числення, яка теж оперує з цифрами "0" і "1". Ціле число у двійковій системі числення можна представити еквівалентним йому числом у десятковій або вісімковій системах числення. Ці числа будемо називати номерами наборів у двійковій, вісімковій або десятковій системах числення, що дозволить записувати в компактній формі набори значень змінних логічних функцій. При цьому будемо вважати, що в двійковому числі молодший розряд розташований праворуч. Наприклад, набір 10111_2 може бути представлений десятковим числом

$$1\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 23_{10},$$

а набір 1011_2 – числом

$$1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 11_{10}.$$

Для представлення набору (двійкового числа) у вісімковій системі числення слід його розділити на **триади**, починаючи з молодшого розряду, і кожну триаду записати цифрою у вісімковій системі числення. Наприклад, набір 100011 може бути представлений вісімковим числом 43_8, оскільки

$$100_2 = 4_{10},\ 011_2 = 3_{10}.$$

При поділі двійкового набору на триади крайня ліва тріада може виявитись неповною. У цьому випадку значення старших розрядів двійкового набору, яких не вистачає, вважаються такими, що дорів-

нюють нулю. Наприклад, двійковому набору 10011110 відповідає вісімкове число 236_8, оскільки 010 – 2, 011 – 3, 110 – 6.

Зручність використання вісімковій системи числення полягає в тому, що при запису двійкового набору у цій системі числення зберігається взаємооднозначна відповідність між розрядами вісімкового числа і триадами двійкового набору (двійкового числа). Це дозволяє легко переходити від завдання двійкового набору значень змінних логічних функцій до вісімкового числа і навпаки.

Для однозначного завдання двійкового набору значень змінних слід зазначати його номер, систему числення, у якій визначений номер, порядок розташування змінних і розмірність, тобто кількість змінних логічних функцій. Так, наприклад, десятковому числу 25 відповідає набір 110001 при розмірності 5 або набір 00110001при розмірності 7.

Умовимось при завданні логічної функції номерами одиничні набори записувати у квадратних дужках, нульові – у круглих дужках, а невизначені – у фігурних дужках. При завданні логічної функції номерами одиничних і нульових наборів першими після квадратної дужки зазначаються номери одиничних наборів, а потім у круглих дужках – номери нульових наборів. За останньою дужкою зазначається основа системи числення.

Запишемо аргументи x_1, x_2, x_n у порядку зростання їх індексів. Наприклад, двійкові набори 0101 і 1000 мають номери 5 і 8 відповідно.

Наприклад, логічна функція z(x) (табл. 2.1) може бути визначена номерами одиничних і нульових наборів у вигляді

$$f(x_4,x_3,x_2,x_1) = [0, 2, 4, 5, 8(1,3,6)]_{10}.$$

Булеві функції, які залежні від великої кількості змінних, задавати таблицею істинності не зручно через її громіздкість. Наприклад, таблиця істинності булевої функції 8 змінних міститиме $2^8 = 256$ рядків. Тому для завдання функцій багатьох змінних зручно використовувати модифікацію таблиці істинності.

При **геометричному** способі булева функція x_1, x_2, x_n задається за допомогою n-мірного куба. У геометричному сенсі кожен двійковий набір $Y = \langle y_1, y_2, \ldots, y_n \rangle$, $y \in \{0, 1\} \in n$-мірним вектором, що визначає точку n-мірного простору. Виходячи з цього, вся безліч наборів, на яких визначена функція n змінних, представляється вершинами n-мірного куба. Відзначаючи точками вершини куба, в яких функція набуває одиничного (або нульового) значення, одержимо геометричне представлення функції. Наприклад, булева функція, геометрично може представляється 3-мірним кубом (рис. 2.7).

При **аналітичному** способі булева функція задається формулами, тобто аналітичними виразами, побудованими на основі операцій булевої алгебри. Аналітичний спосіб завдання булевих функцій займає особливе місце в проектуванні цифрових автоматів. Фактично, всі перетворення над булевими функціями, необхідні для побудови цифрових автоматів, ведуться на аналітичному рівні.

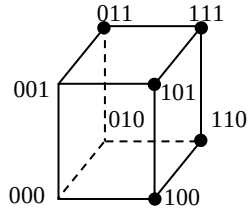


Рис. 2.7. Геометричне представлення логічної функції

2.1.3. Аксіоми алгебри логіки

В алгебрі логіки для того, щоб відрізнитися від звичайної алгебри, операцію логічного додавання позначають знаком $\vee$ та називають диз'юнкцією, а операцію логічного множення позначають знаком $\wedge$ та називають кон'юнкцією.

Для вирішення задач побудови різних логічних функцій необхідно мати систему аксіом та правил перетворення тобто мати алгебру. Використовуючи загальні положення алгебри логіки, не важко переконатися в справедливості таких **аксіом**:

– заперечення $\overline{0} = 1$, $\overline{1} = 0$;

– диз'юнкції:	– кон'юнкції;
$0 \vee 0 = 0$;	$0 \wedge 0 = 0$;
$0 \vee 1 = 1$;	$0 \wedge 1 = 0$;
$1 \vee 0 = 1$;	$1 \wedge 0 = 0$;
$1 \vee 1 = 1$;	$1 \wedge 1 = 1$.

Закони булевої алгебри витікають із аксіом та мають також по дві форми відображення (для диз'юнкції та кон'юнкції):

1. Закон, що переміщує (властивість комутативності):

$$x_1 + x_2 = x_2 + x_1, \quad x_1x_2 = x_2x_1;$$

2. Закон сполучення (властивість комутативності):

$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3,\ x_1(x_2x_3) = (x_1x_2)x_3;$$

3. Закон розподілення (властивість дистрибутивності):

– для кон'юнкції відносно диз'юнкції:

$$x_1 \,\&\, (x_2 + x_3) = (x_1 \,\&\, x_2) + (x_2 \,\&\, x_3);$$

– для диз'юнкції відносно кон'юнкції:

$$x_1 + x_2x_3 = (x_1 + x_2) \,\&\, (x_1 + x_3).$$

4. Закон повторення (тавтології)

$$x \vee x = x;\ x \wedge x = x$$

5. Закон звернення: якщо $x_1 = x_2$, то $\overline{x_1} = \overline{x_2}$.

6. Закон подвійної інверсії: $\overline{\overline{x}} = x$.

7. Закон нульової множини:

$$x \wedge 0 = 0; \quad x \vee 0 = x;$$

8. Закон універсальної множини

$$x \wedge 1 = x; \quad x \vee 1 = 1;$$

9. Закон доповнення:

$$x \wedge \overline{x} = 0; \quad x \vee \overline{x} = 1$$

10. Закон поглинання:

$$x_1 + x_1x_2 = x_1. \qquad [\, x_1 + x_1x_2 = x_1(1 + x_2) = x_1]$$

$$x_1(x_1 + x_2) = x_1.$$

11. Закон склеювання:

$$(x_1 \vee x_2) \wedge (x_1 \vee \overline{x}_2) = x_1; \qquad x_1x_2 \vee x_1\overline{x}_2 = x_1.$$

Закон узагальненого склеювання:

$$\overline{x_1} + x_1x_2 = x_1 + x_2$$

12. Закон де Моргана:

$$\overline{x_1x_2} = \overline{x_1} + \overline{x_2},$$

$$\overline{x_1 + x_2} = \overline{x_1}\,\overline{x_2}.$$

Із законів де Моргана витікають наслідки

$$\overline{\overline{x_1} + \overline{x_2}} = x_1 + x_2,$$

$$\overline{\overline{x_1}\,\overline{x_2}} = x_1x_2,$$

за допомогою яких з'являється можливість виражати кон'юнкцію через диз'юнкцію і заперечення або диз'юнкцію через кон'юнкцію і заперечення.

Закони де Моргана і слідства з них справедливі для будь-якої кількості змінних:

$$\overline{x_1 + x_2 + \ldots x_n} = \overline{x_1}\ \&\ \ldots\ \&\ \overline{x_n},$$

$$\overline{x_1x_2 \ldots x_n} = \overline{x_1} + \overline{x_2} + \ldots + \overline{x_n}.$$

Функція додавання за модулем 2 (mod 2) – функція, що виражається таким чином:

$$x_1 \oplus x_2 = \overline{x_1}x_2 + x_1\overline{x_2} = (x_1 + x_2)(\overline{x_1} + \overline{x_2}).$$

Функція додавання за модулем 2 володіє наступними властивостями:

– комутативності (закон переміщення):

$$x_1 \oplus x_2 = x_2 \oplus x_1;$$

– асоціативності (сполучний закон):

$$x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3;$$

– дистрибутивності (розподільний закон):

$$x_1(x_2 \oplus x_3) = (x_1x_2) \oplus (x_1x_3).$$

Для цієї функції справедливі аксіоми:

$$x \oplus x = 0; \qquad x \oplus 1 = \bar{x};$$

$$x \oplus \bar{x} = 1; \qquad x \oplus 0 = x.$$

На підставі аксіом і властивостей можна вивести правила перетворення функцій І, АБО, НІ через функцію складання за модулем 2 і навпаки:

$$\bar{x}_1 = x_1 \oplus 1;$$

$$x_1 + x_2 = x_1 \oplus x_2 \oplus x_1x_2;$$

$$x_1x_2 = (x_1 \oplus x_2) \oplus (x_1 + x_2)$$

У теорії логічних функцій часто використовують **теорему розкладення**, яку можна навести у вигляді

$$f(x) = f(x_{n-1}, x_{n-2}, \dots, x_k, \dots, x_0) =$$

$$= \bar{x}_k f(x_{n-1}, x_{n-2}, \dots, 0, \dots, x_0) \vee x_k f(x_{n-1}, x_{n-2}, \dots, 1, \dots, x_0).$$

Приклад 2.1.

$$f(x) = x_1 \oplus x_2 \oplus x_1x_2 = \bar{x}_1 (0 \oplus x_2 \oplus 0) \vee x_1(1 \oplus x_2 \oplus x_2) = \bar{x}_1x_2 \vee x_1.$$

Справедливість деяких законів можливо перевірити за допомогою прикладу 2.2.

Приклад 2.2.

$$x_1 + x_2x_3 = (x_1 + x_2) \,\&\, (x_1 + x_3).$$

Насправді $(x_1 + x_2)(x_1 + x_3) = x_1x_1 + x_1x_3 + x_1x_2 + x_2x_3 =$

$$= x_1 + x_1x_2 + x_1x_3 + x_2x_3 = x_1(1 + x_2 + x_3) + x_2x_3.$$

Аналогічно можна довести й інші закони.

Логічні функції

Будь-яку логічну функцію, яка залежить від n змінних ($n > 2$), можна виразити через функції, які залежать від однієї або двох змінних. Розглянемо ці функції.

При $n = 0$ є дві різні функції: $f_0 = 0$ і $f_1 = 1$. Функція $f_0 = 0$ називається константою 0, а функція $f_1 = 1$ називається константою 1.

Елементарні логічні функції, які залежать від однієї змінної, наведені у табл. 2.2.

Таблиця 2.2

f_i	x		Завдання функції формулою	Назва функції
	0	1		
f_0	0	0	$f_0(x) = 0$	Константа 0
f_1	1	0	$f_1(x) = \overline{x}$	Інверсія
f_2	0	1	$f_2(x) = x$	Повторення
f_3	1	1	$f_3(x) = 1$	Константа 1

Зазначимо, що для кожної функції однієї змінної існує інверсна її функція:

$$f_0(x) = \overline{f}_3(x); \quad f_3(x) = \overline{f}_0(x);$$
$$f_1(x) = f_2(x); \quad \overline{f}_1(x) = f_2(x).$$

Логічні функції двох змінних наведені в таблиці 2.3.

Розглянемо логічні функції, які найбільш часто використовуються.

Функція кон'юнкції (логічне перемноження): $F(x_1, x_2) = x_1 \,\&\, x_2$. Логічний елемент 2І, який реалізує цю функцію, та таблиця його функціонування (таблиця істинності) наведені на рис. 2.8.

Функція набуває значення 1, якщо обидва аргументи дорівнюють 1.

Функція диз'юнкції (логічне додавання): $F(x_1, x2) = x_1 \vee x_2$. Логічний елемент 2АБО, який реалізує цю функцію, та таблиця його функціонування (таблиця істинності) наведені на рис. 2.9.

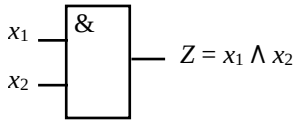
Функція диз'юнкції набуває одиничного значення, якщо хоча б один із аргументів дорівнює 1.

Функція додавання за модулем 2: $F(x_1, x_2) = x_1 \oplus x_2$. Логічний елемент М2, який реалізує цю функцію, та таблиця його функціонування (таблиця істинності) наведені на рис. 2.10.

Функція Пірса: $F(x_1,x_2) = x_1 \downarrow x_2$. Логічний елемент 2АБО-НІ, який реалізує цю функцію та таблиця його функціонування (таблиця істинності), наведено на рис. 2.11.

Таблиця 2.3

	Набір				Завдання функції формулою	Назва функції
x_1	0	1	0	1		
x_2	0	0	1	1		
$f_0[x]$	0	0	0	0	$f_0[x] = 0$	Константа 0
$f_1[x]$	1	0	0	0	$f_1[x] = x_1 \downarrow x_2$	**Функція Пірса (АБО-НІ)**
$f_2[x]$	0	1	0	0	$f_2[x] = x_1 \leftarrow x_2$	Заборона x_2
$f_3[x]$	1	1	0	0	$f_3[x] = \overline{x}_2$	**Заперечення** x_2
$f_4[x]$	0	0	1	0	$f_4[x] = x_2 \leftarrow x_1$	Заборона x_1
$f_5[x]$	1	0	1	0	$f_5[x] = \overline{x}_1$	Інверсія x_1
$f_6[x]$	0	1	1	0	$f_6[x] = x_1 \oplus x_2$	**Додавання за модулем 2**
$f_7[x]$	1	1	1	0	$f_7[x] = x_1 \mid x_2$	**Функція Шеффера (І-НІ)**
$f_8[x]$	0	0	0	1	$f_8[x] = x_1 \wedge x_2$	**Кон'юнкція (І)**
$f_9[x]$	1	0	0	1	$f_9[x] = x_1 \sim x_2$	Еквівалентність
$f_{10}[x]$	0	1	0	1	$f_{10}[x] = x_1$	Повторення x_1
$f_{11}[x]$	1	1	0	1	$f_{11}[x] = x_2 \rightarrow x_1$	Імплікація x_2 в x_1
$f_{12}[x]$	0	0	1	1	$f_{12}[x] = x_2$	Повторення x_2
$f_{13}[x]$	1	0	1	1	$f_{13}[x] = x_1 \rightarrow x_2$	Імплікація x_1 в x_2
$f_{14}[x]$	0	1	1	1	$f_{14}[x] = x_1 \vee x_2$	**Диз'юнкція (АБО)**
$f_{15}[x]$	1	1	1	1	$f_{15}[x] = 1$	**Константа 1**

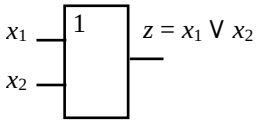


а

x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

б

Рис. 2.8. Умовне графічне позначення логічного елемента 2І (*а*) та таблиця істинності (*б*)

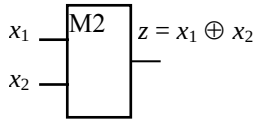


а

x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

б

Рис. 2.9. Умовне графічне позначення логічного елемента 2АБО (*а*) та таблиця істинності (*б*)

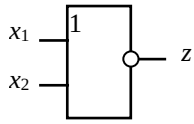


а

x_1	x_2	f(x_1,x_2)
0	0	0
0	1	1
1	0	1
1	1	0

б

Рис. 2.10. Умовне графічне позначення логічного елемента М2 (*а*) та таблиця істинності (*б*)



а

x_1	x_2	f(x_1,x_2)
0	0	1
0	1	0
1	0	0
1	1	0

б

Рис. 2.11. Умовне графічне позначення логічного елемента 2АБО-НІ (*а*) та таблиця істинності (*б*)

Функція імплікації x_1 в x_2: $F(x_1,x_2) = x_2 \rightarrow x_1$.

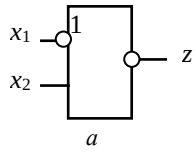
Імплікація двох змінних x_1 та x_2 являє собою складне висловлювання, яке є помилковим при істинному x_2, або при помилковому x_1.

Логічний елемент "НІ x_1 АБО x_2 – НІ", який реалізує цю функцію та таблиця його функціонування (таблиця істинності) наведені на рис. 2.12.

Функція штрих **Шеффера** (логічне перемноження з інверсією або кон'юнкція з інверсією): $F(\overline{x_1,x_2}) = x_1,x_2$. Логічний елемент 2І-НІ, який реалізує цю функцію, та таблиця його функціонування (таблиця істинності) наведені на рис. 2.13.

Функція набуває значення "0", якщо обидва аргументи дорівнюють "1".

Функція еквівалентності (рівнозначності). У результаті виконання операції рівності утвориться функція, значення якої дорівнює 1, коли значення аргументів збігаються, і дорівнює 0 – у протилежному випадку.

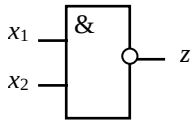


а

x_1	x_2	f(x_1,x_2)
0	0	1
0	1	1
1	0	0
1	1	1

б

Рис. 2.12. Умовне графічне позначення логічного елемента $\overline{x_1}$ АБО x_2-НІ (*а*) та таблиця істинності (*б*)



а

x_1	x_2	f(x_1,x_2)
0	0	1
0	1	1
1	0	1
1	1	0

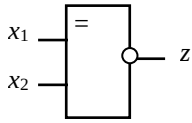
б

Рис. 2.13. Умовне графічне позначення логічного елемента 2І-НІ (*а*) та таблиця істинності (*б*)

Очевидно, що

$$F(x1, x2) = x_1 \leftrightarrow x_2 = \overline{x_1 \oplus x_2} = x_1 x_2 + \overline{x_1}\,\overline{x_2}$$

Логічний елемент еквівалентності, який реалізує цю функцію, та таблиця його функціонування (таблиця істинності) наведені на рис. 2.14.



а

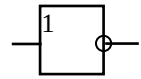
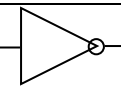
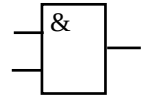
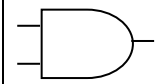
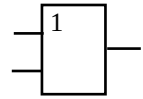
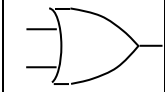
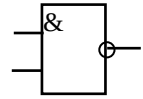
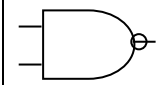
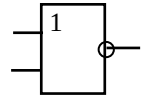
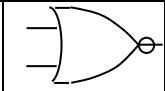
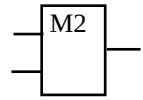
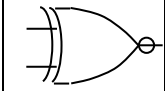
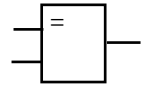
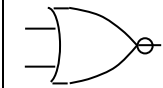
x_1	x_2	f(x_1,x_2)
0	0	1
0	1	0
1	0	0
1	1	1

б

Рис. 2.14. Умовне графічне позначення логічного елемента еквівалентності (*а*) та таблиця істинності (*б*)

У табл. 2.4 наведена деяка характеристика логічних елементів.

Таблиця 2.4

№ з/п	Логічна функція	Назва елемента	Умовне графічне позначення		Маркування
1	НІ, заперечення, інверсія	НІ, інвертор			ЛН
2	І, логічне множення, кон'юнкція	І, кон'юнктор			ЛИ
3	АБО, логічне додавання, диз'юнкція	АБО, диз'юнктор			ЛЛ
4	І -НІ, «штрих Шеффера»	І-НІ, елемент Шеффера			ЛА
5	АБО-НІ, «Стрілка Пірса»	АБО-НІ, елемент Пірса			ЛЕ
6	Функція нерівності, виключне АБО, додавання за модулем 2	Виключне АБО, суматор за модулем 2, елемент нерівності			ЛП
7	Функція рівності (еквівалентності)	Елемент рівності (еквівалентності)			У складі більшості серій відсутній

Принцип суперпозиції та функціональна повнота системи логічних функцій

Принципом суперпозиції називають принцип, при якому є можливість отримувати складні функції шляхом зміни індексів змінних і підстановки інших функцій замість змінних вхідної функції. Можливість такої підстановки обумовлюється тим, що області значень їх змінних збігаються.

Наприклад, маючи елементарні функції

$$f_1(x) = x_1x_2,$$

$$f_2(x) = x_3 \oplus x_4,$$

можна, користуючись принципом суперпозиції, одержати такі нові функції:

$$f_3(x) = x_1 (x_3 \oplus x_4),$$

$$f_4(x) = (x_1x_2) \oplus x_4.$$

Функція $f_3(x)$ одержана шляхом підстановки функції $f_2(x)$ у функцію $f_1(x)$. Функція $f_4(x)$ одержана з функції $f_2(x)$ шляхом підстановки логічної функції $f_1(x)$ замість змінної x_3.

Система логічних функцій $f_1(x)$, $f_2(x)$,..., $f_n(x)$ називається **функціонально повною**, якщо будь-яка функція представляється суперпозицією цих функцій, взятих у будь-якій кінцевій кількості екземплярів.

Функціонально повна система логічних функцій $f_1(x)$, $f_2(x)$,..., $f_S(x)$ називається **мінімальною**, якщо вилучення з неї хоча б однієї функції перетворює систему в неповну.

Аналіз функцій двох змінних показує, що існує велика кількість різних функціонально повних систем елементарних логічних функцій. Перелічимо деякі з них:

– функція Шеффера (І-НІ);
– функція Пірса (АБО-НІ);
– функції І та НІ;
– функції АБО та НІ;
– функція імплікації і константи 0 і т.п.

Наведені функціонально повні системи є мінімальними. Додавання до мінімальних функціонально нових систем інших логічних функцій дозволяє отримати сукупності логічних функцій, які мають властивість повноти, але є не мінімальними за кількістю функцій, що входять до них. На практиці з немінімальних функціонально повних систем логічних функцій широкого застосування набула система, яка складається з функцій І, АБО, НІ.

Кожна функція функціонально повної системи реалізується певним типом логічного елемента. Сукупність логічних елементів, за допомогою яких здійснюється технічна реалізація функціонально повної системи логічних функцій, називається **функціонально повною системою логічних елементів або базисом.**

При практичній побудові цифрових пристроїв системи логічних елементів, які реалізують мінімальні функціонально повні системи,

часто виявляються менш зручними, ніж системи елементів, що мають більшу кількість різних елементів і реалізують немінімальні функціонально повні системи логічних функцій. Наприклад, до такої системи належать сукупності логічних елементів І, АБО, НІ чи логічних елементів І за mod 2, генератор одиниці та ін. Це пояснюється тим, що такі сукупності елементів дозволяють знаходити більш оптимальні за складністю структури проектованих цифрових пристроїв.

З іншого боку, збільшення кількості різних логічних функцій, що реалізуються елементами, призводить до збільшення вартості систем логічних елементів, які випускаються. Звідси виходить, що існує оптимальна кількість різних елементів, які входять в систему, котра дозволяє отримувати цифрові пристрої мінімальної вартості. Фізична реалізація схем логічних елементів може бути різноманітною. Логічні елементи можуть бути побудовані на базі електронних ламп, напівпровідникових приладів, електромеханічних пристроїв, феритів, тощо. Найбільше поширення на даний час мають логічні елементи, виконані на інтегральних мікросхемах.

Диз'юнктивна і кон'юнктивна форми подання логічних функцій

Одним з основних способів завдання логічних функцій є їх подання у вигляді аналітичних виразів (формул). Перевагою такого способу завдання є можливість проведення еквівалентних перетворень логічних функцій. У введеній алгебрі основними аналітичними формами подання логічних функцій є диз'юнктивна і кон'юнктивна форми.

Розглянемо диз'юнктивні форми подання логічних функцій. Спочатку введемо поняття елементарної кон'юнкції.

Елементарною кон'юнкцією називається логічний добуток змінних та їх заперечення.

Кількість змінних, які складають конкретну елементарну операцію, називають її **рангом.**

Елементарна кон'юнкція, яка є функцією усіх аргументів заданого набору, називають **конституентою одиниці.**

Приклад 2.3. Нехай $r = 4$.

У функції $f(x_1x_2x_3x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_2x_3 \vee x_1\bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4$ логічні набори $x_1x_2x_3$ та $x_1\bar{x}_1x_2x_3x_4$ не є конституентами одиниці.

Елементарною диз'юнкцією називається логічний додаток змінних із інверсією або без неї, причому кожна змінна записується тільки один раз.

Елементарна диз'юнкція, яка є функцією усіх аргументів заданого набору, називають **конституентою нуля**.

Приклад 2.4. Нехай задана конституента нуля:

$$Ki(0) = \bar{x}_1 + x_2 + \bar{x}_3 + x_4, \ i = 10.$$

Коли в конституенті нуля змінні без інверсії замінити нулями, а змінні з інверсією замінити одиницями, то отримаємо двійкове число, яке вказує на номер набору.

Кон'юнктивною нормальною формою (КНФ) називається функція, яка складається з елементарних диз'юнкцій. Іншими словами, КНФ – це кон'юнкція будь-якої кінцевої множини попарно різних елементарних диз'юнкцій.

Наприклад, задана функція п'яти змінних:

$$f(x_1x_2x_3x_4x_5) = (x_1 \vee \bar{x}_2 \vee x_5) \wedge (\bar{x}_1 \vee x_3 \vee x_4) \wedge x_3.$$

У розглянутому прикладі, який наведено в КНФ, ранг елементарних операцій не співпадає з рангом функції.

Наприклад, логічні функції

$$f_1(x_1,x_2) = (x_1 \vee \bar{x}_2)(\bar{x}_1 \vee x_2),$$

$$f_2(x_1,x_2,x_3) = x_1(x_1 \vee x_2)(\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$

являють собою кон'юнкції елементарних диз'юнкцій. Отже, вони записані в кон'юнктивній нормальній формі.

Довільна логічна функція, задана аналітичним виразом, може бути приведеною до КНФ шляхом виконання таких операцій:

– використання правила інверсії, якщо операція заперечення застосована до логічного виразу;

– використання аксіоми дистрибутивності відносно множення:

$$f_i(x) \vee f_j(x) f_k(x) = [f_i(x) \vee f_j(x)][f_j(x) \vee f_k(x)];$$

– використання операції поглинання:

$$f_i(x) \vee f_i(x) f_j(x) = f_i(x);$$

– виключення в диз'юнкціях змінних, що повторюються, або їх заперечень;

– вилучення всіх однакових елементарних диз'юнкцій, до яких одночасно входять змінні та їх заперечення.

Диз'юнктивною нормальною формою (ДНФ) логічної функції називається диз'юнкція будь-якої кінцевої множини попарно різних елементарних кон'юнкцій. Наприклад, логічні функції

$$f_1(x_1,x_2) = x_1 \bar{x}_2 \vee \bar{x}_1 x_2,$$

$$f_2(x_1,x_2,x_3) = x_3 \vee \bar{x}_1 \bar{x}_2 \vee x_1 x_2 x_3$$

являють собою диз'юнкції елементарних кон'юнкцій. Отже, вони записані в диз'юнктивній нормальній формі.

Довільна логічна функція, задана аналітичним виразом, може бути приведеною до ДНФ шляхом:

– використання правила інверсії, якщо операція заперечення застосована до логічного виразу;

– розкриття дужок;

– виключення в кон'юнкціях змінних, що повторюються, або їх заперечень;

– виключення всіх однакових елементарних кон'юнкцій, крім одної;

– виключення всіх кон'юнкцій, у яких одночасно є змінна та її заперечення.

Приклад 2.5. Перетворимо до диз'юнктивної нормальної форми логічну функцію

$$f(x) = \bar{x}_1 (x_2 \vee x_1 x_3) \vee \overline{\bar{x}_2 \vee x_1 (\bar{x}_2 \vee x_3)}.$$

Застосуємо правило інверсії. Тоді

$$f(x) = \bar{x}_1 (x_2 \vee x_1 \bar{x}_3) \vee x_2 (\bar{x}_1 \vee x_2 \bar{x}_3).$$

Розкриємо в отриманому виразі дужки:

$$f(x) = \bar{x}_1 x_2 \vee \bar{x}_1 x_1 \bar{x}_3 \vee x_2 \bar{x}_1 \vee x_2 \bar{x}_2 \bar{x}_3$$

і вилучимо кон'юнкції, що повторюються, і які мають як змінну, так і її заперечення, а також однакові змінні, що повторюються і входять до кон'юнкції. У результаті виконання зазначених операцій одержуємо ДНФ

$$f(x) = \bar{x}_1 x_2 \vee x_2 \bar{x}_3.$$

Диз'юнктивна нормальна форма називається *вдосконаленою* (ВДНФ), якщо кожна елементарна кон'юнкція, що входить до неї, містить у прямому або інверсійному вигляді всі змінні, від яких залежить функція. Іншими словами, коли ранг функції збігається з усіма рангами елементарних операцій, то така функція називається ***вдосконаленою***.

Перетворення ДНФ у вдосконалену ДНФ здійснюється шляхом виконання таких операцій:

– множення кожної елементарної кон'юнкції на диз'юнкції змінних та їх заперечень, якщо вони не входять до даної елементарної

кон'юнкції;

– розкриття дужок;

– вилучення всіх однакових елементарних кон'юнкцій, крім одної.

У ВДНФ конституенти одиниці мають найвищий ранг. Для ВДНФ логічна функція може бути записана у вигляді цифрового додатка: Σ_0.

Приклад 2.6. Перетворимо до ВДНФ логічну функцію:

$$f(x) = \bar{x}_1 x_2 \vee x_2 \bar{x}_3.$$

Оскільки функція, яка розглядається, залежить від трьох змінних x_1, x_2, x_3, першу кон'юнкцію помножимо на вираз $x_3 \vee \bar{x}_3$, а другу – на $x_1 \vee \bar{x}_1$:

$$f(x) = \bar{x}_1 x_2(x_3 \vee \bar{x}_3) \vee (x_1 \vee \bar{x}_1) x_1 x_2 \bar{x}_3.$$

Розкривши дужки і вилучивши кон'юнкції, які повторюються, отримаємо шукану вдосконалену диз'юнктивну нормальну форму функції:

$$f(x) = \bar{x}_1 x_2 x_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee x_1 x_2 \bar{x}_3.$$

Характерною властивістю ВДНФ є те, що представлення в ній логічної функції єдині. Елементарні кон'юнкції, які входять у ВДНФ функції, носять назву конституент одиниці. Оскільки константа одиниці містить у прямому або інверсійному вигляді всі змінні, від яких залежить функція, то вона перетворюється на одиницю на єдиному наборі значень змінних. Звідси виходить, що кожна ВДНФ містить стільки конституент одиниці, скільки одиничних наборів має логічна функція. Так, функція, розглянута в прикладі, задана на трьох одиничних наборах, отже, її ВДНФ має три конституенти одиниці.

Логічна функція константи одиниці у ВДНФ представляється диз'юнкцією 2^n конституент одиниці.

У практичних завданнях при первинному описі логічні функції найчастіше задаються таблицями відповідності, у яких кожний одиничний набір зіставляється з конституентою одиниці. При цьому до конституенти одиниці входить змінна, якщо її значення в наборі дорівнює одиниці, та інверсія змінної, якщо її значення в наборі дорівнює нулю. Логічна функція має стільки конституент одиниці, скільки робочих наборів задано в таблиці істинності (відповідності). Звідси випливає правило визначення досконалої ДНФ функції за таблицею істинності.

Для кожного рядка таблиці, в якій функція дорівнює одиниці, складається елементарна кон'юнкція всіх змінних. При цьому до кон'юнкції входить сама змінна, якщо її значення дорівнює нулю. Оде-

ржані елементарні кон'юнкції об'єднуються знаком диз'юнкції**.**

Приклад 2.7. Логічна функція трьох змінних задана табл. 2.5. Необхідно записати аналітичний вираз цієї функції у ВДНФ.

Для запису функції у ВДНФ необхідно вибрати рядки таблиці, у якій $f(x_1x_2x_3)$ дорівнюють одиниці, тобто конституенти одиниці.

Функція у ВДНФ записується таким чином:

$$f(x_1,x_2,x_3,x_4) = K_1(1) + K_3(1) + K_6(1) + K_7(1) =$$
$$= \Sigma_0(1, 3, 6, 7) =$$
$$= \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2\bar{x}_3 \vee x_1x_2x_3.$$

Таблиця 2.5

x_1	x_1	x_3	$f(x_1x_2x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Кон'юнктивна нормальна форма називається вдосконаленою, якщо кожна елементарна диз'юнкція, що входить до неї, має в прямому або інверсійному вигляді всі змінні, від яких залежить функція.

Перетворення КНФ в удосконалену КНФ здійснюється шляхом виконання таких операцій:

– додавання до кожної елементарної диз'юнкції кон'юнкцій змінних та їх заперечень, якщо вони не входять у дану елементарну диз'юнкцію;

– використання аксіоми дистрибутивності;

– вилучення всіх однакових елементарних диз'юнкцій, крім однієї.

Характерною властивістю вдосконаленої КНФ є те, що представлення в ній логічної функції єдині.

Елементарні диз'юнкції, які входять у вдосконалену КНФ функції, носять назву конституент нуля. Кожна конституента нуля, що входить у вдосконалену КНФ, набуває нульового значення на єдиному наборі значень змінних, який є нульовим набором функції. Отже, кількість нульових наборів логічної функції збігається з кількістю конституент нуля, які входять в її вдосконалену КНФ.

Для кожного рядка таблиці істинності, у якій функція дорівнює нулю, складається елементарна диз'юнкція всіх змінних. При цьому в диз'юнкцію входить сама змінна, якщо її значення дорівнює нулю, або заперечення, якщо його значення дорівнює одиниці. Одержані елементарні диз'юнкції об'єднуються знаком кон'юнкції.

Приклад 2.8. Для логічної функції z(x), заданої табл. 2.5 відповідності, визначимо вдосконалену кон'юнктивну форму.

Для визначення ВКНФ необхідно для конституент нуля записува-

ти номери наборів 0, 2, 4, 5, використовуючи між змінними диз'юнкцію:

$$f(x_1x_2x_3) = K_0(0)\ K_2(0)\ K_4(0)\ K_5(0) =$$
$$= (x_1 \vee x_2 \vee x_3)(x_1 \vee \bar{x}_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee \bar{x}_3).$$

Слід зазначити, що для функцій, кількість одиничних наборів яких перевищує кількість нульових наборів, більш компактним є запис у вигляді СКНФ і навпаки.

2.1.4. Розв'язання задач щодо мінімізації логічних функцій мет одом безпосередніх перет ворень

***Приклад* 2.9.** Представити логічну функцію $f(x_1,x_2,x_3,x_4) = \overline{x_1x_2x_3x_4}$ у ДНФ.

Розв'язання. Послідовно виконуючи теореми де Моргана перетворимо функцію в ДНФ:

$$f(x_1, x_2, x_3, x_4) = \overline{x_1x_2x_3x_4} = \bar{x}_1 + \overline{x_2x_3x_4} = \bar{x}_1 + \bar{x}_2\overline{x_3x_4} = \bar{x}_1 + \bar{x}_2(\bar{x}_3 + \bar{x}_4) = \bar{x}_1 + \bar{x}_2\bar{x}_3 + x_2x_4.$$

***Приклад* 2.10.** Спростити логічну функцію:

$$F(x_1,x_2,x_3,x_4) = \overline{\overline{x_1\bar{x}_2} + \bar{x}_1x_3}\ \overline{\bar{x}_1x_2 + \overline{\bar{x}_2\bar{x}_3}} =$$

$$= \overline{\overline{x_1\bar{x}_2} + \overline{\bar{x}_1x_3}}\ + \overline{\overline{\bar{x}_1x_2 + \overline{\bar{x}_2\bar{x}_3}}} = x_1\bar{x}_2 + \bar{x}_1x_3 + \bar{x}_1x_2 + \overline{\bar{x}_2\bar{x}_3} =$$

$$= \underline{x_1\bar{x}_2} + \underline{\bar{x}_1x_3} + \bar{x}_1x_2 + (\underline{\bar{x}_2} + \underline{\bar{\bar{x}}_3}) = \bar{x}_2(x_1 + 1) + x_3(\bar{x}_1 + 1) + \bar{x}_1x_2 =$$

$$= \bar{x}_2 + x_3 + \bar{x}_1x_2.$$

***Приклад* 2.11.** Представити логічну функцію в ДНФ.

$$F(x_1,x_2,x_3,x_4) = x_1\bar{x}_2x_3(x_1x_4 + x_3\bar{x}_4) + \overline{(x_1 + \bar{x}_2)(x_1 + \bar{x}_4)} =$$
$$= x_1\bar{x}_2x_3x_4 + x_1\bar{x}_2x_3\bar{x}_4 + \overline{x_1 + x_1\bar{x}_4 + x_1\bar{x}_2 + \bar{x}_2\bar{x}_4} =$$

$$= x_1\bar{x}_2x_3x_4 + x_1\bar{x}_2x_3\bar{x}_4 + \overline{x_1 + \bar{x}_2\bar{x}_4} = x_1\bar{x}_2x_3 + \overline{\bar{x}_1\bar{\bar{x}}_2\bar{x}_4} = x_1\bar{x}_2x_3 + \bar{x}_1(x_2 + x_4) =$$
$$= x_1\bar{x}_2x_3 + \bar{x}_1x_4 + \bar{x}_1x_4.$$

***Приклад* 2.12.** Логічну функцію, яка подана таблицею істинності виразити у ВДНФ та ВКНФ, мінімізувати їх методом безпосередніх перетворень. Побудувати функціональні схеми, які реалізують ці функції у загальному базисі та базисах І-НІ та АБО-НІ.

x_1	x_2	x_3	f(x_1,x_2,x_3)	x_1	x_2	x_3	f(x_1,x_2,x_3)
0	0	0	0	1	0	0	1
0	0	1	1	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	0

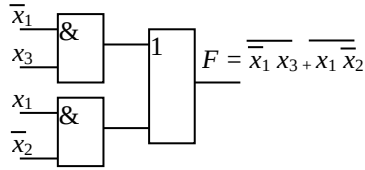
Представимо спочатку функцію у ВДНФ, тобто визначимо її одиничні значення за допомогою конституент одиниць.

$$F(x_1,x_2,x_3,) = \Sigma_0(1, 3, 4, 5) = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 .$$

Мінімізуємо цю функцію методом безпосередніх перетворень:

$$F(x_1,x_2,x_3,) = \overline{x}_1x_3(\overline{x}_2 + x_2) + x_1\overline{x}_2(x_3 + x_3) = \overline{x}_1 x_3 + x_1 \overline{x}_2.$$

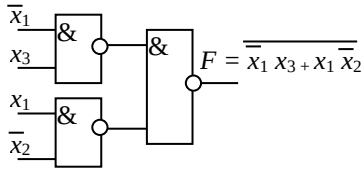
Далі побудуємо функціональну схему, яка реалізує цю функцію в основному базисі.



Щоб реалізувати цю ж функцію в базисі І-НІ, необхідно ДНФ функції надати до такої форми, яка б не вміщала операції логічного додатка. Для цього необхідно від функції взяти подвійне заперечення (що не змінює його значення) та за теоремою де Моргана виконати перетворення:

$$F(x_1,x_2,x_3,) = \overline{\overline{\overline{x}_1 x_3 + x_1 \overline{x}_2}}. = \overline{\overline{\overline{x}_1 x_3} \cdot \overline{x_1 \overline{x}_2}}.$$

Цей логічний вираз вміщує тільки операції І-НІ. Побудуємо його функціональну схему.



Представимо цю ж функцію у ВКНФ. Це означає, що потрібно визначити її за нульовими наборами за допомогою конституент нуля:

$$F(x_1,x_2,x_3,) = \Pi_0(0, 2, 6, 7) =$$
$$= (x_1 + x_2 + x_3)(x_1 + \overline{x}_2 + x_3)(\overline{x}_1 + \overline{x}_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3).$$

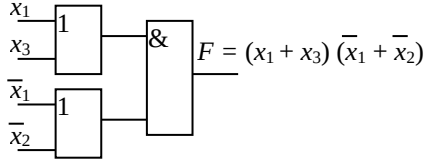
Для мінімізації отриманої функції використаємо закон склеювання:

$$(x_1 + x_2 + x_3)(x_1 + \overline{x}_2 + x_3) = (x_1 + x_3);$$
$$(\overline{x}_1 + \overline{x}_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) = (\overline{x}_1 + \overline{x}_2).$$

Остаточно маємо

$$F(x_1,x_2,x_3,) = (x_1 + x_3)(\overline{x}_1 + \overline{x}_2).$$

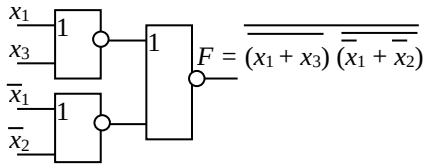
Побудуємо функціональну схему, яка реалізує цю функцію в основному базисі.



Для побудови цієї функції у базисі АБО-НІ необхідно в КНФ позбавитись операції кон'юнкції. Ця операція відбувається при використанні подвійного заперечення від функції за допомогою теореми де Моргана:

$$F(x_1,x_2,x_3,) = \overline{\overline{(x_1 + x_3)(\overline{x}_1 + \overline{x}_2)}} = \overline{\overline{x_1 + x_3} + \overline{\overline{x}_1 + \overline{x}_2}}.$$

Будуємо функціональну схему в базисі АБО-НІ:



***Приклад* 2.13.** Навести у мінімальній ДНФ та мінімальній КНФ логічну функцію трьох змінних, які подані у таблиці істинності.

x_1	x_2	x_3	f(x_1,x_2,x_3)	x_1	x_2	x_3	f(x_1,x_2,x_3)
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	0

Запишемо функцію у ВДНФ та мінімізуємо її:

$$F(x_1,x_2,x_3,) = x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 = x_1\overline{x}_2.$$

Представимо цю ж функцію у ВКНФ і також мінімізуємо її:

$$F(x_1,x_2,x_3,) = = (x_1 + x_2 + x_3)(x_1 + x_2 + \overline{x}_3)\,(x_1 + \overline{x}_2 + x_3)\,(x_1 + \overline{x}_2 + \overline{x}_3) +$$
$$+ (x_1 + \overline{x}_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) = (x_1 + x_2)(x_1 + \overline{x}_2)(\overline{x}_1 + \overline{x}_2) = x_1\overline{x}_2.$$

Наведені приклади дозволяють ще раз переконатися, що функції у ДНФ та КНФ дозволяють отримати одні й ті ж результати.

***Приклад* 2.14.** Мінімізувати логічну функцію та привести до базису АБО-НІ. Побудувати функціональну схему на двовходових елементах.

$$Y = \overline{x}_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2x_3 = \overline{x}_1x_2x_3 + x_1\overline{x}_2x_3 + x_1x_2x_3 + x_1x_2x_3 =$$

$$= \overline{\overline{x_2x_3 + x_1x_3}} = \overline{\overline{x_2x_3}\ \overline{x_1x_3}} = \overline{(\overline{x}_2 + \overline{x}_3)(\overline{x}_1 + \overline{x}_3)} = \overline{\overline{x}_2 + \overline{x}_3} + \overline{\overline{x}_1 + \overline{x}_3}.$$

Функціональна схема, яка реалізує функцію прикладу 2.14, наведена на рис. 2.15.

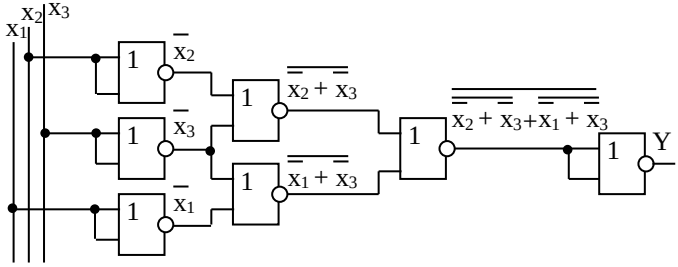


Рис. 2.15. Функціональна схема до прикладу 2.14

***Приклад* 2.15.** Мінімізувати логічну функцію та привести до базису І-НІ. Побудувати функціональну схему на двовходових елементах.

$$Y = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2x_3 + \overline{x}_1\overline{x}_2x_3 = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2x_3 + \overline{x}_1\overline{x}_2x_3 =$$

$$= \overline{x}_1\overline{x}_2 + \overline{x}_1x_2x_3 = \overline{x}_1(\overline{x}_2 + x_2x_3) = \overline{x}_1(\overline{x}_2 + x_3) = \overline{x}_1\overline{x}_2 + \overline{x}_1x_3 = \overline{\overline{x}_1\overline{x}_2} \quad \overline{\overline{x}_1x_3}.$$

Функціональна схема до прикладу 2.15 наведена на рис. 2.16.

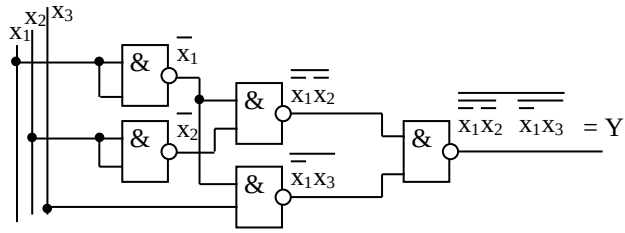


Рис. 2.16. Функціональна схема до прикладу 2.15

***Приклад* 2.16.** Мінімізувати логічну функцію та привести до базису І-НІ.

$$Y = \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1x_2\overline{x}_3 = x_2\overline{x}_3 + \overline{x}_1x_2x_3 = x_2(\overline{x}_3 + \overline{x}_1x_3) =$$

$$= x_2(\overline{x}_3 + \overline{x}_1) = x_2\overline{x}_3 + x_2\overline{x}_1 = \overline{\overline{x_2\overline{x}_3}\ \overline{x_2\overline{x}_1}}.$$

***Приклад* 2.17.** Мінімізувати логічну функцію та привести до базису АБО-НІ.

У прикладі використаємо теорему розкладення.

$$Y = x_2x_4 + x_1\overline{x}_3\overline{x}_4 + x_2\overline{x}_3x_4 + x_1x_3 = x_2x_4 + x_1\overline{x}_3\overline{x}_4 + x_1x_3 =$$

$$= x_4(x_2 + x_1\overline{x}_3\overline{1}) + \overline{x}_4(x_2\,0 + x_1\overline{x}_3\,\overline{0}) + x_1x_3 = x_4x_2 + \overline{x}_4x_1\overline{x}_3 + x_1x_3 =$$

$$= x_4x_2 + x_1(\overline{x}_4\overline{x}_3 + x_3) = x_4x_2 + x_1(x_3 + \overline{x}_4) = x_4x_2 + x_1x_3 + x_1\overline{x}_4 =$$

$$= x_4(x_2 + x_1\,\overline{1}) + \overline{x}_4(x_2\,0 + x_1\,\overline{0}) + x_1x_3 = x_4x_2 + \overline{x}_4x_1 + x_1x_3 =$$

$$= \overline{\overline{x}_4 + \overline{x}_2} + \overline{\overline{x}_4 + \overline{x}_1} + \overline{\overline{x}_1 + \overline{x}_3}.$$

***Приклад* 2.18.** Мінімізувати логічну функцію та привести до базису АБО-НІ.

$$Y = x_1x_3x_4 + x_2x_3x_4 + x_1\overline{x_3x_4} + \overline{x}_2x_4 = x_1x_3x_4 + x_4(x_2x_3 + \overline{x}_2) + x_1\overline{x_3x_4} =$$

$$= x_1x_3x_4 + x_4(x_3 + \overline{x}_2) + x_1\overline{x_3x_4} = x_1x_3x_4 + x_3x_4 + \overline{x}_2x_4 + x_1\overline{x_3x_4} =$$

$$= x_3x_4 + \overline{x}_2x_4 + x_1\overline{x_3x_4} = \overline{x}_2x_4 + x_3x_4x_1 = \overline{\overline{x}_2x_4 \;\; x_3x_4x_1} = \overline{\overline{x}_2 + \overline{x}_4} + \overline{\overline{x}_3 + \overline{x}_4 + \overline{x}_1}.$$

2.1.5. Завдання до самостійних досліджень простих логічних пристроїв

Завдання 2.1

2.1. Запустіть Electronics Workbench.

2.2. Підготуйте новий файл для роботи.

2.3. Зібрати задану схему в програмі моделювання EWB згідно з варіантами на рис. 2.17 – 2.32.

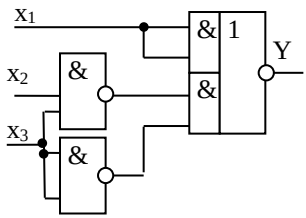


Рис. 2.17. Варіант 1

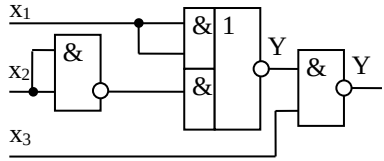


Рис. 2.18. Варіант 2

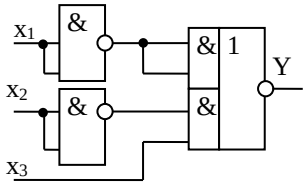


Рис. 2.19. Варіант 3

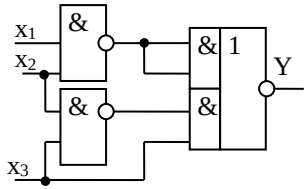


Рис. 2.20. Варіант 4

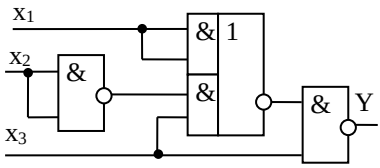


Рис. 2.21. Варіант 5

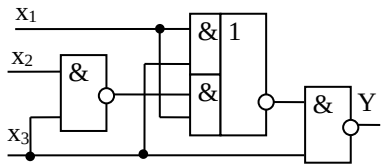


Рис. 2.22. Варіант 6

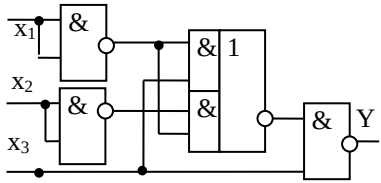


Рис. 2.23. Варіант 7

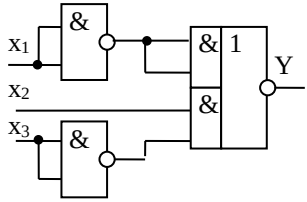


Рис. 2.24. Варіант 8

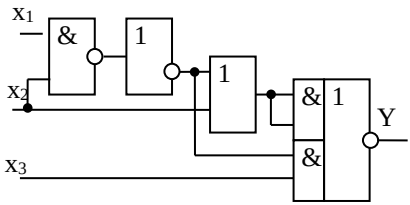


Рис. 2.25. Варіант 9

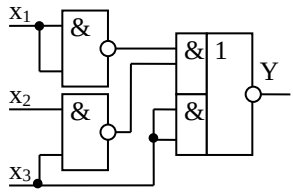


Рис. 2.26. Варіант 10

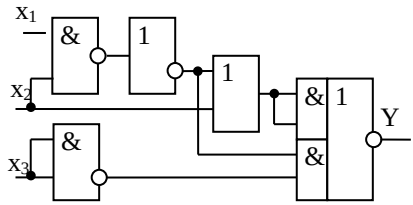


Рис. 2.27. Варіант 11

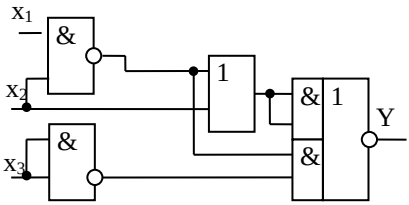


Рис. 2.28. Варіант 12

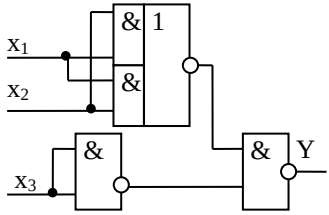


Рис. 2.29. Варіант 13

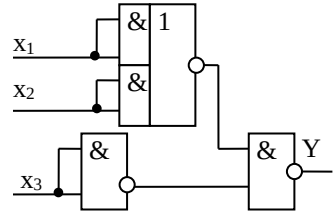


Рис. 2.30. Варіант 14

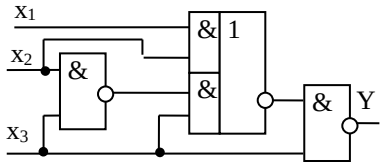


Рис. 2.31. Варіант 15

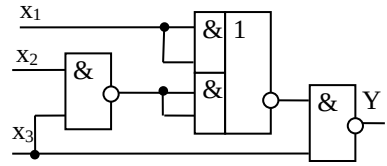


Рис. 2.32. Варіант 16

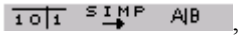
2.4. Аналітично отримати значення логічної функції.

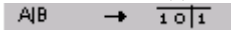
2.5. Під'єднати до входів та виходів схеми логічний конвертор та отримати таблицю істинності. Для цього необхідно входи схеми підключити до відповідних входів ЛП, а вихід зв'язати з правою клемою (out). Отримана в результаті перетворення таблиця істинності може бути конвертована в будь-яку іншу форму представлення при використанні кнопок на лицевій панелі ЛП.

Схему із приєднаним логічним конвертором наведено на рис. 2.33.

2.6. Одержати таблицю істинності за допомогою кнопки логічного конвертора з позначенням у вигляді  , яка перетворює побудовану схему у таблицю істинності.

2.7. Одержати логічний вираз за допомогою кнопки логічного конвертора з позначенням у вигляді 

2.8. Провести автоматичне мінімізування отриманої схеми за допомогою кнопки логічного конвертора з позначенням у вигляді  , якщо отримана таблиця не є мінімальною.

2.9. За допомогою кнопки логічного конвертора у вигляді  одержати таблицю істинності. Дані таблиці істинності будуть новими при успішній мінімізації схеми.

2.10. За допомогою кнопки логічного конвертора у вигляді  отримати із логічного виразу нову схему.

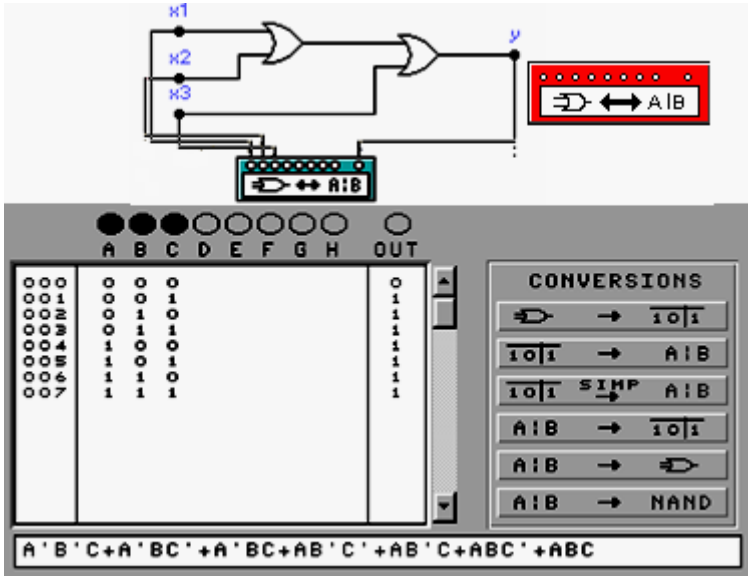


Рис. 2.33. Логічний пристрій із логічним конвертором

2.11. Аналітично перетворити значення логічної функції у базис АБО-НІ та перевірити отриманий вираз за допомогою кнопки логічного конвертора у вигляді A|B → NAND із побудованою схемою в базисі АБО-НІ.

2.12. Під'єднаємо до досліджуваної схеми генератор слів з позначенням у вигляді 0000 XXXX. Схема логічного пристрою із під'єднаними генератором слів та логічним аналізатором наведена на рис. 2.34.

У генератор слів у 16-річній системі числення необхідно ввести двійкові набори в кількості, значення яких відповідають кількості входів побудованої схеми.

За допомогою логічного аналізатора отримаємо часові діаграми функціонування схеми, які повинні відповідати таблиці істинності.

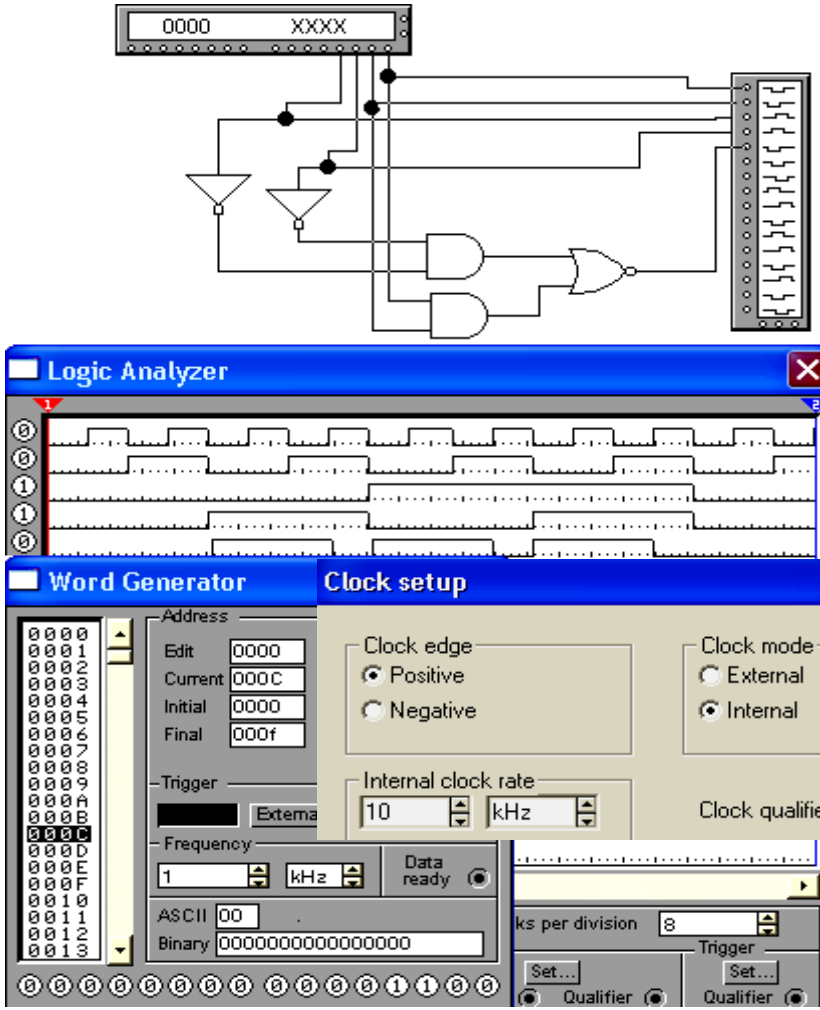


Рис. 2.34. Схема логічного пристрою із під'єднаними генератором слів та логічним аналізатором

Завдання 2.2

2.13. Побудувати відповідно до заданого логічного виразу комбінаційну схему згідно з варіантами 1 – 10):

$Y = (\overline{X1}\,\wedge X2 \wedge X3) \vee (X1 \wedge \overline{X2} \wedge X3) \vee (X1 \wedge X2 \wedge \overline{X3}) \vee (\overline{X1} \wedge X2 \wedge \overline{X3});$ (1)

$Y = \overline{X1 \vee X2} \vee \overline{X3};$ (2)

$Y = \overline{X1 \wedge X2 \wedge X3} \wedge \overline{X4};$ (3)

$Y = (X1 \vee X2) \vee (X3 \vee X4);$ (4)

$Y = (\overline{X1 \wedge X2}) \vee (\overline{X3 \wedge X4});$ (5)

$Y = (\overline{X1} \wedge \overline{X2}) \vee (\overline{X1} \wedge X2) \vee (X1 \wedge \overline{X2}) \vee (X1 \wedge X2);$ (6)

$Y = (X1 \vee X2) \wedge (\overline{X3 \vee X4});$ (7)

$Y = (\overline{X1 \vee X2}) \wedge (X3 \vee X4);$ (8)

$Y = (\overline{X1 \vee X2}) \wedge \overline{(X3 \vee X4)};$ (9)

$Y = (\overline{X1} \wedge X2) \vee (\overline{X3} \wedge X4).$ (10)

2.14. Увести заданий логічний вираз у логічний конвертор і перетворити його в схему. Для цього необхідно в діалоговому вікні, яке розміщено у нижній частині лицьової панелі логічного конвертора, задати логічну функцію. В логічний конвертор можливо вводити тільки такі змінні, які наведені на передній панелі конвертора у розширеному режимі: A, B. C, D, E, F, G, H. Тому потрібно зробити відповідні переназначення змінних: x1, x2, x3, x4 і.д. Після цього доцільно користуватися трьома нижніми кнопками в правій частині лицевої панелі (можна, наприклад, перетворити вихідний логічний вираз і побудувати схему в базисі "I-HI").

2.15. За допомогою останніх кнопок логічного конвертора одержати всі дані таблиці істинності, схеми та вирази і перевірити їх з отриманими аналітичним чином.

2.16. На підставі таблиці істинності за допомогою генератора слів та логічного конвертора протестувати отриману схему і зняти її часову діаграму.

2.17. Спробувати спростити вихідний логічний вираз (вибравши

третю кнопку зверху).

2.18. Перетворити спрощений логічний вираз у схему (вибравши останню кнопку знизу).

2.1.6. Завдання до самостійної підготовки

1. Що таке логічна змінна? Що таке логічна функція? Які існують способи подання логічних функцій?

2 . Назвіть основні аксіоми та закони алгебри логіки.

3. Сформулюйте та поясніть принцип двоїстості.

4. Що називають функціонально повною системою логічних функцій або логічним базисом? Наведіть приклади логічних базисів.

5. Наведіть і поясніть класифікацію логічних пристроїв.

6. Що таке кон'юнктивна та диз'юнктивна нормальні форми подання логічних функцій? Коли КНФ та ДНФ вважаються досконалими?

7. Записати логічні функції, що реалізують схеми логічного пристрою на рис. 2.35 та мінімізувати їх методом безпосередніх перетворень:

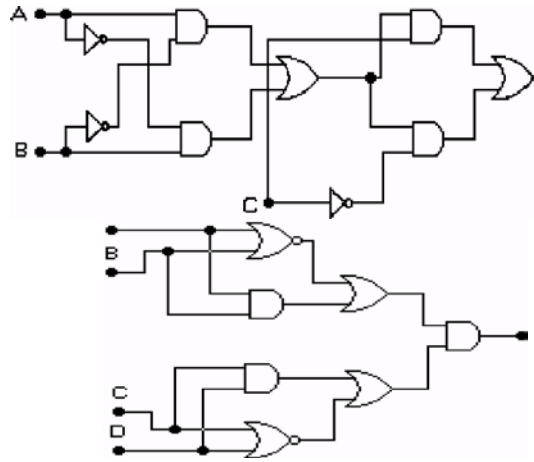


Рис. 2.35. Схеми логічних пристроїв

8. Спростити та перетворити логічну функцію в базис І-НІ, зобразити схему, яка цю функцію реалізує:

$$F = \overline{C}\overline{B}\overline{A} + CBA + \overline{B}\overline{A}.$$

9. Спростити та перетворити логічну функцію в базис АБО-НІ, зобразити схему, яка цю функцію реалізує:

$$F = \overline{D}\overline{B}\overline{A} + \overline{D}CB + \overline{D}\overline{C}BA.$$

10. Навести функцію $f(x_1,x_2,x_3,x_4) = x_1\,x_3 + x_2 + \overline{x_1}x_2\,x_3$ у ВДНФ.

11. Логічну функцію, яка наведена у вигляді

$$F(x_1,x_2,x_3,) = \Sigma_0(0, 1, 3, 5, 7),$$

мінімізувати та представити в базисах (основному, І-НІ, АБО-НІ).

2.2. Основи синтезу логічних пристроїв

У процесі проектування будь-якого цифрового пристрою доводиться виконувати послідовність дій, що можуть бути віднесені до задач аналізу та синтезу. Виконання задач аналізу логічних пристроїв передбачає наявність готової логічної схеми, побудованої на логічних елементах заданого типу, і зводиться до запису його логічної функції в аналітичному вигляді або до побудови таблиці істинності.

Синтез логічного пристрою передбачає побудову його логічної схеми, тобто визначення складу необхідних логічних елементів та сполучень між ними, при яких вхідні цифрові сигнали будуть перетворюватись на вихідні відповідно до заданого алгоритму роботи пристрою.

У процесі синтезу необхідно виконувати мінімізацію апаратних витрат на реалізацію пристрою. Ця мінімізація безпосередньо пов'язана з мінімізацією логічної функції, яка описує алгоритм роботи пристрою.

Алгоритм синтезу комбінаційного цифрового пристрою містить такі ***основні етапи:***

1. Запис умов функціонування цифрового пристрою (ці умови частіше за все задають у таблиці істинності або логічною функцію пристрою, який необхідно синтезувати. Для запису умов функціонування цифрових пристроїв можливе також використання й словесного опису).

2. Запис та мінімізація логічної функції (якщо на першому етапі функція вже була задана в аналітичному вигляді, то виконується лише її мінімізація).

3. Запис мінімізованої логічної функції у заданому базисі.

4. Зображення отриманої структурної схеми, тобто зображення потрібних логічних елементів і зв'язків між ними.

З перелічених етапів найбільш складним і трудомістким є другий – запис та мінімізація логічної функції.

2.2.1. Мінімізація логічних функцій за допомогою карт Карно

ВДНФ та ВКНФ логічних функцій є, як правило, незручними для безпосередньої технічної реалізації, оскільки приводять до надмірно громіздких і складних схем. Тому початкові вирази ВДНФ ті ВКНФ потрібно перетворювати в більш прості. Такими формами можуть бути мінімальні ДНФ та КНФ. Вибір способу мінімізації залежить від кількості аргументів логічної функції. Так, наприклад, застосування карт Карно доцільно для функцій, кількість змінних яких не перевищує шести. При більшій кількості змінних доцільно використовувати алгоритмічні методи мінімізації, які достатньо просто реалізовуються на комп'ютерах.

Карти Карно – це табличний спосіб представлення логічних функцій. Кожен елемент такої таблиці відповідає певному набору вхідних змінних і визначає значення функції на цьому наборі вхідних параметрів. Двійковий номер елемента утворюється з номерів рядка і стовпця, в яких він розташований. Якщо логічна функція є функцією 4-х змінних x_1,x_2,x_3,x_4, то номер рядка визначається значенням x_1,x_2 перших двох змінних, а номер стовпця – значенням двох останніх.

При побудові карти Карно логічних функцій трьох і більше аргументів необхідно здійснити розташування їх аргументів так, щоб кожен елемент карти був сусіднім з елементами, двійкові номери яких відрізнялися б від двійкового номера цього елемента значенням тільки одного розряду. Для цієї побудови можливо скористатися властивістю чисел коду Грея, у якого будь-які дві сусідні комбінації відрізняються значенням тільки в одному розряді.

Розглянемо правило формування коду Грея із двійкового коду.

Нехай задано n-розрядне двійкове число

$$X = x_n, \ldots, x_p, \ldots, x_1 ,$$

де x_p – значення розрядів числа

x_1 – молодший розряд.

Порядок переходу з двійкового числа до коду Грея:

$$A(x) = (\lambda_n , \ldots , \lambda_p, \ldots, \lambda_1),$$

де:

$$\lambda_p = \begin{cases} x_p \oplus x_{p+1} , \text{ якщо } p = 1, 2, \ldots, n-1, \\ x_n, \text{ якщо } p = n . \end{cases}$$

Зворотне перетворення коду Грея у двійковий код виконується за таким виразом:

$$x_p = \begin{cases} \lambda_p \oplus x_{p+1}, \text{ якщо } p = 1, 2, \ldots, n-1, \\ \lambda_n, \text{ якщо } p = n \end{cases}$$

Розглянемо схему прямого та зворотного перетворення коду Грея на конкретному прикладі (рис. 2.36).

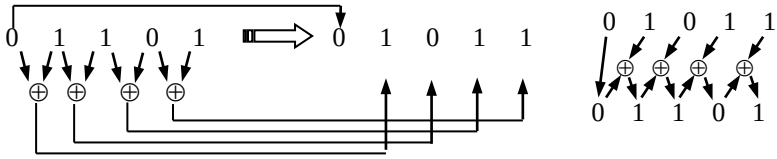


Рис. 2.36. Схема прямого та зворотного перетворення коду Грея

У табл. 2.6 наведені дані відповідності 3-розрядного двійкового коду – коду Грея.

Таблиця 2.6

Двійковий код			Код Грея		
x_3	x_2	x_1	λ_3	λ_2	λ_1
0	0	1	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Слід звернути увагу, що перший та останній рядки відрізняються значенням тільки в одному розряді.

Для визначення мінімальної ДНФ за допомогою карт Карно необхідно поєднати усі клітинки, які вміщують одиницю, у замкнені прямокутні області (контури). У цих контурах кількість клітинок у кожній може дорівнювати 2^k, де $k = 0, 1, 2, 3, \ldots, n$. Зазначені області можуть перетинатися, і одна й та сама клітинка може належати кільком областям. Але слід пам'ятати, що будь-який контур повинен мати хоча б одну клітинку, яка належить виключно йому.

Кількість таких областей має бути якомога меншою, а кількість клітинок у кожній такій області – якомога більшою. За цих умов кількість аргументів для членів мінімізованої функції буде мінімальною. При охопленні усіх точок карти Карно слід пам'ятати, що права та ліва границі карти – це сусідні стовпчики, так само, як верхній та нижній рядки карти Карно – це сусідні рядки (наприклад, чотири одиниці, розташовані в кутах карти, утворюють один контур).

Слід також зазначити, що якщо для карти Карно кількість нулів значно менша за кількість одиниць, то зручніше склеювати саме їх. Не варто тільки забувати, що у цьому випадку ми отримаємо не саму мінімальну функцію, а її інверсію, яку потім необхідно проінвертувати.

Розглянемо приклади мінімізації по картах Карно.

***Приклад* 2.19.** Мінімізувати логічну функцію

$$f(x_1,x_2) = (\overline{x_1} + \overline{x_2})(x_1 + \overline{x_2})(x_1 + x_2).$$

x_1 \ x_2	0	1
0	0	0
1	1	0

$$f(x_1,x_2) = x_1\overline{x_2}.$$

***Приклад* 2.20.** Мінімізувати логічну функцію

$$У(x_1,x_2,x_3) = \Sigma_0(2, 3, 4, 6).$$

x_1 \ x_2x_3	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$f(x_1,x_2) = \overline{x_1}x_2 + x_1\overline{x_3}.$$

***Приклад* 2.21.** Мінімізувати логічну функцію

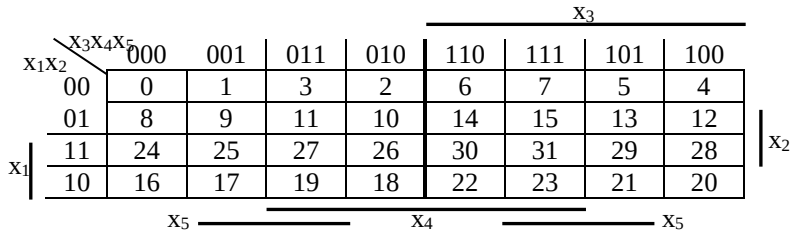
$$f(x_1,x_2,x_3,x_4) = \Sigma_0(0, 2, 4, 7, 8, 10, 12, 15).$$

x_1x_2 \ x_3x_4	00	01	11	10
00	1	0	0	1
01	1	0	1	0
11	1	0	1	0
10	1	0	0	1

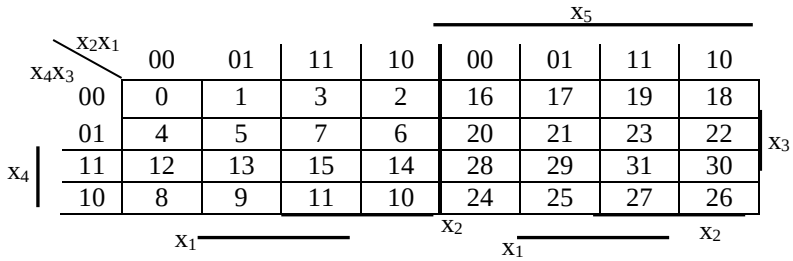
$$f(x_1,x_2) = \overline{x_3}\,\overline{x_4} + \overline{x_2}\,\overline{x_4} + x_2x_3x_4.$$

***Приклад* 2.22.** Побудувати карту Карно для п’яти змінних.

Скористаємося правилом побудови карт. Карта для п’яти змінних має такий вигляд:



Для цієї карти сусідніми є клітини, які розташовуються симетрично проведеній осі. Частіше карти Карно для $n > 4$ складаються з ідентичних карт Карно для чотирьох змінних. Дві карти Карно для чотирьох змінних будемо називати сусідніми, коли вони мають загальну грань. Клітини, які розташовані в однакових місцях сусідніх карт Карно, для 4-х змінних є сусідніми, наприклад:

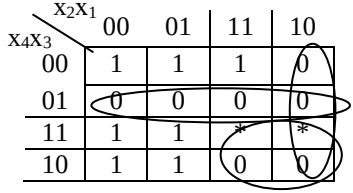


Мінімізація неповністю визначених функцій за допомогою карт Карно

Відомо, що деякі комбінації вхідних змінних логічних функцій з’явитися не можуть (їх значення неістотно). У таких випадках говорять про невизначені набори. На картах Карно невизначені набори позначаються яким-небудь символом, наприклад, “*”. Такі елементи можуть довільним чином входити до груп як одиничних, так й нульових об’єднань.

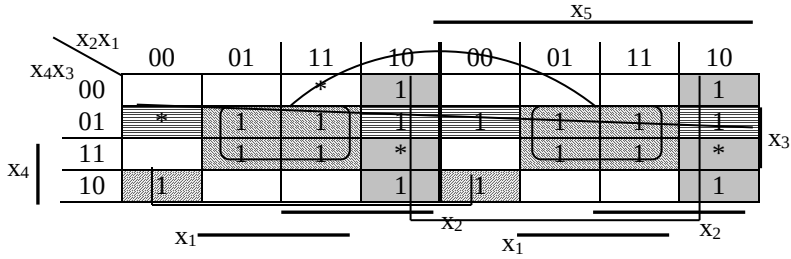
***Приклад* 2.23.** Мінімізувати логічну функцію, яка задана своїми нульовими наборами та станами невизначеності

$$f(x) = \Pi_0(2, 4, 5, 6, 7, 10, 11) + \Pi_н(14, 15).$$



$$f(x) = (x_3 + \overline{x}_4)(x_2 + \overline{x}_1)(x_4 + x_2).$$

***Приклад* 2.24.** Мінімізувати логічну функцію, яка задана картою Карно.



$$f(x) = \overline{x}_4x_3 + x_3x_1 + x_2\overline{x}_1 + x_4\overline{x}_3\overline{x}_2\overline{x}_1.$$

За допомогою карт Карно можна визначати **скорочені** і **тупикові** диз'юнктивні нормальні форми логічних функцій.

Для **отримання скороченої ДНФ** логічної функції по карті Карно слід знайти всі максимальні правильні контури, які охоплюють одиничні клітинки, виписати відповідні їм прості імпліканти й об′єднати їх знаком диз'юнкції. При мінімізації неповністю визначених логічних функцій контури можуть охоплювати клітинки, відповідні умовним наборам, однак кожний виділений максимальний контур повинен охоплювати хоча б одну клітинку, яка містить в собі одиницю.

Для **отримання тупикової ДНФ** логічної функції по карті Карно необхідно знайти таку сукупність максимальних правильних контурів, у якій кожний контур охоплював хоча б одну клітинку, яка містить в собі одиницю, яка не охоплена іншими контурами, а сукупність контурів була б повною, тобто охоплювала б усі одиничні клітинки карти Карно. Крім того, необхідно виписати відповідні їм прості імпліканти й об'єднати їх знаком диз'юнкції.

При визначенні тупикових ДНФ логічних функцій по картах Карно насамперед мусять бути визначені максимальні правильні контури, відповідні дійсним простим імплікантам,. які складають ядро логічної

функції. Характерною ознакою такого контуру є входження в нього хоча б однієї одиничної клітини, яка може бути охоплена тільки цим контуром.

Після набору контурів, відповідних ядру логічної функції, для покриття одиничних клітин, які залишились, вводяться додаткові максимальні контури, доки сукупність контурів не стане повною. При цьому необхідно щоб кожний із виділених контурів охоплював хоча б одну клітину, не охоплену іншими контурами.

Приклад 2.25. Визначимо тупикові ДНФ логічної функції, заданої на карті Карно (рис. 2.37, *а*).

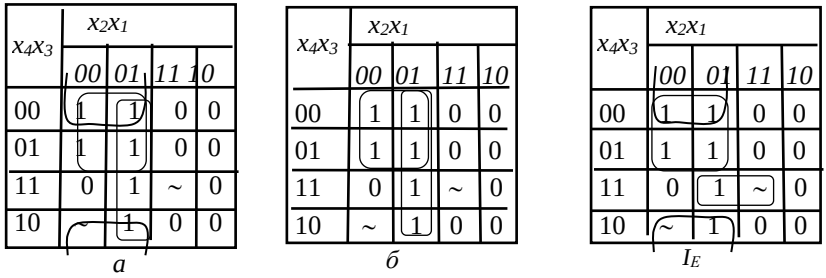


Рис. 2.37. Повні сукупності максимальних правильних контурів, які відповідають скороченій (а), мінімальній (б) і тупиковій (в) формам логічної функції

На рис. 2.37, *б* показані карти Карно, на яких виділені сукупності контурів, що відповідають двом різним тупиковим формам логічної функції. Знайдемо відповідні виділеним контурам прості імпліканти і запишемо дві тупикові форми даної логічної функції:

$$f_{T1}(x_4,x_3,x_2,x_1) = x_1\bar{x}_2 \vee \bar{x}_2\bar{x}_4;$$

$$f_{T2}(x_4,x_3,x_2,x_1) = \bar{x}_2\bar{x}_4 \vee \bar{x}_2\bar{x}_3 \vee x_1x_3x_4.$$

Тупикова форма $f_{Ti}(x_4,x_3,x_2,x_1)$ містить меншу кількість змінних і є мінімальною диз'юнктивною нормальною формою логічної функції, що розглядається.

За допомогою карти Карно можуть бути отримані скорочені й тупикові кон'юнктивні нормальні форми логічних функцій. Методика отримання скорочених і тупикових КНФ повністю аналогічна отриманню відповідних диз'юнктивних форм.

На рис. 2.38 показані двоклітинний і чотириклітинний максимальні правильні контури, які охоплюють нульові і неозначені клітинки карти Карно. Відповідні їм прості імпліценти мають вигляд:

$$\delta_1(x) = x_2 \vee x_3 \vee \overline{x}_4; \quad \delta_2(x) = \overline{x}_2 \vee x_4.$$

x_4x_3 \ x_2x_1	00	01	11	10
00	1	1	0	~
01	1	~	~	0
11	0	~	1	1
10	1	~	1	1

1 2

Рис. 2.38. Сукупность максимальних правильних контурів, які відповідають простим імпліцентам $\delta_1(x)$, $\delta_2(x)$

Сформулюємо правило отримання скороченої і тупикової кон'юнктивних форм.

Для отримання скороченої КНФ логічної функції по карті Карно треба знайти всі максимальні правильні контури, які охоплюють нульові та неозначені клітини, виписати відповідні їм прості імпліценти й об'єднати їх знаком кон'юнкції. При цьому кожний з виділених максимальних контурів повинен охоплювати хоча б одну нульову клітин.

Для отримання тупикової КНФ логічної функції по карті Карно слід знайти таку сукупність максимальних правильних контурів, у якій кожний контур цієї сукупності охоплював хоча б одну клітин, яка містить нуль і є неохопленою іншими контурами. Сукупність контурів повинна бути мінімальною. Виписати відповідні їм прості імпліценти і об′єднати їх знаком кон'юнкції.

2.2.2. Алгебра логіки при аналізі та синтезі логічних функцій

***Приклад* 2.26.** Синтезувати перетворювач кодів, який призначений для управління семисегментним індикатором, що використовується в програмі моделювання логічних схем EWB-5.12. Структурна схема управління індикатором наведена на рис. 2.39.

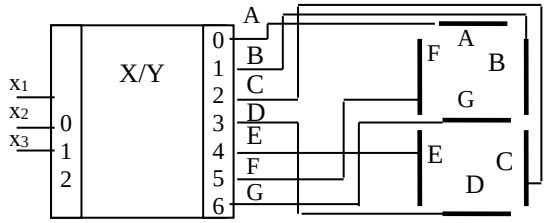


Рис. 2.39. Структурна схема управління індикатором

Функціональна схема цифрового автомата управління індикатором наведена на рис. 2.40.

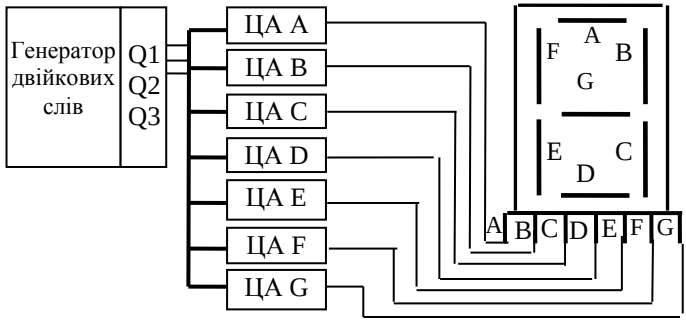


Рис. 2.40. Функціональна схема цифрового автомата управління індикатором

Для побудови функціональної схеми управління індикатором спочатку необхідно заповнити таблицю істинності (табл. 2.7), в яку заносяться послідовно значення сегментів кожної літери.

Таблиця 2.7

x_1	x_1	x_3	A	B	C	D	E	F	G	
0	0	0	1	1	1	0	1	1	0	П
0	0	1	1	0	0	1	1	1	1	Е
0	1	0	1	0	0	1	1	1	0	**Г**
0	1	1	1	1	0	0	1	1	1	Р
1	0	0	1	1	1	1	1	1	0	О
1	0	1	1	1	1	1	1	1	1	В
1	1	0	0	0	0	0	0	0	0	
1	1	1	0	0	0	0	0	0	0	

Запишемо в карти Карно значення функцій кожних літер та мінімізуємо їх (звернемо увагу, що f(A) = f(E) = f(F)):

x_1 \ x_2x_3	00	01	11	10
0	1	1	1	1
1	1	1	0	0

f(A) = f(E) = f(F) = $\overline{x_1} + \overline{x_2}$;

x_1 \ x_2x_3	00	01	11	10
0	1	0	1	0
1	1	1	0	0

f(B) = $\bar{x}_2\bar{x}_3 + x_1\bar{x}_2 + \bar{x}_1x_2x_3$;

x_1 \ x_2x_3	00	01	11	10
0	1	0	0	0
1	1	1	0	0

f(C) = $\bar{x}_2\bar{x}_3 + x_1\bar{x}_2$;

x_1 \ x_2x_3	00	01	11	10
0	0	1	0	1
1	1	1	0	0

f(D) = $x_1\bar{x}_2 + \bar{x}_2x_3 + \bar{x}_1x_2\bar{x}_3$;

x_1 \ x_2x_3	00	01	11	10
0	0	1	1	0
1	1	1	0	0

f(G) = $\bar{x}_1x_3 + x_1\bar{x}_2$.

За отриманими логічними функціями побудуємо функціональну схему послідовного відображення слова ПЕТРОВ, яка наведена на рис. 2.41. Якщо з отриманої схеми вилучити однакові логічні елементи, то спрощена таким чином функціональна схема матиме вигляд, наведений на рис. 2.42.

Слід відмітити, що для двох входових елементів можливо збільшити кількість входів до восьми, відкриваючи подвійним натисканням на знак компонента діалогове вікно (рис. 2.43).

На практиці можливо проводити мінімізацію з використанням логічного конвертора, в якому необхідно вказувати по одній логічній функції, потім перетворити таблицю істинності в логічний вираз і далі в функціональну схему.

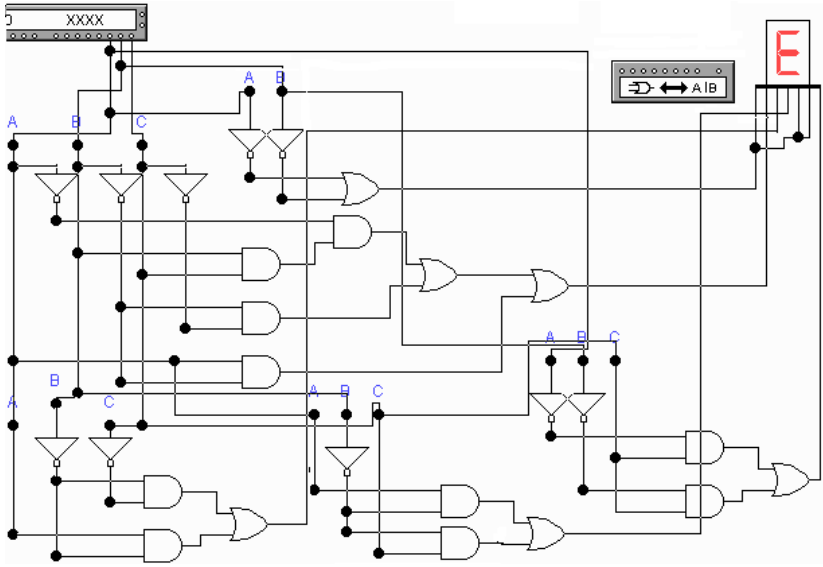


Рис. 2.41. Функціональна схема послідовного відображення слова ПЕТРОВ

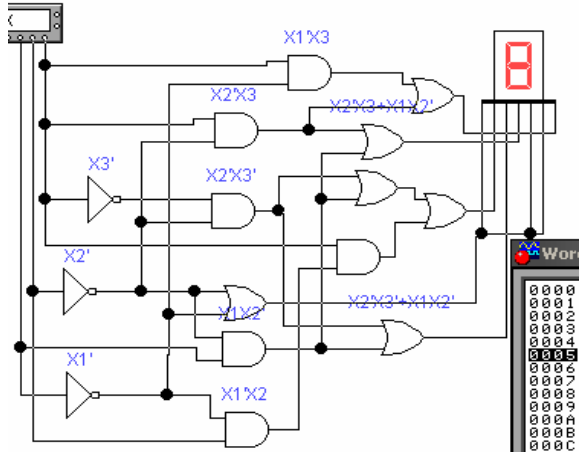


Рис. 2.42. Спрощена функціональна схема послідовного відображення слова ПЕТРОВ

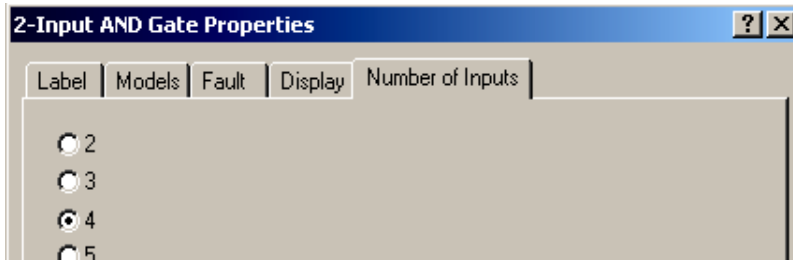


Рис. 2.43. Вікно встановлення кількості входів логічного елемента

2.2.3. Завдання до самостійних досліджень простих логічних елементів

1. *Мет а*

Закріплення теоретичного матеріалу, набуття навиків створення і моделювання схем простих цифрових логічних елементів, робота з різними вимірювальними приладами.2. *Теми для попереднього опрацювання*

2.1. Логічні функції та способи їх завдання.

2.2. Основні аксіоми та закони алгебри логіки.

3. *Завдання*

Дослід 2.1. Дослідження функціонування логічних елементів у динамічному режимі роботи.

Включити EWB. Вибрати логічний елемент згідно з табл. 2.8. Для цього необхідно натиснути на піктограму з іменем DIGIT. Із вікна, що з'явилося натисканням на мишку витягнути на робоче поле необхідну серію елементів, наприклад 74xx. Вибрати згідно з варіантом мікросхему та натиснути кнопку Accept. Для коректного проведення досліджень елементів з відкритими колекторами необхідно на вихід елемента навантажити резистор R = 1кОм.

1.1. Зібрати електричну схему проведення дослідження з отриманим логічним елементом, як наведено на рис. 2.44. Вибрати в нижньому рядку піктограм піктограму з ім'ям Sources та витягнути на робоче поле зображення заземлення  та джерела напруги V_{cc}. Сигнал заземлення під'єднати до контакту GVD мікросхеми, а джерело напруги V_{cc} – до контакту V_{cc} мікросхеми.

На один із входів логічного елемента подати сигнал з виходу гене-

ратора прямокутних імпульсів, а на останні входи – такі логічні сигнали, які дозволяють провести дослідження зміни сигналу на виході.

Таблиця 2.8

№ з/п	Серія SN74	Вітчизняні МС	Функціональне призначення
1	7400	155ЛА3	4 елементи 2І-НІ
2	7402	155ЛЕ1	4 елементи 2АБО-НІ
3	7403	155ЛА9	4 елементи 2І-НІ з відкритим колектором
4	7408	155ЛИ1	4 елементи 2І
5	7409	155ЛИ2	4 елементи 2І з відкритим колектором
6	7410	155ЛА4	3 елементи 3І-НІ
7	7411	555ЛИ3	3 елементи 3І
8	7412	155ЛА10	3 елементи 3І-НІ з відкритим колектором
9	7420	155ЛА1	2 елементи 4І-НІ
10	7421	155ЛИ6	2 елементи 4І
11	7422	155ЛА7	2 елементи 4І-НІ з відкритим колектором
12	7426	155ЛА11	4 елементи 2І-НІ з відкритим колектором
13	7428	155ЛЕ5	4 елементи 2АБО-НІ
14	7430	155ЛА2	Елемент 8І-НІ
15	7432	155ЛЛ1	4 елементи 2АБО
16	7438	155ЛА13	4 елементи 2І-НІ з відкритим колектором

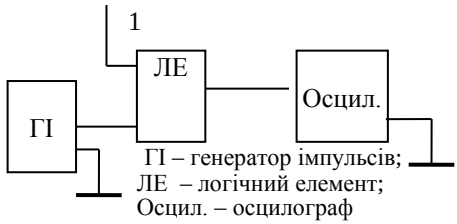


Рис. 2.44. Схема проведення дослідження елемента 2І-НІ

1.2. На виході генератора прямокутних імпульсів одержати позитивні імпульси з амплітудою $U_{вх}$ = 4 В та частотою згідно з варіантом із табл. 2.9:

Таблиця 2.9

№ з/п	Частота f, кГц	№ з/п	Частота f, кГц
1	0,5	9	1
2	1	10	2
3	2	11	3
4	4	12	4
5	5	13	5
6	6	14	6
7	7	15	7
8	8	16	9

1.3. Занести до звіту схему та результати досліджень згідно з прикладом (рис. 2.45).

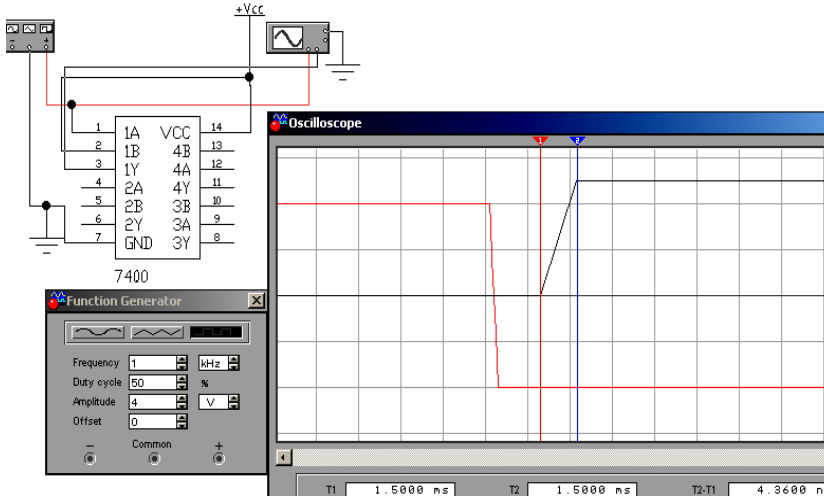


Рис. 2.45. Схема та результати досліджень

Визначити тривалість фронтів $t_Ф$ сигналу на виході, час затримки вихідного сигналу та занести результати до звіту. Тривалість імпульсу можливо отримати шляхом підведення лівої червоної вертикальної лінії до початку інтервалу сигналу, який виміряються, та підведення синьої вертикальної лінії до закінчення інтервалу. В правому нижньому віконці осцилографа відображається необхідний параметр сигналу.

Дослід 2.2. Дослідження впливу ємності навантаження на форму вихідних сигналів.

Для цього до виходу логічного елемента підключити спочатку ємність C_1 = 10 пФ, а потім C_2 = 10 мФ). Навести схему, таблицю та графіки фронтів та спадів вихідного сигналу в залежності від різних ємностей. Схеми дослідження та діаграми наведені на рис. 2.46 та 2.47. Зробити необхідні висновки.

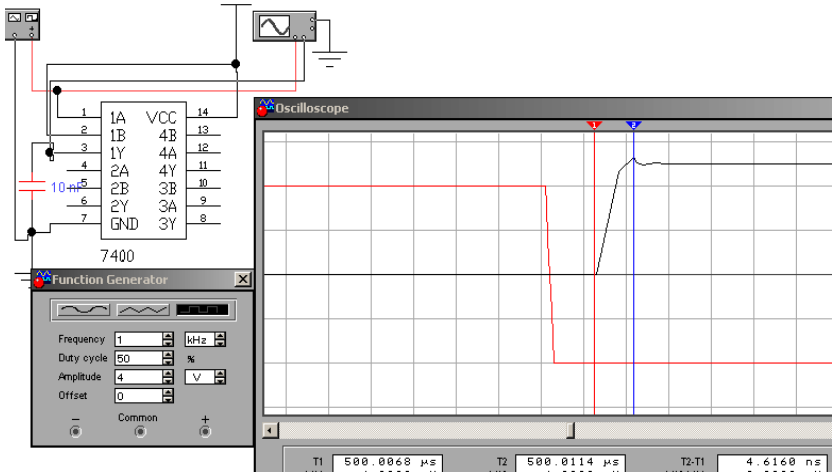


Рис. 2.46. Схема та результати досліджень з ємністю C = 10 нФ

Дослід 2.3. Дослідження мікросхеми з використанням генератора слів, багатоканального логічного аналізатора та світлодіоду.

3.1. Для дослідження мікросхеми необхідно підключити генератор тестів, логічний аналізатор та світлодіод за схемою, яка наведена на рис. 2.48. Подаючи на входи вибраного елемента мікросхеми усі можливі комбінації сигналів “0” та “1” та спостерігаючи за станом світлодіоду (випромінює, не випромінює), впевнитися, що елемент виконує необхідну логічну функцію.

Наприклад, у зв’язку з тим, що логічний елемент 7440 виконує функцію 4I-HI, для проведення досліджень його працездатності необхідно подати чотири управляючі сигнали.

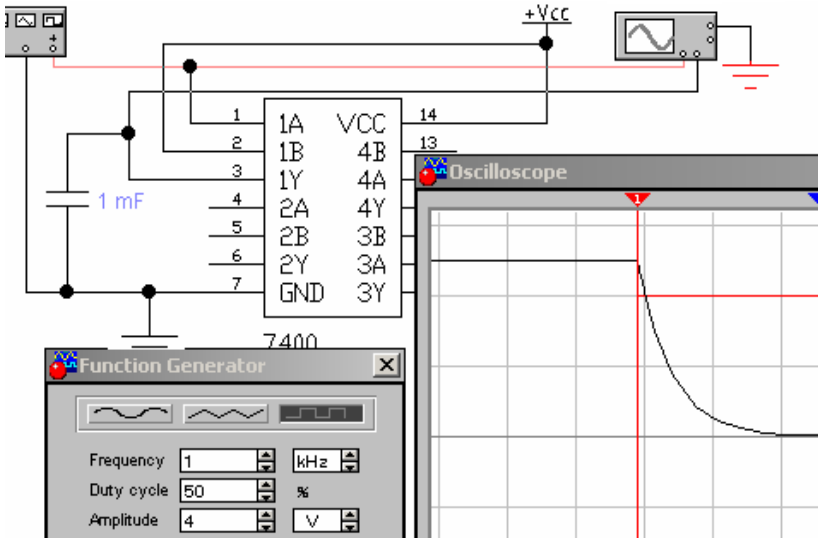


Рис. 2.47. Схема та результати досліджень з ємністю C = 1 мФ

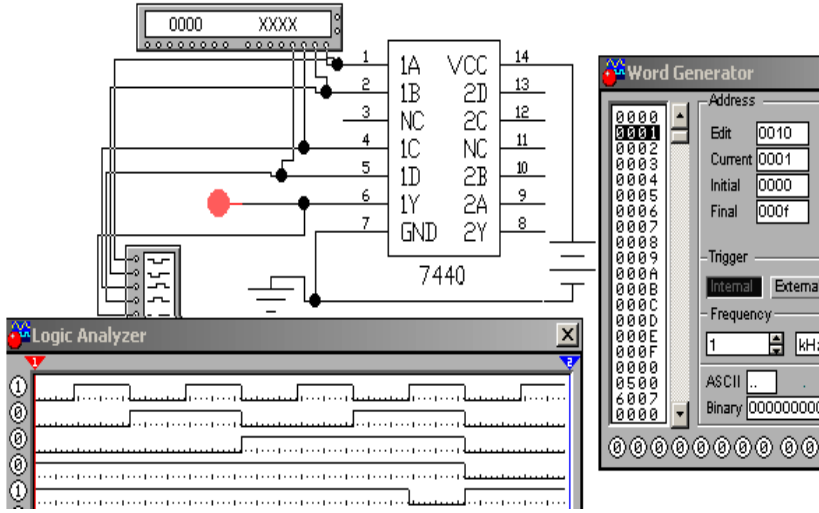


Рис. 2.48. Дослідження логічного елемента 7440 з підвищеною навантажувальною спроможністю

Для цього в розширеному вікні генератора слів слід виставити необхідні двійкові набори у шістнадцятковому вигляді. Ця дія може бути виконана як вручну, так і за допомогою властивостей самого генератора. Для цього необхідно натиснути на клавішу Pattern... та вибрати **Up counter**, як наведено на рис. 2.49.

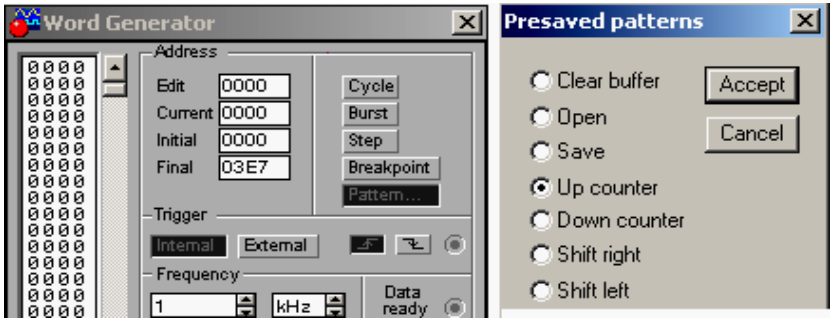


Рис. 2.49. Вид вікна для автоматичного заповнення генератора

3.2. Для дослідження порогового значення переключення світлодіоду необхідно скористатися змінним резистором, під'єднавши його до схеми згідно з рис. 2.50.

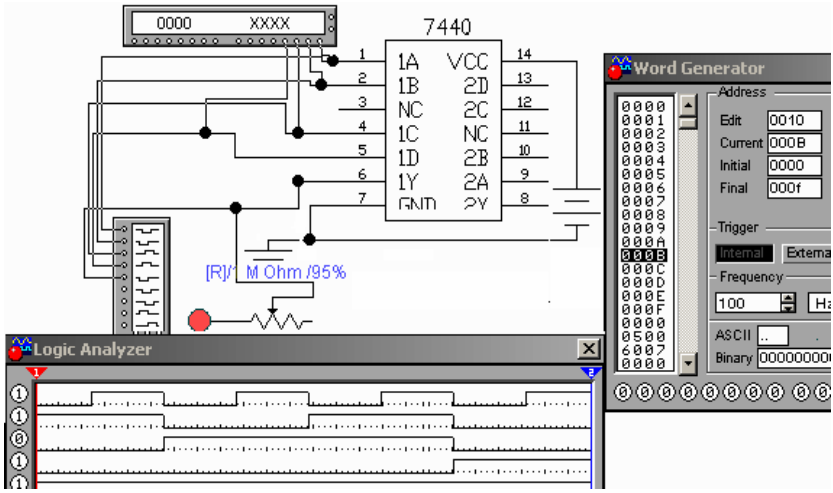


Рис. 2.50. Схема дослідження світлодіоду

Для зміни властивостей резистора необхідно натиснути праву клавішу мишки, як показано на рис. 2.51.

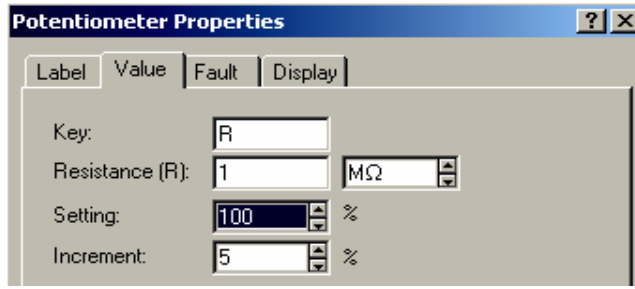


Рис. 2.51. Вікно вказування властивостей резистора

У вікні, що з'явилося, указати максимальне значення опору. Включити живлення схеми і натисканням клавіші поступово зменшувати опір. Записати величину опору, при якому засвітиться світлодіод.

4. *Питання для самоперевірки*

4.1. Перерахуйте основні аксіоми алгебри логіки.

4.2. Як на тривалість фронтів впливають навантажені резистори та ємності?

2.3. Транзистори як технічна основа реалізації логічних функцій

З кінця 1961 року інтегральні схеми у великих об'ємах стали надходити у продаж. Логічні схеми, відомі в даний час як стандартні інтегральні схеми транзисторно-транзисторної логіки (ІС ТТЛ), з'явилися в результаті робіт, проведених в різних наукових організаціях, але основа їх закладена в 1961 році співробітниками невеликої фірми з Калвер-Сіті (штат Колорадо, США), що називалася Pacific Semiconductors Unc.

У 1962 році на основі пленарного технологічного процесу був створений новий тип польових транзисторів, що працюють за принципами використання тільки основних носіїв, – польовий транзистор МДП-типу з ізольованим затвором. Впровадження пленарного процесу для групового виготовлення схем на основі транзисторів МДП-типу

забезпечило розвиток нового схемотехнологічного напряму – МДП ІС, які, так само, як і біполярні ІС, стали використовуватися перш за все для побудови логічних схем і пристроїв, що запам'ятовують.

З моменту появи МДП ІС почався паралельний розвиток двох головних схемотехнологічних напрямів у області цифрових ІС – біполярних технологій і МДП-технологій, що носять характер суперництва, яке продовжується й, імовірно, продовжуватиметься в майбутньому, оскільки переваги біполярних і МДП ІС до кінця ще не реалізовані, а недоліки постійно усуваються. Усередині цих двох великих категорій залежно від схем, матеріалів і особливостей обробки існує багато різних технологічних процесів.

Логічний елемент (логічний вентиль) – це електронна схема, що виконує деяку просту логічну операцію.

Логічний елемент може бути реалізований у вигляді окремої інтегральної схеми. Часто інтегральна схема містить декілька логічних елементів, які використовуються в пристроях цифрової електроніки (логічних пристроях) для виконання простого перетворення логічних сигналів.

2.3.1. Класифікація і основні парамет ри логічних елемент ів

Розглянемо найбільш широко використовувану класифікацію, що історично склалася. Вона побудована і з урахуванням того, які електронні прилади є основними у відповідних інтегральних схемах, і з урахуванням особливостей використаних рішень схемотехніки.

Виділяються такі класи логічних елементів (так званої логіки):

– резисторно-транзисторна логіка (РТЛ);
– діодно-транзисторна логіка (ДТЛ);
– транзисторно-транзисторна логіка (ТТЛ);
– емітерно-зв'язана логіка (ЕЗЛ);
– транзисторно-транзисторна логіка з діодами Шоттки (ТТЛШ);
– логіка на основі МОП-транзісторів з каналами типу р (р-МДП);
– логіка на основі МОП-транзісторів з каналами типу п (n-МДП);
– логіка на основі комплементарних ключів на МДП-транзисторах (КМДП, КМОП);
– інтегральна інжекційна логіка ІІЛ (І2Л);
– логіка на основі напівпровідника з арсеніду галію GaAs.

У даний час використовуються переважно такі логіки: ТТЛШ і

КМОП. Застаріли і практично не використовуються РТЛ, а також ТТЛ і ЕСЛ. Для пристроїв, що розробляються в даний час, рекомендується використовувати КМОП-логіку, а також логіку на основі GaAs.

Логічні елементи і інші цифрові електронні пристрої випускаються у складі серій мікросхем. **Серія мікросхем –** це сукупність мікросхем, що характеризуються загальними технологічними і схемотехнічними рішеннями, а також рівнями електричних сигналів і напруги живлення.

Приведена класифікація охоплює не тільки власне логічні елементи, але і інші цифрові пристрої, зокрема мікропроцесорні. Проте тут слід враховувати, що при виробництві складних цифрових пристроїв деякі логіки не використовувалися і не використовуються.

Наведемо приклади серії мікросхем: ТТЛ: К155, КМ155, К133, КМ133; ТТЛШ: 530, КР531, КМ531, КР1531, 533, К555, КМ555, 1533, КР1533; ЭСЛ: 100, К500, К1500; КМОП: 564, К561, 1564, КР1554; GaAs: К6500.

Кожна серія мікросхем характеризується деяким набором параметрів, що дають достатньо докладне уявлення про цю серію. При визначенні цих параметрів орієнтуються саме на логічні елементи – прості пристрої серії мікросхем. Відповідно до цього говорять про параметри не серії мікросхем, а про параметри логічних елементів даної серії.

Основою кожної серії цифрових мікросхем є базовий логічний елемент, на якому можуть бути зібрані пристрої, що виконують будь-які логічні операції. Звичайно як базовий беруть елементи, що виконують операції І-НІ або АБО-НІ. ***До основних парамет рів*** базового елемента відносяться швидкодія, споживана потужність, перешкодостійкість, здатність навантаження, рівні напруг джерела живлення, рівні логічних одиниць і нулів.

Розглянемо найбільш важливі з параметрів.

Швидкодію характеризують часом затримки розповсюдження сигналу $t_{зг\,р}$ і максимальною робочою частотою F_{max}. Звернемося до часових діаграм, що ідеалізуються, відповідних елементу НІ (інвертору) (рис. 2.52).

Через $U_{вх1}$ і $U_{вих1}$ позначені рівні вхідної і вихідної напруг, відповідні логічній одиниці, а через $U_{вх0}$ і $U_{вих0}$ – відповідні логічному нулю. Розрізняють час затримки $t_{зр10}$ розповсюдження при перемиканні зі стану “1” до стану “0” і при перемиканні зі стану “0” до стану “1” – $t_{зр01}$, а також середній час затримки розповсюдження $t_{зтр}$, причому $t_{зтр} = 0{,}5(t_{зтр10} + t_{зтр01})$.

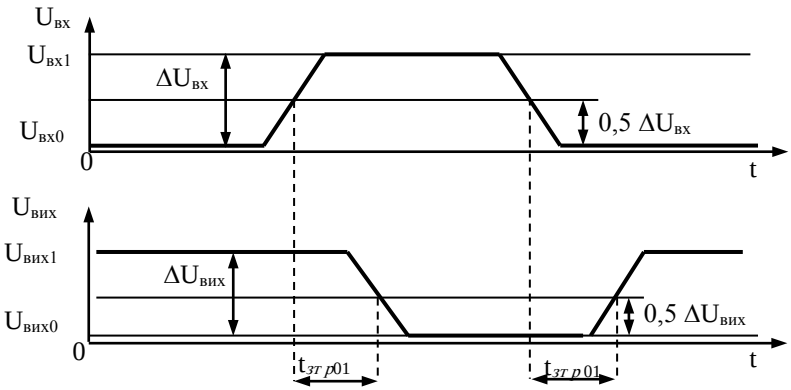


Рис. 2.52. Часові діаграми елемента НІ

Час затримки прийнято визначати по перепадах рівнів 0,5 $\Delta U_{вх}$ і 0,5 $\Delta U_{вих}$. Максимальна робоча частота F_{max} – це частота, при якій зберігається працездатність схеми.

Знаючи час затримки базового елемента, можна шляхом підсумовування $t_{зр}$ розрахувати швидкодію будь-якої складної логічної схеми для всіх послідовно включених елементів. Якщо схема має кола зворотного зв'язку, то черговий перепад вхідної напруги повинен починатися не раніше, ніж закінчиться попередня зміна напруги, що надходить по колу зворотного зв'язку з виходу схеми на її вхід. Ця закономірність пов'язує час затримки $t_{зр}$ розповсюдження з граничною робочою частотою $f_{гр}$, яка є основним параметром цифрових автоматів:

$$f_{гр} \approx 1/ \Sigma\, t_{зр\,i}$$

Навантажувальна спроможність характеризується коефіцієнтом об'єднання по входу $K_{об}$ і коефіцієнтом розгалуження по виходу $K_{роз}$ (іноді використовують термін "коефіцієнт об'єднання по виходу"). Величина $K_{об}$ – це кількість логічних входів, величина $K_{розг}$ – максимальна кількість однотипних логічних елементів, які можуть бути підключені до виходу даного логічного елемента. Типові значення їх такі: $K_{об}$ = 2...8, $K_{роз}$ = 4...10. Для елементів з підвищеною здатністю навантаження $K_{розг}$ = 20...30.

Перешкодостійкість логічних елементів оцінюють у статичному і

динамічному режимах. При цьому ***статична перешкодостійкість*** визначається рівнем випадкової напруги, яка може бути присутньою на його вході без небезпеки помилкового спрацьовування. ***Динамічна перешкодостійкість*** залежить від форми, тривалості і амплітуди перешкоди, а також від швидкості перемикання і статичної перешкодостійкості логічного елемента.

Логічні елементи в процесі роботи знаходяться або в статичному режимі (в стані "1" або "0"), або в динамічному (перехід з "1" в "0" і назад).

Потужність, споживана елементом від джерела живлення, в кожному стані різною. У зв'язку з цим вимірюють статичну середню потужність: $P_{СР} = 0{,}5(P^0 + P^1)$, де P^0 – потужність, споживана елементом у стані "0"; P1 – потужність у стані "1", і динамічну потужність $P_д$, визначувану на граничній робочій частоті. При конструюванні цифрових пристроїв необхідно враховувати, що потужність, споживана мікросхемами, збільшується з підвищенням частоти сигналів.

При серійному випуску мікросхем з'явилася необхідність стандартизації **напруги живлення**. Так, для більшості серій, побудованих на біполярних транзисторах, що працюють у ключовому режимі (так звана транзисторно-транзисторна логіка), стандартною напругою живлення є 5 В + 5%.

Для ТТЛ також встановлені рівні логічного "0" ($0 < U_0 < 0{,}4$ В) і логічної "1" ($2{,}4 > U_1 > 5$ В). Важливими є також такі параметри:

– напруга живлення;

– вхідні порогові напруги високого і низького рівня $U_{вх1\ пор}$ і $U_{вх0\ пор}$, відповідно до зміни станів логічного елемента;

– вихідні напруги високого і низького рівнів $U_{вих1}$ і $U_{вих0}$

Використовують й інші параметри.

2.3.2. Загальні відомості про біполярний транзистор

Дію транзистора можна порівняти з дією греблі. За допомогою постійного джерела (течії ріки) і греблі створений перепад рівнів води. Затрачаючи дуже невелику енергію на вертикальне переміщення затвора, ми можемо керувати потоком води великої потужності, тобто керувати енергією могутнього постійного джерела.

Біполярним транзистором називається напівпровідниковий прилад, із двома взаємодіючими електричними переходами, який має три

або більше виводів і призначений для посилення електричних коливань за струмом, напругою чи потужністю.

Слово “біполярний” означає, що фізичні процеси в транзисторі визначаються рухом носіїв заряду обох знаків (електронів і дірок). Напівпровідники в яких основними носіями заряду є електрони називаються напівпровідниками ***n***-типу, а напівпровідники в яких основними носіями заряду є дірки, тобто позитивно заряджені частини – напівпровідники ***p***-типу. Взаємодія переходів забезпечується тим, що вони розташовуються досить близько – на відстані, яка менша ніж дифузійна довжина (довжина вільного пробігу носіїв заряду).

Два ***p-n***-переходи утворюються в результаті чередування областей з різним типом електропровідності. У залежності від порядку чередування розрізняють біполярні транзистори типу ***n-p-n*** (чи зі структурою ***n-p-n***) і типу ***p-n-p*** (чи зі структурою ***p-n-p***), схематичні та умовні графічні зображення яких показані на рис. 2.53

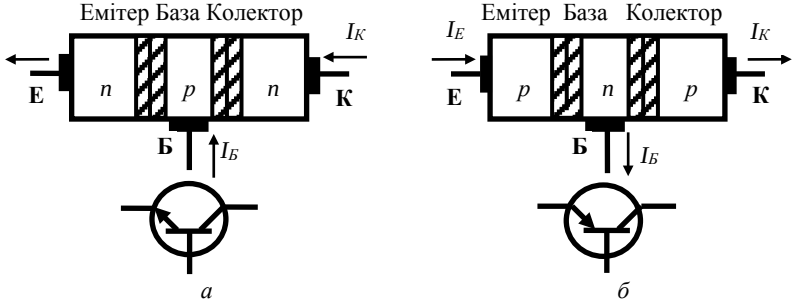


Рис. 2.53. Схематичні та умовні графічні зображення транзистору ***n-p-n*** (а) та ***p-n-p*** (б).

Спрощена структура реального транзистора типу ***n-p-n*** зображена на рис. 2.54. У цій структурі існують два переходи з неоднаковою площею: площа лівого переходу n_1^+- ***p*** менше, ніж у переході n_2 - ***p***. Крім того, у більшості біполярних транзисторів одна з крайніх областей (n_1 з меншою площею) має більшу концентрацію основних носіїв заряду, ніж

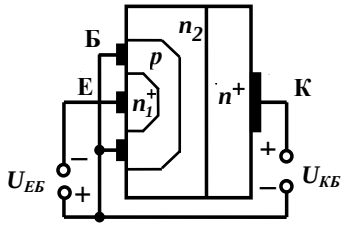


Рис. 2.54. Структура реального біполярного транзистора.

інша крайня область (n_2).

Сильнолегована область позначена верхнім індексом "+" (n^+). Тому транзистор є асиметричним приладом. Асиметрія виражається й у назвах крайніх областей: сильнолегована область з меншою площею (n_1^+) називається емітером, а область n_2 – колектором. Відповідно перехід n_1^+- p називають емітерним, а $n_2 - p$ колекторним. Середня область (p) називається базовою (чи базою). Контакти з областями транзистору позначені на рис. 2.53 і рис. 2.54 буквами: **Е** – емітер; **Б** – база; **К** – колектор.

Принцип роботи біполярного транзистора

Основні властивості біполярного транзистору визначаються процесами в базовій області, яка забезпечує взаємодію емітерного і колекторного переходів (рис. 2.55). Основні фізичні процеси в ідеалізованому транзисторі зручно розглядати на прикладі p-n-p транзистора тому що напрямок руху інжектируємих з емітера носіїв (дірок) співпадає з напрямком струму.

У нормальному активному режимі на емітерному переході діє пряма напруга $U_{ЕБ}$, тобто напруга яка сприяє переносу (інжекції) основних носіїв заряду з емітерної та базової областей. Тому прямий струм переходу

$$I_E = I_{Ep} + I_{En} + I_{Eрек}, \qquad (2.1)$$

де I_{Ep}, I_{En} – інжекційні струми дірок (з емітера в базу) і електронів (з бази до емітеру);

$I_{Eрек}$ – складова струму, викликана рекомбінацією в переході тих дірок і електронів, енергія яких недостатня для подолання потенційного бар'єру переходу. Відносний внесок цієї складової в струм переходу I_E в (2.1) тим помітніше, чим менше інжекційні складові I_{Ep} і I_{En} що визначають прямий струм у випадку ідеалізованого p-n переходу. Якщо внесок $I_{Eрек}$ незначний ($I_{Eрек} \approx 0$), то замість (2.1) можна записати

$$I_E = I_{Ep} + I_{En}. \qquad (2.2)$$

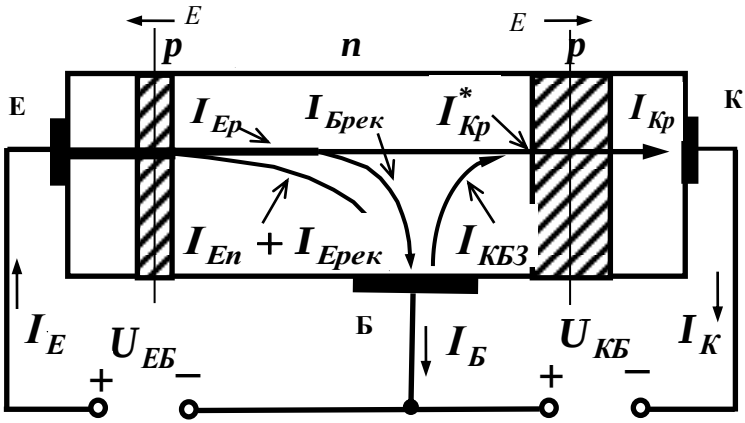


Рис. 2.55. Процеси у базовій області транзистора

Корисним у сумі струмів (вираз (2.1)) є тільки струм I_{Ep}, тому що тільки він буде далі брати участь у створенні струму колекторного переходу. "Шкідливі" складові токи емітеру I_{En} і $I_{Eрек}$ протікають через вивід бази і є складовими струмами бази, а не колектора. Тому компоненти I_{En} і $I_{Eрек}$ намагаються зменшувати.

Ефективність роботи емітерного переходу враховується коефіцієнтом інжекції емітеру

$$\gamma_E = \frac{I_{Ep}}{I_E} = \frac{I_{Ep}}{\left(I_{Ep} + I_{En} + I_{Eрек}\right)}. \qquad (2.3)$$

Цей коефіцієнт показує, яку частину в повному струмі емітеру складає корисний компонент. Якщо вважати $I_{Eрек} \approx 0$, то

$$\gamma_E = \frac{I_{Ep}}{\left(I_{Ep} + I_{En}\right)} = \left[1 + \left(\frac{I_{En}}{I_{Ep}}\right)\right]^{-1}. \qquad (2.4)$$

Коефіцієнт інжекції γ_E тим вище (ближче до одиниці), чим менше відношення $\frac{I_{En}}{I_{Ep}}$. Величина $\frac{I_{En}}{I_{Ep}} << 1$, якщо концентрація акцепторів, тобто дірок в емітерній області ***p-n-p*** транзистора, на кілька порядків вище концентрації донорів, тобто електронів в базі цього транзистора. Ця умова виконується при виробництві транзисторів.

Інжектовані до бази дірки, для якої вони являються неосновними носіями заряду починають рекомбінувати з електронами. Але рекомбінація не є миттєвим процесом, тому майже всі дірки встигають пройти через тонкий шар бази і дістатися колекторного ***p-n***-переходу. Утрату дірок у базі можна врахувати введенням струму рекомбінації дірок $I_{Б\,рек}$, так що струм дійшовших до колекторного переходу дірок можна визначити як

$$I^*_{Кр} = I_{Ep} - I_{Б\,рек} \tag{2.5}$$

Відносні втрати на рекомбінацію в базі враховуються коефіцієнтом переносу

$$X_Б = \frac{I^*_{Кр}}{I_{Ep}} = \left(1 - \frac{I_{Б\,рек}}{I_{Ep}}\right) < 1 \tag{2.6}$$

Коефіцієнт переносу показує, яка частина потоку дірок, інжектованих з емітеру до бази, доходить до колекторного переходу. Значення $X_Б$ тим ближче до одиниці, чим менше число інжектованих дірок рекомбінує з електронами – основними носіями базової області.

Щоб зменшити втрати на рекомбінацію, тобто збільшити $X_Б$, необхідно знову таки зменшувати концентрацію електронів у базі (див.(2.4)) та ширину базової області.

2.3.3. Особливості вихідних каскадів цифрових мікросхем

Часто виникає необхідність підключення виходів декількох цифрових мікросхем до одного навантаження. Одним зі способів об'єднання виходів є використання у вихідних каскадах мікросхем транзисторів, один зі виводів яких (колектор, емітер, стік, витік) нікуди не підключений. Такий вивід називають ***відкритим***.

Покажемо схематично (рис. 2.56), як об'єднуються виходи мікросхем з відкритим колектором. Таку схему називають монтажним (дротяним) АБО.

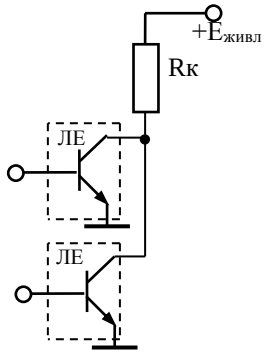


Рис. 2.56. Схема логічного елемента НІ

Якщо відкритим є колектор транзистора *n-p-n*-типу, емітер транзистора *p-n-p*-типу, стік транзистора з каналом *n*-типу, витік транзистора з каналом *p*-типу, то вивід позначають символом  .

Якщо відкритим є колектор транзистора *p-n-p*-типу, емітер транзистора *n-p-n*-типу, стік транзистора з каналом *p-типу*, витік транзистора з каналом n-типу, висновок позначають символом  .

Вихідні каскади деяких мікросхем можуть працювати в такому режимі, коли мікросхема виявляється фактично відключеною від навантаження. Це так званий третій (високоімпедансний) стан мікросхеми. Використання третього стану є ще одним способом об'єднання виходів мікросхем, який широко використовується в обчислювальній техніці, при підключенні до загальної шини багатьох пристроїв. Наведемо фрагмент схеми, що пояснює виникнення третього стану (рис. 2.57).

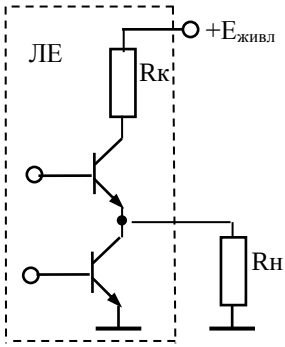


Рис. 2.57. Схема логічного елемента з третім станом

Якщо обидва транзистори закриті, то мікросхема і навантаження фактично є роз'єднаними. Наявність третього стану позначають символом  .

При використанні в єдиному цифровому пристрої мікросхем різних серій, і особливо різних логік, може виникнути проблема узгодження рівнів вхідних і вихідних напруг. Для вказаних цілей проводяться спеціальні мікросхеми, які називають перетворювачами рівня сигналів.

Інвертори, кон'юнктори і диз'юнктори на біполярних транзисторах

Біполярним транзистором називається напівпровідниковий прилад із двома взаємодіючими електричними переходами, який має три або більше виводів і призначений для посилення електричних коливань за струмом, напругою чи потужністю.

Логічний елемент НІ

Для реалізації елемента НІ використовується схема транзисторного ключа (рис. 2.58).

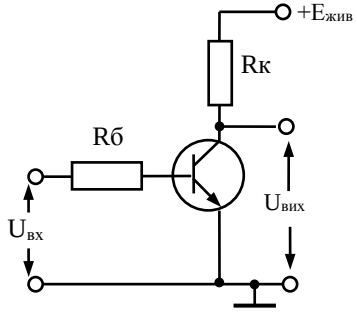


Рис. 2.58. Схема логічного елемента НІ

Транзистор може знаходитися в одному з двох стаціонарних станів – "увімкнено" або "вимкнено".

Якщо транзистор вимкнений, то струми бази і колектора транзистора дорівнюють 0. Вхідна напруга ніяк не впливає на вихід. Напруга на виході визначається тільки напругою джерела живлення $E_{живл}$. Якщо опір навантаження відсутній, то $U_{вих} = E_{живл}$. ***Вимкнений стан транзистора забезпечується вхідною напругою, меншою напруги відкриття емітерного p-n переходу транзистора (0,7 В).*** Таким чином, якщо напруга на вході відповідає рівню логічного 0, то на виході маємо рівень логічної 1.

Щоб відкрити транзистор, на вхід схеми необхідно подати напругу рівня логічної 1 і ввести транзистор в режим насичення. У режимі насичення напруга на ділянці колектор – емітер близька до 0 (~ 0,1 В).

Струм колектора насичення визначається з 2-го закону Кирхгофа для кола колектора: $E_{жив} = I_K R_K + U_{KE}$. Нехтуючи малим значенням U_{KE} при насиченні транзистора, маємо струм насичення колектора $I_{K\ НАС} = E_{жив}/R_K$. Струм бази насичення $I_{БНАС} = I_{кнас}/B$, де B – коефіцієнт передачі струму бази при сильному сигналі. Струм бази повинен перевищувати струм бази насичення $I_Б > I_{БНАС}$..

З другого закону Кирхгофа для вхідного ланцюга $U_{ВХ} = I_Б R_Б + U_{БЕ\ НАС}$ знайдемо опір резистора $R_Б$ для надійного відкриття транзистора при заданій вхідній напрузі:

$$R_Б < (U_{вх} - U_{БЕ\ НАС})\, R_к B / E_{живл}$$

Напруга $U_{БЕ\ НАС}$ – це напруга на відкритому *p-n* переході, вона дорівнює ~ 0,7 В. Таким чином, при одиничному вхідному сигналі на виході маємо рівень логічного 0. Схема виконує логічну функцію НІ.

Елементи АБО-НІ, І-НІ та інші можна побудувати з елементів АБО, І, НІ, розглянутих вище.

Схема логічного елемента АБО-НІ наведена на рис. 2.59, а схема елемента І-НІ – на рис. 2.60.

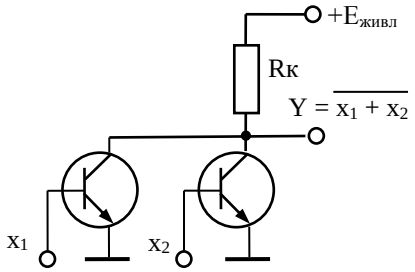


Рис. 2.59. Схема логічного елемента АБО-НІ

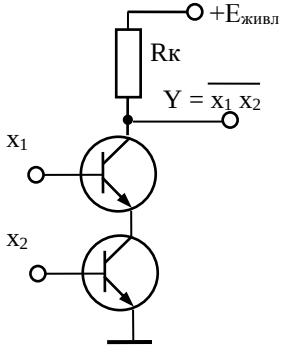


Рис. 2.60. Схема логічного елемента І-НІ

Як вже згадувалося вище, існують набори логічних функцій, які називаються повними.

До повного набору відносяться функції АБО-НІ та І-НІ. Це означає, що, маючи достатню кількість логічних елементів, наприклад, І-НІ, можна побудувати будь-яку, скільки завгодно складну, цифрову схему. Елемент І-НІ, реалізований в інтегральній технології, одержав назву базового.

Базовий логічний елемент ТТЛ І-НІ

На рис. 2.61 наведена одна з схем логічного елемента ТТЛ-елемента І-НІ. Точки *A, B, C, D, E* – основні вузли схеми, для яких будуть розраховані напруги. Приймемо за постійні величини напруг на відкритому *p-n*-переході – 0,7 В і на ділянці колектор – емітер насиченого транзистора – 0,1 В.

Розглянемо послідовно всі рядки таблиці істинності логічного елемента І-НІ.

При подачі на обидва входи *x1* і *x2* рівнів логічного 0 емітерні переходи багатоемітерного транзистора *VT1* є відкритими.

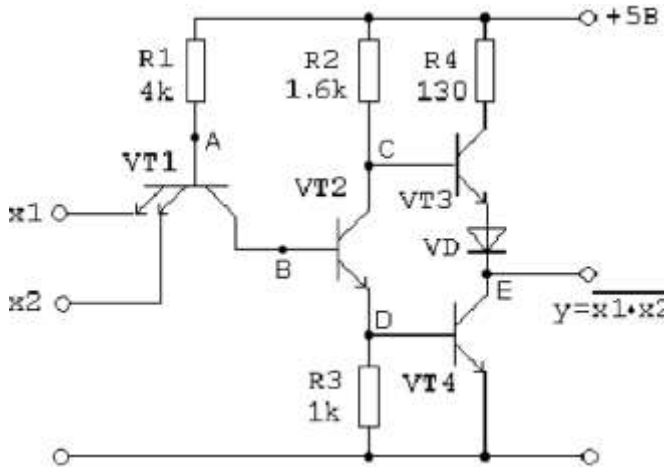


Рис. 2.61. Схема базового елемента І-НІ ТТЛ

Напруга в точці А (напруги в усіх точках схеми вимірюються по відношенню до загального дроту) складається з вхідної напруги логічного 0 і напруги на відкритому *p-n* переході. Вона може змінюватися від 0,7 В при вхідній напрузі, яка дорівнює 0, – до 1,1 В при вхідній напрузі 0,4 В (максимум напруги логічного 0).

Між точкою А і загальним дротом послідовно включені 3 *p-n*-переходи – колекторний *p-n*-перехід транзистора *VT1,* емітерний перехід транзистора *VT2* і емітерний перехід транзистора *VT4.* Для відкриття кожного необхідна напруга ~ 0,7 В, а для трьох *p-n*-переходів ~2,1 В. Напруга ж у точці А істотно менше, отже, вищеперелічені *p-n* переходи. Оскільки закриті емітерні *p-n*-переходи транзисторів *VT2* і *VT4,* то закриті й самі транзистори, тобто їх колекторні струми дорівнюють 0.

Транзистор *VT3* відкритий, оскільки на його базу подається напруга джерела живлення через резистор *R2.* Напруга в точці *C,* за відсутності навантаження на логічний елемент, близька до напруги джерела живлення (5 В). Напруга в точці *E,* тобто на виході *Y* елемента, менше напруги в точці *C* на подвоєну напругу відкритого *p-n*-переходу (напруга на емітерному переході транзистора *VT3* і на діоді *VD),* тобто дорівнює $5 - 0{,}7 \times 2 = 3{,}6$ В, а це є рівнем логічної 1. При підключенні

схеми до навантаження збільшується колекторний струм транзистора *VT3*, отже, збільшується і його базовий струм, який тече через резистор *R2*. Напруга в точці *C* зменшується, внаслідок чого зменшується напруга на виході схеми. Схема розрахована так, що при максимальному вихідному струмі напруга на виході не стає менше мінімуму рівня логічної 1 (2,4 В). Резистор *R4* обмежує вихідний струм при замиканні виходу на загальний дріт, тобто при короткому замиканні вихідних клем.

У другому і третьому рядках таблиці істинності логічного елемента І-НІ на один з входів поданий рівень логічної 1, але як і раніше на інший вхід поданий рівень логічного 0. Один з емітерних переходів транзистора *VT1* відкритий і весь аналіз схеми залишається у силі. На виході логічного елемента – рівень логічної 1.

У четвертому рядку таблиці істинності на обидва входи логічного елемента подані рівні логічної 1. Емітерні переходи транзистора *VT1* закриті, тому напруга в точці А зараз не залежить від вхідних напруг. Пригадаємо, що між точкою А і загальним дротом послідовно включені 3 *p-n*-переходи і анод верхнього по схемі сполучений з джерелом живлення через резистор *R1*. Всі ці *p-n*-переходи відкриті і напруга в крапці А дорівнює 0,7 × 3 = 2,1 В. Відкриті та насищені транзистори *VT2* і *VT4*. Напруга в точці *E,* тобто на виході елемента дорівнює ~ 0,1 В, що відповідає рівню логічного 0. Напруга в точці *D* дорівнює 0,7 В, а в точці *C* – 0,8 В (ділянка колектор–емітер транзистора *VT*2 і база–емітер *VT*4). Між точками *C* і *E* існує два *p-n*-переходи – эмитерний транзистора *VT*3 і діодний *VD*. Напруга між точками *C* і *E* дорівнює 0,8 – 0,1 = 0,7 В і є недостатньою для відкриття цих *p-n*-переходів. Отже, транзистор *VT*3 закритий, а діод *VD* необхідний для його надійного закриття.

Таким чином, наведена на рис. 2.61 схема виконує функцію логічного елемента І-НІ. Напруги у вузлових точках при різних рівнях вхідних напруг зведені до табл. 2.10.

Таблиця 2.10

x1	x2	A	B	C	D	E
0 0 1	0 1 0	0,7 – 1,1	0	– 5	0	3,6
1	1	2,1	1,4	0,8	0,7	0,1

Якщо входи елемента залишити вільними і не підключати до дже-

рела сигналу, то це буде сприйнято елементом як наявність логічних одиниць на його входах. Тому у багатьох випадках, коли на вхід повинен постійно подаватися сигнал рівня логічної 1, його нікуди не підключають. Проте для отримання від логічного елемента максимальної швидкодії рекомендується такий вхід підключати до плюса джерела живлення через резистор з опором 1 – 2 кілооми. До одного такого резистора можуть бути підключені відразу декілька входів.

У результаті розгляду процесу функціонування можна зробити такі висновки:

– базовий елемент ТТЛ містить багатоемітерний транзистор, що виконує логічну операцію І, і складний інвертор;

– якщо на один або обидва входи одночасно поданий низький рівень напруги, то багатоемітерний транзистор знаходиться в стані насичення і транзистор VT2 закритий, а отже, закритий і транзистор VT4, тобто на виході буде високий рівень напруги;

– якщо на обох входах одночасно діє високий рівень напруги, то транзистор VT2 відкривається і входить у режим насичення, що приводить до відкриття і насичення транзистора VT4 і замикання транзистора VT3, тобто реалізується функція І-НІ.

Для збільшення швидкодії елементів ТТЛ використовуються транзистори з діодами Шоттки (транзистори Шоттки).

Базовий логічний елемент ТТЛШ (на прикладі серії К555). Як базовий елемент серії мікросхем К555 використаний елемент І-НІ. На рис. 2.62, *а* зображена схема цього елемента, а умовне графічне позначення транзистора Шоттки наведене на рис. 2.62, *б*. Такий транзистор еквівалентний розглянутій вище парі, яка складається із звичайного транзистора і діода Шоттки. Транзистор VT4 – звичайний біполярний транзистор.

Якщо обидві вхідні напруги $u_{вх1}$ і $u_{вх2}$ мають високий рівень, то діоди VD3 і VD4 закриті, транзистори VT1, VT5 відкриті і на виході має місце напруга низького рівня. Якщо хоча б на одному вході є напруга низького рівня, то транзистори VT1 і VT5 закриті, а транзистори VT3 і VT4 відкриті, і на вході має місце напруга низького рівня. Слід відзначити, що транзистори VT3 і VT4 утворюють так званий складений транзистор (схему Дарлінгтона).

Мікросхеми ТТЛШ серії К555 характеризуються такими параметрами:

- напруга живлення +5 В;
- вихідна напруга низького рівня – не більше 0,4 В;

- вихідна напруга високого рівня – не менше 2,5 В;
- перешкодостійкість – не менше 0,3 В;
- середній час затримки розповсюдження сигналу – 20 нс;
- максимальна робоча частота – 25 МГц.

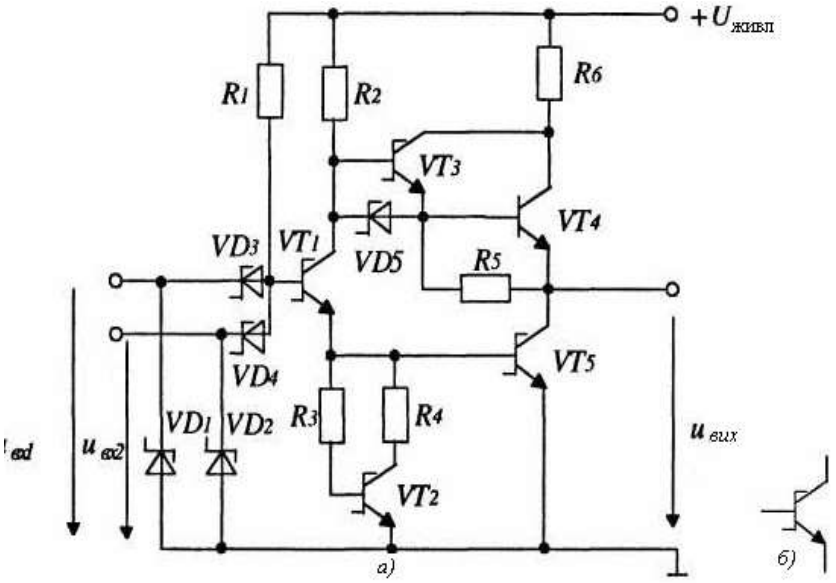


Рис. 2.62. Схема базового елемента ТТЛШ

Мікросхеми ТТЛШ сумісні за логічними рівнями, перешкодостійкостю і напругою живлення з мікросхемами ТТЛ. Час затримки розповсюдження сигналу елементів ТТЛШ у середньому в два рази менший порівняно з аналогічними елементами ТТЛ.

Серед загальних **переваг** ТТЛ-елементів можна виділити: схемно-технологічну відпрацьованість і як наслідок – високий відсоток виходу схем; низьку вартість, широкий функціональний набір логічних елементів; високу перешкодостійкість; велику здатність навантаження при прийнятних динамічних параметрах.

До **недоліків** ТТЛ-елементів можна віднести: наявність стрибків струму по колах живлення і землі при перемиканні ТТЛ-елементів, що, у свою чергу, накладає обмеження на максимальну тривалість фронтів вхідних сигналів (для серії К155 ця величина складає 150 нс); навіть

короткочасне замикання виходів на шину живлення може призвести до виходу з ладу ТТЛ-елемента; не для всіх схем ТТЛ можна реалізувати монтажну логіку (для цих цілей розроблені модифікації ТТЛ-елементів з відкритим колектором і з трьома станами виходу).

Особливості інших логік

Основою базового логічного елемента **емітерно-зв'язаної логіки** (ЕЗЛ) є струмовий ключ. Схема струмового ключа (рис. 2.63) є подібною до схеми диференціального підсилювача. Необхідно звернути увагу на те, що мікросхеми ЕЗЛ живляться негативною напругою (наприклад – 4,5 В для серії К1500). На базу транзистора VT2 подано негативну постійну опорну напругу $U_{оп.}$ Зміна вхідної напруги $u_{вх1}$ приводить до перерозподілу постійного струму I_{e0}, заданого опором R_e, між транзисторами, що є результатом зміни напруг на їх колекторах. Транзистори не входять в режим насичення, і це є однією з причин високої швидкодії елементів ЕЗЛ. Мікросхеми серій 100, 500 мають такі параметри: напруга живлення – 5,2 В; споживана потужність – 100 мВт; коефіцієнт розгалуження по виходу – 15; затримка розповсюдження сигналу – 2,9 нс. Принципова схема базового елемента серії 500 наведена на рис. 2.64, *а*, а функціональне позначення – на рис. 2.64, *б*. Даний елемент виконує функцію логічного складання і логічного складання з інверсією (АБО, АБО-НІ). Наявність парафазних виходів істотно збільшує функціональні можливості ЕЗЛ-схем.

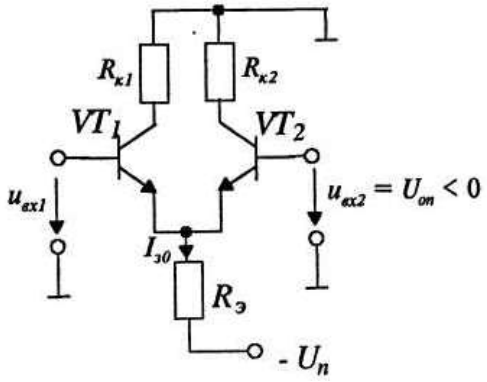


Рис. 2.63. Схема струмового ключа для ЕЗЛ

Для досягнення ще більших логічних можливостей ЕЗЛ-елементів, зокрема для реалізації монтажного АБО по виходах, виводи емітерів транзисторів VT5 і VT6 роблять ізольованими, що дозволяє об'єднувати декілька емітерів різних логічних елементів і підключати до ним тільки один зовнішній резистор навантаження.

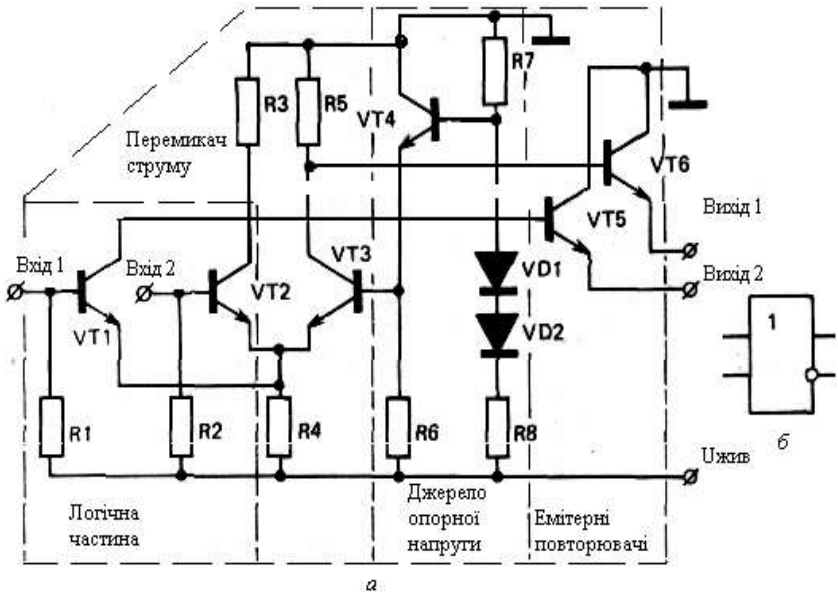


Рис. 2.64. Принципова схема базового елемента серії 500

Резистори навантажень виконуються як набір резисторів в тих же корпусах, що і логічні елементи.

Висока швидкодія ЕЗЛ-елементів забезпечується трьома основними чинниками:

- ❖ активним режимом роботи транзисторів в обох логічних станах елемента, завдяки чому усуваються час накопичення і розсмоктування надмірних зарядів;
- ❖ використанням низького значення логічного перепаду, завдяки чому зменшується час заряду і розряду власних складових ємностей схеми;
- ❖ використанням на виходах елемента емітерних повторювачів, що забезпечують значну величину струму для перезаряду складової, ємності навантаження.

Мікросхеми на основі ЕЗЛ-схем мають ряд **переваг**, серед яких всі переваги розглянутих вище ТТЛ-схем, а також:

- ➢ висока швидкодія при середній споживаній потужності або надвисока швидкодія при великій споживаній потужності;
- ➢ висока стабільність динамічних параметрів при зміні темпера-

тури і напруги живлення;

- здатність працювати на низькоомні узгоджені лінії зв'язку;
- можливість реалізації монтажної логіки;
- застосування дво- і трирівневого перемикання струму для ще більшого розширення логічних можливостей ЕЗЛ-елемента;
- можливість конструктивного зниження рівня перешкод в системах, виконаних на ЕЗЛ-елементах.

До **недоліків** ЕЗЛ-елементів можна віднести:

- високу споживану потужність при субнаносекундних затримках;
- вужчий температурний діапазон роботи в порівнянні з ТТЛ-елементами;
- відносно велику площу, яку займає вентиль на кристалі ($> 10^4$ мкм^2);
- проблематично застосування ЕЗЛ-елементів в надвеликих інтегральних схемах ВВІС, хоча вони з успіхом використовуються у ВІС.

МОН-технології (Р-МОН, N-МОН і КМОП)

Переваги метал-окисел-напівпровідникових інтегральних схем (МОН ІС) були відомі із самого початку: процес їх виготовлення значно простіший, ніж для біполярних ІС, оскільки кількість необхідних технологічних операцій зменшувалася більш ніж у 2 рази; вони споживали набагато меншу потужність і, отже, допускали вищий рівень інтеграції, ніж біполярні прилади, і нарешті, їх виготовлення обходилося дешевше. Ці схеми проте не були позбавлені і недоліків: їх виробництву заважали дефекти в оксиді; вони були надзвичайно чутливі до статичних зарядів – невелике перенапруження могло пробити тонкий оксид і миттєво зруйнувати МОН-транзістор. Крім того, робочі напруги МОН ІС були значно вищі, ніж робочі напруги серій логічних біполярних інтегральних схем, що випускалися у той час; МОН ІС володіли значно меншою швидкодією, ніж біполярні схеми. На сьогодні більшість з цих недоліків усунена.

Канал МОН-транзістора є легованим кремнієм *p*- або *n*-типу, що визначає тим самим *p*-МОН і *n*-МОН-технологію. Не зважаючи на простоту виготовлення, р-МОН-технологія була повсюдно витиснена *n*-МОН-технологією завдяки вищій швидкодії. Виготівники МОН-прикладів виявили, що можна істотно підвищити швидкодію, якщо замінити *p*-канальні структури структурами з каналом *n*-типу. Цей перехід був складним, оскільки прецизійне легування у разі *n*-МОН-

структур здійснити набагато складніше, ніж при виготовленні *p*-канальних структур. Проте підвищення швидкодії було очевидним: носії в каналі *n*-типу (електрони) рухаються швидше, ніж носії у *p*-каналі (дірки).

У третій основній МОН-технології використовуються як *n*-, так і *p*-МОН-транзистори. Вона називається комплементарною МОН-технологією (КМОН). Починаючи зі структури з металевими затворами і малою щільністю упаковки елементів (що використалися в ІС наручного годинника), вони пройшли через етапи зменшення розмірів елементів, переходу на ізоляцію оксидом, заміни металевих затворів на кремнієві. Це дозволило збільшити швидкодію і щільність упаковки елементів, так що КМОН-структури досягли рівня *n*-МОН-структур.

Основна перевага всіх МОН-технологій полягає у відносній простоті виробничих процесів і високій щільності компоновки. Тому МОН-технології застосовувалися майже у всіх без виключення додатках ВІС, забезпечуючи тим самим випуск надзвичайно недорогих виробів (наприклад, мікропроцесорів і великих кристалів пам'яті).

У мікросхемах ***n-МОП*** **і** ***p-МОП*** використовуються ключі відповідно на МОН-транзисторах з *n*-каналом і динамічним навантаженням і на МОН-транзісторах з *p*-каналом.

Як приклад розглянемо елемент логіки *n-МОН*, що реалізовує функцію АБО-НІ (рис. 2.65).

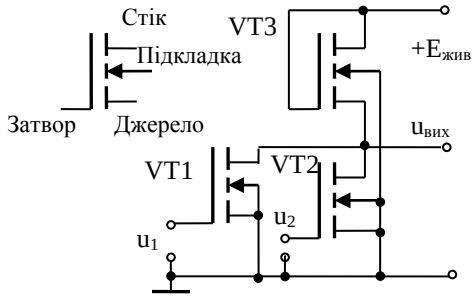


Рис. 2.65. Схема елемента АБО-НІ на *n-МОН* структурі

Він складається з транзистора, навантаження VT3 і двох транзисторів VT1 і VT2, що управляють. Якщо обидва транзистори VT1 і VT2 закриті, то на виході встановлюється високий рівень напруги. Якщо одна або обидві напруги u_1 і u_2 мають високий рівень, то відкривається один або обидва транзистори VT1 і VT2 і на вході встановлюється низький рівень напруги, тобто реалізується функція $u_{вих} = \overline{u_1 + u_2}$.

Для виключення споживання потужності логічним елементом у

статичному стані використовуються **комплементарні МДП** – логічні елементи (КМДН або КМОН-логіка). У мікросхемах КМОН використовуються комплементарні ключі на МОН-транзисторах.

Вони відрізняються високою перешкодостійкістю.

Логіка КМОН є дуже перспективною. Розглянутий раніше комплементарний ключ фактично є елементом НІ (інвертором).

Розглянемо КМОН логічний елемент, що реалізовує функцію АБО-НІ (рис. 2.66).

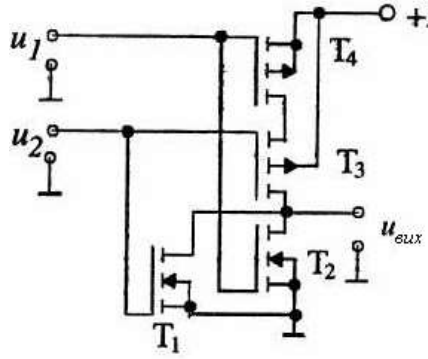


Рис. 2.66. Схема елемента АБО-НІ на *n-КМОН* структурі

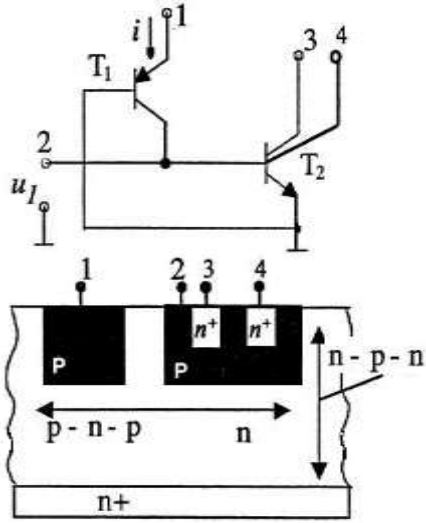
Якщо вхідні напруги мають низькі рівні *(u_1* і u_2 менше порогової напруги МОН-транзистора U_3), то транзистори VT1 і VT2 закриті, а транзистори VT3 і VT4 відкриті і вихідна напруга має високий рівень. Якщо одна або обидві вхідні напруги u_1 і u_2 мають високий рівень, що перевищує U_3, то відкриваються один або обидва транзистори VT1 і VT2, а між витоком і затвором одного або обох транзисторів VT3 і VT4 встановлюється низька напруга, що приводить до замикання одного або обох транзисторів VT3 і VT4, а отже, на виході встановлюється низька напруга. Таким чином, цей елемент реалізує функцію $u_{вих} = \overline{u_1 + u_2}$ і споживає потужність від джерела живлення лише в короткі проміжки часу, коли відбувається його перемикання.

Інтегральна інжекційна логіка (ІІЛ або $І^2Л$) побудована на використанні біполярних транзисторів і застосуванні оригінальних і технологічних рішень схемотехніки. Для неї характерне дуже економічне використання площі кристала напівпровідника. Елементи $І^2Л$ можуть бути реалізовані тільки в інтегральному виконанні і не мають аналогів в дискретній схемотехніці, Структура такого елемента і його еквівалентна схема наведені на рис. 2.67, з якого видно, що транзистор Т1 *(р-n-р)* розташований горизонтально, а багатоколекторний транзистор Т2 *(n-p-n)* розташований вертикально.

Транзистор Т1 виконує роль інжектора, що забезпечує надходження дірок з емітера транзистора Т1 (при подачі на нього позитивної напруги через обмежуючий резистор) у базу транзистора Т2.

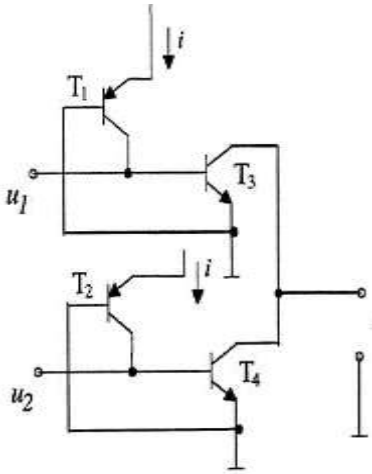
Якщо u_1 відповідає логічному 0, то інжекційний струм не протікає по базі багатоколекторного транзистора Т2 і струми в колах колекторів транзистора Т2 не протікають, тобто на виходах транзистора Т2 встановлюються логічні 1. При напрузі u_1, відповідній логічній 1, інжекційний струм протікає по базі транзистора Т2 і на виходах транзистора Т2 – логічні нулі.

Розглянемо реалізацію елемента АБО-НІ на основі елемента, наведеного на рис. 2.68 (для спрощення інші колектори багатоколекторних транзисторів Т3 і Т4 на не показані).



а

Рис. 2.67. Структура елемента $I^2Л$ і його еквівалентна схема



в

Рис. 2.68. Схема елемента АБО-НІ на $I^2Л$

Коли на один або обидва входи подається логічний сигнал “1”, та напруга $u_{вих}$ відповідає логічному нулю. Якщо на обох входах логічні сигнали “0”, то напруга $u_{вих}$ відповідає логічній одиниці.

Логіка на основі напівпровідника з **арсеніду галію GaAs** характеризується найбільш високою швидкодією, що є наслідком високої рухливості електронів (в 3...6 разів більше в порівнянні з

кремнієм). Мікросхеми на основі GaAs можуть працювати на частотах близько 10 ГГц і більше.

Прагнення до збільшення швидкодії таких схем і їх логічної гнучкості привело до розробки великої кількості інтегральних структур і різноманітних варіантів схемотехнік базового ключа.

Одним з найважливіших напрямів розвитку інжекційних схем стало використання в них діодів Шоттки. При цьому досягається збільшення швидкодії завдяки обмеженню ступеня насичення ключового *n-p-n* транзистора (шунтування переходу база-колектор), використанню як перехід база-колектор діода Шоттки (транзистор з металевим колектором), зменшенню логічних перепадів у схемі.

З використанням діодів Шоттки можливо збільшення функціональної логічної гнучкості – об'єднання входів за допомогою діодів Шоттки (логічна функція І).

До переваг $І^2$Л-елементів можна віднести:

- можливість роботи в широкому діапазоні струмів ($10^{-9} – 10^{-2}$ А);
- можливість збереження логічного стану $І^2$Л-схем переведенням їх в режим мікрострумів, коли вони не працюють у граничному частотному режимі або взагалі повинні знаходитися в неробочому стані;
- простоту розгалуження сигналу на виході за рахунок додавання колекторів у ключовому транзисторі;
- надзвичайно низьку потужність розсіяння і високу щільність компоновки, що робить її ідеальною для НВІС;
- можливість створення інжекційно-польових структур, для чого замінюють ключовий транзистор на польовий з вертикальним каналом (у цьому напрямі ведуться великі дослідження).

Основний **недолік $І^2$Л-элемента**: площі колекторів менше площі емітера, тобто транзистор працює в інверсному включенні. Це в значній мірі визначає технологію виготовлення $І^2$Л-елементів і створює основні труднощі при їх реалізації.

До недоліків інтегральних схем інжекційно-польової логіки слід віднести перш за все технологічні труднощі отримання і відтворення геометричних розмірів каналу при масовому виробництві схем. Не зважаючи на великі успіхи у області розробки ВІС і НВІС на $І^2$Л-елементах, розробникам і технологам належить вирішити ще багато складних завдань.

3. СХЕМОТЕХНІКА ЦИФРОВИХ ЕЛЕМЕНТІВ

3.1. Характеристика та класифікація цифрових елементів

Тригером (від англ. trigger – спусковий гачок) називають пристрій, який має два стійких стани і здатний стрибком переходити з одного стану в інший під дією зовнішнього керуючого сигналу. В загальному випадку тригер є ***запам'ятовуючим елементом***. Для переходу тригера з одного стійкого стану в інший треба, щоб вихідний сигнал переважував деяке порігове значення.

Основні області застосування тригера:

- ✓ комірка, що запам'ятовує, в пристроях електронної пам'яті ЕОМ;
- ✓ елемент ділення на 2 в імпульсних лічильниках і дільниках частоти;
- ✓ пристрій для розширення (збільшення тривалості) імпульсів;
- ✓ пристрій, який поновлює форму прямокутного імпульсу.

Тригерний пристрій є елементарним автоматом Мура, у якому є два вихідних сигнали. При цьому один вихідний сигнал Q (прямий вихід) має значення, тотожне стану тригера в момент часу, що розглядається, а другий вихідний сигнал $\bar{Q}$ (інверсний вихід) має значення, протилежне значенню першого вихідного сигналу. Тому принцип функціонування тригерного пристрою можна описати всіма способами, прийнятими для опису таких автоматів. На практиці для опису принципу функціонування тригерів широко використовують таблиці переходів і часові діаграми. При цьому таблиця переходів одночасно є і таблицею виходів для прямого виходу Q, оскільки новий стан тригера та його вихідний сигнал на прямому виході за значенням тотожні.

Прийнято позначати старий стан тригера через Q^n або Q(t), а новий стан – через Q^{n+1} або Q(t + 1) відповідно.

Для характеристики стану виходів і входів у таблиці переходів

приймаються такі позначення :

- 1 або 0 – якщо сигнал на відповідному вході або виході має значення, яке зіставляється з логічною одиницею або логічним нулем;
- н/в – при невизначеному стані тригера, тобто коли тригер може знаходитись із рівною ймовірністю або в стані “1”, або в стані “0”
- х – якщо немає значення, який сигнал (1 або 0) діє на вході тригера;
- 0/1 – якщо тригер спрацьовує при перепаді вхідного сигналу від 0 до 1;
- 1/0 – якщо тригер спрацьовує при перепаді вхідного сигналу від 1 до 0.

Класифікація тригерів. Найбільш загальними класифікаційними ознаками є функціональний спосіб і спосіб керування. Можна зазначити такі ***функціональні типи*** тригерів, які найчастіше зустрічаються на практиці:

З роздільною установкою станів “0” та “1” (**RS-тригери**). Тут S (Set – установка) – вхід для роздільної установки тригера в стан “1” ($Q = 1$, $\bar{Q} = 0$); R (Reset – скид) – вхід для роздільної установки тригера в стан “0” ($Q = 0$, $\bar{Q} = 1$).

З лічильним входом (**Т – тригери**). Тут Т (Toggle – релаксатор) – лічильний вхід тригера.

Універсальні з роздільною установкою станів “0” і “1” (**JK-тригери)**. Тут J (Jerk – раптове включення) – вхід для роздільної установки тригера в стан “1”, К (Kill – раптове відключення) – вхід для роздільної установки тригера в стан ″0″.

З прийомом інформації по одному входу (**D-тригери**). Тут D (Delay – затримка, Drive – передача) – інформаційний вхід для установки тригера в стан “0” або “1”.

Універсальні з керованим прийомом інформації по одному входу(**DV – тригери**). Тут V(Valve – клапан, вентиль) – керуючий вхід для дозволу прийому інформаційних або тактових сигналів.

Комбіновані (наприклад, RST-, JKRS-, DRS-тригери тощо).

Зі **складною вхідною логікою** (наприклад, коли декілька ідентичних входів пов′язані в групи операцій І, чи АБО).

Класифікація ***за способом запису*** інформації характеризує тимчасову діаграму роботи тригерів, тобто визначає хід процесу запису інформації в тригер.

За цією класифікацією тригери підрозділяються на дві групи:

- асинхронні (нетактовані);
- синхронні (тактовані).

У свою чергу синхронні тригери бувають зі ***статичним управлінням*** та з динамічним управлінням записом. У синхронних тригерах зі статичним управлінням записом тактовий імпульс починає впливати тільки тоді, коли його рівень або зростає до рівня "1", або зменшується до рівня "0" в залежності від елементної бази, на якій виконується тригер.

Специфіка синхронних тригерів зі статичним управлінням така, що протягом часу дії тактового імпульсу зміна сигналів на інформаційних входах веде до нових спрацьовувань тригера. В багатьох випадках ця властивість є недоліком, оскільки може виявитись причиною порушень у роботі. Від цього недоліку вільні тригери з динамічним управлінням.

Тригери з ***динамічним управлінням*** залежно від схеми виконання реагують на перепади напруги від "0" до "1" (активний фронт) або від "1" до "0" (активний зріз) керуючого імпульсу, тобто переключення тригера в новий стан відбувається тільки в момент часу, який збігається або з наростанням, або зі спадом фронту синхроімпульсу.

В інтегральній схемотехніці ***динамічне управління досягається шляхом побудови двоступінчастих тригерів,*** перехід яких у новий стан відбувається в момент часу, що збігається зі зрізом синхроімпульсу.

Асинхронний тригер також може бути зі статичним або динамічним управлінням залежно від того, як реагує тригер на вхідні інформаційні сигнали. Характерною особливістю асинхронних тригерів є те, що запис інформації в них здійснюється безпосередньо з надходженням на його вхід інформаційного сигналу.

Запис же інформації в тактованому тригері здійснюється тільки при подачі роздільних тактуючих імпульсів.

Для прямого динамічного С-входа використовуються позначення, які наведені на рис. 3.1, *а*, а для інверсного – на рис. 3.1, *б*.

Тактовані тригери зі статичним управлінням можуть бути однотактними і багататактними. Таким чином, у синхронних тригерах можна чітко виділити різні моменти часу – такти, під час яких відбуваються їх переключення.

Рис. 3.1. Зображення на схемах динамічних синхровходів:
а – по фронту, б – по зрізу

В асинхронних тригерах такі моменти часу виділити неможливо (інформаційні сигнали викликають переключення тригерів у моменти своєї зміни).

3.2. Синтез асинхронних тригерів. Синхронні тригери

3.2.1. RS -тригери

Асинхронні RS-тригери. RS-тригер – це тригер з роздільною установкою 0 та 1. RS (або SR)-тригери є найпростішим типом тригерів. Прикладом самостійного використання асинхронних RS-тригерів може бути використання їх як елементарних комірок пам′яті в оперативних запам′ятовуючих пристроях (ОЗП) статичного типу. Виконуються такі ОЗП у вигляді самостійних виробів або функціональних вузлів мікросхем підвищеного рівня інтеграції.

Відповідно до стану, який набуває тригер під дією вхідних сигналів, вхід S називають одиничним входом тригера, а вхід R – нульовим.

Асинхронні RS-тригери з роздільним управлінням по входах будуються на основі логічних елементів АБО-НІ чи І-НІ, вихід кожного з яких з’єднаний з одним із входів іншого. У найпростішому випадку для побудови тригерів використовуються двовходові логічні елементи. В результаті зазначеного з’єднання елементів у схемі вводиться позитивний зворотний зв’язок і створюються умови (вихідний транзистор однієї схеми закритий, а іншої – відкритий) виникнення двох стійких станів.

Функціонування тригера визначається таблицею переходів (табл. 3.1).

Цей тригер має чотири різні режими роботи:

1) зберігання інформації;
2) записи стану 0 (установка 0);
3) записи стану 1 (установка 1);
4) заборона вхідної дії.

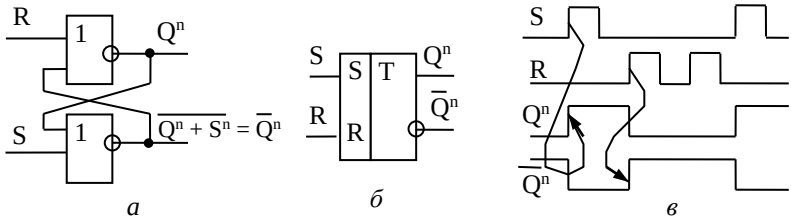
Таблиця 3.1

S	R	Q^n	Q^{n+1}	
0	0	0	0	Збереження
0	0	1	1	
0	1	0	0	Установка 0
0	1	1	0	
1	0	0	1	Установка 1
1	0	1	1	
1	1	0	н/в	Заборонені
1	1	1	н/в	

Запишемо констинуенти одиниці та мінімізуємо їх:

$$Q^{n+1} = \overline{S}\,\overline{R}Q^n + \overline{S}\overline{R}Q^n + S\overline{R}Q^n = \begin{vmatrix} 1-3 \\ 2-3 \end{vmatrix} = \overline{\overline{R}Q^n \cdot \overline{R}S} = R + S + Q^n.$$

Технічна реалізація отриманого виразу для Q^{n+1} на логічних елементах АБО-НІ, умовно графічне позначення такого RS-тригера та часові діаграми роботи асинхронного RS-тригера наведені на рис. 3.2.



а *б* *в*

Рис. 3.2. Схема, умовно графічне позначення та часові діаграми роботи асинхронного RS-тригера на елементах АБО-НІ

В основному полі умовно графічного позначення одно тактового тригера проставляється одна літера “T”.

Занесемо Q^{n+1} стани тригера до карти Карно та мінімізуємо їх:

S \ RQ^n	00	01	11	10
0	0	1	0	0
1	1	1	X	X

$$Q^{n+1} = S + \overline{R}Q^n$$

Для побудови RS-тригера на елементах І-НІ необхідно надати дві інверсії для функції функціонування RS-тригера:

$$Q^{n+1} = S + \overline{R}Q^n = \overline{\overline{S + \overline{R}Q^n}} = \overline{\overline{S}\ \overline{\overline{R}Q^n}}.$$

Технічна реалізація отриманого виразу для Q^{n+1} на логічних елементах І-НІ, умовно графічне позначення такого RS-тригера та часові діаграми роботи асинхронного RS-тригера на елементах І-НІ наведені на рис. 3.3.

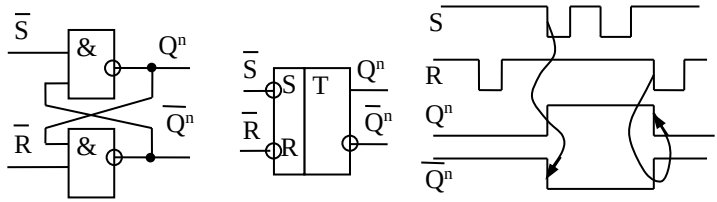


Рис. 3.3. Схема, умовно графічне позначення та часові діаграми роботи асинхронного RS-тригера на елементах І-НІ

Одним із застосувань RS-тригера з інверсними входами служить схема усунення "брязкоту" контактів клавіатури. Брязкотом контактів називається процес багатократного розмикання і замикання контактів при їх перемиканні. Схема і діаграми показані на рис. 3.4.

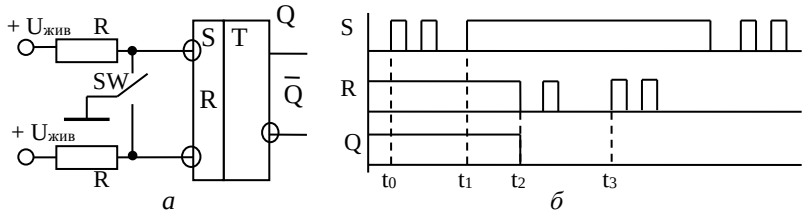


Рис. 3.4. Схема усунення брязкоту контактів та часові діаграми роботи асинхронного RS-тригера, побудованого на елементах І-НІ

У момент t_0 натиснення на клавішу починаються зіткнення верхнього і середнього контактів. До моменту t_1 сигнали S, R по черзі набувають значення 1, 1 і 0, 1, що відповідає режимам пам'яті і установки в 1. При цьому, початкове значення Q = 1 не зміниться, що і потрібно.

У інтервалі t_1 ... t_2 середній контакт знаходиться в "вільному польоті". Перше його торкання нижнього контакту в момент t_2 скине тригер (S = 1, R = 0). До моменту t_3 сигнали S, R по черзі набуває значення 1, 0 і 1, 1, що відповідає режимам скидання і пам'яті, тобто Q = 0. При відпусканні клавіші (момент t_3) розвивається зворотний процес. У ре-

зультаті дії схеми вихідний сигнал стає чистий від імпульсних перешкод.

Щоб мати можливість ***синхронізувати*** роботу як окремих вузлів і пристроїв, так і ЕОМ в цілому, використовують синхронні тригери, що відрізняються наявністю окрім інформаційних входів спеціального входу синхронізації С (Clock – час).

Простим представником синхронних тригерів є одноступінчатий синхронний RS-тригер (рис. 3.5).

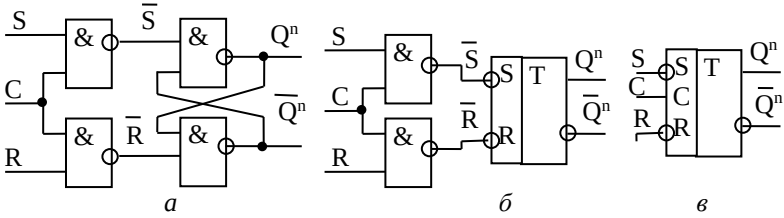


Рис. 3.5. Схеми та умовно графічне позначення синхронного RS-тригера

Даний тригер має статичне управління, тобто при активному сигналі синхронізації прийом інформації відбувається протягом всього інтервалу часу, поки С = 1. Зміна сигналів S, R протягом цього інтервалу небажано. Якщо при С = l S = l, R = 0, то в тригер запишеться одиниця. Якщо при С = 1, S = 0, R = 1, то в тригер запишеться нуль. Комбінація сигналів S = R = l при С = 1 є забороненою. При С = 0 або при S = R = 0 тригер зберігає інформацію.

Одноступінчаті тригери із статичним управлінням мають один ступінь запам'ятовування інформації. У них прийом (запис) і передача інформації на виходи схеми відбуваються одночасно. В результаті на час запису інформації можливо порушення інформаційного стану на виходах схеми. Щоб цього уникнути, використовують двоступінчаті тригери.

Двоступінчатий синхронний RS-тригер будується на базі двох послідовно сполучених одноступінчатих RS-тригерів зі спеціальною організацією ланцюга синхронізації (рис. 3.6).

При С = 1 здійснюється прийом інформації в перший тригер, а при С = 0 здійснюється передача інформації з першого тригера в другий і блокуються інформаційні входи першого тригера. Виходами всієї схеми є виходи другого тригера. Тому зміна стану тригера для зовнішніх схем відбувається у момент переходу синхросигналу з 1 в 0.

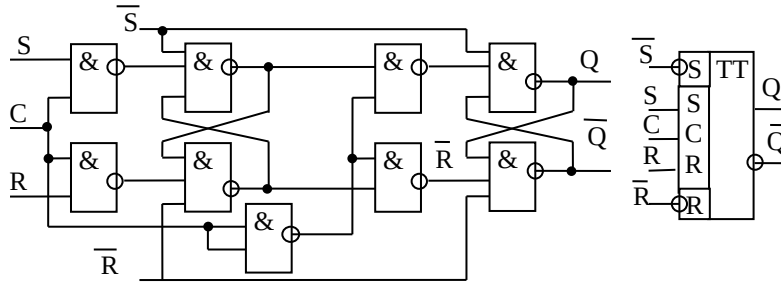


Рис. 3.6. Схеми та умовне графічне позначення синхронного двотактного RS-тригера

3.2.2. Т –тригери

На базі двоступінчатого RS-тригера шляхом замикання зворотних зв'язків будується асинхронний Т-тригер (Toggle – перекидатися), тобто тригер з рахунковим входом (рис. 3.7, *а*), таблиця переходів якого представлена табл. 3.2.

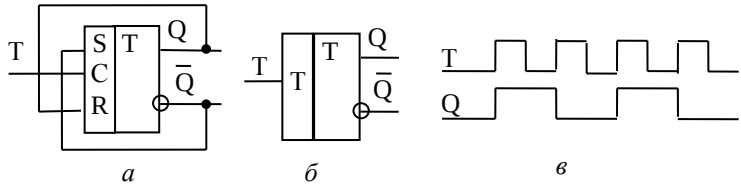


Рис. 3.7. Схема, умовне графічне позначення та часові діаграми роботи однотактового асинхронного Т-тригера

З приходом кожного імпульсу на рахунковий вхід тригер міняє свій стан на протилежний.

Т-тригером називається тригер, який за кожним одиничним сигналом змінює свій стан на протилежний. Т-тригер – єдиний вид тригера, стан якого визначається не інформацією на входах, а станом його в попередньому такті.

Таблиця 3.2

T	Q^n	Q^{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

З табл. 3.2 визначається рівняння Т-тригера:

$$Q^{n+1} = \overline{T}Q^n + T\overline{Q}^n.$$

Синхронний Т-тригер (рис. 3.8) рахує імпульси, що надходять на вхід синхронізації при подачі на вхід Т активного рівня (Т = 1).

Схеми взаємного перетворення тригерів у Т-тригер наведені на рис. 3.9.

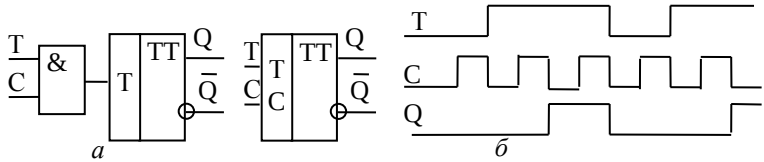


Рис. 3.8. Схема, умовне графічне позначення та часові діаграми роботи двотактового синхронного Т-тригера

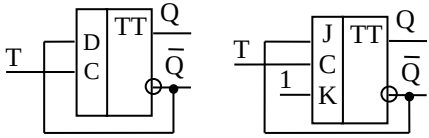


Рис. 3.9. Схеми інших варіантів побудови Т-тригерів

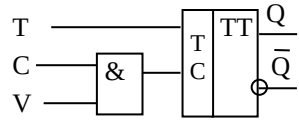


Рис. 3.10. Схема перетворення Т-тригера в TV-тригер

Для додаткового дозволу запису інформації використовують вхід V. При V = 1 тригеру дозволено приймати інформацію, а при V = 0 – ні (рис. 3.10).

3.2.3. D-тригер

При цифровій техніці широке застосування знаходять D-тригери, які приймають інформацію з входу D (Delay – затримка) за сигналом синхронізації С.

D-тригер – це тригер, який зберігає інформацію, що надійшла на його вхід. Несинхронний D-тригер не використовується, тому що на його виході буде просто повторюватись вхідний сигнал. Синхронний однотактний D-тригер реалізує затримку вхідного сигналу на час паузи між синхросигналами. Для затримки на період (на один такт) використовується двотактний D-тригер, тобто $Q^{n+1} = D$.

Таблиця станів наведена в табл. 3.3.

Таблиця 3.3

C	D	Q^n	Q^{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Занесемо стани Q^{n+1} тригера до карти Карно та мінімізуємо їх:

C \ DQ^n	00	01	11	10
0	0	1	1	0
1	0	0	1	1

Здійснимо перетворення отриманого виразу:

$$Q^{n+1} = \overline{C}Q + CD + DQ = CD + Q(\overline{C} + D)(\overline{C} + C) = CD + Q(\overline{C} + D\overline{C} + DC) =$$

$$= CD + Q(\overline{C} + DC) = \overline{CD + Q\overline{C} + QDC} = \overline{\overline{CD}\ \overline{Q\overline{C}}\ \overline{QDC}}.$$

Отриманій формулі відповідає схема на рис. 3.11.

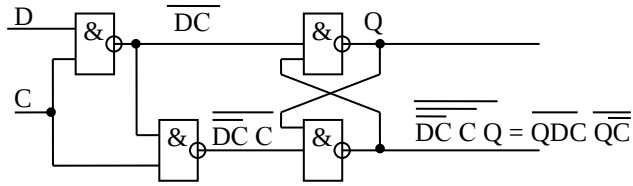


Рис. 3.11. Схема синхронного однотактового D-тригера, виконаного на елементах І-НІ

D-тригер із статичним управлінням (рис. 3.12, *а*) приймає інформацію з входу D протягом всього інтервалу, поки C = 1 (активний). Зміна сигналу D в цей час є небажаною.

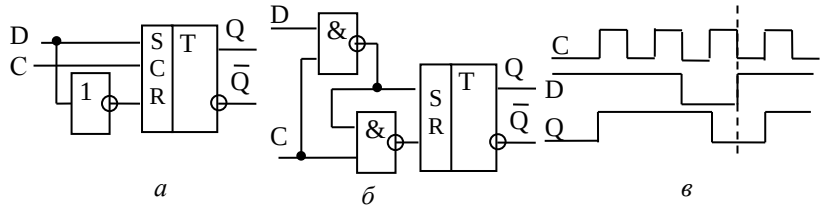


Рис. 3.12. Схема, умовне графічне позначення та часові діаграми роботи однотактового синхронного D-тригера

D-тригер може мати додатковий вхід V. У такому випадку такий тригер називається DV-тригером.

При V = 1 тригер функціонує як D-тригер, при V = 0 він переходить у режим зберігання інформації незалежно від зміни сигналів на входах D і C. Наявність входу V розширює функціональні можливості D-тригера, дозволяючи в потрібні моменти часу зберігати інформацію на виходах протягом кількості тактів, яка вимагається.

У DV-тригер можна перетворити кожний тактований D-тригер зі статичним, динамічним чи двоступінчатим управлінням, додавши вхід V та логічно пов'язавши його операцією І з керуючим входом C.

Як приклад на рис. 3.13 показані функціональна схема DV-тригера зі статичним управлінням та його графічне позначення.

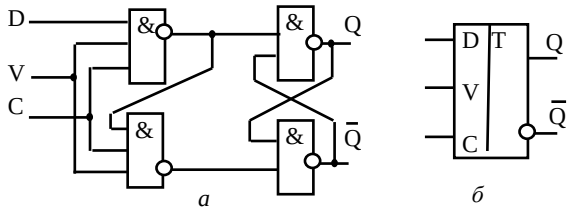


Рис. 3.13. DV-тригер зі статичним записом інформації:
а – функціональна схема, *б* – умовне графічне позначення

Усі можливі переходи в DV-тригері представлені в табл. 3.4, а узагальнені дані переходів наведені в табл. 3.5.

Таблиця 3.4
Повна таблиця переходів DV-тригера

t_n			t_{n+1}
V	D^n	Q^n	Q^{n+1}
0	0	0	Q^n
0	0	1	Q^n
0	1	0	Q^n
0	1	1	Q^n
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Таблиця 3.5
Узагальнена по входах таблиця переходів DV-тригера

t_n		t_{n+1}
V	D^n	Q^{n+1}
1	0	0
1	1	1
0	x	Q^n

DV, як і JK-тригери, широко застосовуються в пристроях запам'ятовування двійкової інформації, розрядах регістрів, лічильниках та інших вузлах цифрової техніки.

3.3.4. JK-тригер

При створенні цифрових пристроїв різного призначення можуть бути потрібними тригери всіх розглянутих вище трьох типів, тому зручно виготовити один тип універсального тригера, який можна було б використовувати як RS-, T- та D-тригер. До універсального тригера відноситься JK-тригер.

Найбільше застосування знайшов JK-тригер (рис. 3.14, а). J – вхід установки тригера в одиничний стан (Jark – раптове включення), K – вхід установки тригера в нульовий стан (Kill – раптове відключення).

Тригери цього типу відрізняються від RS-тригерів тим, що при значеннях вхідної інформації, забороненої для RS-тригерів (R = S = 1), вони інвертують інформацію, яка зберігається в них. Для того, щоб побудувати однотактний JK-тригер необхідно між елементами 2I та RS-тригером (рис. 3.13, *а*) вставити лінії затримки. Це необхідно для надійного встановлення в вихідні стани. У зв'язку з неоднорідністю такої структури однотактні JK-тригера практичного використання не зазнали.

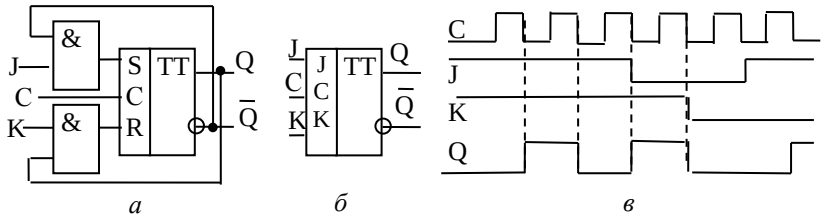


Рис. 3.14. Схема, умовне графічне позначення та часові діаграми роботи двотактового синхронного JK-тригера

Функціонування тригера описується таблицею переходів (табл. 3.6).

Таблиця 3.6

J	K	Q^n	Q^{n+1}	Примітка
0	0	-	Q^n	Збереження
0	1	-	0	Установка 0
1	0	-	1	Установка 1
1	1	-	інвQ^n	Режим Т-тригера

Рівняння функціонування JK-тригера записується виходячи зі схеми з'єднань вихідних та вхідних сигналів:

$$Q^{n+1} = J\overline{Q}^n + \overline{K}Q^n.$$

При J = K = 1 реалізується функція Т-тригера. Тому на базі JK-тригера легко реалізується синхронний Т-тригер шляхом об'єднання входів J і K та використання їх як вхід Т. Схеми використання JK-тригера наведені на рис. 3.15.

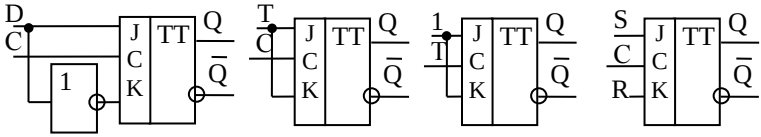


Рис. 3.15. Схеми використання JK-тригера

3.3. Завдання до самостійних досліджень функціонування тригерів

1. *Мета*

Побудова та ознайомлення з роботою основних схем тригерів за допомогою інструментальних засобів цифрової частини пакета EWB, закріплення теоретичного матеріалу, набуття навиків створення і моделювання комбінаційних цифрових пристроїв.

2. *Теми для попереднього опрацювання*

2.1. Синхронні та асинхронні тригери

3. *Завдання*

3.1. Дослід 1. Дослідження роботи тригерів у статичному режимі

3.1.1. Зібрати схему тригера на логічних елементах І, АБО, НІ згідно з варіантом (табл. 3.7) у пакеті EWB 5.12 та провести його дослідження.

У зв'язку з тим, що програма EWB програмно емулює функціо-

нування дискретних логічних елементів, то при дослідженні тригерів вона вносить деякі помилки – робить самозбудження тригерів. Тому доцільно використати дискретні логічні елементи, які виконані у вигляді закінчених мікросхем, наприклад, на МС 7400 (4 елементи 2І-НІ). Наприклад, схема D-тригера на основі МС 7400 наведена на рис. 3.16.

При дослідженні такого тригера доцільно не використовувати початковий стан генератора слів, який дорівнює 0000, тому що програма EWB також самозбуджується. Тому початковий стан взято 0002.

3.2. Зібрати тригер заданого типу на основі базових тригерів та необхідних логічних елементів.

Таблиця 3.7

№	Завдання	№	Завдання
1	RS-тригер однотактний асинхронний	9	D-тригер двотактний синхронний
2	T-однотактний асинхронний	10	DV-тригер однотактний синхронний
3	JK-тригер	11	RS-тригер однотактний синхронний
4	D-тригер однотактний синхронний	12	T-однотактний синхронний
5	TV-тригер однотактний асинхронний	13	D-однотактний синхронний
6	DV-тригер двотактний синхронний	14	T-двотактний синхронний
7	RS-тригер двотактний синхронний	15	T-двотактний асинхронний
8	TV-двотактний синхронний		

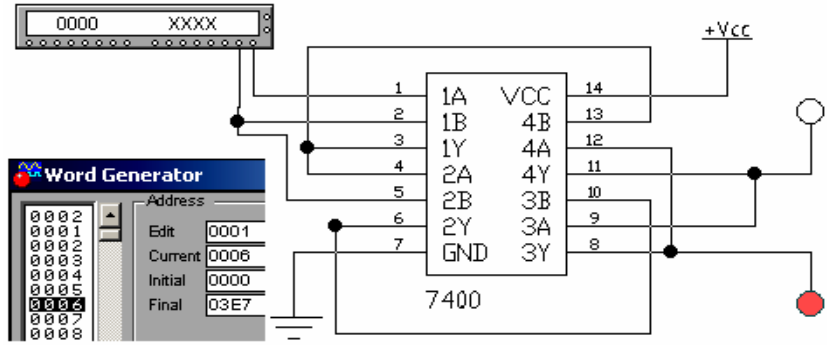


Рис. 3.16. Схема дослідження D-тригера в статичному режимі

Варіант завдання вибрати з табл. 3.8.

Навести зібрану схему та часові діаграми її дослідження.

Схема дослідження мікросхеми 7474 наведена на рис. 3.17.

Пряма статична синхронізація – це спрацювання тригера по фронту сигналу, а зворотна – по зрізу.

Тип базового тригера, який наведено в табл. 3.8:

1 – МС 555ТР2 (74279) – 4 RS-тригера-клямки (защелка – рос.);

2 – МС 555ТМ2 (7474) – 2 D-тригера, виводи якого мають призначення: CLR, PRE – R та S входи(інверсні); CLK – тактовий вхід;

3 – МС 555ТМ5 (7477) – 4 D-тригера із прямими виходами;

4 – МС 555ТВ1 (7472) – JK-тригер із елементом 3І на входах, який має 3-входові елементи J та K входи, позначені відповідно J1, J2, J3 та K1, K2, K3;

5 – МС 555ТВ6 (74107) – 2 JK-тригера із роздільною установкою нуля.

Таблиця 3.8

№	Тип тригера	Синхронізація	Кільк. МС	Тип базового Тг
1	RS	Пряма статична	2	1
2	D	Зворотна статична	1	1
3	T	Зворотна статична	2	1
4	TV	Пряма статична	2	2
5	DV	Зворотна статична	2	3
6	D	Зворотна статична	2	4
7	D	Зворотна статична	2	5
8	RS	Пряма статична	2	5
9	T	Зворотна статична	2	2
10	D	Пряма статична	1	1
11	JK	Зворотна статична	1	1
12	DV	Пряма статична	1	1
13	RS	Пряма статична	1	1
14	DV	Зворотна статична	2	2
15	T	Зворотна статична	2	3

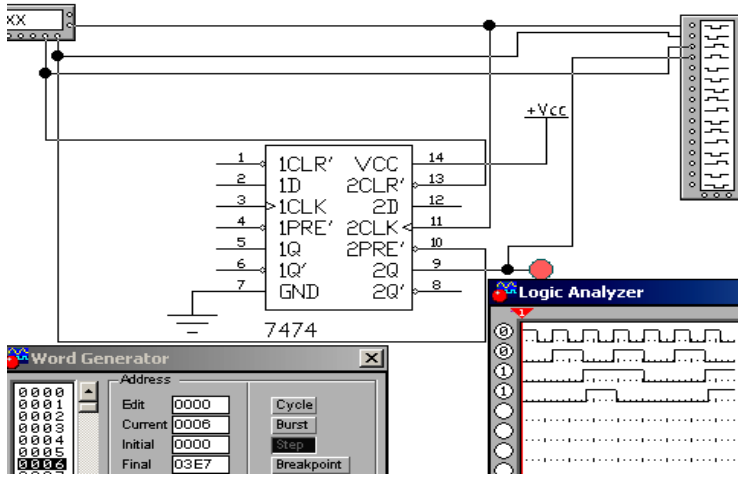


Рис. 3.17. Схема дослідження мікросхеми 7474

4. *Питання для самоперевірки*

4.1. Які типи тригерів ви знаєте та чим обумовлено їх різноманіття?

4.2. Наведіть функціональне призначення кожного з тригерів.

4.3. Приведіть таблиці виходів тригерів та рівняння їх функціонування.

4.4. Обґрунтуйте використання V-входів у тригерах.

4.5. Поясніть, чому на практиці не використовуються однотактні JK-тригери?

4. СХЕМОТЕХНІКА КОМБІНАЦІЙНИХ ВУЗЛІВ

4.1. Типові комбінаційні вузли: шифратори, дешифратори, мультиплексори, демультиплексори, цифрові компаратори

4.1.1. Класифікація комбінаційних цифрових пристроїв

Типовими прийнято називати функціонально закінчені цифрові пристрої, які широко застосовуються в засобах обчислювальної техніки та зв'язку і випускаються у вигляді окремих інтегральних мікросхем.

Як правило типові цифрові пристрої реалізуються у вигляді схем середнього або великого ступеня інтеграції.

Типові комбінаційні цифрові пристрої можуть бути згруповані:

- за наявністю керуючих входів;
- за типом входів (виходів);
- за функціями, які реалізуються (виконуваними операціями);
- за можливістю побудови (програмування) функцій, які реалізуються;
- за типом внутрішньої структури тощо.

За першою ознакою комбінаційні пристрої можуть бути поділені на дві групи: синхронні і асинхронні пристрої. Перші з них характеризуються наявністю спеціального керуючого входу (або групи керуючих входів), який визначає режим роботи пристрою, наприклад, дозволяє або блокує видачу інформації. Некеровані пристрої такого входу не мають.

За типом входів (виходів) комбінаційні пристрої поділяються на пристрої з прямими, інверсними або прямими та інверсними входами (виходами). Поділ пристроїв за цією ознакою визначається тим, у якому вигляді надходить (видається) інформація.

Найбільш істотною класифікаційною ознакою є тип операції – функція, яка реалізується типовим пристроєм. За цією ознакою серед комбінаційних пристроїв належить виділити:

- дешифратори;
- шифратори;

- перетворювачі кодів;
- мультиплексори;
- демультиплексори;
- комбінаційні суматори;
- порівнюючі пристрої;
- багатофункціональні елементи;
- пристрої контролю на парність;
- мажоритарні пристрої;
- комбінаційні помножувачі;
- схеми прискореного переносу тощо.

4.1.2. Дешифратори

Дешифратор – це функціональний вузол комбінаційного типу з n входами та 2^n виходами, який здійснює перетворення вхідного двійкового *n*-розрядного числа в сигнал 1 або 0 тільки на одному виході. Іншими словами – для перетворення вхідного позиційного коду в унітарний. **Унітарний код** – код, в якому тільки в одному розряді 1, а в інших – 0. Він відноситься до перетворювачів кодів. Двійкові дешифратори перетворюють двійковий код у код "1 із N". У кодовій комбінації цього коду тільки одна позиція зайнята одиницею, а всі інші є нульовими.

Призначення дешифраторів полягає в розпізнаванні наборів вхідних змінних, тобто в дешифрації кодових комбінацій, які надходять на входи, і формуванні сигналів на одному з виходів.

Класифікація дешифраторів зображена на рис. 4.1. Основними класифікаційними ознаками є кількість входів і виходів, принципи побудови і дії дешифраторів.

У залежності, від вхідного двійкового коду на виході дешифратора збуджується один і тільки один з вихідних ланцюгів. **Повним** називається дешифратор, кількість виходів якого *m* дорівнює 2^n. Якщо частина вхідних наборів не використовується, то дешифратор називають **неповним**. Дешифратор позначається літерами DC (від англійського Decoder). Входи дешифратора прийняті позначати двійковими вагами. Крім інформаційних входів дешифратор зазвичай має один або більш входів дозволу роботи, що позначаються як EN (Enable). При наявності дозволу по цьому вході дешифратор працює описаним вище чином, при його відсутності усі виходи дешифратора є пасивними. Якщо входів дозволу небагато, то сигнал дозволу роботи утвориться як

кон'юнкція сигналів окремих входів.

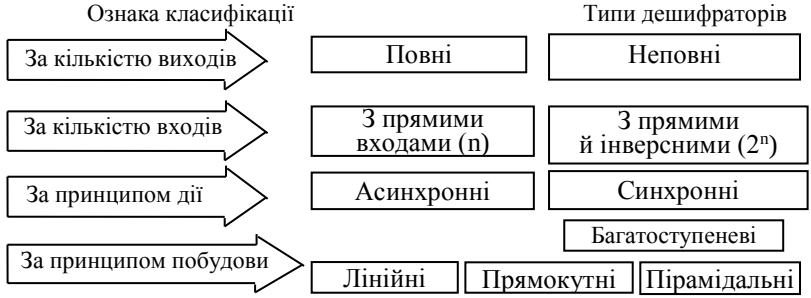


Рис. 4.1. Класифікація дешифраторів

Умовне графічне позначення повного двійкового дешифратора показане на рис. 4.2.

Часто дешифратор має інверсні виходи. У цьому випадку тільки один вихід мас нульове значення, а всі інші одиничне. При забороні роботи такого дешифратора на всіх його виходах будуть присутні логічні одиниці.

Як перетворювач позиційного коду в унітарний дешифратор має свою таблицю істинності (табл. 4.1 для числа входів n = 4).

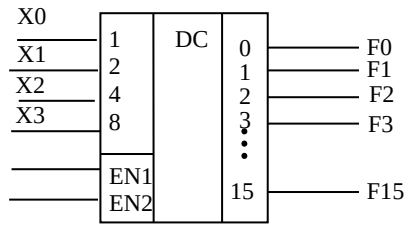


Рис. 4.2. Умовне графічне позначення дешифратора

Таблиця 4.1

X1	X2	X3	X4	F(i)	Унітарний код														
0	0	0	0	F(0)	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	F(1)	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	F(2)	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	F(3)	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	1	1	1	F(15)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

Функціонування дешифратора описується системою рівнянь:

$$F0 = \bar{x}0\,\bar{x}1\,\bar{x}2_{\ldots}\,\bar{x}_{n-2}\,\bar{x}_{n-1}\,EN$$

$$F1 = \bar{x}0\,\bar{x}1\,\bar{x}2_{\ldots}\,\bar{x}_{n-2}\,x_{n-1}\,EN$$

$$F2 = \bar{x}0\,\bar{x}1\,\bar{x}2_{\ldots}\,x_{n-2}\,\bar{x}_{n-1}\,EN$$

- - - - - - - - - - - - - - - - - - - -

$$F_{2^n-2} = \bar{x}0\,x1\,x2_{\ldots}\,x_{n-2}\,x_{n-1}\,EN$$

$$F_{2^n-1} = x0\,x1\,x2_{\ldots}\,x_{n-2}\,x_{n-1}\,EN$$

4.1.2.1. Схемотехнічна реалізація дешифраторів

Схемотехнічно дешифратор представляє собою сукупність кон'юнкторів (чи елементів І-НІ в дешифраторах з інверсними виходами), не зв'язаних між собою. Кожен кон'юнктор (чи елемент І-НІ) виробляє одну з вихідних функцій (рис. 4.3). Дешифратори такого типу не будуються на велику кількість входів. ***Основним обмеженням*** виступає коефіцієнт об'єднання по входу логічних елементів і коефіцієнт розгалуження по виходу для змінних.

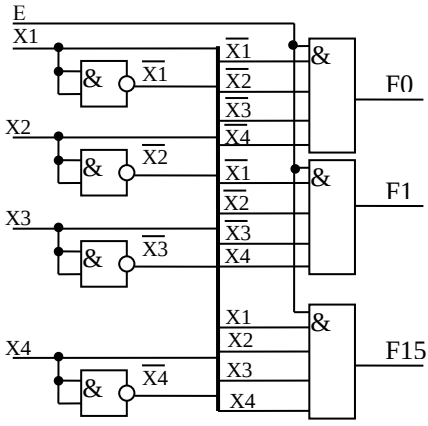


Рис. 4.3. Схема внутрішньої структури одноступеневого дешифратора

Дешифратори можуть також будуватися й на елементах АБО-НІ. Наприклад, для логічної функції одного із входів візьмемо подвійну інверсію:

$$F(0) = x1x2x3x4 = \overline{\overline{x1x2x3x4}} = \overline{\bar{x}1 + \bar{x}2 + \bar{x}3 + \bar{x}4}$$

Крім елементів, призначених для вироблення вихідних функцій, дешифратор має схеми для вироблення парафазних сигналів з однофазних (прямих), що надходять на входи ІС. Вхідна пряма змінна безпосередньо в схемі не використовується, а виробляється повторно як подвійна інверсія від вхідної. Таке

перетворення необхідно для того, щоб максимально розвантажувати лінії зовнішніх входів (тут зовнішні входи навантажені тільки на один вхід інвертора). Зовнішні лінії входів максимально розвантажуються завжди, оскільки вони і без того навантажені ємністю через свою велику довжину виводів корпуса ІС, що знижує швидкість передачі сигналу по лінії.

Дешифратор відноситься до числа швидкодіючих вузлів. Корпуса ІС із великою кількістю виводів виготовляти складно, і вони дуже коштовні. З цього погляду дешифратори відносяться до вкрай невдалих схем, тому що в них при простій внутрішній структурі і малій кількості схемних елементів багато зовнішніх виводів. Для розміщення у звичайному недорогому корпусі придатний тільки дешифратор з 4-ма інформаційними входами. Більш "розмірних" дешифраторів у серіях ІС немає.

4.1.2.2. Нарощування розмірності дешифратора

Малорозрядність стандартних дешифраторів порушує питання про нарощування їхньої розрядності.

На рис. 4.4. наведена схема реалізації дешифратора на 6-ти вхідних змінних з використанням лінійних дешифраторів з організацією (4 × 16).

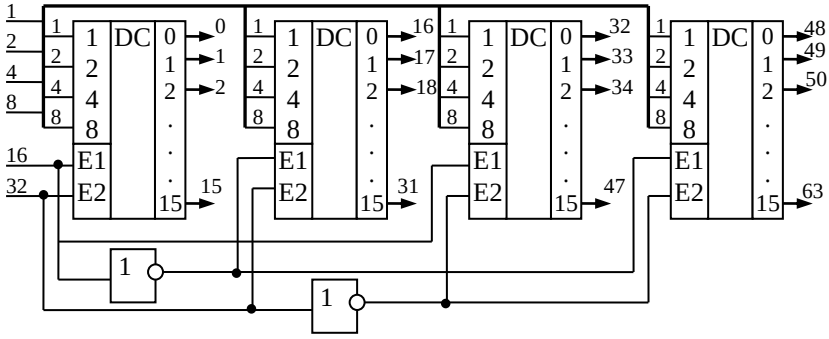


Рис. 4.4. Схема дешифратора з $n = 6$ на лінійних елементах

З малорозрядних дешифраторів можна побудувати схему, еквівалентну дешифратору більшої розрядності. Для цього вхідне слово по-

діляється на частини. Функціонування дешифратора з $n = 4$ можна описати такою системою рівнянь:

$$
\begin{array}{llll}
y0 = h0g0, & y4 = h1g0, & y8 = h2g0, & y12 = h3g0, \\
y1 = h0g1, & y5 = h1g1, & y9 = h2g1, & y13 = h3g1, \\
y2 = h0g2, & y6 = h1g2, & y10 = h2g2, & y14 = h3g2, \\
y3 = h0g3, & y7 = h1g3, & y11 = h2g3, & y15 = h3g3,
\end{array}
$$

де $h0 = a3a2$, $y0 = \overline{a1}\,\overline{a0}$,
$h1 = a3a2$, $y1 = a1\overline{a0}$,
$h2 = a3a2$, $y2 = a1a0$,
$h3 = a3a2$, $y3 = a1a0$.

Схема багатоступінчатого прямокутного дешифратора з $n = 4$ відповідно з отриманими виразами наведена на рис. 4.5.

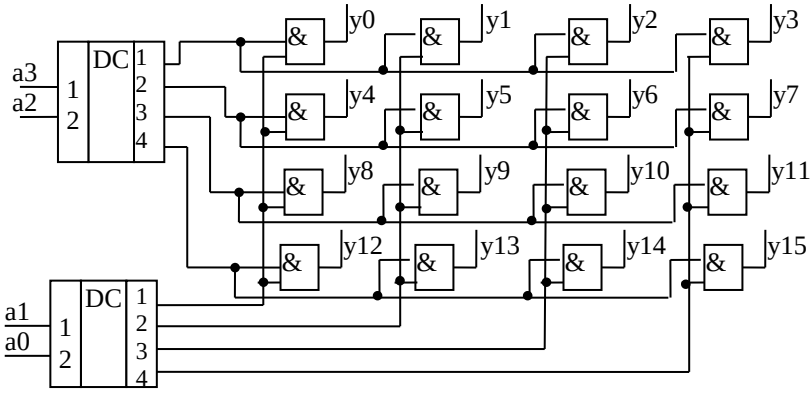


Рис. 4.5. Схема багатоступінчатого прямокутного дешифратора з $n = 4$

Структурна схема багатоступінчатого прямокутного дешифратора з $n = 13$ наведено на рис. 4.6. Як правило, вже при $n > 8$ будуються багатоступінчаті дешифратори.

На відміну від прямокутних багатоступінчатих дешифраторів находять використання й пірамідальні багатоступінчаті дешифратори, схема якого наведена на рис. 4.7.

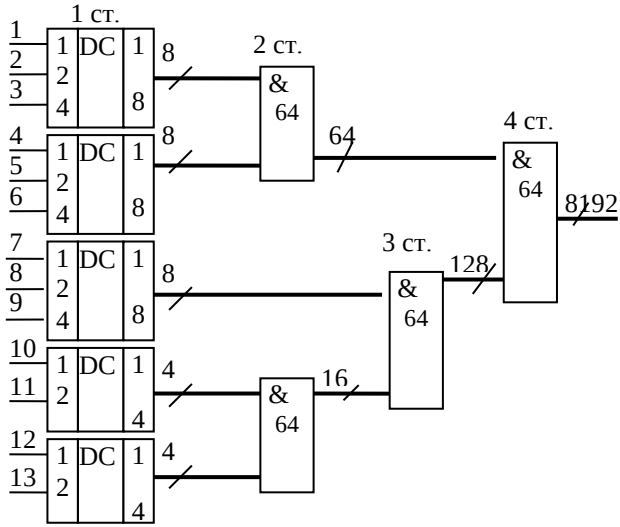


Рис. 4.6. Схема багатоступінчатого дешифратора з n = 13

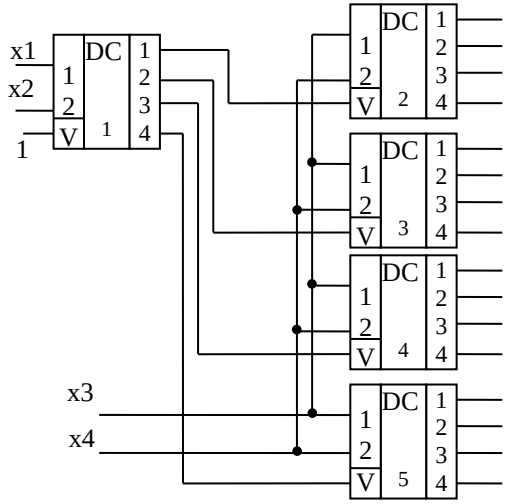


Рис. 4.7. Схема пірамідального багатоступінчатого дешифратора

4.1.3. Шифратори

Двійкові шифратори виконують операцію, зворотну стосовно операції дешифратора: вони перетворюють код “1 з N” у двійковий. У залежності від зміни унітарного входу шифратора на його виході формується двійковий код номера збудженої вхідної лінії. **Повний двійковий шифратор** має 2^n входів і n виходів. Наприклад, умовне графічне позначення шифратора з кількістю виходів $n = 3$ наведено на рис. 4.8. Таблиця істинності такого шифратора наведена в табл. 4.2.

Таблиця 4.2

a2	a1	a0	y_i
0	0	0	y0
0	0	1	y1
0	1	0	y2
0	1	1	y3
1	0	0	y4
1	0	1	y5
1	1	0	y6
1	1	1	y7

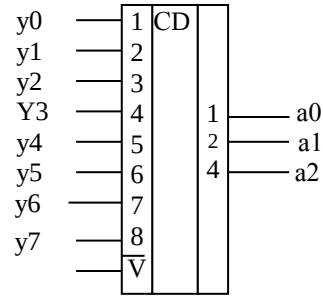


Рис. 4.8. Умовне графічне позначення шифратора

Функціонуванню шифратора з $n = 3$ відповідає система логічних функцій:

$$a0 = y1 + y3 + y5 + y7;$$
$$a1 = y2 + y3 + y6 + y7;$$
$$a2 = y4 + y5 = y6 + y7.$$

Пріоритетні шифратори виконують більш складну операцію. При роботі ЕОМ часто виникає задача визначення пріоритетного претендента на користування яким-небудь ресурсом. Деякі конкуренти виставляють свої запити на обслуговування, які не можуть бути виконані одночасно. Потрібно вибрати той запит, якому надається право першочергового обслуговування. Найпростіший варіант вирішення зазначеної задачі – присвоєння кожному джерелу запитів фіксованого пріоритету. Наприклад, група з восьми запитів R7...R0 (R від англійського Request) формується так, що вищий пріоритет має джерело номер сім, а далі пріоритет буде зменшуватися на одиницю від номера до номера.

Наймолодший пріоритет – у нульового джерела; він буде обслуговуватися тільки при відсутності всіх інших запитів. Якщо надійшли одночасно кілька запитів, обслуговується запит з найбільшим номером. Пріоритетний шифратор виробляє на виході двійковий номер старшого запиту. Легко бачити, що при наявності лише одного збудженого входу пріоритетний шифратор працює так само, як і двійковий. Тому в серіях елементів двійковий шифратор як самостійний елемент може бути відсутнім. Режим його роботи – окремий випадок роботи пріоритетного шифратора. Покажчики старшої одиниці вирішують ту ж задачу, що і пріоритетні шифратори, але виробляють результат в іншій формі – у вигляді коду "1 з N". ***Таким чином, при наявності на входах декількох збуджених ліній (запитів) на виході буде порушена лише одна, що відповідає старшому запиту.*** Кількість входів у цьому випадку дорівнює кількості виходів схеми. Покажчики старшої одиниці застосовуються в пристроях нормалізації чисел з точкою, що плаває, і т. д. У промислових серіях елементів існують шифратори пріоритету для восьмирозрядних і десятирозрядних слів. Функціонування їх відображається в табл. 4.3.

Таблиця 4.3

E1	R7	R6	R5	R4	R3	R2	R1	R0	a2	a1	a0	G	E0
1	1	X	X	X	X	X	X	X	1	1	1	1	0
1	0	1	X	X	X	X	X	X	1	1	0	1	0
1	0	0	1	X	X	X	X	X	1	0	1	1	0
1	0	0	0	1	X	X	X	X	1	0	0	1	0
1	0	0	0	0	1	X	X	X	1	1	1	1	0
1	0	0	0	0	0	1	X	X	0	1	0	1	0
1	0	0	0	0	0	0	1	X	0	0	1	1	0
1	0	0	0	0	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	X	X	X	X	X	X	X	X	0	0	0	0	0

Таблиця цілком характеризує роботу пріоритетного шифратора при всіх можливих комбінаціях сигналів: E1 – сигнал дозволу роботи даного шифратора; E0 – сигнал, який виробляється на виході даного шифратора при відсутності запитів на його входах для дозволу роботи наступного (молодшого) шифратора для нарощування розмірності шифраторів; G – сигнал, що визначає наявність запитів на вході даного шифратора; R7.....R0 – кількість запитів на входах шифратора; a2...a0 – значення розрядів вихідного двійкового коду, що формує номер стар-

шого запиту.

Усі зазначені сигнали формуються за умови E1 = 1 (робота шифратора дозволена). При E1 = 0 незалежно від станів входів запитів усі вихідні сигнали шифратора стають нульовими. З таблиці можна одержати такі вирази для функцій a2, a1, a0, E0, G.

4.1.4. Мультиплексори і демультиплексори

Мультиплексором називається комбінаційна схема, яка здійснює передачу сигналів із вхідних ліній у вихідну під керуванням адресного коду. Розрядності ліній можуть бути різними, мультиплексори для комутації багаторозрядних слів складаються з одноразрядних. Входи мультиплексора поділяються на дві групи: інформаційні та такі, що адресують. Роботу мультиплексора можна спрощено представити за допомогою багато-позиційного ключа.

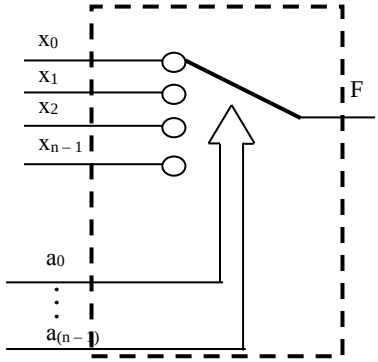


Рис. 4.9. Контактна модель мультиплексора

Для одноразрядного мультиплексора це наведено на рис. 4.9.

Адресний код задає перемикачу певне положення, з'єднуючи з виходом F один з інформаційних входів x_i. При нульовому адресному коді перемикач займає верхнє положення x_0, зі збільшенням коду на одиницю він переходить у сусіднє положення x_1 і т. д.

Робота мультиплексора описується співвідношенням

$$F = x_{n-1}\bar{a}_{n-1}\bar{a}_{n-2}\ldots\bar{a}_1\bar{a}_0 \vee \ldots \vee x_1\bar{a}_{n-1}\bar{a}_{n-2}\ldots\bar{a}_1\bar{a}_0 \vee x_0\bar{a}_{n-1}\bar{a}_{n-2}\ldots\bar{a}_1\bar{a}_0,$$

яке іноді називається мультиплексною формулою. Схемотехнічно мультиплексор реалізує електронну версію показаного перемикача, маючи, на відміну від нього, тільки однобічну передачу даних.

На рис. 4.10 наведено умовне графічне позначення мультиплексора. В основному полі схеми можуть використовуватись літери MS або MUX.

Функціонування 4-розрядного мультиплексора визначається таблицею істинності (табл. 4.4). На рис. 4.11 показаний мультиплексор з чотирма інформаційними входами, та двома адресними входами. У стандартних серіях розмірність мультиплексора не перевищує 16 × 1.

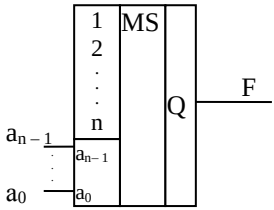


Рис. 4.10. Умовне позначення мультиплексора

Таблиця 4.4

a_1	a_0	Y
0	0	x0
0	1	x1
1	0	x2
1	1	x5

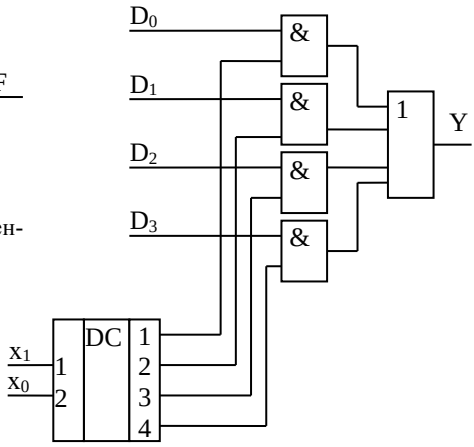


Рис. 4.11. Функціональна схема мультиплексора

Демультиплексори виконують операцію, зворотну операції мультиплексорів. Вони передають дані з одного вхідного каналу в один з декількох каналів-приймачів. Багаторозрядні демультиплексори складаються з декількох однорозрядних. Неважко помітити, що дешифратор зі входом дозволу роботи буде працювати в режимі демультиплексора, якщо на вхід дозволу подавати інформаційний сигнал.

Мультиплексори доцільно використовувати для реалізації логічних функцій. Нехай задана функція $y = \Sigma_0(0, 3, 4, 5)$. Необхідно її реалізувати. Варіант її реалізації наведено на рис. 4.12.

Нарощування розмірності мультиплексорів

Нарощування розмірності мультиплексорів можливо за допомогою пірамідальної структури з декількох мультиплексорів.

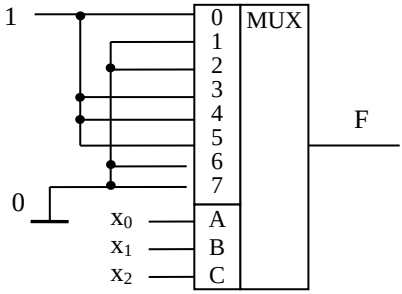


Рис. 4.12. Схема використання мультиплексора для реалізації логічної функції

При цьому перший ярус схеми має вигляд стовпця, в якому міститься стільки мультиплексорів, скільки необхідно для одержання потрібної кількості інформаційних входів. Усі мультиплексори стовпця адресуються тим самим кодом, складеним з відповідною кількістю молодших розрядів загального адресного коду (якщо кількість інформаційних входів схеми дорівнює 2^n, то загальна кількість адресних розрядів дорівнює n, молодше поле n_1 адресного коду використовується для адресації мультиплексорів першого ярусу). Старші розряди адресного коду, кількість яких дорівнює $n - n_i$ використовуються в другому ярусі, мультиплексор якого забезпечує почергову роботу мультиплексорів першого ярусу на загальний вихідний канал.

На рис. 4.13 наведена схема двокаскадного мультиплексора з організацією 32 × 1 на мультиплексорах з організацією 4 × 1.

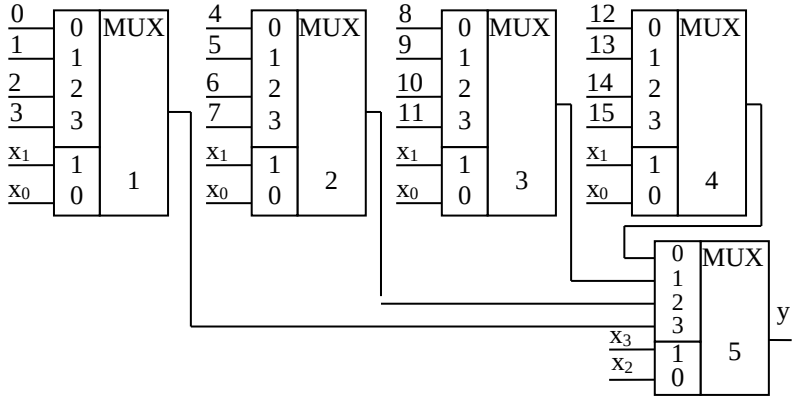


Рис. 4.13. Схема двокаскадного мультиплексора організації 16 × 1 з використанням тільки мультиплексорів організації 4 × 1

На рис. 4.14 наведена схема мультиплексора на 16 входів, який побудовано на 4-входових мультиплексорах.

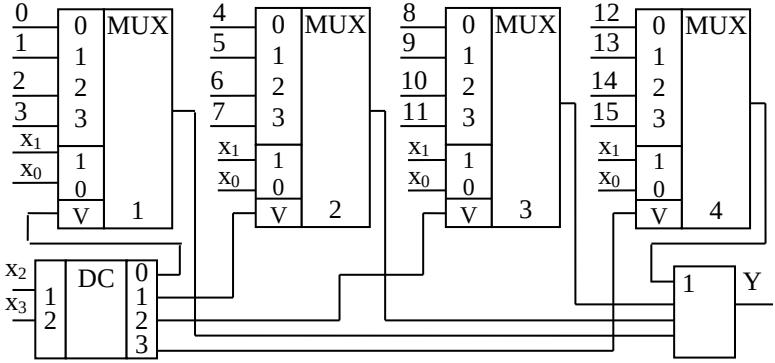


Рис. 4.14. Схема мультиплексора організації 16 × 1 на мультиплексорах організації 4 × 1 з використанням різних логічних елементів

В цій схемі для вибора відповідного мультиплексора використовується дешифратор.

4.1.5. Цифрові компаратори

Цифрові компаратори (пристрої порівняння) виконують операцію визначення відношення між двома словами. До основних відношень належать операції "дорівнює", "більше". Інші відношення можуть бути визначені через основні. Так, ознаку нерівності слів можливо отримати як заперечення ознаки рівняння ($F_{A \neq B} = \overline{F}_{A=B}$); відношення "менше" – шляхом заперечення ознаки ($F_{A<B} = \overline{F}_{A \geq B}$) , а нестрогі нерівняння – згідно з виразами

$$(F_{A \geq B} = F_{A=B} \vee F_{A>B} = \overline{F_{B>A}}),$$
$$(F_{A \leq B} = F_{A=B} \vee F_{A<B} = \overline{F_{A>B}}).$$

4.1.5.1. Пристрої порівняння на рівність

Ці пристрої будуються на основі порозрядних операцій над однаковими розрядами обох слів. Для однорозрядних слів функція $F_{A=B}$ визначається табл. 4.5.

Таблиця 4.5

a_i	b_i	$r_{i(a=b)}$
0	0	1
0	1	0
1	0	0
1	1	1

Ознака r рівності розрядів має одиничні значення, коли в обох розрядах є або одиниці, або нулі:

$$r_i = \overline{a_i}\,\overline{b_i} \vee a_i b_i = \overline{\overline{a_i}\,\overline{b_i} \vee a_i b_i} = \overline{a_i \oplus b_i}.$$

У свою чергу, ознака нерівності розрядів має вигляд:

$$r_i = \overline{a_i}b_i \vee a_i\overline{b_i} = \overline{\overline{a_i}b_i \vee a_i\overline{b_i}} = \overline{\overline{a_i}b_i}\;\overline{a_i\overline{b_i}} = (\overline{a_i} \vee b_i)(a_i \vee \overline{b_i}) = \overline{\overline{a_i}b_i \vee a_i\overline{b_i}} = \overline{a_i \oplus b_i}.$$

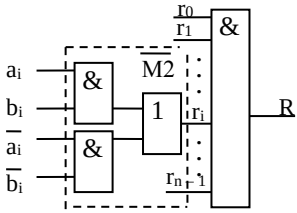


Рис. 4.15. Схема пристрою порівняння

Ознака рівняння R слів, у свою чергу, набуває одиничного значення, коли усі розряди дорівнюють один одному:

$$R = r_{n-1}\, r_{n-2} \ldots r_0.$$

Згідно з отриманими виразами для r_i та R будується схема, яка наведена на рис. 4.15.

4.1.5.2. Пристрій порівняння "на більше"

Для однорозрядних слів функція $F_{A>B}$ визначається табл. 4.6.

Таблиця 4.6

a_i	b_i	$F_{A>B}$
0	0	0
0	1	0
1	0	1
1	1	0

Функція $F_{A>B}$ для одного розряду з табл. 4.6 записується у вигляді:

$$F_{ai>bi} = a_i\overline{b_i}.$$

Функцію $F_{A>B}$ можна отримати виходячи з таких міркувань. Коли у старшому розряді слова А – одиниця, а слова В – нуль, то незалежно від молодших розрядів A > B та $F_{A>B} = 1$. Коли старші розряди ідентичні, то необхідно переходити до аналізу молодших розрядів, використавши ту ж умову, що й для старших розрядів. Таким чином, справедливо співвідношення:

$$F_{A>B} = a_1\overline{b_1} \vee a_0\overline{b_0}r_1,$$

де $r_1 = a_1 b_1 \vee \overline{a}_1 \overline{b}_1$ – ознака рівності розрядів.

Слід зазначити, що порівняння багаторозрядних слів повинно базуватися на тих же міркуваннях. Отже можна записати:

$$F_{A>B} = a_{n-1}\overline{b}_{n-1} \vee a_{n-2}\overline{b}_{n-2}r_{n-1} \vee a_{n-3}\overline{b}_{n-3}r_{n-2}r_{n-1} \vee \ldots \vee a_0\overline{b}_0 r_1 \ldots r_{n-1}.$$

Якщо розглядати пристрій порівняння на більше, то можна отримати більш прості вирази для функцій $F_{A>B}$. Відобразимо цю функцію на карті Карно:

a_1a_0 \ b_1b_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

Функція $F_{A>B}$ має вигляд:

$$F_{A>B} = a_1\overline{b}_1 \vee a_0\overline{b}_0\overline{b}_0 \vee a_1a_0\overline{b}_0 = a_1\overline{b}_1 \vee a_0\overline{b}_0(a_1 \vee \overline{b}_1).$$

Перехід до аналізу молодших розрядів виконується за умови, що стан старших розрядів не суперечить цьому переходу, але в даному випадку ця умова має вигляд: $a_1 \vee \overline{b}_1$.

Введемо позначення :

$$c_i = a_i\overline{b}_i,\ d_i = a_i \vee \overline{b}_i \text{ або } d_i = r_i.$$

Таким чином, вираз $F_{A>B}$ для порівняння багаторозрядних слів має вигляд

$$F_{A>B} = c_{n-1} + cn_{-2}d_{n-1} + \ldots c_i d_{n-1} \ldots d_{i+1} + \ldots c_0 d_{n-1} d_1.$$

4.1.5.3. Пристрій порівняння "на менше "

Провівши аналогічні міркування можна переконатися, що

$$F_{A<B} = \overline{a}_{n-1}b_{n-1} \vee \overline{a}_{n-2}b_{n-2}r_{n-1} \vee \overline{a}_{n-3}b_{n-3}r_{n-2}r_{n-1} \vee \ldots \vee \overline{a}_0 b_0 r_1 \ldots r_{n-1}.$$

Якщо відомі схеми для $F_{A=B}$ та $F_{A>B}$, то функцію $F_{A<B}$ легше реалізувати згідно з виразом

$$F_{A<B} = \overline{F}_{A=B}\,\overline{F}_{A>B} = \overline{F_{A=B} \vee F_{A>B}}.$$

4.1.5.4. Схемні рішення цифрових компараторів

У загальному випадку виходи компаратора відповідають трьом результатам порівняння: =, >, <.

Узагальнена таблиця істинності простішого однорозрядного компаратора має дві вхідні змінні та три вихідні функції $F_{A=B}$, $F_{A>B}$, $F_{A<B}$ (табл. 4.7).

Таблиця 4.7

A	B	$F_{A=B}$	$F_{A>B}$	$F_{A<B}$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

Згідно з табл. 4.7 одержуємо рівняння, що описують роботу пристрою:

$$F_{A=B} = A \oplus B,\ F_{A>B} = A\overline{B},\ F_{A<B} = \overline{A}B$$

Відповідно до цих рівнянь будується схема (рис. 4.16).

На рис. 4.17 зображений чотирирозрядний компаратор мікросхеми К1533СП1 з можливістю збільшення розрядності.

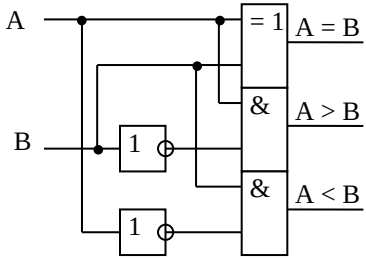


Рис. 4.16. Однорозрядний компаратор

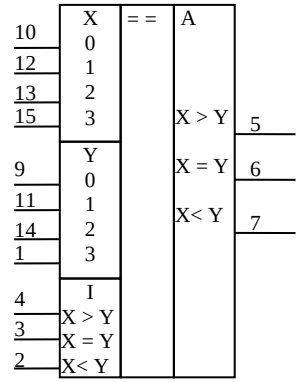


Рис. 4.17. Мікросхема К1533СП1 (чотирирозрядний компаратор)

Для нарощування розрядності в схемі передбачено три входи каскадування Y > X, Y = X, Y < X. При порівнянні чотирьохрозрядних чисел на вхід каскадування Y = X необхідно подати сигнал високого рівня. При більшій кількості розрядів використовують декілька мікросхем, сполучених послідовно: виходи мікросхем, що порівнюють молодші розряди чисел, з'єднуються з відповідними входами каскадування мікросхем, що порівнюють старші розряди.

4.1.5.5. Схеми контролю парності

Схема контролю парності аналізує кількість одиниць у двійковому числі. Сигнал парності формується на виході при парній кількості одиниць. Таблиця істинності схеми контролю парності зазвичай містить дві вихідні змінні Y1 та Y2, які відповідають парній і непарній кількості одиниць в числі. Кількість вхідних змінних відповідає розрядності схеми (табл. 4.8).

Таблиця 4.8

X2	X2	X3	Y1 (парність)	Y2 (непарність)
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Відповідно до таблиці логічна функція схеми контролю парності легко виражається за допомогою операції ВИКЛЮЧНЕ АБО:

$$Y1 = X1 \oplus X2 \oplus X3, \qquad Y2 = \overline{Y1}$$

Відповідно до таблиці логічна функція схеми контролю парності легко виражається за допомогою операції ВИКЛЮЧНЕ АБО:

$$Y1 = X1 \oplus X2 \oplus X3, \qquad Y2 = \overline{Y1}.$$

Типова восьмирозрядна схема контролю парності зображена на рис. 4.18. При OE = 0 схема здійснює контроль парності. При парному числі одиниць на входах DI на виходах PE = 1, P0 = 0. При OE=1 схема контролює непарність суми. В цьому випадку при парній кількості одиниць на входах DI маємо PE = 0, P0 = 1.

4.1.6. Завдання до самостійних досліджень мультиплексорів, дешифраторів та компараторів

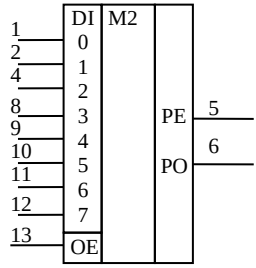


Рис. 4.18. Мікросхема К1533ИП5 (8-розрядна схема контролю парності)

1. *Мета*

Закріплення теоретичного матеріалу, набуття навиків створення і моделювання комбінаційних цифрових пристроїв, робота з різними вимірювальними приладами.

2. *Теми для попереднього опрацювання*

2.1. Мультиплексори та демультиплексори

2.2. Шифратори та дешифратори

2.3. Цифрові компаратори

3. *Завдання*

Дослід 1. Дослідження функціонування селектора-мультиплексора.

У відповідності з варіантом (табл. 4.9) вибрати селектор-мультиплексор.

Таблиця 4.9

№ з/п	Серія SN74	Вітчизняні МС	Функціональне призначення
1	Generic	–	Селектор-мультиплексор 8 × 1
2	74150	155КП1	Селектор-мультиплексор 16 × 1
3	74151	155КП7	Селектор-мультиплексор 8 × 1
4	74153	155КП2	2 селектора-мультиплексора 4 × 2
5	74157	533КП16	4-розряд. селектор-мультиплексор 2 × 1
6	74158	1533КП18	4-розряд. селектор-мультиплексор 2 × 1 з інвер.
7	74251	155КП15	Селектор-мультиплексор 8 × 1 з 3-ма станами
8	74253	155КП12	2 селектора-мультиплексора 4 × 1 з 3-ма станами
9	74257	155КП11	4 селектора-мультиплексора 2 × 1 з 3-ма станами
10	74258	155КП14	4 селект.-мультиплек. 2 × 1 з 3-ма станами та інвер.
11	74298	155КП13	4 2-входових мультиплексора з запам’ятанням
12	74352	–	
13	74353	555КП17	Здвоєний MUX 4 × 1 з інвер. та 3-ма станами вих.

1.1. У програмі схемотехнічного моделювання аналогових та цифрових радіоелектронних пристроїв Electronics Workbenct зібрати електричну схему проведення дослідження з цифровим пристроєм з використанням генератора слів, багатоканального осцилографа та підключити до виходу пристрою світлодіоду, як показано, наприклад, на рис. 4.19.

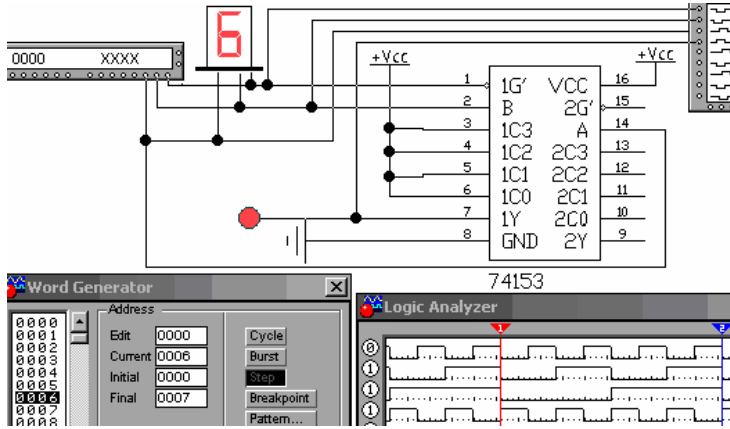


Рис. 4.19. Схема дослідження ІМС 74153

Вивід G’ позначає інверсний вхід дозволу вибору елемента. Вибір ІМС можна провести двома шляхами: або натиснути на піктограму з назвою Digital, а потім перетягнути на робоче поле піктограму з ім’ям MUX, вибрати згідно з варіантом свою ІМС та натиснути кнопку Accept, або вибрати піктограму з ім’ям Digit, натиснути на піктограму класу ІМС, перетягнути його на робоче поле, вибрати згідно з варіантом свою ІМС та підтвердити свій вибір кнопкою Accept.

На схемі для візуального контролю генератора слів паралельно йому підключений 7-сегментний індикатор з внутрішнім дешифратором.

1.2. Підключити до всіх входів даних всі 0 (рис. 4.19). Зняти осцилограму функціонування ІМС.

1.3. Підключити до всіх входів даних всі 1. По осцилограмі функціонування ІМС зробити висновки щодо змін, які з’явились. Також виміряти час затримки вихідного сигналу відносно вхідних. Додати цю характеристику до звіту.

Виводи МС мають таке призначення:

– V_{CC} – напруга живлення +5 В для ТТЛ;

– V_{dd} – напруга живлення для КМОП-логіки;
– GND – загальний для ТТЛ;
– U_{SS} – загальний для КМОП-логіки;
– I, A, B, C, … – входи;
– Y, O – виходи;
– W – інверсний вихід;
– G’ – вхід дозволу (активний рівень – низький).

Для складніших ІМС визначення функціонального призначення їх виводів доцільно проводити шляхом зіставлення з вітчизняними аналогами. Для більш оперативної орієнтації при роботі з цифровими ІМС приводиться перелік найбільш поширених мнемонічних позначень на їх функціональних схемах і в таблицях станів:
– A = B (Parity) – вихід рівності операндів A і B;
– A/S (Asynchro/Synchro) – вхід асинхронного і синхронного режимів;
– B/D (Binary/Decimal) – вхід перемикання рахунку з двійкового на десятковий;
– (Clock input) – вхід тактових імпульсів;
– CD (Count down) – вхід тактових імпульсів на зменшення рахунку (у реверсивних лічильниках);
– Ci (Count up) – вхід тактових імпульсів на збільшення рахунку;
– CE (Clock enable) – вхід дозволу для тактових імпульсів;
– CEP (Count enable parallel) – вхід паралельного нарощування розрядів лічильника;
– CET (Count enable trickle) – вхід дозволу рахунку при нарощуванні розрядів лічильника;
– CLR (Clear) – вхід скидання;
– C, (Carry in) – вхід для розряду перенесення;
– CS (Chip select) – вибір кристала; визначає доступ до однієї з ІМС пристрою;
– D (Data input) – вхід даних тригера, лічильника, регістра;
– DSI (Data serial input) – вхід послідовних даних;
– DS (Data select) – вхід вибору даних;
– DL, DR (Data left, Data right) – входи для послідовного завантаження (регістра) зліва, справа;
– DSL, DSR (Data shift left, Data shift right) – входи для зрушення даних вліво, вправо;
– E (Enable) – вхід сигналу дозволу;
– EC (Enable count) – вхід сигналу дозволу рахунку;

– EE (Enable even) – вхід сигналу дозволу, рахунковий;
– EI (Enable input) – вивів ІМС, за яким дається дозвіл на прийом даних;
– EIO (Enable input/output) – вивід для одночасного дозволу по входу і виходу;
– EO (Enable output) – вивід для дозволу по виходу;
– LSB (Least significant bit) – молодший значущий розряд (МЗР). M (Mode control) – вибір режиму "Арифметики-логіка" в АЛП;
– PE (Parallel enable load) – вхід дозволу паралельного завантаження;
– P/S (Parallel/serial) – вхід перемикання режимів паралельного або послідовного завантаження;
– R (Reset) – асинхронне скидання даних;
– RE (Read enable) – вхід дозволу читання;
– S (Set) – установка тригера, лічильника, регістра;
– S (Set enable) – дозвіл попереднього паралельного запису;
– SI (Serial input) – вхід послідовний;
– SIR, SIL (Serial input right, SI left) – вхід послідовний справа, зліва;
– SR (Synchro reset) – вхід скидання синхронно з тактовим імпульсом;
– TC (Terminal count) – вихід закінчення рахунку;
– TCD (Terminal count down) – те ж на зменшення рахунку;
– TCU (Terminal count up) – те ж на збільшення рахунку.

Наприклад, для позначення мультиплексорів найбільш часто використовуються символи:

– для 4-канального мультиплексора 74153 (К155КП2): A, B – адресні входи, 1G, 2G – інверсні входи дозволу першого і другого мультиплексорів, 1C0...1C3 і 2C0...2C3, 1Y і 2Y – входи і виходи першого і другого мультиплексорів відповідно;

– для 4-канальних і 2-канальних мультиплексорів 74298 (К555КП13): A1, A2; B1, B2; C1, C2; D1, D2 – входи однойменних 2-канальних мультиплексорів, QA, QB, QC, QD – виходи відповідних мультиплексорів, CLK – синхросигнал запам'ятовування результату, WS – сигнал вибору напрямку прийому інформації з першого або другого каналу..

1.2. Занести до звіту схему та результати досліджень згідно з варіантом та схему внутрішньої структури цифрового пристрою, яку теоретично може мати пристрій, що досліджується.

Дослід 2. Дослідження функціонування дешифратора

У відповідності з варіантом табл. 4.10 вибрати дешифратор.

Таблиця 4.10

№ з/п	Серія	Вітчизняні МС	Функціональне призначення
1	Generic	–	8-to-1 DEMUX
2	Generic	–	3-to-8 DEMUX
3	7442	555ИД6	DC 4 × 10
4	7445		
5	7447		
6	74138	155ИД7	Дешифратор-демультиплексор 3 × 8
7	74139	155ИД14	2 дешифратори-мультиплексора 2 × 4
8	74145	155ИД10	Двійково-десятковий DC з відкр. колектором
9	74154	155ИД3	Дешифратор-демультиплексор 4 × 16
10	74155	155ИД4	2 дешифратори-мультиплексора 2 × 4
11	74156	555ИД5	2 DC-DMUX 2 4 з відкритим колектором
12	74159		
13	74445		
14	4028	561ИД1	Двійково-десятковий DC
15	4514		
16	4515		

Зібрати схему дослідження дешифратора, під'єднати до нього генератор слів, світлодіод та 7-сегментний індикатор з дешифрацією адреси вхідних слів згідно з рис. 4.20. Разом зі світлодіодами можна підключити й багатоканальний осцилограф та прослідити зміну станів у динаміці.

Навести в звіті необхідні схеми та рисунки.

Дослід 3. Дослідження елемента перевірки на парність (IMC 74280 – аналог МС К555ИП5)

Для цього необхідно відкрити піктограму з назвою Digital ICs. Вибрати елемент 742xx, а потім й необхідну мікросхему.

3.1. Для дослідження мікросхеми підключити генератор слів на два світлодіоди на вихід парності та непарності. МС має 9 входів (A, B, C, …, I) та два виходи (EVEN, ODD), один із яких – інверсний. Вхід I використовується для управління видом контролю (0 – контроль парності, 1 – контроль непарності), на який необхідно подати напругу + 5 В. Вивід NC не використовується. Правильність функціонування перевіряється за допомогою генератора слів. Стан виходів МС контролю-

ється світлодіодами. Один із варіантів дослідження МС наведено на рис. 4.21.

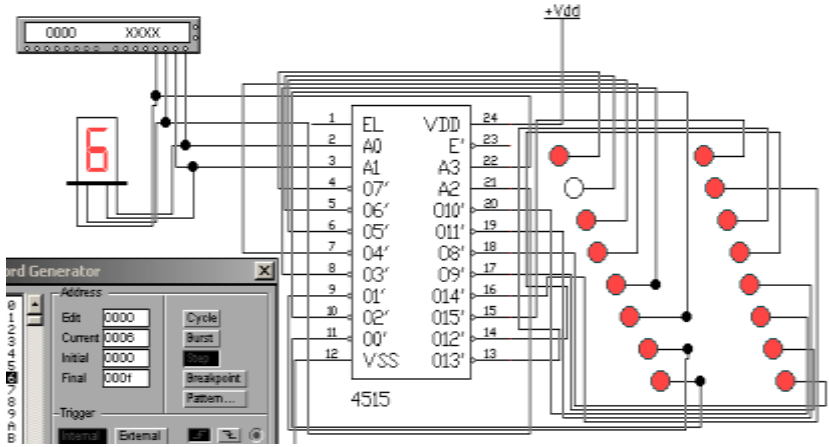


Рис. 4.20. Схема включення IMC 4515

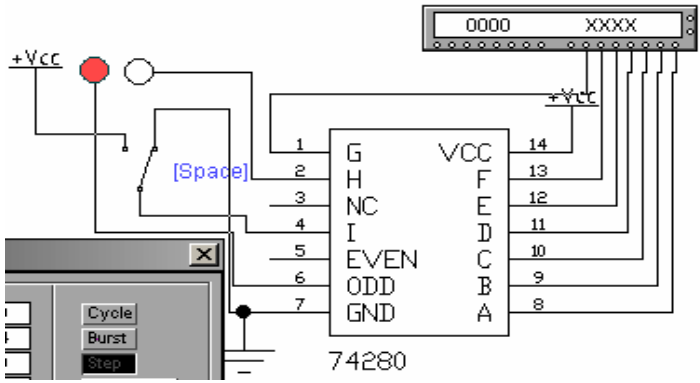


Рис. 4.21. Схема дослідження IMC 74280

Переключення між 0 та 1 відбувається за допомогою перемикача, який управляється натисканням на клавішу “пропуску”.

Дослід 4. Складіть схему компаратора та проведіть його дослідження згідно з варіантами, які наведені в табл. 4.11.

4.1. Для побудови компаратора, який виконує функцію $F_{A=B}$, використовується формула

$$R = r_{n-1}\, r_{n-2} \ldots r_0,$$

де $r_i = \overline{a_i}\,\overline{b_i} \vee a_i b_i = \overline{\overline{a_i}\,\overline{b_i} \vee a_i b_i} = \overline{a_i \oplus b_i}$.

Таблиця 4.11

№ з/п	Функція компаратора		
1	$F_{A=B}$ для n = 4	9	$F_{A>B}$ для n = 3
2	$F_{A>B}$ для n = 3	10	$F_{A<B}$ для n = 3
3	$F_{A<B}$ для n = 3	11	$F_{A>B}$ для n = 4
4	$F_{A>B}$ для n = 4	12	$F_{A\le B}$ для n = 2
5	$F_{A\ge B}$ для n = 2	13	$F_{A\ge B}$ для n = 2
6	$F_{A\le B}$ для n = 3	14	$F_{A<B}$ для n = 4
7	$F_{A\ge B}$ для n = 3	15	$F_{A=B}$ для n = 5
8	$F_{A=B}$ для n = 6		

Порівняння двох однорозрядних слів виконує елемент NXOR. Схема виконання функції $F_{A>B}$ для двох 3-розрядних слів (n = 3) наведена на рис. 4.22.

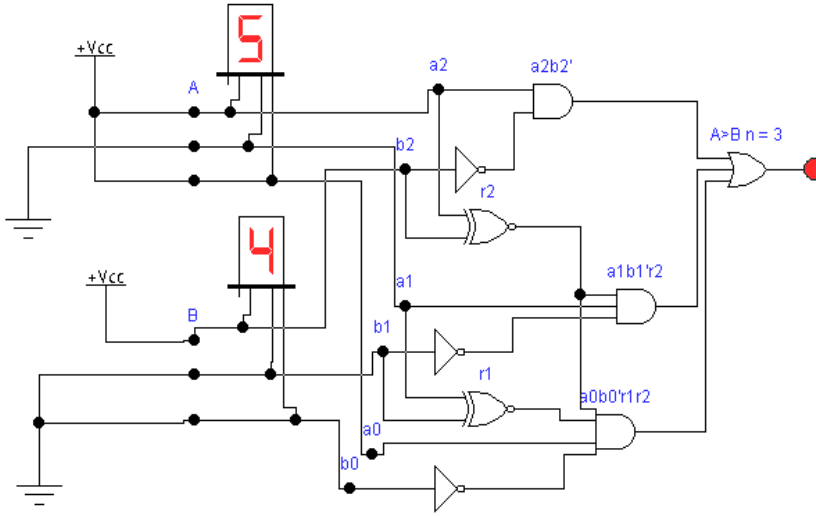


Рис. 4.22. Схема виконання функції F(A > B) для n = 3

4.2. Для побудови компаратора, який виконує функцію $F_{A>B}$, ви-

користовується формула

$$F_{A>B} = a_{n-1}\overline{b}_{n-1} \vee a_{n-2}\overline{b}_{n-2}r_{n-1} \vee a_{n-3}\overline{b}_{n-3}r_{n-2}r_{n-1} \vee \ldots \vee \overline{a_0 b_0} r_1 \ldots r_{n-1}.$$

Наприклад, функція $F_{A>B}$ порівняння двох 3-розрядних слів має вигляд

$$F_{A>B} = a_2\overline{b}_2 \vee a_1\overline{b}_1 r_2 \vee a_0\overline{b}_0 r_2 r_1.$$

4.3. Для побудови компаратора, який виконує функцію $F_{A<B}$, використовується формула

$$F_{A<B} = \overline{a}_{n-1} b_{n-1} \vee \overline{a}_{n-2} b_{n-2} r_{n-1} \vee \overline{a}_{n-3} b_{n-3} r_{n-2} r_{n-1} \vee \ldots \vee \overline{a_0 b_0} r_1 \ldots r_{n-1}.$$

Наприклад, функція $F_{A<B}$ порівняння двох 4-розрядних слів має вигляд

$$F_{A<B} = \overline{a}_3 b_3 \vee \overline{a}_2 b_2 r_3 \vee \overline{a}_1 b_1 r_3 r_2 \vee \overline{a}_0 b_0 r_3 r_2 r_1.$$

4.4. Для побудови компараторів, які виконують функції $F_{A\le B}$ та $F_{A\ge B}$, використовуються формули

$$F_{A\le B} = \overline{F_{A>B}}\ ;\ F_{A\ge B} = \overline{F_{A<B}}.$$

4. *Питання для самоперевірки*

4.1. Що таке мультиплексор та демультиплексор, та яке їх призначення?

4.2. Для вирішення яких задач використовується дешифратор?

4.1. Які функції виконує цифровий компаратор та в яких пристроях він може використовуватись?

4.2. Яке значення мають формувачі парності та де вони можуть використовуватись?

4.2. Типові комбінаційні вузли: кодоперетворювачі, програмовані логічні матриці, комбінаційні суматори

4.2.1. Перетворювачі кодів

Кодоперетворювачі – це комбінаційні пристрої, які призначені для перетворення одного кода в інший. Перетворювачі кодів здійснюють переведення двійкових чисел в інші коди, які використовуються на практиці: двійково-десятковий код, код Грея, код “1 з N” та ін.

4.2.1.1. Двійково-десятковий код

Двійково-десятковий код дозволяє відображати десяткові числа за допомогою двійкових символів. Для подання кожної цифри десяткового числа використовується чотири біти (тетрада), а десяткове число записується у вигляді послідовності тетрад. Кожна десяткова цифра подається в звичайному двійковому коді з ваговими коефіцієнтами цифр 8421 і двійково-десятковий код називається **натуральним**. Наприклад, запис десяткового числа 379 у натуральному двійково-десятковому коді має вигляд 12-бітової послідовності: 0011 0111 1001. Оскільки для запису двійково-десяткових чисел використовується тільки 10 комбінацій двійкових символів з 16 можливих для кожної тетради, то двійково-десяткові числа виходять довше двійкових. Перетворення двійкового коду, який часто позначається як код 8421 (з позначкою ваги кожного розряду), у двійково-десятковий наведено в табл. 4.12.

Перетворювачі двійкового коду в двійково-десятковий зазвичай виконуються багаторозрядними. Наприклад, мікросхема 155ПР7 (рис. 4.23) перетворює шестирозрядний двійковий код у двійково-десятковий.

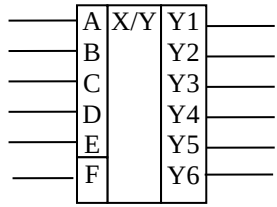


Рис. 4.23. Умовне графічне позначення кодоперетворювача ІМС ПР7

Таблиця 4.12

Код 8421	Код 2-10
x3x2x1x0	y3y2y1y0
0000	0000
0001	0001
0010	0010
0011	0011
0100	0100
0101	0101
0110	0110
0111	0111
1000	1000
1001	1001
1010	0000
1011	0000
1100	0000
1101	0000
1110	0000
1111	0000

Мікросхема має тільки 5 інформаційних входів: A – E. Мається на увазі, що молодший розряд A двійкового коду завжди збігається з молодшим розрядом Y0 двійково-десяткового коду, а перетворення Y0 = A проводиться поза мікросхемою. Виходи мікросхеми Y1 – Y6 виконані на схемі з відкритим колектором, і можуть бути переведені в закритий стан одиничним сигналом на вході дозволу F.

Зворотне перетворення двійково-десяткового коду в двійковий виконує мікросхема 555ПР6, побудована за аналогічним принципом. Схема каскадного включення ІМС типу ПР6 при перетворенні двох тетрад двійково-десяткового коду в двійковий наведена на рис. 4.24.

4.2.1.2. Код Грея

Код Грея відрізняється від звичайного двійкового коду тим, що при зміні будь-якого числа на одиницю змінюється тільки один його двійковий розряд. Код Грея не дозволяє виконувати арифметичні операції і його використовують звичайно тільки при передачі інформації. Для подання будь-якого числа за допомогою коду Грея необхідно стільки ж біт, як і для звичайного двійкового запису. Тому кількість входів в перетворювачі двійкового коду в код Грея завжди дорівнює кількості виходів. У табл. 4.13 для коду Грея наведені всі десяткові числа від 0 до 15.

З таблиці видно, що вихідні змінні y0, yl y2, y3 і вхідні x0, x1, x2, x3 перетворювача визначені конституентами одиниці на наборах:

$$y3 = \Sigma_0(8, 9, 10, 11, 12, 13, 14, 15);$$
$$y2 = \Sigma_0(4, 5, 6, 7, 8, 9, 10, 11);$$
$$y1 = \Sigma_0(2, 3, 4, 5, 10, 11, 12, 13);$$
$$y0 = \Sigma_0(1, 2, 5, 6, 9, 10, 13, 14).$$

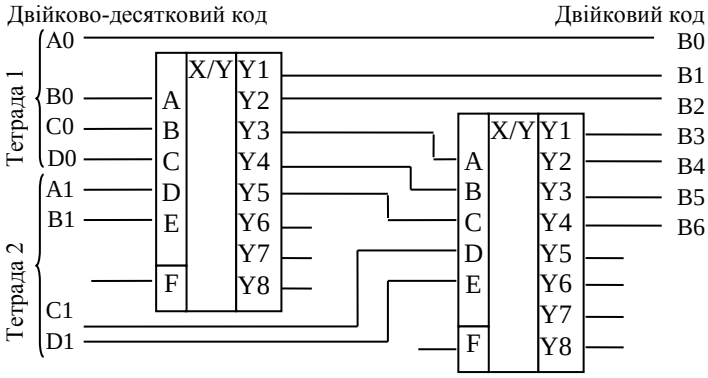


Рис. 4.24. Схема каскадного включення ІМС типу ПР6
при перетворенні двох тетрад двійково-десяткового коду в двійковий

Таблиця 4.13

Десятк. запис	Двійк. код	Код Грея	Десятк. запис	Двійк. код	Код Грея
	x3x2x1x0	y3y2y1y0			
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Запишемо в карти Карно значення функцій на десяткових наборах та мінімізуємо їх:

x_3x_2 \ x_1x_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$y3 = x3$

x_3x_2 \ x_1x_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$y2 = \overline{x}_3x_2 + x_3\overline{x}_2 = x_3 \oplus x2$

x_1x_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1		
10			1	1

$y1 = x_2\overline{x}_1 + \overline{x}_2x_1 = x_2 \oplus x_1$

x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$y0 = \overline{x}_1x_0 + x_1\overline{x}_0 = x_1 \oplus x_0$

Зведені рівняння функцій після мінімізації мають вигляд:

$$y3 = x3,\ y2 = x3 \oplus x2,$$
$$y1 = x2 \oplus x1,\ \ y0 = x1 \oplus x0.$$

На їх основі побудовані схеми на рис. 4.25.

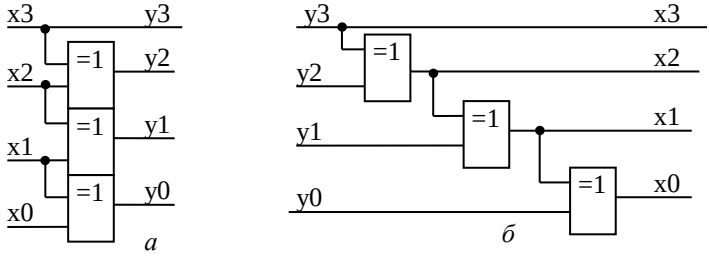


Рис. 4.25. Схема перетворення: *а* – двійкового коду в код Грея; *б* – коду Грея у двійковий код

Перетворення двійкового коду в код Грея виконує схема (рис. 4.25, *а*) на логічних елементах ВИКЛЮЧНЕ АБО.

Зворотне перетворення коду Грея в двійковий код також проводиться за допомогою елементів ВИКЛЮЧНЕ АБО (рис. 4.25, *б*), яка отримана після відповідної мінімізації.

У цьому випадку рівняння виглядають таким чином:

$$x3 = y3,\ x2 = y3 \oplus y2,\ x1 = x2 \oplus y1,\ x0 = x1 \oplus y0.$$

4.2.1.3. Семисегментний код

Семисегментний код використовується для управління пристроями індикації, що складаються з семи сегментів, які світяться, на світлодіодах або рідких кристалах. Розташовані у певному порядку сім сегментів індикатора (A, B, C, D, E, F, G) здатні відобразити всі десяткові цифри від 0 до 9.

Перетворювач двійкового коду в семисегментний (семисегментний дешифратор) будується за таблицею істинності.

Як приклад розглянемо **перетворювач двійково-десяткового коду** в код для семисегментних світодіодних індикаторів. На рис. 4.26 також показана схема підключення до семисегментного індикатора та фрагмент підключення одного сегменту до виходу схеми із загальним емітером і наведені зображення перших п’яти цифр.

Такий перетворювач повинен мати чотири входи, оскільки для кодування десяткових цифр від 0 до 9 достатньо чотирьох двійкових, і сім виходів, по одному на кожен сегмент. Функціонування семисегментного індикатора визначається таблицею істинності (табл. 4.14).

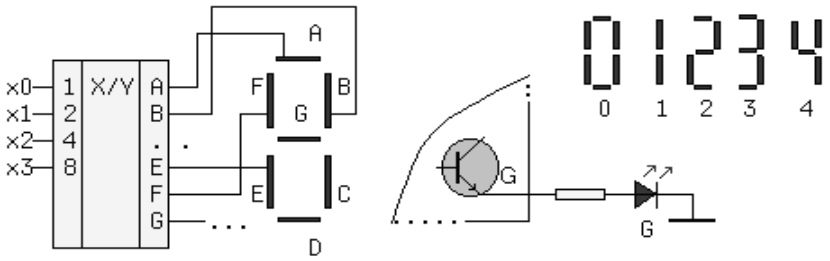


Рис. 4.26. Схема підключення до семисегментного індикатора

Таблиця 4.14

Двійк. код				Сегменти							Двійк. код				Сегменти						
x_3	x_2	x_1	x_0	A	B	C	D	E	F	G	x_3	x_2	x_1	x_0	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	1	1	1	1	1	1	1
0	0	0	1	0	1	1	0	0	0	0	1	0	0	1	1	1	1	1	0	1	1
0	0	1	0	1	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0	1	1	0	1	1	0	0	0	0	0	0	0
0	1	0	0	0	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
0	1	0	1	1	0	1	1	0	1	1	1	1	0	1	0	0	0	0	0	0	0
0	1	1	0	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0

Запишемо в карти Карно значення функцій кожних літер та мінімізуємо для приклада тільки f(A):

x_3x_2 \ x_1x_0	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	0	0	0
10	1	1	0	0

$$F(A) = \overline{x}_3x_1 + \overline{x}_3x_2x_0 + x_3\overline{x}_2\overline{x}_1 + \overline{x}_2\overline{x}_1\overline{x}_0$$

4.2.1.4. Код 1 з N

Код “1 з N” зв’язує N двійкових чисел від 0 до N – 1 і N вихідних змінних Y_N. Кожна вихідна змінна набуває одиничного значення при

появі на вході схеми двійкового числа *N*. Двійкові числа представляються *n* двійковими змінними, причому обов'язково $2^n > N$.

Перетворювач коду, що виконує це завдання, називається дешифратором (decoder). Схема, що виконує зворотне перетворення коду "1 з *N*" у двійковий, називається пріоритетним шифратором (coder). На його виходах формується двійкове число, відповідне старшому входу, на якому присутня логічна одиниця. Значення решти входів береться до уваги.

4.2.1.5. Перетворення прямого коду в обернений

Таке перетворення реалізується складанням за mod 2 двох значень: даного розряду і сигналу управління перетворювачем. При нульовому значенні сигналу забезпечується проходження на вихід схеми прямого коду числа. При одиничному значенні сигналу управління кожен розряд вихідного коду буде інверсією відповідного розряду вхідного слова. Якщо перетворюється число, що містить знаковий розряд, то його значення може бути безпосередньо використано як сигнал управління.

Схема перетворення прямого коду в обернений наведена на рис. 4.27.

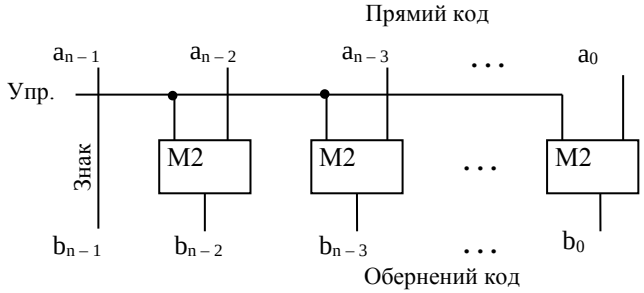


Рис. 4.27. Схема перетворення прямого коду в обернений

4.2.1.6. Перетворення прямого коду в додатковий

У цьому випадку операція перетворення не є порозрядною. Для отримання додаткового коду необхідно проінвертувати всі розряди перетворюваного коду і потім до результату додати одиницю.

Зіставлення прямого і додаткового кодів показує, що останній відрізняється від першого інвертуванням старших розрядів від $n-1$ до $i+1$ включно, де i – номер першого справа розряду, що містить одиницю.

Такі викладки для значущих цифр можна записати математичними виразами:

$$b_0 = a_0 \oplus 0;$$
$$b_1 = a_1 \oplus a_0;$$
$$b_2 = a_2 \oplus (a_1 \vee a_0);$$
$$b_3 = a_3 \oplus (a_2 \vee a_1 \vee a_0)$$
$$\dots$$
$$b_{n-2} = a_{n-2} \oplus (a_{n-3} \vee a_{n-4} \vee \dots \vee a_0).$$

Наприклад, для прямого коду 1.110011000 додатковим кодом буде код 1.001101000.

На рис. 4.28 наведена схема перетворення прямого коду в додатковий.

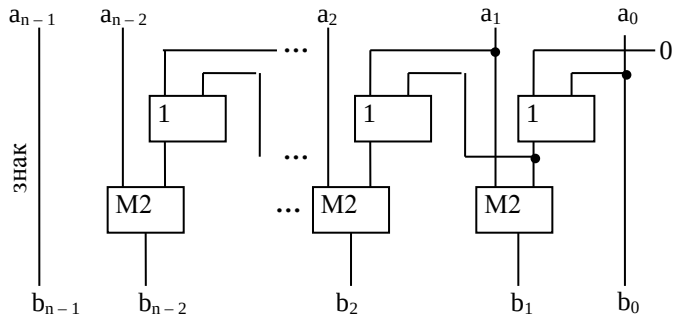


Рис. 4.28. Схема перетворення прямого коду в додатковий

Знаковий розряд можна використати як управляючий.

4.2.2. Програмовані логічні матриці

Програмована логічна матриця (ПЛМ) являє собою функціональний блок, виконаний у вигляді великої інтегральної схеми (ВІС) на базі напівпровідникової технології і призначений для реалізації дискретних

пристроїв.

ПЛМ набули більш широкого застосування в різних цифрових системах як комбінаційні пристрої з нерегулярною структурою, нетипові перетворювачі кодів тощо. ПЛМ є невід'ємною частиною мікропроцесорів, у яких вони виконують функції місцевого пристрою керування, який формує сигнали мікрооперацій у відповідності до коду операції, що надходить на вхід мікропроцесора. На цей час промисловістю розпочато випуск ПЛМ з елементами пам'яті.

Основні причини широкого розповсюдження ПЛМ такі:

1. У зв'язку з ускладненням програмного забезпечення намітилась тенденція апаратної реалізації багатьох програмних функцій ЕОМ.

2. Розширення можливостей інтегральної технології різко збільшило інформаційну ємність ПЛМ.

3. Поява ПЛМ, які програмуються користувачем, істотно розширило сферу їх застосування.

4. Створення систем автоматизованого проектування дискретних пристроїв на базі універсальних і спеціалізованих ЕОМ значно спростило процес синтезу таких пристроїв на основні ПЛМ.

Програмовані логічні матриці можуть бути класифіковані за такими основними ознаками:

- за наявністю елементів пам'яті;
- за способом програмування;
- за способом технічної реалізації.

У залежності від наявності елементів пам'яті розпізнають комбінаційні ПЛМ і ПЛМ з пам'яттю. ПЛМ другого типу дозволяють реалізувати складні дискретні пристрої з пам'яттю, які мають нерегулярну структуру. За способом програмування, тобто способом організації сполучень у матрицях ПЛМ поділяються на 3 типи.

До першого типу відносяться ПЛМ, програмовано одноразово в процесі виготовлення (ОПЛМ). У ПЛМ цього типу міжсполучення шин у матрицях здійснюється на останньому етапі виготовлення за допомогою спеціального шаблону (маски) і в подальшому змінені бути не можуть. Найчастіше використовується спосіб маскового програмування, який припускає використання маски при металізації окремих ділянок кристала з метою підключення виводів елементів до шин матриць ПЛМ. Застосування ПЛМ з масковим програмуванням економічне виправдане при виготовлені великої кількості виробів.

До другого типу можуть бути віднесені ПЛМ, програмовані користувачем. У цьому випадку ПЛМ програмується користувачем за до-

помогою спеціального складного устаткування – тобто процес виготовлення ПЛМ та їх контролю завершується користувачем. Найбільш широко використовуються два способи програмування: перший спосіб оснований на усуненні міжсполучень під дією струму підвищеної амплітуди, другий спосіб реалізується шляхом пробою переходів транзисторів.

При використанні репрограмованих ПЛМ (РПЛМ) реалізовані ними функції можуть бути змінені багаторазово в процесі експлуатації. Повторне програмування (репрограмування) таких ПЛМ виконується електричним способом після стирання вмісту матриць під дією ультрафіолетового опромінювання або електричним способом.

Матричні схеми є сіткою ортогональних провідників, на місцях перетину яких встановлені елементи односторонньої провідності (діоди, транзистори). Матричні схеми бувають 2- і 3-рівневі. Кожен рівень називається матрицею. Матриця першого рівня називається матрицею М1, матриця другого рівня – М2.

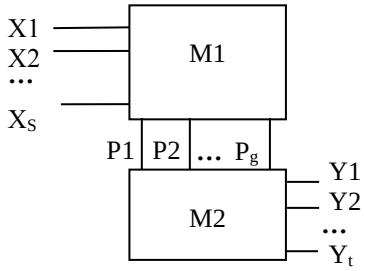


Рис. 4.29. Функціональна схема ПЛМ

Зазвичай матриця М1 реалізує елементарні кон'юнкції і називається матрицею кон'юнкцій, а матриця М2 – матрицею диз'юнкцій, оскільки дозволяє реалізувати диз'юнкції змінних.

Розглянемо дворівневу матричну схему (рис. 4.29).

Функціональна схема матриці М1 представлена на рис. 4.30. Дана матриця може реалізувати чотири кон'юнкції, за кількістю ортогональних провідників:

$$P1 = X1X2\overline{X1};\ P2 = \overline{X2}X1;$$
$$P3 = \overline{X3}X2;\ P4 = \overline{X3}X1.$$

Реалізація необхідних кон'юнкцій здійснюється шляхом "про палення" перемичок (включених послідовно з напівпровідниковим діодом), розташованих на місцях перетину ортогональних провідників, що не беруть участі в утворенні кон'юнкцій.

У початковому стані на всіх перетинах провідників матриці М1 є з'єднання, тобто матриця реалізує всі кон'юнкції змінних, причому в кожну кон'юнкцію входять всі змінні як із запереченням так і без нього. Для отримання необхідних кон'юнкцій слід "пропалювати" всі легкоплавкі перемички, що знаходяться на вузлах, які не беруть участі в

кон’юнкціях.

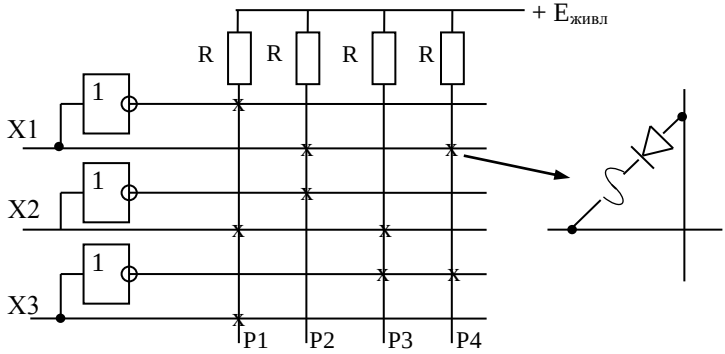


Рис. 4.30. Функціональна схема матриці М1: S = 3; q = 4

Розглянемо процес реалізації кон’юнкції на прикладі Р1. На провіднику Р1 підтримуватиметься рівень логічної 1 (за рахунок напруги джерела живлення Е) тільки за умови наявності рівня логічної “1” на всіх трьох вказаних перетинах. Очевидно, що на верхньому (по схемі) перетині рівень “1” матиме місце тільки за умови подачі на вхід матриці рівня логічного "0", оскільки сигнал Х1 “потрапляє” на цей перетин через інвертор. Поява “0” хоч би на одному з входів Х3 або Х2 приводить до “зникнення” логічної “1” на дроті P_1, оскільки вся напруга живлення Е падатиме на обмежувальному опорі R. У результаті цього на виході P_1 з’явиться “0”.

На рис. 4.31 наведена схема матриці М2 для двох виходів (кількість провідників Р однаково для М1 і М2 і в даному прикладі $q = 4$).

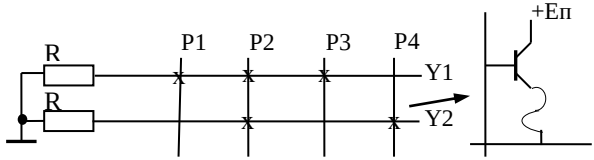


Рис. 4.31. Матриця диз'юнкцій М2

Схема матриці диз’юнкції М2 містить опори навантаження і транзисторні ключові з’єднувачі (на місцях перетинів ортогональних провідників).

У даному випадку матриця М2 реалізує дві диз’юнкції:

$$Y1 = P1 + P2 + \underline{P3} = X3\underline{X2}\overline{X1} + \overline{X2}X1 + \overline{X3}X2;$$
$$Y2 = P2 + P4 = \overline{X2}X1 + \overline{X3}X1.$$

Ємність інформації, яку можна записати в матричну схему, визначається інформаційною площею матриць, вірніше сумою S_{m1} і S_{m2}.

$$S_m = S_{m1} + S_{m2} = 2Sq + qt.$$

На практиці використовується матриця М2 з елементом ВИКЛЮЧНЕ АБО.

***Приклад* 4.1.** Реалізувати логічну функцію F = AB.

Приклад реалізації функції F = AB на ПЛМ наведено на рис. 4.32.

***Приклад* 4.2.** Реалізувати логічну функцію $D = \overline{C}(\overline{A} + B)$.

Для реалізації такої функції спочатку необхідно перетворити її до виukzду:

$$D = \overline{C + \overline{(\overline{A} + B)}} = \overline{C + A\overline{B}}.$$

Приклад реалізації функції $D = \overline{C + \overline{(\overline{A} + B)}} = \overline{C + A\overline{B}}$ на ПЛМ наведено на рис. 4.33.

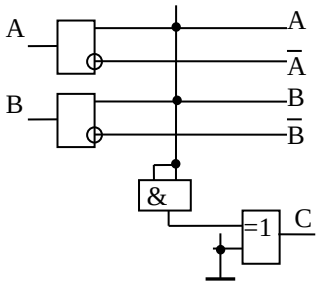


Рис. 4.32. Приклад реалізації функції F = AB на ПЛМ

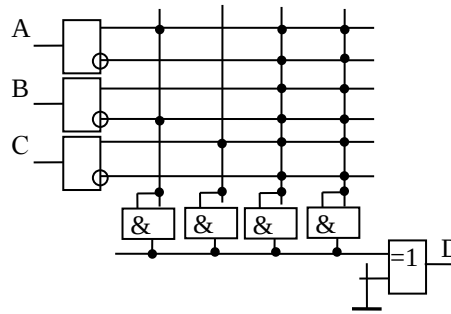


Рис. 4.33. Реалізація функції до прикладу 4.2

***Приклад* 4.3.** Реалізувати на ПЛМ дешифратор з організацією 2 × 4 із входом дозволу та низькими активними рівнями.

Приклад реалізації функцій:

$$Q0 = \overline{\overline{A}1\overline{A}0E};$$

$$Q1 = \overline{\overline{A}1A0E};$$

$$Q2 = \overline{A1\overline{A}0E};$$

$$Q3 = \overline{A1A0E}.$$

на ПЛМ наведено на рис. 4.34.

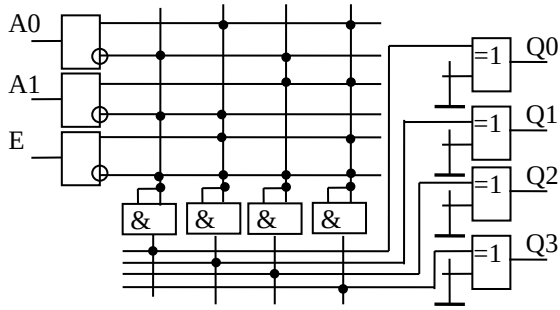


Рис. 4.34. Реалізація функцій до прикладу 4.3

Умовно графічне позначення ПЛМ КР556РТ1 наведено на рис. 4.35. ІМС РТ1 виконана з ніхромними плавкими перемичками та вихідним буфером з відкритим колектором. ІМС РТ2 має такі ж властивості, але відрізняється тільки наявністю тристабільного вихідного каскаду. Ці ІМС мають 8 функцій від 16 змінних та сигнали управління:

CE – інверсний вхід дозволу вибору кристала;

FE – лінія дозволу програмування.

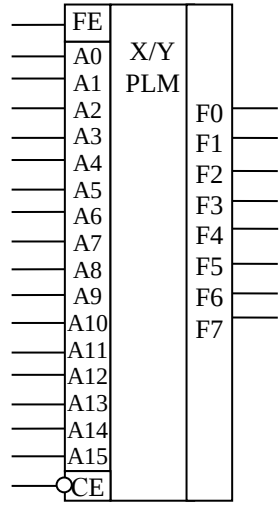


Рис. 4.35. Умовно графічне позначення ІМС 556РТ1

На практиці часто зустрічаються схеми, що складаються з матриць М2 і дешифратора (повного). Такі схеми звичайно називають **постійними пристроями, що запам'ятовують** (**ПЗП**). ПЗП – це елемент (пристрій) пам'яті, що дозволяє зберігати записану в ньому інформацію, і після виключення напруги джерела живлення. За способом запису ПЗП підрозділяються на масочні, програмовні і репрограмовні. Масочні ПЗП програмуються заводом-виготівником за допомогою спеціальних масок, тобто з'єднання на місцях перетину ортогональних провідників закладені в технологію виробництва ПЗП.

Програмовні ПЗП (ППЗП). ППЗП випускаються заводом-виготівником у "чистому вигляді", тобто за всіма адресами записані "0". Програмування ППЗП здійснюється користувачем ППЗП на спеціальній установці, яка називається програматором. У ППЗП можна записати (його програмувати) інформацію тільки один раз. Змінити записану інформацію або виправити її не можна. ППЗП набули широкого застосування в ПК для зберігання програм запуску комп'ютера. Вони володіють більшою швидкодією, ніж репрограмовні ПЗП (РПЗП).

Репрограмовні ПЗП дозволяють, при необхідності, перепрограмувати ПЗП, тобто стерти раніше записану інформацію і записати нову.

За способом стирання раніше записаної інформації РПЗП бувають з ультрафіолетовим (ультрафіолетовими променями) і електричним стиранням. РПЗП дозволяють десятки (деякі 1000) разів перепрограмувати і зберігати записану інформацію десятки і сотні тисяч годин. Швидкодія РПЗП дещо гірша за швидкодію ППЗП.

Незалежно від типу і способу стирання ПЗП мають структуру, наведену на рис. 4.36.

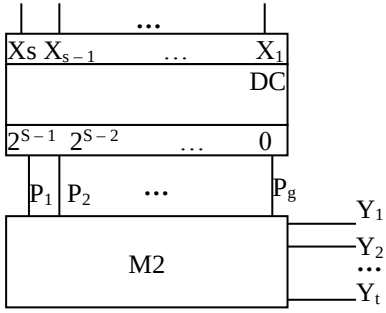


Рис .4.36. Структурна схема ПЗП

Структурна схема ПЗП містить дешифратор на S входів і 2^S-виходів, а також матрицю М2.

Інформаційна ємність ПЗП визначається як $S_{ПЗП} = 2^S$, де S – кількість входів Х. "Бітова" місткість ПЗП визначається як

$S_{ПЗП}$ (біт) $= 2^S$ t (біт).

Промисловістю випускаються ПЗП з об'ємом пам'яті (інформаційною ємністю) на 2 кбайт, 4 кбайт, 16 кбайт, 32 кбайт і т.д., де к = 1024; 1байт = 8 біт.

На функціональних і принципових схемах РПЗП з ультрафіолетовим стиранням зображається так, як показано на рис. 4.37, де:

А – адресні входи;

D – інформаційні виходи;

U_{ce} – вхід подачі напруги запису (у режимі зберігання на цей вхід подається U_{cc});

CE і OE – входи управління станом виводів, якщо CE = OE = 1, входи D мають високоімпедансний стан. При CE = OE = 0 виведення інформації дозволене.

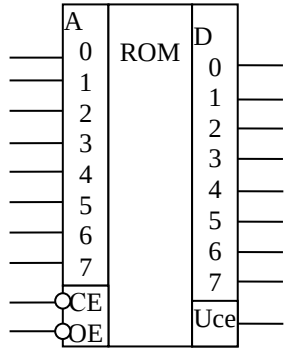


Рис. 4.37. Умовно графічне позначення ІМС К573РФ2, К573РФ5

Мікросхема РПЗП К573РФ2(РФ5) має одинадцяти-розрядний дешифратор, виходи якого сполучені з восьмирозрядною матрицею М2. У процесі запису вихідні елементи РПЗП знаходяться в режимі прийому інформації через виводи D0 ... D7 (на вході "OE" рівень "1"). У режимі зчитування записаної інформації виводи "U_{ce}" і "U_{cc}" об'єднуються, і на них подається напруга живлення +5В.

4.2.3. Суматори

Суматорами називаються дискретні пристрої, які здійснюють арифметичне додавання кодів двох чисел.

Призначення суматорів полягає у визначенні двох чисел, які надходять на їх входи паралельним кодом, і видачі результату додавання цих чисел паралельним кодом. Суматор n-розрядних двійкових чисел являє собою дискретний пристрій, який має 2*n* входів і *n* виходів.

Суматори широко застосовуються в дискретних системах обробки інформації при побудові арифметичних і керуючих пристроїв. Їх використання дозволяє виконувати як операцію додавання, так і інші арифметичні операції за рахунок різного представлення кодів чисел.

Класифікація суматорів може бути виконана за такими основними ознаками (рис. 4.38):

– за основою системи числення чисел, якими оперує суматор;

– за кількістю розрядів;

– за способом організації кіл переносу.

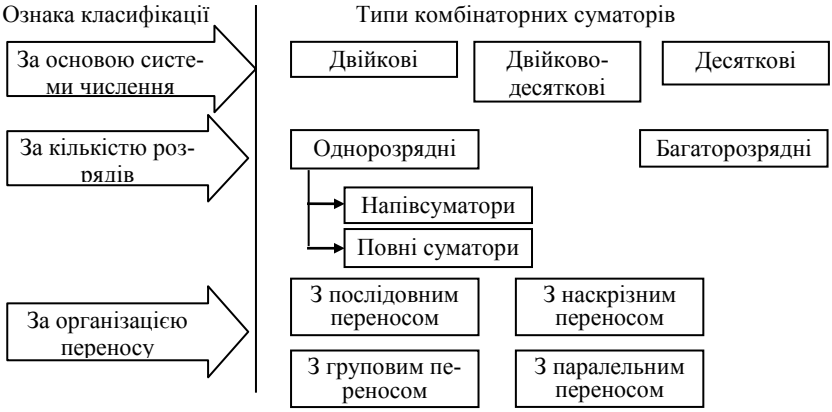


Рис. 4.38. Класифікація комбінаційних суматорів

За основою системи числення розрізняють двійкові, двійково-десяткові, десяткові та інші комбінаційні суматори. Найбільш широке застосування отримали комбінаційні суматори.

За кількістю розрядів суматори поділяються на одно- і багаторозрядні. У свою чергу, одноразрядні комбінаційні суматори за кількістю входів можуть бути розділені на двовходові, які називаються **напівсуматорами,** і трьохвходові, які називаються **повними суматорами**.

У залежності від способу організації ланцюгів переносу розрізняють комбінаційні суматори з послідовним, наскрізним, груповим і паралельним переносом.

Умовне графічне позначення комбінаційних суматорів наведено на рис. 4.39.

Принципи побудови і функціонування

Для розв'язання задачі синтезу уявимо *n*-розрядний комбінаційний суматор як сукупність *n* одноророзрядних комбінаційних суматорів. Для формування значення *i*-го розряду суми на входи одноразрядного суматора необхідно подавати значення і-х розрядів додатків X_i, Y_i, а також так званий сигнал переносу P_{i-1} одиниці з попереднього (і – 1)-го розряду. У свою чергу, завданням одноразрядного суматора крім визначення значення і-го розряду суми S_i є формування сигналу переносу P_i в наступний (і + 1)-й розряд.

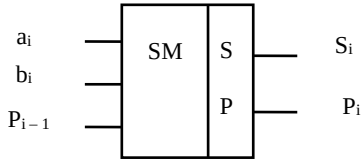


Рис. 4.39. Умовне графічне позначення
комбінаційного суматора

Таким чином, повний однорозрядний суматор являє собою дискретний пристрій, що має три входи, на які надходять сигнали X_i, Y_i та P_{i-1}, і два виходи, на яких формуються сигнали S_i та P_i.

Умови функціонування однорозрядного суматора визначаються табл. 4.15, а логічні функції, які описують його роботу, мають вигляд:

$$Si = \Sigma o(1, 2, 4, 7) = \overline{a_i}\,\overline{b}_i p_{i-1} + \overline{a_i} b_i \overline{p}_{i-1} + a_i \overline{b}_i \overline{p}_{i-1} + a_i b_i p_{i-1},$$

$$Pi = \Sigma o(3, 5, 6, 7) = \overline{a_i} b_i p_{i-1} + a_i \overline{b}_i p_{i-1} + a_i b_i \overline{p}_{i-1} + a_i b_i p_{i-1}.$$

Таблиця 4.15

Умови функціонування однорозрядного суматора

Входи			Виходи	
a_i	b_i	P_{i-1}	S_i	P_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Методом безпосередніх перетворень функція P_i перетворюється до вигляду:

$$P_i = a_i b_i + a_i p_{i-1} + b_i p_{i-1}.$$

Згідно з отриманими виразами S_i та P_i будується схема.

У тому випадку, коли $P_{i-1} = 0$ (переносу у цей розряд немає), умови функціонування однорозрядного суматора визначаються табл. 4.16.

В цьому випадку логічні функції мають вигляд:

$$S_i = a_i \oplus b_i,$$

$$P_i = a_i b_i.$$

Таблиця 4.16

Входи		Виходи	
a_i	b_i	S_i	P_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Внутрішня структурна схема напівсуматора наведена на рис. 4.40. Умовне графічне позначення напівсуматора наведено на рис. 4.41.

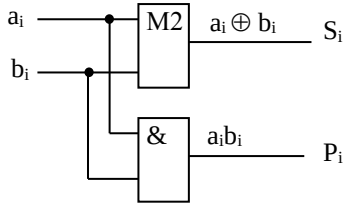


Рис. 4.40. Внутрішня структурна схема напівсуматора

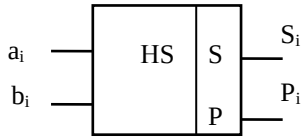


Рис. 4.41. Умовне графічне позначення

Синтез повного суматора з використанням напівсуматорів відбувається згідно з такими виразами:

$$Si = \overline{a_i}\,\overline{b}_i p_{i-1} + \overline{a_i} b_i \overline{p}_{i-1} + a_i \overline{b}_i \overline{p}_{i-1} + a_i b_i p_{i-1} =$$

$$= (\overline{a_i}\,\overline{b}_i + a_i b_i) p_{i-1} + (\overline{a_i} b_i + a_i \overline{b}_i) \overline{p}_{i-1} = a_i \oplus b_i \oplus p_{i-1}.$$

$$Pi = \overline{a_i} b_i p_{i-1} + a_i \overline{b}_i p_{i-1} + a_i b_i \overline{p}_{i-1} + a_i b_i p_{i-1} = b_i p_{i-1} + a_i p_{i-1} + a_i b_i =$$

$$= a_i b_i + (a_i \oplus b_i)\, p_{i-1}$$

Функціональна схема повного однорозрядного суматора, який побудований на основі напівсуматорів, наведена на рис. 4.42.

Функціональна схема повного однорозрядного суматора, який побудований на основі елементів М2, наведена на рис. 4.43.

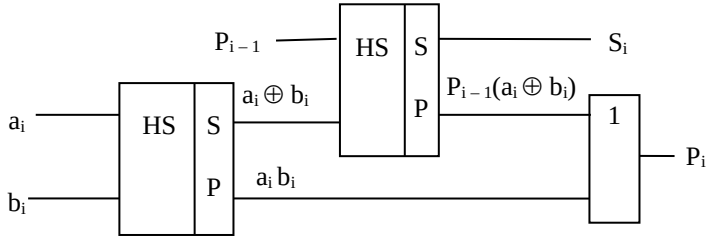


Рис. 4.42. Функціональна схема повного однорозрядного суматора, який побудований на основі напівсуматорів

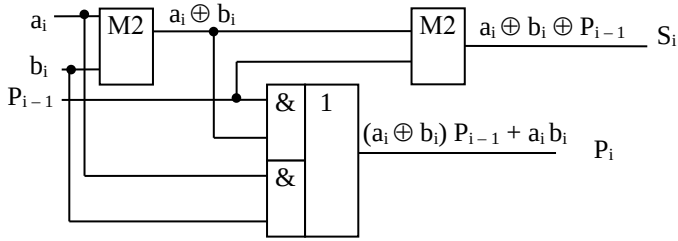


Рис. 4.43. Функціональна схема повного однорозрядного суматора, який побудований на основі елементів М2

З використанням синтезованих вище схем легко побудувати функціональну схему *n*-розрядного комбінаційного суматора.

Для цього достатньо виходи переносу кожного з попередніх розрядів сполучити з відповідними входами наступних розрядів. Функціональна схема n-розрядного комбінаційного суматора наведена на рис. 4.44.

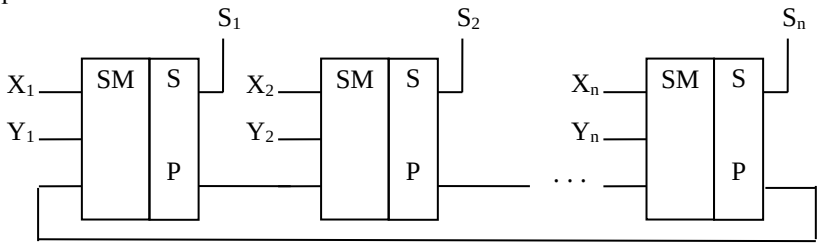


Рис. 4.44. Функціональна схема n-розрядного комбінаційного суматора

Сигнал переносу з *n*-го розряду на перший подається в тому випадку, якщо додавання чисел проводиться у зворотному коді. У противному разі цей зв'язок відсутній.

До складу серій інтегральних логічних елементів входять одно-, дво- і чотирирозрядні комбінаційні суматори. Особливостями їх побудови на основі СІС є наявність вхідної логіки, яка дозволяє керувати полярністю вхідних сигналів або приймати коди чисел з декількох напрямків, а також те, що для зменшення затримки у ланцюгу переносу на виході переносу використовується швидкодіюча ТТЛ-логіка і вводиться тільки один каскад інвертування між входом переносу і виходом переносу.

Наприклад, мікросхема К155ИМ1 являє собою однорозрядний двійковий повний суматор, функціональна схема якого наведена на рис. 4.45.

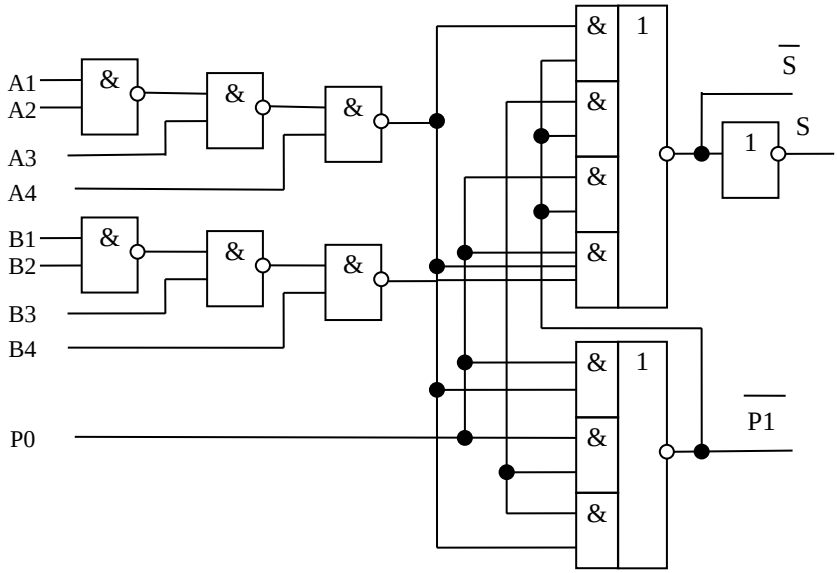


Рис. 4.45. Функціональна схема комбінаційного суматора К155ИМ1

Схема має по чотири входи для чисел A і B, які забезпечують прийом інформації з двох регістрів, вхід переносу з попереднього розряду $\overline{P_0}$, прямий S та $\overline{S}$ інверсний – виходи суми, а також $\overline{P_i}$ інверсний P_i,

який є виходом переносу в наступний розряд. Однорозрядний суматор використовується для організації *n*-розрядних підсумуючих і віднімаючих пристроїв, в яких операції додавання і віднімання проводяться послідовно. Для інвертування вхідних сигналів передбачені елементи І усередині мікросхеми (входи A_3, A_4 і B_3, B_4).

4.2.4. Завдання до самостійних досліджень кодоперетворювачів та комбінаційних суматорів

1. *Мета*

Закріплення теоретичного матеріалу, набуття навиків створення і моделювання комбінаційних цифрових пристроїв (кодоперетворювачів, комбінаційних суматорів), робота з різними вимірювальними приладами.

2. *Теми для попереднього опрацювання*

2.1. Кодоперетворювачі

2.2. Комбінаційні суматори

3. *Завдання*

3.1. Дослід 1. Синтезувати кодоперетворювач, який перетворює один із видів кода в двійковий (табл. 4.17). Відповідальність між кодами наведена в табл. 4.18.

Таблиця 4.17

№ з/п	Код 8421	Код 2421	Код 3321	Код 7421	Код з надм. 3	Код з надм. 6	Код Грея	Код 2 із 5	Код 3а+2	Код Хемминга
	1	2	3	4	5	6	7	8	9	10
0	0000	0000	0000	0000	0011	0110	0000	11000	00010	0000000
1	0001	0001	0001	0001	0100	0111	0001	00011	00101	1101001
2	0010	0010	0010	0010	0101	1000	0011	00101	01000	0101010
3	0011	0011	0011	0011	0110	1001	0010	00110	01011	1000011
4	0100	0100	0100	0100	0111	1010	0110	01001	01110	1001100
5	0101	1011	1010	0101	1000	1011	0111	01010	10001	0100101
6	0110	1100	1011	0110	1001	1100	0101	01100	10100	1100110
7	0111	1101	1101	1000	1010	1101	0100	10001	10111	0001111
8	1000	1110	1110	1001	1011	1110	1100	10010	11010	1110000
9	1001	1111	1111	1010	1100	1111	1101	10100	11101	0011001

Таблиця 4.18

№ з/п	Відповідність між кодами	№ з/п	Відповідність між кодами	№ з/п	Відповідність між кодами
1	1 – 2	10	4 – 6	19	4 – 7
2	2 – 3	11	7 – 5	20	1 – 5
3	3 – 4	12	1 – 4	21	2 – 6
4	4 – 5	13	2 – 5	22	3 – 7
5	5 – 6	14	3 – 6	23	1 – 6
6	6 – 7	15	4 – 7	24	2 – 7
7	1 – 3	16	1 – 4	25	1 – 7
8	2 – 4	17	2 – 5	26	2 – 1
9	3 – 5	18	3 – 5	27	3 – 2

Слід зазначити, що в даному випадку необхідно перетворювати неоднакову кількість змінних одного коду в інший.

Для прискорення процесу мінімізації логічних функцій на практиці використовується логічний конвертор, в якому необхідно вказувати по одній логічній функції, потім перетворювати таблицю істинності в логічний вираз, а потім – в схему. Отримані таким чином елементи схем об'єднуються в одну сумарну.

Функціональна схема цифрового автомата управління індикатором наведена на рис. 4.46.

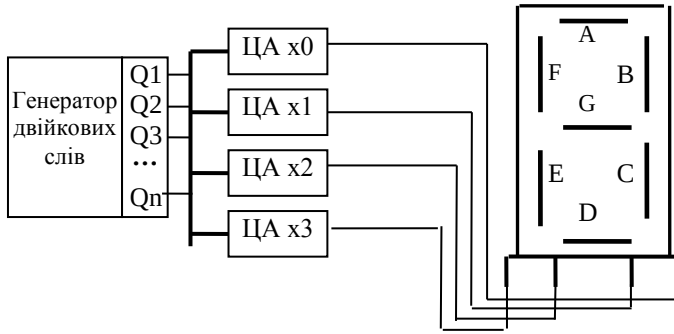


Рис. 4.46. Функціональна схема цифрового автомата управління індикатором з внутрішнім дешифратором

Приклад 1. Синтезувати кодоперетворювач, який перетворює код, заданий у табл. 4.19.

Таблиця 4.19

№ з/п	Код з надм. 3	Код Грея				№ з/п	Код з надм. 3	Код Грея			
	y3y2y1y0	X3	X2	X1	X0		y3y2y1y0	X3	X2	X1	X0
3	0011	0	0	0	0	8	1000	0	1	1	1
4	0100	0	0	0	1	9	1001	0	1	0	1
5	0101	0	0	1	1	10	1010	0	1	0	0
6	0110	0	0	1	0	11	1011	1	1	0	0
7	0111	0	1	1	0	12	1100	1	1	0	1

3.1.1. Відкриваємо EWB. Переносимо на робочий стіл зображення логічного конвертора. Подвійним натисканням на логічний конвертор відкриваємо його розширений режим відображення. Натискаємо мишкою на ту цифру, яка дорівнює кількості змінних у вибраному коді. В нашому випадку ця кількість дорівнює чотирьом (x3x2x1x0). У таблиці істинності, яка з'явилася у правій колонці, в ручну проставляємо необхідні значення вихідних змінних (рис. 4.47).

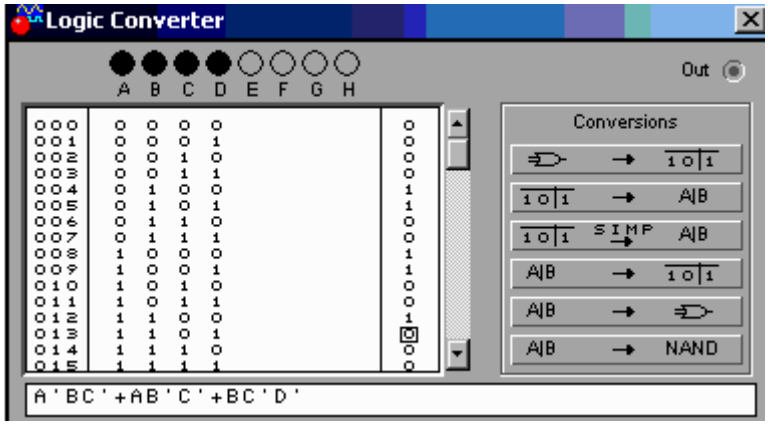
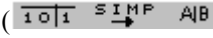


Рис. 4.47. Вікно логічного конвертора у розширеному режимі з наведеними значеннями функції x0

Для цього необхідно підвести мишку до значення функції на заданому наборі та натиснути ліву клавішу. В результаті цих дій повинен з'явитися прямокутний растр.

Натискаємо по черзі кнопки логічного конвертора, які перетворюють таблицю у мінімізований логічний вираз (), а

потім – у схему ( ).

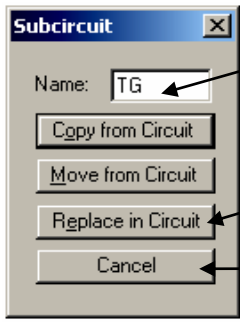


Рис. 4.48. Вікно створення макросу

3.1.2. Створити макрос.

Для створення складного елемента (макросу), необхідно виділити ті елементи, які він буде включати (рис. 4.48).

Якщо не стоїть точка на виході елемента, то її необхідно поставити з піктограми з позначенням та з'єднати з виходом схеми.

Далі вибрати меню "Circuit → Create Subcircuit" (або натиснути Ctrl + B).

Якщо точка на виході елемента своєчасно не поставлена, то треба вже після створення макросу довести лінію виходу до кінця вікна, доки не з'явиться відповідна позначка.

Приклад створення макросу змінної x1 для розглянутого прикладу наведено на рис. 4.49.

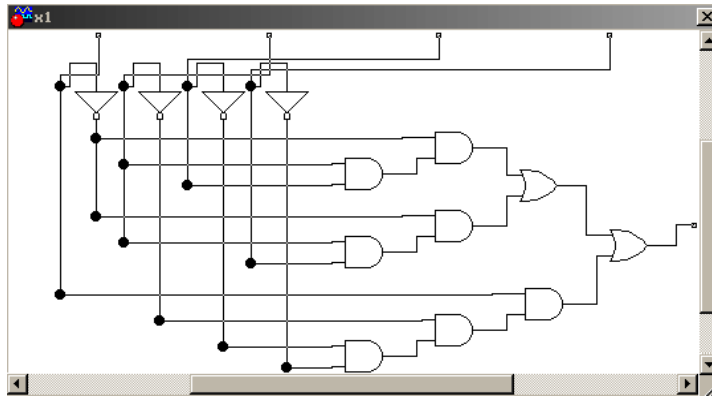


Рис. 4.49. Вікно відображення створеного макросу змінної x1

3.1.3. З'єднати вихідні сигнали з отриманих макросів з 7-сегментним індикатором без дешифратора, а входи макросів – з генератором слів.

У вікні генератора слів потрібно вручну обов'язково вказати по порядку, як вказано в таблиці кодів, номери відповідних наборів у 16-річній системі числення. Приклад з'єднання кодоперетворювача згідно з табл. 4.19 наведено на рис. 4.50.

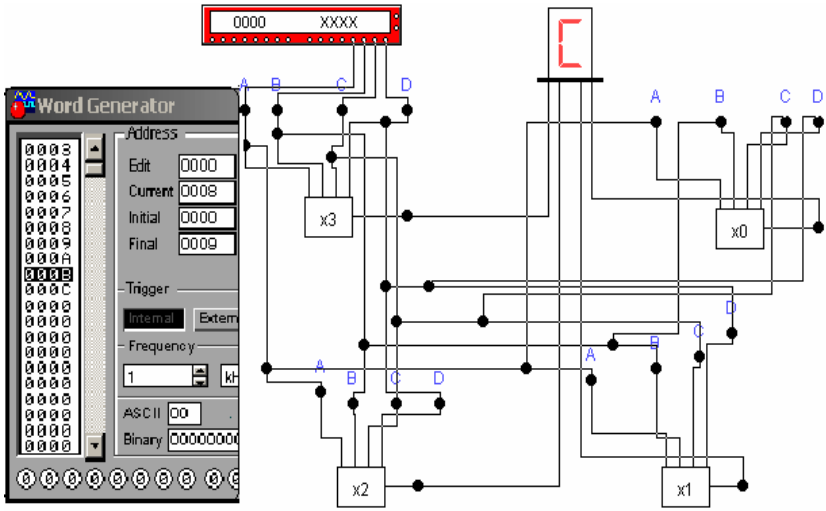


Рис. 4.50. Приклад з'єднання кодоперетворювача згідно з табл. 4.20

Приклад 2. Синтезувати кодоперетворювач, який перетворює код, заданий у табл. 4.20.

Таблиця 4.20

№ набору	Код 3321	Код 8421	№ набору	Код 3321	Код 8421
	3	1		3	1
0	0000	0000	10	1010	0101
1	0001	0001	11	1011	0110
2	0010	0010	13	1101	0111
3	0011	0011	14	1110	1000
4	0100	0100	15	1111	1001

На рис. 4.51 та 4.52 наведені таблиці істинності змінних x0, x1,x2, x3 прикладу 2.

Схема з'єднання кодоперетворювача згідно з табл. 4.20 наведена на рис. 4.53.

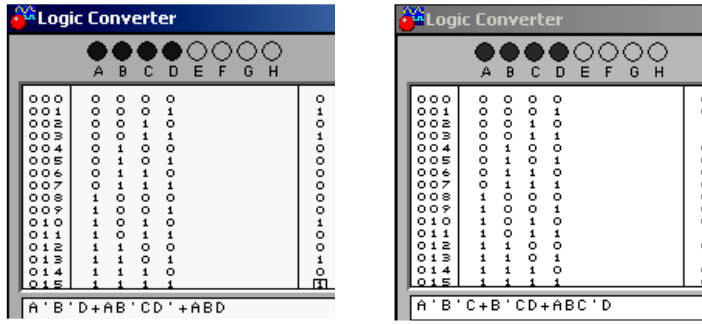


Рис. 4.51. Таблиці істинності змінних x0 та x1

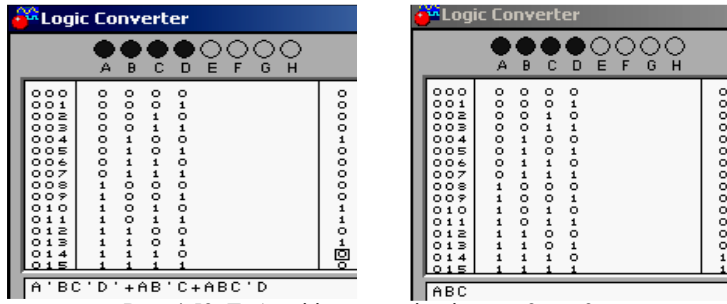


Рис. 4.52. Таблиці істинності змінних x2 та x3

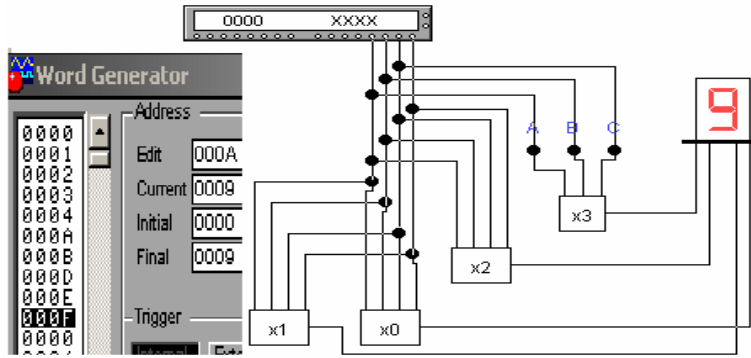


Рис. 4.53. Схема з’єднання кодоперетворювача згідно з табл. 4.20

3.2. Дослід 2. Провести дослідження суматорів згідно з варіантами (табл. 4.21).

Таблиця 4.21

№	Завдання на дослідження
1	3-розрядний СМ на повних суматорах
2	2-розрядний СМ на напівсуматорах
3	2-розрядний СМ з використанням елементів М2
4	4-розрядний СМ на повних суматорах
5	3-розрядний СМ на напівсуматорах
6	3-розрядний СМ з використанням елементів М2
7	5-розрядний СМ на повних суматорах
8	4-розрядний СМ на напівсуматорах
9	4-розрядний СМ з використанням елементів М2
10	6-розрядний СМ на повних суматорах
11	5-розрядний СМ на напівсуматорах
12	5-розрядний СМ з використанням елементів М2
13	7-розрядний СМ на повних суматорах
14	6-розрядний СМ на напівсуматорах
15	6-розрядний з використанням елементів М2

У програмі EWB арифметичні суматори наведені в бібліотеці Comt I двома базовими пристроями, наведеними на рис. 4.54; напівсуматором і повним суматором.

Рис. 4.54. Зображення напівсуматора та повного суматора в програмі EWB

Вони мають наступні призначення виводів: A, B – входи операндів; Σ – результат додавання; C_o – вихід переносу; Ci – вхід переносу. Багаторозрядний суматор створюється на базі одного напівсуматора і *n* повних суматорів. Як приклад на рис. 4.55 наведена структурна схема 2-розрядного суматора. На входи A1, A2 і B1, B2, подаються перші і другі доданки відповідно, а з виходів SI та S2 знімається результат додавання.

Схема повного суматора однорозрядного на напівсуматорах наведена на рис. 4.56.

Схема повного однорозрядного суматора на елементах mod2 наведена на рис. 4.57.

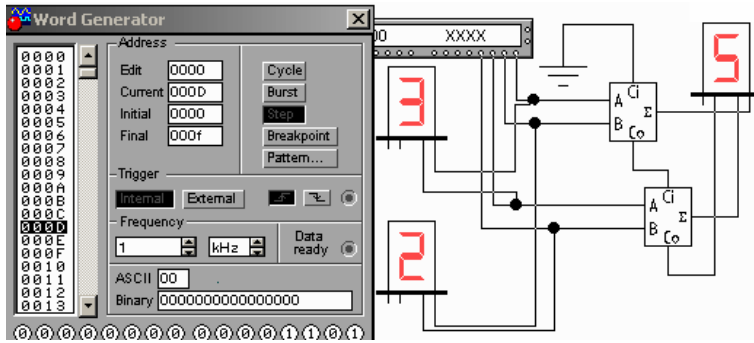


Рис. 4.55. Структурна схема 2-розрядного суматора

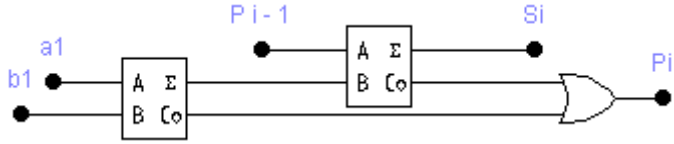


Рис. 4.56. Схема повного суматора однорозрядного
на напівсуматорах

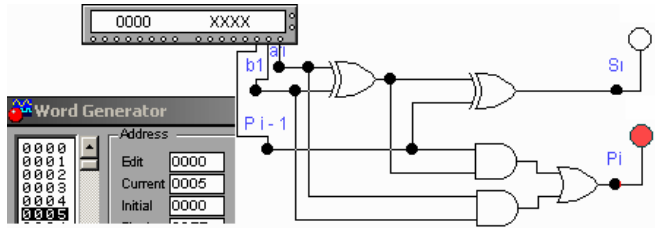


Рис. 4.57. Схема повного однорозрядного суматора на елементах mod2

Напівсуматори та логічні елементи mod 2 завжди спрощують складність схем, які виконані з використанням комбінаторної логіки.

4. *Питання для самоперевірки*

4.1. Які функції виконує кодоперетворювач та області його використовання?

4.2. За допомогою яких кнопок логічного конвертора можна провести машинну мінімізацію логічної функції?

4.3. Чим відрізняється напівсуматор від повного суматора?

5. СХЕМОТЕХНІКА ЦИФРОВИХ ВУЗЛІВ

5.1. Регістри

5.1.1. Загальна характеристика регістрів

Регістром називається цифровий пристрій, який здійснює прийом, зберігання і видачу двійкових кодів, а також виконання над ними деяких логічних операцій. Регістри призначені для короткочасного запам'ятовування двійкової інформації або її обробки за незмінними правилами.

Регістр є впорядкованою послідовністю тригерів, кількість яких відповідає кількості розрядів у слові. З кожним регістром зазвичай зв'язаний комбінаційний цифровий пристрій, за допомогою якого забезпечується виконання деяких операцій над словами.

Типовими є такі операції:

- прийом слова в регістр;
- передача слова з регістра;
- порозрядні логічні операції;
- зсув слова праворуч або ліворуч на задане число розрядів;
- перетворення послідовного коду слова в паралельний і назад;
- установка регістра в початковий стан (скидання).

Фактично будь-який цифровий пристрій можна представити у вигляді сукупності регістрів, сполучених один з одним за допомогою комбінаційних цифрових пристроїв.

Регістри класифікуються по наступних видах:

- накопичувальні (регістри пам'яті, зберігання);
- регістри зсуву.

У свою чергу регістри зсуву поділяються:

а) за способом введення-виведення інформації на:

- паралельні;
- послідовні;

- ❖ комбіновані,

б) за напрямом передачі інформації на:

- ❖ однонаправлені;
- ❖ реверсивні.

5.1.2. Регістри пам'яті

Регістри пам'яті є найпростішими вузлами за складністю їх реалізації. Їх призначення полягає у збереженні двійкового коду протягом короткого проміжку часу. Вони являють собою набір тригерів, кожний з яких зберігає розряд коду.

Прийом (запис, занесення, введення) інформації у регістри пам'яті здійснюється паралельно.

Однофазний прийом інформації у регістри на тригерах з роздільними входами (RS, JK) реалізовано в схемі на рис. 5.1, *а, в* і пояснюються на прикладі тригера першого розряду часовими діаграмами (рис. 5.1, *б*).

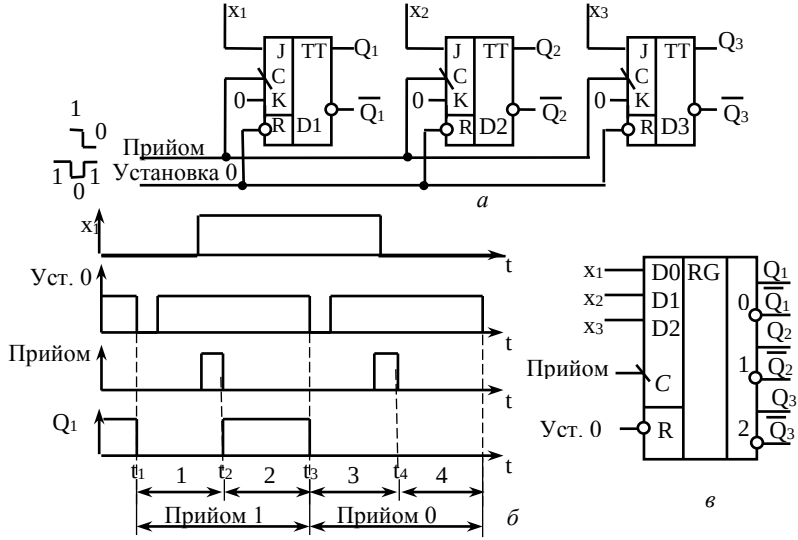


Рис. 5.1. Однофазний прийом інформації у регістр на синхронних JK-тригерах

Прийом відбувається у два такти. У першому такті (моменти t_1, t_3) видається керуючий сигнал “Установка 0”, за яким усі тригери установлюються в нульовий стан. У другому такті (моменти t_2, t_4) подається сигнал “Прийом”. Якщо в цей час розряд x_i, який приймається, дорівнює одиниці (момент t_2), то на входах відповідного тригера з'являється комбінація сигналів, яка установлює його в стан “1” (для JCK-тригера $x_J = 1$, $x_K = 0$, а для RS тригера $x_S = 0$, $x_R = 1$). При нульовому значенні x_i (моменти t_3) комбінація сигналів забезпечує зберігання вихідного нульового стану тригера (відповідно $x_J = x_K = 0$ і $x_S = x_K = 0$). Таким чином, спочатку необхідно “обнулити” тригери, а вже потім записувати в них інформацію. Недоліком таких регістрів є неможливість прийому інформації за один такт, тому вони застосовуються в цифрових пристроях з невисокою швидкодією. Цей недолік відсутній у регістрах, побудованих на D-тригерах (рис. 5.2). Для прийому інформації подається сигнал “Прийом” тобто витрачається один такт.

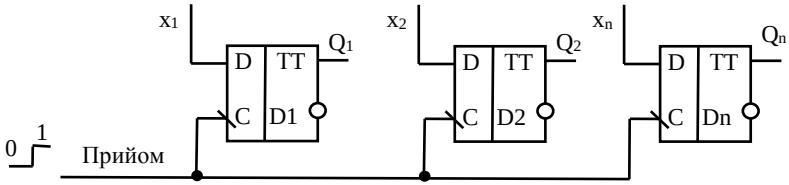


Рис. 5.2. Регістр з однофазним прийомом інформації за один такт на D-тригерах

Такі регістри широко застосовуються в інтегральній схемотехніці. Однак без використання додаткових до регістрів елементів неможливо виконувати порозрядні логічні операції.

При парафазному способі (рис. 5.3 і 5.4) прийом інформації у регістр відбуваються за сигналом “Прийом” за один такт.

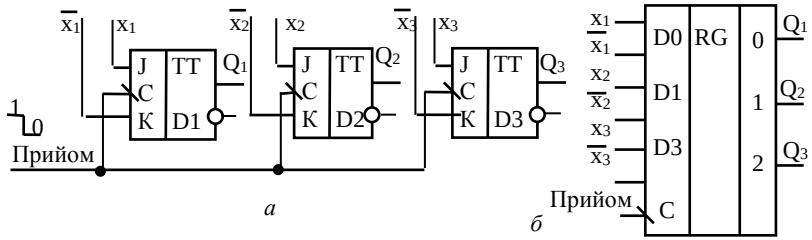


Рис. 5.3. Парафазний прийом інформації в регістрах на синхронних JK-тригерах

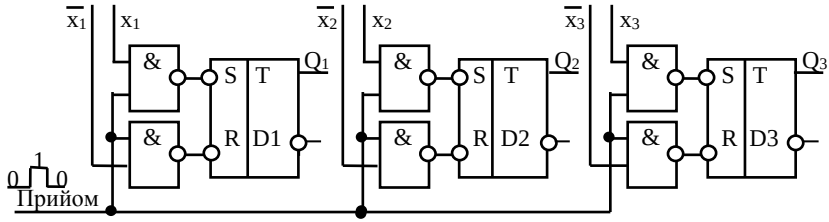


Рис. 5.4. Парафазний прийом інформації в регістр на асинхронних RS-тригерах

Тригери регістра установлюються в новий стан незалежно від попереднього, тому їх попередня установка в нульовий стан не обовязкова. Недоліком цього способу є підвищені апаратурні витрати, оскільки для прийому інформації потребується більше виводів.

У схемі, наведеній на рис. 5.5, подача сигналу дозволу видачі зворотного коду (Звор), приводить до видачі інформації у зворотному коді, а подача сигналу Пр – до видачі її у прямому коді.

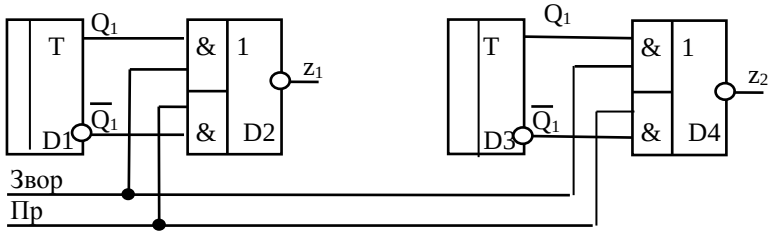


Рис. 5.5. Видача інформації з регістра в прямому або зворотному коді

При передачі кодів із регістра в регістр можна виконати порозрядні логічні операції. Наприклад, операцію **логічного додавання** можна реалізувати на RS-тригерах (рис. 5.6), один розряд якого функціонує згідно з рівнянням:

$$Q_{n+1} = S + Q_n\overline{R} = \left|\begin{matrix} S = a_n \\ R = 0 \\ Q_N = b_n \end{matrix}\right| = a_n + b_n.$$

На основі рівняння функціонування RS-тригера

$$Q_{n+1} = S + Q_n\overline{R} = \left|\begin{matrix} S = 0 \\ \overline{R} = a_n \end{matrix}\right| = a_n\, b_n.$$

можна побудувати регістр, який виконує операцію **логічного множення** (рис. 5.7).

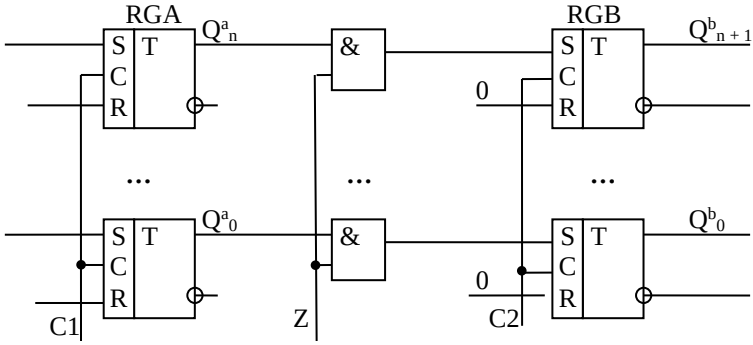


Рис. 5.6. Схема виконання операції логічного додавання

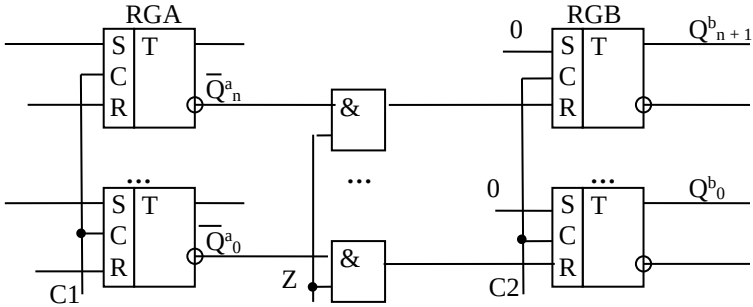


Рис. 5.7. Схема виконання операції логічного множення

Операцію порозрядного додавання за mod2 можна виконати на лічильних тригерах (Т-тригерах). На основі рівняння Т-тригера

$$Q_{n+1} = \overline{T}Q_n + T\overline{Q}_n = \left| \begin{matrix} T = a_n \\ Q_n = b_n \end{matrix} \right| = a_n \overline{b}_n \oplus \overline{a}_n b_n.$$

будується (рис. 5.8) схема регістра, який виконує операцію mod2.

5.1.3. Регістри зсуву

Регістри зсуву складають основну масу регістрів, які використовуються на практиці. Вони здатні переміщувати всі розряди записаного

в них коду вправо (у бік старших розрядів) або вліво (у бік молодших розрядів) на однакову кількість розрядів, зазвичай, на один розряд за один такт.

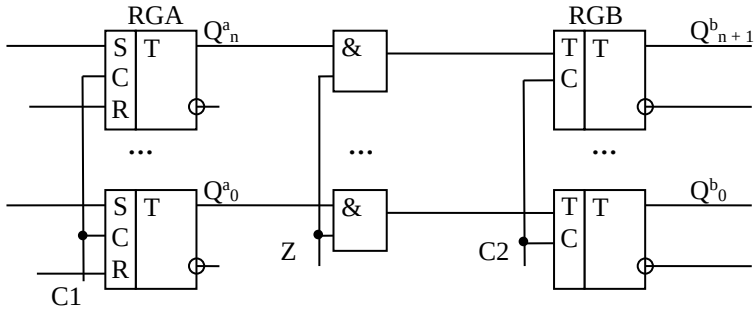


Рис. 5.8. Схема виконання операції mod2

Для передачі інформації з розряду в розряд між тригерами регістра зсуву організуються спеціальні зв'язки.

У регістрах без зворотних зв'язків розряд коду, що вийшов з регістра вправо або вліво, зникає, а у заряд, який звільнився, може бути записано нове значення. Воно визначається сигналами на входах крайнього лівого тригера при зсуві вправо або крайнього правого тригера при зсуві вліво. Тим самим при зсувах можна здійснювати послідовне занесення розрядів нового коду в регістр і одночасно видавати з регістра в послідовному коді ту інформацію, яка в ньому зберігалася. У технічній літературі вхід для послідовного занесення інформації при зсуві вправо позначається як DR (від англ. right), при зсуві вліво – як DL (від англ. left). Виходи, через які відбувається послідовна видача інформації, спеціального позначення не мають, оскільки для цього використовується вихід крайнього розряду: при зсуві вправо – правого, при зсуві вліво – лівого.

У регістрі зсуву зі зворотними зв’язками існують ланцюги зв’язку між старшими і молодшими розрядами. Найпростіший регістр зсуву – кільцевий – отримується шляхом замикання звичайного регістра зсуву в кільце. Більш складні регістри зі зворотними зв’язками відносяться до спеціальних пристроїв і розглядаються у спеціальній літературі.

Оскільки в момент зсуву тригер *i*-го розряду повинен передати інформацію, яка зберігається в *i* + 1-й розряд і одночасно прийняти інформацію з *i* – 1-го розряду, то при потенційній системі елементів обидві

ці операції не можуть бути виконані на однотактному тригері. Тому для побудови регістрів зсуву використовуються двотактні тригери.

На рис. 5.9 зображені схеми регістрів із зсувом управо, побудовані на різних тригерах.

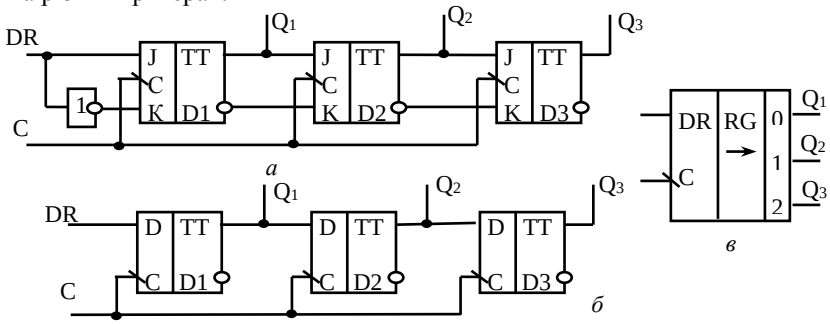


Рис. 5.9. Регістри зі зсувом управо: *а* – на JK-тригерах; *б* – на D-тригерах; *в* – умовне позначення

У них по передньому фронту (зміна з 0 на 1) синхроімпульсу на вході С відбувається прийом інформації з кожного *i*-го тригера в *i* + 1-й тригер, розташованого праворуч від нього. По задньому фронту синхроімпульсу (зміна з 1 на 0) відбувається прийом цієї інформації у тригер. Цей процес відбувається у всіх тригерах одночасно (уся інформація в регістрі зсувається вправо на один розряд).

Правий розряд, що надходить з виходу Q_3, втрачається, а в лівий розряд записується інформація, що надійшла в момент спаду синхросигналу на вхід послідовного занесення DR. Інформація, яка зберігалася в регістрі раніше, у процесі виконання зсувів буде надходити на вихід Q_3 у послідовному коді.

Регістр зі зсувом вліво будується і функціонує аналогічно (рис. 5.10).

Послідовне занесення інформації до нього відбувається через вхід DL, а її видача – через вихід Q_1.

Реверсивний регістр зсуву (рис. 5.11) у міжрозрядних колах містить елементи комутації, які під впливом сигналу на спеціальному керуючому вході (звичайно керуючі входи позначаються через V) сполучають тригери в схему зі зсувом управо або вліво.

При V = 1 з виходу інвертора D7 нижні вентилі цих елементів замикаються, а верхні відмикаються.

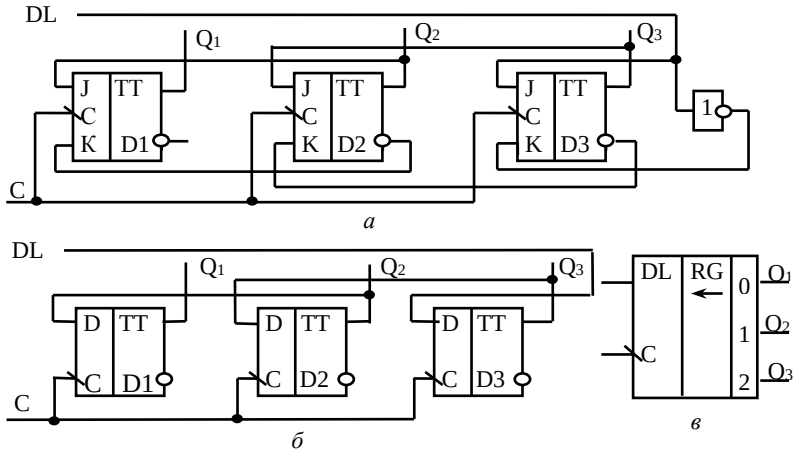


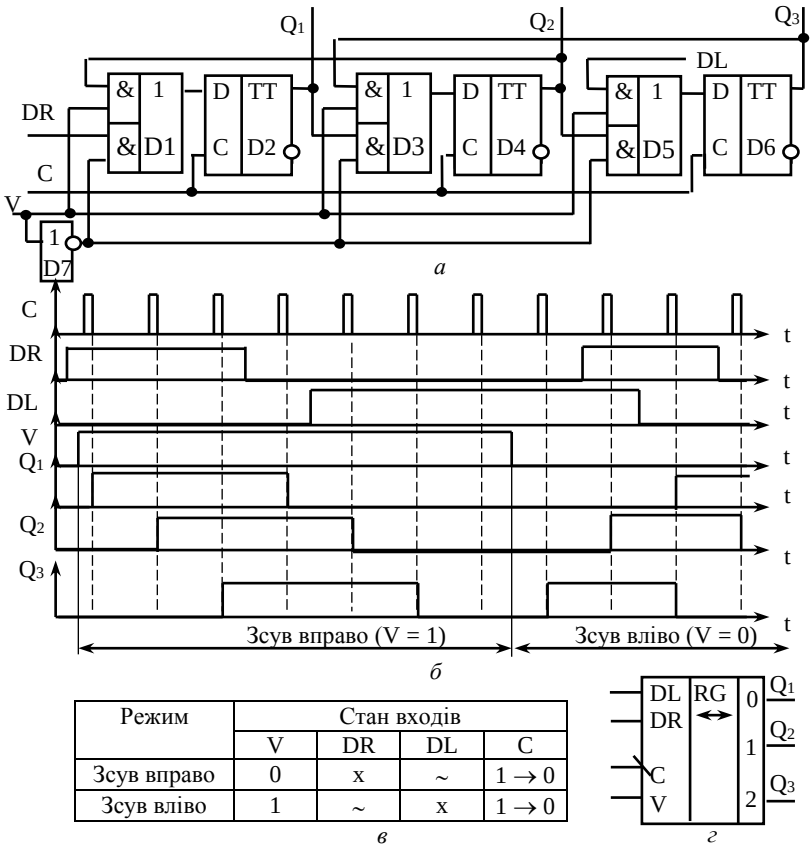
Рис. 5.10. Регістри зі зсувом вліво: *а* – на JK-тригерах; *б* – на D-тригерах; *в* – умовне позначення

У результаті на D-вхід кожного тригера подається сигнал з виходу Q тригера, який знаходиться ліворуч від нього.

При V = 0 верхні вентилі елементів І-АБО замикаються, а нижні вентилі відмикаються. Тому на D-вхід кожного тригера буде надходити сигнал з виходу Q тригера, який знаходиться від нього праворуч, що відповідає регістру зі зсувом уліво.

При фіксованому значенні сигналу на вході V схема функціонує аналогічно відповідному регістру. Зміна цього сигналу приводить до реверсу, тобто до зміни напряму зсуву.

На рис. 5.11, *б* зображені часові діаграми, що поясняють функціонування регістра, усі розряди якого в момент $t = 0$ знаходиться у стані “0”. На рис. 5.11, *в* зображена таблиця режимів роботи цього регістра. У ній символ ~ (тільда) означає, що сигнали на вході в даний момент на схему не впливають і тому їх логічне значення байдуже. Символ *x* означає, що стан (*x*) даного входу у випадку, який розглядається, буде впливати на схему і повинен враховуватися при аналізі її функціонування. Символ “1 → 0” показує, що сигнал на даному вході впливає на роботу схеми тільки в момент зміни стану з “1” на “0”, що характерно для інверсних динамічних входів.



Режим	Стан входів			
	V	DR	DL	C
Зсув вправо	0	x	~	$1 \to 0$
Зсув вліво	1	~	x	$1 \to 0$

Рис. 5.11. Реверсивний регістр зсуву: *а* – схема;
б – часові діаграми роботи; *г* – умовне позначення; *в* – таблиця режимів роботи

***Приклад* 5.1.** Побудувати схему 3-розрядного регістра, який створено на D-тригерах та виконує операції: установка в “0”; зсув слова на один розряд вліво; зсув слова на один розряд вправо; паралельний прийом інформації.

У зв’язку з тим, що необхідно реалізувати 4 операції, то кількість управляючих сигналів визначається згідно з виразом

$$n_{упв} = \rceil N_{оп} \lceil,$$

де $N_{оп}$ – кількість операцій регістра.

Поставимо у відповідність типам операцій значення управляючих сигналів:

00 – установка в “0”;

01 – зсув управо на один розряд;

10 – зсув уліво на один розряд;

11 – паралельний прийом інформації.

Для виконання послідовного управління регістром можна використати як дешифратор, так і мультиплексор. Схема багатофункціонального регістра, призначеного для виконання необхідних операцій з використанням дешифратора наведена на рис. 5.12.

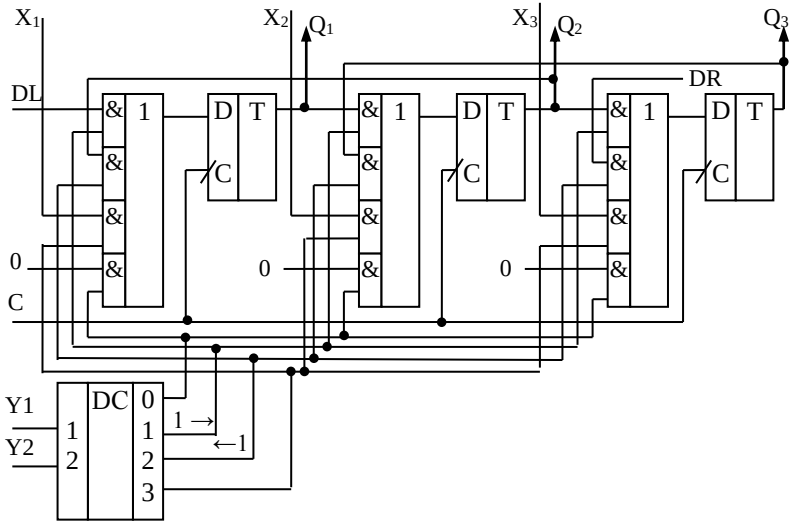


Рис. 5.12. Схема багатофункціонального регістра, призначеного для виконання необхідних операцій з використанням дешифратора до прикладу 5.1

***Приклад* 5.2.** Побудувати багатофункціональний регістр, який виконує операції: паралельний прийом інформації; зсув ліворуч на 1 розряд; зсув праворуч на 2 розряди; скидання регістра.

Схема багатофункціонального регістра, призначеного для виконання необхідних операцій з використанням дешифратора наведена на рис. 5.13.

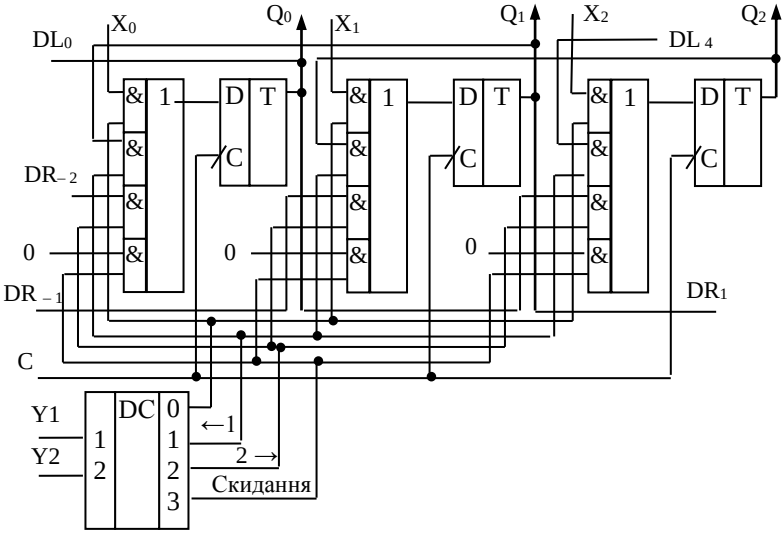


Рис. 5.13. Схема багатофункціонального регістра, призначеного для виконання необхідних операцій з використанням дешифратора до прикладу 5.2

5.1.4. Завдання до самостійних досліджень регістрів

1. *Мета*

Ознайомлення з роботою регістрів за допомогою інструментальних засобів цифрової частини пакета EWB, закріплення теоретичного матеріалу, набуття навиків створення і моделювання цифрових пристроїв.

2. *Теми для попереднього опрацювання*

2.1. Регістри

3. *Завдання*

3.1. **Дослід 1.** Паралельний запис коду до регістра по входах попередньої установки

3.1.1. Зібрати схему паралельного регістра на JK-тригерах у пакеті EWB 5.12.

Заповнити двійковим кодом комірки генератора слів. Для цього в нульову комірку проставити всі нулі, в першу комірку – код свого номера за списком в групі, в другу комірку – № + 3, у четверту – код “об-

нуління” всіх тригерів регістра. Приклад заповнення генератора слів та схема дослідження регістра на JK-тригерах наведені на рис. 5.14. “Обнуління” всіх тригерів регістра виконується на 4-му такті роботи генератора слів (0010_{16}).

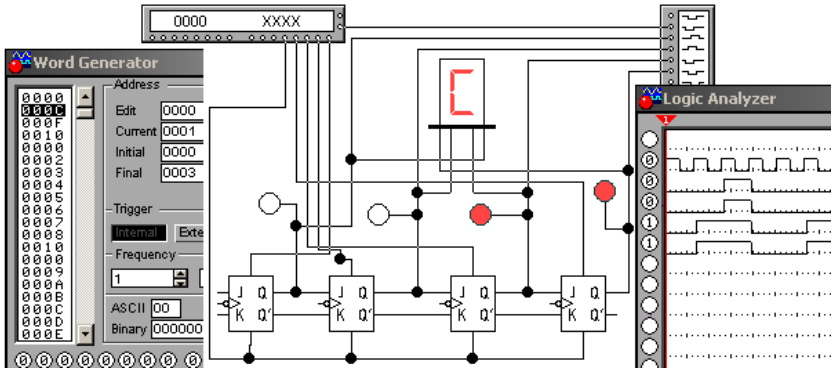


Рис. 5.14. Схема дослідження регістра на JK-тригерах по входах попередньої установки

3.2. **Дослід 2**. Послідовний запис коду в регістр зсуву по інформаційних входах.

3.2.1. Зібрати схему послідовного регістра, яка наведено на рис. 5.15.

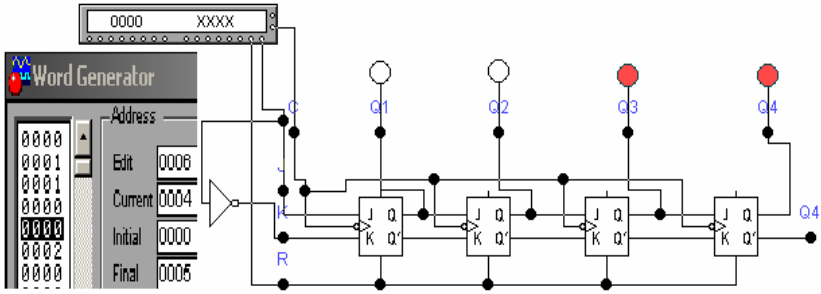


Рис. 5.15. Схема дослідження регістра зсуву на JK-тригерах з послідовним записом

За допомогою генератора слів у послідовний регістр записати свій номер за порядком в групі. Наприклад, необхідно записати код номера $12_{10} = C_{16}$. Для цього в нульову комірку генератора слів

записуються всі нулі, в першу – вага старшого розряду, тобто 0001, в другу – теж 0001, а в третю та четверту – всі нулі (рис. 5.15). Таким чином до генератора слів послідовно записано код C_{16}.

3.3. **Дослід 3.** Створити макрос зі схеми досліду 2, дослідити та побудувати 8-розрядний регістр зсуву вправо.

3.3.1. Для створення макросу зі схеми досліду 2 необхідно проставити точку на тих виводах макросу, які потрібні. Виділити точку, нажати праву клавішу мишки, ввійти до меню: Open – Label та проставити ім'я. Потім виділити необхідні елементи, натиснути на панелі інструментів піктограму Circuit, потім – Cubcircuit та вказати ім'я, наприклад: RG_JK. Схема з'єднання елементів до макросу RG_JK наведена на рис. 5.16.

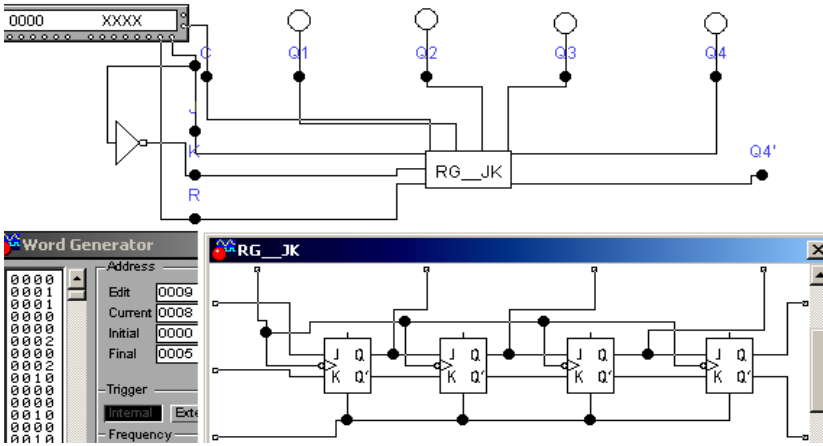


Рис. 5.16. Схема з'єднання елементів до макросу RG_JK

3.3.2. Виберіть другий макрос із піктограми з ім'ям Favorites та з'єднайте їх так, як наведено на рис. 5.17. Запишіть у генератор слів таку послідовність, яка б дозволяла відобразити на світлодіодах два однакових числа, які відповідають номеру за порядком в групі.

3.4. **Дослід 4.** Дослідження функціонування регістра, який побудовано на конкретних мікросхемах.

У відповідності з варіантом з табл. 5.1 вибрати мікросхему та побудувати регістр. Навести функціональну схему та часові діаграми його функціонування.

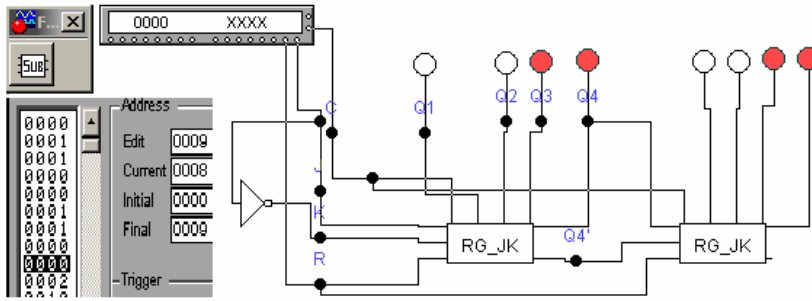


Рис. 5.17. Схема 8-розрядного регістра на основі макросу RG_JK

3.5. **Дослід 5.** Дослідити регістр згідно з варіантом, який наведено в табл. 5.2.

Навести функціональну схему та часові діаграми його функціонування.

Таблиця 5.1

№ з/п	Серія SN74	Вітчизняні МС	Функціональне призначення
1	7472	155ТВ1	JK-тригер з елементом 3І на виходах
2	7474	155ТМ2	2 D-тригери
3	7475	155ТМ7	4 D-тригери із прямими та інверсними виходами
4	7476	155ТВ7	2 JK-тригери
5	7477	155ТМ5	4 D-тригери із прямими виходами
6	7478	133ТВ14	2 JK-тригери
7	74107	155ТВ6	2 JK-тригери із роздільною установкою нуля
8	74109	155ТВ15	2 JK-тригери
9	74112	155ТВ9	2 JK-тригери
10	74113	155ТВ10	2 JK-тригери із поперед. установкою нуля та одиниці
11	74114	155ТВ11	2 JK-тригери із поперед. установ. 0 або 1 та “обнулінням”
12	74174	155ТМ9	6 D-тригерів
13	74175	155ТМ8	4 D-тригери
14	74279	555ТР2	4 RS-тригери із защіпкою

Таблиця 5.2

№ з/п	Серія SN74	МС	Функціональне призначення
1	7491	134ИР2	8-розр. регістр зсуву
2	74164	155ИР8	8-розр. регістр зсуву з паралельними виходами
3	74165	555ИР9	8-розр. регістр зсуву з паралельним введенням інформації
4	74166	555ИР10	8-розр. регістр зсуву з синхронним паралельним введенням
5	74173	155ИР15	4-розр. регістр із трьома станами
6	74194	155ИР11	4-розр. універсальний регістр
7	74195	155ИР12	4-розр. регістр зсуву із паралельним введенням
8	74198	155ИР13	8-розр. універсальний регістр зсуву
9	74273	155ИР35	8-розр. регістр із установкою 0
10	74373	155ИР22	8-розр. буф. регістр із 3 станами та потенціальним управлінням
11	74374	155ИР23	8-розр. буф. регістр із 3 станами та імпульсним управлінням
12	74377	155ИР27	8-розр. регістр із дозволом запису
13	74395	533ИР25	4-розр. паралельний регістр зсуву

4. *Питання для самоперевірки*

4.1. Що таке регістр та які функції він може виконувати?

4.2. Наведіть типи регістрів та їх можливе використання.

4.3. Побудувати багатофункціональний регістр, який виконує операції: паралельний прийом інформації; зсув ліворуч на 1 розряд; зсув праворуч на 2 розряди; скидання регістра.

4.4. Побудувати багатофункціональний регістр, який виконує операції: паралельний прийом інформації; зсув ліворуч на 2 розряди; зсув праворуч на 2 розряди; скидання регістра.

4.5. Побудувати багатофункціональний регістр, який виконує операції: паралельний прийом інформації; зсув ліворуч на 2 розряд; зсув праворуч на 1 розряд; скидання регістра.

5.2. Лічильники

5.2.1. Загальна характеристика лічильників. Принципи побудови та функціонування двійкових лічильників

Лічильником називається цифровий пристрій, сигнали на вході якого у певному коді відображають кількість вхідних імпульсів, що

надійшли. Основу лічильника складають тригери, сполучені ланцюгами переносу інформації з розряду в розряд. ***Призначення лічильників*** полягає в підрахунку кількості різних подій, які фізично представляються потенційними або імпульсними сигналами, а також у поділі частоти проходження цих сигналів. В останньому випадку лічильник називається лічильником-дільником.

Галузь застосування. Лічильники, як і регістри, відносяться до найбільш широко розповсюджених типових цифрових пристроїв і застосовуються як перетворювачі кількості сигналів у певний код, дільники частоти, пристрої додавання або віднімання кількості сигналів. Вони застосовуються для побудови розподілювачів сигналів, цифрових фазоперетворювачів тощо.

Експлуатаційними характеристиками лічильників є:

– ***коефіцієнт перерахування*** (модуль рахування) $К_{лч}$, який визначає кількість можливих стійких станів лічильника, що циклічно змінюються під впливом сигналів рахунку. Повернення лічильника в початковий стан свідчить про його переповнення. Тому, якщо лічильник знаходився в початковому стані і сигнал рахування надійшов N-разів, то при $N > К_{лч}$ за станом лічильника можна визначити лише остачу від ділення N на $К_{лч}$, тобто рахування ведеться за модулем $К_{лч}$ (звідси і назва “модуль рахування”). При використанні лічильника як дільника коефіцієнт перерахування називають коефіцієнтом ділення;

– ***ємність***, що визначається максимальною кількістю сигналів, яка може бути зафіксована на лічильнику. Ємність на одиницю менше коефіцієнта перерахування, оскільки початковий стан лічильника не враховується;

– ***час установки*** $t_{уст}$, який визначається інтервалом часу між моментами надходження вхідного сигналу і моментом закінчення найтривалішого перехідного процесу в схемі, іншими словами, – максимальним часом установки на лічильнику нового коду після надходження сигналу рахування;

– ***мінімальний період проходження вхідних сигналів*** $t_{дозв}$, при якому ще не виникають перебої в роботі. Обернена величина називається максимальною частотою лічби.

Класифікація лічильників здійснюється за такими ознаками:

- за напрямком рахування;
- за способом побудови міжрозрядних кіл переносу;
- за коефіцієнтом перерахування;
- за способом організації роботи тригерів.

1. За напрямком рахування лічильники поділяються на підсумовуючі, віднімаючі і реверсивні.

У підсумовуючому лічильнику кожний сигнал рахування збільшує число, записане на лічильнику, на одиницю (табл. 5.3.). Такий напрямок (порядок) рахування називається прямим. При цьому перенесення одиниці в старший розряд відбувається при зміні стану молодшого розряду з "1“ на “0”.

Таблиця 5.3

№ сигналу	Стан розрядів			Число на лічильнику
	Q_3	Q_2	Q_1	
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7
8	0	0	0	0

У віднімаючому лічильнику кожен сигнал рахування зменшує вміст лічильника на одиницю (табл. 5.4). Його переповнення відбувається після установки в нульовий стан. Перенесення нуля у старший розряд відбувається при зміні стану молодшого розряду з “0” на “1”.

Таблиця 5.4

№ сигналу	Стан розрядів			Число на лічильнику
	Q_3	Q_2	Q_1	
0	0	0	0	0
1	1	1	1	7
2	1	1	0	6
3	1	0	1	5
4	1	0	0	4
5	0	1	1	3
6	0	1	0	2
7	0	0	1	1
8	0	0	0	0

Реверсивний лічильник може функціонувати як підсумовуючий або віднімаючий. Режим роботи визначається сигналом на додатковому керуючому вході.

При використанні лічильника як дільника напрямок рахування не має значення.

2. За способом побудови міжрозрядних ланцюгів переносу лічильники бувають з послідовним переносом, з наскрізним переносом, з груповим переносом.

3. За коефіцієнтом перерахування розрізняють лічильники зі змінним і з постійним коефіцієнтом перерахування. В останньому випадку вони можуть бути двійкові ($К_{Лч} = 2^n$, n – кількість розрядів), десяткові ($К_{Лч} = 10$) або з іншим коефіцієнтом перерахування.

Недвійковий коефіцієнт перерахування забезпечується введенням у лічильник зворотних зв'язків.

4. За способом організації роботи тригерів лічильники можуть бути:

- асинхронними, у яких для тригера молодшого розряду керуючим сигналом є сигнал рахування, а для кожного наступного тригера – потенційний сигнал з виходу попереднього тригера;
- синхронними, у яких для тригера кожного розряду керуючим сигналом є сигнал рахування, а сигнал з виходу тригера попереднього розряду є інформаційним.

Для побудови лічильників використовуються Т-тригери, а також JK- і D-тригери, які працюють у лічильному режимі.

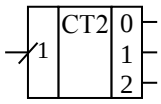


Рис .5.18. Умовне графічне позначення 3-розрядного лічильника

На схемах лічильники позначаються символом СТ (від англ. counter – лічильник). Умовне графічне позначення двійкового лічильника наведено на рис. 5.18. При необхідності праворуч від символу вказують у вигляді числа коефіцієнт перерахування .

5.2.2. Лічильники з послідовним перенесенням

Лічильники з послідовним перенесенням є ланцюжком тригерів, в якому імпульси, що підлягають рахуванню надходять на вхід першого тригера, а сигнал перенесення передається послідовно від одного розряду до іншого. В цих лічильниках використовуються асинхронні Т-тригери з прямим або з інверсним управлінням, а також JK- і D-тригери в рахунковому режимі. Функціональна схема 3-розрядного

підсумовуючого лічильника та часові діаграми його роботи наведені на рис. 5.19.

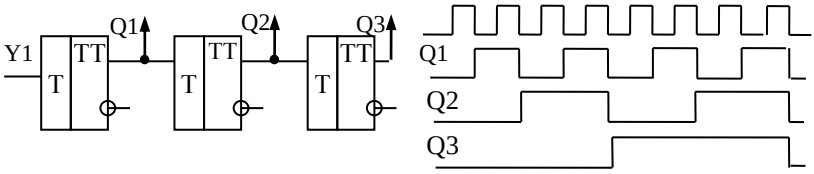


Рис. 5.19. Функціональна схема 3-розрядного підсумовуючого лічильника та часові діаграми його роботи

Функціональна схема 3-розрядного віднімаючого лічильника та часові діаграми його роботи наведені на рис. 5.20.

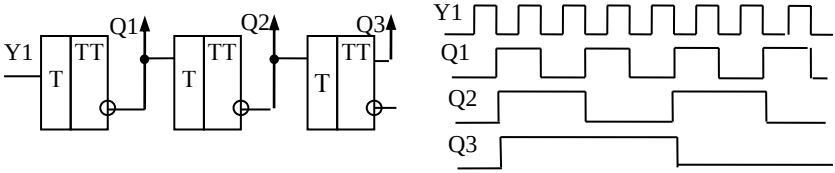


Рис. 5.20. Функціональна схема 3-розрядного віднімаючого лічильника та часові діаграми його роботи

Головною перевагою лічильників з послідовним перенесенням є простота схеми. Збільшення розрядності (нарощування) здійснюється підключенням потрібної кількості тригерів до виходу останнього тригера. Оскільки вхідні сигнали надходять на вхід тільки першого тригера, такий лічильник недостатньо навантажує попередній каскад.

Функціональна схема реверсивного лічильника наведена на рис. 5.21.

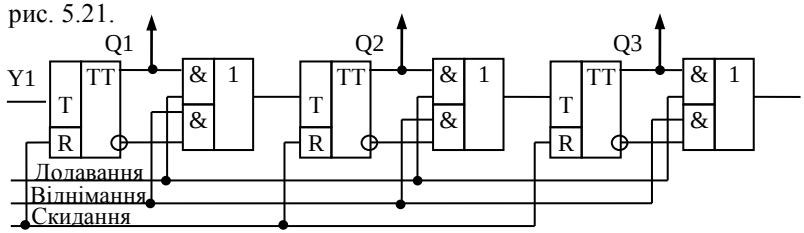


Рис. 5.21. Функціональна схема 3-розрядного реверсивного лічильника

Основний недолік лічильників з послідовним перенесенням – порівняно низька швидкодія, оскільки тригери тут спрацьовують послідовно, один за одним. Другий недолік, обумовлений цією ж причиною, полягає в тому, що через накопичення тимчасових зсувів у розрядах на виходах дешифраторів таких лічильників можуть з'являтися короткочасні помилкові імпульси, особливо помітними вони є на високих частотах.

Максимальна частота лічби визначається режимом роботи.

5.2.3. Лічильники з наскрізним перенесенням

Функціональна схема лічильника з наскрізним перенесенням наведена на рис. 5.22.

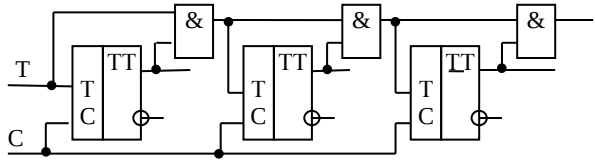


Рис. 5.22. Функціональна схема лічильника з наскрізним перенесенням

Крізне перенесення забезпечується додатковими логічними елементами 2І. Час занесення інформації в лічильник визначається як

$$t_{зан} = t_{Тг} + (n-1)t_{сх.\,І}.$$

Недоліком лічильників з наскрізним перенесенням є те, що при великій кількості розрядів час затримки розповсюдження сигналу різко збільшується.

5.2.4. Лічильники з паралельним перенесенням

Лічильники з паралельним перенесенням складаються з синхронних тригерів. Рахункові імпульси подаються одночасно на всі тактові входи, а кожний з тригерів кола служить по відношенню до подальших тільки джерелом інформаційних сигналів. Спрацьовування тригерів паралельного лічильника відбувається синхронно, і затримка перемикання всього лічильника дорівнює затримці для одного тригера (рис. 5.23).

Час занесення інформації в лічильник визначається як:

$$t_{зан} = t_{Тг} + t_{сх.\,І}.$$

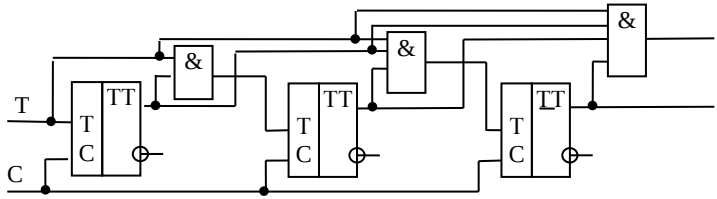


Рис. 5.23. Функціональна схема лічильника з паралельним перенесенням

5.2.5. Лічильники з довільним коефіцієнтом лічби

Усі розглянуті вище лічильники були двійковими, тобто їх коефіцієнт лічби дорівнював ступеню числа два (за модулем 2^n). Це означає, що після кожного циклу, який містить 2^n сигналів рахування, на виходах останнього тригера виникали перепади напруги, тобто відбувався поділ частоти цих сигналів (або просто їх кількості) з коефіцієнтом $К_{Лч} = 2^n$. Однак на практиці часто виникає необхідність їх поділу з недвійковим коефіцієнтом.

Розрізняють способи побудови лічильників за модулем із коефіцієнтом лічби, який не дорівнює 2^n. До цих способів слід віднести:

- виключення кінцевих станів;
- виключення початкових станів;
- синтез лічильників;
- комбінаційне з'єднання тригерів.

При побудові лічильників із виключенням останніх станів слід в момент отримання на тригерах необхідного числа за рахунок зворотних зв'язків "обнулити" усі тригери. Приклад функціональної схеми лічильника з коефіцієнтом рахунку, який дорівнює 10, наведено на рис. 5.24.

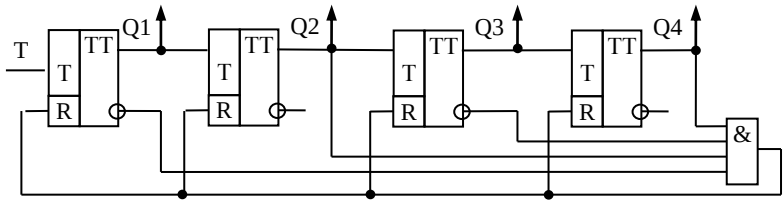


Рис. 5.24. Функціональна схема лічильника з $К_{Лч} = 10$
із виключенням кінцевих станів

При побудові лічильників із виключенням початкових станів необхідно виконати таке (наприклад, для $К_{ЛЧ} = 10_{10}$):

– визначити розрядність лічильника *n*:

$$n = \lceil \log 10 \rceil = 4;$$

– визначимо кількість виключних станів:

$$m = 2^n - К_{ЛЧ} = 16 - 10 = 6_{10} = 0110_2.$$

Функціональної схеми лічильника з коефіцієнтом рахунку, який дорівнює 10, із виключенням початкових станів, наведено на рис. 5.25.

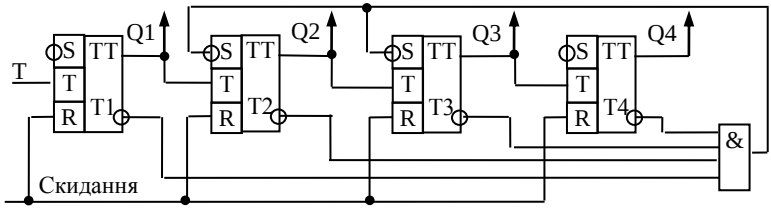


Рис. 5.25. Функціональна схема лічильника з $К_{ЛЧ} = 10$ із виключенням початкових станів

Після подання сигналу "Скидання" на усіх виходах Q1 – Q4 тригерів Т1 – Т4 установляться логічні нулі, а з інверсних виходів через час затримки елемента 4І потенціал логічних одиниць переведе тригера Т2 та Т3 в одиничні стани.

5.2.6. Синтез лічильників

Наприклад, необхідно синтезувати лічильник з $К_{ЛЧ} = 5$.

У відповідності з таблицею переходів Т-тригера (табл. 5.5) побудуємо таблицю функціонування лічильника (табл. 5.6).

Таблиця 5.5

Переходи	T
0 → 0	0
0 → 1	1
1 → 0	1
1 → 1	0

Таблиця 5.6

№	Стан Q_{n-1}			Стан Q_n			Виходи тригерів		
з/п	Q1	Q2	Q3	Q1	Q2	Q3	T1	T2	T3
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	0	1	1	0	0	1
3	0	1	1	1	0	0	1	1	1
4	1	0	0	0	0	0	1	0	0

Перенесемо значення до карт Карно та промінімізуємо їх:

Q_1 \ Q_2Q_3	00	01	11	10
0	0	0	1	0
1	1	x	x	x

$T_1 = Q_1 + Q_2Q_3$

Q_1 \ Q_2Q_3	00	01	11	10
0	0	1	1	0
1	0	x	x	x

$T_2 = Q_3$

Q_1 \ Q_2Q_3	00	01	11	10
0	1	1	1	1
1	0	x	x	x

$T_3 = \overline{Q}_1$

Функціональна схема синтезованого лічильника наведена на рис. 5.26.

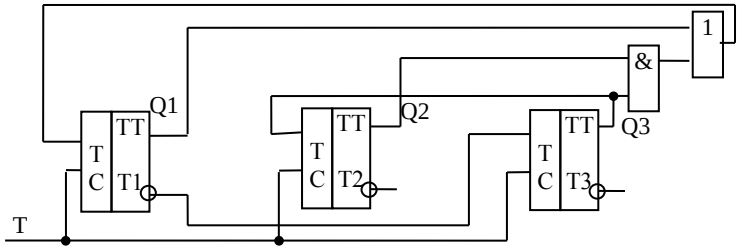


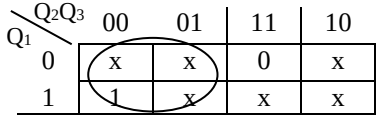
Рис. 5.26. Функціональна схема синтезованого лічильника з $K_{лч} = 5$

Приклад 5.3. Необхідно синтезувати лічильник на ***RS-тригерах*** із коефіцієнтом лічби K = 5.

У відповідності з таблицею переходів RS-тригера (табл. 5.7) побудуємо таблицю функціонування лічильника (табл. 5.8).

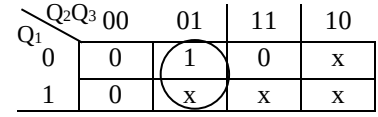
Q_1 \ Q_2Q_3	00	01	11	10
0	0	0	1	0
1	0	x	x	x

$S_1 = Q_2Q_3$



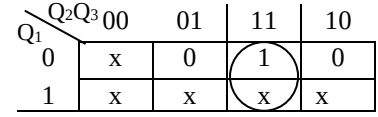
Q_1 \ Q_2Q_3	00	01	11	10
0	x	x	0	x
1	1	x	x	x

$R_1 = \overline{Q}_2$



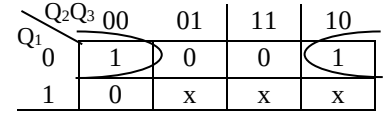
Q_1 \ Q_2Q_3	00	01	11	10
0	0	1	0	x
1	0	x	x	x

$S_2 = \overline{Q}_2 Q_3$



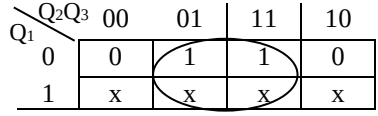
Q_1 \ Q_2Q_3	00	01	11	10
0	x	0	1	0
1	x	x	x	x

$R_2 = Q_2 Q_3$



Q_1 \ Q_2Q_3	00	01	11	10
0	1	0	0	1
1	0	x	x	x

$S_3 = \overline{Q}_1 \overline{Q}_3$



Q_1 \ Q_2Q_3	00	01	11	10
0	0	1	1	0
1	x	x	x	x

$R_3 = Q_3$

Таблиця 5.7

Переходи	S	R
0 → 0	0	x
0 → 1	1	0
1 → 0	0	1
1 → 1	x	0

Таблиця 5.8

№	Стан Q_{n-1}			Стан Q_n			Виходи тригерів					
з/п	Q1	Q2	Q3	Q1	Q2	Q3	S1	R1	S2	R2	S3	R3
0	0	0	0	0	0	1	0	x	0	x	1	0
1	0	0	1	0	1	0	0	x	1	0	0	1
2	0	1	0	0	1	1	0	x	x	0	1	0
3	0	1	1	1	0	0	1	0	0	1	0	1
4	1	0	0	0	0	0	0	1	0	x	0	x

5.2.7. Завдання до самостійних досліджень лічильників

1. *Мета*

Ознайомлення з роботою лічильників за допомогою інструментальних засобів цифрової частини пакета EWB, закріплення теоретичного матеріалу, набуття навиків створення і моделювання цифрових пристроїв.

2. *Теми для попереднього опрацювання*

2.1. Лічильники

3. *Завдання*

3.1. Дослід 1. Побудувати лічильник на D-тригерах із послідовним переносом та зняти часові діаграми його роботи. Навести функціональну схему із під'єднаним 7-сегментним індикатором та вікна з розширеними режимами генератора слів і логічного аналізатора.

Функціональна схема послідовного лічильника на D-тригерах наведена на рис. 5.27.

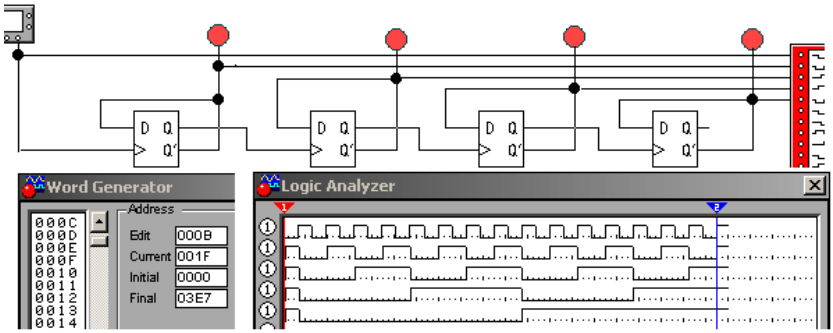


Рис. 5.27.Функціональна схема послідовного лічильника на D-тригерах

3.2. Дослід 2. Побудови лічильник на JK-тригерах із виключенням кінцевих станів за модулем К = № + 10, де № – номер за списком у групі.

Приклад функціональної схеми лічильника із виключенням останніх станів за модулем К = 10 на JK-тригерах наведена на рис. 5.28.

Приклад функціональної схеми лічильника із виключенням останніх станів за модулем К = 21 на JK-тригерах наведена на рис. 5.29.

3.3. Дослід 3. Синтезувати лічильник на тригерах згідно з варіантом (табл. 5.9).

Приклад 5.4. Необхідно синтезувати лічильник на ***D-тригерах*** із коефіцієнтом лічби К = 5.

У відповідності з таблицею переходів D-тригера (табл. 5.10) побудуємо таблицю функціонування лічильника (табл. 5.11).

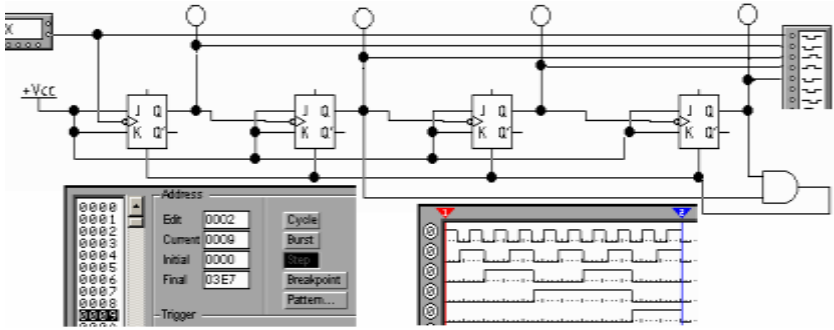


Рис. 5.28. Функціональна схема лічильника із виключенням кінцевих станів за модулем К = 10 на JK-тригерах

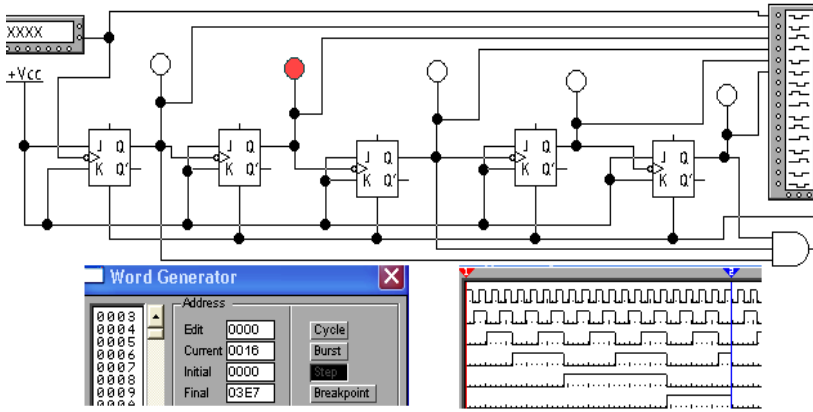


Рис. 5.29. Функціональна схема лічильника із виключенням останніх станів за модулем К = 21 на JK-тригерах

Таблиця 5.9

№	Тригер	Ключ	№	Тригер	Ключ
1	D	4	7	D	8
2	JK	4	8	JK	8
3	D	6	9	D	9
4	JK	6	10	JK	9
5	D	7	11	D	10
6	JK	7	12	JK	10

Таблиця 5.10

Переходи	D
0 → 0	0
0 → 1	1
1 → 0	0
1 → 1	1

Таблиця 5.11

	Стан Q_{n-1}			Стан Q_n			Виходи тригерів		
№	Q1	Q2	Q3	Q1	Q2	Q3	T1	T2	T3
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	0
2	0	1	0	0	1	1	0	1	1
3		0	1	1	1	0	0	1	0
4		1	0	0	0	0	0	0	0

Перенесемо значення до карт Карно та промінімізуємо їх:

Q_1 \ Q_2Q_3	00	01	11	10
0	0	0	1	0
1	0	x	x	x

$D_1 = Q_2Q_3$

Q_1 \ Q_2Q_3	00	01	11	10
0	0	1	0	1
1	0	x	x	x

$D_2 = \overline{Q}_2Q_3 + Q_2\overline{Q}_3 = Q_2 \oplus Q_3$

Q_1 \ Q_2Q_3	00	01	11	10
0	1	0	0	1
1	0	x	x	x

$T_3 = \overline{Q}_1\overline{Q}_3$

Схема дослідження синтезованого лічильника наведена на рис. 5.30.

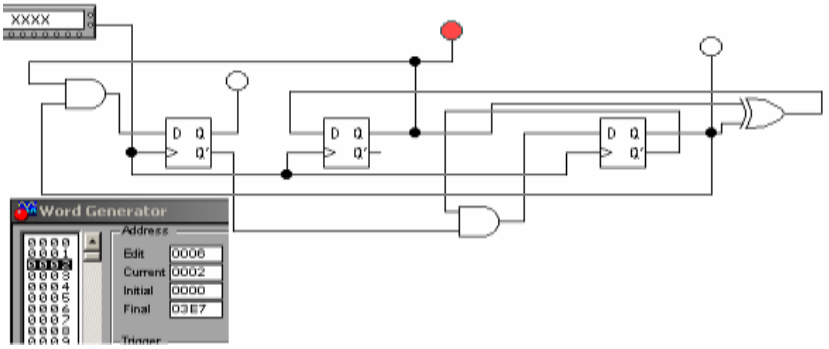


Рис. 5.30. Схема дослідження синтезованого лічильника з $K_{Лч} = 5$ на D-тригерах

Приклад 5.5. Необхідно синтезувати лічильник на ***JK-тригерах*** із коефіцієнтом лічби $K = 5$.

У відповідності з таблицею переходів ***JK-тригера*** (табл. 5.12) побудуємо таблицю функціонування лічильника (табл. 5.13).

Таблиця 5.12

Переходи	J	K
0 → 0	0	x
0 → 1	1	x
1 → 0	x	1
1 → 1	x	0

Таблиця 5.13

	Стан Q_{n-1}			Стан Q_n			Виходи тригерів					
№	Q1	Q2	Q3	Q1	Q2	Q3	J1	K1	J2	K2	J3	K3
0	0	0	0	0	0	1	0	x	0	x	1	x
1	0	0	1	0	1	0	0	x	1	x	x	1
2	0	1	0	0	1	1	0	x	x	0	1	x
3	0	1	1	1	0	0	1	x	x	1	x	1
4	1	0	0	0	0	0	x	1	0	x	0	x

Q_1 \ Q_2Q_3	00	01	11	10
0	0	0	1	0
1	x	x	x	x

$J_1 = Q_2Q_3$

Q_1 \ Q_2Q_3	00	01	11	10
0	x	x	x	x
1	1	x	x	x

$K_1 = 1$

Q_1 \ Q_2Q_3	00	01	11	10
0	0	1	x	x
1	0	x	x	x

$J_2 = Q_3$

Q_1 \ Q_2Q_3	00	01	11	10
0	x	x	1	0
1	x	x	x	x

$K_2 = Q_3$

Q_1 \ Q_2Q_3	00	01	11	10
0	1	x	x	1
1	0	x	x	x

$J_3 = \overline{Q_1}$

Q_1 \ Q_2Q_3	00	01	11	10
0	x	1	1	x
1	x	x	x	x

$K_3 = 1$

Схема дослідження синтезованого лічильника наведена на рис. 5.31.

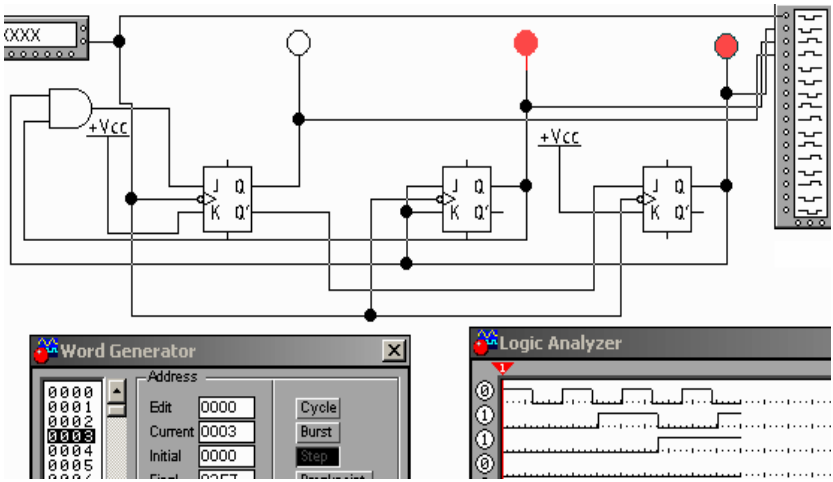


Рис. 5.31. Схема дослідження синтезованого лічильника з $К_{лч} = 5$ на JK-тригерах

3.4. Дослід 4. Дослідження лічильників в інтегральному виконанні.

Зібрати схему дослідження 4-розрядного асинхронного двійкового лічильника МС 7493 (155ИЕ5) як наведено на рис. 5.32.

Змінити схему рис. 5.32 згідно з варіантом (табл. 5.12) та навести часові діаграми.

Логічна структура мікросхеми К155ИЕ5 наведена на рис. 5.33.

3.5. Дослід 5. На основі МС 7493 (155ИЕ5) побудувати лічильник з $К_{лч}$ згідно з варіантом із табл. 5.13.

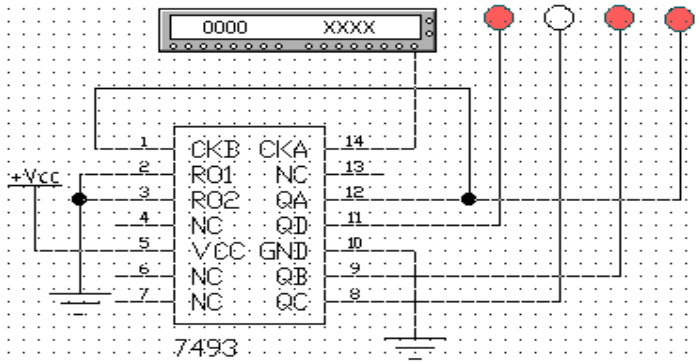


Рис. 5.32. Схема дослідження МС 7493 (155ИЕ5)

Таблиця 5.12

Варіанти	1	2	3	4	5	6	7	8	9	10
$К_{Лч}$	4	5	6	7	8	9	10	11	12	13

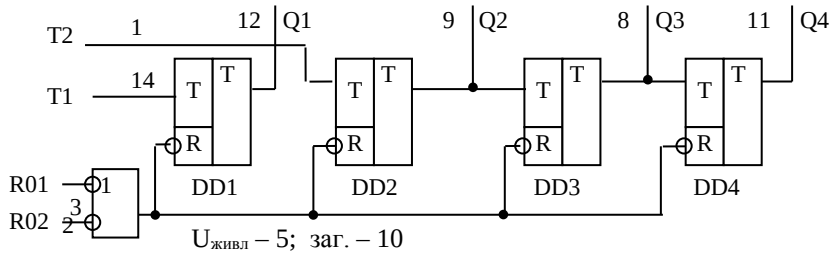


Рис. 5.33. Логічна структура мікросхеми К155ИЕ5

Таблиця 5.13

№	$К_{Лч}$	№	$К_{Лч}$	№	$К_{Лч}$
1	104	7	88	7	128
2	120	8	145	8	225
3	100	9	177	9	153
4	201	10	172	10	156
5	170	11	161	11	204
6	240	12	129	12	193

Як приклад на рис. 5.34 наведена схема лічильника з $К_{Лч} = 146_{10} = 1001\ 0010_2$ з виключенням кінцевих станів на МС 7493. Логічний елемент 3І використовується для скидання тригерів лічильника при досягненні кількості імпульсів, яка дорівнює 146.

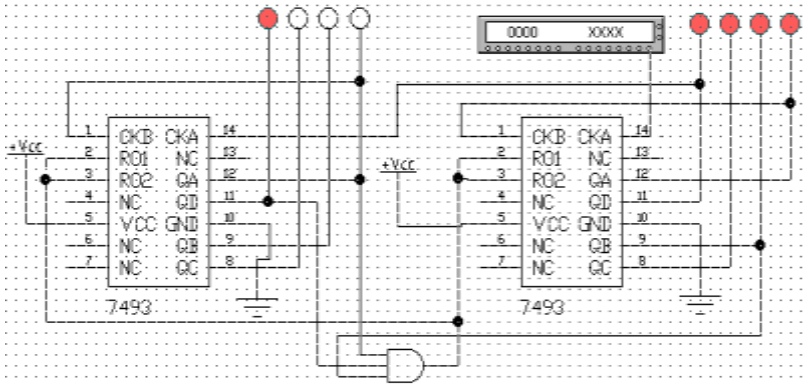


Рис .5.34. Схема лічильника з $К_{лч} = 146_{10}$ на МС 7493

4. *Питання для самоперевірки*

4.1. Яким чином будуються лічильники із коефіцієнтом лічби, не кратним 2?

4.2. Що таке програмований лічильник?

4.3. Побудувати 3-розрядний лічильник на додавання на синхронних Т-тригерах та навести часові діаграми.

4.4. Побудувати 3-розрядний лічильник на віднімання на синхронних Т-тригерах та навести часові діаграми.

4.5. Побудувати 3-розрядний лічильник на додавання на D-тригерах та навести часові діаграми.

4.6. Побудувати 3-розрядний лічильник на віднімання на D-тригерах та навести часові діаграми.

4.7. Побудувати 3-розрядний лічильник на додавання на RS-тригерах та навести часові діаграми.

4.8. Побудувати 3-розрядний лічильник на віднімання на RS-тригерах та навести часові діаграми.

4.9. Побудувати 3-розрядний лічильник на додавання на JK-тригерах та навести часові діаграми.

4.10. Побудувати 3-розрядний лічильник на віднімання на JK-тригерах та навести часові діаграми.

4.11. Побудувати схему лічильника з $К_{лч} = 8$ з використанням D-тригерів з виключенням кінцевих станів.

4.12. Побудувати схему лічильника з $К_{лч} = 8$ з використанням D-тригерів з виключенням початкових станів.

6. СХЕМОТЕХНІКА ОБСЛУГОВУЮЧИХ ЕЛЕМЕНТІВ

6.1. Мультивібратори

Мультивібратор, що очікує, називають також **одновібратором**. **Одновібратори призначені для вироблення одиночних імпульсів із заданою тривалістю.** При цьому тривалість імпульсу, що запускає, особливого значення не має, необхідно лише щоб вона була не більше тривалості імпульсу, що виробляється одновібратором, тобто $t_{\text{і зап}} < t_{\text{і}}$, де $t_{\text{і зап}}$ – тривалість імпульсу, що запускає; $t_{\text{і}}$ – тривалість вихідного імпульсу одновібратора.

Схема одновібратора наведена на рис. 6.1, *а*. Він виконаний на двох елементах логіки типу 2І-НІ шляхом введення позитивного зворотного зв'язку (вихід другого елемента сполучений з входом першого).

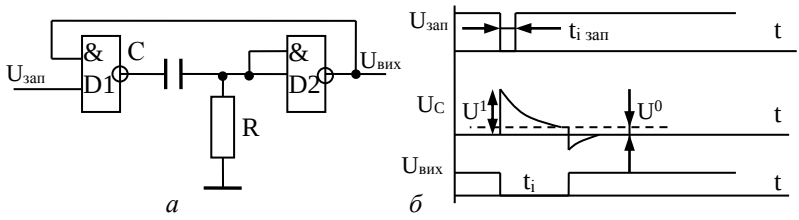


Рис. 6.1. Схема та часові діаграми одновібратора

У початковому стані на виході елемента D2 є рівень "1", а на виході елемента D1 – "0", оскільки на обох його входах є "1" (імпульси, що запускають, мають негативний перепад напруги). При подачі на вхід імпульсу з негативним перепадом напруги на виході першого елемента з'явиться рівень "1", тобто позитивний стрибок, який через конденсатор *С* надійде на вхід другого елемента. Елемент D2 інвертує цей сигнал і рівень "0" по колі зворотного зв'язку подається на другий вхід елемента D1. На виході елемента D2 підтримується рівень "0" до тих

пір, поки не зарядиться конденсатор C до рівня $U_{\text{с пор}} = U^1 - U_{\text{пор}}$, а напруга на резисторі R не досягне порігового рівня $U_{\text{пор}}$ (рис. 6.1, *б*).

Тривалість вихідного імпульсу одновібратора може бути визначена за допомогою виразу

$$t_{\text{і}} = C(R + R_{\text{D1}})\ln(U^1 / U_{\text{пор}}) \approx \tau \ln 2 = \tau\, 0{,}7,$$

де R_{D1} – вихідний опір першого елемента; $U_{\text{пор}}$ – порогова напруга логічного елемента.

Несиметричний мультивібратор. На базі логічних елементів можна побудувати різні генератори імпульсів. Найбільш широкого застосування в цифрових пристроях набули два типи – несиметричний і симетричний мультивібратори. У несиметричному мультивібраторі (рис. 6.2, *а*) резистор R виводить в підсилювальний режим перший інвертор, а вихідна напруга цього інвертора повинна утримувати в режимі підсилення другий інвертор.

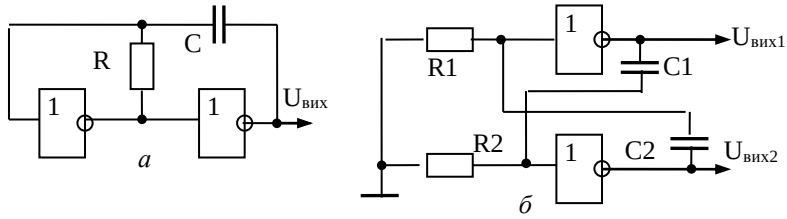


Рис. 6.2. Схеми несиметричного та симетричного мультивібратора

Позитивний зворотний зв'язок через конденсатор C викличе м'яке (що не потребує первинного поштовху) самозбудження автоколивального процесу релаксації. Період T імпульсів, що виробляються мультивібратором, визначається в першому наближенні постійної часу $\tau = RC$ ($T = k\tau$, де k звичайно має значення 1...2). Частоту проходження імпульсів можна оцінити (з точністю до 10 %) з виразу $f = 1/2RC$.

Симетричний мультивібратор. Схема симетричного мультивібратора показана на рис. 6.2, *б*. Симетричність вихідних імпульсів може бути досягнута при виконанні умов: $R1 = R2$; $C1 = C2$. Період проходження імпульсів Т визначається як сума двох часів заряду конденсаторів, тобто

$$T = \tau_{\text{зар1}} + \tau_{\text{зар2}},$$

де $\tau_{\text{зар1}} = \tau 1 \ln(U^1/U_{\text{пор}})$; $\tau_{\text{зар2}} = \tau_2 \ln(U^1/U_{\text{пор}})$.

Значення τ_1 і τ_2 визначаються з урахуванням вихідних опорів інверторів $R_{\text{вих D1}}$, $R_{\text{вих D2}}$:

$$\tau_1 = C1\ (R2 + R_{\text{вих D1}});$$
$$\tau_2 = C2\ (R1 + R_{\text{вих D2}}).$$

Частота проходження вихідних імпульсів симетричного мультивібратора визначається зі співвідношення

$$f_{\text{і}} = 1/T = 1/(\tau_1 + \tau_2) = 1/2RC.$$

6.2. Генератори напруги, що лінійно змінюються (ГЛЗН)

ГЛЗН є електронними пристроями, вихідна напруга яких протягом деякого часу змінюється за лінійним законом. Часто така напруга змінюється періодично. В цьому випадку ГЛЗН називається генератором пилкоподібної напруги (ГПН) або генератором напруги трикутної форми (рис. 6.3, *а*).

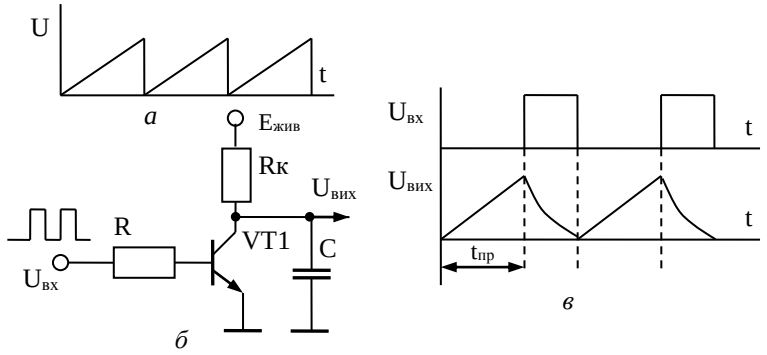


Рис. 6.3. Форми сигналів та схема генератора напруги, що лінійно змінюється

Якщо напруга змінюється від мінімального значення до максимального (за абсолютною величиною), то її називають лінійно наростаючою напругою. Якщо ж вона змінюється від максимального значення до мінімального, то – лінійно падаючою.

ГЛЗН набули широкого застосування у відхиляючих системах осцилографів, телевізорів, у радіолокації, у перетворювачах "напруга – часовий інтервал", у широтно-імпульсних модуляторах і т.д.

ГЛЗН будуються за принципом зарядки і розрядки конденсатора. Схема простішого ГПН, працюючого за принципом заряду конденсатора, наведена на рис. 6.3, *в*. Вона складається з часозадаючого конденсатора C, резистора $R_{\text{к}}$ і транзисторного ключа VT1. На вхід транзисторного ключа подається послідовність прямокутних імпульсів із зада-

ним інтервалом між імпульсами і тривалістю (рис. 6.3, *г*). Коли на базі транзистора – нульова напруга (проміжок часу між імпульсами), транзистор закритий і відбувається заряд конденсатора через резистор $R_к$. Якщо постійна часу кола $R_кC$ достатньо велика, тобто істотно більше за період проходження прямокутних імпульсів, напруга на конденсаторі наростає лінійно. Зарядка конденсатора триває до надходження імпульсу, що відкриває транзистор VT1. Коли транзистор відкривається, починається процес розрядки конденсатора. Інтервал часу між імпульсами, що відкривають, повинен бути достатнім для повної розрядки конденсатора *C*.

Напруга на конденсаторі змінюється згідно із законом

$$U_с = E_{живл}(1 - e^{-t/\tau}),$$

де $\tau = RC$ – постійна часу кола, що складається з $R_к$ і *C*; *t* – поточне значення часу, коли $t = 0$, $U_с = E_{жив}(1 - 1) = 0$.

Відомо, що функцію e^x можна представити у вигляді ряду

$$e^x = 1 + X + X^2/2! + X^3/3! + \ldots + X^n/n!$$

Для значень $X << 1$ функцію можна визначити першими двома членами ряду

$$e^x = 1 + X,$$

тоді, використовуючи цей вираз для випадку заряду конденсатора при $t << \tau$, визначаємо напругу на конденсаторі

$$U_с = E_{живл}(1 - e^{-t/\tau}) = E_{жив}(1 - 1 + t/\tau) = E_{живл}t/\tau,$$

де $t/\tau << 1$.

У разі використання цього процесу в ГПН, $t = t_і = t_{зар}$; $\tau = R_кC$, тоді

$$U_с = E_{живл}\, t/RC.$$

Напруга *U*c(t), що лінійно змінюється, характеризується такими параметрами:

- тривалістю прямого ходу $t_{пр}$, тобто часом, протягом якого конденсатор заряджає через опір $R_к$ до напруги $U_с$;
- тривалістю зворотного ходу $t_{зв}$ (час відновлення) – це час, протягом якого відбувається розрядка конденсатора;
- періодом повторення напруги (пилкоподібних імпульсів), що лінійно змінюється $T = t_{за} + t_{пр}$;
- амплітудою пилкоподібних імпульсів $U_а$;
- коефіцієнтом нелінійності $К_{нл}$.

Одним з найважливіших параметрів ГЛЗН є **коефіцієнт нелінійності**. Для визначення $К_{нл}$ скористаємося відомим твердженням, що лінійна функція характеризується постійністю похідної у всіх її точках,

тому відхилення від лінійного закону можна оцінити коефіцієнтом нелінійності. Нелінійність визначається максимальним відхиленням реальної форми сигналу від ідеальної лінійної форми. Коефіцієнт нелінійності знаходять як відношення змін похідних функції на початку і в кінці процесу наростання. Однак при введенні деяких припущень коефіцієнт нелінійності має вигляд:

$$К_{нл} = t_{пр}/RC.$$

Простий генератор лінійної напруги характеризується також **коефіцієнтом використання напруги джерела живлення:**

$$К_{живл} = U_а/E_{живл}.$$

Якщо підставити значення $U_а$ у вираз для коефіцієнта використання напруги джерела живлення, отримаємо

$$К_{живл} = (E_{живл}\, t_{пр}/\tau)/E_{живл} = t_{пр}/\tau.$$

З одержаного виразу для коефіцієнта нелінійності витікає, що чим краща лінійність пилкоподібної напруги, тим менше амплітуда напруги ГЛЗН. Наприклад, якщо напруга джерела живлення 10 В, для отримання коефіцієнта нелінійності $К_{нл}$ = 1 % амплітуда напруги імпульсів ГПН не повинна перевищувати 0,1.

Для підвищення коефіцієнта використання напруги живлення при малих значеннях коефіцієнта нелінійності застосовуються стабілізатори постійного струму (ГСТ).

Схема простого генератора пилкоподібної напруги із стабілізатором струму в колі розряду конденсатора показана на рис. 6.4, *а*. Заряд конденсатора здійснюється через транзистор VT1 і опір $R_к$. За час заряду напруга на конденсаторі досягає практично напруги джерела живлення. Коли приходить на базу транзисторів нульовий рівень, перший транзистор закривається, а транзистор VT2 переходить в режим генератора стабільного струму (ГСТ) і через нього протікає стабільний постійний струм розряду конденсатора (рис. 6.4, *б*).

При визначенні коефіцієнта нелінійності імпульсів цього генератора пилкоподібної напруги необхідно враховувати вплив опору навантаження $R_н$ на процес розряду конденсатора. Виходить, що для зменшення γ бажано використовувати високоомні навантаження або зменшувати амплітуду імпульсу сигналу.

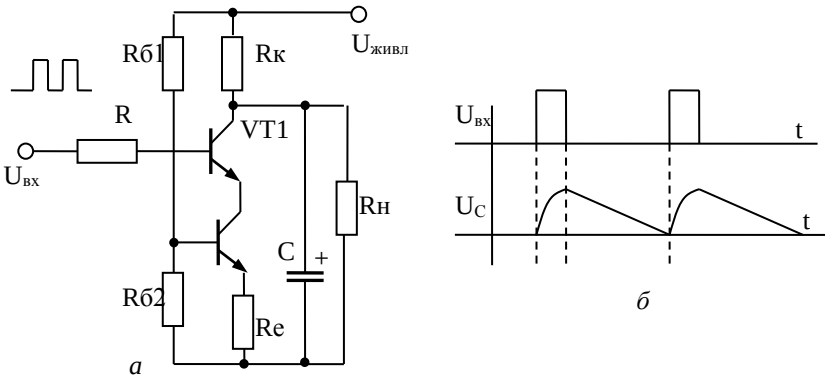


Рис. 6.4. Простий генератор пилкоподібної напруги: *а* – схема генератора пилкоподібної напруги із ГСТ; *б* – його часові діаграми

6.3. Генератори псевдовипадкових чисел

Автономний лінійний автомат, побудований як n-розрядний регістр зсуву з лінійними зворотними зв'язками (рис. 6.5) згідно з характеристичним поліномом $P(x) = x^n \oplus a_{n-1}x^{n-1} \oplus \ldots \oplus a_1 x \oplus a_0$ є генератором псевдовипадкових чисел (ГПВЧ).

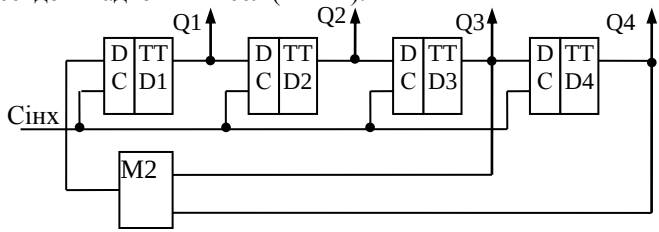


Рис. 6.5. Функціональна схема генератора псевдовипадкової послідовності з поліномом $P(x) = x^4 \oplus x^3 \oplus 1$

Характеристичний поліном P(x) визначає структуру ланцюгів зворотного зв'язку ($a = 1$ за наявності зв'язку від i-го розряду, $a = 0$ при її відсутності). Двійкова послідовність на виході будь-якого розряду ГПВЧ має період максимальної довжини $M = 2^n - 1$. Для того, щоб генератор на основі n-розрядного регістра зсуву з лінійним зворотним зв'язком мав максимальний період, необхідно і достатньо, щоб відповідний характеристичний поліном P(x) був таким, що не приводиться.

Подібні генератори широко застосовуються в системах діагностування цифрової апаратури. На їх основі будуються керовані джерела випробувальних послідовностей, які можуть, зокрема, змінювати вірогідність одиничних сигналів на кожному з входів об'єкта контролю.

Якщо в схемі генератора псевдовипадкових чисел додатково на суматор за модулем 2 подається вхідна послідовність, то така схема називатиметься сигнатурним аналізатором (СА).

Метод синтезу багатоканальних сигнатурних аналізаторів (БСА)

Розглянемо математичну модель функціонування одноканального (ОСА). При цьому припустимо, що аналізується послідовність довжиною n за допомогою регістру зсуву, який має r елементів.

Позначимо попередній стан j – го елементу b_j, а наступний – b^1_j. Тоді попередній стан регістру, який включає 16 елементів, буде $B = ||b_1, b_2, \ldots, b_{16}||$, а наступний – $B^1 = ||b_1^1, b_2^1, \ldots, b_{16}^1||$.

Стан кожного r_i залежить від стану інших r_j відповідно до виразу

$$b_j^1 = a_1b_1 \oplus a_2b_2 \oplus a_3b_3 \oplus \ldots \oplus a_jb_j \oplus \ldots \oplus a_{16}b_{16},$$ де $a_j = \{0/1\}$.

Наприклад, для лінійного регістра, який побудовано з використанням полінома degP(x) = 16, кожний новий стан тригерів визначається за такими формулами:

$$b_1^1 = b_7 \oplus b_9 \oplus b_{12} \oplus b_{16}; \quad b_2^1 = b_1; \quad b_3^1 = b_2; \ldots ; b_{16}^1 = b_{15}.$$

Такий взаємозв'язок тригерів можна описати матрицею S, яку назвемо матрицею зв'язку:

$$S = (a_{ij})^{mn}_{i,j=1} = \begin{vmatrix} a_{11}\ a_{12} \ldots a_{1n} \\ a_{21}\ a_{22} \ldots a_{2n} \\ a_{m1}\ a_{m2} \ldots a_{mn} \end{vmatrix}, \qquad m, n = \overline{1, 16} \qquad (6.1)$$

Вектор наступного стану регістра можна визначити з виразу

$$B_к^T = S \times B_{k-1}^T,$$

де k – поточний номер такту;

T – символ транспонування.

Тоді $B_2^T = S \times B_1^T$, $B_3^T = S^2 \times B_2^T$.

Із (2.1) через k тактів отримаємо:

$$B_k^T = S^{k-1} B_1^T. \qquad (6.2)$$

Оскільки запис проводиться в перший тригер регістра, то отримаємо $B_1^T = B^T V_1$, де $V_1 = \{1/0\}$ – перший розряд кодової комбінації.

Відомо, що сигнатурний аналізатор являє собою регістри зсуву зі зворотними зв'язками. Відповідно до схеми аналізатора наступний стан кожного *r* залежить від поточного стану інших тригерів і визначається за формулою

$$b_j^1 = \sum_{j=1}^{r} a_j b_j . \qquad (6.3)$$

Вираз (6.2) можна записати у вигляді

$$B_k^T = H \times V(t), \qquad (6.4)$$

де V(t) – вектор вхідної послідовності.

Вирази (6.3) та (6.4) описують стан регістра після надходження на його вхід *n*-розрядної кодової комбінації.

З виразу (6.4) слідує висновок, що кожний *k*-й розряд вхідної послідовності повинен подаватися на *k*-й розряд перевірочної матриці H. Ця умова однозначно відповідає послідовним і паралельним СА (рис. 6.6).

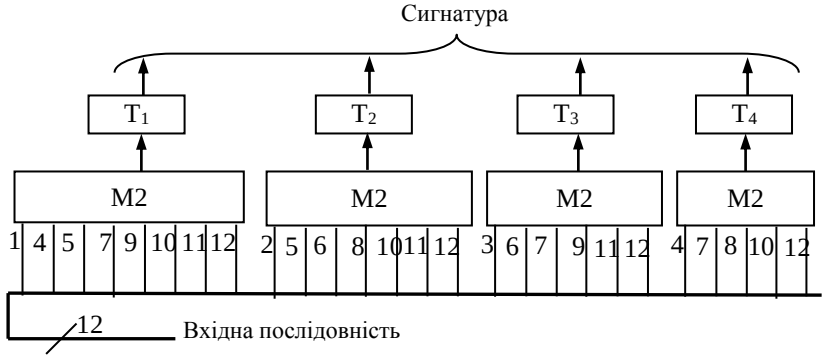


Рис. 6.6. Схема паралельного сигнатурного аналізатора з $n = 12$

При синтезі послідовно-паралельного СА поза залежності від математичних перетворень ця умова повинна виконуватися також. Виразу (6.4) також можливо надати вигляду

$$Sig^T V(t) = H \times V(t) = \| h_1, h_2, h_3 ..., h_n \| \times \| v_1, v_2, v_3 ..., v_n \| .$$

Розглянемо побудову **нелінійного** генератора псевдовипадкової послідовності. Такий генератор будується на основі полінома, коефіцієнти при аргументах якого дорівнюють не тільки {0, 1}. Для кінцевого

поля GF(3) функціональна схема нелінійного сигнатурного аналізатора з поліномом $P(x) = 2x^4 \oplus_3 x \oplus_3 1$ наведена на рис. 6.7.

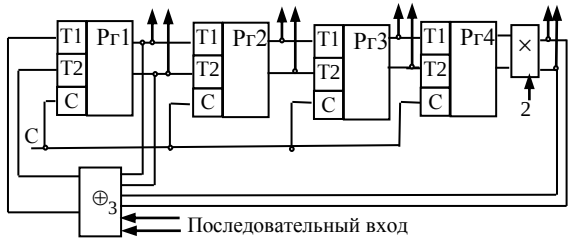


Рис. 6.7. Функциональная схема НСА с $P(x) = 2x^4 \oplus_3 x \oplus_3 1$

Коефіцієнт 2 при старшому ступені полінома вказує на те, що інформація, яка записувана в 4-й регістр зсуву повинна бути помножена за правилами арифметики кінцевого поля GF(3):

$\oplus_3$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\otimes_3$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Схему суматора за mod3 можна побудувати на основі таблиці істинності (табл. 1) за допомогою, наприклад програми EWB.

Таблиця 1

B1	B0	A1	A0	S1	S0	B1	B0	A1	A0	S1	S0
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	1	1	0	0	1	0	0
0	0	1	0	1	0	1	0	1	0	0	1
0	0	1	1	0	0	1	0	1	1	1	0
0	1	0	0	0	1	1	1	0	0	0	0
0	1	0	1	1	0	1	1	0	1	0	1
0	1	1	0	0	0	1	1	1	0	1	0
0	1	1	1	0	1	1	1	1	1	0	0

В цій програмі в логічному конверторі потрібно проставити значення однієї з вихідних функцій та отримати за таблицею істинності логічний вираз та схему. Отримані схеми для молодшого та старшого

розряду об’єднуються в одну схему (макрос) (рис. 6.8), використавши, наприклад, клавіші Ctrl+B.

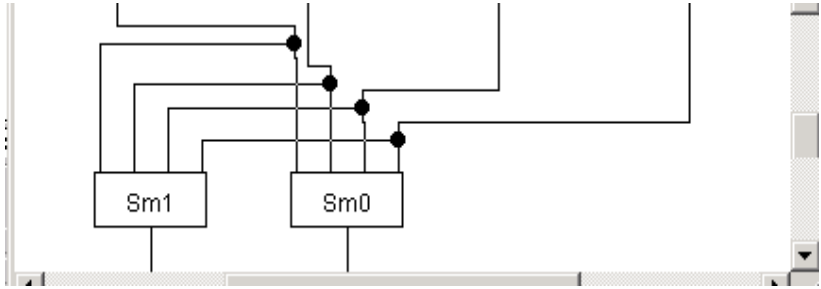


Рис. 6.8. Внутрішня частина схеми суматора за mod3

Схема формування молодшого розряду Sm0 суматора за mod3 наведена на рис. 6.9.

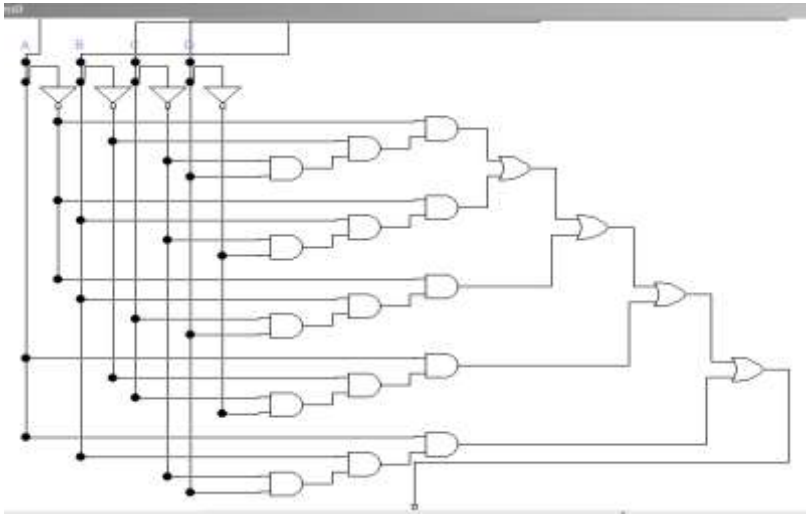


Рис. 6.9. Схема формування молодшого розряду Sm0 суматора за mod3

Схема генератора псевдовипадкової послідовності на основі нелінійного регістру зсуву з $P(x)= x^4 \oplus_3 x^3 \oplus_3 1$ наведена на рис. 6.10.

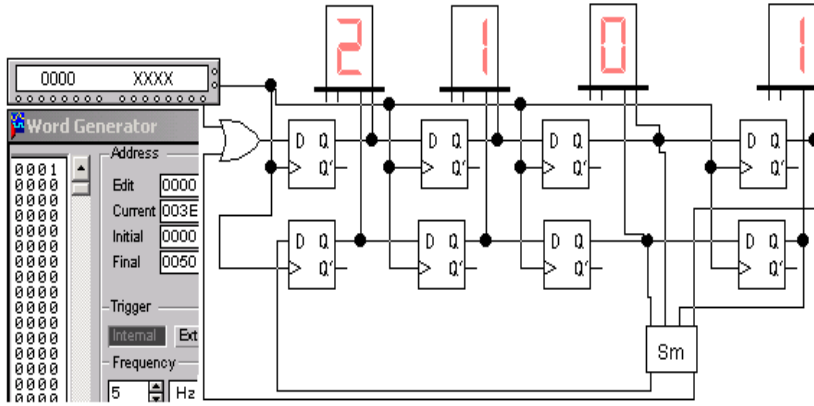


Рис. 6.10. Схема генератора псевдовипадкової послідовності на основі нелінійного регістру зсуву

Число станів l визначається як сума переборів всіх наборів із трьох (0, 1, 2):

$$l = 3^n - 1,$$

де "– 1" – стан всіх нулів в регістрі зсуву;

n – розрядність регістра зрушення із зворотними зв'язками.

Таким чином, число станів l при degP(x)= 4 в полі GF(3) дорівнює:

$$I = 3^n - 1 = 80.$$

Отже, до поліномів максимальної (М) довжини при degP(x)= 4 в кінцевому полі GF(3) необхідно відносити поліноми, які генерують число своїх станів $l = 80$.

Повний список поліномів з $degP(x) = 4$ та $l_м = 80$ має вигляд:

$P(x) = x^4 \oplus_3 x \oplus_3 1;$

$P(x) = x^4 \oplus_3 x^3 \oplus_3 1;$

$P(x) = 2x^4 \oplus_3 2x^3 \oplus_3 1;$

$P(x) = x^4 \oplus_3 x^3 \oplus_3 2x^2 \oplus_3 2x \oplus_3 1;$

$P(x) = x^4 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 2x \oplus_3 1;$

$P(x) = 2x^4 \oplus_3 2x \oplus_3 1;$

$P(x) = 2x^4 \oplus_3 x^3 \oplus_3 2x^2 \oplus_3 x \oplus_3 1;$

$P(x) = 2x^4 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 x \oplus_3 1.$

У зв'язку з тим, що значення вільного члена x^0 поліномів може бути $x^0 \in \{1, 2\}$, то кількість таких поліномів подвоюється. Дослідження показали, що на довжину циклу полінома і його вид (послідовність станів) значення $x^0 \in \{1, 2\}$ не впливають.

6.4. Формувачі імпульсів

Формувачі імпульсів призначені для створення імпульсів у потрібний момент часу і заданої наперед тривалості. Дуже часто виникає завдання формування короткого імпульсу по фронту або зрізу іншого сигналу будь-якої тривалості.

Формувач коротких імпульсів із застосуванням ліній затримки. Формувач коротких імпульсів формує імпульси, тривалість яких істотно менше за тривалість початкових імпульсів.

Для побудови схеми формувача буде потрібний один елемент кон'юнкції, один інвертор і лінія затримки. Тривалість вихідного імпульсу формувача визначається тривалістю часу затримки лінії затримки $\Delta t_з$ і середнім часом розповсюдження сигналу через інвертор $t_{з\,ср\,Е1}$.

На рис. 6.11 наведена схема формувача, а на рис. 6.12, *а* і *б* – часові діаграми, що ілюструють їх роботу. З рис 6.12, *а* витікає, що для формування імпульсу від переднього фронту (початкового імпульсу) необхідно подавати на лінію затримки інвертований імпульс.

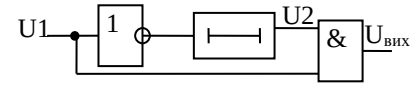


Рис. 6.11. Схема формувача коротких імпульсів з використанням лінії затримки

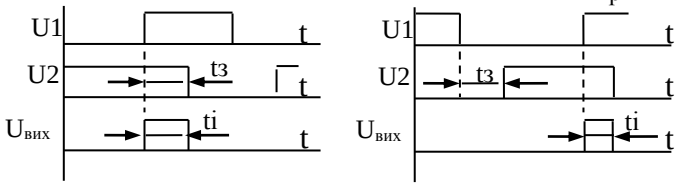


Рис. 6.12. Часові діаграми функціонування формувачів коротких імпульсів з використанням лінії затримки

У разі формування імпульсу від заднього фронту слід інвертувати незатриманий (прямою) сигнал, тобто сигнал, що подається на елемент "І", обминаючи лінію затримки (рис. 6.12, *б*).

Використання у формувачах ліній затримки не завжди виправдане економічно і з конструктивних міркувань. Якщо не потрібне формування строго певної тривалості коротких імпульсів, у формувачах замість **лінії затримки застосовуються логічні елементи** (рис. 6.13).

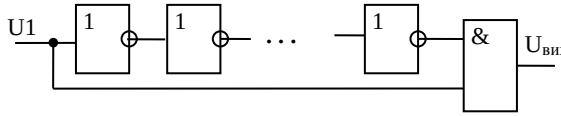


Рис. 6.13. Схема формувача коротких імпульсів
із використанням елементів логіки

Оскільки кожен логічний елемент має властивість затримувати розповсюдження сигналу, то час затримки в такій схемі визначатиметься числом використовуваних елементів логіки *n*.

Вважається, що інвертор має значно менший час затримки сигналу, і тому як елементи затримки використовуються логічні елементи з малою швидкодією.

Формувач імпульсів на елементах логіки з використанням RC-кола. RC-кола широко застосовуються в імпульсній техніці для формування сигналів різної форми. Залежно від поєднання з'єднань RC-коло може виконувати функцію кіл, що як укорочують, так і подовжують. Формувач імпульсу з RC-кіл, що подовжує, і його часові діаграми наведені на рис. 6.14, *а* і *б*, відповідно.

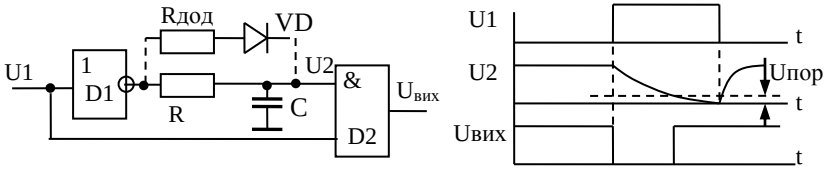


Рис. 6.14. Схема формувача імпульсу з RC-колом, що подовжує,
і його часові діаграми

Тривалість виробленого формувачем імпульсу можна обчислити виходячи з умови розряду конденсатора *C*. Дійсно, поки конденсатор *C* розряджається до рівня порогової напруги $U_{пор}$, напруга U2

сприймається елементом D2 як рівень логічної “1” і на його виході підтримується “0”. З часом *t*і напруга на конденсаторі *C* стає рівною $U_{пор}$ і на виході елементу D2 з’являється “1”. Якщо вважати, що напруга до початку розряду на конденсаторі дорівнювала напрузі рівня “1”, тобто U1, та зміна напруги Uc з часом можна представити як

$$U_C = U^1 e^{-t/\tau}.$$

Звідси маємо

$$e^{-t/\tau} = U^1/U_C.$$

Тривалість імпульсу дорівнює часу розряджання конденсатора до порогового значення $U_{пор}$:

$$t_i = RC \ln(U^1/U_C).$$

Для прискореного відновлення зарядки конденсатора в схему може бути включений додатковий діод VD. Із-за великого зворотного опору діода його вплив в процес розрядки конденсатора можна не враховувати, тобто розрядка конденсатора здійснюватиметься тільки через опір *R*.

У тих випадках, коли потрібно одержати імпульси великої тривалості і в схемі використовується конденсатор великої ємності, послідовно з діодом включають додатковий резистор $R_{дод}$, що обмежує струм зарядки конденсатора. Величину опору *R* вибирають виходячи з таких умов:

– по-перше, величина опору R не повинна перевищувати максимально допустимого значення $R_{мак}$ = 2,2 кОм, при якому на цьому опорі за рахунок зворотного вхідного струму елемента логіки може створитися напруга, порівнянна з напругою $U_{пор}$ (для елементів ТТЛ структури вона має максимальне значення);

– по-друге, мінімальне значення опору обмежене допустимою здатністю навантаження логічного елемента D1 і визначається як

$$R_{min} = U1/I^{o}_{дод} = U^1/nI_{вх\ мах},$$

де U^1 – напруга на виході елемента D1 у стані логічною “1”; *n* – коефіцієнт розгалуження (здатність, навантаження) виходу логічного елементу; $I_{вх}$ – вхідний струм одного елемента.

Схема формувача коротких імпульсів за допомогою RC-кола, що укорочує (диференційний)**,** показана на рис. 6.15.

Тривалість вихідного імпульсу формувача може бути визначена зі співвідношення

$$t_i/(C(R + R_{вих})) = \ln(Y^1/U_{пор}),$$

де $R_{вих}$ – вихідний опір першого елемента формувача.

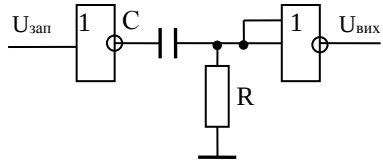


Рис. 6.15. Схема формувача імпульсів
із диференційним колом

Тригер Шмітта. Тригер Шмітта застосовується для формування вхідного сигналу довільної форми в сигнали, що приймають стандартні рівні ”0” і “1”. Варіанти схем таких формувачів показані на рис. 6.16.

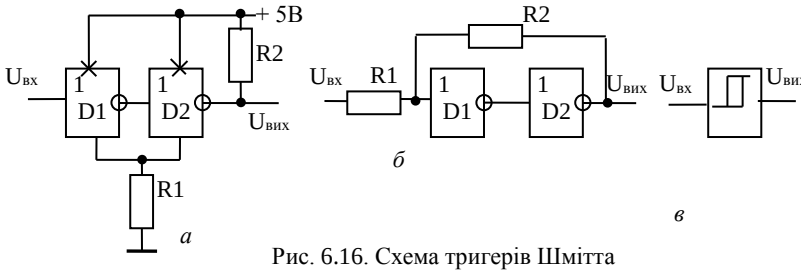


Рис. 6.16. Схема тригерів Шмітта

На рис. 6.16, *а* показана схема тригера Шмітта, в якій застосовані два інвертори, що входять до серії логічних транзисторно-транзисторних інтегральних схем. Позитивний зворотний зв’язок між інверторами забезпечується за рахунок резистора *R*1, включеного в загальне коло живлення елементів. Для збільшення впливу кола зворотного зв’язку струм через другий інвертор збільшується шляхом включення додаткового резистора *R*2 між виходом D2 і джерелом живлення. Подібний формувач на інтегральних схемах серії К1533 задовільно працює до частоти декілька мегагерц при подачі на вхід синусоїдної напруги амплітудою 0,5 – 0,8 В.

У тригерах Шмітта позитивний зворотний зв’язок можна ввести також шляхом включення резистора між виходом другого інвертора і входом першого (рис. 6.16, *б*). Вхідна напруга в цьому формувачі подається через додатковий резистор R1, опір якого також впливає на глибину позитивного зворотного зв’язку. Збільшення опору цього резис-

тора збільшує коефіцієнт позитивного зворотного зв'язку і зменшує чутливість формувача до вхідної напруги.

На практиці як формувачі імпульсів часто застосовують спеціальні інтегральні схеми формувачів (рис. 6.16, *в*). Позначення функціонального призначення таких інтегральних схем містить дві букви "ТЛ".

Формувач імпульсів від механічних контактів. При проектуванні цифрових пристроїв часто виникає необхідність чіткого формування імпульсів від механічних контактів (при спрацьовуванні реле, кнопок, перемикачів і т.д.), оскільки безпосередня подача цих сигналів на входи цифрових пристроїв недопустима через "брязкіт" контактів. "Брязкіт" контактів – це явище багатократного неконтрольованого замикання і розмикання контактів у моменти їх зіткнення і розбіжності. Це явище приводить до формування пачки імпульсів (замість необхідного одиночного імпульсу або перепаду напруги), що можуть викликати багатократне непередбачуване спрацювання тригерів і лічильників схеми цифрового пристрою.

Існує безліч варіантів побудови кіл придушення імпульсів брязкоту контактів за допомогою статичного тригера, диференціюючих і інтегруючих кіл, а також вузла, що має властивості інтегруючого кола і тригера Шмітта. На рис. 6.17 наведені приклади схем придушення "брязкоту" контактів.

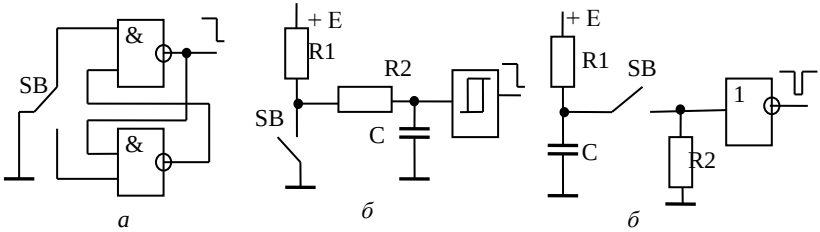


Рис. 6.17. Схеми формування імпульсів від механічних контактів

Найбільш надійною і простою в схемному рішенні є схема придушення "брязкоту" на статичному *RC*-тригері (рис. 6.17, *а*). Сигнал "0", що подається за допомогою перемикача до одного з входів цього тригера, перекидає його. Причому при кожному спрацьовуванні перемикача (кнопки) тригер реагує на перше ж замикання відповідної контактної пари, і подальші замикання вже не змінюють його стан. Недоліком такої схеми придушення брязкоту є необхідність використання контак-

тів на перемикання, що не завжди прийнятно. У тих випадках, коли кнопка (перемикач) має всього одну пару контактів тільки на замикання, застосовуються схеми, що використовують постійну часу перезарядки конденсатора.

Формувач, показаний на рис. 6.17, *б* позбавлений цього недоліку. Він складається з тригера Шмітта, на вході якого включене інтегруюче коло (R_2C). При замиканні контактів кнопки SB напруга на вході кола R_2C падає до нульового значення. Імпульси, викликані "брязкотом", згладжуються інтегруючим колом. Постійна часу інтегруючого кола вибирається так, щоб амплітуда пульсацій сигналу на її виході була менше порогу чутливості тригера Шмітта.

Даний формувач може працювати і без опору R2 (його включають як струмообмежувач через замкнуті контакти кнопки). Завдяки малому опору замкнутих механічних контактів перше ж їх замикання приводить до повної розрядки конденсатора. Подальші ж розмикання контактів, викликані брязкотом, практично не збільшують напруги на конденсаторі унаслідок відносно великої постійної часу його зарядки.

Формувач імпульсів на одному інверторі (рис. 6.17, *в*) дозволяє одержати відносно велику постійну часу перезарядки конденсатора при малій його місткості. При замиканні контактів кнопки конденсатор *C* швидко розряджається через R2. На відміну від розглянутих вище формувачів, тут на виході виробляється імпульс, тривалість якого визначається постійному часу *RC*-кола.

Для формування імпульсів від механічних контактів можна використовувати також одновібратор.

6.5. Завдання до самостійних досліджень генераторів імпульсів

1. *Мета*

Ознайомлення з роботою генераторів імпульсів за допомогою інструментальних засобів цифрової частини пакета EWB, закріплення теоретичного матеріалу, набуття навиків створення і моделювання цифрових пристроїв.

2. *Теми для попереднього опрацювання*

2.1. Одновібратори;

2.2. Мультивібратори.

3. *Завдання*

3.1. Дослід 1. Побудувати генератор одиночних імпульсів (мультивібратор, що очікує) згідно із функціональною схемою, наведеною на рис. 6.18, та варіантом вибору резистора із табл. 6.1.

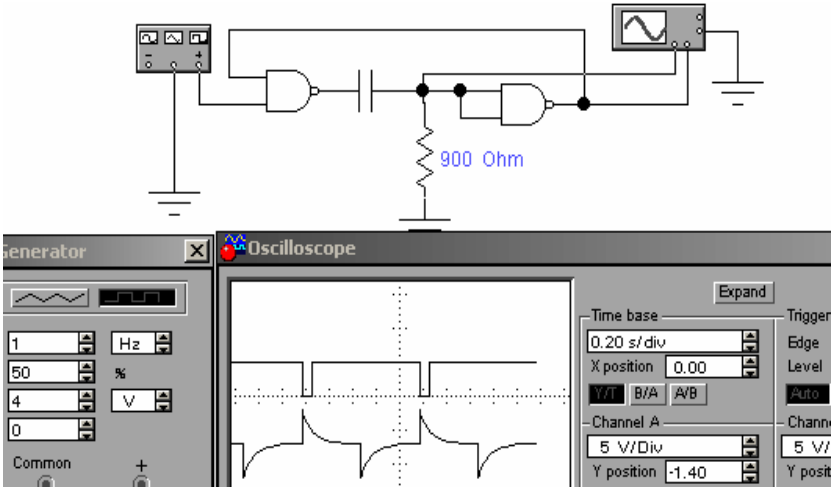


Рис. 6.18. Функціональна схема мультивібратора, який очікує

При проведенні досліджень сформувати $t_{\text{і вих.}} = 1/2 \times t_{\text{ген}}$

Навести функціональну схему та копії екрану, які пояснюють її функціонування, заміряти $t_{\text{і вих.}}$.

Таблиця 6.1

№, з/п	R, Ом	№, з/п	R, Ом	№, з/п	R, Ом
1	400	11	3200	21	300
2	4200	12	3400	22	600
3	800	13	3800	23	700
4	1500	14	4000	24	1000
5	1200	15	1500	25	200
6	3000	16	1300	26	4500
7	1800	17	5500	27	300
8	2000	18	6000	28	2500
9	5000		6500	29	500
10	1400	20	7000	30	100

Тривалість вихідного імпульсу одновібратора може бути визначена за допомогою виразу

$$t_i = C(R + R_{вих})\ln(U^1 / U_{пор}) \approx \tau \ln 2 = \tau\ 0{,}7,$$

де $R_{вих}$ – вихідний опір першого елемента; $U_{пор}$ – порогова напруга логічного елемента.

3.2. Дослід 2. Побудувати та дослідити схему генератора прямокутних імпульсів згідно з варіантом із табл. 6.1 та функціональною схемою, наведеною на рис. 6.19.

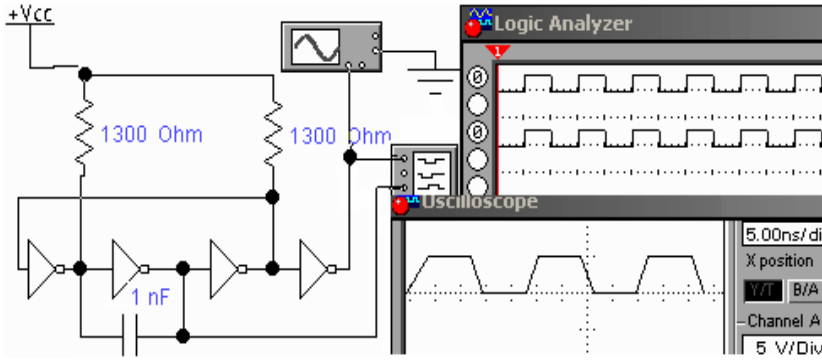


Рис. 6.19. Схема генератора прямокутних імпульсів

3.3. Дослід 3. На основі схемі, наведеної на рис. 6.19, побудувати та дослідити схему, яку зібрати з мікросхем згідно з варіантом (табл. 6.2).

Для дослідження імпульсів під'єднати осцилограф.

3.4. Дослід 4. Побудувати та дослідити схему генератора псевдовипадкової послідовності згідно з варіантом, наведеним у табл. 6.3.

Як приклад на рис. 6.20 наведена функціональна схема генератора псевдовипадкової послідовності із $P(X) = x^4 \oplus x^3 \oplus 1$.

У звіті навести кількість станів тригерів до моменту їх повторення (один цикл генерації випадкових чисел).

Таблиця 6.2

№ з/п	Серія SN74	Вітчизняні МС	Функціональне призначення
1	7400	155ЛА3	4 елементи 2І-НІ
2	7402	155ЛЕ1	4 елементи 2АБО-НІ
3	7403	155ЛА9	4 елементи 2І-НІ з відкритим колектором
4	7404	155ЛН1	6 елементів НІ
5	7405	555ЛН2	6 елементів НІ з відкритим колектором
6	7406	155ЛН3	6 елементів НІ з відкритим колектором
7	7410	155ЛА4	3 елементи 3І-НІ
8	7412	155ЛА10	3 елементи 3І-НІ з відкритим колектором
9	7416	155ЛН5	6 буферних елементів НІ
10	7420	155ЛА1	2 елементи 4І-НІ
11	7422	155ЛА7	2 елементи 4І-НІ з відкритим колектором
12	7426	155ЛА11	4 елементи 2І-НІ з відкритим колектором
13	7428	155ЛЕ5	4 елементи 2АБО-НІ
14	7430	155ЛА2	Елемент 8І-НІ
15	7437	155ЛА12	4 елементи 2І-НІ з відкритим колектором
16	7438	155ЛА13	4 елементи 2І-НІ з відкритим колектором
17	7440	155ЛА6	2 елементи 4І-НІ з підвищеною загрузочною спром.

Таблиця 6.3

№, з/п	P(x)	№, з/п	P(x)	№, з/п	P(x)
1	51	6	65	11	45
2	61	7	57	12	65
3	67	8	47	13	23
4	47	9	73	14	21
5	75	10	63	15	31

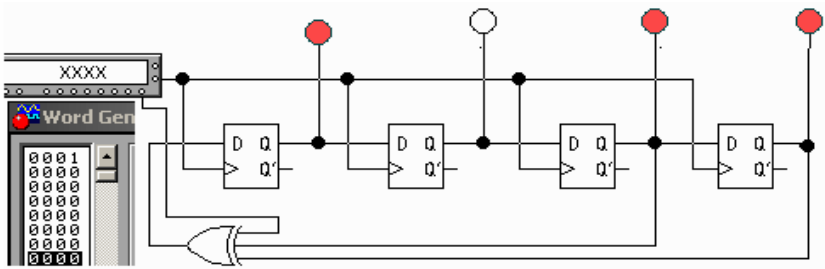


Рис. 6.20. Функціональна схема генератора псевдовипадкової послідовності із $P(X) = x^4 \oplus x^3 \oplus 1$

4. *Питання для самоперевірки*

4.1. Що таке одновібратор та де він може використовуватись?

4.2. Від яких параметрів залежить тривалість вихідного імпульсу з генератора імпульсів?

4.3. Що називається генератором псевдовипадкових чисел та де він може використовуватись?

4.4. Який генератор псевдовипадкових чисел формує послідовність максимальної довжини?

4.5. Наведіть приклади використання нелінійних генераторів псевдо випадкових чисел.

4.6. Побудуйте генератор псевдовипадкової послідовності на основі нелінійного регістру зсуву та одного з поліномів з $degP(x) = 5$ та $l = 242$:

1. $P(x) = x^5 \oplus_3 x^3 \oplus_3 2x \oplus_3 1$;
2. $P(x) = x^5 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 1$;
3. $P(x) = x^5 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 x \oplus_3 1$;
4. $P(x) = x^5 \oplus_3 x^4 \oplus_3 x^2 \oplus_3 2x \oplus_3 1$;
5. $P(x) = x^5 \oplus_3 x^4 \oplus_3 2x^2 \oplus_3 1$;
6. $P(x) = x^5 \oplus_3 x^4 \oplus_3 x^3 \oplus_3 2x \oplus_3 1$;
7. $P(x) = x^5 \oplus_3 x^4 \oplus_3 x^3 \oplus_3 2x^2 \oplus_3 x \oplus_3 1$;
8. $P(x) = x^5 \oplus_3 x^4 \oplus_3 2x^3 \oplus_3 2x^2 \oplus_3 2x \oplus_3 1$;
9. $P(x) = x^5 \oplus_3 2x^4 \oplus_3 x \oplus_3 1$;
10. $P(x) = x^5 \oplus_3 2x^4 \oplus_3 x^3 \oplus_3 1$;
11. $P(x) = x^5 \oplus_3 2x^4 \oplus_3 x^3 \oplus_3 x^2 \oplus_3 1$;
12. $P(x) = x^5 \oplus_3 2x^4 \oplus_3 2x^3 \oplus_3 x^2 \oplus_3 2x \oplus_3 1$;
13. $P(x) = 2x^5 \oplus_3 x \oplus_3 1$;
14. $P(x) = 2x^5 \oplus_3 2x^2 \oplus_3 2x \oplus_3 1$;
15. $P(x) = 2x^5 \oplus_3 x^3 \oplus_3 2x^2 \oplus_3 2x \oplus_3 1$;
16. $P(x) = 2x^5 \oplus_3 2x^3 \oplus_3 2x^2 \oplus_3 1$;
17. $P(x) = 2x^5 \oplus_3 x^4 \oplus_3 1$;
18. $P(x) = 2x^5 \oplus_3 x^4 \oplus_3 x^2 \oplus_3 x \oplus_3 1$;
19. $P(x) = 2x^5 \oplus_3 x^4 \oplus_3 x^3 \oplus_3 x \oplus_3 1$;
20. $P(x) = 2x^5 \oplus_3 x^4 \oplus_3 2x^3 \oplus_3 2x^2 \oplus_3 1$;
21. $P(x) = 2x^5 \oplus_3 2x^4 \oplus_3 2x \oplus_3 1$;
22. $P(x) = 2x^5 \oplus_3 2x^4 \oplus_3 x^3 \oplus_3 x^2 \oplus_3 2x \oplus_3 1$.

7. СХЕМОТЕХНІКА АНАЛОГОВИХ ТА КОМБІНАТОРНИХ ВУЗЛІВ

7.1. Аналогові інтегровані схеми

Операційним підсилювачем називається диференціальний підсилювач постійного струму з великим коефіцієнтом посилення. Операційний підсилювач використовується для посилення різниці вхідних сигналів і має два входи: вхід, що інвертує і вхід, що не інвертує полярність сигналу і, як правило, один вихід.

Операційний підсилювач – це високоякісний пристрій, призначений для посилення як постійних, так і змінних сигналів. Такі підсилювачі використовували головним чином в аналогових обчислювальних пристроях для виконання математичних операцій (складання, віднімання і т. д.). Це пояснює походження терміну "операційний".

Для побудови стабільних підсилювачів необхідно зробити підсилювач з великим коефіцієнтом посилення (≈ 1 000 000), а потім застосувати негативний зворотний зв'язок (НЗВ). При цьому не важливо, що велике значення коефіцієнта посилення виходить з нерівномірною частотною і фазовою характеристиками і т.д. Величина НЗВ задається пасивними елементами, наприклад резисторами, а вони мають непогану стабільність.

Розглянемо схему підсилювача із загальним емітером (ЗЕ), виконаним на *n-p-n* транзисторі (рис. 7.1).

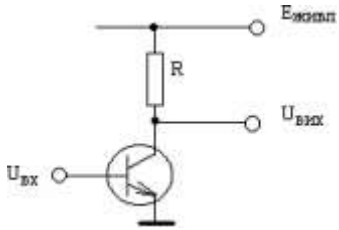


Рис. 7.1. Схема підсилювача
із загальним емітером

Вираз для $U_{вих}$ має вигляд:

$$U_{вих} = E_{живл} - RI_к \approx E_{живл} - RI_е = E_{жив} - RI_0 \exp(gU/kT),$$

де T – абсолютна температура;
g – заряд одного електрона;
k – постійна Больцмана.

Цей вираз залежить переважно від температури. Але при цьому вважають, що при зміні температури одночасно змінюється і вхідний сигнал. Отже $U_{вих}$, $E_{жив}$, R і I_0 залишаються постійними. Тому можна вважати, що змінюються тільки $U_{бе}$ та T, тобто напруга база-емітер і абсолютна температура. Продиференціюємо по T і прирівняємо до нуля попереднє рівняння:

$$[g(dU_{бе}T/dT - U_{бе}]/kT^2 = 0.$$

Скорочуючи цей вираз, отримаэмо

$$dU_{бе}T/dT = U_{бе}/T.$$

У цьому виразі температура вимірюється в градусах Кельвіна, і є близькою до кімнатної. Це приблизно 300 К. А напруга є величиною, яка приблизно дорівнює різниці потенціалів на контактах, оскільки *p-n*-перехід емітер-база зміщений в прямому напрямі. Отже, все залежить від матеріалу (для кремнію це 0,6 В, а для германію 0,3 В).

У кремнієвих транзисторах температурний дрейф на вході складає всього 2 мВ/К. Щоб дізнатися, що буде на виході, треба цю величину помножити на перепад температури і коефіцієнт посилення. У працюючого транзистора перепад температури цілком може бути 10 К, а коефіцієнт посилення у дво- трикаскадного підсилювача може бути 1000...100 000. Виходить 20...2000 В.

Можна використовувати польові транзистори, бо у них температурний дрейф значно менше. Є декілька способів боротьби з температурним дрейфом. Наприклад, існує засіб розділення сигналу на постійну і змінну складові за допомогою розділових конденсаторів. Крім того, можна перетворити сигнал у високочастотний, а після підсилення випрямити (модуляція – посилення – демодуляція).

Але найбільшого поширення набув метод диференціального каскаду.

На рис. 7.2 наведена схема, що складається з двох однакових транзисторів, двох колекторних резисторів і одного емітерного резистора, загального для обох транзисторів. В такій схемі використовуються два джерела живлення.

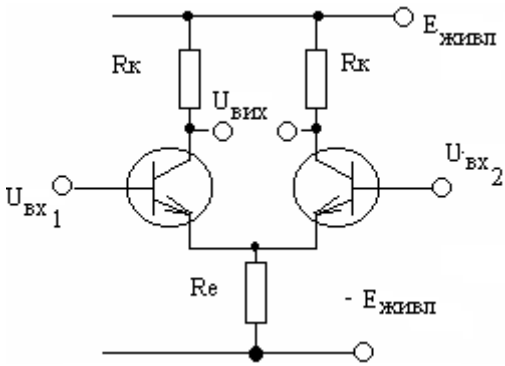


Рис. 7.2. Схема вхідного диференціюючого каскаду операційного підсилювача

Якщо $U_{вх}$ близьке до нуля, то на емітерному опорі маємо постійну напругу, тому емітерний струм, що протікає через цей опір, є теж майже постійним.

Теоретично якщо на входах присутній синфазний сигнал, то струм, що протікає через транзистори, буде теж однаковим (розподілиться навпіл). Але цей струм заданий резистором і майже не залежить від вхідного сигналу. Тому відгук на синфазний сигнал буду дуже малим, а оскільки на виході використовується різницевий сигнал, то він взагалі наближатиметься до нуля. Це обумовлено тим, що в емітері напруга змінюватиметься майже так, як і в базах: різниця потенціалів між базою і емітером змінюється значно менше, ніж на входах.

Диференціальний сигнал також однаковий на обох входах, але протилежний за фазою. Тому на емітерах напруга майже не міняється, повний емітерний струм теж, а на базах транзисторів напруга міняється набагато сильніше, і це призводить до того, що струми через транзистори міняються в різні боки: на одному транзисторі збільшується, а на другому зменшується, хоча в сумі величина струму залишається незмінною.

Тому сигнал на виході (на колекторах) буде в два рази більшим, оскільки він є різницею між двома колекторами.

Диференціальний сигнал посилюється, як у схемі з загальним емітером (ЗЕ), а синфазний сигнал ослабляється, як у схемі з загальним колектором (ЗК), по-перше, і за рахунок віднімання колекторних сигналів, по-друге.

Якщо сигнали $U_{вх1}$ і $U_{вх2}$ є довільними, то можна обчислити синфазну і диференціальну складові за формулами:

$$U_{синф} = (U_{вх1} + U_{вх2})/2;\ U_{диф} = (U_{вх1} - U_{вх2})/2.$$

Отже

$$U_{вх1} = U_{синф} + U_{диф};\ U_{вх2} = U_{синф} - U_{диф}.$$

Зазвичай для диференціальних каскадів важко підібрати достатньо близькі за параметрами транзистори і навіть резистори колекторів, тому на практиці стали конструювати спарені транзистори, виготовлені в одному технологічному режимі. Такі транзистори не потрібні підбирати – вони створюються спеціально схожими, щоб одержувати дуже низький коефіцієнт підсилення синфазного сигналу $К_{синф}$. А при переході на мікроелектроніку взагалі всі диференціальні каскади стали виготовляти інтегральним способом. Звичайно в цьому випадку $К_{диф}$ = 100 ... 400, а $К_{синф}$ = 0,1 ... 1. Для оцінки якості диференціального каскаду вводять коефіцієнт ослаблення синфазного сигналу (КОСС):

$$КОСС = К_{диф}/ К_{синф}$$

Цей коефіцієнт лежить в межах 400 ... 1000 (або в децибелах 50...60 дБ).

Синфазний сигнал дуже важливий. Річ у тому, що різні дрейфи транзисторів (старіння, тепловий дрейф і т. д.) є еквівалентними подачі на входи однакових сигналів, тобто синфазному сигналу. Тому якщо синфазний сигнал сильно ослаблений, то і тепловий дрейф теж ослаблений. Коефіцієнт посилення диференціального сигналу в 1000 разів сильніший теплового дрейфу. Але це означає, що диференціальний каскад годиться для першого каскаду підсилювача, який буде призначений для посилення з великим коефіцієнтом посилення, щоб потім використовувати його для підсилювача з НЗВ. Такі підсилювачі називаються *операційними*).

Отже, майже завжди для виготовлення операційних підсилювачів роблять першим каскадом диференціюючим. Але у різних операційних підсилювачах він буває різним. Часто замість звичайних транзисторів беруть здвоєні (рис. 7.3).

Прийняте в мікроелектроніці умовне позначення транзисторів без кружечка означає, що у транзистора є свій корпус.

У такого каскаду коефіцієнт посилення здвоєних транзисторів значно більше ($100^2 = 10000$), ніж у одинарного. Саме із-за великого коефіцієнта посилення вони й використовуються.

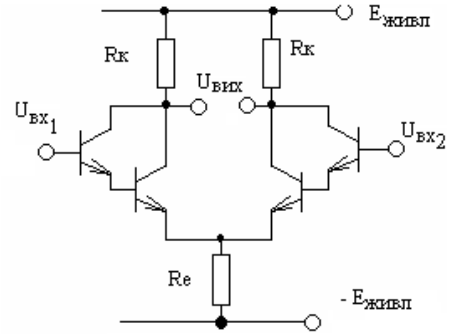


Рис. 7.3. Схема вхідного диференціюючого каскаду операційного підсилювача на здвоєних транзисторах

Але можна використовувати супер-бета транзистори. Це спеціально виготовлені транзистори з дуже маленькою базою і великими перепадами концентрацій в емітерній і базовій областях. Коефіцієнт підсилення у них може досягати 5000 і більш. Ці транзистори вимагають дуже точної технології, і, крім того, вони не витримують великих напруг. Тому для захисту від пробою до них треба додавати ще по одному транзистору. Із-за великої технологічної складності супер-бета транзистори використовуються рідко.

Іноді вхідні каскади корисно зробити на основі польових транзисторів, оскільки вони мають дуже великий вхідний опір. Частіше використовують польові транзистори з *p*-*n*-переходом. Але все-таки це теж дуже велике ускладнення технології.

Тому в більшості операційних підсилювачів використовують одинарні біполярні транзистори, але вживають заходів щодо поліпшення генератора струму емітера, для чого замість резистора використовують

транзистор. Але найчастіше для цієї мети використовується схема, яка називається “струмове дзеркало” (рис. 7.4).

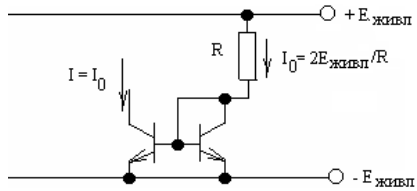


Рис. 7.4. Схема спарених транзисторів з назвою “струмове дзеркало”

Тут використані два однакові транзистори (краще ті, які виготовлені в одному технологічному циклі), і через правий, включений за схемою діода (колекторний *p-n*-перехід закорочений, і залишається тільки емітерний *p-n*-перехід) пропускається прямий струм. Цей струм визначається за формулою

$$I_0 = (2E_{живл} - 0{,}7) / R.$$

Цей струм постійний. Отже й напруги на його базі і базі сусіднього транзистора однакові й такі, що забезпечують протікання такої величини струму і через сусідній транзистор. Вийшло як би дзеркало: струм, який протікає через правий транзистор, протікає і через лівий (відображається). Але цей струм не залежить від напруги на колекторі лівого транзистора. Тобто вийшов генератор струму з дуже великим вихідним опором, рівним диференціальному опору колектора, який складає 100 кОм ...10 МОм. Якщо використовувати такий генератор струму, вийде збільшення КОСС до 1 000 000 (120 дБ).

У диференціальному каскаді вихід повинен бути різницевим. Це означає, що його не можна заземлити. За допомогою струмового дзеркала можна зробити пристрій, що віднімає (рис. 7.5).

Два верхні транзистори є транзисторами *p-n-p* типу. Правий верхній транзистор, як і у струмового дзеркала, використовується як діод (база-колектор закорочені). Тому він точно пропускає струм, який проходить через правий транзистор диференціального каскаду. І цей же струм проходить через лівий транзистор струмового дзеркала. Але по схемі він сполучений з колектором лівого транзистора диференціального каскаду. Виходить, що нижній транзистор дає струм I_1, а верхній

– струм I_2. І як наслідок, різниця цих струмів передається до наступного каскаду.

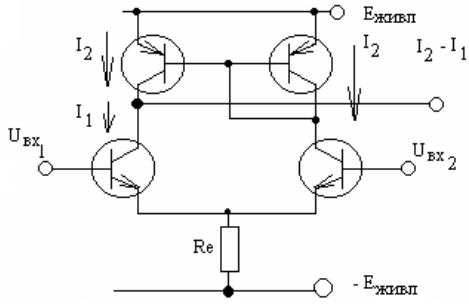


Рис. 7.5. Схема пристрою, що віднімає

По суті справи відбулася заміна колекторного опору активним навантаженням. Це навантаження має дуже великий диференціальний опір, а значить, дає ще більше посилення каскаду.

Тепер розглянемо наступний каскад посилення. Тут не потрібно боротися з температурним дрейфом, оскільки сигнал, що надійшов, має велике значення. Тому можна узяти звичайний каскад з ЗЕ, але для більшого коефіцієнта посилення виконаний на здвоєному транзисторі (рис. 7.6).

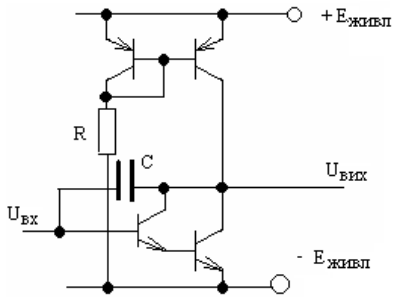


Рис. 7.6. Схема другого каскаду

Основні транзистори – це здвоєний транзистор внизу, включений відповідно до схеми з загальним емітером (ЗЕ). На базу цього транзис-

тора подається вхідний сигнал. У колекторі транзистора стоїть активне навантаження – другий транзистор струмового дзеркала. Крім того, тут зображений конденсатор *C*, який виконує корекцію частотної характеристики. Ця корекція необхідна для запобігання нестабільності операційного підсилювача. Слід зазначити, що він не завжди включається в схему. Існують операційні підсилювачі і без корекції. Тоді, у разі виникнення нестабільності, слід використовувати конденсатор у зворотному зв'язку всього операційного підсилювача.

Подальше підсилення в операційному підсилювачі неможливе, оскільки підсилювач з трьома каскадами посилення стає нестабільним. Проте можна зробити посилювач потужності за рахунок каскаду з ЗК.

Схема третього каскаду зображена на рис. 7.7, *а*.

Пунктирна лінія відокремлює ліву частину – деталі другого каскаду – від правої частини – деталей третього каскаду. У третьому каскаді два транзистори, включені за схемою з ЗК, але двотактною. Коли напруга позитивна, то відкритий і верхній транзистор, а нижній виконує роль дуже великого опору, оскільки він закритий. І навпаки, при негативній напрузі працює (відкритий) нижній транзистор, а верхній – закритий і виконує роль великого опору. Це двотактний емітерний повторювач.

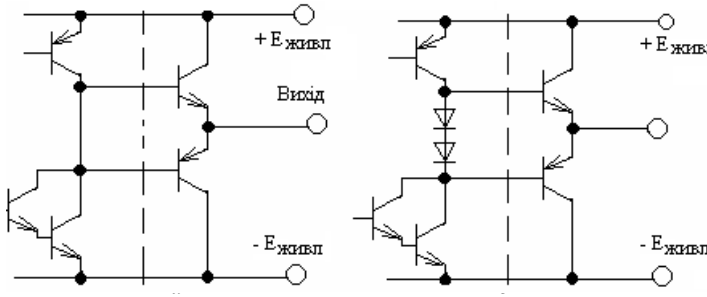


Рис. 7.7. Схема третього каскаду операційного підсилювача

Складність виникає, коли напруга мало відрізняється від нуля (менше, ніж на контактну різницю потенціалів), оскільки в цьому випадку обидва транзистори практично закриті. Рішенням цієї проблеми є включення у вихідний ланцюг двох діодів, як вказано на рис. 7.7, *б*. Ці діоди включені так, що вони завжди відкриті – на проходження струму у вихідному колі другого каскаду вони не впливають.

На рис. 7.8 наведена повна схема операційного підсилювача. Тут 12 транзисторів і 2 діоди. Але каскадів всього 3, та й то третій не підсилює напругу, а підсилює тільки струм або потужність.

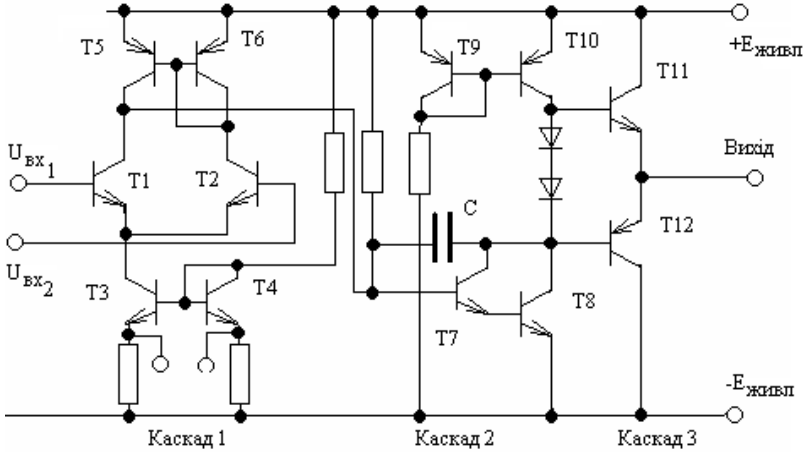


Рис. 7.8. Повна схема операційного підсилювача

У цій схемі транзистори Т1 і Т2 поєднаються паралельно, для виключення температурного дрейфу, а отже, і дрейфу, пов'язаного зі старінням схеми. Транзистори Т3, Т4 можуть використовуватися для збільшення коефіцієнта підсилення. Вони є допоміжними для генератора струму (струмового дзеркала Т3, Т4). Транзистори Т5, Т6 використовуються як активне навантаження. У дійсності ж у першому каскаді може використовуватися ще більше транзисторів, наприклад, для захисту від перевантаження.

У другому каскаді 4 транзистори: два транзистора (Т7, Т8) здвоєні – для підсилювання і Т9, Т10 – для активного навантаження (струмове дзеркало). Крім того, тут використовуються два діоди, а в мікроелектроніці замість діодів, як правило, використовуються транзистори. Всього виходить 6.

Найпростішим є останній каскад: у ньому всього два транзистори – Т11 та Т12.

Сучасні операційні підсилювачі виготовляються тільки за технологією мікросхемотехніки. А в мікросхемотехніці дуже просто робити транзистори, дещо складніше – діоди і резистори, ще складніше – кон-

денсатори і зовсім складно – індуктивності. Тому кількість транзисторів абсолютно неістотне (правда, дещо складніше робити комплементарні транзистори). У сучасних ОПід кількість транзисторів досягає 50 штук і більш. Але при сучасній нагоді виготовляти мікросхеми із ступенем інтеграції в 10^6.

Великий коефіцієнт посилення потрібний для застосування НЗЗ. Завдяки диференціальному каскаду, що складається з 6 ... 10 транзисторів, вдається усунути температурний та інші типи дрейфів. Коефіцієнт посилення першого каскаду можна підвищити до 1000 і більш.

Вхідні опори виходять не дуже великими, але вони різні для синфазних (великі) і диференціальних (трохи менше) сигналів при застосуванні біполярного транзистора. Якщо поставити польові транзистори, то вхідний опір буде більший. Ще більшого вхідного опору можна досягти при застосуванні в диференціальних каскадах супер-бета транзисторів.

Малий вихідний опір обумовлений застосуванням каскаду з ЗК, який підсилює струм.

Ще одна важлива характеристика – швидкодія. Вона визначається верхньою межею частотної характеристики, оскільки нижньої межі у операційних підсилювачів немає.

У зв'язку з тим, що в цієї характеристики існує ділянка з нахилом в 45^0, яка доходить до одиничного посилення (у логарифмічному масштабі 0), операційний підсилювач ніколи не буде самозбуджуватися.

ОПід ніколи не використовуються без НЗЗ.

Розглянемо найбільш широко використовувані різновиди операційних підсилювачів, для живлення яких застосовуються два джерела напруги (– 15 В і + 15 В). По-іншому це називають живленням від джерела з нульовим виводом або від розщепленого джерела ± 15В.

Наведемо один з варіантів умовного графічного позначення операційного підсилювача (рис. 7.9).

Часто на схемах виводи + U, – U і 0V не вказують і використовують спрощене умовне графічне позначення.

У зарубіжній літературі часто використовують інше умовне графічне позначення (рис. 7.10).

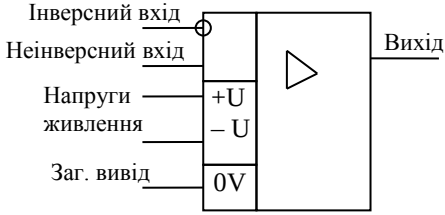


Рис. 7.9. Умовне графічне позначення операційного підсилювача

Позначимо напруги на виводах операційного підсилювача (рис. 7.11).

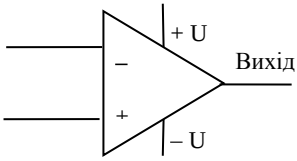


Рис. 7.10. Альтернативне умовне графічне позначення операційного підсилювача

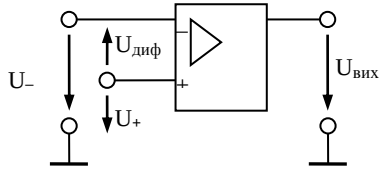


Рис. 7.11. Схема операційного підсилювача з позначеними напругами

Напруга $U_{диф}$ між напругами U_+ та U_- називають диференціальною напругою (диференціальним сигналом). Ясно, що

$$U_{диф} = U_+ - U_-.$$

Операційні підсилювачі конструюють так, щоб вони якомога більше змінювали напругу $U_{вих}$ при зміні диференціального сигналу та менше змінювали напругу $U_{вх}$ при однаковій зміні напруг U_+ та U_-..

Передавальна характеристика. Операційний підсилювач добре характеризує його передавальна характеристика – залежність вигляду

$$U_{вих} = f(U_{диф})$$

де f – деяка функція.

Зобразимо графік цієї залежності (рис. 7.12) для операційного підсилювача К140УД1Б (це один з перших вітчизняних операційних підсилювачів).

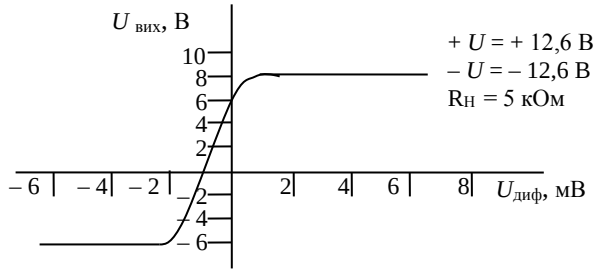


Рис. 7.12. Передавальна характеристика операційного підсилювача

Ця конкретна характеристика не проходить через початок координат. У різних екземплярів операційних підсилювачів одного й того ж типу ця характеристика може проходити як ліворуч, так і праворуч від початку координат. Передбачити точне положення цієї характеристики неможливо. Значення напруги $U_{диф}$, при якій виконується умова $U_{вих} = 0$, називають ***напругою зміщення*** (напругою зміщення нуля) і позначають через $U_{зм}$. Для операційного підсилювача К140УД1 відомо тільки те, що напруга $U_{зм}$ лежить в діапазоні від – 10 мВ до +10 мВ. А це означає, що при нульовій напрузі $U_{диф}$ напруга $U_{вих}$ може лежати в межах від мінімально можливого (біля –7 В) до максимально можливого (біля + 10 В).

Для того, щоб при нульовому підсилюваному сигналі напруга на виході дорівнювала нулю, тобто для того, щоб передавальна характеристика проходила через початок координат, передбачають заходи щодо компенсації напруги зсуву (балансування, корекція нуля). У деяких операційних підсилювачах (у тому числі і типу К140УД1Б) не передбачені спеціальні виводи, впливаючи на які можна було б компенсувати напругу зміщення. В цьому випадку на входи операційного підсилювача окрім підсилюваного сигналу, потрібно подавати напругу, яка компенсує напругу зміщення: У деяких операційних підсилювачах для компенсації напруги зміщення передбачені спеціальні виводи. Зобразимо типову схему включення операційного підсилювача типу К140УД8А, в якому передбачені такі виводи (рис. 7.13.).

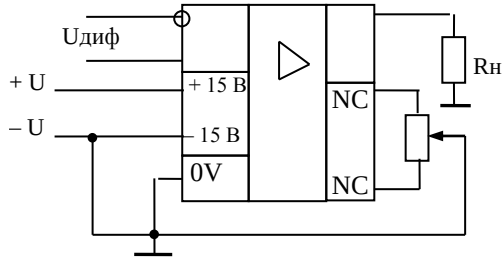


Рис. 7.13. Типова схема включення операційного підсилювача типу К140УД8А

Через NC позначені спеціальні виводи для балансування.

Діапазон вихідної напруги, відповідний майже вертикальній ділянці передавальної характеристики, називається ***областю посилення***. Відповідний цьому діапазону режим роботи називають режимом посилення (лінійним, активним режимом).

У лінійному режимі

$$U_{вих} = K\,U_{диф},$$

де K – коефіцієнт посилення за напругою (коефіцієнт посилення напруги, коефіцієнт посилення диференціального сигналу).

Завдяки великому коефіцієнту посилення (сучасні операційні підсилювачі мають коефіцієнт посилення К = 10^5 ...10^6) і малим вхідним струмам, підсилювачі, побудовані на базі операційних підсилювачів, мають унікальні властивості. Зокрема, параметри багатьох пристроїв визначаються тільки зовнішніми колами – колами зворотного зв'язку, що сполучають вихід операційного підсилювача з його входом.

Для з'ясування призначення виводів зобразимо типову схему на операційному підсилювачі – схему інвертуючого підсилювача (рис. 7.14).

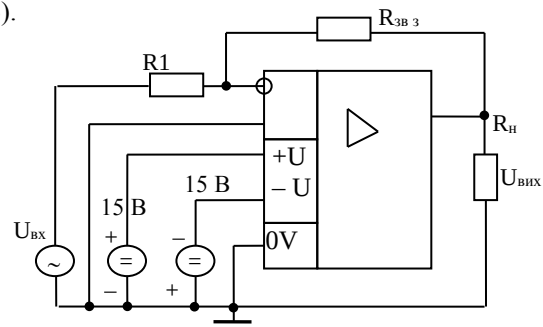


Рис. 7.14. Умовне графічне позначення операційного підсилювача

Наприклад, ***коефіцієнт посилення*** підсилювача, схема якого показана на рис. 7.15, а, визначається з високою точністю відношенням опорів двох резисторів:

$$K = - R_{зв\ з}/R_1.$$

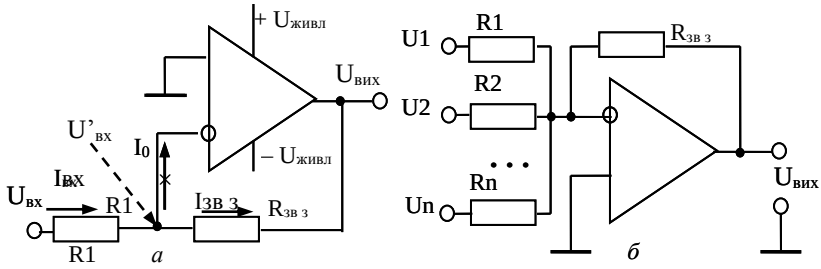


Рис. 7.15. Схеми підсилювачів на операційних підсилювачах: *а* – інвертуючий підсилювач; *б* – підсилювач додавання

Якщо вхідна напруга $U_{вх}$ достатньо мала за модулем, то вихідна напруга $U_{вих}$ визначається виразом:

$$U_{вих} = - U_{вх}R_{зв\ з}/R_1.$$

Тепер розглянемо протікання струму в точці з'єднання опорів. Зліва втікає вхідний струм $I_{вх}$, справа витікає струм зворотного зв'язку $I_{зв\ з}$, і ще можлива поява струму, що протікає до інвертуючого входу операційного підсилювача. Але останній дуже малий. Річ у тому, що вхідний опір операційного підсилювача зазвичай високий, а вхідна напруга дуже мала. Тому цей струм вимірюється нано- і навіть пікоамперами, і ним можна знехтувати (на рис. цей струм перекреслений).

Отже, є тільки два струми: $I_{вх}$ і $I_{зв\ з}$, які мають бути однаковими:

$$I_{вх} = I_{зв\ з};$$

де

$$I_{вх} = (U_{вх} - U'_{вх}) / R_1;$$

$$I_{зв\ з} = (U'_{вх} - U_{вих}) / R_{зв\ з}.$$

Тому, що $U'_{вх} \approx 0$

$$U_{вх}/R_1 = - U_{вих}/R_{зв\ з}$$

У зв'язку з тим, що $K_{зв\ з} = U_{вих}/U_{вх}$, отримаємо

$$K_{зв\ з} = - R_{зв\ з}/R_1.$$

Отже, отримано коефіцієнт підсилення з інверсією. Його значення на одиницю менше, ніж у неінвертуючого підсилювача на операційному підсилювачі. Якщо $R_{зв\ з} = R_1$, то $K_{зв\ з} = - 1$. Таким чином при такому з'єднанні реалізується інвертор.

Якщо на інвертуючий вхід підсилювача подати сигнал від декількох джерел (рис. 7.15, *б*), то вихідний сигнал визначається як добуток суми вхідних струмів на величину опору резистора зворотного зв'язку:

$$U_{вих} = -R_{зв\ з}(I_{вх1} + I_{вх2} + \ldots + I_{вх.n}).$$

Вхідний струм від кожного джерела визначається як співвідношення

$$I_{вх} = U_{вх}/R_i,$$

де R_i – опір резистора в колі *i*-го входу.

Властивість операційного підсилювача додавати вхідні струми з подальшим перетворенням в напругу широко використовується при побудові ЦАП і АЦП. На базі операційного підсилювача можна побудувати **компаратори напруги** (пристрої, що порівнюють). При використанні операційного підсилювача як компаратор напруги на один його вхід подається опорна напруга Uоп, на другій – напруга оброблюваного (перетворюваного) сигналу Ux. За відповідних умов на виході компаратора формується сигнал логічної 1, якщо ($U_{оп} - U_x$) > $\Delta U_{кв}$, і логічного 0, якщо ($U_{оп} - U_x$) < ΔUкв (рис. 7.16).

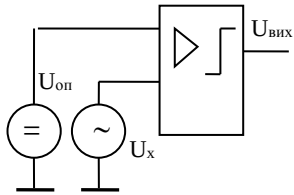


Рис. 7.16. Схемне позначення компаратора напруг

Крок квантування $\Delta U_{кв}$ звичайно вибирається в межах 5 ... 10 мВ. Значення опорної напруги і час установки компаратора залежать від конкретного типу використовуваної інтегральної мікросхеми і умов його експлуатації.

Логарифмічний підсилювач

Схема логарифмічного підсилювача наведена на рис. 7.17.

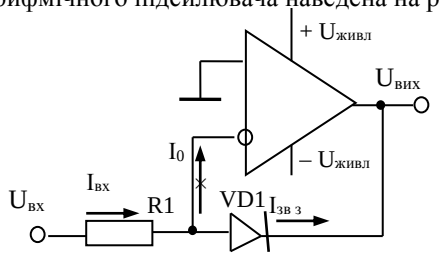


Рис. 7.17. Схеми логарифмічного підсилювача

Для побудови підсилювачів з логарифмічною амплітудною характеристикою часто використовується пряма гілка вольт-амперної характеристики *p-n*-переходу. В цій схемі замість регістра зворотного зв'язку використовується діод.

В такому випадку вирази для струмів будуть мати вигляд

$$I_{вх} = U_{вх}/R_1 = I_{зв\ з} = I_0 \exp(-gU_{вих}/kT),$$

де $U_{вих}$ – напруга на виході операційного підсилювача;

T –абсолютна температура в градусах Кельвина;

q та k – зарядка одного електрона та постійна Больцмана.

Вирішивши це рівняння щодо $U_{вих}$, отримаємо

$$U_{вих} = -kT/g \times \lg(U_{вх}/R_1 I_0).$$

За останньою формулою й функціонує логарифмічний підсилювач.

7.2. Аналого-цифрове та цифроаналогове перетворення

7.2.1. Поняття про аналого-цифрове перетворення

Основними напрямами використання методів цифрової обробки сигналів є цифрова фільтрація, спектральний аналіз, цифрові системи управління. Найважливіша проблема цифрової обробки – квантування за рівнем і дискретизація за часом сигналів (перетворення). Причому залежно від призначення цифрової системи необхідно розрізняти переваги і недоліки кожного методу перетворення.

При ***квантуванні за рівнем*** діапазон можливих змін аналогової величини – інтервал (а, б) – розбивається на n інтервалів квантування: $\Delta x_i = x_i - x_{i-1}$, $i = 1, 2 ..., n$ з межами $x_0 = a$, $x_1, ... , x_{n-1}, x_n = b$. У результаті квантування будь-яке зі значень, що належить інтервалу (x_i, x_{i-1}) округляється до деякої величини $\Delta x_{кв}$. Величини $\Delta x_{кв}$ називають ***рівнем (кроком) квантування***. У результаті квантування за рівнем проводиться заміна дійсних значень величини x дискретними значеннями рівнями квантування $\Delta x_{кв}$.

У процесі ***дискретизації за часом*** поточні значення сигналів у вигляді дискретних величин фіксуються в моменти часу $t_j = j\Delta t$, $j = 0, 1, 2..., k$. При введенні інформації в ЕОМ проводиться квантування за рівнем і дискретизація за часом.

Квантування може бути ***рівномірним*** ($\Delta x = \text{const}$, $\Delta t = \text{const}$) і ***нерівномірним*** ($\Delta x \neq \text{const}$, $\Delta t \neq \text{const}$).

7.2.2. Цифроаналоговий перетворювач

Цифро аналоговий перетворювач (ЦАП) призначений для перетворення цифрової інформації в аналогову форму, тобто вихідний сигнал ЦАП в загальноприйнятих одиницях вимірювання (мВ, В, мА) відповідає чисельному значенню вхідної кодової комбінації.

Наприклад, при подачі на вхід ЦАП кодової комбінації (у десятковому еквіваленті), яка дорівнює 150, на його виході з'являється напруга 1500 мВ. Це означає, що зміна значення вхідної кодової комбінації (вхідного числа) на одиницю приводить до зміни вихідної напруги на 10 мВ. У цьому випадку одержуємо ЦАП з кроком перетворення цифрової інформації 10 мВ. При подачі на вхід ЦАП послідовної цифрової комбінації, яка змінюється від 0 до *N*, на його виході з'явиться ступінчато-наростаюча напруга (рис. 7.18).

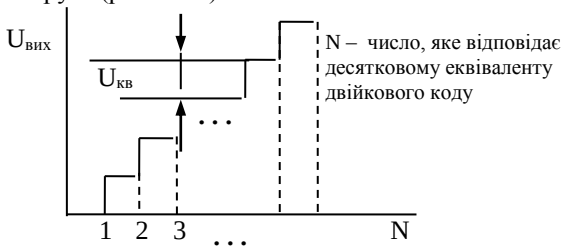


Рис. 7.18. Діаграма вихідної напруги ЦАП

Висота кожного ступеня відповідає одному кроку квантування $\Delta U_{кв}$.

До основних параметрів ЦАП відносять дозволяючу спроможність, час встановлення, похибку нелінійності та ін.

Дозволяюча спроможність – величина, зворотна максимальній кількості кроків квантування вихідного аналогового сигналу.

Час встановлення $t_{вст}$ – інтервал часу від подачі коду на вхід до моменту, коли вихідний сигнал увійде до заданих меж, визначуваних похибкою.

Похибка нелінійності – максимальне відхилення графіка залежності вихідної напруги від напруги, що задається цифровим сигналом, по відношенню до ідеальної прямої у всьому діапазоні перетворення.

Види ЦАП умовно можна розділити на дві групи: з матрицями резисторів, безматричні. У інтегральному виконанні застосовуються тільки ЦАП з прецизійними матрицями резисторів, що формують вихідні сигнали шляхом підсумовування струмів.

ЦАП містить елементи цифрової і аналогової схемотехніки. Як аналогові елементи використовуються операційні підсилювачі, аналогові ключі (комутатори), матриці, резисторів, і т.д.

Аналогові елементи, що входять в склад ЦАП, практично повністю визначають його якісні і експлуатаційні параметри, основну роль при цьому відіграють точність підбору номіналів резисторів матриці, резистора, і параметрів операційного підсилювача.

Найбільш простими, точними і надійними схемами перетворення коду в напругу є схеми цифроаналогових перетворювачів (ЦАП) з ви-

користанням операційних підсилювачів. У основу перетворення покладено ступінчасте змінення коефіцієнта передачі масштабного підсилювача коду, що пропорційно задається, комутацією опорів на входах. Схема 4-розрядного ЦАП наведена на рис. 7.19.

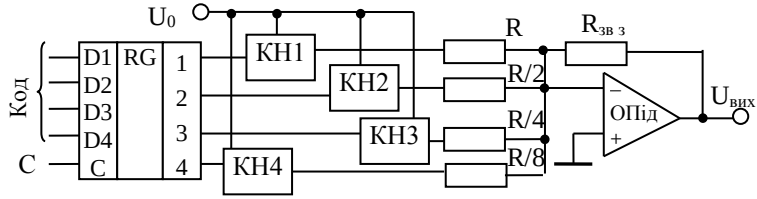


Рис. 7.19. Схема ЦАП із операційним підсилювачем, що додає

Схема містить регістр RG, комутатори напруги КН, операційний підсилювач ОПід, опір зворотного зв'язку $R_{зв\ з}$, вхідні опори: R, $R/2$, $R/4$, $R/8$. Комутатори напруг КН спрацьовують під час надходження сигналу одиниці з прямих виходів регістра. В результаті комутації струми, що протікають через вхідні опори, підсумовуються. Вихідна напруга схеми

$$U_{вих} = -\Sigma b_i R_{зв.\ з} U_0 / R_i.$$

де b_i – значення i-розряду регістра RG.

Опори, що комутують, можуть бути також включені до кола зворотного зв'язку.

Розглянемо ЦАП на основі резистивної матриці R – 2R (матриці постійного опору) (рис. 7.20).

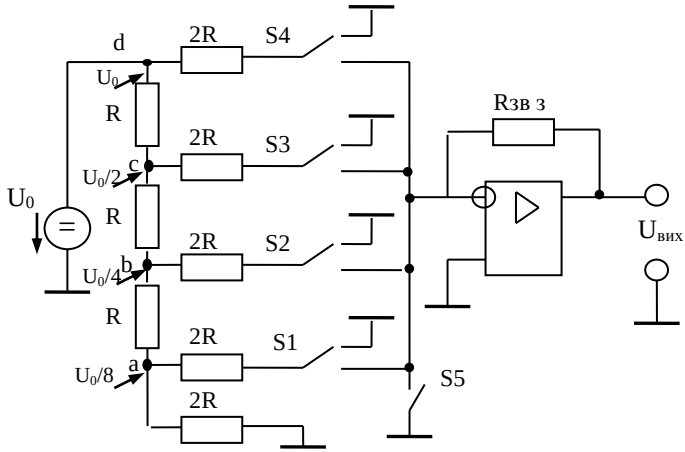


Рис. 7.20. ЦАП на основі резистивної матриці R – 2R

У схемі використані так звані ключі S1...S4 (еквіваленти двійкового коду), кожний з яких в одному зі станів підключений до загальної точки, тому напруги на ключах невеликі. Ключ S5 замкнутий тільки тоді, коли всі ключі S1...S4 підключені до загальної точки (при необхідності перетворити цифровий код 0000). У цьому випадку на інверсний вхід треба підключити 0.

У вхідному колі використані резистори всього з двома різними значеннями опорів. З аналізу схеми можна побачити, що для неї модуль вихідної напруги пропорційний числу, двійковий код якого визначається станом ключів S1...S4. Хай кожний з ключів S1...S4 підключений до загальної точки. Тоді, напруга щодо загальної точки в кожній наступній з точок a ... d в 2 рази більше, ніж в попередній. Наприклад, напруга в точці b в 2 рази більше, ніж в точці a (напруги U_a, U_b, U_c і U_d у вказаних точках визначаються таким чином:

$$U_d = U_0,\ U_c = U_0/2,\ U_b = U_0/4,\ U_a = U_0/8$$

Допустимо, що стан вказаних ключів змінився. Тоді напруги в точках *a ... d* не зміняться, оскільки напруга між входами операційного підсилювача практично нульова.

Для прикладу, в табл. 7.1 наведені деякі ЦАП та їх основні характеристики.

Таблиця 7.1

Тип схеми	Кіл. розрядів	$t_{вст}$, мкс	U_0, В	$U_{жив}/I_{жив}$, В/А	$I_{вих}$, мА
К594ПА1	12	3,5	9 ÷ 11	(5 ÷ 15)/2,5 ÷ 15/3,5	2
К1108ПА1	12	0,4	2,2 ÷ 10,5	+5 /15 ÷ 16/46	5
К572ПА1А	10	5	– 17 ÷ 17	(5 ÷ 17)/2	1
К575ПА2А	10	15	– 15 ÷ 15	5/2 ÷ 15/2	0,8

7.2.3. Аналого-цифрові перетворювачі

Аналого-цифрові перетворювачі (АЦП) – це пристрої, які призначені для перетворення аналогових сигналів в цифрові. Для такого перетворення необхідно здійснити квантування аналогового сигналу, тобто миттєві значення аналогового сигналу обмежити певними рівнями, званими рівнями квантування.

Квантування – це округлення аналогової величини до найближчого рівня квантування, тобто максимальна похибка квантування дорівнює ± 0,5h (h – крок квантування).

До ***основних характеристик*** АЦП відносять число розрядів, час перетворення, нелінійність і ін.

Кількість розрядів – кількість розрядів коду, пов'язаного з аналоговою величиною, яка може вироблятися АЦП.

Дозволяюча спроможність –величина, зворотна максимальній кількості кодових комбінацій на виході АЦП. Так, 10-розрядний АЦП має роздільну здатність $2^{10} - 1 = 1024 - 1$, що при шкалі АЦП, яка дорівнює 10 В, має абсолютне значення кроку квантування, яке не перевищує 10 мВ.

Час перетворення $t_{пр}$ – інтервал часу від моменту заданої зміни сигналу на вході АЦП до появи на його виході відповідного стійкого коду.

За принципом дискретизації і структури побудови АЦП діляться на дві групи: 1-група АЦП із застосуванням ЦАП і 2-група АЦП без застосування ЦАП.

До першої групи відносяться:

- АЦП послідовної лічби (тип, що розгортає);
- АЦП послідовного наближення (порозрядного урівноваження);
- АЦП послідовного наближення, що стежить.

До другої групи відносяться:

- АЦП прямого перетворення;
- АЦП подвійної інтеграції;
- АЦП із застосуванням генератора, керованого напругою.

Кожен тип АЦП має свої переваги і недоліки. На практиці зустрічаються всі вище зазначені типи АЦП.

Паралельне та послідовне перетворення є основними методами, які використовуються при побудові АЦП.

Розглянемо АЦП з паралельним перетворенням вхідного аналогового сигналу. При паралельному методі вхідну напругу одночасно порівнюють з n опорними напругами. При цьому результат одержують швидко, але схема виявляється достатньо складною.

Розглянемо принцип дії АЦП із паралельним перетворенням аналогової величини (рис. 7.21).

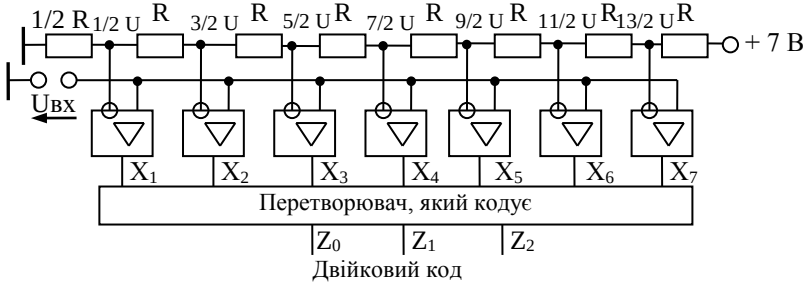


Рис. 7.21. Схема АЦП із паралельним перетворенням аналогової величини

При $U_{вх} = 0$ напруги на виході всіх операційних підсилювачів дорівнюють – $E_{живл}$, а на виходах перетворювача, який кодує (КП) Z_0, Z_l Z_2, встановлюються нулі. Якщо ($U_{вх} > 0{,}5U$, але менше $3/2U$, лише для нижнього операційного підсилювача $U_+ - U_- > 0$ і лише на його виході з'являється напруга $E_{живл}$, що приводить до появи на виходах КП сигналів: $Z_0 = 1$, $Z_2 = Z_1 = 0$. Якщо $U_{вх} > 3/2U$, але менше за $5/2U$, то на виході двох нижніх операційних підсилювачів з'являється напруга + $E_{живл}$, що приводить до появи на виходах КП коду 010 і т.д. Розглянемо схему АЦП **послідовного наближення** вимірювального перетворення, яке стежить (рис. 7.22, *а*).

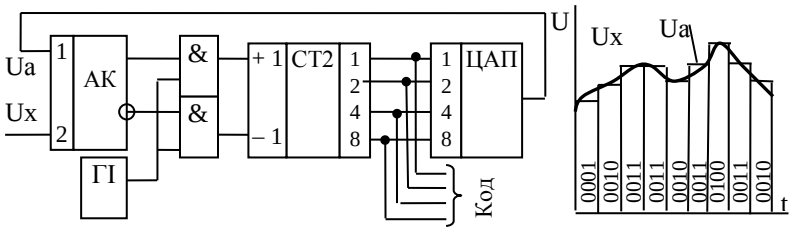


Рис. 7.22. Схема АЦП послідовного наближення вимірювального перетворення, яке стежить, та його часова діаграма

Схема містить реверсивний лічильник СТ2, ЦАП, генератор імпульсів ГІ, амплітудний компаратор АК, дві схеми І. АЦП – цифрова система, яка стежить та в якій вхідна напруга U_x порівнюється АК з апроксимуючою напругою U_a, що надходить з ЦАП.

При $U_x - U_a \geq 0$ СТ2 працює на додавання, при $U_x - U_a < 0$ – на віднімання. Мінімальна напруга спрацьовування $U_{ср}$ АК повинна лежати в межах $\Delta U/2 < U_{ср} < \Delta U$, де ΔU – крок квантування за рівнем. Обробка цифрових еквівалентів аналогової величини U_x показана на рис. 7.22, *б*. Внизу записані двійкові еквіваленти значень U_x. Точність перетворення в даних АЦП залежить від стабільності параметрів АК, кількості розрядів СТ2 і складає 0,1 – 0,05 %.

Перетворювачі з вимірювальним перетворенням, яке стежить, мають низьку швидкодію, що обмежується зростанням кількості операцій порівняння при різкому зростанні вхідної напруги U_x. Цей недолік усунений в АЦП з порозрядним урівноваженням.

Схема 3-розрядного АЦП з порозрядним урівноваженням наведена на рис. 7.23, *а*.

Вона містить автомат, що управляє, УА, ЦАП, АК і три D-тригери.

На початку циклу перетворення (урівноваження) сигнали на виходах УА мають значення Q1 = Q2 = Q3 = 0, Q4 = l. Сигналом Q4 тригер Т1 переводиться в стан 1. На інформаційні входи ЦАП із виходів D-тригерів надходить код 100.

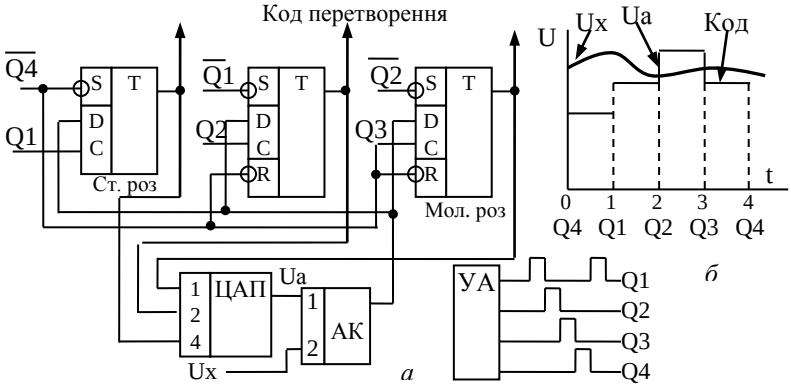


Рис. 7.23. Схема АЦП з порозрядним урівноваженням

Вихідна напруга ЦАП порівнюється АК з вхідною напругою U_x. При $U_a < U_x$ вихідний сигнал АК дорівнює одиниці, при $U_a \geq U_x$ – нулю. У момент надходження сигналу Q1 вихідний сигнал АК заноситься в D-тригер старшого розряду. Процес порівняння U_a, U_x і занесення результату в наступний розряд поновлюється з приходом сигналів Q2, Q3 (рис. 7.23, *б*). Після закінчення дії сигналу Q3 на прямих виходах D-тригерів остаточно формується цифровий еквівалент перетворюваної напруги U_x.

Метод послідовного перетворення реалізується і в АЦП із часово-імпульсного перетворення (АЦП з генератором напруги, що лінійно змінюється (ГЛЗН)). Принцип дії такого АЦП (рис. 7.24) заснований на підрахунку кількості імпульсів у відрізку часу, протягом якого напруга (ЛЗН), що лінійно змінюється, збільшуючись від нульового значення, досягає рівня вхідної напруги $U_{вх}$.

На рис. 7.24 використані такі позначення: СП – схема порівняння; ГІ – генератор імпульсів; Кл – електронний ключ; Ліч – лічильник імпульсів.

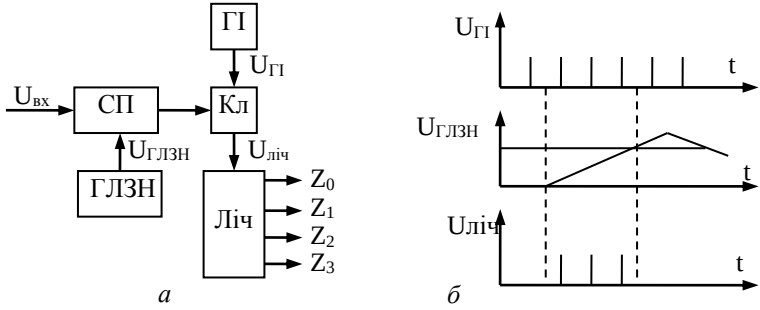


Рис. 7.24. Схема АЦП з порозрядним урівноваженням:
а – схема АЦП з порозрядним урівноваженням; *б* – часові діаграми

Момент часу t_1 відповідає початку вимірювання вхідної напруги, а момент часу t_2 відповідає рівності вхідної напруги і напруги ГЛЗН. Похибка вимірювання визначається кроком квантування часу. Ключ Кл підключає до лічильника генератор імпульсів від моменту початку вимірювання до моменту рівності $U_{вх}$ і $U_{ГЛЗН}$. Через $U_{ліч}$ позначено напругу на вході лічильника.

Код на виході лічильника пропорційний вхідній напрузі. Одним з недоліків цієї схеми є невисока швидкодія.

Розглянемо АЦП з подвійною інтеграцією, який також реалізує метод послідовного перетворення вхідного сигналу (рис. 7.25, *а*).

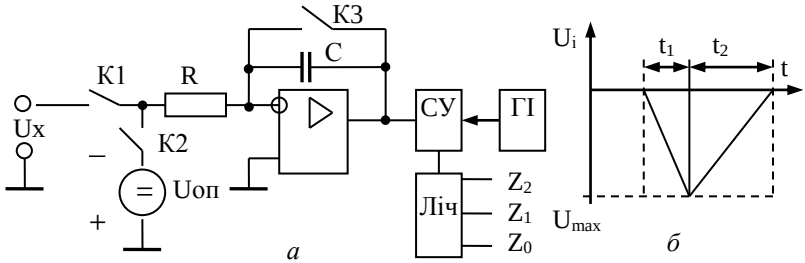


Рис. 7.25. Схема АЦП послідовного перетворення з подвійною інтеграцією та його часова діаграма

Використані наступні позначення: СУ – система управління, ГІ – генератор імпульсів, Ліч – лічильник імпульсів. Принцип дії АЦП полягає у визначенні відношення двох відрізків часу, протягом одного з

яких виконується інтеграція вхідної напруги $U_{вх}$ інтегратором на основі операційних підсилювачів (напруга $U_і$ на виході інтегратора змінюється від нуля до максимальної за модулем величини), а протягом наступного – інтеграція опорної напруги $U_{оп}$ ($U_і$ змінюється від максимальної за модулем величини до нуля) (рис. 7.25, *б*).

Нехай час t_1 інтеграції вхідного сигналу є постійним. Тоді чим більшим буде другий відрізок часу t_2 (відрізок часу, протягом якого інтегрується опорна напруга), тим більше вхідна напруга. Ключ К3 призначений для установки інтегратора в початковий нульовий стан. У перший з вказаних відрізків часу ключ К1 замкнений, ключ К2 розімкнений, а в другій відрізок часу їх стан є зворотним по відношенню до вказаного. Одночасно із замиканням ключа К2 імпульси з генератора імпульсів ГІ починають надходити через схему управління СУ на лічильник Ліч. Надходження цих імпульсів закінчується тоді, коли напруга на виході інтегратора виявляється рівною нулю.

Код на виході лічильника визначає величину вхідної напруги.

Однією з основних переваг АЦП даного типу є їх висока перешкодозахищеність. Випадкові викиди вхідної напруги, що мають місце протягом короткого часу, практично не впливають на похибку перетворення. Недоліком АЦП є мала швидкодія.

Найбільш поширеними є АЦП серій мікросхем 572, 1107, 1138 й ін. (табл. 7.2).

Таблиця 7.2

Тип м/сх	Кіл. розр.	$t_{пр}$, мкс	$U_{жив}$	$P_{жив}$,мВт	Перетворення
К1107ПВ1	6	0,1	+ 5; – 6	800	Парал.
К1107ПВ2	8	0,1	+ 5; – 6	3000	Парал.
КР572ПВ1	12	170	5 – 15; – 15	30	Послід.
КР572ПВ3	8	15	5	25	Послід.
КР572ПВ4	8	32	5	15	Послід.
К1108ПВ1	10	0,9	9; – 5,2	800	Послід.
К1138ПВ1А	10	30	5; – 15	225	Послід.

З таблиці видно, що найкращою швидкодією володіє АЦП паралельного перетворення, а найгіршою – АЦП послідовного перетворення.

7.2.4. Завдання до самостійних досліджень АЦП та ЦАП

1. *Мета*

Ознайомлення з роботою аналого-цифрових та цифроаналогових перетворювачів за допомогою інструментальних засобів цифрової частини пакета EWB, закріплення теоретичного матеріалу, набуття навиків створення і моделювання цифрових пристроїв.

2. *Теми для попереднього опрацювання*

2.1. Аналого-цифрові перетворювачі;

2.2. Цифроаналогові перетворювачі.

3. *Завдання*

3.1. Дослід 1. Побудувати схему дослідження бібліотечного ЦАП згідно з функціональною схемою, наведеною на рис. 7.28, та дискретністю згідно з виразом $\Delta x_{кв} = N + 30$, де N – номер за порядком.

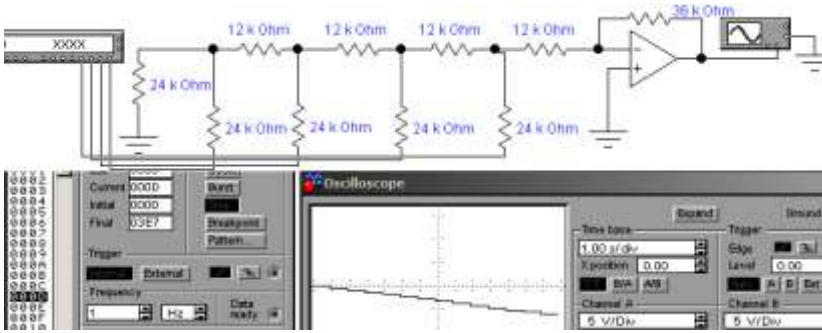


Рис. 7.28. Схема дослідження ЦАП на основі резистивної матриці R – 2R

Значення $\Delta x_{кв}$ занести до генератора слів та навести їх у звіті. За початковий стан прийняти всі нулі, а за останній стан – максимальне значення сигналу.

3.2. Дослід 2. Дослідження ЦАП на основі резистивної матриці R – 2R.

Зібрати ЦАП, послідовно подавати цифрові комбінації на входи та за допомогою осцилографа зняти рівні напруг (рис. 7.29).

Визначити номінали резисторів: *R*, 2*R*, 3*R*, та дослідити значення рівня квантування від значення резисторів. Навести схеми досліджень.

3.3. Дослід 3. Побудувати схему дослідження АЦП послідовної лічби (рис. 7.30).

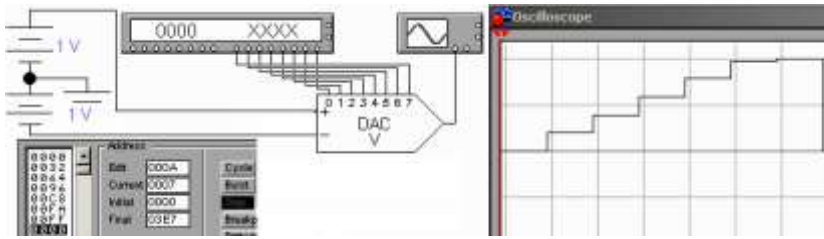


Рис. 7.29. Схема дослідження аналого-цифрового перетворювача

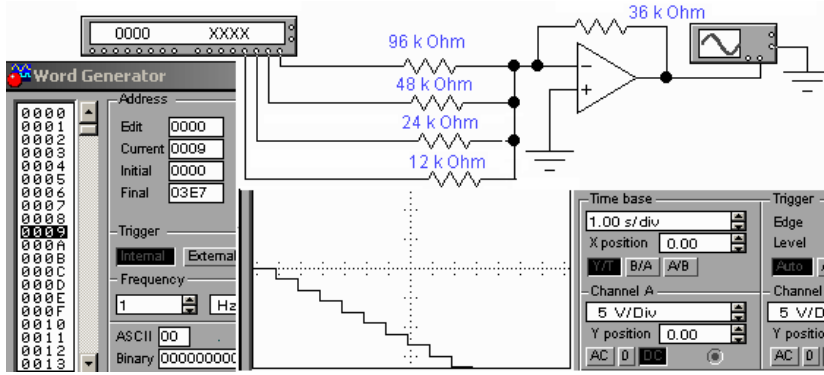


Рис. 7.30. Схема дослідження ЦАП послідовної лічби

Визначити номінали резисторів, дослідити значення рівня квантування від значення резисторів. Навести схеми досліджень.

Визначити номінали резисторів, дослідити значення рівня квантування від значення резисторів. Навести схеми досліджень.

4. *Питання для самоперевірки*

4.1. Які основні параметри визначення ЦАП та за допомогою чого вони досягаються?

4.2. Поясніть роботу ЦАП з резистивною матрицею *R*-2*R*?

4.3. Як впливає величина опору зворотного зв’язку операційного підсилювача на параметри ЦАП?

4.4. Чим обмежується швидкодія ЦАП?

4.5. Поясніть роботу АЦП послідовної лічби.

4.6. На який основний параметр впливає крок квантування та від чого він залежить?

8. СХЕМОТЕХНІКА МІКРОКОНТРОЛЕРА i8051

8.1. Загальна характеристика МК i8051

Мікроконтролер (МК) i8051 виконується на основі багаторівневої *n*-МОН технології та випускається у корпусі ВІС, що має 40 зовнішніх виводів. Для роботи МК51 необхідне одне джерело електроживлення напругою +5 В. Через чотири порти вводу–виводу, що програмуються, МК51 взаємодіє із середовищем у стандарті ТТЛ-схем із трьома станами виходу. Корпус МК51 має два виводи для підключення кварцового резонатора, чотири виводи для сигналів, управляючих режимом роботи МК51, та 8 ліній порту 3, що можуть бути запрограмовані користувачем на виконання спеціалізованих функцій обміну інформацією із зовнішнім середовищем.

Технічні дані МК51:

1. Внутрішній ПЗП (4 кбайти) з можливістю поширення до 64 кбайтів.
2. Внутрішній ОЗП (128 байтів).
3. 20 регістрів спеціального призначення.
4. Чотири банки регістрів загального призначення (РЗП).
5. Стек (теоретична глибина стека – 128 байтів).
6. 128 контрольованих користувачем окремих розрядів, що розміщені за фіксованими адресами в пам'яті даних.
7. 32 лінії вводу–виводу, організовані в чотири 8-розрядних порти.
8. Швидкодіючий програмований послідовний порт.
9. Два 16-розрядних таймери-лічильники.
10. Дворівнева система переривань.
11. П'ять джерел переривань.
12. Наявність команд множення та ділення.

Електричні параметри:

1. $U_{cc} = +5$ В $\pm$ 5%
2. f = 1, 2 ÷ 12 МГц

3. $P_{спож} = 2$ Вт

Діапазон робочої температури від 0 до + 70 °С.

Умовне графічне позначення МК51 наведене на рис. 8.1.

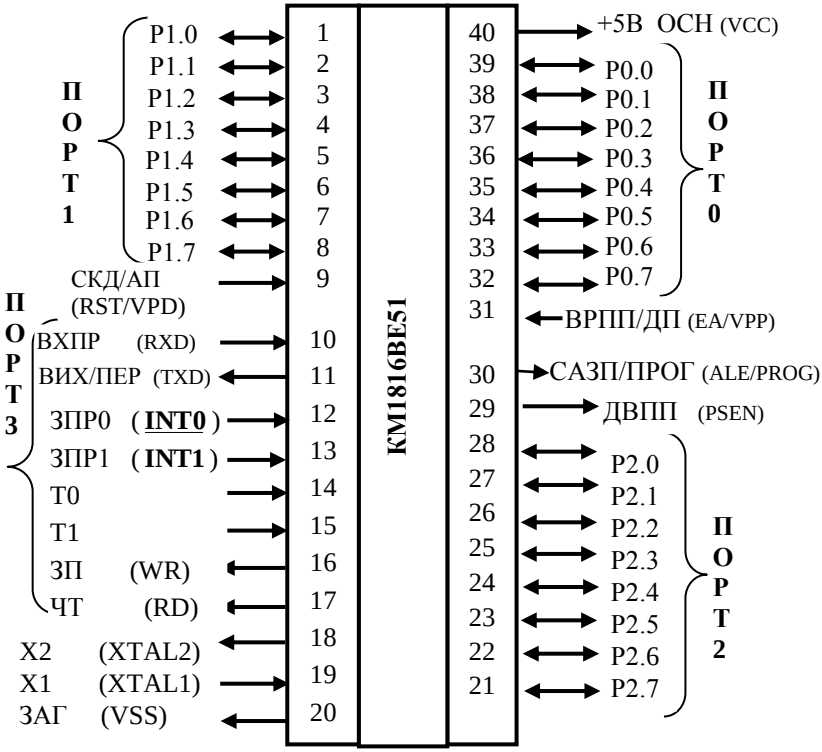


Рис. 8.1. Умовне графічне позначення МК51

Найменування виводів МК51:

- P1, P2, P3 – 8-розрядні двонаправлені порти;
- вхід/вихід RST – сигнал загального скидання;
- X1, X2 – виводи для підключення кварцового резонатора;
- вихід EA – блокування роботи з внутрішньою пам'яттю програм (при EA = 0 внутрішня пам'ять блокується);
- RxD – вхід послідовних даних;
- TxD – вихід передатчика;

- INT0 – вхід зовнішнього переривання 0;
- INT1 – вхід зовнішнього переривання 1;
- T0 – вхід лічильника T/C0;
- T1 – вхід T/C1 лічильника;
- WR – вихід стробуючого сигналу при запису до внутрішньої пам'яті даних;
- RD – вихід стробуючого сигналу при зчитуванні із зовнішньої пам'яті даних;
- PME – дозвіл програмної пам'яті;
- ALE – вихід сигналу стробу адреси.

8.2. Структурна організація МК51

8.2.1. Структурна схема МК51

Основу структурної схеми МК51 (рис. 8.2) утворює внутрішня двоспрямована 8-бітова шина, що пов'язує всі основні вузли і пристрої: резидентну пам'ять, АЛП, блок регістрів спеціальних функцій, обладнання управління та порти вводу–виводу.

Джерело живлення підключається до виводів V_{ss} (20) "земля" і V_{cc} (40) +5 В. Для управління роботою всіх пристроїв в МК використовується генератор імпульсів, який може працювати від зовнішнього джерела або автономно (у режимі самозбудження). В останньому випадку до виводів XTAL2 (18) і XTAL1 (19) підключається кварцовий резонатор на частоту не більш 12 МГц. Після увімкнення живлення необхідно встановити внутрішні пристрої МК у початковий стан подачею імпульсу високого рівня на вивід RST (9). МК починає роботу з виконання команди, записаної за нульовою адресою.

Паралельні порти працюють у байтовому форматі, тобто мають по 8 виводів: P0 (32 – 39), P1 (1 – 8), P2 (21 – 28) і P3 (10 – 17).

Порти P0 і P2 використовуються також для видачі адреси при зверненні до зовнішніх пристроїв.

Виводи, до яких приєднані шини порту 3, можуть використовуватись:

- для послідовного порту як приймальна RxD (10) або передавальна TxD (11) лінії;
- для введення сигналів зовнішніх переривань INT0# (12) і INT1# (13);
- для підрахунку зовнішніх імпульсів T0 (14) і T1 (15);

➢ для видачі сигналів запису WR# (16) та зчитування RD# (17) на зовнішні пристрої, що запам'ятовують.

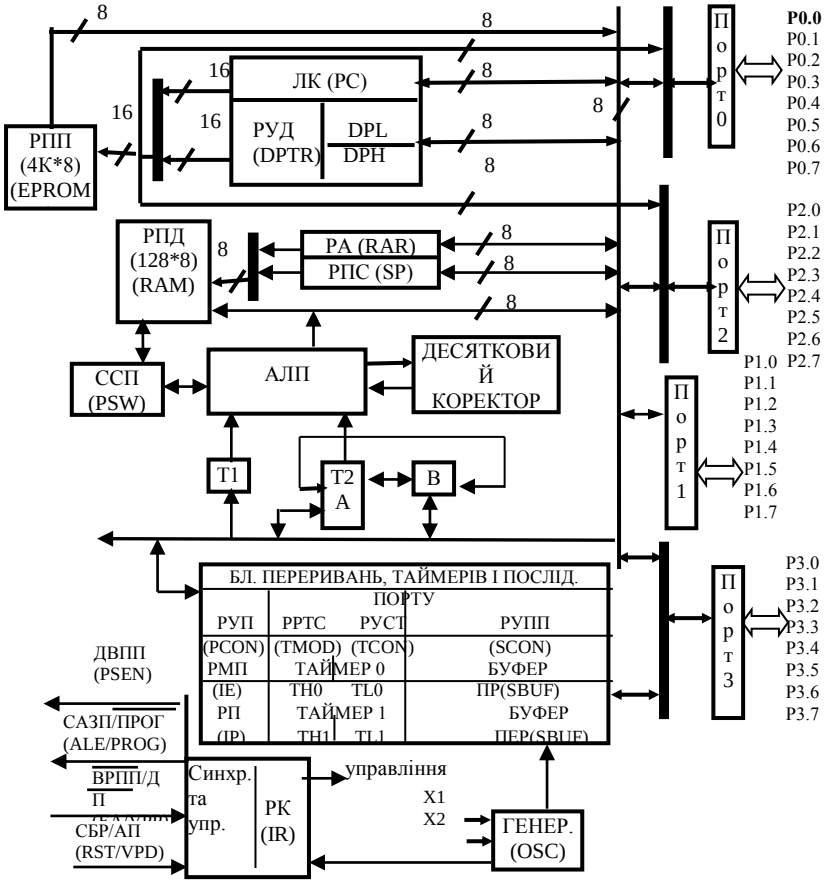


Рис. 8.2. Структурна схема МК51

Крім того, для роботи із зовнішніми ЗП використовуються виводи PSEN# (29), ALE#(30) та EA#(31).

Програма і константи записуються в ПЗП (постійний ЗП, або постійна пам'ять), що англійською мовою – ROM (Read Only Memory).

Для зберігання змінних даних використовується ОЗП (оперативний ЗП, або оперативна пам'ять), що англійською мовою – RAM

(Random Access Memory).

Для спілкування із зовнішнім ЗП МК51 має два **16-розрядні регістри**: PC (Program Counter) і DPTR (Data PoinTeR). Перший з них (програмний лічильник) використовується тільки для зчитування команд з ПЗП, а другий (покажчик даних) – для зчитування даних з ПЗП і ОЗП і для запису в ОЗП. Таким чином МК51 може використовувати адресний простір до 65 536 байтів.

Внутрішній ОЗП МК51 має ємність всього 128 байтів. Але адресний простір у нього 8-розрядний, тобто 256 байтів. Старші адреси призначені для звернення до функціональних регістрів МК51.

В тім, у найдосконаліших моделей цієї сім'ї ємність внутрішнього ОЗП дорівнює 256 байтам, але старші адреси доступні тільки при непрямій адресації. Початкові елементи оперативної пам'яті (32 байти) використовуються під однобайтові регістри загального призначення: R0, Rl, R2, R3, R4, R5, R6 і R7. Їх фізичні адреси залежать від вмісту 3-го і 4-го розрядів регістра PSW (Processor Status Word – однобайтове слово стану процесора).

Розподіл адрес для цих груп комірок, називаних банками 0, 1, 2 і 3, наведений в табл. 8.1.

Таблиця 8.1

RSI	**RS0**	**R0**	**Rl**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	
0	0	00h	01h	02h	03h	04h	05h	06h	07h	**Банк 0**
0	1	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh	**Банк 1**
1	0	10h	11h	12h	13h	14h	15h	16h	17h	**Банк 2**
1	1	18h	19h	1Ah	1Bh	1Ch	1Dh	1Eh	1Fh	**Банк 3**

Адреси від 20h до 7Fh програміст може використовувати на власний розсуд. Частина адресного простору від 80h до 0FFh зайнята для звернення до різних пристроїв усередині МК, яке здійснюється програмуванням регістрів спеціальних функцій (SFR - Special Function Register).

Розглянемо ці функціональні регістри. Для кожного з них наведені мнемонічні імена й адреси. Більшість функціональних регістрів МК 8-розрядні (табл. 8.2).

Для виконання арифметичних і логічних дій використовується так званий накопичувач А або ACC (Accumulator). Для операцій множення і ділення використовується додатковий регістр В. Залежно від виконуваної команди відповідно до результату операції можуть вироблятися ознаки, які записуються в регістр **PSW**.

Таблиця 8.2

Символ	Найменування	Адреса
* A (ACC)	Акумулятор	0E0H
* B	Регістр-розширювач акумулятора	0F0H
* PSW	Слово стану програми	0D0H
SP	Регістр-покажчик стека	81H
DPTR	Регістр-покажчик даних (DPH)	83H
	(DPL)	82H
* P0	Порт 0	80H
* P1	Порт 1	90H
* P2	Порт 2	0A0H
* P3	Порт 3	0B0H
* IP	Регістр пріоритетів	0B8H
* IE	Регістр маски переривань	0A8H
TM0D	Регістр режиму таймера/лічильника	89H
* TC0N	Регістр керування/статусу таймера	88H
TH0	Таймер 0 (старший байт)	8CH
TL0	Таймер 0 (молодший байт)	8AH
TH1	Таймер 1 (старший байт)	8DH
TL1	Таймер 1 (молодший байт)	8BH
* SCON	Регістр керування приймачем-передавачем	98H
SBUF	Буфер приймача-передавача	99H
PCON	Регістр керування потужністю	87H

Знак "*" показує, що даний регістр допускає адресацію окремих бітів.

Програмний лічильник ніяк не відображається на адресному просторі ОЗП, щоб випадково не спотворити його вміст. Покажчик даних складається з двох байтів: старшого DPH і молодшого DPL (H і L – від слів High і Low відповідно). Звернення до оперативної пам'яті може здійснюватися в стековому режимі за допомогою регістра SP (Stack Pointer).

Мнемонічні імена регістрів ***паралельних портів*** збігаються з позначеннями їх виводів: P0, P1, P2 і P3.

Послідовний порт здійснює передачу або прийом через регістр SBUF (Serial BUFfer) (99h), а управління режимом його роботи здійснюється за допомогою регістра SCON (Serial CONtrol) (98h).

У МК51 є два лічильники з номерами 0 і 1, які можуть використо-

вуватись як для підрахунку тактових імпульсів (режим таймера), так і для рахунку імпульсів на зовнішніх входах. Кожен із них складається з двох байтів: старшого TH0, TH1 і молодшого TL0, TL1. Для керування ними використовується регістр керування TCON (Timer CONtrol). Щоб забезпечити роботу МК із зовнішніми об'єктами в реальному масштабі часу є ***система переривання***, для управління якою використовуються регістр дозволу переривань IE (Interrupt Enable) і регістр пріоритетів переривань IP (Interrupt Priority).

У МК є можливість обміну не тільки у форматі байта, але і окремими бітами. Частина бітів функціональних регістрів має мнемонічні імена. Але можна звертатися до бітів і на ім'я регістра і номеру біта в байті, використовуючи як розділювач між ім'ям і номером точку. Наймолодший біт має номер 0, а найстарший – 7.

Наведемо перелік початкових станів всіх функціональних регістрів МК на момент його ініціалізації:

ACC, B, PSW, DPH, DPL	00h
SP	07h
P0, P1, P2, P3	0FFh
SBUF	xxxxxxxxb
SCON, TH0, TL0, TH1, TL1, TCON	00h
IE	0x000000b
IP	xx000000b

У переліку буквою *x* позначений довільний вміст, а символи *b* і *h* відповідають шістнадцятковому і двійковому кодам.

Стан комірок ОЗП після скидання носить довільний характер.

Слід відзначити, що при програмуванні на Асемблері інформація про адресний простір ОЗП потрібна програмісту тільки для оцінювання доступних ресурсів пам'яті. Спроби звернення до ОЗП з неіснуючими адресами (вищий 7Fh) можуть призвести до непередбачуваних результатів. В тім при програмуванні на Асемблері такі випадковості виключені, хоча можна зробити це і навмисно.

Розглянемо ***цикл виконання команди***. Залежно від складності виконуваних дій цикл виконання команди займає від одного до чотирьох тактів, тривалість кожного з яких дорівнює дванадцяти періодам коливань кварцового резонатора. Якщо використовується кварцовий резонатор на 12 МГц, то тривалість такту дорівнює 1 мкс. Цикл виконання чергової команди починається зі зчитування її коду з ПЗП за адресою, записаною в регістр PC (програмний лічильник). Після зчитування кожного з байтів команди вміст цього регістра збільшується на 1,

таким чином після зчитування останнього байта команди програмний лічильник містить адресу першого байта наступної команди. Потім здійснюються зчитування операндів, обробка одержаної інформації і запам'ятовування результату операції. Залежно від коду операції команди ті чи інші дії можуть не виконуватися. У системі команд МК є такі, які можуть змінити вміст програмного лічильника. Ці команди називаються керуючими. Одні з них здійснюють це залежно від результатів поточної або попередніх операцій, інші – незалежно від виконання будь-яких умов. За рахунок використання керуючих команд при виконанні програми мікроконтролер не тільки здійснює обчислення, але й виконує логічні дії, наприклад, вибір варіанта або циклічного повторення одних і тих же дій.

Виконання наступної команди починається тільки після завершення поточної, тобто одночасне виконання хоча б двох команд неможливе. Проте під час виконання програми окремі пристрої МК можуть працювати автономно.

Арифметично-логічний пристрій

Арифметично-логічний пристрій, розрахований на 8 біт інформації може виконувати арифметичні операції додавання і віднімання, множення і ділення; логічні операції І, АБО, виключне АБО, а також операції циклічного зсуву, зсуву, скидання тощо. В АЛП є регістри Т1 і Т2, програмно недоступні, та призначені для тимчасового зберігання операндів, схема десяткової корекції і схема формування ознак.

Найпростіша операція додавання використовується в АЛП для інкрементування вмісту регістрів, просування регістра-покажчика даних і автоматичного обчислення наступної адреси РПП. Найпростіша операція віднімання використовується в АЛП для декрементування регістрів і порівняння змінних. Найпростіші операції автоматично утворюють "тандеми" для виконання в АЛП таких операцій, як, наприклад, інкрементування 16-бітних регістрових пар. В АЛП реалізується механізм каскадного виконання найпростіших операцій для реалізації складних команд.

Так, наприклад, при виконанні однієї з команд умовної передачі керування за результатом порівняння в АЛП тричі інкрементується ЛК, двічі здійснюється зчитування з РПД, виконується арифметичне порівняння двох змінних, формується 16-бітова адреса переходу і прийма-

ється рішення про те, чи виконувати перехід за програмою. Всі перераховані операції виконуються в АЛП всього лише за 2 мкс.

Важливою особливістю АЛП є його здатність оперувати не тільки байтами, але і бітами.

Окремо біти, які є програмно доступними, можуть бути встановлені, скинуті, інвертовані, передані, перевірені та використані в логічних операціях. Ця здатність АЛП оперувати бітами настільки важлива, що в багатьох описах МК51 говориться про наявність в ньому булевського процесора. Для управління об'єктами часто застосовуються алгоритми, що містять операції над вхідними і вихідними булевськими змінними (істина/неправда), реалізація яких засобами звичайних мікропроцесорів пов'язана з певними труднощами.

Таким чином, АЛП може оперувати чотирма типами інформаційних об'єктів: булевськими (1 біт), цифровими (4 біти), байтовими (8 біт) і адресними (16 біт). В АЛП виконується 51 різноманітна операція пересилання або перетворення цих даних. Оскільки використовується 11 режимів адресації (7 для даних і 4 для адрес), то шляхом комбінування "операція/ режим адресації" базова адресація команд із 111 зростає до 255 з 256 можливих при однобайтовому коді операції.

Резидентна пам'ять

Пам'ять програм і пам'ять даних, розміщених на кристалі МК51, фізично і логічно поділені, мають різноманітні механізми адресації, працюють під управлінням різноманітних сигналів і виконують різноманітні функції.

Пам'ять програм (ПЗП або ЗППЗП) має ємність 4 кбайти і призначена для зберігання команд, констант, керуючих слів ініціалізації, таблиць перекодування вхідних та вихідних змінних тощо. РПП має 16-бітову шину адреси, через яку забезпечується доступ з лічильника команд або з регістра-покажчика даних. Останній виконує функції базового регістра при непрямих переходах за програмою або використовується в командах, що оперують з таблицями.

Пам'ять даних (ОЗП) перевизначена для зберігання змінних у процесі виконання прикладної програми, адресується одним байтом і має ємність 128 байт. Крім того, до адресного простору РПД приєднуються адреси регістрів спеціальних функцій (РСФ), зазначених в табл. 2.4. Пам'ять програм, так само як і пам'ять даних, може бути поширена до 64 кбайт шляхом підключення зовнішніх ВІС.

Акумулятор і ССП. Акумулятор є джерелом операнда і місцем фіксації результату при виконанні арифметичних, логічних операцій і ряду операцій передачі даних. Крім того, тільки з використанням акумулятора можуть бути виконані операції зсуву, перевірка на "0", формування прапорця паритету тощо. При виконанні багатьох команд в АЛП формується ряд ознак операції (прапорців), що фіксуються в регістрі ССП. У табл. 8.3 наводиться перелік прапорців ССП, подаються їхні символічні імена і описуються умови їхнього формування.

Таблиця 8.3

Формат слова стану програми (ССП)

Символ	Позиція	Адреса	Ім'я і призначення
C	PSW.7	0D7h	Прапорець переносу. Установка/скидання при виконанні арифметичних та логічних операцій
AC	PSW.6	0D6h	Прапорець додаткового переносу. Установка тільки апаратно. Сигналізує про перенос чи позику в біті С
F0	PSW.5	0D5h	Прапорець 0. Програмна установка/скидання користувачем
RS1 RS0	PSW.4 PSW3.	0D4h 0D3h	Вибір банка регістрів. Програмна установка/скидання (див. примітку)
OV	PSW.2	0D2h	Прапорець переповнення. Установка/скидання при виконанні арифметичних операцій
_	PSW.1	0Dlh	Не використовуються
P	PSW.0	0D0h	Прапорець паритету. Установка/скидання апаратно в кожному циклі команди. Контроль за парністю

Примітка. Вибір робочого банку регістрів

RS1	RS0	Банк	Межі адрес
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH

Найбільш "активним" прапорцем ССП є прапорець переносу, що бере участь і модифікується в процесі виконання безлічі операцій, включаючи додавання, віднімання і зсув.

Крім того, прапорець переносу (С) виконує функції "булевого акумулятора" в командах, що маніпулюють з бітами. Прапорець пере-

повнення (OV) фіксує арифметичне переповнення при операціях над цілими числами зі знаком і робить можливим використання арифметики в додаткових кодах. АЛП не управляє прапорцями селекції банка регістрів (RS0, RS1), і їх значення повністю визначається прикладною програмою та використовується для вибору одного з чотирьох регістрових банків.

Широке розповсюдження отримало уявлення про те, що в МП, архітектура яких спирається на акумулятор, більшість команд працюють з ним, використовуючи адресацію "за замовчуванням" (неявну). У МК51 – по-іншому. Хоча процесор в МК51 має в своїй основі акумулятор, однак він може виконувати безліч команд і без участі акумулятора. Наприклад, дані можуть бути передані з будь-якої комірки РПД в будь-який регістр, може бути завантажений безпосереднім операндом тощо.

Безліч логічних операцій можуть виконуватись без участі акумулятора. Крім того, змінні можуть бути інкрементовані, декрементовані та перевірені (test) без використання акумулятора. Прапорці і керуючі біти можуть бути перевірені та змінені аналогічно.

Регістри-покажчики. 8-бітовий покажчик стека (SP або РПС) може адресувати будь-яку область РПД. Його вміст інкрементується раніше, ніж дані можуть бути запам'ятовані стеком в ході виконання команд PUSH та CALL. Вміст РПС декрементується після виконання команд POP та RET. Подібний спосіб адресації елементів стека називають передікрементним/ постдекрементним. У процесі ініціалізації МК51 після сигналу СКД в РПС автоматично завантажується код 07h. Це означає, що якщо прикладна програма не перевизначає стек, то перший елемент даних буде розташовуватися в комірці РПД з адресою 08h. Двобайтовий регістр-покажчик даних (РПоД) використовується для фіксації 16-бітової адреси в операціях зі зверненням до зовнішньої пам'яті. Командами МК51 регістр-покажчик даних може бути використаний або як 16-бітовий регістр, або як два незалежних 8-бітових регістри (DPH, DPL).

Таймер/лічильник. До складу засобів МК51 входять регістрові пари з символічними іменами TH0, TL0, TH1, TL1, на основі яких функціонують два незалежних 16-бітових таймери/лічильники подій, що програмно управляються.

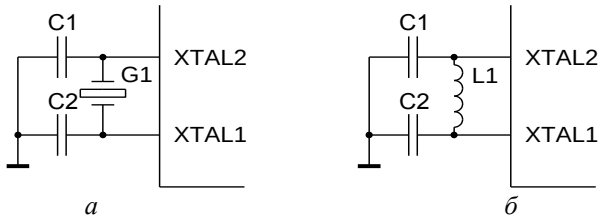
Буфер послідовного порту. Регістр з символічним ім'ям SBUF являє собою два незалежних регістри – буфер приймача і буфер передавача. Завантаження байта в SBUF негайно викликає початок процесу

передачі через послідовний порт. Якщо байт зчитується з SBUF, це означає, що його джерелом є приймач послідовного порту.

Регістри спеціальних функцій. Регістри з символічними іменами IP, IE, TMOD, TCON, SCON і PCON використовуються для фіксації і програмної зміни керуючих бітів і бітів стану схеми переривання, таймера/лічильника, приймопередавача послідовного порту і для управління потужністю електроживлення МК51.

Пристрій управління і синхронізації

Кварцовий резонатор, що підключається до зовнішніх виводів X1 і X2 корпусу МК51 (рис. 8.3, *а*), управляє роботою внутрішнього генератора, який в свою чергу формує сигнали синхронізації. Конденсатори C1, C2 мають ємність 30 пФ ± 10 пФ.



а *б*

Рис. 8.3. Підключення пристроїв синхронізації

Замість кварцового резонатора можна використовувати LC-контур (рис. 8.3, *б*). Ємність конденсаторів C1 та C2 дорівнює C. Частота синхронізації визначається за формулою

$$F = 1/(2\pi(LC')^{1/2}),$$

де $C' = (C + 3C_{pp})/2$, $C_{pp} \approx 10$пФ – ємність виводу.

Стабільність частоти у цьому випадку зменшується.

У багатопроцесорних системах, та системах, що вимагають загальної системи синхронізації, МК може синхронізуватись від зовнішнього джерела синхронізуючих імпульсів. Схема підключення зовнішнього джерела залежить від типу МК. Для вітчизняних МК К1816ВЕ51 підключення відбувається за схемою, наведеною на рис. 8.4, *а*.

Для МК К1830ВЕ51 підключення відбувається за схемою, наведеною на рис. 8.4, *б*.

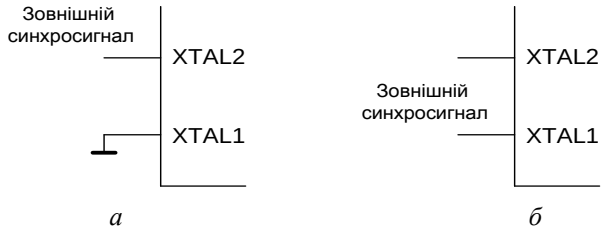


Рис. 8.4. Підключення зовнішньої синхронізації

Пристрій управління МК51 на основі сигналів синхронізації формує машинний цикл фіксованої тривалості, що дорівнює 12 періодам резонатора або шести станам первинного керуючого автомата (S1 – S6), як показано на рис. 8.5.

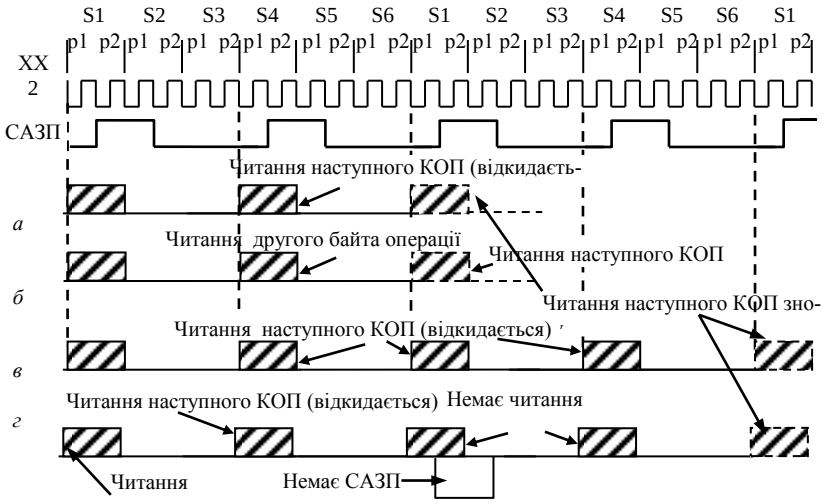


Рис. 8.5. Послідовність вибірки та виконання команд в МК51:
а – команда 1 байт/1 цикл, наприклад INC A;
б – команда 2 байти/1 цикл, наприклад ADD A, #d;
в – команда 1 байт/2 цикли, наприклад INC DPTR;
г – команда 1 байт/2 цикли, наприклад MOVX

Кожний стан керуючого автомата містить 2 фази (P1, P2) сигналів резонатора. У фазі P1, як правило, виконується операція в АЛП, а у

фазі Р2 здійснюється регістрова передача. Весь машинний цикл складається з 12 фаз, починаючи з фази S1P1 і закінчуючи фазою S6P2.

Ця часова діаграма ілюструє роботу МК51 при виборці і виконанні команд різноманітного ступеня складності. Всі заштриховані сигнали є внутрішніми і недосяжними користувачу МК51 для контролю.

Зовнішніми сигналами, що спостерігаються, є тільки сигнали резонатора і стробу адреси зовнішньої пам'яті. Як видно з часової діаграми, сигнал **САЗП** формується двічі за один машинний цикл (S1P2-S2P1; S4P2-S5P1) і використовується для управління процесом звернення до зовнішньої пам'яті.

Більшість команд МК51 виконується за один машинний цикл. Деякі команди, що оперують з 2-байтовими словами або пов'язані зі звертанням до зовнішньої пам'яті, виконуються за 2 машинних цикли. Тільки команди ділення і множення вимагають чотири машинні цикли. На основі цих особливостей роботи пристрою управління МК51 здійснюється розрахунок часу виконання прикладних програм.

8.2.2. Порти вводу–виводу інформації

Всі чотири порти МК51 призначені для введення або виведення інформації.

Кожен з портів містить фіксатор, який є восьмирозрядним регістром, і має байтову і бітову адресацію для установки (скидання) розрядів за допомогою програмного забезпечення.

Фізичні адреси фіксаторів Р0, Р1, Р2, Р3 складають для:

Р0 – 80Н, при бітовій адресації 80Н – 87Н;

Р1 – 90Н, при бітовій адресації 90Н – 97Н;

Р2 – А0Н, при бітовій адресації А0Н – А7Н;

Р3 – В0Н, при бітовій адресації В0Н – В7Н.

Спрощені функціональні схеми портів наведено на рис. 8.6 – 8.8.

Кожен порт містить керований регістр-фіксатор, вхідний буфер і вихідний драйвер. Схемотехніка портів 1 та 2 вводу–виводу МК51 для одного біта має приблизно таку ж структуру, як і порт 3. Вихідні драйвери портів 0 і 2, а також вхідний буфер порту 0 використовуються у разі звертання до зовнішньої пам'яті (ЗП). При цьому через порт 0 у режимі часової мультиплексації спочатку виводиться молодший байт адреси ЗП, а після цього видається або приймається байт даних.

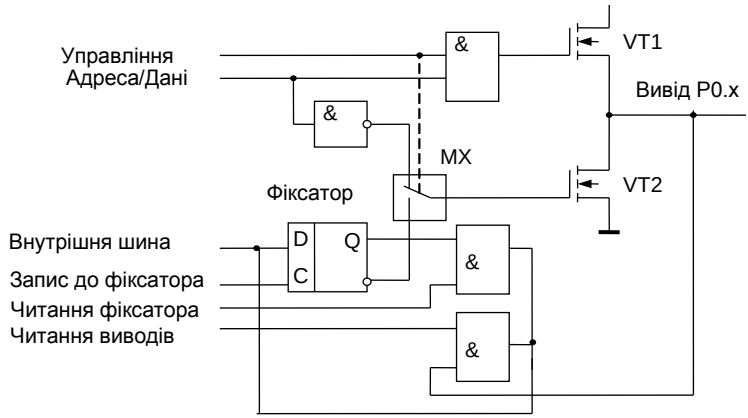


Рис. 8.6. Схема порту P0

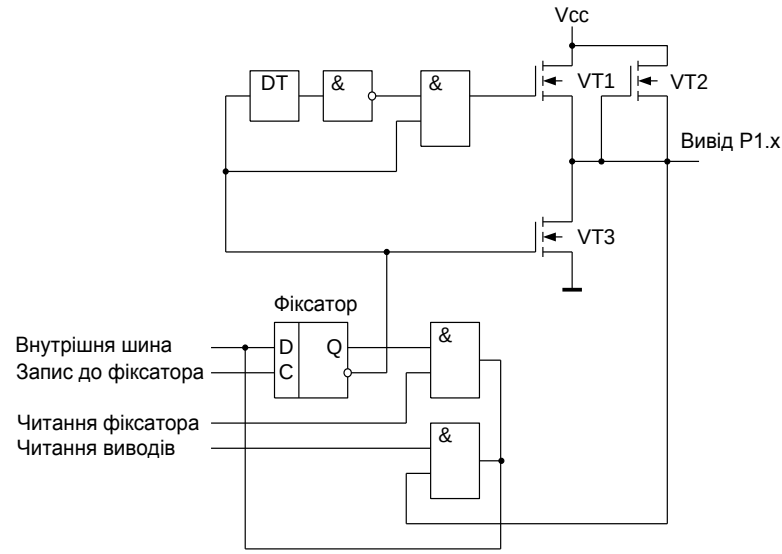


Рис. 8.7. Схема порту P1

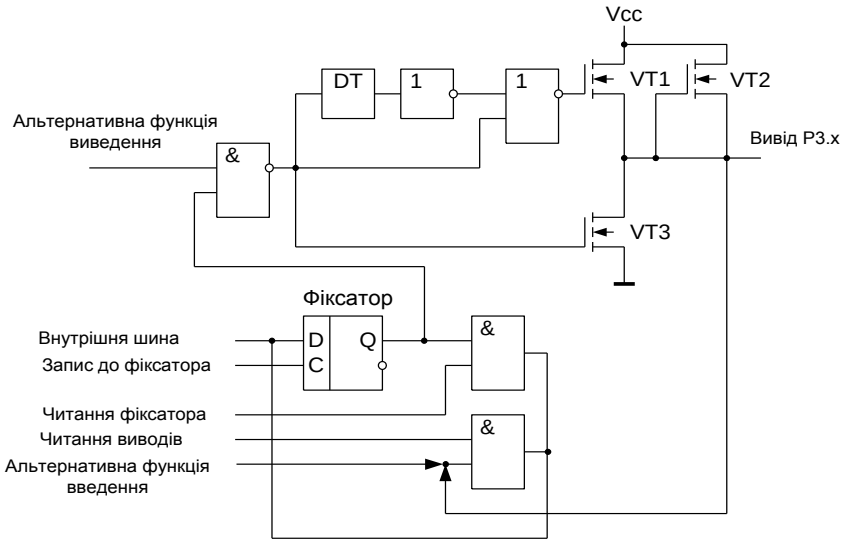


Рис. 8.8. Схема порту P3

Через порт 2 виводиться старший байт адреси в тих випадках, коли розрядність адреси дорівнює 16 бітам. Всі виводи порту 3 можуть бути використані для реалізації альтернативних функцій, поданих в табл. 8.4.

Альтернативні функції можуть бути задіяні шляхом запису 1 у відповідні біти регістра-фіксатора (P3.0 – P3.7) порту 3.

Порт 0 є двонаправленим, а порти 1, 2 і 3 – квазідвонаправленими. Кожна лінія порту може бути використана незалежно для введення або виведення інформації.

Для того, щоб деяка лінія порту використовувалася для введення інформації, в D-тригер регістра-фіксатора порту повинна бути записана 1, що закриває МОН-транзистор вихідного кола.

За сигналом СКД в регістри-фіксатори всіх портів автоматично записується 1, що настроює їх завдяки цьому на режим введення.

Всі порти можуть бути використані для організації введення–виведення інформації по двонаправлених лініях передачі.

Таблиця 8.4

Альтернативні функції порту 3

Символ	Позиція	Ім'я і призначення
RD	P3.7	Зчитування. Активний сигнал низького рівня формується апаратно при звертанні до ВПД
WR	P3.6	Запис. Активний сигнал низького рівня формується апаратно при звертанні до ВПД
T1	P3.5	Вхід таймера/лічильника 1 або тест-вхід
T0	P3.4	Вхід таймера/лічильника 0 або тест-вхід
INT1	P3.3	Вхід запиту переривання 1. Сприймається сигнал низького рівня або зріз
INT0	P3.2	Вхід запиту переривання 0. Сприймається сигнал низького рівня або зріз
TXD	P3.1	Вихід передавача послідовного порту в режимі УАПП. Вихід синхронізації в режимі зсуву Рг
RXD	P3.0	Вхід приймача послідовного порту в режимі УАПП. Введення-виведення даних у режимі зсуву Рг

Однак порти 0 і 2 не можуть використовуватись, якщо МК-система має зовнішню пам'ять, зв'язок з якою організується через загальну шину, що поділяє адреси/дані, та працює в режимі часового мультиплексування.

Однак порти 0 і 2 не можуть використовуватись з цією метою у випадку, якщо МК-система має зовнішню пам'ять, зв'язок з якою організується через загальну шину, що поділяє адреси/дані, та працює в режимі часового мультиплексування.

Запис в порт. При виконанні команди, що змінює вміст регістра-фіксатора порту, нове значення фіксується в регістрі у момент S6P2 останнього циклу команди. Однак опитування вмісту регістра-фіксатора вихідною схемою здійснюється під час фази P1 і, отже, новий вміст регістра-фіксатора з'являється на вихідних контактах порту тільки в момент S1P1 наступного машинного циклу.

Навантажувана здатність портів. Вихідні лінії портів 1, 2 і 3 можуть працювати на одну ТТЛ-схему. Лінії порту 0 можуть бути навантажені на два входи ТТЛ-схем кожна. Лінії порту 0 можуть працювати і на *n*-МОН-схемі, однак при цьому їх необхідно підключати до джерела електроживлення через зовнішні навантажувальні резистори за винятком випадку, коли шина порту 0 використовується як шина адреси/даних зовнішньої пам'яті.

Вхідні сигнали для МК51 можуть формуватися ТТЛ-схемами або *n*-МОН-схемою. Допустимо використовувати як джерело сигналів для МК51 схеми з відкритим колектором або відкритим стоком. Однак при цьому час зміни вхідного сигналу при переході з 0 в 1 виявиться надто довгим.

Навантажувальна здатність портів різних мікроконтролерів сімейства 8051 може сильно відрізнятися від наведеної.

Особливості роботи портів. Звернення до портів введення–виведення можливе з використанням команд, що оперують з байтом, окремим бітом і довільною комбінацією бітів. При цьому в тих випадках, коли порт є водночас операндом і місцем призначення результату, пристрій управління автоматично реалізує спеціальний режим, що називається "зчитування-модифікація-запис". Цей режим звернення припускає введення сигналів не з зовнішніх виводів порту, а з його регістра-фіксатора, що дозволяє виключити невірне зчитування раніше виведеної інформації.

Подібний механізм звернення до портів реалізований в таких командах:

ANL – логічне І, наприклад ANL P1, A;

ORL –логічне АБО, наприклад ORL P2, A;

XRL – виключне АБО, наприклад XRL P3, A;

JBC – перехід, якщо в біті одиниця і наступне скидання біта, наприклад JBC P1.1, LABEL;

CPL – інверсія біта, наприклад CPL P3. 3;

INC – інкремент порту, наприклад INC P2;

DEC – декремент порту, наприклад DEC P2;

DJNZ – декремент порту і перехід, якщо його вміст не дорівнює нулю, наприклад DJNZ P3, LABEL;

MOV PX.Y , C – передача біта переносу в біт Y порту X;

SET PX.Y – встановлення біта Y порту X;

CLR PX.Y – скинення біта Y порту X.

Останні три команди з наведеного списку є командами "зчитування-модифікація-запис". За цими командами спочатку зчитується байт з порту, а після цього записується новий байт у регістр-фіксатор. Причиною, з якої команди "зчитування-модифікація-запис" забезпечують розподілений доступ до регістра-фіксатора порту і до зовнішніх виводів порту, є необхідність виключити можливість невірного читання рівнів сигналів на зовнішніх виводах. Припустимо для прикладу, що лінія Y порту X з'єднується з базою потужного транзистора і вихідний

сигнал на ній визначений для його управління. Коли в даний біт записана 1, то транзистор вмикається. Якщо для перевірки стану виконавчого механізму (в нашому випадку потужного транзистора) прикладній програмі потрібно зчитати стан вихідного сигналу в тому ж біті порту, то зчитування сигналу із зовнішнього виводу порту, а не з D-тригера регістра-фіксатора порту призведе до неправильного результату: одиничний сигнал на базі транзистора має відносно низький рівень і буде інтерпретований МК як сигнал 0. Команди "зчитування–модифікація–запис" реалізують зчитування з регістра-фіксатора, а не із зовнішнього виводу порту, що забезпечує отримання необхідного значення 1.

8.2.3. Доступ до зовнішньої пам'яті

У МКС, побудованих на основі МК51, можливе використання двох типів зовнішньої пам'яті: постійної пам'яті програм (ЗПП) і оперативної пам'яті даних (ЗПД). Ємність кожного типу пам'яті складає 64 кбайти.

Доступ до ЗПП здійснюється за допомогою керуючого сигналу **ДВПП**, що виконує функцію строб-сигналу зчитування. Доступ до ЗПД забезпечується керуючими сигналами **ЧТ** і **ЗП**, що формуються на лініях P3.7 і P3.6 при виконанні портом 3 альтернативних функцій.

У разі звертання до ЗПП завжди використовується 16-бітова адреса. Доступ до ЗПД можливий з використанням:

– 16-бітової адреси (**MOVX A, @DPTR**);

– 8-бітової адреси (**MOVX A, @Ri**).

У будь-яких випадках використання 16-бітової адреси старший байт адреси фіксується (і зберігається незмінним протягом одного циклу запису або читання) в регістрі-фіксаторі порту 2.

Якщо черговий цикл зовнішньої пам'яті (MOVX A, @DPTR) відбувається не відразу ж за попереднім циклом зовнішньої пам'яті, то вміст, що не змінюється, регістра-фіксатора порту 2 відновлюється в наступному циклі. Якщо використовується 8-бітова адреса (MOVX A, @Ri), то вміст регістра-фіксатора порту 2 залишається незмінним на його зовнішніх виводах протягом всього циклу зовнішньої пам'яті.

Через порт 0 у режимі часового мультиплексування здійснюється видача молодшого байта адреси і передача байта даних. Сигнал **САЗП** повинен бути використаний для запису байта адреси в зовнішній регістр. Після цього в циклі запису байтів даних, що виводиться, з'являється на зовнішніх виводах порту 0 тільки перед появою сигналу

ЗП. У циклі читання байтів даних, що виводяться, приймається в порт 0 по фронту сигналу **ЧТ,** що стробує.

При будь-якому зверненні до зовнішньої пам'яті пристрій управління МК51 завантажує в регістр-фіксатор порту 0 код 0FFH, стираючи завдяки цьому інформацію, яка в ньому зберігалась. На рис. 8.9 наведено цикл читання зовнішньої пам'яті даних.

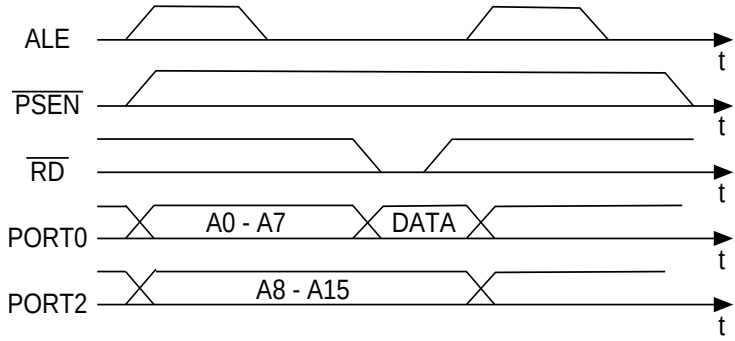


Рис. 8.9. Цикл зчитування зовнішньої пам'яті даних

На рис. 8.10 наведений цикл запису зовнішньої пам'яті даних.

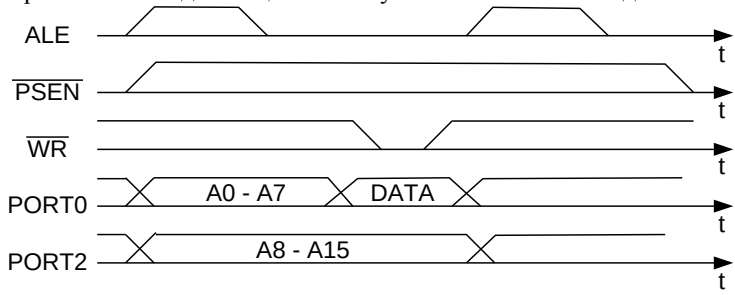


Рис. 8.10. Цикл запису зовнішньої пам'яті даних

Доступ до ЗПП можливий при виконанні двох умов: або на вхід відключення резидентної пам'яті програм **ВРПП** подається активний сигнал, або вміст лічильника команд перевищує значення 0FFFH. Наявність сигналу **ВРПП** необхідна для забезпечення доступу до молодших 4 кбайтів адрес адресного простору ЗПП при використанні МК31 (МК без резидентної пам'яті програм).

Доступ до ЗПП можливий з використанням команд **MOVC A, @A+DPTR**, **MOVC A, @A+PC**.

На рис. 8.11 наведено цикл зчитування зовнішньої пам'яті програм, а на рис. 8.12 – карта пам'яті мікроконтролера МК51.

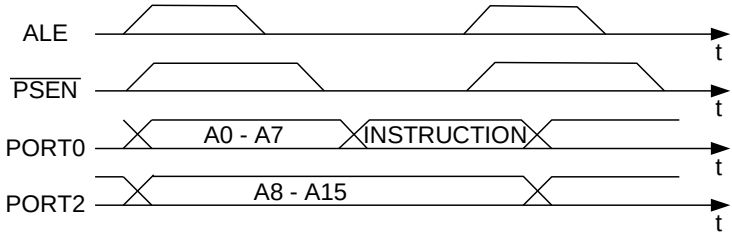


Рис. 8.11. Цикл читання зовнішньої пам'яті програм

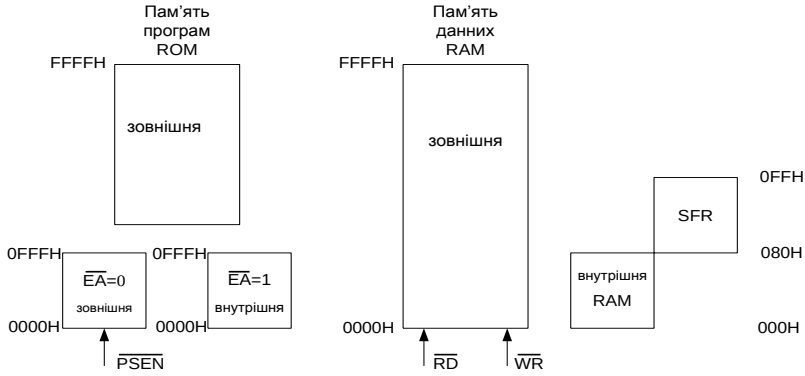


Рис. 8.12. Карта пам'яті мікроконтролера МК51

Особливий режим роботи МК51. Вміст пам'яті програм МК51 заповнюється один раз на етапі розробки МКС і не може бути модифікований у завершеному виробі. Оперативна пам'ять даних не може бути використана для зберігання кодів програми, бо в МК вибірка команд здійснюється тільки з області адрес пам'яті програм.

Ця особливість архітектури МК51 пояснюється тим, що в більшості застосувань МК вимагається наявність однієї незмінної прикладної програми, яка зберігається в ПЗП, наявність ОЗП невеликої ємності для

тимчасового зберігання змінних і ефективних, а отже, різних способів адресації пам'яті програм і пам'яті даних.

Однак на етапі розробки і відладки прикладних програм машина "фон-нейманівського" типу виявляється дуже зручною, бо дозволяє розробнику оперативно змінювати коди прикладної програми, що міститься в ОЗП. З цією метою МКС може бути модифікована для сумі-щеного адресного простору ЗПП і ЗПД шляхом підключення елемента 2І до сигналів ЧТ та ДВПП.

Подібний спосіб організації управління зовнішньою пам'яттю може бути використаний у тих застосуваннях МК51, де вимагається оперативне перевантаження або модифікація прикладних програм, як в ЕОМ класичної архітектури.

8.2.4. Таймер/лічильник

Два 16-бітових таймери/лічильники, що програмуються, (Т/С0 і Т/С1) можуть бути використані як таймери або лічильники зовнішніх подій. Виконуючи функції таймера вміст Т/С інкрементується в кожному машинному циклі, тобто через кожні 12 періодів резонатора, а виконуючи функції лічильника вміст Т/С інкрементується під впливом переходу з 1 в 0 зовнішнього вхідного сигналу, що подається на відповідний (Т0, Т1) вивід МК51.

Опитування значення зовнішнього вхідного сигналу виконується на момент часу S5P2 кожного машинного циклу. Вміст лічильника буде збільшений на 1 в тому випадку, якщо в попередньому циклі був зчитаний вхідний сигнал високого рівня (1), а в наступному – сигнал низького рівня (0). Нове (інкрементоване) значення лічильника буде сформоване в момент S3P1 у циклі, наступному за тим, в якому був виявлений перехід сигналу з 1 в 0. Оскільки на розпізнання переходу вимагається 2 машинних цикли, то максимальна частота підрахунку вхідних сигналів дорівнює 1/24 частоти резонатора. На тривалість періоду вхідних сигналів обмежень згори немає. Для гарантованого зчитування вхідного сигналу він повинен утримувати значення 1 як мінімум протягом одного машинного циклу МК51.

Для управління режимами роботи Т/С і для організації взаємодії таймерів з системою переривання використовуються 2 регістри спеціальних функцій (РРТЛ і РУСТ), опис яких наведено у табл. 8.6 – 8.8.

Таблиця 8.6

Регістр режиму роботи таймера/лічильника

Символ	Позиція	Ім'я та призначення
GATE	TMOD.7 для T/C1 і TMOD.3 для T/C0	Управління блокуванням. Якщо біт встановлений, то таймер/ лічильник "x" дозволений до тих пір, доки на вході "INT x" високий рівень і біт управління "TR x" встановлений. Якщо біт скинутий, то T/C дозволяється, як тільки біт управління "TR x" встановлюється
C/T	TMOD.6 для T/C1 і TMOD.2 для T/C0	Біт вибору режиму таймера або лічильника подій. Якщо біт скинутий, то працює таймер від внутрішнього джерела сигналів синхронізації. Якщо біт встановлений, то працює лічильник зовнішніх сигналів на вході "Tx"
M1	TMOD.5 для T/C1 і TMOD.1 для T/C0	Режим роботи (див. примітку)
M0	TMOD.4 для T/C1 і TMOD.0 для T/C0	Режим роботи (див. примітку)

Таблиця 8.7

M1	M0	Режим роботи
0	0	"TLx" працює як 5-бітовий розподільник
0	1	16-бітовий таймер/лічильник. "THx" і "TLx" включені послідовно
1	0	8-бітовий таймер/лічильник, що автоматично перевантажується. "THx" зберігає значення, яке необхідно перевантажити в "TLx" кожного разу за переповненням
1	1	Таймер/лічильник 1 встановлюється. Таймер/лічильник 0: TL0 працює як 8-бітовий таймер/лічильник, і його режим визначається керуючими бітами таймера 0. TH0 працює як 8-бітовий таймер, і його режим визначається керуючими бітами таймера 1

Режим 0. Переведення будь-якого T/C в режим 0 робить його 8-бітовим лічильником, на вхід якого підключений 5-бітовий перетворювач частоти на 32. У цьому режимі регістр таймера має розрядність 13 бітів. При переході зі стану "всі одиниці" до стану "всі нулі" встановлюється прапорець переривання від таймера TF1.

Таблиця 8.8

Регістр управління/ статуса таймера

Символ	Позиція	Ім'я та призначення
TF1	TCON.7	Прапорець переповнення таймера 1. Встановлюється апаратно при переповненні таймера/лічильника. Скидається при обслуговуванні переривання апаратно
TR1	TCON.6	Біт керування таймера 1. Встановлюється/скидається програмою для пуску/зупинки
TF0	TCON.5	Прапорець переповнення таймера 1. Встановлюється апаратно. Скидається при обслуговуванні переривання
TR0	TCON.4	Біт керування таймера 1. Встановлюється/скидається програмою для пуску/зупинки таймера/ лічильника
IE1	TCON.3	Прапорець фронту переривання 1. Встановлюється апаратно, коли детектується зріз зовнішнього сигналу ЗПР1 (INT1). Скидається при обслуговуванні переривання
IT1	TCON.2	Біт керування типом переривання 1. Встановлюється/скидається програмно для специфікації запиту ЗПР1 (зріз/ низький рівень)
IE0	TCON.1	Прапорець фронту переривання 0. Встановлюється по зрізу сигналу ЗПР0. Скидається при обслуговуванні переривання
IT0	TCON.0	Біт управління типом переривання 1. Встановлюється/скидається програмно для специфікації запиту ЗПР0 (зріз/низький рівень)

Відзначимо попутно, що встановлення біта GATE в 1 дозволяє використати таймер для вимірювання тривалості імпульсного сигналу, що подається на вхід запиту переривання.

Режим 1. Робота будь-якого Т/С в режимі 1 така ж, як і в режимі 0, за винятком того, що таймерний регістр має розрядність 16 бітів.

На рис. 8.13 наведена схема таймера/лічильника 1 в режимі 1.

Режим 2. У режимі 2 робота організована таким чином, що переповнення (перехід зі стану "всі одиниці" до стану "всі нулі") 8-бітового лічильника TL1 приводить не тільки до встановлення прапорця TF1, але і автоматично перевантажує в TL1 вміст старшого байта (TH1) таймерного регістра, що попередньо було задано програмно. Перевантаження залишає вміст TH1 незмінним. У режимі 2 Т/С0 і Т/С1 працюють однаково.

На рис. 8.14 наведена схема таймера/лічильника 1 в режимі 2.

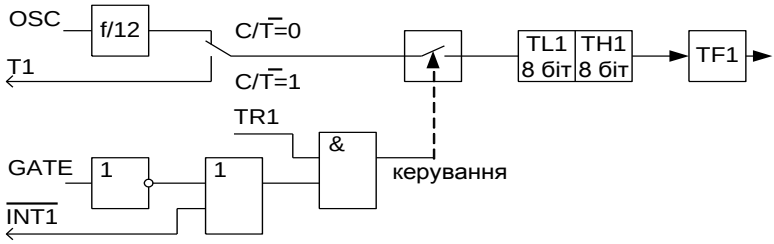


Рис. 8.13. Таймер/лічильник 1 в режимі 1

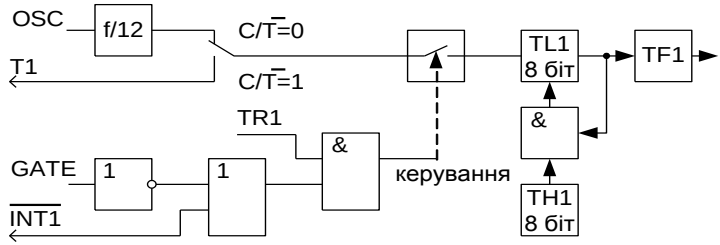


Рис. 8.14. Таймер/лічильник 1 в режимі 2

Режим 3. У режимі 3 T/C0 і T/C1 працюють по-різному. T/C1 зберігає незмінним свій поточний вміст. Іншими словами, ефект такий же, як і при скиданні керуючого біта TR1 в нульовий сигнал.

На рис. 8.15 наведена схема таймера/лічильника 0 в режимі 3: два 8-бітові лічильники.

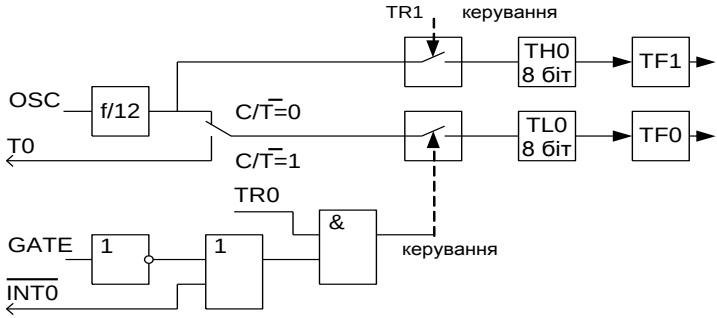


Рис. 8.15. Таймер/лічильник 0 у режимі 3: два 8-бітові лічильники

8.2.5. Послідовний інтерфейс

Через універсальний синхронний, асинхронний прийомопередавач (УАПП) здійснюється прийом та передача інформації, наведеної послідовним кодом (молодшими бітами спочтку), у повнодуплексному режимі обміну.

Функціональне призначення регістра УАПЦ наведено в табл. 8.9.

Таблиця 8.9

Регістр керування/статусу УАПП

Символ	Позиція	Ім'я та призначення
SM0	SCON.7	Біт керування режимом роботи УАПП. Встановлюється/скидається програмно (див. примітку)
SM1	SCON.6	Біт керування режимом роботи УАПП. Встановлюється програмно для заборони прийому повідомлення, в якому 9-й біт дорівнює 0
SM2	SCON.5	Біт дозволу прийому. Встановлюється/скидається програмно для дозволу/заборони прийому послідовних даних
REN	SCON.4	Біт дозволу прийому. Встановлюється/ скидається програмно для дозволу/заборони прийому послідовних даних
TB8	SCON.3	Передача біта 8. Встановлюється/ скидається програмно для задання 9-го біту, що передається, в режимі УАПП-9 біт
RB8	SCON.2	Передача біта 8. Встановлюється/ скидається апаратно для задання 9-го біту, що передається, в режимі УАПП-9 біт
TI	SCON.1	Прапорець переривання передавача. Встановлюється апаратно після закінчення передачі байта. Скидається програмно після обслуговування переривання
RI	SCON.0	Прапорець переривання передавача. Встановлюється апаратно при прийомі байта. Скидається програмно після обслуговування переривання

До складу УАПП, що часто називається послідовним портом, входять регістри зсуву, що приймають і передають, а також спеціальний буферний регістр (SBUF) прийомопередавача. Запис байта в буфер приводить до автоматичного перезапису байта в регістр зсуву передавача та ініціює початок передачі байта. Наявність буферного регістра

приймача дозволяє суміщати операцію зчитування раніше прийнятого байта з прийомом чергового байта. Якщо на час закінчення прийому байта попередній байт не був зчитаним з SBUF, то він буде втрачений.

Режими роботи УАПП наведені у табл. 8.10.

Таблиця 8.10

SM0	SM1	Режим роботи УАПП
0	0	Регістр зсуву, розширення введення–виведення
0	1	УАПП-8 біт. Швидкість передачі, що змінюється
1	0	УАПП-9 біт. Фіксована швидкість передачі
1	1	УАПП-9 біт. Швидкість передачі, що змінюється

Послідовний порт МК51 може працювати в чотирьох різноманітних режимах.

Режим 0. У цьому режимі інформація і передається, і приймається через зовнішній вивід входу приймача (RXD). Приймаються або передаються 8 бітів даних. Через зовнішній вивід виходу передавача (TXD) видаються імпульси зсуву, що супроводжують кожен біт. Частота передачі бітів інформації дорівнює 1/12 частоти резонатора.

Режим 1. У цьому режимі через TXD передаються або з RXD приймаються 10 бітів інформації: старт-біт (0), 8 бітів даних і стоп-біт (1). Швидкість прийому/передачі є величиною змінною і задається таймером.

Режим 2. У цьому режимі через TXD передаються або з RXD приймаються 11 бітів інформації: старт-біти, 8 бітів даних, 9-й біт, що програмується, і стоп-біти. При передачі 9-й біт даних може приймати значення 0 або 1, або, наприклад, для підвищення вірогідності передачі шляхом контролю парності в нього може бути поміщене значення ознаки паритету з слова стану програми (PSW.0). Частота прийому/передачі вибирається програмою і може дорівнювати або 1/32, або 1/64 частоти резонатора в залежності від керуючого біта SMOD.

Режим 3. Режим 3 збігається з режимом 2 в усіх деталях, за винятком частоти прийому/передачі, що є величиною змінною і задається таймером.

Регістр управління/статусу УАПП

Управління режимом роботи УАПП здійснюється через спеціальний регістр з символічним ім’ям SCON. Цей регістр містить не тільки

керуючі біти, визначаючі режим роботи послідовного порту, але і дев'ятий біт даних (RB8 і TB8), що приймаються або передаються, а також біти переривання прийомопередавача (RI і TI).

Швидкість прийому/передачі

Швидкість прийому/передачі залежить від значення керуючого біта SMOD регістра РУМ, значення якого наведене в табл. 8.11.

Таблиця 8.11

Регістр керування потужністю РКП

Символ	Позиція	Ім'я та призначення
SMOD	PCON.7	Подвійна швидкість передачі. Якщо біт встановлений в 1, то швидкість передачі вдвічі більша, ніж при SMOD = 0
- - -	PCON.6 PCON.5 PCON.4	Не використовуються
GF1 GF0	PCON.3 PCON.2	Прапорці, що специфікуються користувачем (прапорці загального призначення)
PD	PCON.1	Біт зниженої потужності. При встановленні біта в 1 МК переходить в режим зниженої споживаної потужності
IDL	PCON.0	Біт холостого ходу. Якщо біт встановлений в 1, то МК переходить в режим холостого ходу

Примітка. При одночасному записувані 1 у PD і IDL біт PD має перевагу. Скидання вмісту РУП виконується шляхом завантаження в нього коду 0XXX0000.

8.2.6. Система переривань

Спрощена схема переривань МК51 показана на рис. 8.16.

Зовнішні переривання **INT0** і **INT1** можуть бути викликані або рівнем, або переходом сигналу з 1 в 0 на входах МК51 у залежності від значень керуючих бітів IT0 і IT1 в регістрі TCON. Від зовнішніх переривань встановлюються прапорці IE0 і IE1 в регістрі TCON, що ініціює виклик відповідної підпрограми обслуговування переривання. Скидан-

ня цих прапорців виконується апаратурно тільки в тому випадку, якщо переривання було викликане по переходу (зрізу) сигналу.

Якщо ж переривання викликане рівнем вхідного сигналу, то скиданням прапорця IE керує відповідна підпрограма обслуговування переривання шляхом впливу на джерело переривання з метою зняття ним запиту.

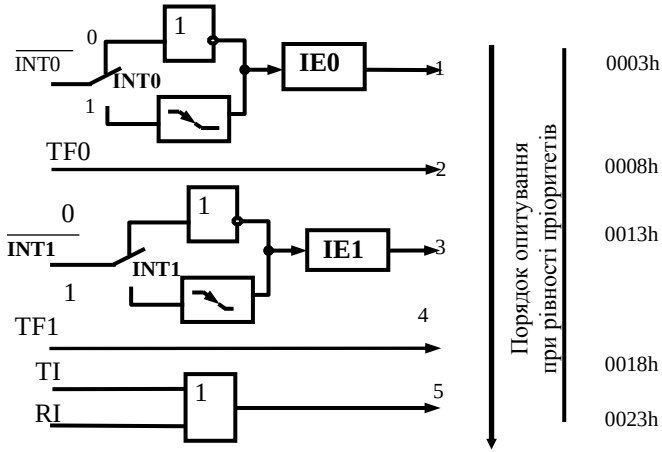


Рис. 8.16. Схема переривань МК51

Прапорці запиту переривання від таймерів TF0, TF1 скидаються автоматично при передачі управління підпрограмі обслуговування. Прапорці запитів переривання RI, TI встановлюються блоком управління УАПП апаратурно, але скидатися повинні програмою. Переривання можуть бути викликані або скасовані програмою, бо всі перераховані прапорці є програмно доступними і можуть бути встановлені/скинуті програмою з тим же результатом, якби вони були встановлені/скинуті апаратурними способами.

У блоці регістрів спеціальних функцій є два регістри, що призначені для управління режимом переривань і рівнями пріоритету. Формати цих регістрів, що мають символічні імена IE, IP, описані в табл. 8.12 і 8.13 відповідно.

Таблиця 8.12

Регістр масок переривань (РМП)

Символ	Позиція	Ім'я та призначення
EA	IE.7	Зняття блокування переривань. Скидається програмно для заборони всіх переривань, незалежно від стану IE4-IE0
–	IE.6, IE.5	Не використовуються
ES	IE.4	Біт дозволу переривання від УАПП. Встановлення/скидання програмою для дозволу/заборони переривань від прапорців TI або RI
ET1	IE.3	Біт дозволу переривання від таймера 1. Встановлення/скидання програмно для дозволу/заборони переривань від таймера 1
EX1	IE.2	Біт дозволу зовнішнього переривання від таймера 0. Встановлення/скидання програмно для дозволу/заборони переривань
ET0	IE.1	Біт дозволу переривання від таймера 0. Працює аналогічно IE.3
EX0	IE.0	Біт дозволу зовнішнього переривання 0. Працює аналогічно IE.2

Таблиця 8.13

Регістр пріоритетів переривань

Символ	Позиція	Ім`я та призначення
–	IP.7 – IP.5	Не використовується
PS	IP.4	Біт пріоритету УАПП
PT1	IP.3	Біт пріоритету таймера 1
PX1	IP.2	Біт пріоритету зовнішнього переривання 1
PT0	IP.1	Біт пріоритету таймера 0
PX0	IP.0	Біт пріоритету зовнішнього переривання 0

Можливість програмного встановлення/скидання будь-якого керуючого біта в цих двох регістрах робить систему переривань МК51 винятково гнучкою.

Прапорці переривань опитуються в момент S5P2 кожного машинного циклу. Ранжування переривань за рівнем пріоритетів виконується за певний проміжок наступного машинного циклу.

Система переривань сформує апаратурно виклик (LCALL) відповідної підпрограми обслуговування, якщо вона не заблокована однією з таких умов:

1) на цей момент обслуговується запит переривання рівного або більш високого рівня пріоритету;

2) поточний машинний цикл – не останній в циклі команди, що виконується;

3) виконується команда RETI або будь-яка команда, пов'язана зі звертанням до регістрів IE або IP.

Якщо прапорець переривань був встановлений, але за однією з вище перерахованих умов не отримав обслуговування і на час закінчення блокування вже був скинутий, то запит переривання втрачається і ніде не запам'ятовується.

За кодом LCALL, що апаратурно сформувався, система переривань розміщує в стек тільки вміст лічильника команд (PC) і завантажує в лічильник команд адресу вектора відповідної підпрограми обслуговування. За адресою вектора повинна бути розміщена команда безумовної передачі управління (JMP) до початкової адреси підпрограми обслуговування переривання. Підпрограма обслуговування у разі необхідності повинна починатися командами запису в стек (PUSH) слова стану програми (PSW), акумулятора, розширювача, покажчика даних і т. ін. та закінчуватися командами відновлення зі стека (POP). Підпрограма обслуговування переривання обов'язково завершується командою RETI, за якою в лічильник команд перевантажується зі стека збережена адреса повернення до основної програми. Команда RET також повертає управління перериванням основній програмі, але при цьому не знімає блокування переривань, що приводить до необхідності мати програмний механізм аналізу закінчення процедури обслуговування даного переривання.

8.2.7. Скидання, режим холостого ходу і режим зниженого енергоспоживання

Скидання. Скидання МК51 здійснюється шляхом подачі на вхід RST сигналу 1. Для впевненого скидання МК51 цей сигнал 1 повинен бути утриманий на вході RST щонайменше протягом двох машинних циклів (24 періоди резонатора).

Під впливом сигналу RST скидається вміст регістрів: PC, ACC, B, PSW, DPTR, TMOD, TCON, T/C0, T/C1, IE, IP і SCON, в регістрі PCON скидається тільки старший біт, в регістр-покажчик стека завантажується код 07H, а в порти P0 – P3 – коди 0FFH. Стан регістра

SBUF невизначений. Сигнал RST не впливає на вміст комірок РПД. Коли вмикається електроживлення (V_{CC}), вміст РПД залишається невизначеним, за винятком операції повернення з режиму зниженого енергоспоживання.

На рис. 8.17 показана схема автоматичного формування сигналу RST при вмиканні електроживлення. Час, необхідний для повної зарядки ємності, забезпечує упевнений запуск резонатора і його роботу протягом більш ніж двох машинних циклів. Рекомендоване значення ємності конденсатора С1 дорівнює 10 мкФ. Опір резистора R1 складає 10 кОм. У деяких типах МК резистор R1 вбудовано до схеми МК, і в схемі формування сигналу RST елемент R1 є відсутнім.

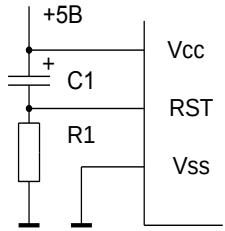


Рис. 8.17. Схема формування сигналу RST

Режим холостого ходу. Будь-яка команда, за якою встановиться керуючий біт IDL(PCON.0) в регістрі управління потужністю, переведе МК51 у режим холостого ходу. При цьому продовжує роботу внутрішній генератор синхро-сигналів. Всі регістри зберігають своє значення. На виводах всіх портів утримується логічний стан, в якому вони були у момент переходу в режим холостого ходу. На виводах ALE і PSEN формується рівень 1.

Вийти з режиму холостого ходу можна або за сигналом RST, або за перериванням. Будь-який з дозволених сигналів переривання приведе до апаратурного скидання біта PCON.0 і припинить таким чином режим холостого ходу. Після виконання команди RETI (вихід з підпрограми обслуговування переривання) буде виконана команда, яка йде в програмі за командою, що перевела МК51 у режим холостого ходу (у деяких модифікаціях МК51 цей режим може бути відсутнім).

Режим зниженого енергоспоживання. У тих застосуваннях МКС, де споживання електроенергії, а отже, габаритні розміри і маса джерела електроживлення є одними з основних показників якості виробу, можливе використання МК51 у режимі зниженого енергоспоживання. Переведення МК51 у цей режим можливий за командою, яка встановить біт PCON.1 у регістрі управління потужністю. У цьому режимі зупиняється генератор синхросигналів, вміст ОЗП і регістрів спеціальних функцій зберігається, а на вихідних

контактах портів утримуються значення, відповідні вмісту їх буферних регістрів. Виходи сигналів ALE і PSEN скидаються. При цьому електроживлення здійснюється через вивід RST/VPD. Напруга електроживлення (V_{CC}) може бути відключена. Перед виходом з режиму вона повинна бути відновлена до номінального значення.

Вихід з режиму зниженого енергоспоживання можливий тільки за сигналом RST. При цьому перевизначаються всі регістри спеціальних функцій, але вміст ОЗП не змінюється (у деяких модифікаціях МК51 цей режим може бути відсутнім).

Захист від падіння напруги. При немиттєвих відмовах блока електроживлення МКС можна забезпечити збереження вмісту ОЗП за допомогою малопотужного (батарейного) аварійного джерела електроживлення, під'єднаного до виводу RST/VPD. Для цього МК при отриманні повідомлення про загрозливе падіння напруги VCC (переривання від системи контролю електроживлення) повинен перенавантажувати в ОЗП основні параметри перерваного процесу і видати сигнал, що дозволяє підключити до виводу RST/VPD аварійне джерело живлення. Всі ці процедури МК повинен встигнути виконати до того, як напруга основного джерела електроживлення стане нижче робочої межі. Після відновлення номінального значення напруги в основному колі електроживлення виконується системне скидання і джерело аварійного електроживлення може бути відключене.

9. СИСТЕМА КОМАНД МК i8051

9.1. Формати і способи адресації команд

Команди можуть займати від одного до трьох байтів. Розмір команди визначається кодом операції, записаним у першому байті. Додаткові байти можуть містити адреси і/або дані. У символічному записі команд як правило указуються всі операнди. У машинних командах частина інформації, необхідної для адресації, може міститися в коді операції. За допомогою байта коду операції можна закодувати 256 команд, але для мікроконтролера 8051 використовується тільки 255. Код 0A5h зарезервований для подальшого розвитку сімейства.

Адресація команд (тобто їх виконання) одна за одною називається *природною*. При завершенні зчитування чергової команди вміст програмного лічильника містить адресу коду операції наступної команди. Команди, у результаті виконання яких може бути змінений природний порядок їх виконання, називаються *керуючими*. Передача керування може відбуватися в залежності від виконання деяких умов. Тоді це називається умовною передачею керування (conditional jump). Якщо команда завжди передає керування в іншу частину програми, то це називається *безумовною передачею керування* (unconditional jump). Деякі з умовних керуючих команд використовують інформацію, що міститься в слові стану програми, а інші самі порівнюють байти чи перевіряють вміст бітів ОЗП. В операнді керуючої команди в загальному випадку повинна міститися інформація для зміни коду в програмному лічильнику. Способи адресації керуючих команд у мікроконтролері типу 8051 розрізняються за дальністю переходу на короткі (short), абсолютні (absolute) і довгі (long). Для передачі керування програмісту досить вказати символічну адресу переходу у відповідному операнді асемблерної команди.

При використанні короткого способу адресації в останньому байті команди міститься різниця між адресою тієї команди, якій передається керування, і адресою команди, що надходить за керуючою командою.

Ця різниця може складати від –128 до +127. Для обчислення нового вмісту програмного лічильника із вмісту останнього байта команди спочатку формується двобайтовий код за допомогою запису старшого (знакового) біта в усі розряди старшого байта. Потім двобайтовий код додається до вмісту програмного лічильника. Такий спосіб часто називають *відносним* (relative).

Назва абсолютного переходу успадкована від попередньої моделі мікроконтролера, у якого обсяг ПЗП був обмежений двома кілобайтами. При переході до 64 кбайтів старий адресний простір стали називати *сторінкою* (page). Тому 3 старших біти адреси переходу містяться в коді операції, а 8 молодших – у другому байті команди. Цей спосіб забезпечує адресацію в межах однієї з 32-х сторінок ПЗП, номер якої визначається 5 старшими розрядами коду операції. При абсолютному способі адресації 11 молодших розрядів вмісту програмного лічильника заміняються вмістом адресної частини команди. Для довгого переходу адресна частина команди складається з двох байтів, вміст яких заноситься в програмний лічильник. Короткий, абсолютний і довгий безумовні переходи позначаються в мнемокодах команди початковими літерами S (Short), A (Absolute) і L (Long) відповідно.

Команди безумовного переходу можна розділити на команди без запам’ятовування адреси повернення, команди із запам’ятовуванням адреси повернення і команди повернення. В останньому різновиді команд безумовного переходу адресна частина відсутня. Є також команда безумовного переходу, що використовує індексну адресацію.

Список команд поділено функціонально на 4 групи: команди передачі даних, команди порозрядної обробки інформації (логічні операції та операції з бітами), арифметичні команди і команди передачі управління. Існує одна команда, яку не можна віднести ні до однієї з груп, оскільки вона не робить нічого протягом одного такту: NOP.

Однак ця команда (No OPeration – немає операції) необхідна для роботи в реальному масштабі часу, щоб забезпечити короткочасну затримку перед виконанням наступної команди.

9.2. Команди передачі даних

Незважаючи на передачу даних у незмінному вигляді, ці команди здійснюють один зі способів обробки інформації. Як приклад такої

обробки можна навести сортування. У командах пересилання використовуються всі варіанти використання способів адресації даних. Пересилання даних може здійснюватися у форматах байта, половини байта, двох байтів і біта.

Почнемо з байтового формату. Команда MOV (MOVe з англійської "пересунути") копіює вміст джерела в приймач (при виконанні цієї команди первісний вміст приймача зникає):

```
MOV     A, #src
MOV     A,  Rn
MOV     A,  @Ri
MOV     A,  src
MOV     Rn,  A
MOV     Rn,  #src
MOV     Rn,  src
MOV     @Ri,  A
MOV     @Ri,  # src
MOV     @Ri,  src
MOV     dst,  A
MOV     dst,  #src
MOV     dst,  Rn
MOV     dst,  @Ri
MOV     dst,  src
```

При описі команд використовуються такі позначення:

src – число зі значенням src;

src – адреса зі значенням src;

@ src – непряма адресація з джерела src (R0, R1, DPTR, A + DPTR, A + + PC).

Для посилання нуля в нагромаджувач простіше використовувати команду очищення CLR (CleaR означає "очистити"):

```
CLR  A
```

Читання і запис даних байтового формату при звертанні до зовнішнього ОЗП здійснюється за допомогою команд MOVX, де літера X означає eXternal (зовнішня пам'ять):

```
MOVX  A, @Ri      MOVX  @Ri,  A
MOVX  A, @DPTR    MOVX  @DPTR,  A
```

Перед виконанням цієї команди у відповідний регістр потрібно записати адреси.

Читання даних із ПЗП здійснюється за допомогою команди MOVC, при цьому літера C– це Code (програма):

```
MOVC  A,  @A + DPTR
MOVC  A,  @A + PC
```

Ці команди дуже зручні для зчитування з таблиць, записуваних у ПЗП.

Запис в ОЗП і зчитування з нього за допомогою стекового способу адресації здійснюється командами:

```
PUSH   src
POP    dst
```

Мнемокоди стекових команд відповідають дієсловам “заштовхнути” і “виштовхнути”.

Існує ще одна команда копіювання XCH (eXCHange означає “обміняти”), що здійснює обмін вмісту джерела і приймача. Також обмін можна здійснити за допомогою трьох команд пересилання. Команди, наведені нижче, виконують це за той же час, але займають менше місця в ПЗП і не вимагають використання додаткової комірки ОЗП:

```
XCH  A,  Rn
XCH  A,  @Ri
XCH  A,  src
```

Є також команда, що обмінює молодші половини байтів:

```
XCHD  A,  @Ri
```

Тут D означає Digit (чотири біти використовуються для двійкового подання десяткової цифри).

Одна з команд пересилання даних записує два байти в регістр покажчика даних:

```
MOV  DPTR,  #src
```

Інших команд для явного пересилання двобайтових даних не існує.

Кілька команд пересилання інформації працюють у бітовому форматі. У команді MOV джерелом або приймачем повинен бути біт переносу C:

```
MOV  C,  flag
MOV  flag,  C
```

Для запису констант 0 і 1 використовуються команди очищення CLR і установки SETB (SET Bit означає “установити біт”):

```
CLR  C          SETB  C
CLR  flag       SETB  flag
```

Команда пересилання не впливає на вміст слова стану програми, за винятком випадків пересилання інформації в цей регістр або в один з його бітів.

Група команд передачі даних наведена у табл. 9.1.

Таблиця 9.1

Група команд передачі даних

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Пересилання регістра в акумулятор	MOV A,Rn	11101rrr	1	1	1	(A)←(Rn)
Пересилання в акумулятор байта, що адресується прямо	MOV A,ad	11100101	3	2	1	(A)←(ad)
Пересилання в акумулятор байта із РПД (i = 0,1)	MOV A,@Ri	1110011i	1	1	1	(A)←((Ri))
Завантаження в А константи	MOV A,#d	01110100	2	2	1	(A)←#d
Пересилання в регістр із А	MOV Rn, A	11111rrr	1	1	1	(Rn) ←(A)
Пересилання в регістр байта, що адресується прямо	MOV Rn, ad	10101rrr	3	2	1	(Rn)←(ad)
Завантаження в регістр константи	MOV Rn, #d	01111rrr	2	2	1	(Rn) ←(#d)
Пересилання за прямою адресою Рг	MOV ad, Rn	10001rrr	3	2	2	(ad) ← (Rn)
Пересилання за прямою адресою А	MOV ad, A	11110101	3	2	1	(ad) ← (A)
Пересилання байта, що адресується прямо, за прямою адресою	MOV add, ads	10000101	9	3	2	(add) ← (ads)
Пересилання байта з РПД за прямою адресою	MOV ad,@Ri	1000010i	3	2	2	(ad) ←((Ri))
Пересилання за прямою адресою незмінної	MOV ad, #d	01110101	7	3	2	(ad) ←#d
Пересилання в РПД з А	MOV@Ri, A	1111011i	1	1	1	((Ri)))←(A)
Пересилання в РПД байта, що адресується прямо	MOV@Ri, ad	0110011i	3	2	2	((Ri)))← (ad)
Пересилання в РПД константи	MOV@Ri, #d	0111011i	2	2	1	((Ri)))← #d
Завантаження покажчика даних	MOV DPTR,#d16	10010000	13	3	2	(DPTR)←#d16
Пересилання в акумулятор байта із ПП	MOVC A, @A + + DPTR	10010011	1	1	2	(A)←((A) + + (DPTR))

Закінчення табл. 9.1

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Пересилання в акумулятор байта із ПП	MOVC A,@A+PC	10000011	1	1	2	(PC) ← (PC)=1 (A)←((A)+ (PC))
Пересилання в акумулятор байта із ЗПД	MOVX A,@Ri	1110001i	1	1	2	(A) ← ((Ri))
Пересилання в акумулятор байта із розширеної ЗПД	MOVX A,@DPTR	11100000	1	1	2	(A)← ((DPTR))
Пересилання в ЗПД із акумулятора	MOVX @Ri, A	1111001i	1	1	2	((Ri)) ←(A)
Пересилання в ЗПД, що розширена, із акумулятора	MOVX @DPTR, A	11110000	1	1	2	((DPTR)) ←(A)
Завантаження в стек	PUSH ad	11000000	3	2	2	(SP)←(SP) + 1 ((SP))←(ad)
Вивантаження із стека	POP ad	11010000	3	2	2	(ad)←(SP) (SP)←(SP) – 1
Обмін акумулятора з регістром	XCH A, Rn	11001rrr	1	1	1	(A)↔(Rn)
Обмін акумулятора з байтом, що адресується прямо	XCH A, ad	11000101	3	2	1	(A)↔ (ad)
Обмін акумулятора з байтом із РПД	XCH A, @Ri	1100011i	1	1	1	(A)↔((Ri))
Обмін молодшої тетради акумулятора з молодшою тетрадою байта із РПД	XCHB A,@Ri	1101011i	1	1	1	$(A_{0-3})\leftrightarrow((Ri)_{0-3})$

9.3. Команди порозрядної обробки інформації

Команди порозрядної обробки інформації відрізняються від команд арифметичних операцій тим, що працюють з окремими бітами незалежно від вмісту байта в цілому. Такі команди призначені для байтового і бітового формату даних. Їх також називають логічними командами, тому що з їх допомогою можна обчислювати функції алгебри логіки НІ, І, АБО і ВИКЛЮЧНЕ АБО. Наведемо для довідки таблицю значень цих функцій:

X	У	/X	X.AND.Y	X.OR.Y	X.XOR.Y
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Усі команди порозрядної обробки інформації не впливають на вміст слова стану програми, за винятком випадків безпосереднього запису результату в цей регістр чи його біти.

Для одержання зворотного коду (логічна функція АБО) при байтовому форматі даних призначена команда

```
CPL   A
```

Мнемоніка цієї команди ComPLement означає “доповнення”, хоча в результаті її виконання виходить не додатковий, а зворотний код. Існує дві команди аналогічного призначення, що працюють з окремими бітами:

```
CPL   C
CPL   flag
```

Крім того, інверсія біта може бути використана в командах І та АБО, як показано далі.

Команди для обчислення функції І в байтовому форматі використовують різноманітні способи адресації. Очевидно, мнемоніка команди розшифровується як ANd Logical:

```
ANL  A, #src
ANL  A,  Rn
ANL  A,  @Ri
ANL  A,  src
ANL  dst,  A
ANL  dst,  #src
```

Ці команди можуть використовуватися, наприклад, для очищення окремих бітів двійкового коду, або для перевірки наявності 1 у деякому наборі бітів. Існує дві команди, що обчислюють функцію І в бітовому форматі:

```
ANL    C,  flag
ANL    C,  /flag
```

Символ косої риски в другій команді означає, що для обчислення логічної функції використовується інвертоване значення біта.

Аналогічний набір команд існує і для функції АБО в байтовому форматі з мнемонікою OR Logical:

```
ORL  A,  #src
ORL  A,  Rn
ORL  A,  @Ri
ORL  A,  src
ORL  dst,  A
ORL  dst,  #src
```

Ці команди можуть використовуватися, наприклад, для установки окремих бітів двійкового коду в 1. Маються дві аналогічні команди, що обчислюють функцію в бітовому форматі:

```
ORL   C, flag
ORL   C, /flag
```

Символ косої риски в другій команді означає, що для обчислення логічної функції використовується інвертоване значення біта.

Для функції ВИКЛЮЧНЕ АБО аналогічні команди існують тільки в байтовому форматі. Відомо, що відповідна команда розшифровується як exclusive o Logical:

```
XRL   A,  #src
XRL   A,  Rn
XRL   A,  src
XRL   A,  @Ri
XRL   dst, A
XRL   dst,  #src
```

Ці команди можуть використовуватися, наприклад, для зміни значення окремих бітів двійкового коду на зворотне (toggle). Вони можуть також використовуватися для перевірки кодів на збіг.

До команд порозрядної обробки інформації можна віднести команди циклічного зсуву вліво і вправо, що працюють з 8 бітами (нагромаджувач) або з 9 бітами (нагромаджувач + біт переносу):

```
RL    A
RR    A
RLC   A
RRC   A
```

Перша літера в мнемокодах цих команд означає Rotate (повертати), друга вказує на напрямок (Left чи Right), а третя – на участь біта переносу. При зсуві вліво в усі біти нагромаджувача, крім наймолодшого, записується старий вміст сусіднього правого біта. При зсуві вправо в усі біти нагромаджувача, крім найстаршого, записується старий вміст сусіднього лівого біта. Якщо біт переносу не бере участі в операції циклічного зрушення, то при зсуві вліво у наймолодший байт записується старий вміст найстаршого біта, а при зсуві вправо у найстарший байт записується старий вміст наймолодшого біта. При участі біта переносу його вміст включається в коло циклічного переносу, що дозволяє здійснювати зсув вмісту багатобайтових кодів.

До операції циклічного зсуву на 4 розряди без участі біта переносу можна віднести команду: SWAR A

Вона здійснює обмін напівбайтів вмісту нагромаджувача, так що її можна інтерпретувати як зсув молодшого напівбайта на 4 розряди вліво і зсув старшого напівбайта на 4 розряди вправо.

Якщо перед виконанням команди зсуву очистити біт переносу, то зсув за участю цього біта може використовуватися як команда арифметичної операції. Зсув вправо відповідає поділу додатного числа на 2, при цьому у біт переносу записується залишок від ділення. Зсув уліво відповідає множенню додатного числа на 2, при цьому 1 у біті переносу сигналізує про переповнення.

Група команд логічних операцій наведена в табл. 9.2.

Таблиця 9.2

Група команд логічних операцій

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Логічне І акумулятора та Рг	ANL A,Rn	01011rrr	1	1	1	(A)←(A)∧(Rn)
Логічне І акумулятора та байта, що адресується прямо	ANL A,ad	01010101	3	2	1	(A)←(A)∧(ad)
Логічне І акумулятора та байта із РПД	ANL A,@Ri	0101011i	1	1	1	(A)←(A)∧((Ri))
Логічне І акумулятора та константи	ANL A,#d	01010100	2	2	1	(A)←(A)∧#d
Логічне І байта, що адресується прямо, та акумулятора	ANL ad,A	01010010	3	2	1	(ad)←(ad)∧(A)
Логічне І байта, що адресується прямо, та константи	ANL ad,#d	01010011	7	3	2	(ad)←(ad)∧#d
Логічне АБО акумулятора та регістра	ORL A,Rn	01001rrr	1	1	1	(A)←(A)∨(Rn)
Логічне АБО акумулятора та байта, що адресується прямо	ORL A,ad	01000101	3	2	1	(A)←(A)∨ (ad)
Логічне АБО акумулятора та байта із РПД	ORL A,@Ri	0100011i	1	1	1	(A)←(A)∨((Ri))
Логічне АБО акумулятора та константи	ORL A,#d	01000100	2	2	1	(A)←(A)∨ #d
Логічне АБО байта, що адресується прямо, та акумулятора	ORL ad,A	01000010	3	2	1	(ad)←(ad)∨ (A)
Логічне АБО байта, що адресується прямо, та константи	ORL ad,#d	01000011	7	3	2	(ad)←(ad)∨ #d
Виключне АБО акумулятора та регістра	XRL A,Rn	01101rrr	1	1	1	(A)←(A)∀(Rn)

Закінчення табл. 9.2

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Виключне АБО акумулятора та байта, що адресується прямо	XRL A,ad	01100101	3	2	1	(A)←(A)∀ (ad)
Виключне АБО акумулятора та байта із РПД	XRL A,@Ri	0110011i	1	1	1	(A)←(A)∀((Ri))
Виключне АБО акумулятора та константи	XRL A,#d	01100100	2	2	1	(A)←(A)∀ #d
Виключне АБО байта, що адресується прямо, та акумулятора	XRL ad,A	01100010	3	2	1	(ad)←(ad)∀(A)
Виключне АБО акумулятора та константи	XRL ad,#d	01100011	7	3	2	(ad)←(ad)∀ #d
Скидання акумулятора	CLR A	11100100	1	2	1	(A)←0
Інвертування акумулятора	CPL A	11110100	1	2	1	(A)←інв(A)
Зсув акумулятора ліворуч циклічно	RL A	00100011	1	3	1	(A_{n+1})←(A_n), n=0÷6, (A_0)←(A_7)
Зсув акумулятора ліворуч циклічно через перенос	RLC A	00110011	1	1	1	(A_{n+1})←(A_n), n=0÷6, (A_0)←(C), (C)←(A_7)
Зсув акумулятора праворуч циклічно	RR A	00000011	1	1	1	(A_n)←(A_{n+1}), n=0÷6, (A_7)←(A_0)
Зсув акумулятора праворуч циклічно через перенос	RRC A	00010011	1	1	1	(A_n)←(A_{n+1}), n=0÷6, (A_7)←(C), (C)←(A_0)
Обмін місцями тетрад в акумуляторі	SWAP A	11000100	1	1	1	(A_{0-3})↔(A_{4-7})

Група команд операцій з бітами наведена в табл. 9.3.

9.4. Команди арифметичних операцій

У раніше розглянутих командах обробки інформації кодування вмісту окремих бітів байта не мало значення. В арифметичних операціях потрібно дотримуватись чітко визначених правил запису даних відповідно до порядкових номерів розрядів двійкового коду. У зв'язку з обмеженими ресурсами мікроконтролера в ньому використовуються тільки чотири арифметичних операції з цілими числами. Недотримання правил кодування призводить до неточного виконання арифметичних операцій.

Таблиця 9.3

Група команд операцій з бітами

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Скидання переносу	CLR C	11000011	1	1	1	(C)←0
Скидання біта	CLR bit	11000010	4	2	1	(b) ←0
Встановлення переносу	SETB C	11010011	1	1	1	(C)←1
Встановлення біта	SETB bit	11010010	4	2	1	(b) ←1
Інверсія переносу	CPL C	10110011	1	1	1	(C)←(інвC)
Інверсія біта	CPL bit	10110010	4	2	1	(b) ← (інвb)
Логічне І біта і переносу	ANL C,bit	10000010	4	2	2	(C)←(C) ∧ (b)
Логічне І інверсії біта і переносу	ANL C,/bit	10110000	4	2	2	(C)←(C) ∧ (інвb)
Логічне АБО біта і переносу	ORL C,bit	01110010	4	2	2	(C)←(C) ∨ (b)
Логічне АБО інверсії біта і переносу	ORL C,/bit	10100000	4	2	2	(C)←(C) ∨ (інвb)
Пересилання біта в перенос	MOV C,bit	10100010	4	2	1	(C)←(b)
Пересилання переносу в біт	MOV bit,C	10010010	4	2	2	(b) ←(C)

В одному байті може бути закодовано 256 значень цілого числа. При роботі з додатними числами це відповідає значенням від 0 до 255. Усі команди арифметичних операцій призначені для роботи з додатними цілими числами байтового формату, хоча команди додавання і віднімання у разі відсутності переповнення забезпечують одержання коректного результату при спеціальному способі кодування від'ємних чисел. При необхідності роботи з числами, що не можуть бути подані в байтовому форматі, необхідно розробляти відповідні підпрограми.

Команда додавання працює з даними байтового формату, при цьому як приймач завжди використовується тільки нагромаджувач:

```
ADD     A, #src
ADD     A, Rn
ADD     A, @Ri
ADD     A, src
```

Мнемоніка цієї команди відповідає слову ADDition (додавання).

Для роботи з числами, що не можуть бути подані у вигляді одного байта, використовується команда додавання, яка враховує перенос, отриманий при додаванні попередньої пари байтів:

```
ADDC    A, #src
ADDC    A, Rn
ADDC    A, @Ri
```

ADDC A, src

Додавання літери C до позначення команди вказує на використання біта переносу (ADDition with Carrier).

Існує також команда додавання, за допомогою якої здійснюється збільшення заданого операнда на одиницю (INCrement):

```
INC     A
INC     Rn
INC     @Ri
INC     src
```

Така ж команда є і для роботи з двома байтами регістра покажчика даних:

INC DPTR

За допомогою цієї команди можна змінювати вміст покажчика для зчитування послідовності байтів із ПЗП.

При додаванні чисел, поданих двійково-десятковими кодами, після операції додавання потрібно використовувати команду десяткової корекції суми

DA A

Двійково-десяткове кодування має дуже обмежене застосування і тому далі не розглядається.

Набір команд для віднімання є набагато меншим. Команда обчислення різниці існує тільки у варіанті з відніманням вмісту біта переносу (не вистачило кодів команд!):

```
SUBB    A, #src
SUBB    A, Rn
SUBB    A, @Ri
SUBB    A, src
```

Мнемоніка цієї команди відповідає словам SUBtraction with Borrow (тобто віднімання з урахуванням позики, тому що при відніманні утворюється позика, а не перенос). З цієї причини перед обчисленням різниці молодших байтів потрібно обов'язково очищати біт переносу, якщо немає впевненості в його вмісті. При обчисленні різниці старших байтів цього робити не потрібно.

Існує також команда віднімання, за допомогою якої здійснюється зменшення заданого операнда на одиницю (DECrement):

```
DEC  A
DEC  Rn
DEC  src
DEC  @Ri
```

Команди зменшення для роботи з двобайтовим форматом даних не існує.

Результати виконання команд додавання і віднімання впливають на вміст бітів переносу, додаткового переносу і переповнення в слові стану програми. Результати виконання команд додавання і віднімання не впливають на вміст слова стану програми.

Команди множення (MULliplication) і ділення (Division) працюють при записі операндів у нагромаджувач і регістр B. Для команди множення порядок запису співмножників у ці регістри не може бути довільним: MUL AB

Добуток має двобайтовий формат. Молодший байт добутку записується в нагромаджувач, а старший – у регістр B.

Для команди ділення ділене повинне бути записане в нагромаджувач, а дільник – у регістр B:

DIV AB

Після виконання команди в акумуляторі знаходиться частка, а в регістрі B – залишок. Після виконання команд множення і ділення в біт переносу заноситься 0. Якщо старший байт добутку не дорівнює нулю, то в байт переповнення заноситься 1. При діленні на 0 у байт переповнення також заноситься 1.

Група команд арифметичних операцій наведена в табл. 9.4.

Таблиця 9.4

Група команд арифметичних операцій

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Додавання акумул. до регістра (n = 0÷7)	ADD A, Rn	00101rrr	1	1	1	(A)←(A)+(Rn)
Додавання акумул. до байта, що адресується прямо	ADD A, ad	00100101	3	2	1	(A)←(A)+(ad)
Додавання акумулятора до байта із РПД (i = 0,1)	ADD A, @Ri	0010011i	1	1	1	(A)←(A)+((Ri))
Додавання акумулятора до незмінної	ADD A, #d	00100100	2	2	1	(A)←(A)+ #d
Додавання акумулятора до регістра з переносом	ADDC A, Rn	00111rrr	1	1	1	(A)←(A)+(Rn)+(C)
Додавання акумулятора до байта, що адресується прямо, з переносом	ADDC A, ad	00110101	3	2	1	(A)←(A)+(ad)+(C)
Додавання акумулятора до байта із РПД (i = 0,1) з переносом	ADDC A, @Ri	0011011i	1	1	1	(A)←(A)+((Ri))+(C)

Закінчення табл. 9.4

Назва команди	Мнемокод	КОП	Т	Б	Ц	Операція
Додавання акумулятора до незмінної з переносом	ADDC A, #d	00110100	2	2	1	(A)←(A)+#d+(C)
Десяткова корекція акумулятора	DA A	11010100	1	1	1	Якщо (A_{0-3}) >9 ∨((AC)=1), то (A_{0-3}) ← (A_{0-3})+6, потім, якщо (A_{4-7}) >9 ∨ ((C)=1), то (A_{4-7})← (A_{4-7})+6
Віднімання з акумулятора регістра і позики	SUBB A, Rn	10011rrr	1	1	1	(A)←(A)-(Rn)-(C)
Віднімання з акумулятора байта, що адресується прямо, та позики	SUBB A, ad	10010101	3	2	1	(A)←(A)-(ad)-(C)
Віднімання з акумулятора байта РПД та позики	SUBB A,@Ri	1001011i	1	1	1	(A)←(A)-((Ri))-(C)
Віднімання з акумулятора незмінної та позики	SUBB A, #d	10010100	2	2	1	(A)←(A)- #d -(C)
Інкремент акумулятора	INC A	00000100	1	1	1	(A)←(A)+1
Інкремент регістра	INC Rn	00001rrr	1	1	1	(Rn)←(Rn)+1
Інкремент байта, що адресується прямо	INC ad	00000101	3	2	1	(ad)←(ad)+1
Інкремент байта в РПД	INC @Ri	0000011i	1	1	1	((Ri))←((Ri))+1
Інкремент покажчика даних	INC DPTR	10100011	1	1	2	(DPTR)←(DPTR)+1
Декремент акумулятора	DEC A	00010100	1	1	1	(A)←(A) -1
Декремент регістра	DEC Rn	00011rrr	1	1	1	(Rn)←(Rn) -1
Декремент байта, що адресується прямо	DEC ad	00010101	3	2	1	(ad)←(ad) -1
Декремент байта в РПД	DEC @Ri	0001011i	1	1	1	((Ri))←((Ri)) -1
Множення акумулятора на РгВ	MUL AB	10100100	1	1	4	(B)(A)←(A)*(B)
Ділення акумулятора на РгВ	DIV AB	10000100	1	1	4	(A).(B)← (A)/(B)

9.5. Команди передачі управління

Опис керуючих команд почнемо з команд умовного переходу. Ці команди використовують тільки відносний спосіб адресації, тому для них будемо використовувати умовне позначення адреси переходу rel. Для кожної умови існує пара команд, одна з яких здійснює передачу

керування при її дотриманні, а інша – при недотриманні. У поле коментарів вводяться розшифровки мнемоніки цих команд. Передача керування залежить від того, чи дорівнює нулю вміст регістра нагромаджувача:

```
JZ    rel      ;Jump if Zero
JNZ   rel      ;Jump if No Zero
```

Можна також використовувати як умову переходу рівність біта переносу одиниці або нулю:

```
JC    rel      ;Jump if Carry
JNC   rel      ;Jump if Not Carry
```

Існують команди, що використовують як умову переходу рівність одиниці чи нулю вмісту будь-якого біта у функціональному регістрі чи адресованого біта в ОЗП:

```
JB    flag, rel      ;Jump if Bit
JNB   flag, rel      ;Jump if No Bit
```

Команда передачі керування за рівністю біта одиниці має варіант з очищенням умісту цього біта:

```
JBC   flag, rel      ; Jump if Bit and Clear
```

Команди з взаємовиключаючими умовами дозволяють обійти обмеження, пов'язані зі способом адресації. Якщо адреса команди, на яку потрібно передати керування, відрізняється від адреси наступної команди на велику величину (додатну чи від'ємну), то можна використовувати пару команд, перша з якої при дотриманні зворотної умови передає керування через один рядок вихідного тексту, а друга є командою безумовного переходу з абсолютною або далекою адресацією.

Перераховані команди здійснюють перехід у залежності від результатів попередніх обчислень. Однак є керуючі команди, які самі здійснюють обчислення для одержання умов передачі керування. Мнемокод першої з таких команд – CJNE (Compare and Jump if Not Equal означає "порівняти і перейти, якщо не дорівнює"). Це єдина команда мікроконтролера, що має 3 операнди. Її чотири різновиди відрізняються способами адресації джерела й приймача:

```
CJNE  A,    src, rel
CJNE  A,    #src, rel
CJNE  Rn,   #src, rel
CJNE  @Ri,  #src, rel
```

Команда обчислює різницю першого і другого операндів, але результат віднімання нікуди не записується, за винятком біта переносу. Передача керування за зазначеною адресою здійснюється при нерівності операндів. При порівнянні додатних чисел біт переносу встановлюється в 1, якщо перший операнд менше другого. Якщо за адресою переходу записати команду передачі керування за вмістом біта переносу, то можна виконати три різні блоки програми.

Інша команда – DJNZ (Decrement and Jump if Not Zero) означає "зменшити і перейти, якщо не дорівнює нулю") зменшує вміст першого операнда на одиницю. Якщо операнд не дорівнює 0, то керування передається за зазначеною адресою:

DJNZ Rn, rel

DJNZ dst, rel

Ця команда зручна для програмування циклу за лічильником. Перед початком циклу за адресою приймача треба записати число, яке дорівнює кількості повторень циклу. Якщо в другому операнді записати адресу початку циклу і не змінювати вміст першого операнда іншими командами в циклі, то задана ділянка програми буде повторена певну кількість разів.

Перейдемо до команд безумовного переходу JuMP без повернення. Вони використовують адресацію всіх трьох довжин:

SJMP rel
AJMP adr11
LJMP adr16

Цифри в умовних позначках операндів останніх двох команд вказують на кількість розрядів адресної частини їхніх кодів.

Команда безумовного переходу з індексною адресацією дозволяє змінювати адресу переходу за вмістом нагромаджувача:

JMP @A + DPTR

За допомогою цієї команди можна здійснювати переходи за кожною з 256 адрес відносно вмісту регістра покажчика. Ця команда може використовуватися як перемикач, якщо в заданій області програми записати команди безумовного переходу на деякі блоки програми. Оскільки ці команди будуть займати більше одного байта,

розглянута команда може використовуватися для передачі керування не більше ніж за 128 адресами.

Мнемоніка команд безумовного переходу з поверненням CALL перекладається як виклик. Ці команди не використовують короткої адресації:

ACALL adr11

LCALL adr16

Вони застосовуються для виклику підпрограм, після виконання яких керування має повертатися команді, яка надходить за командою виклику. Для цього перед занесенням адресної частини команди виклику в програмний лічильник попереднє значення програмного лічильника записується в два байти ОЗП, адреса яких указується покажчиком стека. Спочатку в стек заноситься молодший байт програмного лічильника, а потім старший байт.

Для коректного повернення до програми останньою виконуваною командою підпрограми повинна бути

RET

Мнемоніка цієї команди відповідає слову RETurn (повернутися). Команда не має адресної частини, тому що при її виконанні в програмний лічильник записуються два байти, які адресуються покажчиком стека (першим записується старший байт, а другим – молодший). Програміст повинен забезпечити необхідний вміст покажчика стека до моменту виходу з підпрограми. Ця вимога означає, що при виконанні підпрограми кількість команд запису в стек повинна дорівнювати кількості команд зчитування зі стека. Оскільки адресний простір стека розміщується в ОЗП, програміст повинен подбати про те, щоб команди, в яких використовуються інші способи адресації, не здійснювали запис інформації в адресний простір стека.

Інша команда, що працює точно таким же способом (RETI), застосовується для повернення до програми, що виконувалася в момент апаратного переривання. Літера I у її мнемокоді відповідає слову Interrupt (переривання). Апаратні переривання передають керування фіксованим адресам із запам’ятовуванням адреси повернення.

Група команд передачі управління наведена в табл. 9.5.

Таблиця 9.5

Група команд передачі управління

Назва команди	Мнемокод	КОП	Т	Б	П	Операція
Довгий перехід у повному обсязі пам`яті програм	LJMP ad 16	00000010	12	3	2	(PC)←ad 16
Абсолютний перехід усередині сторінки в 2 кбайти	AJMP ad 11	$a_{10}a_9a_8$ 00001	6	2	2	(PC)←(PC) + 2 ($PC_{0\text{-}10}$)←ad 11
Короткий відносний перехід усередині сторінки розмыром в 256 байтів	SJMP rel	10000000	5	2	2	(PC)←(PC) + 2 (PC)←(PC) + re1
Непрямий відносний перехід	JMP @A+DPTR	01110011	1	1	2	(PC)←(A) + (DPTR)
Перехід, якщо акумулятор дорівнює нулю	JZ rel	01100000	5	2	2	(PC)←(PC) + 2, якщо (A) = 0, то (PC)←(PC)+ re1
Перехід, якщо акумулятор не дорівнює нулю	JNZ rel	01110000	5	2	2	(PC)←(PC) + 2, якщо (A) ≠ 0, то (PC)←(PC)+re1
Перехід, якщо перенос дорівнює одиниці	JC rel	01000000	5	2	2	(PC)←(PC) + 2, якщо (C) = 1, то (PC)←(PC) + re1
Перехід, якщо перенос дорівнює нулю	JNC rel	01010000	5	2	2	(PC)←(PC) + 2, якщо (C) = 0, то (PC)←(PC) + re1
Перехід, якщо біт дорівнює одиниці	JB bit, rel	00100000	11	3	2	(PC)←(PC) + 3, якщо (b) = 1, то (PC)←(PC) + re1
Перехід, якщо біт дорівнює нулю	JNB bit, rel	00110000	11	3	2	(PC)←(PC) + 3, якщо (b) = 0, то (PC)←(PC) + re1
Перехід, якщо біт встановлений, з наступним скиданням біта	JBC bit, rel	00010000	11	3	2	(PC)←(PC) + 3, якщо (b) = 1, то (b)←0 та (PC)←(PC) + re1
Декремент Рг та перехід, якщо не нуль	DJNZ Rn, rel	11011rrr	5	2	2	(PC)←(PC) + 2, (Rn)←(Rn) - 1, якщо (Rn) ≠ 0, то (PC)←(PC) + re1
Декремент байта, що адресується прямо, та перехід, якщо не нуль	DJNZ ad, rel	11010101	8	3	2	(PC)←(PC)+2, (ad)←(ad) -1, якщо (ad)≠0,то (PC)←(PC) + re1

Зведення команд i8051 за абеткою

```
ACALL   n                   ; перехід до підпрограми, ближній
ADD     A,      {#|R|@|d}   ; додавання
ADDC    A,      {#|R|@|d}   ; додавання з урахуванням переносу
AJMP    n                   ; перехід  безумовний, ближній
ANL.    A,      {#|R|@|d}   ; I
ANL     d,  {A|#}           ; I
ANL     C, b                ; I  зі значенням біта
ANL     C, /b               ; I  зі зворотним значенням біта
CJNE    A,  {#|d},  s       ; перехід за нерівністю, короткий
CJNE    {R|@}, #,  s        ; перехід за нерівністю, короткий
CLR     {A|b|C}             ; очищення {нагромаджувача|біта}
CPL     {A|b|C} ; одержання зворотного коду {нагромаджувача|біта}
 DA     A                   ; корекція двійково-десяткового коду
 DEC    {A|R|@|d}           ; зменшення на 1
 DIV    AB                  ; ділення
 DJNZ   {R|d},  s           ; зменшення на 1 та перехід  якщо ні,
                               ; 0 короткий
 INC    {A|R|@|d}           ; збільшення на  1
 INC    D                   ; збільшення на  1
 JB      b,  s              ; перехід по 1, короткий
 JBC     b,  s              ; перехід по 1 з очищенням, короткий
 JNB     b,  s              ; перехід по 0, короткий
 JC      s                  ; перехід по 1 переносу, короткий
 JNC     s                  ; перехід по 0 переносу, короткий
 JMP     @A+D               ; безумовний  перехід непрямий
 JNZ     s                  ; перехід якщо не нуль, короткий
 JZ      s                  ; перехід якщо нуль, короткий
 LCALL   f                  ; перехід до підпрограми, далекий
 LJMP    f                  ; безумовний перехід, далекий
 MOV     A,  {#|R|@|d}      ; пересилання
MOV      d,  {A|#|R|@|d}  ; пересилання
 MOV     {R|@}  {A|#|d}   ; пересилання
 MOV     D,  #             ; пересилання
 MOV     {b,C|C,b}         ; пересилання біта
 MOVC    A,  @A+{D|P}      ; пересилання з  ПЗП
 MOVX    A,  {@D|@}        ; пересилання з зовнішнього ОЗП
 MOVX    {@D|@},  A        ; пересилання в зовнішнє  ОЗП
```

```
MUL     AB              ; множення
NOP                     ; немає операції
ORL     A,  {#|R|@|d}   ; АБО
ORL     d,  {A|#}       ; АБО
ORL     C, [ / ]  b     ; АБО з [зворотним] кодом біта
POP     d               ; зчитування зі стека
PUSH    d               ; запис у стек
RET                     ; повернення
RETI                    ; повернення з переривання
RL      A               ; зсув уліво
RLC     A               ; зсув уліво через біт переносу
RR                      A           ; зсув управо
RRC     A               ; зсув управо через біт переносу
SETB    {b|C}           ; запис 1 у біт
SJMP    s               ; безумовний перехід, короткий
SUBB    A,  {#|R|@|d}   ; обчислення з урахуванням позики
SWAP    A               ; обмін напівбайтів
XCH     A,  {R|@|d}     ; обмін байтів
XCHD    A,   @          ; обмін молодших напівбайтів між байтами
XRL     A,   {#|R|@|d}  ; виключне АБО
XRL     d,   {A|#}      ; виключне АБО
```

УМОВНІ ПОЗНАЧЕННЯ

```
{  |  }  обов'язковий вибір одного з варіантів у дужках
[ ]      використання варіанта при необхідності
#        операнд, записаний у ПЗП (відповідно до  розміру приймача)
@        непряма адресація через регістр R0 або R1
@A + D   непряма адресація через додавання DPTR і  нагромаджувача
@A + P   непряма адресація через додавання лічильника команд
         і нагромаджувача
@D       непряма  адресація через DPTR
A        операндом є вміст нагромаджувача
AB       операнди містяться в акумуляторі і регістрі  B
C        операндом є вміст біта переносу
D        операндом є вміст регістра DPTR
R        операндом є вміст регістра (від R0 до R7)
b        операндом є адресований біт
d        операндом є адресований байт
f        адреса в ПЗП від 0 до 64 кбайтів
n        адреса в поточній сторінці ПЗП (розмір сторінки 4 кбайти)
s        зсув відносно адреси команди від – 128 до + 127
```

10. ПРОГРАМУВАННЯ МК51

10.1. Загальні відомості про програмування МК51

Для програмування МК сімейства 8051 можна використовувати мови Асемблер, Бейсік, Паскаль, Сі. Перетворення тексту програми, написаної на одній з цих мов, в машинні коди здійснюється по-різному. Для Асемблера проводиться безпосередня підстановка відповідних кодів операції і необхідних даних. Мови високого рівня використовують перетворення операторів мови в машинні коди підстановкою заготівок, які містять стандартне для даного оператора рішення. Використання стандартних рішень призводить до надмірної функціональності машинного коду, збільшення його об'єму і зниження швидкодії. Після етапу компіляції може здійснюватись оптимізація коду, при якій віддаляються формально невживані ділянки, змінюється порядок проходження команд. Однією з особливостей мов високого рівня є активне використання ОЗП МК і його невелика ємність, що в свою чергу ускладнює отримання ефективного коду. Незважаючи на оптимізацію, програми, розроблені на мовах високого рівня, не дозволяють одержати таку ж швидкодію і ємність, як при програмуванні на Асемблері. З іншого боку, мови високого рівня полегшують програмування задач зі складними обчисленнями і розвиненим інтерфейсом користувача, які не критичні за часом виконання.

Програма перекладається з мови Асемблера в машинні коди МК за допомогою програм, які називаються транслятором, бібліотекарем і укладачем. На сьогодні всі вони об'єднуються в одну програму – асемблер. Необхідно розуміти різницю між мовою Асемблера – системою позначень і мнемонічними кодами операцій і програмою асемблер – програмою для перетворення початкової програми на мові Асемблера в машинний код. Для уникнення плутанини, надалі програму, що перетворює мнемонічні коди операцій в машинний код називатимемо асемблером або транслятором, а мову – мовою

Асемблера.

При розробці програми використовуються два види операторів: команди, які перетворюються транслятором на машинні коди, і директиви, які управляють процесом трансляції. На відміну від команд директиви можуть складатися з декількох рядків, тобто бути складовими. Область дії таких директив обмежена відкриваючими і закриваючими рядками. Частина директив також переводиться транслятором в машинні коди або впливає на них, а деякі директиви використовуються тільки для зручності роботи програміста.

Структура програми

Програма на мові Асемблера складається з рядків, що мають такий вигляд:

мітка: команда/директива операнди ; коментар.

Причому використання цих полів не є необов'язковим.

Для зручності сприйняття тексту програми на асемблері, прийнято, що мітка починається з першої позиції в рядку, команда/директива відокремлена табуляцією, операнди і коментар – також на відстані табуляції:

```
Label_1: MOV    A, #25    ; приклад структури програми
                ADD     A, R1     ;
; якщо в рядку тільки коментар, то він може починатися з першої позиції
```

Окрім простоти зчитування тексту програми, таке розташування міток і команд дозволяє уникнути помилок, пов'язаних з алгоритмом роботи деяких трансляторів, що приймають будь-яке ім'я на першій позиції рядка за мітку.

Ім'я мітки – рядок букв і цифр, що починається з букви – може включати службові символи, список яких залежить від програми транслятора. Використовуються букви, розташовані в першій половині таблиці ASCII символів. Після мітки, як правило, ставиться оператор ":" (двокрапка).

Ім'я мітки, як і ім'я змінної, в більшості випадків є регістрозалежним: імена Label, label, LABEL – різні, але для усунення

помилок такі імена для різних змінних використовувати не рекомендується (адже деякі транслятори не розрізняють регістр символів імені).

Числові константи повинні починатися з десяткової цифри. Шістнадцятковим константам, що починаються з літери A – F, повинна передувати цифра 0 , наприклад, 0A5B, але правильним буде запис 5AF1, оскільки константа починається з цифри "5". Для позначення системи числення, в якій записано число, використовується система префіксів і суфіксів. Префікси і суфікси систем числення наведені у табл. 10.1.

Еквівалентне записом є таке:

1234H	або		$1234	
100d		або		100
177400O	або		@177400	
01011000b	або		%01011000	

Таблиця 10.1

Префікси і суфікси систем числення

Основа системи числення	Префікс	Суфікс
2	%	B або b
8	@	O або o
10 (за замовчуванням)		D або d або нічого
16	$	H або h

Символьні константи містять один символ, взятий в лапки (“s”, “W”), і повертають ASCII значення цього символу.

Програма асемблера дозволяє на етапі трансляції виконувати деякі арифметичні і логічні обчислення. Список операторів повторює стандартний набір операторів мови C. Оператори асемблера наведені у табл. 10.2.

Для завдання послідовності виконання операцій необхідно використовувати дужки:

MOV A, #(255 – 32) еквівалентне MOV A, #223.

MOV A, #((1<<3)|(1<<1)) еквівалентне MOV A, ((1<<3)|(1<<1)) 001010b.

Можна пересувати декілька одиниць одночасно, зрушення здійснюється з нульової позиції:

MOV A, #((3<<5)|(1<<0)) еквівалентне MOV A, #01100001b.

Таблиця 10.2

Оператори асемблера

Оператор	Тип	Опис
+	Арифметичні	Додавання
-		Віднімання
*		Множення
/		Ділення
%		Ділення за модулем
<<		Логічний зсув уліво
>>		Логічний зсув управо
~	Унарні	Доповнення (порозрядне заперечення)
-		Заперечення
=	Відносини	Дорівнює
==		Дорівнює
!=		Не дорівнює
<		Менше
>		Більше
<=		Менше або дорівнює
>=		Більше або дорівнює
&	Бінарні	Двійкове “І”
|		Двійкове “АБО”
^		“Виключне АБО”

Замість цифр можуть використовуватися мітки, яким привласнені числові значення. Якщо в акумуляторі необхідно встановити біт BBB (BBB = 6), то можна використовувати команду:

ORL A, #(1<<BBB), яка еквівалентна ORL A, (1<<BBB) 1000000b.

Для запису в 8-розрядний регістр R3 старшого (молодшого) байта двобайтового числа Number можна скористатися командами:

MOV R3, #HIGH(Number), MOV R3, #LOW(Number).

Для асемблерів, які не підтримують такої форми запису, необхідно використовувати команди:

MOV R3, #(0ABCDh>>8) та MOV R3, #(0ABCDh).

10.2. Приклади програм обробки даних у МК51

10.2.1. Приклади використання команд передачі даних

***Приклад* 10.1.** Передати вміст буфера УАПП у РПД за непрямою адресою з R0:

```
MOV @R0, SBUF   ; передача прийнятого по послідовному
                ; каналу байта в РПД
```

***Приклад* 10.2.** Завантажити в покажчик даних початкову адресу 7F00H масива даних, розташованого у ВПД:

```
MOV DPTR, #7F00H ;завантаження початкового значення покажчика
                 даних
```

***Приклад* 10.3.** Завантажити керуюче слово в регістр управління таймером:

```
MOV TCON, #00000101B
```

***Приклад* 10.4.** Скинути всі прапорці користувача (область РПД з адресами 20H – 2FH):

```
        MOV R0, #20H  ; завдання початкової адреси області
  прапорців
        MOV R1, #10H  ; лічильник (довжина області прапорців)
LOOP:  MOV @R0, #0    ; скидання одного байта  (8 прапорців)
        INC  R0       ; перехід до наступного байта
        DJNZ  Rl, LOOP  ; цикл
```

***Приклад* 10.5.** Запам'ятати у ВПД вміст регістрів банку 1. Початкова адреса ВПД – 5000H:

```
        MOV  PSW,  #01000B ; вибір банку регістрів 1
        MOV  R0, #8       ; лічильник ← 8
        MOV  DPTR, #5000H ; визначення початкової адреси ВПД
        MOV Rl, #0                ; визначення початкової адреси РПД
  LOOP: MOV  A, @R1 ;
        MOVX  @DPTR, A ; передача з акумулятора у ВПД
        INC  R1       ; перехід до наступного регістра
        INC  DPTR  ; приріст покажчика адреси
        DJNZ  R0, LOOP ; R0 = R0 – 1,  якщо R0 > 0, то повторити
```

***Приклад* 10.6.** Звернення до пам'яті програм. Часто необхідно мати в пам'яті програм таблиці готових рішень. Для можливості роботи з такими таблицями, що зберігаються в РПП і ВПП, є спеціальні команди звернення до пам'яті програм – MOVC. Пояснимо використання цих команд на такому прикладі. Необхідно скласти підпрограму обчислення синуса кута X (X змінюється в межах від 0 до 89° з дискретністю 1°). Найшвидше обчислення функції можна одержати шляхом вибірки готового значення синуса з таблиці. Така таблиця для діапазону 0 – 89° займе 90 байтів при похибці 0,4%. Кожен байт таблиці містить дробову частину двійкового подання синуса.

Початковим параметром для підпрограми служить значення кута X, що знаходиться в акумуляторі.

Такий алгоритм отримання даних з таблиці може застосовуватися при розташуванні таблиці безпосередньо за фрагментом програми, який використовує таблицю. При цьому розмір таблиці не може бути більше 255 елементів. Якщо таблиця розташована в довільному місці пам'яті програм, то для отримання даних необхідно використовувати команду MOVC A, @A + + DPTR. У цьому випадку адреса елемента таблиці визначається сумою значень регістрів (A + DPTR).

Розглянемо два приклади отримання даних на прикладі підпрограми, що використовує таблицю значень синуса кута.

Таблиця розміщена в довільному місці пам'яті програм, розмір таблиці не більш 256 елементів.

```
   ; Обчислення  sin(x) за таблицею значень
   ; вхід:  A < номер елемента таблиці в межах від 0 до 255
   ; вихід: A < дробова частина значення синуса
SINX:    MOV DPTR, #SINUS      ; передача в регістр-покажчик даних
адреси
                                          ; таблиці
    MOVC A, @A + DPTR  ; завантаження значення синуса з таблиці
                                          ;
         RET                              ; повернення
         . . .
; Таблиця значень синуса
SINUS:  DB  0                    ; SIN(0) = 0
         DB  00000100B    ; SIN(1) = 0.017
         DB  00001001B    ; SIN(2) = 0.035
         . . .
         DB  11111111B    ; SIN(89) = 0.999
```

Для отримання наступного значення з таблиці необхідно збільшити номер комірки (INC A).

Таблиця розміщена в перших 255 байтах пам'яті програм, розмір таблиці обмежений доступною ємністю пам'яті програм.

```
   ; Обчислення  sin(x) за таблицею значень
   ; вхід:  DPTR < номер елемента таблиці в межах від 0 до 65 535
   ; вихід: A < дробова частина значення синуса
.ORG    40H                     ; таблиця починається з адреси 40H
; Таблиця значень синуса
SINUS:  DB  0                   ; SIN(0) = 0
        DB  00000100B   ; SIN(1) = 0.017
        DB  00001001B   ; SIN(2) = 0.035
        . . .
        DB  11111111B   ; SIN(89) = 0.999
        . . .
SINX:   MOV A, #SINUS                   ; передача в акумулятор
адреси  таблиці
        MOVC A, @A + DPTR   ; завантаження значення синуса з
   таблиці
        RET                             ; повернення
```

Для отримання наступного значення з таблиці необхідно збільшити номер комірки (INC DPTR)

***Приклад* 10.7.** Операції зі стеком. Механізм доступу до стека МК51: перед завантаженням в стек вміст регістра-покажчика стека (SP) інкрементується, а після зчитування зі стека декрементується. Покажчик стека зберігає адресу останнього зайнятої комірки структури стека.

За сигналом системного скидання в SP заноситься початкове значення 07H. Перший елемент даних зберігатиметься за адресою 08H. Для двобайтових адрес молодший байт зберігається за молодшою адресою.

Перевизначити SP можна командою MOV SP, #d.

Стек може розташовуватися в будь-якому місці РПД. Він використовується для організації звернень до підпрограм. У цьому випадку в стеку зберігається адреса, на яку перейде програма після виконання підпрограми.

При обробці переривань стек може бути використаний для передачі параметрів підпрограмам і для часового зберігання вмісту

регістрів спеціальних функцій.

Підпрограма обробки переривання повинна зберегти в стеку вміст тих регістрів, які вона сама використовуватиме. Перед поверненням в перервану програму вона повинна відновити їх значення.

Підпрограма обробки зовнішнього переривання може, наприклад, мати таку структуру:

```
ORG      3       ; завдання адреси вектора
         SJMP  SUBINO          ; перехід на підпрограми  обробки
ORG      30H
SUBINO: PUSH           PSW     ; збереження в стеку PSW
         PUSH  ACC ; збереження в стеку акумулятора
         PUSH  B         ; збереження в стеку B
         PUSH  DPL       ; збереження в стеку DPL
         PUSH  DPH ; збереження в стеку DPH
         MOV PSW, #1000B ; вибір банку регістрів
TEST:
         . . .
         POP  DPH ; відновлення
         POP  DPL
         POP  B
         POP ACC
         POP PSW
         RETI ; повернення
```

В табл. 10.3 наведено зміст памяті, зайнятою стеком для випадку, якщо значення SP до переходу на виконання підпрограми обробки переривання було 1FH.

Іноді при завершенні підпрограми необхідно повернутися не в те місце програми, звідки була викликана підпрограма. Для вирішення цієї задачі використовується зміна вмісту стека. У прикладі фрагмента програми наведені умовні адреси, за якими розміщуються команди. Виклик підпрограми ST здійснювався з адреси 0001, повернення здійснюється на адресу 0003. Для повернення в інше місце програми, встановлюємо мітку. Необхідно відзначити, що мітка не займає елемента пам’яті, а, отже, її адреса збігається з адресою команди, на яку вказує мітка.

Таблиця 10.3

Зміст пам’яті, зайнятої стеком

Адреса РПД	Після входу в підпрограму обробки	Після виконання підпрограми до мітки test
26 H		DPH
25 H		DPL
24H		B
23H		ACC
22H		PSW
21H	PCH	PCH
20H	PCL	PCL
1FH		

```
0001        ACALL ST
0003        NOP
0004        NOP
0005        NOP
0006    M2:
0006        NOP
0007        NOP
. . .
ST:
        NOP    ; на SP адресу повернення 0003
        . . .  ;
        DEC SP      ; зменшуємо покажчик стека
        DEC SP      ; (видаляємо зі стека адресу повернення)
        MOV DPTR, #M2 ; завантажуємо нову адресу повернення
        PUSH DPL    ; поміщаємо його в стек
        PUSH DPH    ;
        RET         ; повертаємося за адресою мітки M2
                    ; (адреса 0006)
```

10.2.2. Приклади використання команд арифметичних операцій

***Приклад* 10.8.** Скласти два двійкових багатобайтових числа. Обидва доданки розташовуються в РПД, починаючи з молодшого байта. Початкові адреси доданків задані в R0 і R1. Формат доданків у байтах заданий в R2:

```
        CLR   C ; скидання перенесення
LOOP: MOV  A, @R0 ; завантаження в акумулятор поточного байта
                  ; першого доданка
        ADDC  A, @R1 ; складання з урахуванням перенесення
        MOV @R0, A ; збереження результату
        INC  R0
        INC  Rl
DJNZ: R2, LOOP ; цикл, якщо не всі байти підсумовані
```

***Приклад* 10.9.** Множення. Команда MUL обчислює множення двох цілих беззнакових чисел, що зберігаються в регістрах A і B. Молодша частина результату розміщується в регістрі A, а старша – в регістрі-розширювачі B. Якщо вміст регістра B дорівнює нулю, то прапорець OV скидається, інакше – встановлюється. Прапорець перенесення завжди скидається. Наприклад, якщо акумулятор містив число 200 (0C8H), а розширювач 160 (0A0H), то в результаті виконання команди MUL AB вийде результат 32 000 (7D00H). Якщо акумулятор міститиме нуль, а розширювач – 7DH, прапорець OV буде встановлений, а прапорець C – скинутий.

Нехай необхідно помножити ціле двійкове число довільного формату на константу, що займає байт, наприклад, 173. Початкове число розміщується в РПД, адреса молодшого байта знаходиться в регістрі R0. Формат числа в байтах зберігається в регістрі R1:

```
        CLR A                   ; скидання акумулятора
        XCH  A, @R0     ; завантаження множеного
LOOP3: MOV  B, #AD      ; завантаження множника (ADH = 173D)
        MUL  AB                 ; множення
        ADD A,@R0       ; одержання молодшого байта
                          ; часткового добутку
        MOV  @R0, A     ; запис молодшого байта часткового
добутку
        INC  R0         ; приріст адреси
        MOV  A, B               ; пересилка старшого байта
часткового
                          ; добутку в акумулятор
        ADDC A, #0      ; урахування біта переносу
        XCH  A, @R0     ; попереднє формування чергового
                          ; байта добутку
        DJNZ   R1, LOOP3 ;
```

Одержаний добуток розміщується на місці початкового числа і займає в РПД на один байт більше.

***Приклад* 10.10.** Для МК51 скласти програму множення двох чисел: одне число двобайтове – 1003Н, інше однобайтове – 02Н.

Схема виконання множення подана на рис. 10.1

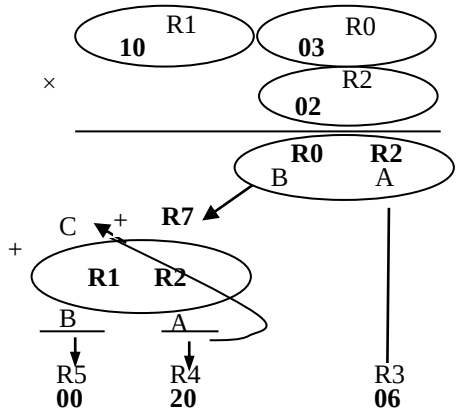


Рис. 10.1. Схема множення двобайтового числа на однобайтове

Програма множення має вигляд:

```
0000  MOV R0, #03h ; занесення мол. частини першого числа в Рг0
0002  MOV R1, #10h ; занесення ст. частини першого числа в Рг1
0004  MOV R2, #02h  ; занесення мол. частини другого числа в Рг2
0006     MOV A, R2       ; A ← R2
0007     MOV B, R0       ; B ← R0
0009     MUL  AB         ; множення R2 на R0
000A     MOV R3, A       ; запис мол. частини результату в R3
000B     MOV R7, B       ; запис ст. частини результату в R7
000D     MOV A, R2       ; A ← R2
000E     MOV B, R1       ; B ← R1
0010     MUL AB          ; множення R2 на R1
0011     ADD A, R7       ; додавання ст. частини першого добутку
                                  ; до мол. частини другого добутку
0012     MOV R4, A       ;запис результату в R4
0013     MOV A, #00      ; підготовка для збереження С
0015     ADDC A, B  ; додавання ст. частини другого добутку до С
0017     MOV R5, A       ; запис результату в R5
0018     NOP
```

Наступна програма розміщує добуток у регістри R7, R6, R5 і R4 (старші байти знаходяться в регістрах з більшими номерами):

```
        MOV   A,   R0
        MOV   B,   R2
        MUL   AB            ; добуток  молодших байтів
        MOV   R4,  A        ; у  0-й  байт  добутку
        MOV   R5,  B        ; у  1-й  байт  добутку
        MOV   A,  R1
        MOV   B,   R3
        MUL   AB            ; добуток старших байтів
        MOV   R6,  A        ; у  2-й  байт  добутку
        MOV   R7,  B        ; у  3-й  байт  добутку
        MOV   A,   R1
        MOV   B,   R2
        MUL   AB            ; добуток  старшого  байта  на  мол.  байт
        ADD   A,   R5
        MOV   R5, A         ; у  1-й  байт  добутку
        MOV   A,   R6
        ADDC  A,   B
        JNC   A1            ; перехід, якщо немає переносу в 3-й байт
        INC   R7            ; корекція 3-го байта добутку

A1:     MOV   R6, A         ; занесення у 2-й байт добутку
        MOV   A,   R0
        MOV   B,   R3
        MUL   AB            ; добуток молодшого байта на старший
        байт
        ADD   A,  R5
        MOV   R5,  A        ; занесення у 1-й  байт добутку
        MOV   A,  R6
        ADDC  A,  B
        JNC   A2            ; перехід,  якщо немає переносу в 3-й байт
        INC   R7            ; корекція 3-го байта добутку
A2:     MOV   R6, A         ; занесення у 2-й байт добутку
```

У наведеному прикладі після запису добутку молодших байтів здійснюється обчислення добутку старших, оскільки їхні результати записуються в ті байти результату, що не перекриваються. Потім до них додаються добутки старшого на молодший та молодшого на

старший. Для обчислення добутку довелося використовувати 4 команди множення, 4 команди додавання і багато операцій пересилання.

***Приклад* 10.11.** Ділення. Команда DIV здійснює ділення вмісту акумулятора на вміст регістра-розширювача. Після ділення акумулятор містить цілу частину частки, а розширювач залишок. Прапорці C і OV скидаються. При діленні на нуль встановлюється прапорець переповнення, а частка залишається невизначеною. Команда ділення може бути використана для швидкого перетворення двійкових чисел на десяткові двійково кодовані (BCD) числа.

Як приклад розглянемо програму, яка переводить двійкове число, що міститься в акумуляторі, в BCD-код. При такому перетворенні може вийти трьохрозрядне BCD-число. Старша цифра (число сотень) буде розміщена в регістрі R0, а дві молодші в акумуляторі:

```
    MOV B, #64 ; B ← 100D = 64h для обчислення сотень в числі
    DIV AB ;
    MOV R0, A ;
    XCH A, B ;
    MOV B, #A ; B ← 10D для обчислення числа десятків
    DIV AB ;
SWAP A ; розміщення числа десятків у старшій тетраді акумулятора
    ADD A, B ; додавання числа одиниць
```

Час перетворення складає 16 мкс.

10.2.3. Приклади використання команд логічних операцій

***Приклад* 10.12.** Вибрати нульовий регістровий банк:

```
ANL   P2, #10111010B  ; скидання бітів 0, 2, 6 порту 2
```

***Приклад* 10.13.** Встановити біти 0 – 3 порту 1:

```
ORL P1, #00001111B
```

***Приклад* 10.14.** Скинути біти 0, 2, 6 PSW:

```
ANL PSW, #11100111B ; скидання бітів RS1 і RS0
```

***Приклад* 10.15.** Проінвертувати біти порту P1, відповідні одиничним бітам акумулятора:

XRL P1, A ; виключне АБО порту 1 та акумулятора

***Приклад* 10.16.** Проінвертувати біти 7,6, 5 порту 0:
XRL A, #0FH ; виключне АБО акумулятора та константи

***Приклад* 10.17.** Проінвертувати біти 0 – 3 акумулятора:
XRL P0, #11100000B ; виключне АБО порту 0 та константи

***Приклад* 10.18.** Управління групою бітів порту. У РПД знаходиться масив розпакованих десяткових цифр. Вимагається передати масив зовнішньому пристрою відповідно до протоколу, пояснюваного рис. 10.2.

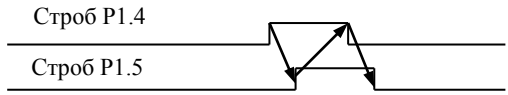


Рис. 10.2. Протокол передачі бітів порту 1

Для передачі чотирьох бітів даних використовуються молодші лінії порту 1. Лінії P1.4 і P1.5 використовуються як сигнали квитування. Передачу даних на вихід МК супроводжує сигнал на лінії P1.4. Зовнішній пристрій, прийнявши дані, повідомляє про це в МК сигналом на вході P1.5. Біти P1.6 – P1.7 в процесі передачі даних не повинні змінювати свого значення. Початковими параметрами для програми є початкова адреса масиву (R0) і довжина масиву (R1). Незадіяні в програмі біти порту 1 необхідно зберегти в незмінному вигляді:

```
        ORL P1, #00100000B ; настроювання P1.5 на
        введення
LOOP:   MOV A, @R0 ; завантаження байта в
        акумулятор
        ANL A, #00001111B ; виділення даних
        ANL P1, #11100000B ; скидання даних і строба
        ORL P1, A ; видача даних
        ORL P1, #00010000B ; видача строба
WAIT:   JNB P1.5, WAIT ; очікування відповіді
        INC R0 ; просування покажчика адреси
        ANL P1, #11101111B ; скидання строба
        DJNZ R1, LOOP ; цикл, якщо не всі дані передані
```

10.2.4. Приклади операцій з бітами

***Приклад* 10.19.** Видати вміст РгА послідовним кодом через лінію P1.0:

```
        MOV  R7, #8 ;  ініціалізація лічильника циклів
LOOP: RRC A ; привласнення прапорця перенесення значення біта A.0
        MOV P1.0, C ; передача біта
        DJNZ  R7, LOOP ; цикл, якщо не всі біти передані
```

Час виконання програми – 41 мкс, час передачі біта 5 мкс (швидкість 200 кбіт/с).

***Приклад* 10.20.** Обчислити булеву функцію трьох змінних.

Для позначення інверсії сигналу використовуватимемо підкреслення символу.

$$Y = X \wedge \underline{V} \vee W \wedge (X \vee V).$$

Змінні X, і W надходять на лінії 2, 1 і 0 порту 1 відповідно. Результат необхідно вивести на лінії 3 порту 1:

```
P1.0    .EQU    90h
P1.1    .EQU    91h
P1.2    .EQU    92h
P1.3    .EQU    93h
F0      .EQU    0D5h
Y       .EQU  P1.3
X       .EQU  P1.2
V       .EQU  P1.1
W       .EQU  P1.0
        MOV  C, X           ; введення X
        ANL C, /V ;
        MOV F0, C           ; зациклення результату в F0
        MOV C, X ; введення X
        ORL C, V  ;
        ANL C, W ;
        ORL C, F0 ;
        MOV Y, C ; виведення
```

Прапорець F0 використовується для проміжного зберігання першої кон'юнкції $X \wedge \underline{V}$. Час виконання програми складає 14 мкс.

***Приклад* 10.21.** Організувати послідовну передачу даних з

акумулятора на нульовий вивід порту 2. Передачу необхідно вести в манчестерському коді (кожен біт передається двома інтервалами: перший інтервал містить інверсію біта, інший – пряме значення):

```
        MOV  R0, #8 ; лічильник бітів
LOOP: RRC  A  ; C ← біт
        CPL   C        ; інверсія біта
        MOV P2.0, C ; передача інверсії біта
        CPL   C        ; відновлення прямого значення біта
        NOP            ; три команди NOP для вирівнювання
        NOP            ;  тривалості інтервалів
        NOP
        MOV  P2.0, C  ; передача прямого значення біта
        DJNZ  R0, LOOP ; цикл, якщо лічильник бітів ненульовий
```

Передача виконується, починаючи з молодших бітів вперед. Тривалість одного інтервалу дорівнює шести машинним циклам (6 мкс), час передачі біта дорівнює 12 мкс, час передачі байта – 96 мкс (швидкість 83 кбіт/с або 10,4 кбайт/с).

10.3. Обробка даних в МК51

***Приклад* 10.22.** Для МК51 складемо програму додавання двох двобайтових чисел A = 1A25h, B = 2A36h із запам'ятовуванням результату за адресою першого операнда:

```
MOV   R0, #25h ; запис молодшої частини першого числа в Рг0
MOV   R1, #1Ah ; запис старшої частини  першого числа в Рг1
MOV   R2, #36h ; запис молодшої частини  другого числа в Рг2
MOV   R3, #2Ah ; запис старшої частини  другого числа в Рг3
MOV   A, R0   ; молодший байт першого числа в акумуляторі
ADD   A,  R2  ; додавання молодших байтів
MOV   R0,  A  ; запам'ятовування молодшого байта суми
MOV   A,  R1  ; старший байт першого числа в акумуляторі
ADDC  A, R3 ; додавання старшого байта другого числа та C
MOV   R1,  A  ; запам'ятовування старшого байта суми
```

***Приклад* 10.23.** Для МК51 складемо програму додавання чисел.

Число $A = 428_{10} = 1AC_{16}$ знаходиться в Рг5, Рг6. Число $B = 461_{10} = 1CD_{16}$ знаходиться в РПД у комірках пам'яті $12_{10} = C_{16}$, $13_{10} = D_{16}$ із запам'ятовуванням результату за адресою другого операнда:

```
CLR                C
```

```
MOV     R5, #ACh
MOV     R6, #01
MOV     0Ch, #CDh
MOV     0Dh, #01
MOV     A, R5
ADD     A, 0Ch
MOV     0Ch, A
MOV     A, R6
ADDC    A, 0Dh
MOV     0Dh, A
```

Слід відзначити, що при виконанні команд пересилання кодів стан біта переносу не змінюється. Для віднімання потрібно здійснити аналогічні дії, але необхідно враховувати, що в системі команд і8051 відсутнім є звичайне віднімання, тому програма віднімання повинна починатися з очищення біта переносу.

***Приклад* 10.24.** Для МК51 складемо програму віднімання двох двобайтових чисел:

```
CLR     C;      ; очищення біта переносу
MOV     A, R0   ; мол. байт першого числа в акумуляторі
SUBB    A, R2   ; віднімання мол. байта другого числа
MOV     R0, A   ; запам'ятовування мол. байта різниці
MOV     A, R1   ; ст. байт першого числа в акумуляторі
SUBB    A, R3   ; віднімання ст. байта другого числа і позики
MOV     R1, A   ; запам'ятовування ст. байта різниці
```

Різниця між діями додавання і віднімання полягає ще і в тому, що при додаванні потрібно враховувати перенос, а при відніманні – позику. Але для збереження позики при відніманні використовується той же самий біт переносу. В обох прикладах використані реєстрова адресації операндів і результату.

За допомогою непрямої адресації можна написати більш компактну програму додавання чи віднімання чисел, що складаються з будь-якої кількості байтів (аби вистачило пам'яті).

***Приклад* 10.25.** Написати програму додавання двох масивів чисел, кожен з яких містить 5 байтів. Результат розмістити за адресою першого масиву.

Нехай молодший байт першого операнда записаний в комірці з ім'ям A1, а всі наступні – у наступних комірках. Аналогічним чином байти другого операнда повинні бути записані в масив, перша комірка

якого має ім'я А2. Припустимо, що ці два числа складаються з 5 байтів. Для визначення адрес пам'яті запишемо

```
A1  .EQU   50  ;перший масив починається з адреси 50 у ОЗП
A2  .EQU   55  ;другий масив починається з адреси 55 у ОЗП
```

Вважаючи, що запис операндів у ці комірки здійснений в іншій частині програми, напишемо програму додавання першого числа до другого із записом суми на місце першого числа:

```
        MOV   R0, #A1      ; запис адреси першого операнда в регістр
        MOV   Rl, #A2      ; запис адреси другого операнда в регістр
        MOV   R3, #5       ; запис кількості байтів у регістр
        CLR   C            ; очищення біта переносу
addm:   MOV   A, @R0       ; байт першого числа в акумуляторі
        ADDC  A, @R1       ; додавання байта другого операнда
        MOV   @R0, A       ; запам'ятовування байта суми
        INC   R0           ; обчислення адреси байта першого числа
        INC   R1           ; обчислення адреси байта другого числа
        DJNZ  R3, addm     ; підрахування кількості неопрацьованих
        байтів
```

Три команди на самому початку програми використовують безпосередню адресацію джерела. При цьому для двох з них транслятор здійснює підстановку фактичних значень адрес молодших байтів операндів. Команда очищення біта переносу потрібна тому, що в циклічній частині програми використовується команда додавання з урахуванням переносу. За рахунок використання непрямої адресації вдається обробити всі байти першого і другого операндів тими самими командами. А для того, щоб у наступному циклі звернутися до наступної комірки ОЗП, вміст регістрів R0 і R1 треба збільшувати на 1. Рахунок кількості неопрацьованих байтів здійснюється останньою командою. При її виконанні число в регістрі R3 зменшується на 1, і якщо результат не дорівнює 0, то управління передається на початок циклу. При показаному раніше способі для додавання 5 пар байтів довелося б використовувати 15 команд, у наведеному прикладі їх тільки 10. Крім економії пам'яті програм, цей фрагмент більш універсальний. З іншого боку, тривалість роботи цієї програми більша, тому що для її завершення потрібно виконати 34 команди замість 15. Таким чином, економія одного ресурсу, як правило, здійснюється за рахунок витрат інших.

Переходячи до двох інших арифметичних дій, слід зазначити, що множення чи ділення двійкового числа, що складається з декількох

байтів, на цілий ступінь двійки здійснюється за допомогою зсуву всіх байтів цього числа вліво чи вправо. Оскільки вага двійкової цифри 1 зростає вдвічі при переході в сусідній лівий розряд, то для множення на 2 потрібний зсув уліво на 1 розряд, для множення на 4 — на 2 розряди і так далі. Для передачі бітів коду числа з одного байта в інший потрібно використовувати операцію циклічного зсуву вліво за участю біта переносу. Оскільки перед черговим зсувом крайнього байта значення біта переносу може бути довільним, треба встановлювати потрібне значення або після завершення зсувів коректувати вміст відповідної кількості розрядів молодшого або старшого байта. Множення на попередньо задану константу, у коді якої міститься невелика кількість одиниць, також може здійснюватись послідовністю операцій зсувів і додавання. Однак у тому випадку, коли значення множника заздалегідь невідоме, потрібно використовувати операції множення.

***Приклад* 10.26.** Програма занесення у внутрішній ОЗП константи 5D3F за адресами 50, 51 (32_{16} та 33_{16}) та переміщення цієї константи в зовнішній ОЗП за адресами 31,32 ($1F_{16}$, 20_{16}):

```
  ORG 0  ; директива вказує транслятору: розташувати коди команди,
  ; яка надходить за нею, в комірках пам'яті, починаючи з адреси 0
  ALMP START ;
  ORG 100h  ; директива розміщення кодів команди, яка надходить
  ;  за нею, в комірках пам'яті, починаючи з адреси 100h
START:                    ; мітка початку програми
      MOV     32h, #5Dh
      MOV     33h, #3Fh
      MOV     DPTR, #1Fh
      MOV     A, 32h
      MOVX    @ DPTR, A
      INC     DPTR
      MOV     A, #33h
      MOVX    @ DPTR, A
```

***Приклад* 10.27.** Програма переміщення вмісту комірок пам'яті 3 ... 12 зовнішнього ОЗП в комірки пам'яті з адресами 5 ... 14 внутрішнього ОЗП:

```
  MOV A, #0A  ; A ← кількість комірок
```

```
  MOV R0, #03 ; R0 ← номер першої комірки інформації
  MOV Rl, #05; R1 ← номер першої комірки, в яку переміщується інф.
m106:  MOV   R3, A
       MOVX  A, @R0
       MOV   @R1, A
       MOV   A, R3
       INC    R0
       INC    Rl
       DEC    A
       JNZ    m106
       RET
```

***Приклад* 10.28.** Зчитування вмісту портів P1, P2 і занесення зчитаних даних у внутрішній ОЗП за адресами 31, 32:

```
MOV          31, P1
MOV          32, P2
```

***Приклад* 10.29.** Обчислення значень елементів масиву $Z_j = X_i \& Y_j$ i = l ,..., 9. Елементи масивів X_i, та Y_j знаходяться у внутрішньому ОЗП, починаючи з адрес 5 та 35 відповідно. Елементи масиву Z_j розмістити в зовнішньому ОЗП, починаючи з адреси 0. Програма має вигляд:

```
       MOV R0, #35        ; в R0 початкова адреса масиву Y
       MOV R1, #5         ; в R1 ← початкову адресу масиву X
       MOV DPTR, #0       ; "онулення" початкової адреси масиву Z
       MOV R7, #9  ; кількість операцій кон'юнкцій (елементів масиву)
M1:   MOV A, @R1
       ANL A, @R0
       MOVX @DPTR, A
       INC R0
       INC R1
       INC DPTR
       DJNZ R7, M1
```

10.4. Використання переривань у МК51

Перериванням називається виконання певних команд не у порядку виконання програми, а за деякою подією, яку називають запитом на переривання. Обробка запиту на переривання відкладається до завершення поточної команди програми, що переривається. Потім

проводиться перевірка можливості виконання переривання. Якщо даний вид переривання дозволений, то перевіряється пріоритет запиту на переривання. Якщо запит прийшов під час обробки переривання з вищим пріоритетом, то його виконання відкладається. Управління пріоритетами переривань і дозволом переривань проводиться програмно. Команди переривання не входять в набір команд мікроконтролера. Логічно вони є аналогами команди дальнього переходу. Виконання запиту на переривання починається із запам'ятовування в стеку адреси тієї команди, виконання якої відкладається для обробки переривання. Потім управління передається за адресою, де зберігається підпрограма обробки з відповідним видом переривання. А у зв'язку з тим, що місця в пам'яті для зберігання цієї підпрограми недостатньо, то за даною адресою зберігається тільки адреса переходу до підпрограми обробки переривання.

Мікроконтролери сімейства I8051 працюють не менше ніж з 5-ма видами переривань. Два види переривання проводяться за зовнішніми сигналах (INT0, INT1), два – по переповнюванню лічильників (T/C0, T/C1) і один – по завершенню прийому або передачі байта через послідовний порт. Старші моделі мікроконтролерів мають більше видів переривань. Кожному перериванню відводиться своя початкова адреса в ПЗП, яка називається вектором переривання:

0000h ; перша команда, виконувана після включення
0003h ; перша команда обробки переривання по INT0
000Bh ; перша команда обробки переривання по T/C0
0013h ; перша команда обробки переривання по INT1
00lBh ; перша команда обробки переривання по T/C1
0023h ; перша команда обробки переривання по послідовн. порту

Відведений на ці команди адресний простір (3 байти для ініціалізації і по 8 байтів для обробки переривань) використовується тільки для запису команд передачі управління, тому що воно є дуже малим для розміщення відповідних блоків програми. Програма обробки переривання повинна закінчуватися командою повернення з переривання. За цією командою здійснюється повернення до виконання перерваної програми.

Дозвіл на обробку переривань здійснюється установкою в 1 відповідних бітів регістра дозволу переривання IE (Interrupt Enable):

IE.7 = EA (Enable All) всі переривання;
IE.4 = ES (Enable Serial) переривання по послідовному каналу;
IE.3 = ET1 (Enable Timer 1) переривання по таймеру 1;

IE.2 = EX1 (Enable external 1) переривання по зовнішньому входу 1;

IE.1 = ET0 (Enable Timer 0) переривання по таймеру 0;

IE.0 = EX0 (Enable external 0) переривання по зовнішньому входу 0.

Запис нуля в старший біт цього регістра забороняє обробку всіх переривань, а одиниця в цьому біті дозволяє перевірку запитів у бітах регістра, що залишилися. За інших рівних умов найвищий пріоритет має переривання з меншим номером біта.

У разі потреби можна розділити пріоритети всіх сигналів переривання на дві групи за допомогою запису одиниць і нулів у відповідні біти регістра пріоритетів переривання IP (InterruptPriority):

IP.4 = PS (Priority Serial) пріоритет послідовного каналу;

IP.3 = PT1 (Priority Timer 1) пріоритет таймера 1;

IP.2 = PX1 (Priority external 1) пріоритет зовнішнього входу 1;

IP.l = PT0 (Priority Timer 0) пріоритет таймера 0;

IP.0 = PX0 (Priority external 0) пріоритет зовнішнього входу 0.

Група з одиницями в бітах пріоритету перевіряється в першу чергу, а група з нулями – у другу; причому всередині групи перевага надається перериванню з меншим номером біта. У старших моделях для управління дозволами переривань і їх пріоритетом використовуються біти в тих же або в додаткових регістрах.

Синхронізація роботи загальної програми можлива як від внутрішнього так і від зовнішнього сигналу.

При об'єднанні програм, які вирішують приватні завдання в загальну програму, необхідно синхронізувати їх роботу. Для цього потрібно розробити програму, що синхронізує, яка повинна забезпечити визначені послідовність і періодичність виконання програм.

Існує **два види синхронізації**. У **першому випадку** необхідно повторювати вирішення приватної задачі з певним періодом. У другому **випадку** приватне завдання повинне розв'язуватися одноразово для кожного із зовнішніх запитів, що ініціюють її виконання.

Синхронізуюча програма "нарізає" відрізки часу і надає їх програмам, які вирішуютьтермінові завдання. Після розв’язання термінової задачі сума відрізків часу, що залишилися, надається фоновим програмам. При використанні більш ніж одного виду переривання управління дозволами переривань і призначеннями пріоритетів ускладнюється у зв'язку з можливістю конфліктів і

необхідністю обслуговування черг переривань.

При розробці синхронізуючої програми потрібно скласти розклад черговості розв'язання термінових задач з розподілом їх по часу роботи цієї програми. Слід враховувати ситуації, коли запити від інших пристроїв можуть прийти в одному і тому ж циклі переривання. Якщо час розв'язання перевищує тривалість часу, що відведено, то потрібно або перерозподілити розв'язання задач між фазами, або розробити програми розв'язання задач з великою швидкодією, або використовувати більш швидкодіючий процесор.

Необхідно забезпечити завершення фонового завдання до появи наступного запиту. Інакше необхідно створювати чергу обслуговування запитів, що ще більш ускладнить синхронізуючу програму.

Програму синхронізації не можна розглядати у відриві від **програми ініціалізації**, яка повинна бути виконана при включенні пристрою. Програма ініціалізації записує початкову інформацію у функціональні регістри процесора і в комірки ОЗП, задає режими роботи зовнішніх пристроїв, а також (у разі потреби) зчитує інформацію з незалежного ЗП. Тільки після цього можна перейти до виконання програми синхронізації, що забезпечує розподіл часу між терміновими і фоновими завданнями. Блок обробки періодичних переривань повинен у першу чергу забезпечити своєчасне переривання для наступного циклу. Також необхідно виконати термінові завдання чергового циклу, після закінчення яких потрібно повернутися до розв'язання фонових задач. Після ініціалізації управління повинно передаватися фоновій частині програми.

Наведемо приклад розміщення зазначених блоків програми у разі використання зовнішнього сигналу синхронізації. Синхронізація перериванням від зовнішнього сигналу може здійснюватися за допомогою підключення до входу INT0 імпульсного сигналу потрібної частоти з двома рівнями, відповідними логічним 0 і 1 сигналів мікроконтролера. У простому випадку він може бути сформований від напруги мережі (частотою 50 Гц) або від двонапівперіодного випрямляча до фільтрації напруги (100 Гц). Такі частоти переривань прийнятні для багатьох застосувань мікроконтролерів, хоча через перешкоди в мережі тривалість інтервалу між перериваннями є нестабільною. Якщо тактова частота мікроконтролера задана кварцем на 12 МГц, то в першому випадку за один цикл синхронізації з тривалістю 20 мс може бути виконано 20 000 коротких команд (1

команда = 1 мкс), а в другому за один цикл з тривалістю 10 мс – 10 000.

Для запиту переривання по переходу вхідного сигналу з 1 в 0 необхідно записати одиницю в біт TCON.0. Блоки програми можуть бути розташовані таким чином:

```
        .CODE
        .ORG    00h     ; адреса першої виконуваної команди
        JMP     init    ; перехід на блок ініціалізації
        .ORG    03h     ; адреса команди для переривання по INTO
        JMP     main    ; перехід на основну частину програми
        .ORG    30h     ; адреса блоку ініціалізації
init:                   ; початок блоку ініціалізації
                ; запис кодів у порти мікроконтролера
                ; установка режимів роботи зовнішніх пристроїв
                ; запис кодів у функціональні  регістри
                ; зчитування початкових даних з незалежного ЗП
                ; запис початкових значень комірки
        SETB    TCON.0  ; установка режиму переривань по INT0
        MOV     IE, #01h ; дозвіл переривань по INT0
                ; кінець блоку ініціалізації
        bckg:           ; початок фонової частини програми
                ; виконання  фонових  завдань
        SJMP  bckg      ; нескінченний  цикл
main:           ; початок основної частини програми
        CLR   EA        ; заборона переривань
                ; збереження даних фонового завдання
                ; виконання термінових завдань
                ; відновлення даних фонового завдання
        SETB  EA        ; дозвіл переривань
        RETI            ; повернення до фонової частини програми
```

При синхронізації перериванням від зовнішнього сигналу підготовка до наступного циклу не потрібна. На початку програми обробки переривань необхідно зберегти вміст всіх тих функціональних регістрів, які використовуються і основною, і фоновою частинами програми. Перед виходом з основної програми потрібно відновити вміст цих регістрів.

Більш широкі можливості надають таймери-лічильники.

10.5. Завдання до самостійного дослідження простих операцій в МК

1. *Мета*

– навчитися працювати з програмою-імітатором МК-51;

– поглибити і закріпити знання з архітектури МК-51 і навички з його програмування;

– набути практичних навичок у складанні, налагодженні та виконанні програм, написаних мовою Асемблера для програмування МК-51.

2. *Теми для попереднього опрацювання*

Перед виконанням лабораторної роботи необхідно вивчити програмну модель і систему команд мови Асемблера МК-51.

Вивчити основні відомості про роботу програми-імітатора МК-51, функціональні можливості та режими роботи програми-імітатора, способи програмування ПЗП МК і введення даних у регістри МК.

Відповідно до варіантів завдань скласти програми мовою Асемблера. Для полегшення введення програм у ПЗП при виконанні лабораторної роботи на ПЕОМ бажано вказати початкові адреси всіх команд і адреси використовуваних у програмі міток, з огляду на реальний адресний простір. Якщо передбачається програмування ПЗП у машинних кодах, то необхідно перевести складені програми в ці коди. Накреслити **алгоритми** виконання програм.

3. *Постановка задачі*

Закріплення теоретичного матеріалу, дослідження особливостей виконання окремих команд, набуття практичних навичок запису та відлагодження простих програм на мові Асемблера.

4. *Завдання до самостійних досліджень*

4.1. Програмування виконання простих операцій в МК

– програмування операцій додавання та віднімання операндів;

– програмування операцій множення операндів;

– програмування операцій ділення операндів.

5. *Варіанти завдань*

Навести алгоритм виконання операції, номери комірок та їх зміст.

5.1. Програмування виконання простих операцій в МК

Номер варіанта вибирається відповідно до останньої цифри номера за журнальним списком.

10.5.1. Програмування операцій додавання та віднімання операндів

Варіант 1. Додати два числа. Число A = 750_{10} знаходиться в R0, R1 число B – у R2, R3. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 0000h.

Варіант 2. Додати два числа. Число A = 273_{10} знаходиться в R2, R3 число B – у РПД у комірках пам'яті: 31_{10}, 32_{10}. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 0002h.

Варіант 3. Додати два числа. Число A = 275_{10} знаходиться в РПД у комірках пам'яті з адресами 33_{10}, 34_{10} число B – у РПД у комірках пам'яті з адресами 35_{10}, 36_{10}. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 0004h.

Варіант 4. Додати два числа. Число A = 348_{10} знаходиться в РПД у комірках з адресами 35_{10}, 36_{10} число B – у РПД у комірках пам'яті з адресами 37_{10}, 38_{10}. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 0006h.

Варіант 5. Додати два числа. Число A = 128_{10} знаходиться в РПД у комірках пам'яті з адресами 40_{10}, 41_{10}, число B – у зовнішньому ОЗП у комірках пам'яті з адресами 05_{10}, 06_{10} Результат розташувати в зовнішньому ОЗП, починаючи з адреси 07h.

Варіант 6. Відняти два числа. Число A знаходиться у R4, R5 число B – у R6, R7. Результат розташувати у РПД, починаючи з адреси 21h.

Варіант 7. Відняти два числа. Число A =373_{10} знаходиться в R0, R1 число B – у РПД у комірках пам'яті з адресами 42_{10}, 43_{10}. Результат розташувати в R4, R5.

Варіант 8. Відняти два числа. Число A = 375_{10} знаходиться в РПД у комірках пам'яті з адресами 21h, 22h, число B – у РПД у комірках пам'яті з адресами 40_{10}, 41_{10}. Результат розташувати в зовнішньому ОЗП за адресою 0010h.

Варіант 9. Відняти два числа. Число A = 448_{10} знаходиться в РПД у комірках пам'яті з адресами 42_{10}, 43_{10}, число B – у РПД у комірках пам'яті з адресами 2Ch, 2Dh. Результат розташувати в зовнішньому ОЗП за адресою 000Ah.

Варіант 10. Відняти два числа. Число A = 148_{10} знаходиться в РПД у комірках пам'яті з адресами 44_{10}, 45_{10}, число B – у РПД у

комірках пам'яті з адресами 40h, 41h. Результат розташувати в РПД в чарунках пам'яті з адресами 42h, 43h.

Варіант 11. Додати два числа. Число А знаходиться в РПД у комірках пам'яті з адресами 30h, 31h, число В – у зовнішньому ОЗП у комірках пам'яті з адресами 00h, 01h. Результат розташувати в РПД у комірках, починаючи з адреси 32h.

Варіант 12. Додати два числа. Число А знаходиться в РПД у комірках пам'яті з адресами 30h, 31h, число В – у РПД у комірках пам'яті з адресами 32h, 33h. Результат розташувати в РПД у комірках, починаючи з адреси 34h.

Варіант 13. Відняти два числа. Число А знаходиться в РПД у комірках пам'яті з адресами 40h, 41h, число В – у РПД у комірках пам'яті з адресами 42h, 43h. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 00h.

Варіант 14. Відняти два числа. Число А знаходиться в зовнішньому ОЗП у комірках пам'яті з адресами 00h, 01h, число В – у РПД у комірках пам'яті з адресами 50h, 51h. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 02h.

Варіант 15. Відняти два числа. Число А знаходиться в РПД у комірках пам'яті з адресами 45h, 46h, число В – у R3, R4. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 10h.

Приклад. Додати два числа. Число А знаходиться в РПД у комірках пам'яті з адресами 85_{10}, 86_{10}, число В – у зовнішньому ОЗП у комірках пам'яті з адресами 09_{10}, 10_{10}. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 0020h.

Програма має вигляд:

```
.ORG 00h  ; директива, яка вказує, що наступна команда
                    ; повинна розміщуватися з адреси 00h
                    ; (початкова адреса пуску МК-51)
 AJMP ST ; короткий безумовний перехід на першу команду програми
.ORG 30h ; адреса першої команди програми
; модуль додавання молодших частин чисел
   ST: MOV DPTR,#10     ; занесення в DPTR числа 10₁₀ = 0Ah
MOVX A,@DPTR ; зчитування числа з комірки за адресою 10 зовн. ОЗП
    ADD A,86             ; додавання числа з РгА та числа з внутр. ОЗП
    MOV DPTR,#20H; занесення в DPTR номера комірки 20h зовн. ОЗП
                         ;          для збереження РгА
    MOVX @DPTR,A ; занесення змісту РгА в комірку 20h зовн. ОЗП
```

```
; модуль додавання старших частин чисел
        MOV DPTR,#09     ;
        MOVX A,@DPTR     ;
        ADDC A,85        ;
        MOV DPTR,#21H    ;
        MOVX @DPTR,A     ;
; модуль виявлення та збереження С при додаванні ст. частих чисел
        JC M1            ;
        MOV DPTR,#22H    ;
        MOV A,#00        ;
        MOVX @DPTR,A     ;
        AJMP M2          ;
  M1: MOV DPTR,#22H      ;
        MOV A,#01        ;
        MOVX @DPTR,A     ;
  M2: NOP
```

Приклад. Додати два числа. Число $A = 448_{10} = 1C0_{16}$ знаходиться в РПД у комірках пам'яті з адресами $58_{10} = 3A_{16}$, $59_{10} = 3B_{16}$ число $B = 431_{10} = 1AF_{16}$– у R3, R4. Результат розташувати в R6, R7.

```
0000    C3          CLR     C;
0001    753AC0      MOV     3A, #C0;
0004    753D01      MOV     3B, #01;
0007    7DAF        MOV     R3, #AF;
0009    7C01        MOV     R4, #01;
000B    EB          MOV     A, R3;
000C    253A        ADD     A, 3A;
000E    FD          MOV     R5, A;
000F    EC          MOV     A, R4;
0010    353B        ADDC    A, 3B;
0012    FE          MOV     R6, A;
```

Приклад. Для МК51 складемо програму віднімання двох двобайтових чисел, якщо A > B:

```
.ORG 00h      ; початкова адреса пуску МК-51
AJMP ST ; короткий безумовний перехід на першу команду програми
.ORG 30h      ; адреса першої команди програми
ST: CLR     C;        ; очищення біта переносу
    MOV     A,  R0    ; молодший байт першого числа в акумуляторі
    SUBB    A,  R2    ; віднімання молодшого байта другого числа
```

```
MOV     R0, A     ; запам'ятовування молодшого байта різниці
MOV     A,  R1    ; старший байт першого числа в акумуляторі
SUBB    A,  R3 ; віднімання старшого байта другого числа і позики
MOV     R1,  A ; запам'ятовування старшого байта різниці
```

Приклад. Відняти два числа. Число А знаходиться в РПД у комірках пам'яті з адресами 45h, 46h, число В – у R3, R4. Результат розташувати в зовнішньому ОЗП, починаючи з адреси 10h.

Блок-схема алгоритму виконання програми наведена на рис. 10.3

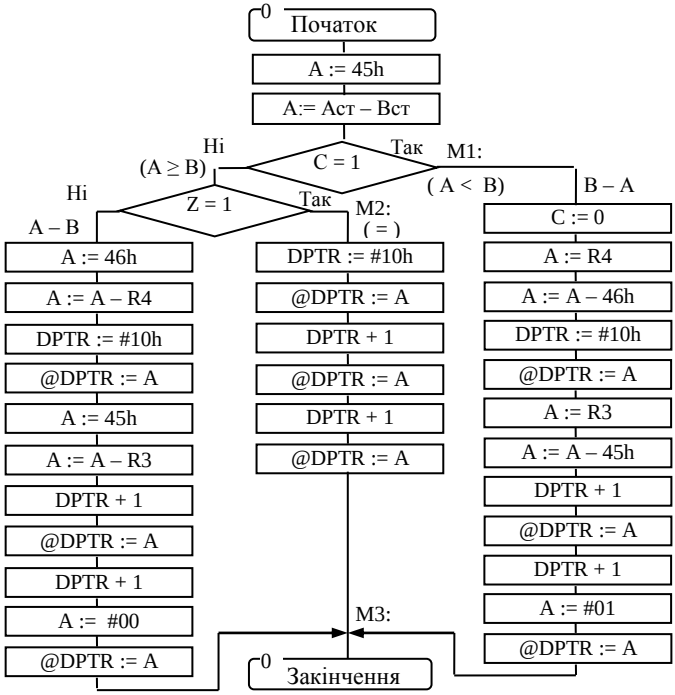


Рис. 10.3. Блок-схема алгоритму виконання прикладу

Програма на асемблері МК-51 може мати вигляд:

```
.ORG 00h ; директива, яка вказує, що наступна команда повинна
; розміщуватися  з адреси 00h (початкова адреса пуску МК-51)
AJMP ST ; короткий безумовний перехід на першу команду програми
 .ORG 30h     ; адреса першої команди програми
```

```
ST:  MOV A,45H
     SUBB A,R3
     JC M1        ; перехід на мітку M1 якщо A < B
     JZ M2        ; перехід на мітку M2 якщо A = B
M5: MOV A,46H
     SUBB A,R4
     MOV DPTR, #10H
     MOVX @DPTR,A
     MOV A,45H
     SUBB A,R3
     INC DPTR
     MOVX @DPTR,A
     INC DPTR             ; збільшення номера комірки зовн. пам'яті
     MOV A,#00            ; занесення знака результату в A
     MOVX @DPTR,A    ; збереження знаку результату в комірці 12h
     AJMP M3
M1: CLR C ; очищення біта переносу
     MOV A,R4
     SUBB A,46H
     MOV DPTR,#10H
     MOVX @DPTR,A
     MOV A,R3
     SUBB A,45H
     INC DPTR
     MOVX @DPTR,A
     INC DPTR             ; збільшення номера комірки зовн. пам'яті
     MOV A,#01            ; занесення знаку результату в A
     MOVX @DPTR,A    ; збереження знаку результату в комірці 12h
     AJMP M3
; запис нулів в комірки 10h, 11h, 12h
M2: MOV DPTR,#10H
     MOVX @DPTR,A
     INC DPTR             ; збільшення DPTR – номера 11h в DPTR
    MOVX @DPTR,A; занесення в зовн. комірку 11h  числа з регістра A
     INC DPTR
     MOVX @DPTR,A
M3: NOP          ; END
```

10.5.2. Програмування операцій множення операндів

Варіант 1. Скласти програму реалізації C = A × B. Число A = = 4100_{10}, число B – знаходяться в регістровій пам'яті. Результат розташувати в зовнішньому ОЗП МК51 за адресою 000Ch.

Варіант 2. Скласти програму реалізації C = A × B. Число A = = 4105_{10} знаходиться в регістровій пам'яті, число B – у зовнішньому ОЗП за адресою 0000h. Результат розташувати в зовнішньому ОЗП МК51 за адресою 000Ch.

Варіант 3. Скласти програму реалізації C = A × B. Число A = = 5100_{10}, число B – знаходяться в регістровій пам'яті. Результат розташувати в РПД МК51.

Варіант 4. Скласти програму реалізації C = A × B. Число A = = 5105_{10} знаходиться в регістровій пам'яті, число B – у зовнішньому ОЗП за адресою 0002h. Результат розташувати в зовнішньому ОЗП МК51.

Варіант 5. Скласти програму реалізації C = A × B. Число A = = 4100_{10}, знаходиться в РПД, число B – у регістровій пам'яті. Результат розташувати в регістровій пам'яті МК51.

Варіант 6. Скласти програму реалізації C = A × B. Число A = = 5478_{10} знаходиться в РПД, число B – у R1. Результат розташувати в РПД МК51.

Варіант 7. Скласти програму реалізації C = A × B. Число A = = 6400_{10} знаходиться в РПД, число B – у R2. Результат розташувати в зовнішньому ОЗП МК51 за адресою 000Dh.

Варіант 8. Скласти програму реалізації C = A × B. Число A = = 6405_{10} знаходиться в РПД, число B – у комірці 0080h РПД. Результат розташувати в зовнішньому ОЗП МК51 за адресою 000Eh.

Варіант 9. Скласти програму реалізації C = A × B. Число A = = 6408_{10} знаходиться в R0, R1, число B – у R3. Результат розташувати в зовнішньому ОЗП МК51за адресою 000Eh.

Варіант 10. Скласти програму реалізації C = A × B. Число A = = 4400_{10} знаходиться в РПД, число B – у R0. Результат розташувати в зовнішньому ОЗП МК51 за адресою 000Fh.

Варіант 11. Скласти програму реалізації C = A × B. Число A = = 3078_{10} знаходиться в РПД, число B – у комірці 0081h РПД. Результат розташувати в РПД МК51.

Варіант 12. Скласти програму реалізації C = A × B. Число A = = 4097_{10} знаходиться в РПД, число B – у комірці 0085h РПД. Результат розташувати в зовнішній пам'яті МК51.

Варіант 13. Скласти програму реалізації C = A × B. Число A = = 4300_{10}, знаходиться в РПД, число B – у регістровій пам'яті. Результат розташувати в зовнішньому ОЗП МК51за адресою 0010h.

Варіант 14. Скласти програму реалізації C = A × B. Число A = = 6608_{10} знаходиться в R0, R1, число B – у R7. Результат розташувати в зовнішньому ОЗП МК51за адресою 0000h.

Варіант 15. Скласти програму реалізації C = A × B. Число A = = 5155_{10} знаходиться в регістровій пам'яті, число B – у зовнішньому ОЗП за адресою 0002h. Результат розташувати в РПД МК51, починаючи з адреси 0090h.

10.5.3. Програмування операцій ділення операндів

Варіант 1. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 02_{10} – у регістровій пам'яті. Результат розташувати в зовнішній пам'яті МК51.

Варіант 2. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 4200_{10} – у регістровій пам'яті. Результат розташувати в РПД пам'яті МК51.

Варіант 3. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 4200_{10}. – у РПД. Результат розташувати в зовнішній пам'яті МК51.

Варіант 4. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 28_{10}. – у РПД. Результат розташувати в РПД МК51.

Варіант 5. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 56_{10}. – у R0. Результат розташувати в зовнішній пам'яті МК51.

Варіант 6. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в R0, R1, число B = 43_{10}. – у R3. Результат розташувати в РПД пам'яті МК51.

Варіант 7. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 43_{10}. – у R2. Результат розташувати в зовнішній пам'яті МК51.

Варіант 8. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 23_{10}. – у R1. Результат розташувати в РПД МК51.

Варіант 9. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = 02_{10} – у регістровій пам'яті. Результат розташувати в зовнішній пам'яті МК51.

Варіант 10. Скласти програму реалізації C = A/B. Число A – двобайтове, число B = 18_{10}, знаходяться в регістровій пам'яті. Результат розташувати в РПД МК51.

Варіант 11. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B = ACh – у зовнішній пам'яті за адресою 0000h. Результат розташувати в зовнішній пам'яті в комірках, починаючи з 0009h МК51.

Варіант 12. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в зовнішньому ОЗП, число B = 2200_{10} – в регістровій пам'яті. Результат розташувати в РПД пам'яті МК51.

Варіант 13. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B – у зовнішньому ОЗП. Результат розташувати в зовнішній пам'яті МК51.

Варіант 14. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в регістровій пам'яті, число B – у РПД. Результат розташувати в РПД МК51.

Варіант 15. Скласти програму реалізації C = A/B. Число A – двобайтове, знаходиться в РПД, число B – у R7. Результат розташувати в зовнішній пам'яті МК51.

Приклад. Розділити два числа.

Команда DIV здійснює ділення вмісту акумулятора на вміст регістра-розширювача. Після ділення акумулятор містить цілу частину, а розширювач – залишок. Прапорці C і OV скидаються. При діленні на нуль встановлюється прапорець переповнення, а частка залишається невизначеною. Команда ділення може бути використана для швидкого перетворення двійкових чисел на десяткові двійково-кодовані (BCD) числа.

Як приклад розглянемо програму, яка переводить двійкове число, що міститься в акумуляторі, в BCD-код. При такому перетворенні може вийти трьохрозрядне BCD-число. Старша цифра (число сотень) буде розміщена в регістрі R0, а дві молодші в акумуляторі:

```
MOV B, #64 ; B ← 100D = 64h для обчислення сотень у числі
DIV AB ;
MOV R0, A ;
XCH A, B ;
```

MOV B, #A ; B ← 10D для обчислення кількості десятків
DIV AB ;
SWAP A ; кількость десятків у старшій тетраді акумулятора
ADD A, B ; додавання кількості одиниць
Час перетворення складає 16 мкс.

Порядок роботи в інтегрованому середовищі Pinnacle

Відлагодження початкового тексту програми виконується за допомогою інтегрованого середовища Pinnacle. Для цього необхідно виконати такі дії:

1. Запустити емулятор Pinnacle та вибрати в меню File – New. Написати програму на мові Асемблер МК-51 та зберегти її з розширенням .asm (наприклад, 7.asm). Для цього необхідно вибрати File – Save As.

2. Створити новий проект. Для цього вибрати Projekt – New. Знайти місце, попередньо записаного файлу з розширенням .asm, та вказати ім'я з новим розширенням .pmk (наприклад, 7.pmk).

3. Отредагувати проект, створений раніше. Для цього необхідно вибрати Projekt – Edit – +Include та файл з розширенням .asm. Підтвердити цей вибір.

4. Виконати компіляцію програми. Для цього необхідно вибрати Projekt – Build Projekt. У разі якщо знайдено помилки, виправити їх та знов скомпілювати програму, вибравши Projekt – Build Projekt.

5. Відкрити необхідні вікна для слідкування за виконанням програми. Для цього необхідно вибрати:

View – Registrs;
View – Data Pointer (DPTR);
View – Internal RAM;
View – External RAM;
Code Memory (Disassembly).

6. Відладити програму. Спочатку рекомендується проставити початкові дані у потрібні вузли, а потім відлагодити програму покроково. Для цього необхідно натискати на піктограму  та перевіряти зміну змісту тих вузлів, до яких у досліджуваній команді виконується звернення.

Вікна програми Pinnacle при налагодженні програми мають вигляд, наведений на рис. 10.4.

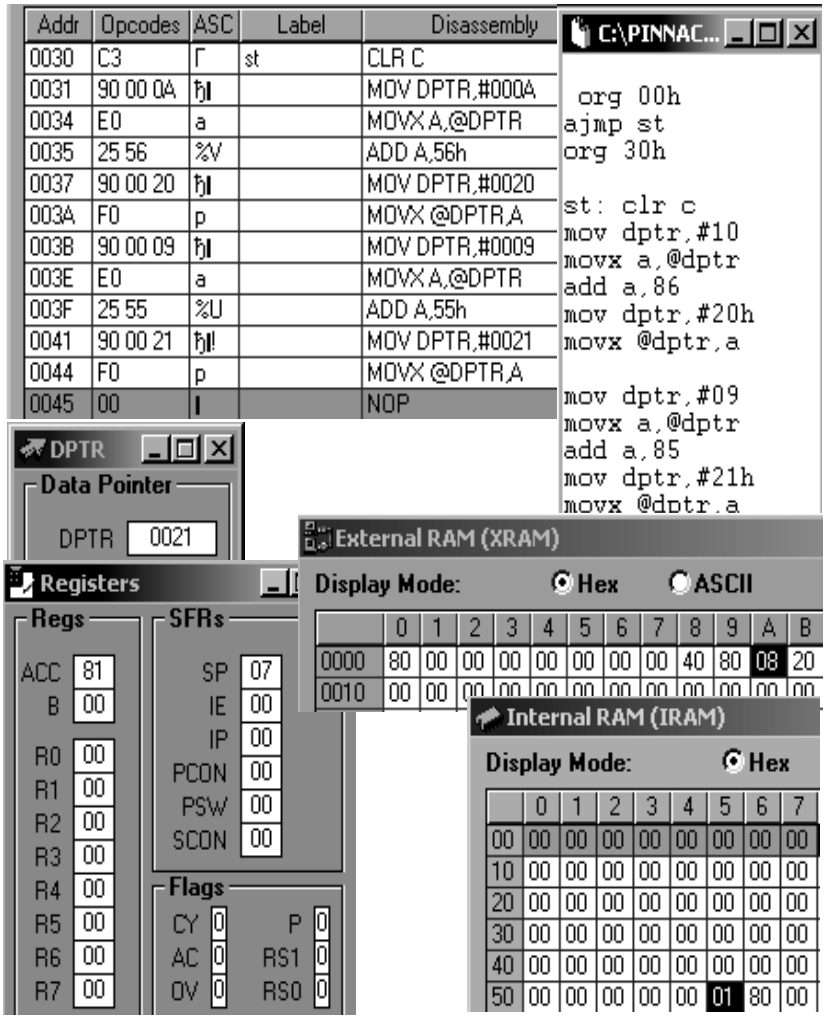


Рис. 10.4. Вікна програми Pinnacle при налагодженні програми

10.6. Завдання до самостійного дослідження команд передачі управління в МК

1. *Мета*

– поглибити і закріпити знання з архітектури МК К1816ВЕ51 і навички з його програмування;

– набути практичних навичок у складанні, налагодженні та виконанні програм, написаних мовою Асемблера для програмування МК К1816ВЕ51.

2. *Теми для попереднього опрацювання*

Перед виконанням лабораторної роботи необхідно вивчити програмну модель і систему команд мови Асемблера МК К1816ВЕ51.

Відповідно до варіантів завдань скласти програми мовою Асемблера. Для полегшення введення програм у ПЗП при виконанні лабораторної роботи на ПЕОМ бажано вказати початкові адреси всіх команд і адреси використовуваних у програмі міток, з огляду на реальний адресний простір. Якщо передбачається програмування ПЗП у машинних кодах, то необхідно перевести це програми в машинні коди. Накреслити **алгоритми** виконання програм.

3. *Постановка задачі*

Закріплення теоретичного матеріалу, дослідження особливостей виконання окремих команд, набуття практичних навичок запису та відлагодження простих програм на мові Асемблера.

4. *Завдання до самостійних досліджень*

4.1. Програмування виконання команд передачі управління в МК.

5. *Варіанти завдань*

Навести алгоритм виконання операції, номери комірок та їх зміст.

Варіант 1. У РПД, починаючи з адреси 60Н, знаходиться масив з 10 елементів. Підрахувати і зберегти в регістрах: R2 – кількість елементів масиву, якщо їх значення менше або більше 128; R3 – кількість елементів масиву, значення яких дорівнюють 128.

Варіант 2. У РПД з адреси 50Н знаходиться масив, що складається з 16 елементів. Підсумовувати елементи масиву доти, доки значення суми не перевищить 512. Видати в R3 номер елемента, на

якому відбулося переповнення. Якщо сума елементів не досягла значення 512, то видати в регістрі R3 значення 0.

Варіант 3. У РПД, починаючи з адреси 20h знаходиться масив з 6 елементів. Підрахувати і зберегти в РПД з комірки 26h кількість елементів масиву, якщо їх значення менше 0Ah.

Варіант 4. Для функції Y = 40X + 65 видати в R2 перше значення аргументу, при якому значення функції перевищить 1024. Початкове значення аргументу X = 20.

Варіант 5. У РПД, починаючи з адреси 22h знаходиться масив з 4 елементів. Підрахувати і зберегти в зовнішньому ЗП з комірки 25h кількість елементів масиву, якщо їх значення більше 0Fh.

Варіант 6. У РПД з адреси 70H знаходиться масив з 16 чисел. Елементами масиву є числа 32, 64, 96 і 128. Підрахувати і зберегти в регістрах R4 – R7 кількість повторень кожного елемента.

Варіант 7. У зовнішньому ЗП, починаючи з адреси 00h знаходиться масив з 5 елементів. Підрахувати і зберегти в РПД з комірки 20h кількість елементів масиву, якщо їх значення більше 0Bh.

Варіант 8. У РПД за адресами 60H – 6FH знаходиться масив. З адреси 30h РПД створити масив, до якого входять адреси елементів першого масиву, що дорівнюють 128. У регістрі R2 зберегти число елементів, що дорівнює 128. Перервати виконання програми, якщо буде знайдено 5 елементів зі значенням 128.

Варіант 9. У РПД з адрес 50H і 60H знаходяться 2 масиви, що складаються з 16 елементів кожен. Підрахувати кількість елементів першого масиву, що мають рівні значення в 2 масиві. Результат занести в регістр R2.

Варіант 10. Для функції Y = 40X + 10 одержати перше значення, що перевищує 512, починаючи з X = 1. Значення аргументу записати в R4, функції – у R5, R6.

Варіант 11. У РПД, починаючи з адреси 65H, знаходиться масив з 10 елементів. Одержати в регістрі R3 кількість елементів, значення яких дорівнює 55H. Рахунок перервати, якщо кількість елементів перевищить 3.

Варіант 12. Для функції 15X + 85 знайти перше значення аргументу, при якому молодший байт функції дорівнює 155, та розмістити його в зовнішній пам’яті за адресою 00h.

Варіант 13. У зовнішньому ОЗП, починаючи з адреси 30H, знаходиться масив з 15 чисел. Елементами масиву є числа 10, 20, 30 і

180. Підрахувати і зберегти в регістрах R4 – R7 кількість повторень кожного елемента.

Варіант 14. У РПД, починаючи з адреси 20H, знаходиться масив з 5 елементів. Підрахувати і зберегти в регістрах: R3 – кількість елементів масиву, якщо їх значення більше 10_{10}; R4 – кількість елементів масиву в інших випадках.

Варіант 15. У РПД, починаючи з адреси 2Ah, знаходиться масив з 4 елементів. Підрахувати і зберегти в зовнішньому ЗП з комірки 10h кількість елементів масиву, якщо їх значення більше або дорівнює 08.

Приклад. У РПД, починаючи з адреси 10H, знаходиться масив з 5 елементів. Підрахувати і зберегти в регістрах: R1 – кількість елементів масиву, якщо їх значення менше 64_{16}; R2 – кількість елементів масиву, значення яких дорівнюють 64_{16}; R3 – кількість елементів масиву, значення яких більше 64_{16}.

Блок-схема виконання прикладу наведена на рис. 10.5.

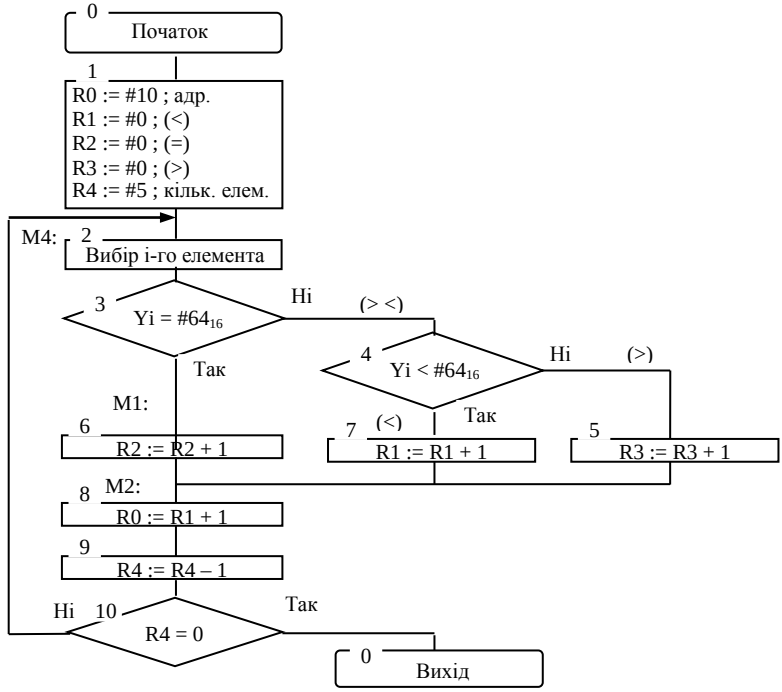


Рис. 10.5. Блок-схема виконання прикладу

Програма на асемблері МК51 може мати вигляд:

```
.ORG 00h  ; директива, яка вказує, що наступна команда
          ; повинна розміщуватися з адреси 00h
          ; (початкова адреса пуску МК-51)
 AJMP ST ; короткий безумовний перехід на першу команду програми
 .ORG 30h     ; адреса першої команди програми
ST:
      MOV R0, #10      ; (вершина 1)
      MOV R1, #0       ; (вершина 1)
      MOV R2, #0       ; (вершина 1)
      MOV R3, #0       ; (вершина 1)
      MOV R4, #5       ; (вершина 1)
M4: MOV A, @R0         ; (вершина 2)
      SUBB A, #40      ; (вершина 3)
      JZ M1            ; (вершина 3)
      JC M3            ; (вершина 4)
      INC R3           ; (вершина 5)
      AJMP M2          ; (вершина 5)
M1:  INC R2            ; (вершина 6)
      AJMP M2          ; (вершина 6)
M3:  INC R1            ; (вершина 7)
M2:  INC R0            ; (вершина 8)
      DEC R4           ; (вершина 9)
      JNZ M4           ; (вершина 10)
      NOP              ; (вершина 0) (END)
```

Приклад. Для функції Y = 20X + 22 видати в R2 перше значення аргументу, при якому значення функції перевищить 1024. Початкове значення аргументу X = 50.

Блок-схема виконання прикладу наведена на рис. 10.6.

Програма на асемблері МК51 може мати вигляд:

```
      .ORG 00h  ; директива, яка вказує, що наступна команда
                ; повинна розміщуватися з адреси 00h
                ; (початкова адреса пуску МК-51)
AJMP ST  ; короткий безумовний перехід на першу команду програми
      .ORG 30h   ; адреса першої команди програми
ST:
       MOV R2, #00    ; (1) Підготовка R2
```

```
      MOV R7, #32    ; (1) 32H = 50D
M5:  MOV A, #14      ; (2) 14H = 20D
      MOV B, R7      ; (2)
      MUL AB         ; (2)
      ADD A, #16     ; (2) 16H = 22D
      JC M1          ; (2)
      AJMP M2        ; (2)
M1:  INC B           ; (2)
M2:  MOV R1, A ; (3) Перезбереження молодшої частини результату
      MOV A, B       ; (3)
      SUBB A, #04 ; (3) Віднімання зі ст. частини рез. ст. частини const
      JC M3          ; (3)
      MOV A, R1      ; (4)
      SUBB A, #00    ; (4) Віднімання молодшої частини const = 00h
      JC M3          ; (4)
      AJMP M4        ; (4)
M3:   INC R7         ; (5)
      AJMP M5        ; (5)
M4:  MOV A, R7       ; (6)
      MOV R2, A      ; (6)
      NOP            ; (0)
```

При налагоджуванні програми на емуляторі необхідно замість міток у програмі підставити номери адрес комірок пам'яті. В рядку після номера комірки пам'яті емулятор підставить коди команд, як це наведено нижче:

```
0000  7A00      MOV R2, #00     ; (1)
0002  7F32      MOV R7, #32     ; (1) 32H = 50D
0004  7414      MOV A, #14      ; (2) 14H = 20D
0006  8FF0      MOV B, R7       ; (2)
0008  A4        MUL AB          ; (2)
0009  2416      ADD A, #16      ; (2) 16H = 22D
000B  4002      JC 000F     ; M1 ; (2)
000D  0111      AJMP 0011 ; M2 ; (2)
000F  05F0 M1:  INC B           ; (2)
0011  F9    M2: MOV R1, A       ; (3)
0012  E5F0      MOV A, B        ; (3)
0014  9404      SUBB A, #04     ; (3)
0016  40EB      JC  001F    ; M3 ; (3)
```

```
0018  E9        MOV A, R1        ; (4)
0019  9400      SUBB A, #00      ; (4)
001B  4002      JC 001F     ; M3 ; (4)
001D  0121      AJMP M4          ; (4)
001F  0F    M3: INC R7           ; (5)
0020  0104      AJMP 0004 ; M5 ; (5)
0022  EF    M4: MOV A, R7        ; (6)
0023  FA        MOV R2, A        ; (6)
0024  00        NOP              ; (0)
```

У результаті виконання цієї програми R2 = 34_{16} .

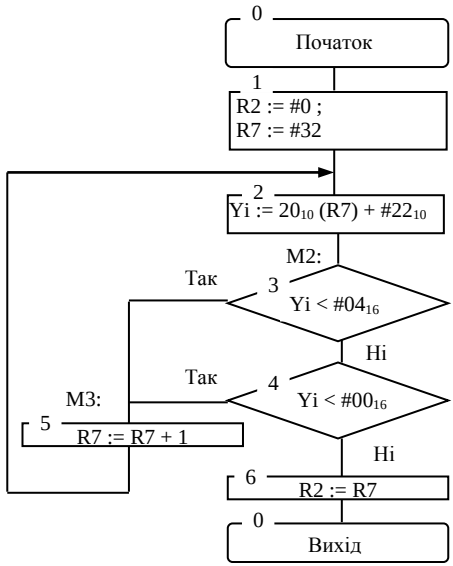


Рис. 10.6. Блок-схема виконання прикладу

Після запам’ятовування програми в пам’яті вона буде мати такий вигляд:

```
00000000:  7A 00 7F 32 74 14 8F F0|A4 24 16 40 02 01 11 05
00000010:  F0 F9 E5 F0 94 04 40 07|E9 94 00 40 02 01 22 0F
00000020:  01 04 EF FA 00
```

10.7. Завдання до самостійного дослідження організації звернень до масивів

1. *Мета*

– поглибити і закріпити знання з архітектури МК К1816ВЕ51 і навички з його програмування;

– набути практичних навичок у складанні, налагодженні та виконанні програм, написаних мовою Асемблера для програмування МК К1816ВЕ51.

2. *Теми для попереднього опрацювання*

Перед виконанням лабораторної роботи необхідно вивчити програмну модель і систему команд мови Асемблера МК К1816ВЕ51.

Відповідно до варіантів завдань скласти програми мовою Асемблера. Для полегшення введення програм у ПЗП при виконанні лабораторної роботи на ПЕОМ бажано вказати початкові адреси всіх команд і адреси використовуваних у програмі міток, з огляду на реальний адресний простір. Якщо передбачається програмування ПЗП у машинних кодах, то необхідно перевести складені програми у ці коди. Накреслити **алгоритми** виконання програм.

3. *Постановка задачі*

Закріплення теоретичного матеріалу, дослідження особливостей виконання окремих команд, набуття практичних навичок запису та відлагодження простих програм на мові Асемблера.

4. *Завдання до самостійних досліджень*

4.1. Програмування виконання функцій в МК.

5. *Варіанти завдань*

Навести алгоритм виконання операції, номери комірок та їх зміст.

Варіант 1. Задані масиви A і B у РПД по N = 11 елементів. Навести алгоритм та програму формування масиву C за правилом: якщо $A_i > B_i$, то $C_i = A_i + B_i$; якщо $A \le B$ то $C_i = A_i - B_i$, та розмістити результати виконання програми в РПД.

Варіант 2. Задано масив A в РПД з N = 12 елементів. Навести алгоритм та програму формування масиву B у РПД з перших 6 позитивних елементів масиву A.

Варіант 3. Задано масив A в РПД з N = 13 елементів. Навести алгоритм та програму формування масиву B у зовнішній пам'яті з комірки 00h з перших 7 позитивних елементів масиву A.

Варіант 4. Задано масив A в РПД з N = 10 елементів. Навести алгоритм та програму формування масиву B в РПД з елементів масиву A, які задовольняють умові $A_i > E$ при E = 5.

Варіант 5. Задано масив A в РПД з N = 11 елементів. Навести алгоритм та програму визначення суми та кількості елементів масиву A, які задовольняють умові $A_i > E$ при E = 10, та розмістити результати виконання програми в зовнішній пам'яті з комірки 10h.

Варіант 6. Задані масиви A і B у РПД по N = 6 елементів. Навести алгоритм та програму визначення кількості пар елементів, які задовольняють умові $A_i > B_i$, та розмістити результати виконання програми в зовнішній пам'яті з комірки 12h.

Варіант 7. Задані масиви A і B у РПД по N = 5 елементів. Навести алгоритм та програму визначення кількості елементів Ai, які задовольняють умові $A_i \leq B$, та розмістити результати виконання програми в РПД.

Варіант 8. Задані масиви A і B у РПД по N = 6 елементів. Навести алгоритм та програму визначення максимального з елементів масиву A, які задовольняють умові $A_i < B$, та розмістити результати виконання програми в зовнішній пам'яті з комірки 14h.

Варіант 9. Задані масиви A і B у РПД по N = 7 елементів. Навести алгоритм та програму сформування масиву C за правилом: якщо $A_i + B_i > 0$, то $C_j = B_j$, та розмістити результати виконання програми в зовнішній пам'яті з комірки 10h.

Варіант 10. Задано масив A у РПД з N = 16 елементів. Навести алгоритм та програму визначення кількості елементів масиву A, які задовольняють умові $L < A_i \leq M$, де L = 3, M = 15, та розмістити результати виконання програми в зовнішній пам'яті з комірки 08h.

Варіант 11. Задано масив A у РПД з N = 8 елементів. Навести алгоритм та програму визначення мінімального з позитивних елементів масиву A та розмістити результати виконання програми в РПД.

Варіант 12. Задано масив A у РПД з N = 9 елементів. Навести алгоритм та програму визначення мінімального з негативних елементів масиву A та розмістити результати виконання програми в зовнішній пам'яті з комірки 04h.

Варіант 13. Задано масив A у РПД з N = 8 елементів. Структура масиву A: X1,Y1;X2,Y2; … . Навести алгоритм та програму визначення

кількості пар, для яких виконується умова $X_i > Y_i$, та розмістити результати виконання програми в зовнішній пам'яті з комірки 06h.

Варіант 14. Задані масиви A і B у РПД по N = 30 елементів. Навести алгоритм та програму визначення суми та кількості елементів A_i, які задовольняють умові $A_i > B_i$, та розмістити результати виконання програми в РПД.

Варіант 15. Задано масив A у РПД з N = 10 елементів. Структура масиву A: $X_1,Y_1;X_2,Y_2;$ … X_n,Y_n. Навести алгоритм та програму визначення кількості пар, для яких виконується умова $X_i \le Y_i$, та розмістити результати виконання програми в РПД.

Варіант 16. Задано масив A у РПД з N = 12 елементів. Структура масиву A: $X_1,Y_1;X_2,Y_2;$ … X_n,Y_n. Навести алгоритм та програму визначення кількості пар, для яких виконується умова $X_i = Y_i$, та розмістити результати виконання програми в зовнішній пам'яті з комірки 08h.

Варіант 17. Задано масив A у РПД з N = 14 елементів. Структура масиву A: $X_1,Y_1;X_2,Y_2;$ … X_n,Y_n. Навести алгоритм та програму визначення кількості пар, для яких виконується умова $X_i < Y_i$, та розмістити результати виконання програми в зовнішній пам'яті з комірки 12h.

Варіант 18. Задано масив A у РПД з N = 8 елементів. Навести алгоритм та програму визначення суми елементів масиву A, для яких біти 2 та 10 збігаються, та розмістити результати виконання програми в зовнішній пам'яті з комірки 16h.

Варіант 19. Задано масив A у РПД з N = 6 елементів. Навести алгоритм та програму визначення суми елементів масиву A, для яких біти 3 та 12 збігаються, та розмістити результати виконання програми в РПД.

Варіант 20. Задані масиви A і B у РПД по N = 4 елементи. Навести алгоритм та програму формування масиву C в РПД за правилом: якщо у елементів A_i та B_i біти 4 та 9 збігаються, то $C_i = A_i + B_i$.

Варіант 21. Задано масив A у РПД з N = 7 елементів. Навести алгоритм та програму формування масиву B з елементів масиву A, у яких 0, 2 та 5 біти мають нулі, та розмістити результати виконання програми в зовнішній пам'яті з комірки 20h.

Варіант 22. Задані масиви A і B у РПД по N = 6 елементів. Навести алгоритм та програму визначення кількості пар елементів, які

задовольняють умові $A_i > B_i$, та розмістити результати виконання програми в зовнішній пам'яті з комірки 30h.

Варіант 23. Задані масиви А і В у РПД по N = 7 елементів. Навести алгоритм та програму формування масиву С за правилом: якщо $A_i + B_i > 0$, то $C_j = B_j$, та розмістити результати виконання програми в РПД.

Варіант 24. Задано масив А у РПД з N = 10 елементів. Структура масиву А: $X_1,Y_1;X_2,Y_2;$ … X_n,Y_n. Навести алгоритм та програму визначення кількості пар, для яких виконується умова $X_i \le Y_i$,та розмістити результати виконання програми в зовнішній пам'яті з комірки 00h.

Варіант 25. Задано масив А в зовнішній пам'яті з N = 6 елементів. Структура масиву А: $X_1,Y_1;X_2,Y_2;$ … X_n,Y_n. Навести алгоритм та програму визначення кількості пар, для яких виконується умова $X_i < Y_i$, та розмістити результати виконання програми в РПД з комірки 30h.

Приклад. Задано масив А в РПД з N = 8 елементів, починаючи з адреси 30h. Навести алгоритм та програму визначення кількості елементів масиву А, у яких біт 0 має нулі.

Блок-схема алгоритму виконання програми наведена на рис. 10.7.

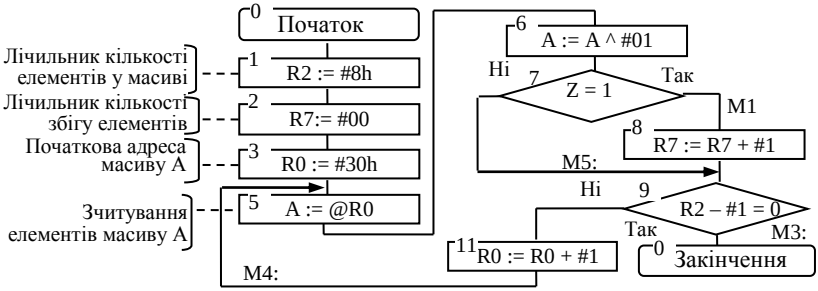


Рис. 10.7. Алгоритм виконання прикладу

Програма на асемблері МК-51 може мати вигляд:

```
ORG 00h ; директива, яка вказує, що наступна команда
        ; повинна розміщуватися з адреси 00h
AJMP ST ; короткий безумовний перехід на першу команду програми
ORG 30H         ; адреса початку розміщення програми
 ST: MOV R2,#8H
     MOV R7,#0
```

```
      MOV R0,#30H
M4: MOV A,@R0
      ANL A,#01
      JZ M1
      JMP M5
 M1: INC R7          ; збільшення лічильника кількості збігу елементів
 M5: DJNZ R2,M2 ; віднімання 1 та перехід на M2, якщо R2 ≠ 0
      AJMP M3
 M2: INC R0         ; підготовка номера наступної адреси масиву A
       JMP M4
 M3: NOP            ; вихід із програми
```

Приклад. Задані масиви A та B у РПД по 4 елементи. Масив A розміщується з адреси 0030H, масив B – з адреси 0034H. Навести алгоритм та програму формування масиву C у РПД з адреси 0040H за правилом: якщо $A_i < B_i$, то $C_i = A_i$, інакше – $C_i = B_i$.

Блок-схема виконання прикладу наведена на рис. 10.8.

Програма на асемблері МК-51 може мати вигляд:

```
      ORG 00h     ; директива, яка вказує, що наступна команда
                  ; повинна розміщуватися з адреси 00h
      AJMP ST     ; безумовна передача управління на мітку ST
      ORG 30H     ; адреса початку розміщення програми
ST:  MOV R0,#30H
      MOV R1,#34H
      MOV R2,#40H
      MOV R3,#4
M1: MOV A,@R0
      SUBB A,@R1
      JC M2

      ; модуль передачі змісту комірок пам'яті, адреса яких вказана в R0,
                                  ; у комірки  пам'яті, адреса яких указана в R2
      MOV A,R1
      MOV R6,A
      MOV A,@R1
      MOV R7,A
      MOV A,R2
      MOV R1,A
```

```
        MOV A,R7
        MOV @R1,A
        MOV A,R6
        MOV R1,A
        AJMP M3
      ; модуль передачі змісту комірок пам'яті, адреса яких указана в R1,
                                  ; у комірки пам'яті, адреса яких вказана в R2

M2: MOV A,R0
        MOV R6,A
        MOV A,@R0
        MOV R7,A
        MOV A,R2
        MOV R0,A
        MOV A,R7
        MOV @R0,A
        MOV A,R6
        MOV R0,A
        AJMP M3

M3: INC R0
        INC R1
        INC R2
        DJNZ R3,M1
        NOP
```

При виконанні програми, якщо в комірках пам'яті, починаючи з адреси 30h, знаходяться числа 02, 03, 04, 05, а з адреси 34h – числа 01, 04, 02, 08, то починаючи з адреси 40h, будуть знаходитися числа 01, 03, 02, 05.

Приклад. У зовнішній пам'яті, починаючи з адреси 02H, знаходиться масив з 5 елементів. Підрахувати і зберегти в РПД у комірці 60h кількість елементів масиву, якщо їх значення менше або більше 10.

Блок-схема алгоритму виконання основної частини прикладу наведена на рис. 10.9.

Рис. 10.8. Блок-схема виконання прикладу

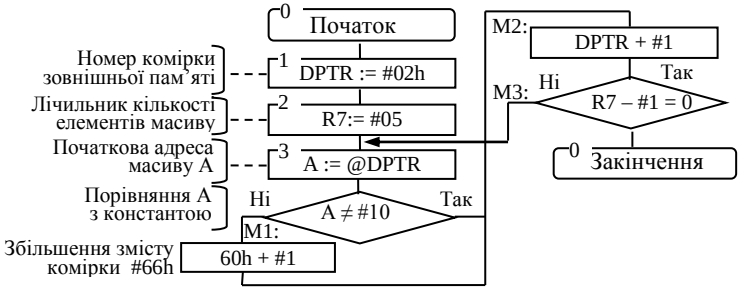


Рис. 10.9. Алгоритм виконання прикладу

Програма на асемблері МК-51 може мати вигляд:

```
    ORG 00H
    AJMP ST
    ORG 30H
ST: MOV DPTR,#02
    MOV R7,#5
M3: MOVX A,@DPTR
    CJNE A,#10,M1       ;  bitC := A – #10
    AJMP M2
M1: INC 60H
M2: INC DPTR
    DJNZ R7,M3
    NOP             ; END
```

Приклад. У зовнішній пам'яті, починаючи з адреси 02H, знаходиться масив з 14 елементів. Підрахувати і зберегти в РПД у комірці 68h кількість елементів масиву, якщо їх значення більше або дорівнює 10.

Блок-схема алгоритму виконання основної частини прикладу наведена на рис. 10.10.

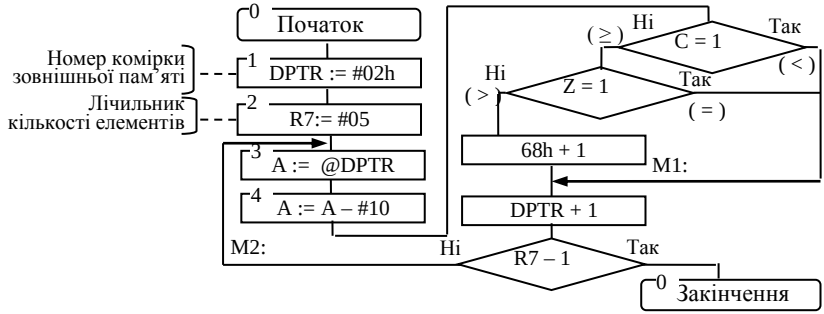


Рис. 10.10. Блок-схема алгоритму виконання прикладу

Програма на асемблері МК-51 може мати вигляд:

```
    ORG 00H
    AJMP ST
    ORG 30H
ST: MOV DPTR,#02
    MOV R7,#5
M2: MOVX A,@DPTR
```

```
    SUBB A,#10
    JC M1
    JZ M1
    INC 68H
M1: INC DPTR
    DJNZ R7,M2
    NOP          ; END
```

10.8. Завдання до самостійного дослідження типових обчислювальних процедур в МК

1. *Мета*

– поглибити і закріпити знання з архітектури МК К1816ВЕ51 і навички з його програмування;

– набути практичних навичок у складанні, налагодженні та виконанні програм, написаних мовою Асемблера для програмування МК К1816ВЕ51.

2. *Теми для попереднього опрацювання*

Перед виконанням лабораторної роботи необхідно вивчити програмну модель і систему команд мови Асемблера МК К1816ВЕ51.

Відповідно до варіантів завдань скласти програми мовою Асемблера. Для полегшення введення програм у ПЗП при виконанні лабораторної роботи на ПЕОМ бажано вказати початкові адреси всіх команд і адреси використовуваних у програмі міток, з огляду на реальний адресний простір. Якщо передбачається програмування ПЗП у машинних кодах, то необхідно перевести складені програми в машинні коди. Накреслити **алгоритми** виконання програм.

3. *Постановка задачі*

Закріплення теоретичного матеріалу, дослідження особливостей виконання окремих команд, набуття практичних навичок запису та відлагодження простих програм на мові Асемблера.

4. *Завдання до самостійного дослідження*

4.1. Програмування виконання типових обчислювальних процедур у МК.

5. *Варіанти завдань*

Навести алгоритм виконання операції, номери комірок та їх зміст.

Написати алгоритм та програму на мові асемблера МК51 формування часової затримки тривалістю N. Період проходження тактових імпульсів – 1 мкс. Тривалість затримки вибрати згідно з номером варіанта, залежно від номера за журнальним списком групи (табл. 10.4).

Таблиця 10.4

№ з/п	T мС	№ з/п	T мС	№ з/п	T мС	№ з/п	T мС	№ з/п	T мС
1	10	14	475	27	800	40	1 400	53	2 700
2	50	15	500	28	825	41	1 500	54	2 800
3	75	16	525	29	850	42	1 600	55	2 900
4	125	17	550	30	900	43	1 700	56	3 000
5	150	18	575	31	925	44	1 800	57	3 500
6	175	19	600	32	950	45	1 900	58	4 000
7	225	20	625	33	975	46	2 000	59	4 500
8	325	21	650	34	1 100	47	2 100	60	5 000
9	350	22	675	35	1 150	48	2 200	61	6 000
10	375	23	700	36	1 175	49	2 300	62	7 000
11	400	24	725	37	1 200	50	2 400	63	8 000
12	425	25	750	38	1 250	51	2 500	64	9 000
13	450	26	775	39	1 300	52	2 600	65	10 000

Приклад. Написати програму затримки часу виконання команд із $t_з = 200$ мкс.

Переведемо десяткове число 200 в шістнадцяткове: $200_{10} = C8_{16}$.

Відомо, що в МК i8051 час виконання одного такту – 1 мкс.

Час виконання однократно виконуваних команд:

```
ACALL TIME2        ; 2 цикли  – 2 мкс
MOV R3, X          ; 1 цикл – 1 мкс
```

Час виконання команд циклу:

```
DJNZ R3, M1 ; 2 цикли  – 2 мкс
RET                ; 2 цикли  – 2 мкс
```

Підставивши у формулу $t_з = t_{ов} + t_ц X$ необхідні значення, отримаємо

$$200 \text{ мкс} = 5 \text{ мкс} + 2 X.$$

Визначаємо необхідну кількість тактів затримки виконання циклу (значення X):

$$X = (200 - 5)/ 2 = 97{,}5 \text{ такту.}$$

Щоб з меншою похибкою отримати кількість тактів циклу, необхідно додати пусту операцію NOP, яка виконується за 1 цикл. Таким чином, з урахуванням часу виконання пустої команди NOP, число X буде дорівнювати

$$X = (200 - 6)/ 2 = 97,$$

що в шістнадцятковій системі числення дорівнює 61h.

Таким чином, програма затримки часу виконання команд із $t_з$ = 200 мкс буде мати вигляд:

```
        ACALL TIME2     ; виклик підпрограми TIME2
        …
TIME2:  MOV R3, #61H    ; завантаження R3 константою 61

M1:  DJNZ R3, M1 ; декремент лічильника (R3): R3 ← R3 – 1
                 ; та продовження рахунку, якщо (R3) ≠ 0
        NOP              ; пуста команда  – 1 мкс
        RET              ; повернення  із підпрограми
```

Часова затримка великої тривалості. Схема алгоритму програмної реалізації часової затримки великої тривалості методом вкладених циклів показана на рис. 10.11.

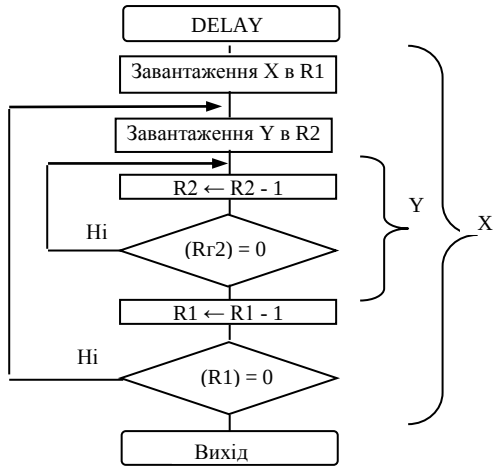


Рис. 10.11. Блок-схема процедури часової затримки великої тривалості

Числа X і Y вибираються в цьому випадку зі співвідношення $T = n_{ti} + n_{tj} + X(n_{tk} + n_{tg} + n_{ts}) + n_{tl}$, де T – реалізований часовий інтервал.

Запис на асемблері підпрограми, що реалізує алгоритм із вкладеними циклами на основі регістрів BX, CX, має вигляд:

```
DELAY1: MOV R1, #X ; завантажити лічильник зовн. циклів – 1 мкс
   L1:    MOV R2, #Y ; завантажити лічильник внутр. циклів – 1 мкс
   L2:   ...
         ...                ; часова добавка,
         DJNZ R2, L2  ; декремент лічильника внутрішніх циклів
               ; та повернутися у внутрішній цикл, якщо (C) ≠ 0  – 2 мкс
         DJNZ R1, L1   ; декремент лічильника зовнішніх циклів
               ; та повернутися у зовнішній цикл, якщо (B) ≠ 0  – 2 мкс
         RET      ; повернення   – 2 мкс
```

Загальний час затримки, внесений програмою, що реалізує алгоритм із вкладеними циклами, визначається виразом

$$t_з = t_{ок} + (t_{ц1} + t_{ц2}\,Y)X,$$

де $t_{ок}$ – час реалізації однократно виконуваних команд;

$t_{ц1}$, $t_{ц2}$ – час реалізації команд зовнішнього і внутрішнього циклів;

X, Y – параметри початкової установки лічильників зовнішнього і внутрішнього циклів.

Алгоритми програмної затримки без корекції тривалості будуються без використання спеціальних операторів часової добавки. Це призводить до спрощення програми, однак не дозволяє в загальному випадку сформувати затримку з необхідною точністю.

Приклад. Написати програму займу часу виконання команд із $t_з$ = 1 мс = 1000 мкс з використанням двох вкладених циклів.

Щляхом підбору значень чисел X2 та X1 обчислюється час затримки виконання підпрограми:

$$t_з = t_{ок} + (t_{ц1} + t_{ц2}\,Y)X =$$

$$= 2_{(CALL)} + 1_{(MOV\ R1)} + 2_{(RET)} + (2_{(DJNZ\ R1)} + 1_{(MOV\ R2)} + 2_{(DJNZ\ R2)} \times 18)25 =$$

980 мкс.

Таким чином, програма затримки часу виконання команд із $t_з$ = 1 мс з використанням двох вкладених циклів буде мати вигляд:

```
.ORG 00h ; початкова адреса пуску МК-51
AJMP ST ; короткий безумовний перехід на першу команду
 програми
```

```
    .ORG 30h ; адреса першої команди програми
ST:          ACALL DELAY1 ; виклик підпрограми DELAY1  – 2 мкс
DELAY1: MOV R1, #12H ; R1 константою 12₁₆ = 18₁₀   – 1 мкс
     M2:    MOV R2, #19H ; R2 константою 19₁₆ = 25₁₀   – 1 мкс
     M1:    DJNZ R2, M1    ; R2 ← R2 – 1 та продовжувати лічбу,
                                ; якщо (R2) ≠ 0   – 2 мкс
            DJNZ R1, M2    ; R1 ← R1 – 1 та продовжувати лічбу,
                                ; якщо (R1) ≠ 0   – 2 мкс
               ; затримка 20 мкс
               MOV R7, #9H      ; 1 мкс
     M3:       DJNZ R7, M3      ; 2 × 9 = 18 мкс
               NOP              ; 1 мкс
               RET              ; повернення з підпрограми   – 2 мкс
```

Приклад. Написати програму затримки часу виконання команд із $t_з = 100$ мс = 100 000 мкс.

Одним із варіантом вирішення такої задачі є можливість використання декілька разів програми затримки тривалості, наприклад 1 мс. Один із варіантів блок-схеми процедури часової затримки 100 мс наведений на рис. 10.12.

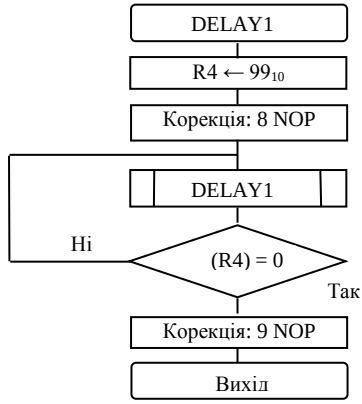


Рис. 10.12. Блок-схема процедури часової затримки 100 мс

Час виконання програми затримки обчислюється так:

$$t_з = t_{ов} + t_ц X1 =$$

$= 1_{(MOV\ R4)} + [\ 1000_{(CALL)} + 8(_{NOP}) + 2_{(JNZ\ M4)}\ ]\ 99 + 9(_{NOP}) = 100\ 000$ мкс.

Програма затримки часу виконання команд із $t_з = 100$ мс може мати вигляд:

```
.ORG 00h ; початкова адреса пуску МК-51
AJMP ST ; короткий безумовний перехід на першу команду програми
.ORG 30h ; адреса першої команди програми
T100MC:   MOV R4, #63H ; – 1 мкс
              ; корекція
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
     M4:  ACALL DELAY1 ; виклик підпрограми DELAY1  – 2 мкс
DELAY1:  MOV R1, #12H ; R1 константою 12₁₆ = 18₁₀   – 1 мкс
     M2:   MOV R2, #19H ; R2 константою 19₁₆ = 25₁₀   – 1 мкс
     M1:    DJNZ R2, M1    ; R2 ← R2 – 1 та продовжувати лічбу,
                                ; якщо (R2) ≠ 0   – 2 мкс
          DJNZ R1, M2     ; R1 ← R1 – 1 та продовжувати лічбу,
                                ; якщо (R1) ≠ 0   – 2 мкс
          ;затримка 20 мкс
          MOV R7, #9H          ; 1 мкс
M3:       DJNZ R7, M3          ; 2 × 9 = 18 мкс
          NOP                  ; 1 мкс
          RET                  ; повернення  із підпрограми    – 2 мкс
          DJNZ R4, M4
              ; корекція
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
          NOP
```

11. НАПІВПРОВІДНИКОВІ ЗАПАМ'ЯТОВУЮЧІ ПРИСТРОЇ

11.1. ВІС ЗП. ОСНОВНІ ПОНЯТТЯ

Пам'яттю ПК називається сукупність пристроїв, що служать для запам'ятовування, збереження і видачі інформації, представленої двійковим кодом. Окремі пристрої, що входять у цю сукупність, називаються ***запам'ятовуючими пристроями*** того або іншого типу.

Продуктивність і обчислювальні можливості ЕОМ у значній мірі визначаються складом і характеристиками її ЗП. У складі ЕОМ використовується одночасно кілька типів ЗП, що відрізняються принципом дії, характеристиками і призначенням.

11.1.1. Класифікація БІС ЗП

Класифікацію БІС ЗП можна провести за такими ознаками:

1. За типом елемента пам'яті (елементарної комірки пам'яті ЗП):
- ЗП на МОП-транзисторах;
- ЗП на біполярних транзисторах.

2. За принципом збереження інформації:
- статичні;
- динамічні.

Статичні елементи здатні зберігати інформацію як завгодно довго, поки подається електроживлення. Основою побудови і збереження одного біта інформації є тригер.

Динамічні запам'ятовуючі елементи здатні зберігати інформацію протягом короткого часу. Тому для збереження інформації її потрібно періодично обновляти (регенерувати). Як динамічний елемент, що зберігає біт інформації, можна скористатися зарядженим конденсатором. Використання динамічних елементів призводить до

спрощення схем, зниження споживаної потужності, а іноді і підвищення швидкості роботи.

3. За способом розміщення і пошуку інформації:
- адресні ЗП;
- безадресні ЗП.

У пам'яті з ***адресною організацією*** розміщення і пошук інформації засновані на використанні адреси збереження слова (числа, команди і т.д.). Адресою служить номер комірки пам'яті, у якій це слово розміщається. Адресні ЗП поділяються на:

1) пам'ять з довільним доступом;
2) пам'ять з послідовним доступом.

У ЗП з ***довільним доступом*** (RAM – Random Access Memory) час, що витрачається на пошук заданого слова, не залежить від його номера.

Термін "***послідовний доступ"*** відноситься до запам'ятовуючих пристроїв, у яких позиції слів стають доступними для читання або запису тільки у визначеному порядку. У ЗП з послідовним доступом кожне слово, що зберігається, не прив'язується до конкретних запам'ятовуючих елементів, а скоріше до свого положення щодо інших слів, що зберігаються. Час доступу залежить від передісторії процесу запису і зчитування інформації (де знаходиться головка і де шуканий блок, наприклад, на магнітному барабані, дискеті, диску).

Безадресні ЗП поділяються на:

1) стекові ЗП;
2) асоціативні ЗП.

Стекову пам'ять можна розглядати як сукупність осередків, що утворюють одновимірний масив, у якому сусідні осередки зв'язані один з одним розрядними ланцюгами передачі слів. Запис нового слова здійснюється у верхній осередок (осередок 0), при цьому всі раніше записані слова (включаючи слово, що знаходилось в осередку 0) зрушуються вниз, у сусідні осередки з більшими на одиницю номерами. Зчитування можливе тільки з верхньої (нульової) комірки пам'яті, при цьому, якщо здійснюється зчитування з видаленням, всі інші слова в пам'яті зміщуються нагору, у сусідні осередки з більшими

номерами. У цій пам'яті порядок зчитування слів відповідає правилу: останній надійшов – першим обслуговується.

У ***пам'яті асоціативного типу*** пошук потрібної інформації здійснюється не за адресою, а за її вмістом (за асоціативною ознакою). При цьому пошук за асоціативною ознакою (або послідовно за окремими розрядами цієї ознаки) відбувається паралельно в часі для всіх комірок пам'яті.

4. За функціональним призначенням всі типи ЗП можна розділити на такі рівні функціональної ієрархії:

- оперативні ЗП (RAM – Random Access Memory);
- постійні ЗП (ROM – Read-Only Memory).

Оперативна пам'ять служить для збереження інформації (даних програм, проміжних і кінцевих результатів обробки), безпосередньо використовуваної в процесі виконання операцій в АЛП й ПУ процесора.

ОЗП поділяються на:

- статичні ОЗП;
- динамічні ОЗП.

Постійні ЗП призначені для тривалого збереження незмінної в процесі роботи ЕОМ інформації (програм, мікропрограм, констант). ПЗП поділяються на

- програмувальні при виготовленні (масочні ПЗП);
- програмовані і репрограмовані.

11.1.2. Основні характеристики ЗП

1. ***Інформаційна ємність*** ЗП – визначається максимально можливою кількістю бітів збереженої інформації. Для порівняння різних типів ЗП зручно використовувати *питому ємність* – відношення ємності ЗП до її фізичного об'єму.
2. ***Організація ЗП*** – добуток числа збережених слів на їхню розрядність. Як бачимо, це дає інформаційну ємність ЗП, однак при одній і тій же інформаційній ємності організація ЗП може бути різною, так що організація є самостійним важливим параметром.
3. ***Швидкодія пам'яті*** визначається тривалістю операції звертання, тобто часом, витраченим на пошук потрібної одиниці

інформації в пам'яті і на її зчитування, або часом на пошук місця в пам'яті, що призначається для збереження даної одиниці інформації, і на її запис у пам'ять.

4. ***Час зчитування*** – інтервал між моментами появи сигналу читання і слова на виході ЗП.

5. ***Час запису*** – інтервал після появи сигналу запису, достатній для встановлення ЗЕ в стан, що задається вхідним словом.

6. ***Час звертання.*** Мінімально припустимий інтервал між послідовними читаннями або записами утворює відповідний цикл. Тривалість циклу (ще називають часом звертання) може перевищувати час читання або запису, тому що після цих операцій може знадобитися час для відновлення необхідного початкового стану ЗП.

7. ***Час вибірки*** – час від подачі сигналу запису або читання до завершення відповідної операції.

8. ***Ширина вибірки*** визначається кількістю інформації, записаної в ЗП або зчитування з нього за одне звертання.

9. ***Швидкість обміну*** інформацією між ЗП та ін. пристроями (визначається числом бітів (байтів), переданих за одиницю часу).

10. ***Енергоспоживання і потужність***, що розсіюється.

11. ***Показник питомої вартості ЗП.*** Застосовується для оцінки економічних характеристик ЗП. Питома вартість ЗП визначається відношенням його вартості до інформаційної ємності, тобто вартістю біта збереженої інформації.

12. ***Надійність*** ЗП, як і будь-якого іншого пристрою, оцінюється імовірністю збереження основних параметрів у заданих межах протягом визначеного проміжку часу при роботі в заданих умовах.

13. ***Габаритно-вагові характеристики ЗП.***

Перераховані вище параметри – традиційні. Для деяких сучасних ЗП вони повинні бути доповнені новими. Причиною є більш складний характер доступу до збереженим даних, коли звертання до першого слова деякої групи слів (пакета) вимагає більшого часу, ніж звертання до наступних. Для таких режимів уводять параметр *часу доступу при першому звертанні* (Latency) і *темпу передач* для наступних слів пакета (Bandwidth). Темп передач у свою чергу оцінюється двома значеннями – *граничним* (усередині пакета) і *усередненим*. Зі зменшенням пакета усереднений темп знижується, все більш відрізняючись від граничного.

11.2. ПОСТІЙНІ ЗАПАМ'ЯТОВУЮЧІ ПРИСТРОЇ (ПЗП)

11.2.1. Архітектура ПЗП

У комп'ютерах і різних цифрових пристроях пам'ять часто служить джерелом інформації, що залишається незмінною (наприклад, списки констант, таблиці перетворення даних, постійні програми і т.п.). У таких випадках використовуються модулі пам'яті, у яких записану інформацію неможливо змінити засобами самої системи, що використовує модуль. Такі модулі називаються постійними ЗП (ПЗП, англ. ROM – Read-Only Memory).

На даний час термін «ПЗП» вживається також стосовно блоків пам'яті з можливістю перепрограмування (таких, як EPROM, EEPROM, Flash та ін.). Більш точною узагальнюючою назвою цього класу пристроїв є «енергонезалежна пам'ять», однак ми надалі будемо використовувати термін «ПЗП».

Для вибірки збереженого слова даних із ПЗП необхідно виконати такі операції:

1) сформувати адресу на вході ПЗП;

2) подати сигнал дозволу на ПЗП (у даному випадку це сигнал Chip Select (CS) – вибір кристала).

Тоді на виході ми одержуємо відповідне збережене слово.

На рис. 11.1 наведена схема внутрішнього пристрою ROM IC 4752.

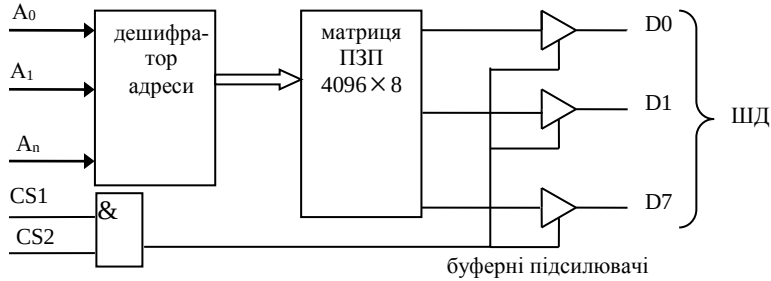


Рис. 11.1. Внутрішній пристрій ROM IC 4752 (4 кбайти)

У даному випадку в дешифратора адреси 12 входів і 2^{12} = 4096 виходів. На виходах кристала розміщені т.зв. трьохстабільні буфери (тобто вони можуть знаходитись у трьох станах: логічний «0», логічна

«1» і в z-стані – стані високого вихідного опору). Коли кристал не обраний, вихідні буфери знаходяться в z-стані, тобто відключені від шини даних. Це необхідно при підключенні декількох пристроїв до загальної шини.

Розглянемо спрощену архітектуру ПЗП (рис. 11.2).

Основними частинами ПЗП є: адресні дешифратори (рядків і стовпців), матриця запам'ятовуючих регістрів (у даному прикладі 4 × 4), вихідний буфер.

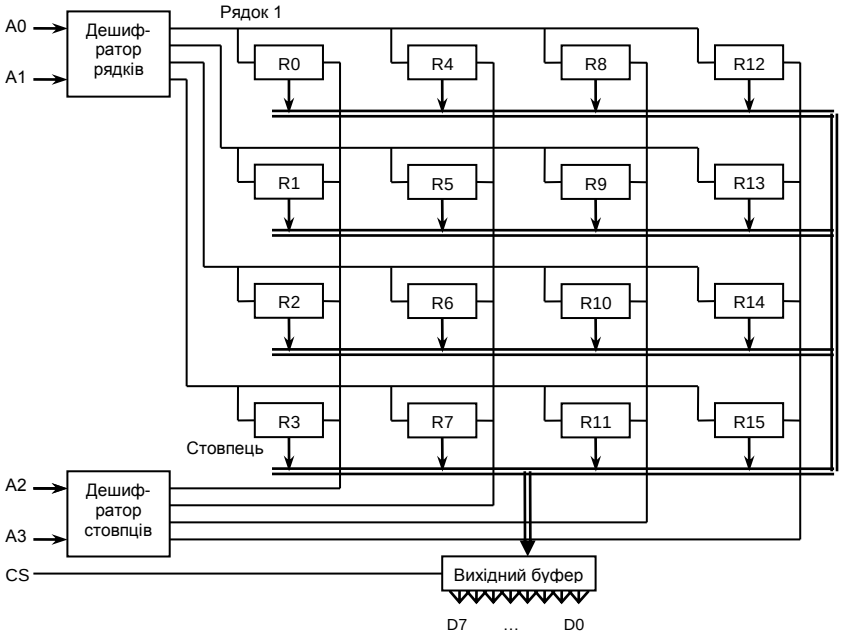


Рис. 11.2. Схема спрощеної архітектури ПЗП

Матриця запам'ятовуючих регістрів забезпечує збереження відповідної інформації, записаної в ПЗП. Кожен осередок зберігає кількість бітів, що дорівнює розрядності слова (у нашому випадку 8). Номер регістра строго фіксований і визначається відповідним перетинанням рядка і стовпця. Кожен регістр має два дозволяючих входи, управляємих адресними дешифраторами. Високий рівень

сигналу на обох входах дозволяє регістру видачу вмісту на внутрішню шину даних.

Адресні дешифратори призначені для визначення тієї комірки пам'яті, яка повинна видати вміст ПЗП за необхідною адресою на шину даних. Вся адреса розбивається на частину розрядів, що визначають рядок, і частину розрядів, що визначають стовпець у матриці запам'ятовуючих осередків. Відповідно перший дешифратор здійснює вибірку по рядках, другий – по стовпцях.

Вихідний буфер призначений для видачі на зовнішню шину даних вмісту осередку за умови, що управляючий сигнал CS має високий рівень потенціалу, інакше вихідний буфер буде мати високий вихідний опір (знаходитись в z-стані). Вхід CS може бути одним з розрядів ША процесора.

Примітка. Матриця запам'ятовуючих елементів не завжди квадратна, а, як правило, прямокутна. Наприклад, ROM I2708 зберігає 1024 восьмирозрядні слова, тобто має розмірність матриці запам'ятовуючих елементів 64 × 16.

11.2.2. Часова діаграма роботи ПЗП

Розглянемо часову діаграму роботи ПЗП (операція читання) (рис. 11.3).

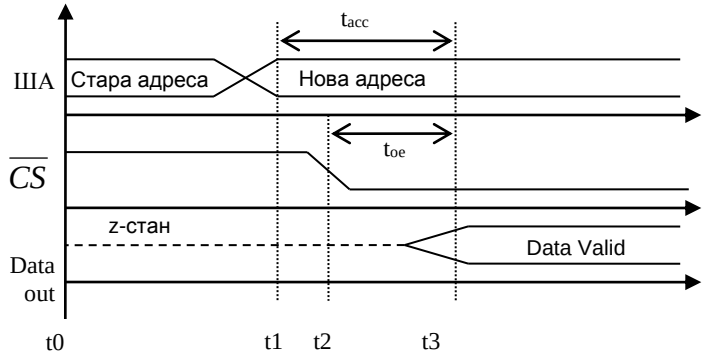


Рис. 11.3. Часова діаграма роботи ПЗП

На інтервалі [t_0, t_1] адреси на ША змінюються на необхідні, відповідно до виконуваної операції читання і, починаючи з моменту часу t_1, ця адреса буде вірною. З цього моменту внутрішні дешифратори ПЗП починають декодувати входи. На момент часу t_2 активізується управляючий вхід, що має вміст "дозвіл", щоб відповідне значення надійшло у вихідний буфер. Виходи починають змінюватися із z-стану в необхідний (0 або 1) у залежності від збережених даних, і на момент часу t_3 на виході з'являються доступні для читання дані (Data Valid). Звідси час доступу (t_{acc} – Access Time) визначається як різниця $t_3 - t_1$ (звичайно від 10 до 30 нс у біполярних ПЗП, 200 – 300 нс – для МОП, 100 – 900 нс – для канальної логіки). Часовий параметр, обумовлений різницею $t_3 - t_2$ (t_{OE} – Output Enable Time), визначає ту затримку часу, що вносить схема ПЗП, коли управляючий сигнал встановлений у значення, що відповідає стану «Вибір кристала дозволений», а виходи ще не одержали потрібне слово (біполярні ПЗП – порядку 20 нс, МОП – порядку 100 нс). Інтервал $t_2 - t_1$ визначає часову затримку між встановленням вірної адреси і переводом сигналу CS в L.

11.2.3. Типи ПЗП

Ознакою класифікації є спосіб запису (програмування) ПЗП.

11.2.3.1. Масочні (звичайні) ПЗП (англ. MROM – Masked ROM)

Це історично перший тип ПЗП. Інформація в таку пам'ять заноситься в процесі виготовлення кристала і надалі не може змінюватись.

На підставі елементної бази поділяються на:

- резисторні ПЗП – якщо на перетинанні рядка і стовпця стояв резистор, то це відповідало логічній одиниці, якщо резистора не було – логічному нулю;
- ємнісні ПЗП – на перетинанні знаходився конденсатор;
- індуктивні ПЗП;
- діодні ПЗП;
- транзисторні ПЗП.

Такі ПЗП звичайно програмуються на одному з останніх технологічних етапів їхнього виробництва. Елементи комутації являють собою просто проміжки, частина з яких перемикає на

останньому етапі металізації схеми. Це здійснюється за допомогою масок – фотошаблонів, що задають точну форму ділянок металізації і виготовлених для кожного конкретного наповнення ПЗП.

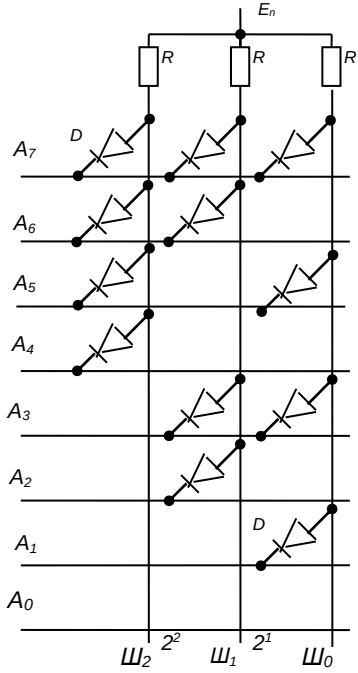


Рис. 11.4. Схема діодного ПЗП

Найпростішим видом ПЗП є діодне ПЗП (рис. 11.4). Вибір потрібного слова здійснюється подачею сигналу низького рівня на відповідну адресну шину A_i. При цьому діоди, що з'єднують розрядні лінії й обрану адресну лінію, мають малий опір, що обумовлює низький рівень напруги на відповідних розрядних лініях. Якщо ж діода в точці перетинання немає, то струм через резистор R не протікає і на виході відповідної розрядної лінії $Ш_j$ встановлюється одиничний сигнал.

У ПЗП на рис. 11.4 записано вісім 3-розрядних кодів, що відповідають першим восьми двійковим числам від 000 до 111.

Матриця ПЗП на МОП-транзисторах показана на рис. 11.5. У ній записані ті ж дані, що й у діодній масці на рис. 11.4. При виборі визначеної адреси транзистори, які підключені до відповідної адресної шини A_i, переходять у провідний стан. Якщо сток транзистора підключений до загальної шини джерела живлення, то на відповідній розрядній шині $Ш_j$ установиться низький потенціал (0).

Якщо металізація стоку транзистора відсутня, то на розрядній шині буде високий потенціал (1).

Запис інформації в ПЗП здійснюється підключенням або непідключенням МОП-транзистора до загальної шини шляхом металізації стоку транзистора у відповідних точках ВІС.

Метод програмування маскою застосовується переважно для матриць МОП-транзисторів. При програмуванні маскою на кристалі формується базова матриця без металізації в областях стоку, замка і джерела. Після виготовлення маски для металізації в процесі металізації за її допомогою до загальної шини підключаються тільки ті транзистори, що повинні забезпечувати рівень “0”.

Багаторічна популярність масочних ПЗП обумовлювалася низькою ціною при багатосерійному виробництві (за технологією виготовлення це досить дорогі пристрої). На даний час, у зв’язку з різким зниженням цін на програмувальну і перепрограмувальну пам’ять, застосовуються рідко. Найбільш розповсюджені мікросхеми цього сімейства – серія Intel 23xxx.

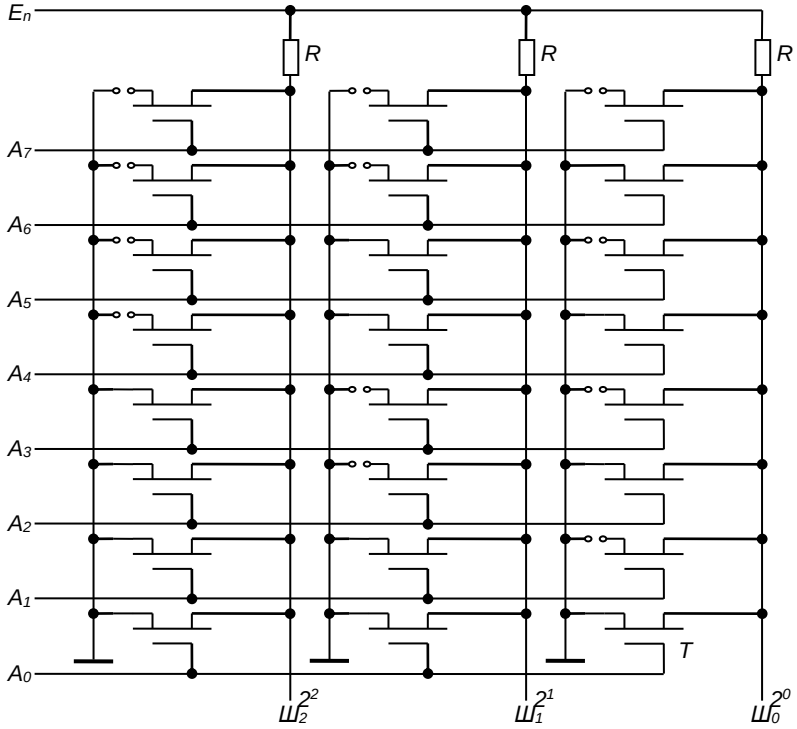


Рис. 11.5. Матриця ПЗП на МОП-транзисторах

Сучасні масочні ІС мають інформаційну ємність до 10 Мбітів при часі доступу близько 100 нс.

11.2.3.2. Програмовані ПЗП (ППЗП, англ. PROM – Programmable ROM)

На відміну від ПЗП, програмованих маскою, у ПЗП, програмованих користувачем, інформація може бути занесена користувачем за допомогою спеціального стенда програмування.

Програмовані ПЗП (ППЗП, англ. PROM – Programmable ROM) будуються на основі біполярних діодних матриць або матриць біполярних транзисторів аналогічно матриці МОП-транзисторів у масочному ПЗП. За своєю структурою ППЗП аналогічні масочним ПЗП, але в усі розряди послідовно з переходами база-емітер біполярних транзисторів або р–п-переходами діодів уставляється плавка вставка (ПВ). ПВ являє собою невелику ділянку металізації, що руйнується (розплавляється) при подачі імпульсу струму (звичайно величиною 50 – 100 мкА і тривалістю 2 мс).

Достоїнства: можливість самостійного запису інформації в ППЗП робить їх придатними для штучного і дрібносерійного виробництва.

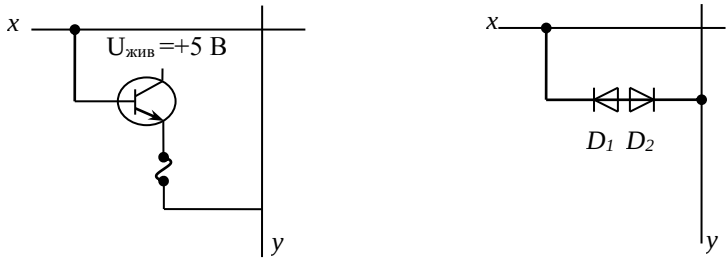
Недоліки: якщо необхідно змінити хоча б один розряд, потрібно замінити весь пристрій, тому що програмування можливе тільки один раз. Крім того, у процесі програмування мікросхема значно нагрівається. Також при виготовленні має місце великий відсоток браку; для підвищення надійності збереження даних необхідне спеціальне тривале термічне випробування. Плавкі перемички займають на кристалі багато місця, тому рівень інтеграції ЗП з такими перемичками істотно нижче, ніж у масочних ЗП.

Після програмування ППЗП стає цілком еквівалентним масочному ПЗП.

Запам’ятовуючі елементи інтегральних ППЗП показані на рис. 11.6. Приклад ППЗП показаний на рис. 11.7.

Пристрій може знаходитись в одному з двох режимів: читання і програмування. Режим визначається значенням живильної напруги $V_{cc.}$ У режимі читання V_{cc} має нормальне значення 5 В; у режимі програмування V_{cc} = 10 В. До лінії живлення V_{cc} підключена гранична схема, що виробляє сигнал «режим читання» RD, рівний логічній 1, коли $V_{cc} < 7{,}5$ В. Цей сигнал разом із зовнішнім сигналом CE

використовується як дозвіл і для лінії вибірки рядків, і для лінії вихідних даних.



а *б*

Рис. 11.6. Запам'ятовуючі (єднальні) елементи ППЗП:
а – елемент із плавкою (випалюваною) перемичкою; *б* – елемент, що пробивається

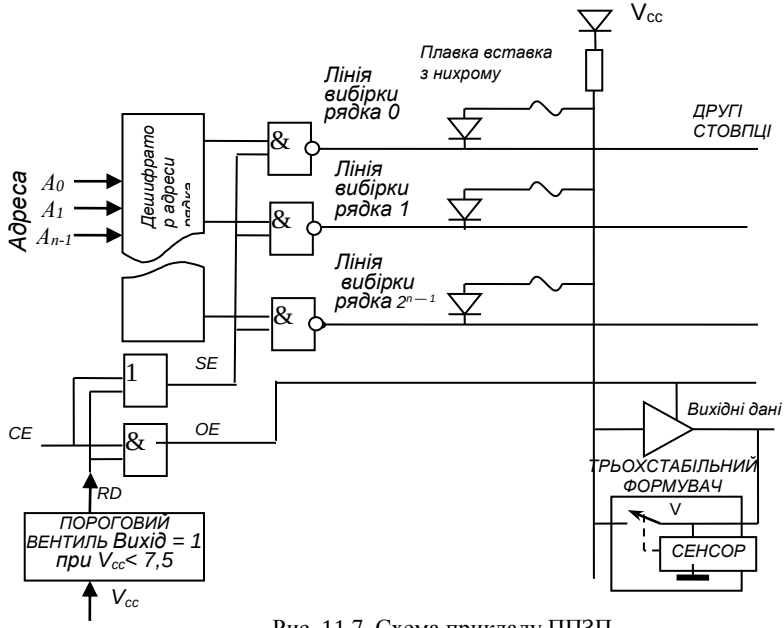


Рис. 11.7. Схема прикладу ППЗП

Якщо плавка перемичка на перетинанні обраного стовпця й обраного рядка ціла, на лінії даних у стовпці буде низький рівень (логічний 0); у противному разі – через резистор – високий (логічна 1). Лінія даних у стовпці підключена до лінії виходу через трьохстабільний формувач, який управляється сигналом «дозвіл виходу» OE (OE = CS ^ RD).

У режимі програмування V_{cc} = 10 В. Тоді сигнал RD стає рівним логічному 0. При цьому, по-перше, сигнал «дозвіл вибірки» SE стає залежним від зовнішнього сигналу CE. Таким чином, на лінії обраного рядка буде логічний 0, тільки якщо на лінії CE буде логічна 1. По-друге, сигнал OE стане логічним 0, і вихідний формувач виявиться закритим незалежно від сигналу CE. Крім того, потенціал на лініях невибраних рядків підніметься до 10 В, оскільки вентилі, що працюють на цій лінії, живляться від напруги V_{cc}, і вона визначає рівень логічної 1. Щоб «перепалити» перемичку, задається потрібна адреса, активується лінія вибірки модуля, що дає дозвіл для вибірки рядків, і потім від джерела струму подається визначений струм (близько 65 мА) на потрібну лінію вихідних даних (у режимі програмування лінії виходу працюють як лінії входу). Цей струм проходить через управляючий напругою ключ на лінію даних відповідного стовпця. Ключ являє собою спеціальну схему, що замикається в тому випадку, коли напруга від джерела струму трохи перевищує максимальний рівень логічної 1, тобто 5 В. Прикладена напруга повинна бути досить високою.

Минаючи ключ, струм потече через плавку перемичку і діод на лінію вибірки рядка, що знаходиться під низьким потенціалом (логічний 0).

Струм, що йде на всі інші лінії вибірки, які знаходяться під потенціалом близько 10 В, буде порівняно малий (для цього напруга живлення і була збільшена до 10 В). Перемичка плавиться швидко. Після цього дезактивується лінія CE. Потім процес можна повторити для інших перемичок у цьому ж рядку і для інших рядків.

На даний час ППЗП з плавкими перемичками мають інформаційну ємність до 64 кбітів при часі доступу близько 60 нс.

11.2.3.3. Електрично-стираючі програмувальні ПЗП

EPROM – такий тип ПЗП, що допускає стирання інформації і перепрограмування настільки часте, наскільки це необхідно. Базовим

для таких ЗП, мабуть, є режим читання. Термін збереження інформації складає кілька років. Зносостійкість – до 10^7 циклів стирання-запису. Час запису (порядку мілісекунд) значно перевищує час читання, для запису необхідні спеціальні програматори. Запис у мікросхему EPROM здійснюється електричними сигналами, а стирання – ультрафіолетовим випромінюванням.

Перші комерційні продукти EPROM з'явилися в 70-х роках, коли фірмою Intel була розроблена технологія FAMOS (мікросхеми серії 27xx).

Властивість перепрограмування забезпечується застосуванням ЕП із властивостями управляючих «перемичок», функції яких виконують транзистори зі структурою МНОП (метал Al – нітрид кремнію Si_3N_4 – окисел кремнію Si_2 – напівпровідник Si) і транзистори *n*-МОП із замком, що плаває, (ПЗ) з використанням механізму лавинної інжекції заряду (ЛИЗМОП, англ. FAMOS – floating-gate avalanche-injection MOS).

МНОП-транзистор відрізняється від звичайного МОП-транзистора двошаровим підзамковим діелектриком. На поверхні кристала розміщений тонкий шар двоокису кремнію Si_2, далі більш товстий шар нітриду кремнію Si_3N_4 і потім уже замок (рис. 11.8).

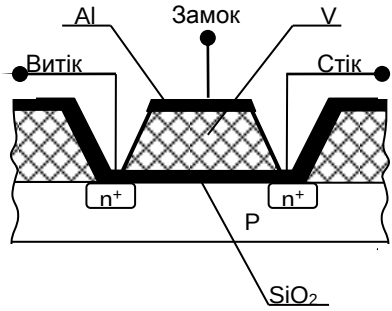


Рис. 11.8. Структура МНОП-транзистора

На границі діелектричних шарів виникають центри захоплення заряду. Завдяки тунельному ефекту носії заряду можуть проходити через тонку плівку окислу товщиною не більше 5 нм і збиратись на границі шарів. Цей заряд і є носієм інформації, збереженої МНОП-транзистором. Заряд записують шляхом створення під замком напруженості

електричного поля, достатньої для виникнення тунельного переходу носіїв заряду через тонкий шар Si_2. На границі діелектричних шарів можна створювати заряд будь-якого знака в залежності від спрямованості електричного поля в підзамковій області. Наявність заряду впливає на граничну напругу транзистора.

Для МНОП-транзистора з *n-каналом* негативний заряд на границі шарів підвищує гранична напруга (екранує вплив позитивної напруги на замку, що відмикає транзистор). При цьому гранична напруга зростає настільки, що робочі напруги на замку транзистора не в змозі його відкрити (створити в ньому провідний канал). Транзистор, у якому заряд відсутній або має інший знак, легко відкривається робочим значенням напруги. Так здійснюється збереження біта в МНОП: один зі станів трактується як відображення логічної одиниці, інший – нуля. Елемент пам'яті зі структурою МНОП являє собою МДП-транзистор з індукованим каналом *p*- або *n*-типу, що має двошаровий діелектрик під замком. Верхній шар формують з нітриду кремнію, нижній – з окислу кремнію, причому нижній шар значно тонше верхнього.

Якщо до замка відносно підкладки прикласти імпульс напруги позитивної полярності з амплітудою 30…40 В, то під дією сильного електричного поля між замком і підкладкою електрони одержують достатньо енергії, щоб пройти тонкий діелектричний шар до границі двох діелектриків. Верхній шар (нітриду кремнію) має значну товщину, так що електрони перебороти його не можуть.

Накопичений на границі поділу двох діелектричних шарів заряд електронів знижує гранична напруга. Цей стан ЕП відповідає логічній 1. Режим занесення заряду під замок називають режимом програмування.

Логічному 0 відповідає стан транзистора без заряду електронів у діелектрику. Режим витиснення заряду з підзамкового діелектрика називають режимом стирання. Для зняття заряду із замка на матрицю EPROM через прозору кварцову кришку подається ультрафіолетове (УФ) випромінювання. При цьому відбувається фотоемісія «гарячих» електронів, що переводить всі транзистори пристрою до непровідного стану (подача УФ-випромінювання на 10 – 20 хв цілком повертає всі запрограмовані транзистори до вихідного стану). У схемах з УФ-стиранням число циклів перепрограмування істотно обмежено, тому що під дією УФ променів властивості матеріалів поступово змінюються.

РПЗП на МНОП-транзисторах енергонезалежні і можуть зберігати інформацію місяцями, роками і десятками років, однак після 10^4...10^6 перезаписів МНОП-транзистор перестає стійко зберігати заряд.

Розглянемо одну з перших EPROM-мікросхем – Intel 2716 (рис. 11.9).

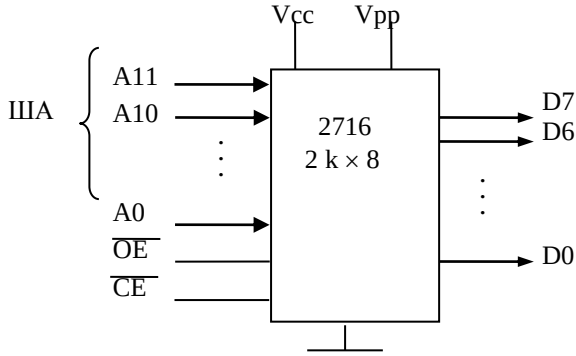


Рис. 11.9. Схема EPROM-мікросхем – Intel 2716

На рис. 11.9 наведено такі позначення:

OE – управління вхідним буфером (Output Enable).

CE – вхідний управляючий сигнал. Виконує двояку функцію:

а) у режимі нормального функціонування низьке значення потенціалу на CE і OE забезпечує виконання операції читання з даного ПЗП і дозволяє роботу його внутрішніх схем (функція вибору кристала). Слово, що зчитується, по виходах D0 – D7 надходить на шину даних. Значення високого потенціалу на CE означає стан спокою, при цьому схема споживає менше енергії (~130 мкВт), ніж в активному режимі (~500 мкВт).

б) у режимі програмування на вхід CE подаються імпульси амплітудою 5 В і тривалістю 50 мс, що достатньо для перезапису осередку ПЗП. При цьому на адресних входах задається необхідна адреса, а на лініях D0 – D7, що тепер виконують роль входів, задається для запису слово. Стан входу OE = 1 (+5 В). Для перевірки правильності запису звичайно проводиться контрольне зчитування осередку.

Напруга живлення V_{cc} = +5 В, V_{pp} = +25 В.

Процес запису інформації здійснюється таким способом: на адресні входи задається необхідна адреса, на шину даних задається необхідне слово, при цьому сигнал OE – високий. Напруга живлення: V_{cc} = +5 В, V_{pp} = +25 В. Прикладається імпульсна напруга до CE і протягом інтервалу часу t = 50 мкс в осередок заноситься необхідне слово. Щоб перевірити правильність запису, проводиться контрольне зчитування осередку. Зчитування здійснюється шляхом подачі низького потенціалу на OE і CE. При цьому відповідні напруги від'єднуються від D0 – D7, адреса залишається такою ж, а вихідні значення зчитуються на D0 – D7.

Такий процес можна робити як вручну, так і автоматично.

Недоліки EPROM: 1) при необхідності зміни хоча б одного розряду потрібно стирати всю інформацію з EPROM, а потім переписувати її заново; 2) для програмування мікросхему EPROM потрібно вийняти з плати і помістити в програматор; 3) обмежена кількість циклів стирання.

11.2.3.4. Репрограмовані ПЗП

Репрограмовані ПЗП (РПЗП, англ. EEPROM – Electrically Erasable Programmable ROM) з'явилися на початку 80-х років. Перший пристрій – Intel 2816 (2K × 8), час доступу – 250 нс.

Архітектура EEPROM подібна архітектурі EPROM, відмінності полягають тільки в побудові осередку.

У 1980 р. фірма Intel розробила технологію FLOTOX (Floating Gate Tunnel-Oxide – плаваючий замок з тунелюванням в окислі). Комірки пам'яті такого типу реалізовані на МОП-елементах (ізольованим) замком, що використовують ефект лавинної інжекції. МОП-транзистор має 2 кремнієві замки (плаваючий і управляючий), ізольованих один від одного шаром ізолятора (Si_2) товщиною 0,2 мкм.

Ізольований замок не має електричного підведення, він призначений для збереження заряду. У результаті тунелювання електричний заряд може як накопичуватися, так і стікати із плаваючого замка.

Принцип роботи ЛИЗМОП з подвійним замком (рис. 11.10) близький до принципу роботи МНОП-транзистора – тут також між управляючим замком і областю каналу міститься область, у яку при програмуванні можна вводити заряд, що впливає на величину граничної напруги транзистора.

Але область введення заряду являє собою не границю шарів діелектрика, а оточену з усіх боків діелектриком провідну область (звичайно з полікристалічного кремнію), у яку, як у пастку, можна ввести заряд, здатний зберігатися в ній протягом тривалого часу. Ця область і називається плаваючим замком.

При подачі на управляючий замок, джерело і стік імпульсу позитивної напруги великої амплітуди (20...25 В) у зміщених назад *p-n* переходах виникає лавинний пробій, область якого насичується електронами. Частина електронів, що мають енергію, достатню для подолання потенційного бар'єра діелектричної області, проникає в плаваючий замок. У цьому полягає процес програмування. Зняття високого програмуючої напруги відновлює звичайний стан областей транзистора і замикає електрони в плаваючий замок, де вони можуть знаходитися тривалий час (у високоякісних приладах декілька років).

Заряджений електронами плаваючий замок збільшує граничну напругу транзистора настільки, що в діапазоні робочих напруг провідний канал у транзисторі не створюється. При відсутності заряду в плаваючому замку транзистор працює в звичайному ключовому режимі. Цей стан відповідає логічній «1». Щоб його забезпечити, на замок подають імпульс напруги негативної полярності з амплітудою 30...40 В. При цьому електрони витісняються в підкладку. При відсутності заряду електронів під замком передатна характеристика зміщається в область високих граничних напруг. Режим витиснення заряду з підзамкового діелектрика називають режимом стирання.

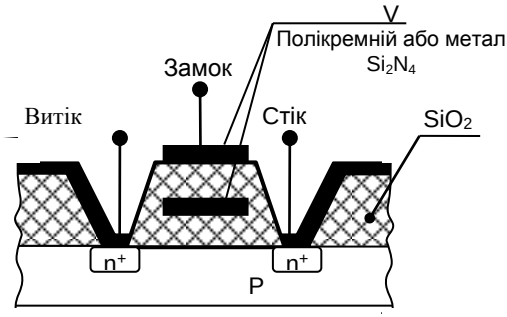


Рис. 11.10. Структура ЛИЗМОП з подвійним замком

Комірка пам'яті типу FLOTOX складається з двох МОП-транзисторів, один із яких (V2) призначений для збереження інформації, інший (V1) – для вибору осередку (рис. 11.11).

Розмір такого елементарного осередку – 3 × 3 × 8 мкм.

Для EEPROM немає необхідності використовувати ультрафіолетове випромінювання, оскільки запис і стирання інформації здійснюється електричним способом. Маючи відповідну електроніку, саме програмування можна здійснювати безпосередньо на платі, без зовнішніх електричних ланцюгів. При зміні одного слова не треба перепрограмовувати весь масив, тому що є доступ до окремих бітів інформації; до того ж, перепрограмування відбувається значно швидше.

Як приклад запам'ятовуючого пристрою розглянемо ВІСА РПЗП типу КР1601РР1 з інформаційною ємністю 1 к × 4 = 4 кбіт.

Умовно-графічне позначення мікросхеми наведено на рис. 11.12.

На рис. 11.12 використані такі позначення:

A0 – A9 – входи адреси;

D0 – D3 – входи / виходи даних;

CS – вибір кристала;

RD – вхід сигналу зчитування;

PR – вхід сигналу програмування;

ER – вхід сигналу стирання;

U_{PR} – вхід напруги програмування.

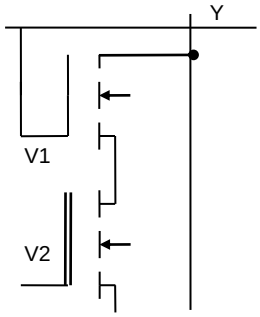


Рис. 11.11. Комірка пам'яті FLOTOX

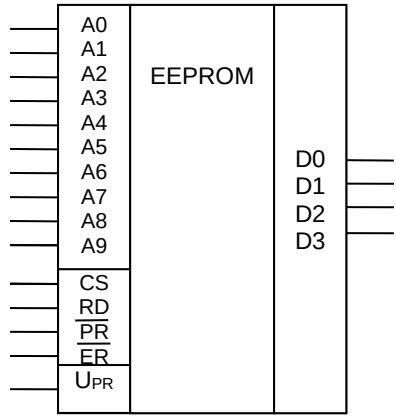


Рис. 11.12. Умовно-графічне позначення мікросхеми РПЗП типу КР1601РР1

Стирання в цій мікросхемі означає установку всіх розрядів у “1”. Режими роботи мікросхеми наведені в табл. 11.1.

Таблиця 11.1

	ER	PR	RD	A0 ÷ A9	UPR	D1/0	Режим
0	×	×	×	×	X	Roff	Збереження
1	0	1	0	×	– 33 – 31 В	×	Загальне стирання
1	0	0	0	A	—//—	×	Виборче стирання
1	1	0	0	A	—//—	D1	Запис даних
1	1	1	1	A	– 33 – 5 В	D0	Зчитування

11.2.3.5. Flash ЗП

Flash-пам’ять (Flash Memory) за типом запам’ятовуючих елементів і основних принципів роботи подібна пам’яті типу EEPROM, однак ряд архітектурних і структурних особливостей дозволяють виділити її в окремий клас. Розробка flash-пам’яті вважається кульмінацією десятирічного розвитку схемотехніки пам’яті з електричним стиранням інформації.

Для збільшення ємності пам’яті в таких ПЗП транзистор адресації вилучений з комірки пам’яті (адресація до окремого біта стає неможливою). Тому стирання інформації здійснюється або для всієї пам’яті одночасно, або для досить великих блоків. Звичайно flash-пам’ять робиться байт-байт-орієнтованою. Такий пристрій стає альтернативним засобом збереження інформації (наприклад, flash-диски).

Робота flash-пам’яті містить три операції – запис або програмування, читання, стирання. На кожну з операцій потрібен визначений час. Наприклад, для читання одного параметричного блока витрачається приблизно 60 нс, а для запису 1 мкс. На операцію стирання інформації в середньому витрачається від 0,6 секунди. Це найдовша операція.

Число циклів репрограмування для flash-пам’яті хоча і велике, але обмежене, тобто осередки при перезаписі “зношуються”. Щоб збільшити довговічність пам’яті, у її роботі використовуються спеціальні алгоритми, що сприяють “розрівнюванню” числа перезаписів по всіх блоках мікросхеми.

Для поліпшення техніко-економічних характеристик у схемах flash-пам'яті застосовуються різні засоби і прийоми:

1. Переривання процесів запису при звертаннях процесора для читання (Erase Suspend). Без цього виникали б тривалі простої процесора, тому що запис займає досить великий час. Після переривання процес запису відновляється під управлінням внутрішніх засобів flash-пам'яті.

2. Внутрішня черга команд, управляючих роботою flash-пам'яті, що дозволяє організувати конвеєризацію виконуваних операцій і прискорити процеси читання і запису.

3. Програмування довжини збережених у ЗП слів для узгодження з різними портами вводу-виводу.

4. Ввід режимів зниженої потужності на час, коли до ЗП немає звертань, у тому числі режиму глибокого спокою, у якому потужність знижується до вкрай малих значень (наприклад, струм споживання знижується до 2 мкА). Ці особливості дуже важливі для пристроїв з автономним (батарейним) живленням.

5. Пристосованість до роботи при різних напругах живлення (5 В; 3,3 В та ін.). Сама схема "почуває" рівень живлення і робить необхідні переключення для пристосування до нього.

6. Введення в структури пам'яті сторінкових буферів для швидкого накопичення нового даного, що підлягає запису. Два таких буфери можуть працювати в режимі, названому "пінг-понг", коли один з них приймає слова, що підлягають запису, а інший на цей час забезпечує запис свого вмісту в пам'ять. Коли перший буфер заповниться, другий уже звільниться, і вони поміняються місцями.

7. Різні міри захисту від випадкового або несанкціонованого доступу.

Схемотехнічні різновиди flash-пам'яті з прямим і послідовним доступом

На даний час однією з базових технологій є технологія пам'яті NAND (NOT AND). Подібного роду модулі функціонують аналогічно накопичувачам на магнітних дисках: система генерує початкову адресу; внутрішні блоки мікросхеми переміщають покажчики на цю адресу й обрані дані передаються в байт-послідовному порядку з використанням стробуючих сигналів.

Стирання вмісту всієї мікросхеми або її частини, часто названої сектором за аналогією з магнітним диском, здійснюється окремим сигналом (звідси і походить назва flash-пам'ять – "спалах"). Це дозволяє скоротити тривалість службових сигналів, необхідних для послідовного стирання байта за байтом.

Більшість виготовлювачів flash-пам'яті NAND використовують 0,18 мкм технологію. Цей метод дозволяє одержати майже чотириразове збільшення щільності заповнення кристалів у порівнянні з пристроями, виготовленими за технологією з топологічними нормами 0,35 – 0,5 мкм, використовуваної зараз для виробництва пристроїв ємністю до 64 Мбітів.

На рис. 11.13 представлена архітектура мікросхеми flash-пам'яті NAND ємністю 64 Мбіти виробництва компанії Samsung.

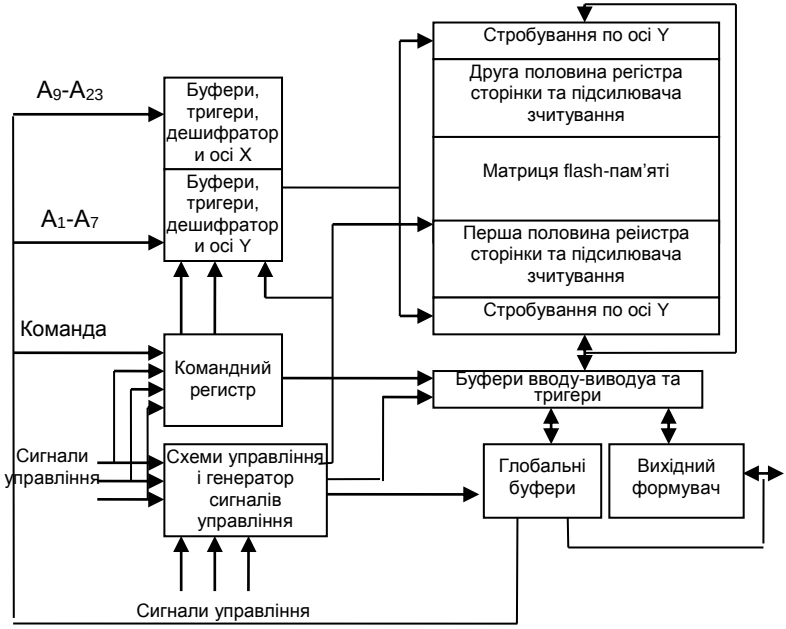


Рис. 11.13. Flash-пам'ять фірми Samsung

Ємність 64 Мбіти типова для пристроїв flash-пам'яті на основі осередків NAND. Дані, адреси і команди передаються через порт

вводу-виводу. Команди завантажуються в регістр команд, адреса надходить у X- і Y-тригери і декодується. При виконанні команди зчитування дані зчитуються з масиву комірок пам'яті, передаються в буфер вводу-виводу, а потім на порт вводу-виводу.

Осередок NAND складається з восьми послідовно з'єднаних транзисторів, з яких у вигляді ланцюжка формується структура NAND.

Пристрої на комірках пам'яті NAND забезпечують більш високу щільність збереження інформації, ніж пристрої на осередках NOR (АБО-НІ) і EEPROM, що робить їх досить привабливими для використання в прикладеннях, де немає необхідності в прямому доступі до пам'яті. За рахунок застосування комірок пам'яті NAND при виробництві пристроїв з дуже великою ємністю пам'яті, розроблювачі можуть одержати додатковий виграш у ємності та домогтися функціонування пристрою подібно магнітному диску.

На відміну від пристроїв EEPROM і EPROM з ультрафіолетовим стиранням, у яких час доступу до визначеного слова даних складає менше 150 нс, модулі пам'яті NAND мають значно більший час чекання. Він може досягати декількох мікросекунд, що подібно часу чекання в декілька мілісекунд для доступу до першого слова файлу на магнітному диску. Однак наступні байти при такому типі доступу можуть бути враховані дуже швидко, звичайно зі швидкістю від 50 до 100 нс на байт, тобто зі швидкістю проходження тактових імпульсів. Таким чином, байт-послідовний формат організації пам'яті подібний роботі накопичувача на магнітних дисках, але забезпечує набагато велику швидкість зчитування інформації.

Компанія Hitachi розробила систему збереження інформації на осередках AND (I), що може скласти конкуренцію описаним вище пристроям. Пристрої на основі осередків AND споживають меншу потужність і вимагають меншої корисної площі на кристалі, ніж аналогічний за ємністю ЗП на основі осередків NAND. На основі структури AND Hitachi робить мікросхеми пам'яті з ємністю до 128 Мбітів зі стандартною організацією один біт на осередок. Мікросхеми мають більш високу швидкість запису сектора – всього 200 мкс. Окремо слід зазначити підвищену зносостійкість мікросхем – до 300 000 циклів стирання-запису, що в три рази перевищує аналогічний параметр у мікросхем ємністю 64 Мбіти.

Як уже відзначалося, мікросхеми flash-пам'яті NAND є усього лише одним напрямком з набору технологій виготовлення електрично стираємих і програмувальних запам'ятовуючих пристроїв. Два інших

напрямки побудови flash-пам’яті можуть бути описані як пристрої прямого доступу і пристрої біт-послідовного доступу. Пристрої прямого доступу мають повнорозрядну адресну 8- або 16-розрядну шину. До будь-якого слова в пам’яті можна звернутися усього за один цикл роботи. Звичайно, для найбільш швидкодіючих інтегральних схем тривалість циклу складає не більше 40 нс, для більшості доступних пристроїв — не більше 100 нс. З іншого боку, біт-послідовні пристрої повинні передавати команду управління (читання, запис і так далі), адреса і строб даних біт за бітом, а це вимагає 20 і більше циклів роботи для однієї операції доступу.

Накопичувачі на основі осередків АБО-НІ (з паралельним включенням ЛИЗМОП-транзисторів з подвійним замком) забезпечують швидкий доступ до слів при довільній вибірці. Вони використовуються для різних застосувань, але найбільш придатним вважається їхнє застосування в пам’яті для збереження рідко обновлюваних даних.

Компанія Mitsubishi розробила власну структуру організації осередків для модулів flash-пам’яті з довільним доступом, що носить назву DINOR (DIvided bit-line NOR – структура NOR з розділеними розрядними лініями). Структура DINOR споживає меншу потужність і вимагає меншої корисної площі на кристалі, ніж алогічний за ємністю ЗП на основі осередків NOR.

Архітектурні різновиди flash-пам’яті

Двома основними напрямками ефективного використання Flash-пам’яті є збереження не дуже часто змінюваних даних (обновлюваних програм, зокрема) і заміна пам’яті на магнітних дисках.

Для першого напрямку в зв’язку з нечастим відновленням вмісту параметри циклів стирання і запису не настільки істотні, як інформаційна ємність і швидкість зчитування інформації. Стирання в цих схемах може бути як одночасним для всієї пам’яті, так і блоковим. Серед пристроїв із блоковим стиранням виділяють схеми зі спеціалізованими блоками (несиметричні блокові структури). По імені так званих Boot-блоків, у яких інформація надійно захищена апаратними засобами від випадкового стирання, ці ЗП називають Bool Block Flash Memory. Boot-блоки зберігають програми ініціалізації системи, що дозволяють ввести її в робочий стан після включення живлення.

Мікросхеми для заміни твердих магнітних дисків (Flash-File Memory) містять більш розвинуті засоби перезапису інформації і мають ідентичні блоки (симетричні блокові структури).

Пам'ять типу Bulk Erase фірми Intel має час запису байта близько 10 мкс, допускає до 10^5 циклів стирання, напруга програмування для неї складає 12 В ± 5%, струм активного режиму близько 10 мА, у режимі спокою близько 50 мкА. Час доступу при читанні дорівнює приблизно 100 нс, час стирання і час програмування всього кристала складає 0,6...4 для кристалів ємністю 256 кбітів...2 Мбіти.

На відміну від традиційного управління схемами пам'яті за допомогою адресних і управляючих сигналів, flash-пам'ять має додаткове управління словами-командами, записуваними процесором у спеціальний регістр, що функціонує тільки при високому рівні напруги на виводі мікросхеми, що позначається U_{pp} (напрузі програмування). При відсутності такого рівня U_{pp} схема працює тільки як пам'ять для читання під управлінням традиційних сигналів, що задають операції читання, зниження потужності, управління третім станом і видачі ідентифікатора.

На рис. 11.14 показана структура flash-пам'яті типу Bulk Erase (схеми 28F010, 28F020 фірми Intel та ін.).

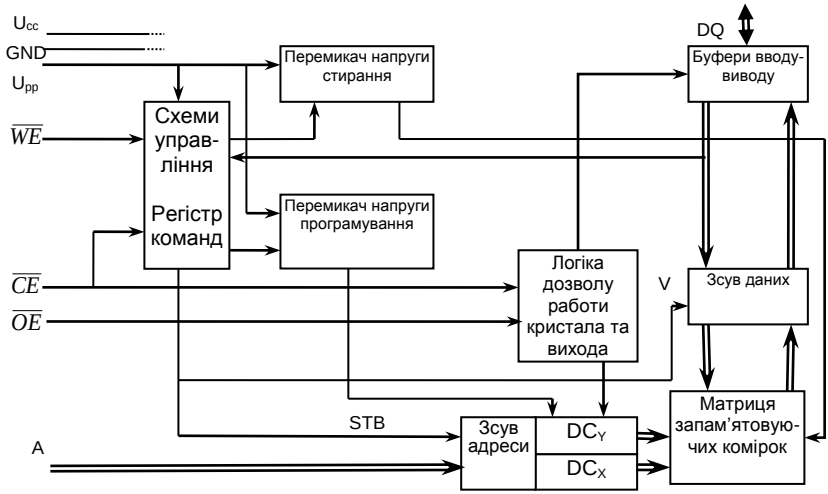


Рис. 11.14. Структура flash-пам'яті зі стиранням даних одночасно з усього кристала (типу Bulk Erase)

Входи A є адресними, причому протягом циклу запису адреси фіксуються в регістрі-засувці за сигналом строба STB. Введення-виведення даних (лінії DQ) здійснюється через буфери з третім станом. Протягом циклу запису дані фіксуються в регістрі-засувці.

Сигнали CE, OE, WE L-активні. Сигнал CE активізує управляючу логіку, буфери вводу-виводу даних, дешифратори адреси DC_Y, DC_X і підсилювачі читання. При високому рівні сигналу CE (схема не обрана) буфери входять у третій стан, а споживання потужності знижується до рівня спокою (Standby).

Сигнал OE низьким рівнем дозволяє вивід даних через буфери протягом циклів читання (природно, тільки при низькому рівні сигналу CE).

Сигнал WE дозволяє запис у регістр команд і матрицю запам'ятовуючих осередків і своїх фронтів, завантажує регістри-засувки (від'ємним – регістр-засувку адреси, додатним – даних).

Схеми управління і вміст регістра команд визначають стан перемикачів рівнів напруги U_{pp}, використовуваних у різних режимах роботи (при стиранні, програмуванні або читанні).

Для одержання вихідних даних при читанні потрібен активний стан сигналів CE і OE. При цьому низький рівень U_{pp} робить можливим читання тільки даних пам'яті, а високий дозволяє зчитувати також коди ідентифікаторів та інформацію для перевірки операцій стирання-програмування. Операції з ідентифікатором виводять код фірми-виготовлювача і мікросхеми. Ці зведення дозволяють погоджувати алгоритми стирання і програмування схеми і програмуючого устаткування, що виробляється автоматично.

Коди ідентифікаторів знаходяться в двох комірках пам'яті і можуть зчитуватися за допомогою визначеної комбінації сигналів або регістра команд (читанням після подачі в регістр команди 90H).

При виконанні операцій запису коди адрес і даних фіксуються у внутрішніх регістрах-засувках. При високому рівні U_{pp} виконуються ті ж операції і додатково дозволяється стирання і програмування пам'яті. Всі дії, пов'язані зі зміною вмісту пам'яті, здійснюються з використанням регістра команд. Регістр команд не займає будь-яку позицію в адресному просторі і завантажується звичайним циклом запису від процесора при низькому рівні U_{pp}. Його вміст відіграє роль вхідної інформації для внутрішнього автомата управління схемами стирання і програмування пам'яті. Використовуються 7 команд, дві з яких задають операції читання (даних і кодів ідентифікатора), дві інші

відносяться до операції стирання (підготовка і перевірка стирання), дві команди відносяться до операції програмування (підготовка і перевірка програмування) і одна команда задає операцію скидання мікросхеми.

При зниженні рівня U_{pp} регістр команд скидається, дозволяючи мікросхемі тільки операції читання.

За командою стирання стираються всі байти матриці паралельно, після чого всі вони повинні бути перевірені. Для цього байти адресуються й активізуються подачею спеціальної напруги. Читання з осередку коду 0FFH показує, що всі біти байта стерті. Якщо зчитується інший код, виконується повторна операція стирання. Потім перевірка відновлюється з адреси останнього перевіреного байта. Процес перевірки продовжується до досягнення останньої адреси.

Програмування пам'яті ведеться байт за байтом (послідовно або при довільному доступі). Цикл читання від процесора виводить дані байта, що порівнюються із заданими. Рівність байтів свідчить про успішне програмування. Після цього процес програмування переходить до наступного байта.

Команда скидання є засобом надійного усунення дії команд стирання-програмування. Після кожної з цих команд у регістр команд можна записати код операції скидання, що усуне можливість будь-яких дій, пов'язаних із зазначеними командами. Вміст пам'яті не зможе змінюватися. Для подальшого приведення схеми до бажаного стану в регістр команд потрібно записати відповідну команду.

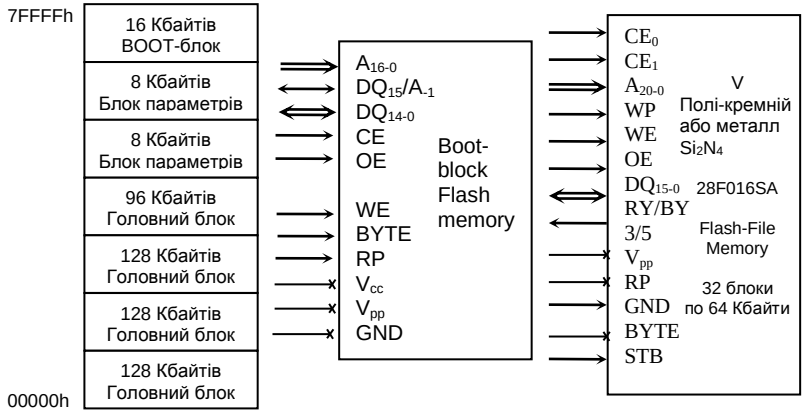
При переході сигналу CE до високого рівня вводиться режим зниженої потужності. Якщо це відбувається при стиранні, програмуванні або перевірках даних, то активний струм зберігається до завершення зазначених операцій.

Схемам типу **Boot Block Flash Memory** (Boot-блок flash-пам'ять, скорочено ББФП) властиве блокове стирання даних і несиметрична блокова архітектура. Блоки спеціалізовані і мають різні розміри. Серед них мається так званий Boot-блок (ББ), вміст якого апаратно захищений від випадкового стирання. У ББ зберігається програмне забезпечення базової системи вводу-виводу мікропроцесорної системи BIOS (Basic Input/Output System), необхідне для правильної експлуатації й ініціалізації системи.

У складі блоків є також БП (блоки параметрів) і ГБ (головні блоки), які не постачаються апаратними засобами захисту від непередбаченого запису. Блоки БП зберігають параметри системи, які відносно часто

змінюються (коди ідентифікаторів, діагностичні програми і т.п.). Блоки ГБ зберігають основні управляючі програми і т.п.

Мікросхеми ББФП призначені для роботи з різними мікропроцесорами і для відповідності їм мають два варіанти розміщення ББ в адресному просторі: вгорі і внизу, що відображається в маркуванні ІС буквами T (Top) або B (Bottom). На рис. 11.15, *а* для прикладу наведена карта пам'яті (розподіл адресного простору) для ІС ємністю 4 Мбіти з верхнім розміщенням ББ.



а *б* *в*

Рис. 11.15. Розподіл адресного простору і зовнішня організація flash-пам'яті:
а, б – з несиметричною блоковою структурою;
в – і зовнішня організація файлової flash-пам'яті

На даний час випускаються ББФП з ємностями 1...16 Мбітів, у наступних поколіннях очікуються ІС з інформаційними ємностями до 256 Мбітів.

За своїм функціонуванням ББФП близькі до пам'яті типу Bulk Erase, в обох типах ІС операції стирання-програмування ведуться під управлінням внутрішнього автомата, вхідною інформацією для якого служать команди, що вводяться від процесора. У схемах ББФП цю роль відіграє так званий командний інтерфейс користувача GUI (Command User Interface).

Зовнішня організація типової ББФП показана на рис. 11.15, *б* на прикладі ІС з інформаційною ємністю 4 Мбіти.

Адреси задаються 19-розрядним кодом $A_{18\text{-}0}$, тобто в пам'яті зберігається до 512 кслів. Сигнал BYTE задає 8- або 16-розрядну організацію пам'яті. При байтовій організації байти передаються по лініях $DQ_{7\text{-}0}$, а лінія DQ_{15} відіграє роль самого молодшого розряду адреси A_{-1}, який визначає байт, що передається (старший або молодший). При словниковій організації виводів $DQ_{15\text{-}0}$ є лініями вводу-виводу даних.

Напруга на виводі RP (Reset/Power Down) може мати три рівні: 12 В ± 5%, рівень логічної одиниці Н і низький рівень L. При напрузі 12 В + 5% ББ відкритий і в ньому можуть виконуватися операції стирання і програмування. При напрузі нижче 6,5 В ББ замкнений.

Маючи ряд режимів економії потужності, схеми ББФП, зокрема, реалізують режим APS (Automatic Power Saving), завдяки якому після завершення циклу читання схема автоматично входить у статичний режим зі споживанням струму близько 1 мА, у якому знаходиться до початку наступного циклу читання.

Коли схема не обрана (при високому рівні сигналу на виводі CE і виводі RP, тобто CE = RP = H) споживання потужності знижується до рівня спокою (десятки мкА). При RP = L не тільки забороняється запис, але і вводиться режим глибокого зниження потужності, у якому струм споживання знижується до часток мкА.

Активному режиму відповідає комбінація сигналів CE = L і RP = H. Сигнали OE і WE мають звичайне призначення. Мікросхеми Boot-блок Flash-пам'яті можуть працювати з різними напругами живлення і програмування (технологія Smart Voltage), мають час доступу при читанні 60...70 нс, струми активних режимів 15...25 мА і вкрай малі струми в режимі глибокого зниження потужності (близько 0,2 мкА).

Важливе місце в ієрархії ЗП займає **файлова flash-пам'ять** (ФФП). Протягом багатьох років збереження великих обсягів даних покладалося в ПК на добре відпрацьовані і порівняно недорогі зовнішні ЗП на магнітних, а згодом і оптичних дисках. У багатьох комп'ютерах система пам'яті організована як поєднання твердого магнітного диска (вінчестера) з динамічним напівпровідниковим ОЗП.

Маючи значні достоїнства, дискові ЗП як електромеханічні пристрої мають ряд недоліків: чутливість до ударів і вібрацій, забруднення, обмежена швидкодія і значне споживання потужності. Ці

недоліки особливо позначаються в портативних пристроях з автономним (батарейним) живленням. Досить відзначити, що дисководи споживають у кращому випадку потужність близько 3 Вт, що в системах з напругами живлення 3,3...5 В означає споживання струмів 0,6...0,9 А, яке швидко виснажує батарейки.

Файлова flash-пам'ять орієнтована на заміну твердих дисків, вона в сотні разів скорочує споживану потужність, збільшує механічну міцність і надійність ЗП, зменшує їхні розміри і вагу, на кілька порядків підвищує швидкодію при читанні даних, зберігаючи при цьому програмну сумісність із засобами управління пам'яттю. Разом з тим, за дисковою пам'яттю залишаються переваги щодо інформаційної ємності та вартості.

Використання ФФП для заміни дискової пам'яті в портативних комп'ютерах – один з найважливіших факторів, що сприяють розвитку цього напрямку. При цьому традиційне сполучення "твердий диск – динамічний ОЗП" може замінятися поєднанням "flash-пам'ять – статичний ОЗП". Команди програми, збережені у ФФП, читаються в цьому випадку безпосередньо процесором, результати теж записуються прямо у ФФП, а операції з інтенсивними обчисленнями, що вимагають найшвидшого доступу до пам'яті та запису даних з байтовою спроможністю, що дозволяє, виконуються з використанням швидкодіючої статичної пам'яті.

Накопичувач ФФП поділяється на блоки, що служать аналогами секторів магнітних дисків. Розроблено програмні засоби, що забезпечують обмін між flash-блоками, подібно тому, як операційна система забезпечує обмін між секторами диска.

Блоки ФФП ідентичні і мають однакову інформаційну ємність (симетрична блокова архітектура). Тому що у ФФП операції запису здійснюються значно частіше, ніж в інших різновидах flash-пам'яті, цим операціям надається велика увага – вводяться сторінкові буфери, що дозволяють з високою швидкістю накопичувати деякий обсяг даного, підлягаючого запису, для їхньої наступної передачі в накопичувач з меншою швидкістю.

Мікросхеми ФФП фірми Intel мають інформаційну ємність до 16 Гбайти при часі доступу в середньому 40 нс, напруги живлення 2,7 В. Вони мають розрядність (16 або 32), напруга програмування в них також, як правило, багатоваріантна (3,3; 5; 12 В).

Зовнішня організація ФФП показана на рис. 11.15, *в*, на прикладі мікросхеми з інформаційною ємністю 16 Мбітів (ІС типу 28F016SA фірми

Intel). Накопичувач схеми з загальною інформаційною ємністю 16 Мбітів розбитий на 32 блоки по 64 кбайти.

Пояснимо зміст деяких виводів і сигналів. Шина адреси: лінії A_{20-16} вибирають один із блоків, лінії A_{15-1} вибирають слово в межах одного блоку (блок з ємністю 64 кбайти містить 32 К слів), лінія A_0 – біт вибірки байта, що визначає старший і молодший байти при байтовій організації пам'яті та відключається при її словниковій організації. Від процесора надходить початкова адреса блоку даних, що запам'ятовується в черзі адрес. Поточна адреса комірки пам'яті для обміну формується адресним лічильником.

У шині даних DQ_{15-0} лінії DQ_{7-0} призначені для введення і виведення молодшого байта даних, передачі команди в командний інтерфейс користувача CUI у циклі запису і виводу даних з буфера, регістрів ідентифікатора або стану у відповідних режимах читання. Лінії DQ_{15-8} призначені для передачі старшого байта при словниковій організації пам'яті. По них виводять дані накопичувача, буфера або ідентифікатора у відповідному режимі читання, але ці лінії не використовуються для читання з регістрів стану. Якщо кристал не обраний або заборонений вивід, лінії шини даних переходять до третього стану.

Лінії CE_0 і CE_1 – входи дозволу кристала, при високому рівні кожного з них кристал не обраний, і споживання потужності знижується до рівня стану спокою (Standby) після завершення поточної операції запису або стирання.

Сигнал OE відкриває вихідні буфери при низькому рівні та переводить їх у третій стан при високому.

Сигнал WE управляє доступом до командного інтерфейсу користувача CUI, сторінкових буферів, регістрів черги даних і засувок черги адрес.

Сигнал RP (Reset/Power-Down) при низькому рівні вводить схему в стан глибокої економії потужності, відключаючи всі схеми, що споживають статичну потужність. При виході з цього стану час відновлення схеми складає 400 нс. При переході до низького рівня операції автомата записи припиняються, схема скидається.

Сигнал RY/BY (Ready/Busy) визначає стан внутрішнього автомата запису. Низький рівень означає зайнятість, високий (сигнал виробляється каскадом з відкритим стоком, що вимагає підключення зовнішнього ланцюжка U_{cc} – R для формування високого рівня) означає або готовність

до нових операцій, або призупинення стирання, або стан глибокої економії потужності в залежності від виконуваної операції.

Сигнал WE (Write Protect) має наступний сенс. Кожен блок має біт заборони запису (Lock-bit). Низький рівень WE дозволяє захист, тобто запис або стирання в блоці можуть виконуватися тільки при Lock-bit = 0. При високому рівні WR у блоках можуть виконуватись операції запису і стирання незалежно від стану бітів, що блокують.

Сигнал BYTE низьким рівнем вводить схему в байтовий режим, високим – у словниковий і виключає буфер лінії A_0.

Напруга програмування U_{pp} і виводу напруги живлення (це може бути 3,3 або 5 В – вхід позначений дробом 3/5) надходять у схему через перемикач напруги, що знаходиться усередині схеми.

Приклади:

1. Параметри ФФП фірми Intel/28F032SA (1997 р.):

- організація 2 М × 16 або 4 М × 8 (на вибір споживача), напруга живлення 3,3 або 5 В (на вибір споживача), напруга програмування 12 В, до 10^6 циклів стирання на блок, 64 незалежних блоки по 64 кбайти або 64 блоки по 32 кслова;
- корпус типу TSOP розмірами 1,2 × 14 × 20 мм із 56 виводів;
- технологія з топологічною нормою 0,6 мкм;
- час доступу при читанні 70 або 150 нс при живленні від 5 і 3 В відповідно;
- час запису слова/байта не більше 9 мкс;
- час запису блока не більше 2,1 для байтового режиму і не більше 1 для словникового режиму;
- час стирання блоку не більше 10 для стирання кристала не більше 25,6 с.

2. Мікросхема KM29U64000 виробництва компанії Samsung.

Мікросхема KM29U64000 виробництва компанії Samsung працює аналогічно магнітному диску і має ємність 64 Мбіти та архітектуру, типову для більшості байт-послідовних модулів flash-пам'яті. Мікросхема містить головну матрицю пам'яті розміром 64 Мбіти і допоміжну матрицю розміром 2 Мбіти, використовувану для контролю парності і корекції помилок. Команди, адресна інформація і дані передаються в мікросхему по шині вводу-виводу. Команди пересилаються в регістр команд, а дані розподіляються в так звані глобальні буфери і потім передаються в головну матрицю

запам'ятовуючого пристрою під час дії сигналу запису-збереження. При читанні даних у мікросхему повинні бути передані команди й адресна інформація. Дані при цьому будуть передаватися з головної матриці на глобальні буфери, а потім на шину вводу-виводу.

3. Мікросхеми Disk-on-Chip Millennium компанії M-Systems.

Типовим представником модулів flash-пам'яті NAND, що функціонують аналогічно магнітному диску, є мікросхеми Disk-on-Chip Millennium компанії M-Systems (корпус з 32 виводами, ємність – від 2 до 144 Мбайтів). Ця мікросхема цілком наслідує принципи функціонування накопичувача на магнітних дисках. Для досягнення цього в неї інтегровані різноманітні функції: flash-пам'ять NAND ємністю 8 Мбайтів, буферне ОЗП ємністю 512 байтів, що забезпечує локальне виконання завантажувального коду, і контролер виявлення та виправлення помилок.

У 1997 р. компанія Intel представила новий вид flash-пам'яті, названий **StrataFlash**, у якій вперше в одному елементі пам'яті зберігаються два біти, а не один. Це забезпечується тим, що в замку транзистора фіксується не тільки наявність або відсутність заряду, але і визначається його величина, що може мати кілька значень. Розрізняючи чотири рівні, можна зберігати в одному елементі два біти.

До винаходу пам'яті StrataFlash для збільшення ємності ЗП йшли шляхом зменшення розмірів схемних елементів та інших удосконалень технологічних процесів літографії. StrataFlash ознаменувала інший підхід до цієї проблеми. Збереження двох бітів домоглися практично в тих же запам'ятовуючих елементах, що раніше зберігали один біт, здолавши труднощі жорсткості допусків на величини зарядів, що вводяться в замок. У другій половині 90-х років з'явилися комерційні зразки пам'яті StrataFlash. При цьому від ємності 32 перейшли до ємності 64 Мбіта без помітних змін площі кристала.

Запам'ятовуючі елементи програмуються введенням у плаваючий замок одного з 4-х величин заряду, кожен з яких відповідає парі двійкових цифр 11, 10, 01, 00. У залежності від заряду, що запам'ятовує, транзистор має одну з чотирьох граничних напруг. При зчитуванні інформації до замка транзистора прикладають напругу зчитування. Струм запам'ятовуючого транзистора залежить від граничної напруги. Визначаючи струм, можна виявити стан плаваючого замка.

Багаторівневий підхід до збереження інформації використовується також компаніями Hitachi і Mitsubishi. Випускаються

пристрої пам’яті ємністю 256 Мбітів, що займають на кристалі площу всього 138,6 мм^2. Матриця запам’ятовуючого пристрою виконується за схемою один транзистор на біт і поділяється на сектори, кожен з яких містить 8192 + 256 комірок пам’яті (8192 для даних і 256 для корекції помилок), що з урахуванням багаторівневого принципу дозволяють зберігати 16384 + 512 бітів або 2048 + 64 байти корисної інформації. Зазначена організація масиву пам’яті дозволяє зберегти час стирання і запису сектора до 1 мс, що дозволяє програмувати мікросхему зі швидкістю 2 Мбайти за секунду.

Небагаторівневий підхід до збільшення ємності flash-пам’яті на прикладі мікросхеми компанії Toshiba

Компанія Toshiba розробила мікросхему ємністю 256 Мбітів, що використовує стандартний спосіб збереження інформації “один біт на осередок”. Завдяки використанню методу ізоляції неглибокими канавками, що забезпечує більш щільне компонування транзисторів, розроблювачі компанії змогли розмістити 256 млн. комірок пам’яті на площі всього 130 мм^2. Завдяки відмові від передачі службових сигналів, необхідних для обслуговування комірок пам’яті, була досягнута швидкість програмування 4,4 Мбайт/с. Використання бітової плаваючої лінії з екраном дозволяє досягти дуже гарної величини часу чекання при першому доступі – всього 3,8 мкс, а також затримки 35 нс для кожної наступної операції.

Приклади застосування FPGA

Одна з переваг у використанні апаратних модулів зі змінюваною конфігурацією – пошук відповідності зразку (pattern matching), що використовується в задачах розпізнавання рукописного тексту, ідентифікації осіб, вибірки в базах даних і автоматичного розпізнавання цілей.

У основі пошуку відповідності лежить операція порівняння вхідного набору бітів (який представляє образ, послідовність символів або інші дані) з безліччю шаблонів (templates). Розпізнавання визначається в тому випадку, якщо число вхідних бітів, що збіглися з бітами в шаблоні, перевищить деяке граничне значення.

DARPA, агентство перспективних досліджень Міністерства оборони США, спонсорувало розробку прототипної системи

розпізнавання цілей з конфігурованим процесором. Основною проблемою цього прикладення є необхідність дуже швидкого порівняння вхідного образу з тисячами шаблонів. Шаблон може представляти, наприклад, вигляд визначеного транспортного засобу спереду або збоку. Вхідний образ звичайно містить тисячі пікселів, а розпізнавана ціль може знаходитись в будь-якому положенні усередині образу. Для розпізнавання цілі всі пікселі вхідного образу порівнюються з усіма пікселями кожного шаблона. Для того щоб це відбулося досить швидко, необхідна швидкість виконання в кілька трильйонів операцій за секунду.

Система розпізнавання, яку з використанням FPGA розробили в університеті штату Каліфорнія, ефективно вирішує цю задачу. Програмувальна матриця реконфігурується для кожного наступного шаблона. При цьому враховується той факт, що в типовому шаблоні більшість пікселів не вносять вклад у результат порівняння. FPGA просто ігнорує такі пікселі, чого не могла робити звичайна система, оскільки точки, які можна пропустити, змінюються від шаблона до шаблона.

Тими ж розроблювачами на основі конфігурованих схем була побудована прототипна система шифрування, у якій за допомогою FPGA реалізується стандарт шифрування даних DES. DES (Data Ecnryption Standard) використовує 56-розрядні ключі для шифрування 64-розрядних блоків даних і виконує шифрування в два етапи: визначення підключів і безпосередньо обробка даних. На першому етапі 56-розрядний ключ шляхом спеціальних перетворень розбивається на послідовність з 16 підключів. Потім кожен підключ обробляє дані в окремому циклі. Кожен 64-розрядний блок даних шифрується і дешифрується за допомогою 16 таких циклів.

У багатьох прикладеннях з використанням DES ключ шифрування не змінюється при передачі великого масиву даних, наприклад під час сеансу зв’язку двох користувачів по мережі з надійними засобами захисту. Користувачі спочатку обмінюються ключем шифрування, а потім використовують його в процесі діалогу для створення підключів для кожного циклу шифрування або дешифрування. У системі з конфігурованою матрицею програма обчислює значення підключей, для яких потім оптимізується FPGA обробки даних. Такий підхід дозволяє цілком виключити із системи апаратну частину визначення підключів. У результаті алгоритм DES реалізується конфігурованою схемою з 13 тис. логічних елементів, на

той час як схема ASIC для аналогічної задачі вимагає використання 25 тис. елементів. Якщо виникає необхідність змінити ключ шифрування, користувач може швидко перепрограмувати FPGA для нового ключа.

Розглянуті вище приклади розпізнавання цілі і шифрування показують, що використання конфігуруємої апаратури дозволяє досягти дуже високого рівня гнучкості в настроюванні комп'ютера на різні набори вхідних даних, що змінюються.

11.2.4. Технології майбутнього

На даний час при створенні та проектуванні пристроїв енергонезалежної пам'яті використовуються досить різні технології, у тому числі і досить "екзотичні", наприклад, з використанням аморфних матеріалів. При цьому використовується відмінність між кристалічним і аморфним станами речовини. Зокрема, на конференції "Суспільства з твердотільних пристроїв" розглядалася технологія енергонезалежної пам'яті OUV, спрямована на підвищення швидкодії і довговічності чипа пам'яті. Запис інформації в такі ЗП здійснюється шляхом переведення речовини з кристалічної форми в аморфну і навпаки. Як матеріал використовуються різні халкогениди. Робочі параметри і робочі характеристики такі: час очищення комірки пам'яті – 10 нс, запис в осередок – 50 нс, зносостійкість – порядку 10^{12} циклів. Інформація зберігається протягом 10 років при температурі не більше 120 ° С.

ЗП майбутнього – це ЗП, що називаються "польовими програмувальними пристроями". Вони базуються на одноелектронних модулях пам'яті (SEM) – новому типі EEPROM. Перший пристрій SEM було випущено фірмою Hitachi у 1993 р. У 1996 р. ємність SEM складала 64 Мбіти, у 1997 р. – 128 Мбітів. Основа побудови таких кристалів – природним чином сформовані структури мікрокристалів. Швидкість запису і стирання – 100 нс. Напруга живлення – ± 4 В. Ці характеристики аналогічні параметрам модулів DRAM (динамічні ОЗП). Час збереження інформації – порядку 10^9 циклів читання/запису $\approx 10^5$ с (~ 50 років).

Існує також технологія ЗП, заснована на штучно програмованих квантових точках. Ці точки (діаметр – 10^{-10}) розміщують в оксидному шарі замків МОП-транзисторів.

Час збереження інформації в таких ЗП при U = 2,5 В – 10^4 с (~ 30 років).

11.3. Статичні оперативні ЗП (SRAM)

11.3.1. Основні структури ЗП

Мікросхеми ОЗП являють собою ЗП, що зберігають N слів по n розрядів кожен. Одними із широко використовуваних минулого модулі пам'яті Intel 2102 (к × 1), вітчизняний аналог – КР132РУ4А (рис. 11.16). Ця мікросхема випускалася в корпусі з 16 виводами.

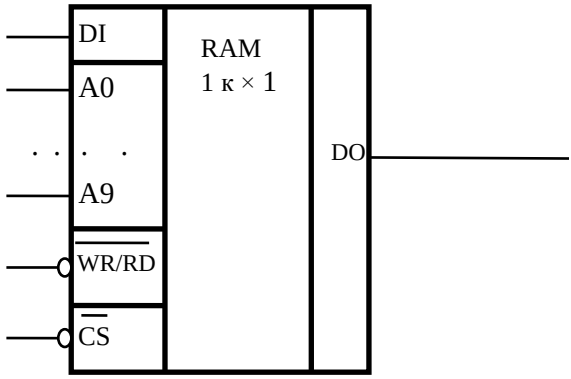


Рис. 11.16. Умовно-графічне позначення МС КР132РУ4А

Далі схема 2102 трансформувалася в 2142 з організацією 1 к × 4 (вітчизняна версія – К514РУ2).

У залежності від числа ліній вибірки (адресних ліній, що задають сукупність запам'ятовуючих елементів, і розрядних ліній, що дозволяють виділити окремий розряд), з'єднаних з одним запам'ятовуючим елементом, розрізняють такі типи ЗП:

– з лінійною або словниковою вибіркою елементів пам'яті (система 2D);

– з двохкоординатною вибіркою елементів пам'яті (3D);

– з комбінованою вибіркою елементів пам'яті (модифікована 2D або 2D-M).

11.3.1.1. Система 2D

Основу таких ЗП складає плоска матриця запам'ятовуючих елементів, згрупованих у $N = 2^m$ осередків по n розрядів. Звертання до осередку задається m-розрядною адресою, іноді здійснюється виділення окремих розрядів розрядними лініями запису і зчитування.

На схемі (рис. 3.17) введені такі позначення:

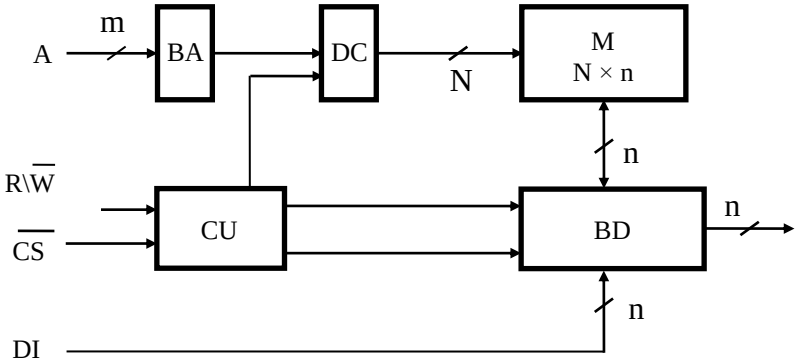


Рис. 11.17. Схема пам'яті типу 2D

A – m-розрядна адреса;

BA – буферний формувач адреси. Він формує парафазний код адреси (сама адреса та її інверсія);

DC (Decoder) – дешифратор;

CU (Control Unit) – пристрій управління;

BD (Bus Driver) – буферний формувач вхідних і вихідних даних, що виконує функцію формування і посилення інформаційних сигналів читання в режимі зчитування інформації.

Дешифратор адресного коду DC при наявності сигналу CS активізує одну з вихідних ліній, дозволяючи одночасний доступ до всіх елементів обраного рядка, що зберігає слово, адреса якого відповідає номеру рядка. Елементи одного стовпця з'єднані вертикальною лінією – внутрішньою лінією даних. Елементи стовпця зберігають однойменні біти всіх слів. Напрямок обміну визначається підсилювачами читання-запису під впливом сигналу R/W.

Достоїнства такої архітектури – простота схеми і досить висока швидкодія.

Недолік – складність побудови дешифратора з 2^m виходами. Тому така організація застосовується тільки в БІСА ОЗП малої ємності.

11.3.1.2. Система 3D

Матриця елементів пам'яті складається з n підматриць розміром q × q, рядки і стовпці кожної з яких зв'язані з виходами дешифратора по осях X і Y. Значення q знаходиться з формули:

$$q = \sqrt{N}\ .$$

При звертанні до такої ВІС вибираються n елементів пам'яті (по одному з кожної підматриці), що знаходяться на перетинанні рядка і стовпця, зв'язаних із виходами дешифраторів по осях X і Y (рис. 11.18).

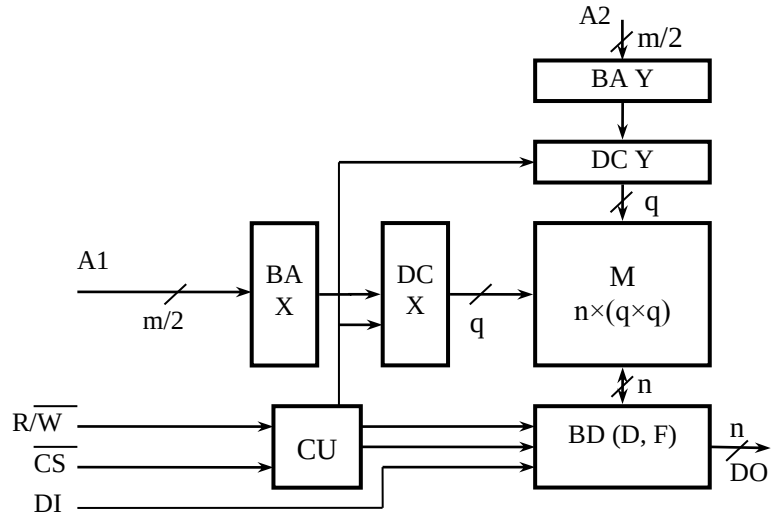


Рис. 11.18. Схема пам'яті типу 3D

Достоїнством такого рішення є використання двох більш простих дешифраторів замість одного складного. Справді, складність двох адресних формувачів ЗП типу 3D пропорційна $2^{\frac{k}{2}+1}$, що значно менше складності адресного формувача ЗП типу 2D, що пропорційна

2^k. Навіть для ЗП невеликої ємності помітно цю істотну різницю: для структури 2D при збереженні 1 К слів потрібен був би дешифратор з 1024 виходами, тоді як для структури типу 3D потрібні 2 дешифратори з 32 виходами кожен. У зв'язку з цим структура типу 3D дозволяє будувати ЗП більшої ємності, ніж структура 2D.

Недолік – більш складна реалізація елементів пам'яті, що повинна допускати двохкоординатну вибірку, що призводить до ускладнення структури матриці, у якій необхідно роз'єднати словникові і розрядні шини.

Структури типу 3D мають також досить обмежене застосування, оскільки в структурах типу 2DM (2D модифікована) поєднуються достоїнства обох розглянутих структур – спрощується дешифрація адреси і не потрібні запам'ятовуючі елементи з двохкоординатною вибіркою.

11.3.1.3. Система 2D-M

Як приклад розглянемо деякий ОЗП ємністю 1 К × 4 і відповідною розрядністю шин (рис. 11.19).

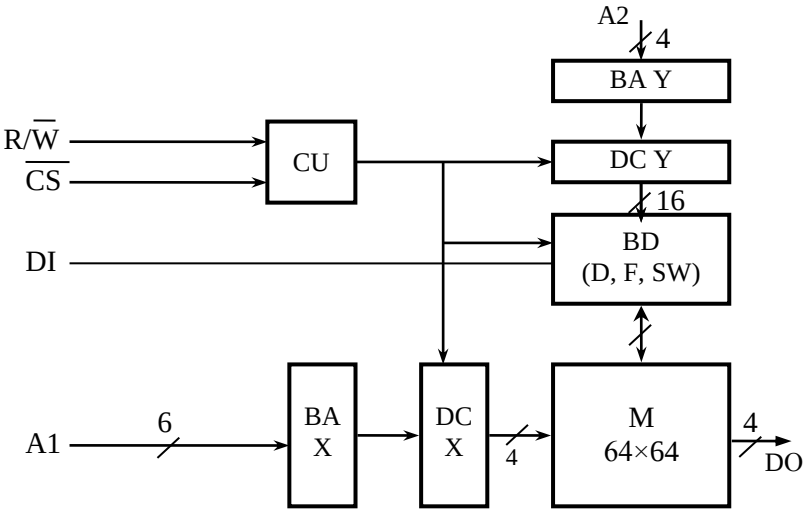


Рис. 11.19. Схема пам'яті типу 2D-M

Тобто N = 1024, n = 4. Матриця запам'ятовуючих елементів складається з q × q елементів пам'яті, $q = \sqrt{N \cdot n} = 64$.

Блок BD складається з n формувачів записів, m підсилювачів зчитування і ключових елементів Switch, управляємих блоком DC Y.

У режимі запису в розглянутому прикладі виходи 4-х формувачів запису за допомогою управляємих дешифратором DC Y ключових елементів підключаються до 4-х з 64 стовпців матриці елементів пам'яті M. При цьому здійснюється запис інформації в ці обрані 4 елементи пам'яті того рядка матриці-накопичувача, що відповідає активному виходу дешифратора DC X. Інші елементи пам'яті даного рядка знаходяться в стані збереження інформації.

У режимі зчитування за допомогою ключових елементів до виходів 4-х підсилювачів зчитування підключається 4 з 64 стовпців матриці елементів. При цьому здійснюється зчитування інформації з 4-х елементів пам'яті з того рядка, що порушена відповідним виходом дешифратора DC X. Інші елементи пам'яті даного рядка знаходяться в стані збереження інформації.

Т.ч. ЗП структури 2D-M для матриці запам'ятовуючих елементів з адресацією від дешифратора DC X має нібито характер структури 2D: збуджений вихід дешифратора вибирає цілий рядок. Однак на відміну від структури 2D, довжина рядка не дорівнює розрядності збережених слів, а багаторазово її перевищує. При цьому число рядків матриці зменшується і, відповідно, зменшується число виходів дешифратора.

Структура 2D-M найбільш зручна для побудови напівпровідникових ЗП і на даний час широко використовується як в оперативних, так і в постійних ЗП.

11.3.2. Елементи пам'яті ЗП статичного типу

На даний час елементи пам'яті ОЗП статичного типу реалізуються як на МДП-транзисторах, так і на біполярних транзисторах.

На рис. 11.20 показана спрощена схема біполярної комірки пам'яті. У такому осередку зберігається 1 біт інформації. Осередок реалізований з використанням технології багатоемітерної транзисторно-транзисторної логіки (ТТЛ). Як випливає зі схеми, комірка пам'яті являє собою не що інше, як звичайний тригер. Цей тригер може бути встановлений або в стан 1, або в стан 0 (скидання). Якщо тригер встановлений у стан 1, то це значення зберігається в

ньому доти, поки не буде здійснено скидання або не буде вимкнено живлення. Для утворення тригера два багатоемітерних транзистори охоплені зворотними зв'язками.

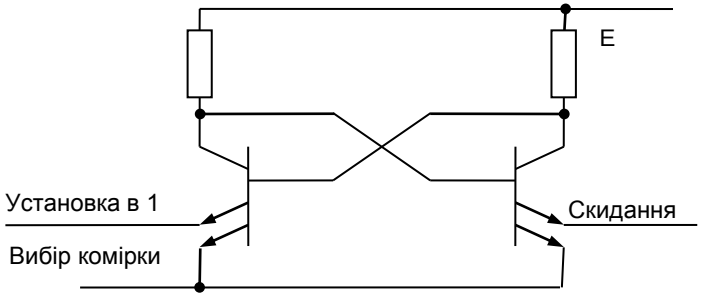


Рис. 11.20. Схема біполярної комірки пам'яті

Установка в стан 1 і скидання тригера здійснюється подачею сигналів на відповідні емітери. При звертанні до осередку по лінії вибору утвориться низькоімпедансний вихідний сигнал припустимого рівня.

На рис. 11.21 наведена спрощена схема осередку статичної МОП-пам'яті.

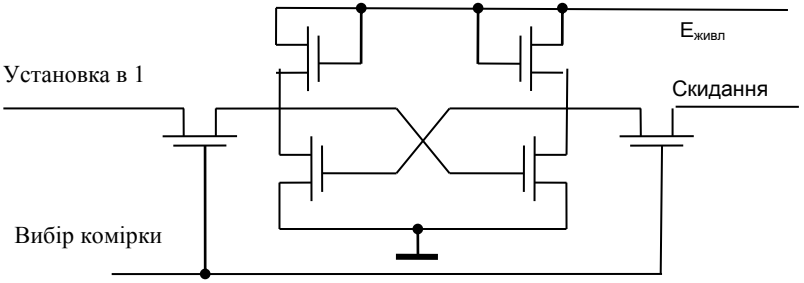


Рис. 11.21. Схема осередку статичної МОП-пам'яті

По суті, вона також являє собою тригер. Як і в більшості пристроїв на МОП-структурах, як навантажувальні опори тут використовуються МОП-транзистори з постійно зміщеними замками. Для введення і виведення інформації з осередку служить ще одна пара МОП-транзисторів. Як і у випадку біполярної комірки пам'яті,

запам'ятовування двійкової інформації в осередку статичної МОП-пам'яті забезпечується за рахунок перехресних зв'язків між двома логічними елементами. Значення 1 або 0 зберігається в осередку поки на нього подається напруга живлення.

Аналіз різних схем побудови пам'яті дозволяє зробити такі висновки: схеми ТТЛ відносяться до інтегральних логічних елементів середньої швидкодії. Час затримки сигналу складає 5 – 10 нс. Навантажувальна спроможність допускає підключення до виходу елемента до 10 логічних схем.

Більш високу швидкодію мають інтегральні мікросхеми з емітерними зв'язками (ЕЗЛ), у яких транзистори не входять у насичення. Елементи ЕЗЛ працюють за принципом переключення струмів при малих змінах вхідних напруг. Унаслідок цього елементи ЕЗЛ часто називають схемами з перемикачами струму.

Час затримки елемента ЕЗЛ менше, ніж елемента ТТЛ, і звичайно має значення 1 – 2 нс.

Інтегральні мікросхеми на МОП (метал-окисел-напівпровідник)-транзисторах є більш повільно діючими, ніж елементи ТТЛ або ЕЗЛ. Час затримки елемента на МОП-транзисторах звичайно 50 – 100 нс. Однак ці елементи відрізняються меншою споживаною потужністю, великою навантажувальною спроможністю і завадостійкістю, що особливо важливо, вимагають меншої площі на поверхні інтегральної мікросхеми. Схеми на МОП-транзисторах технологічні й дешеві.

МОП-транзистори бувають *n*- і *p*-типів. Будують схеми і з одночасним використанням транзисторів *n*- і *p*-типів. Схеми з транзисторами, що доповнюють, (К-МОП-схеми) відрізняються малою споживаною потужністю і більш високою швидкодією (10 – 50 нс), тому що в ланцюгах заряду і розряду паразитних ємностей схеми виявляються включеними малі опори відкритих транзисторів.

11.3.3. Система електричних параметрів виробів електронної техніки

Система електричних параметрів виробів електронної техніки необхідна для класифікації їхніх функціональних можливостей, контролю якості, взаємозамінності та функціональної сумісності.

Існують відповідні стандарти, що містять перелік параметрів на БІСА ОЗП, до яких відносяться:

- функції призначення і тип логіки;

- інформаційна ємність і організація;
- питома потужність споживання (на 1 біт);
- тип корпуса;
- число циклів перезапису (для EEPROM);
- час збереження інформації при відключеному джерелі живлення;
- степінь інтеграції;
- номінальна напруга живлення і додаткове відключення;
- робочий діапазон температур;
- граничні значення прискорення.

Система параметрів розбивається на статичні і динамічно-статичні. До статичних параметрів відносять значення струму і напруги (беруться конкретні контрольні точки). Система динамічних параметрів описує загальну часову діаграму роботи БІС ЗП. На ній для кожного з динамічних параметрів визначаються початок і точки відліку, напрямок в абсолютних одиницях.

Наприклад: для сигналів у TTL-логіці рівень відліку дорівнює 1,5 V, низький – 0,8 V, високий – 2,2 V.

11.3.4. Часові діаграми ОЗП

Часова діаграма операції "Читання" наведена на рис. 11.22.

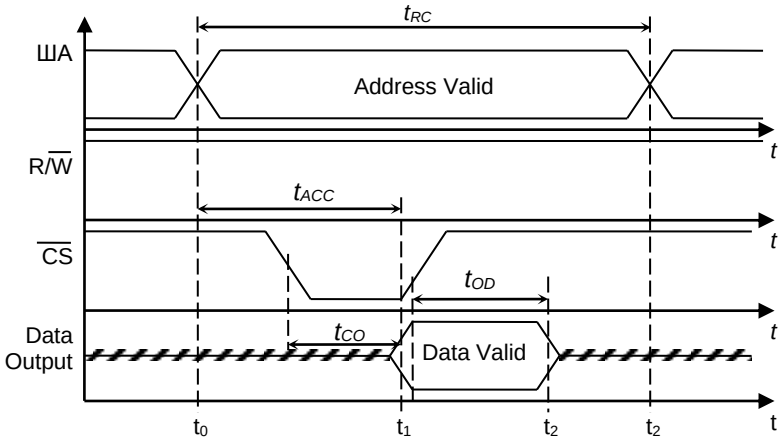


Рис. 11.22. Часова діаграма операції "Читання"

Розгляд діаграми починається з моменту часу t_0, коли на ША виставляється нове значення, що зберігається протягом всієї операції зчитування. Закінчується цикл читання в момент часу t_3, при встановленні наступної адреси.

$$t_3 - t_0 = t_{RC} \text{ (Read cycle).}$$

Сигнал лінії R/W незмінний протягом усього циклу, відповідає рівню High.

$t_{ACC} = t_1 - t_0$ – час доступу – інтервал часу, необхідний для встановлення дійсного значення вихідних даних з моменту появи нової адреси.

З деякою затримкою після встановлення дійсного значення адреси сигнал CS переходить з High у Low. Низький рівень сигналу CS вказує на те, що даний кристал обраний і через інтервал часу t_{CO} на ШД з'являється дійсне значення (DATA VALID).

$[t_1; t_2]$ – інтервал, який відповідає періоду, що відповідає дійсному значенню на ШД.

t_{DO} – період, що визначає затримку повернення вихідних значень буферних каскадів у Z-стан з моменту переходу CS назад у 1.

Цикл запису починається в момент t_0 і завершується в момент t_4 (рис. 11.23).

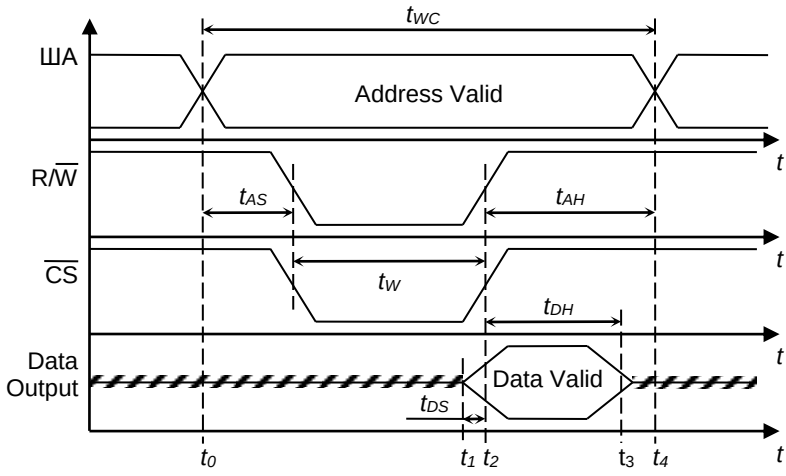


Рис. 11.23. Часова діаграма операції "Запис"

Інтервал t_{WC} визначає тривалість циклу запису:

t_{AS} – address setup time – затримка на встановлення адреси на ША;

t_{DS} – data setup time – часовий інтервал, протягом якого дані на ШД не повинні змінюватися, поки CS і R/W не повернуть в High;

t – період, що відповідає низькому рівню сигналів CS і R/W. Така комбінація визначає операцію запису для даного кристала;

t_{DH} – інтервал часу, при якому дані ще дійсні, а рівень управляючих сигналів відповідає логічній 1;

t_{AH} – address hold time – інтервал часу незмінних даних на ША після переходу управляючих сигналів у 1.

11.4. Динамічні оперативні ЗП (DRAM)

11.4.1. Елементи пам'яті DRAM

Кожен осередок динамічної ЗП містить усього лише один КМОП-транзистор (комплементарні польові транзистори). Запам'ятовуючим елементом у них є конденсатор, що може знаходитись в зарядженому або розрядженому стані (рис. 11.24). Якщо конденсатор заряджений, то в осередок записана логічна 1. Якщо конденсатор розряджений, то в осередок записаний логічний 0.

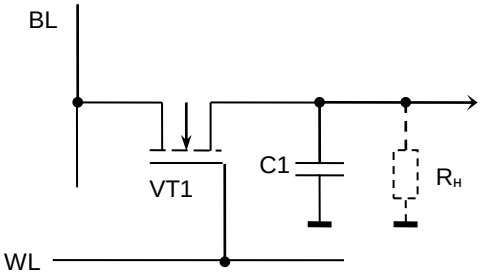


Рис. 11.24. Елемент пам'яті DRAM

Запис інформації в такий елемент пам'яті здійснюється одночасною подачею напруги на адресну (WL) і розрядну та (BL) шини. Напруга на шині WL відкриває транзистор і конденсатор Із заряджається, якщо на шині BL напруга високого рівня (код логічної одиниці) або розряджається, якщо на шині BL напруга низького рівня (код логічного нуля).

При зчитуванні напруга подається тільки на адресну шину. При цьому напруга на конденсаторі передається на вхід підсилювача зчитування.

Фрагмент ЗП (рис. 11.25) показує ЗЕ, підсилювач зчитування ВУС, а також ключі К1 і К0 відповідно запису одиниці і нуля. До лінії запису-зчитування (ЛЗС) = (BL) підключено стільки ЗЕ, скільки рядків мається в запам'ятовуючій матриці. Особливе значення має ємність ЛЗС $C_л$, у силу великої довжини лінії і великого числа підключених до неї транзисторів багаторазово перевищуюча ємність ЗЕ.

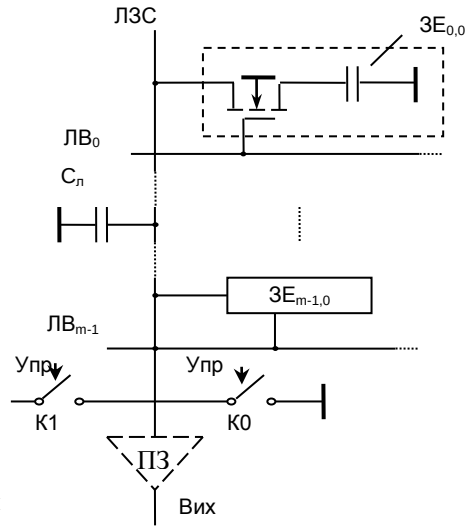


Рис. 11.25. Фрагмент схеми динамічного ЗП

Перед зчитуванням здійснюється передзаряд ЛЗС. Маються варіанти ЗП з передзарядом ЛЗС до рівня напруги живлення і до рівня його половини.

Розглянемо останній варіант у зв'язку з його більшою схемною простотою. Отже, перед зчитуванням ємність $C_л$ заряджається до рівня $U_{cc}/2$. Будемо вважати, що збереження одиниці відповідає зарядженій ємності $З_З$, а збереження нуля – розрядженій.

При зчитуванні нуля до ЛЗС підключається ємність $З_З$, що мала нульовий заряд. Частина заряду ємності $C_л$ перетікає в ємність $З_З$, і напруги на них зрівнюються. Потенціал ЛЗС знижується на величину ΔU, що і є сигналом, який надходить на підсилювач зчитування. При зчитуванні одиниці, навпаки, напруга складала спочатку величину U_{cc} і перевищувала напругу на ЛЗС. При підключенні $З_З$ до ЛЗС частина заряду стікає з запам'ятовуючої ємності в $C_л$ і напруга на ЛЗС збільшується на ΔU.

Якщо обчислити значення ΔU, то можна переконатися, що цей сигнал украй слабкий, до того ж зчитування відбувається з руйнуванням – підключення запам'ятовуючої ємності до ЛЗС змінює її заряд.

Графіки сигналів при зчитуванні нуля й одиниці показані на рис. 11.26.

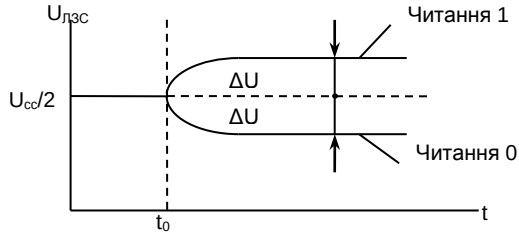


Рис. 11.26. Часові діаграми сигналів
при зчитуванні даних у динамічних ЗП

Мірами подолання відзначених недоліків служать способи збільшення ємності $З_З$ (без збільшення площі ЗЕ), зменшення ємності ЛЗС і застосування підсилювачів-регенераторів для зчитування даних.

У напрямку збільшення $З_З$ можна вказати розробку фірмою Сіменс діелектрика (двоокису титана TiO_2), що має діелектричну постійну в 20 разів більшу, ніж Si_2. Це дозволяє при тій же ємності скоротити площу ЗЕ майже в 20 разів або збільшити $З_З$ навіть при зменшенні її площі. Маються і варіанти з введенням у ЗЕ струмопідсилюючих структур, що також еквівалентно збільшенню ємності ЗЕ.

Зменшення ємності ЛЗС можна досягти “розрізуванням” цієї лінії на дві половини із включенням диференціального підсилювача зчитування в розрив між половинами ЛЗС (рис. 11.27).

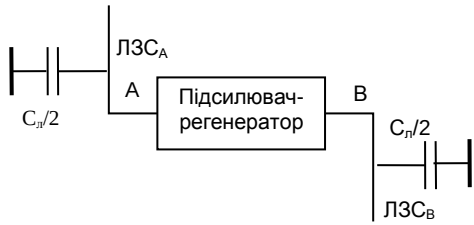


Рис. 11.27. Схема включення підсилювача-регенератора
в розрив лінії зчитування-зчитування-запису-
зчитування динамічного ЗП

Очевидно, що такий прийом удвічі зменшує ємність ліній, до яких підключаються запам'ятовуючі ємності, тобто вдвічі збільшує сигнал ΔU.

Елемент пам'яті динамічного ЗП має як свої достоїнства, так і недоліки.

Достоїнства: для такого елемента пам'яті характерно мале споживання потужності, до того ж він значно простіше тригерного, що містить 6 транзисторів і дозволяє розмістити на кристалі набагато більше ЗЕ (у 4..5 разів) та забезпечує динамічним ЗП максимальну ємність.

Недоліки:

1. Як було розглянуто вище, зчитування відбувається з руйнуванням – підключення запам'ятовуючої ємності до ЛЗС змінює її заряд. Т.ч. після відповідного циклу читання необхідно зробити повторний запис інформації, а це збільшить цикл пам'яті.

2. Конденсатор неминуче втрачає з часом свій заряд, і збереження даних вимагає їхньої періодичної регенерації (через декілька мілісекунд).

11.4.2. Регенерація пам'яті

У ідеальному конденсаторі заряд може зберігатися нескінченно довго. У реальному конденсаторі існує струм витоку, тому записана в динамічну пам'ять інформація згодом буде втрачена, тому що конденсатори запам'ятовуючих елементів цілком розрядяться. Внаслідок цього необхідна періодична регенерація пам'яті (Refresh), причому процесор має доступ до необхідних даних у пам'яті тільки в цикли, вільні від регенерації.

Єдиним способом регенерації збереженої в пам'яті інформації є виконання операції запису або читання даних з пам'яті. Якщо інформація заноситься в динамічну пам'ять, а потім протягом декількох мілісекунд залишається незатребуваною, вона буде втрачена, тому що конденсатори запам'ятовуючих елементів цілком розрядяться.

Регенерація пам'яті відбувається при виконанні кожної операції читання або запису даних в оперативну пам'ять. При виконанні будь-якої програми не можна гарантувати, що відбудеться звертання до всіх комірок пам'яті. Тому є спеціальна схема, що через визначені проміжки часу (наприклад, кожні 2 мс) буде здійснювати доступ (для зчитування) до всіх рядків пам'яті. У ці моменти процесор знаходиться

в стані чекання. За один цикл схема здійснює регенерацію всіх рядків динамічної пам'яті.

11.4.3. Пристрій і функціонування DRAM

На рис. 11.28 наведена блок-схема кристала пам'яті динамічного типу.

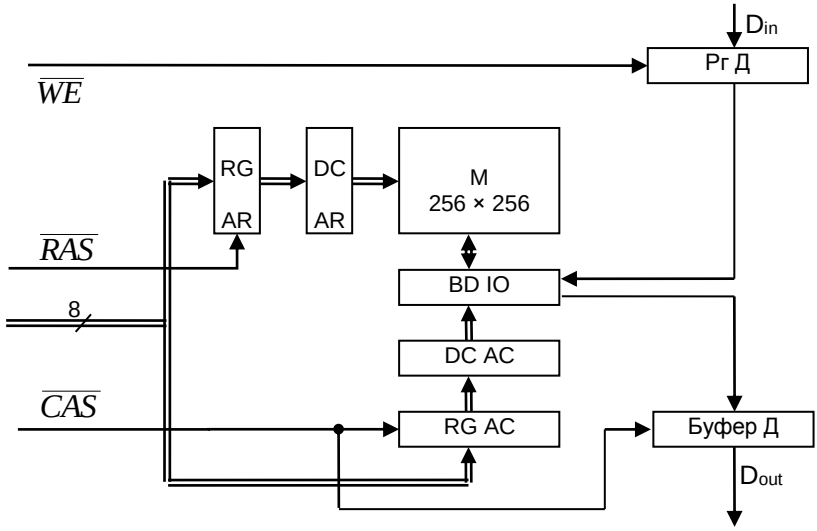


Рис. 11.28. Блок-схема кристала пам'яті динамічного типу

Особливістю динамічних ЗП є мультиплексування шини адреси. Адреса поділяється на дві півадреси, одна із яких представляє собою адресу рядка, а інша – адресу стовпця матриці ЗЕ. Півадреси подаються на ті самі виводи корпуса ІС по черзі.

Подача адреси рядка супроводжується відповідним стробом RAS (Row Address Strobe), а адреси стовпця – стробом CAS (Column Address Strobe). Причиною мультиплексування адрес служить прагнення зменшити число виводів корпуса ІС і тим самим здешевіти її, а також та обставина, що півадреси і сигнали RAS і CAS у деяких режимах і схемах використовуються по-різному (наприклад, у режимах регенерації адреса стовпця взагалі не потрібна). Скорочення числа зовнішніх виводів корпуса для динамічних ЗП особливо актуально, тому що вони мають

максимальну ємність і, отже, велику розрядність адрес. Наприклад, ЗП з організацією 16 М × 1 має 24-розрядну адресу, а мультиплексування скоротить число адресних ліній на 12.

Т.ч. між процесором і ЗП повинен знаходитись мультиплексор, що розділяє сигнали на ША (рис. 11.29).

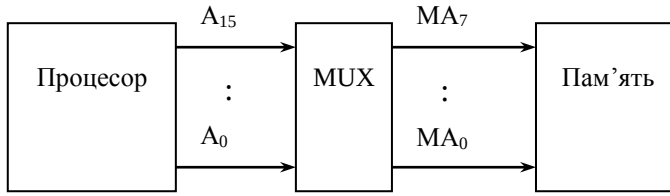


Рис. 11.29. Схема з'єднання процесора та пам'яті

На рис. 11.30 наведені часові діаграми роботи динамічної пам'яті.

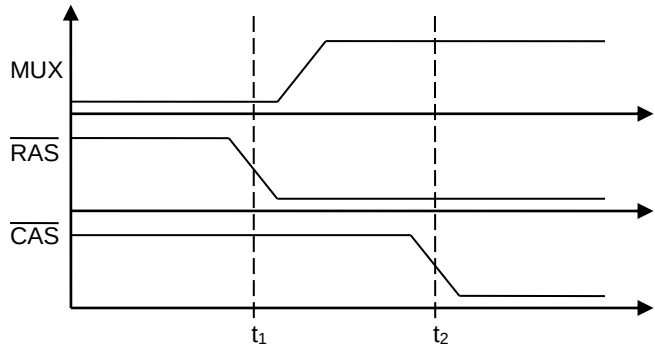


Рис. 11.30. Часові діаграми роботи динамічної пам'яті

На момент часу t_1 (MUX = 0) у RGAR записується адреса рядка на момент часу t_2 (MUX = 1) у RGAC записується адреса стовпця. Конкретний приклад мультиплексора 74LS257 наведено на рис. 11.31.

Для пам'яті динамічного типу передбачені такі режими роботи як збереження, звертання (зчитування або запис), регенерація, а також запис-зчитування-модифікація-запис.

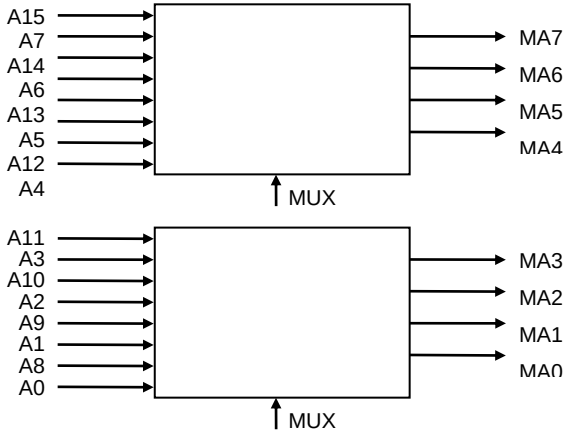


Рис. 11.31. Приклад мультиплексора 74LS257

При звертанні до пам'яті (незалежно від того, читання це або запис) на входи пам'яті подана адреса рядка і сигнал RAS. Оскільки інформація зберігається у вигляді заряду конденсатора, то для того, щоб зчитувати записану в осередку інформацію, необхідний пристрій з високим вхідним опором, що обмежує струм розряду конденсатора, щоб уникнути струму витоку. Таким пристроєм є підсилювач, що зчитує, підключений до кожної загальної шини стовпця динамічної пам'яті. Інформація зчитується з усього рядка запам'ятовуючих елементів одночасно і розміщується в регістрі.

З деяким запізнюванням щодо сигналу RAS на входи динамічної пам'яті подається адреса стовпця і сигнал CAS. При читанні відповідно до адреси стовпця дані вибираються з регістра рядка і подаються на вихід динамічної пам'яті.

При зчитуванні інформації з запам'ятовуючих комірок підсилювачі, що зчитують, руйнують її, тому для збереження інформації необхідний її перезапис. Якщо запам'ятовуюча комірка мала заряд, то вона знову буде заряджена ще до завершення циклу читання. На комірки, що не мали заряду, напруга не подається.

Якщо виконується цикл запису в пам'ять, то подається сигнал WE (Write Enable), що дозволяє запис у регістр вхідних даних. Таким чином, проходження даних при записі визначається комбінацією сигналів адреси стовпця і рядків та дозволами запису даних у пам'ять.

При записі дані з регістра рядка на вихід (DO) не надходять (мається буфер даних, що визначає підключення виходу D_{OUT} за допомогою управляючого сигналу CAS).

Регенерація інформації здійснюється для всіх елементів пам'яті, розміщених на даному рядку матриці М. Тому для неї досить подачі тільки сигналу RAS разом з адресами регенерування рядків. Для регенерації всієї пам'яті М потрібно стільки циклів, скільки рядків мається в цій матриці.

Режим роботи "запис-зчитування-модифікація-запис". У цьому режимі протягом одного циклу звертання до ЗП за однією адресою спочатку зчитується код з обраної комірки пам'яті, отриманий біт інформації передається на вихід D_{OUT}, а потім у цей же ЕП записується код із входу D_{IN} мікросхеми. Тривалість циклу режиму "запис-зчитування-модифікація-запис" більше циклів запису і зчитування, але менше їхньої суми, тому час на корекцію вмісту ЗП скорочується.

11.4.4. Часові діаграми роботи пам'яті динамічного типу

Розглянемо часові діаграми операцій запису, зчитування і регенерації на прикладі мікросхеми К565РУ3.

Часові діаграми роботи пам'яті динамічного типу при виконанні операції ***запису*** наведені на рис. 11.32.

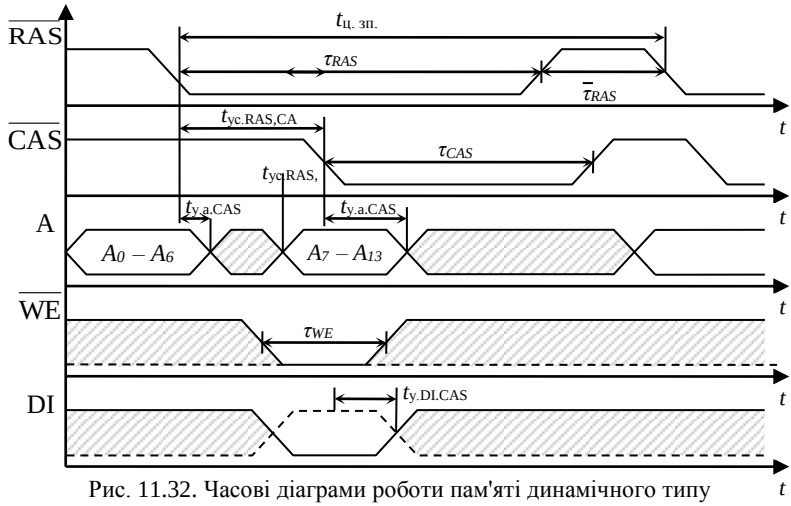


Рис. 11.32. Часові діаграми роботи пам'яті динамічного типу при виконанні операції запису

Для звертання до мікросхеми для запису і зчитування інформації необхідно подати код адреси рядків $A_{0\ -\ A6}$, одночасно з ним або з якоюсь (не нормується) затримкою сигнал RAS, потім з нормованою затримкою на час утримання адреси рядків щодо сигналу RAS повинен бути поданий код адреси стовпців і через час установлення $t_{ус.а.CAS}$ – сигнал CAS.

До моменту подачі коду адреси стовпців на вхід DI підводять записуваний біт інформації, що сигналом WE при наявності CAS = 0 фіксується на вхідному тригері-засувці. Сигнал запису WE може бути поданий рівнем або імпульсом. У останньому випадку він повинен мати тривалість не менш визначеного параметром τ_{WE} значення. Якщо сигнал запису поданий рівнем, то фіксація DI тригером-засувкою робить від'ємний перепад сигналу CAS (при наявності RAS = 0). По закінченні запису повинна бути витримана пауза τ_{WE}, рівна інтервалу між сигналами RAS, для відновлення стану внутрішніх ланцюгів мікросхеми.

Часові діаграми роботи пам'яті динамічного типу при виконанні операції ***зчитування*** наведені на рис. 11.33.

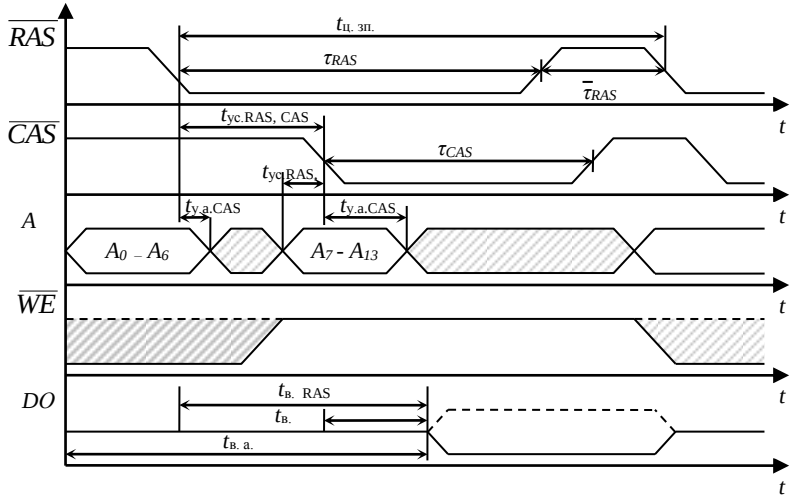


Рис. 11.33. Часові діаграми роботи пам'яті динамічного типу при виконанні операції зчитування

Адресні та управляючі сигнали повинні бути подані в порядку, аналогічному операції запису. Сигнал WE = 1 може бути поданий імпульсом або рівнем. Час появи вихідного сигналу можна відраховувати від моменту надходження сигналів адреси $t_{в.а}$ або сигналів управління, час вибірки сигналу RAS $t_{в.RAS}$, час вибірки сигналу CAS $t_{в.CAS}$.

При оцінці мікросхеми за цими параметрами варто мати на увазі, що вони взаємозалежні, і тому досить знати один з них. Більш інформативним є параметр $t_{в.CAS}$, оскільки інформацію виводить з мікросхеми сигнал CAS при наявності, звичайно, сигналу зчитування WE = 1.

З рис. 11.33 випливає:

$$t_{в.RAS} = t_{в.CAS} + t_{ус.CAS.\ RAS},$$

де параметр $t_{ус.CAS.\ RAS}$ установлює взаємний зсув за часом сигналів RAS і CAS.

Для оцінки швидкодії мікросхеми пам'яті в розрахунок необхідно приймати час циклу запису (зчитування) $t_{ц.зп.}$, $t_{ц.зч.}$.

Інші часові параметри необхідні для забезпечення вірного функціонування мікросхем у складі електронної апаратури. Перелік часових параметрів динамічних ОЗП включає десятки найменувань.

Часові діаграми роботи пам'яті динамічного типу при виконанні операції ***регенерації*** наведені на рис. 11.34.

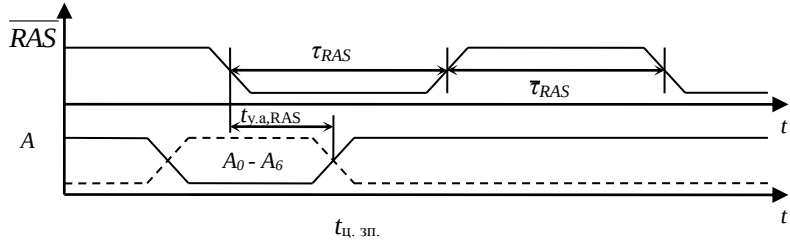


Рис. 11.34. Часові діаграми роботи пам'яті динамічного типу при виконанні операції регенерації

Як уже було сказано, для забезпечення надійного збереження записаної в накопичувачі інформації реалізують режим примусової регенерації.

При її організації найбільш доцільним і зручним для реалізації є режим регенерації сигналом RAS, при якому здійснюють перебір адрес у супроводі стробуючого сигналу RAS при CAS = 1.

11.4.5. Сучасні технології реалізації пам'яті динамічного типу

11.4.5.1. Традиційна пам'ять DRAM

У традиційній пам'яті сигнали RAS# і CAS#, що обслуговують запам'ятовуючі осередки, вводяться безпосередньо по відповідних лініях інтерфейсу. Вся послідовність процесів у пам'яті прив'язується саме до цих зовнішніх сигналів. Дані при читанні будуть готові через час T_{CAC} після сигналу RAS#, але не раніш, ніж через T_{RAC} після сигналу RAS#. Тому традиційна динамічна пам'ять є дещо повільною (час вибірки 70 – 80 нс).

11.4.5.2. FPM DRAM

На основі стандартних осередків будується пам'ять зі швидким сторінковим доступом – FPM (Fast Page Mode**)** DRAM. Тут для доступу до осередків, розміщених у різних колонках одного рядка, використовується всього один імпульс RAS, під час якого виконується серія звертань за допомогою тільки імпульсів CAS. Неважко здогадатись, що в пакетних циклах доступу виходить виграш у часі. Так, пам'ять FPM з часом доступу 60 – 70 нс при частоті шини 66 МГц може забезпечити цикл читання 5-3-3-3 (це означає, що для звертання до першого біта даних необхідно 5 тактів, а для наступних – по 3 такти). Поки не змінився номер сторінки, у циклах обміну вилучені деякі етапи, що скорочує тривалість циклів.

Режим FPM – початок лінії розвитку методів підвищення швидкодії динамічних ЗП. За швидкодією його можливості вже набагато перевищені більш пізніми розробками, проте метод FPM знаходить свою область застосування, і відповідні ЗП дотепер займають досить великий сектор ринку.

Додаткові засоби для організації режиму FPM прості: потрібно лише перевіряти приналежність чергової адреси поточній сторінці (рядку), що дозволяє виконувати цикл сторінкового режиму. В противному разі потрібне виконання звичайного (повного) циклу.

11.4.5.3. EDO DRAM

Починаючи з 1995 року, у комп'ютерах на основі Pentium використовується новий тип оперативної пам'яті – EDO. Ще його

іноді називають Hyper Page Mode. Пам'ять типу EDO була розроблена і запатентована фірмою Micron Technology. Структури типу EDORAM (Extended Data Out RAM, тобто ОЗП з розширеним виводом даних) близькі до структур FPM і відрізняються від них модифікацією процесу виводу даних. У EDORAM дані в підсилювачах-регенераторах не скидаються по закінченні строба CAS. У мікросхеми пам'яті були введені регістри-засувки, тому дані, що зчитуються, присутні на виході навіть після підйому CAS. Це дозволило не чекати, поки зовнішня схема прийме дані, а, отже, скоротити час дії CAS. Таким способом можна прискорити передачу даних усередині пакета і на тих же комірках пам'яті одержати цикл 5-2-2-2. При цьому в FPM у другому і наступному доступах до сторінки потрібно 3 такти: 1) переключення CAS в активний стан; 2) зчитування даних; 3) переключення CAS у пасивний стан.

Розроблені EDORAM допускають роботу на частотах до 50 МГц. Такі ЗП одержали широке поширення, зокрема через тісний зв'язок з розробленими раніше ЗП типу FPM, заміна яких на EDORAM вимагає лише невеликих змін у схемі і синхросигналах ЗП.

11.4.5.4. BEDO DRAM

Архітектура BEDO була запропонована VIA Technologies. У цій технології поряд зі збереженням технології FPM і EDO використовується пересилання даних пакетами. У структурі типу BEDORAM (Burst EDORAM, тобто з пакетним розширеним доступом) мається додатково лічильник адрес стовпців. При звертанні до групи слів (пакетів) адреса стовпця формується звичайним способом тільки на початку пакетного циклу. Для наступних передач адреси утворюються швидко за допомогою інкрементування лічильника.

Характерна пропорційність часу першого і наступного звертань 5-1-1-1 (мається на увазі часто застосовуваний варіант із довжиною пакета, рівною 4).

Перераховані вище типи пам'яті є асинхронними стосовно тактовання системної шини комп'ютера. Це означає, що всі процеси ініціюються тільки імпульсами RAS і CAS, а завершуються через якийсь визначений (для даних мікросхем) інтервал. Протягом цих процесорів шина пам'яті виявляється зайнятою, причому, в основному, чеканням даних.

Тому не дивлячись на свої достоїнства в порівнянні з попередниками пам'ять типу BEDO RAM не одержала широкого поширення через появу сильного конкурента – синхронних DRAM (SDRAM), у яких не тільки досягається пропорційність часу звертань 5-1-1-1, але і сам час істотно скорочується.

11.4.5.5. Синхронна DRAM (SDRAM)

Синхронна оперативна пам'ять (SDRAM) – це перша технологія оперативної пам'яті з випадковим доступом (DRAM), розроблена для синхронізації роботи пам'яті з тактами роботи центрального процесора (1994 рік). Спочатку SDRAM була запропонована в якості більш дешевої за вартістю альтернативи для дорогої відеопам'яті VRAM (Video RAM), використовуваної в графічних підсистемах. Проте, вона швидко набула застосування в багатьох прикладеннях і стала кандидатом номер один на роль основної пам'яті для наступних поколінь PC, хоча споконвічно використання її гальмувалося високою (на 33%) ціною в порівнянні з EDO RAM. "Зоряний час" SDRAM настав у 1997 році, після появи чипсета 440BX, що працює на частоті 100 МГц. Унаслідок цього частка ринку SDRAM за рік виросла в два рази (з 25% у 1997 році до 50% у 1998 році). Модулі SDRAM досягли частот 133 МГц. Також розроблені SDRAM на частоти 143 МГц і вище.

Відмітні риси SDRAM-пам'яті:

- синхронний метод передачі даних на шину;
- конвеєрний механізм пересилання burst пакета;
- використання декількох (від 2 до 4-х) внутрішніх банків пам'яті;
- передача частини функцій контролера пам'яті логіці, розміщеній в самій інтегральній схемі.

Характерні риси SDRAM-пам'яті:

- При синхронній роботі дані на виході ІС з'являються разом з тактовими імпульсами, тому немає часової неузгодженості в роботі різних пристроїв, які беруть участь у передачі даних, що спрощує їхню взаємодію.
- На відміну від простого механізму передачі пакетів, реалізованого в BEDO, конвеєр дозволяє передати весь пакет по тактах і перевага даного способу особливо виявляється при збільшенні довжини пакета з 4-х слів до величини всього рядка банку.

- Ще більше зростання швидкодії досягається за рахунок використання поділу масивів осередків на незалежні внутрішні банки пам'яті. Оскільки ці банки можуть бути задіяні одночасно, безперервний потік даних може забезпечуватися простим переключенням між ними. Цей метод називається чергуванням і він дозволяє знизити загальну кількість циклів звертання до пам'яті та збільшити, в результаті, швидкість передачі даних.

SDRAM виробляється на основі стандартної DRAM і працює подібним чином – здійснюючи доступ до рядків і стовпчиків осередків даних. Тільки SDRAM поєднує свої специфічні властивості синхронного функціонування банків осередків і пакетної роботи для ефективного усунення станів затримки-чекання. Коли процесору необхідно одержати дані з оперативної пам'яті, він може одержати їх у необхідний момент. Таким чином, фактичний час обробки даних безпосередньо не змінився, на відміну від збільшення ефективності вибірки і передачі даних. У табл. 11.2 наведено порівняльний аналіз деяких характеристик розглянутих типів DRAM.

Таблиця 11.2

Порівняльний аналіз деяких характеристик розглянутих типів DRAM

	FPM	**EDO**	**BEDO**	**SDRAM**
Специфікація*	5, 6, 7	5, 6, 7	5, 6, 7	10, 12, 15
Час доступу (ns)	50, 60, 70	50, 60, 70	52, 60, 70	50, 60, 70
Час циклу (ns)	30, 35, 40	20, 25, 30	15, 16.6, 20	10, 12, 15
Макс. швидкість (MHz)	33, 28, 25	50, 40, 33	66, 60, 50	100, 80, 66

* Джерело: EDN, 4 Jan 1996
[специфікація для DRAM вказує час доступу (ns × 10)]
[специфікація для SDRAM вказує час циклу (ns)]

Час доступу (команди за адресою до вибору даних) однаковий для всіх типів пам'яті, як видно з табл. 3.2, оскільки їхня внутрішня архітектура в основному однакова. Більш показовим параметром є час циклу, що показує, наскільки швидко можуть бути здійснені два послідовних доступи в чипі. Перший цикл зчитування однаковий для всіх чотирьох типів пам'яті – 50, 60 або 70 нс. Але реальні розходження можна побачити, подивившись, як швидко здійснюється

другий, третій, четвертий і т.д. цикл зчитування. Для цього ми подивимося на час циклу. Для “6” FPM DRAM (60ns), другий цикл може бути здійснений за 35 нс. Порівняйте це з “12” SDRAM (час доступу 60 нс), коли другий цикл зчитування проходить за 12 нс. Це в три рази швидше, і при цьому, без будь-якої значної переробки системи.

На рис. 11.35 наведена блок-схема пам’яті SDRAM, запропонована фірмою Hitachi.

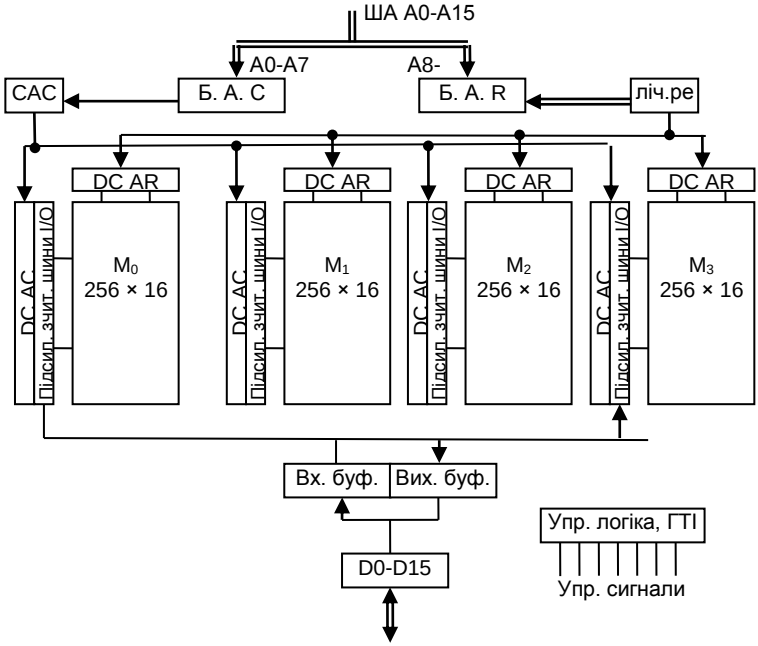


Рис. 11.35. Блок-схема пам'яті SDRAM, запропонована фірмою Hitachi

На рис. 11.35 наведені такі скорочення:
DC AR – дешифратор адреси рядка;
DC AC – дешифратор адреси стовпця;
САС – лічильник адреси стовпця;
Б.А.З – буфер адреси стовпця;
Б.А.R – буфер адреси рядка;
ГТЧ – генератор тактових частот;
ліч. рег. – лічильник регенерації.

***Найбільш значимі поліпшення продуктивності при використанні SDRAM*:**

- Більш швидка і більш ефективна – майже в чотири рази продуктивніше, ніж стандартна DRAM;
- Потенційно може замінити більш дорогу у використанні комбінацію EDO/L2-кеш;
- "При синхронному" функціонуванні – звільняє від обмежень за часом і не гальмує роботу новітніх процесорів;
- Внутрішнє чергування операцій з подвійними банками сприяє безперервному потоку даних;
- Можливість пакетного режиму роботи аж до повної сторінки (використовуючи до x16 мікросхем);
- Конвеєрна адресація дозволяє здійснювати доступ до інших запитаних даних, до завершення обробки даних, запитаних першими.

11.4.5.6. DDR DRAM

DDR DRAM (Double Date Rate DRAM) є синхронною пам'яттю, що реалізує подвоєну швидкість передачі даних у порівнянні зі звичайною SDRAM. Фактично це (SDRAM II). За принципом архітектури організації роботи ця пам'ять схожа з SDRAM, але відмінність полягає в тому, що прийом-передача даних здійснюється по обох фронтах тактових імпульсів, що дозволяє подвоїти передачу даних. На високих тактових частотах (100 МГц) подвійна синхронізація пред'являє дуже високі вимоги до точності витримування часових діаграм.

Для підвищення точності синхронізації вжиті такі заходи:

- Сигнал синхронізації мікросхеми подається в диференціальній формі, що дозволяє знизити вплив зсуву рівнів на точність синхронізації;
- Для синхронізації даних в інтерфейс уведений новий двонаправлений стробуючий сигнал DQS. Строби генеруються джерелом даних: при операціях читання DQS генерується мікросхемою пам'яті, при записі – контролером пам'яті (чипсетом);
- Для синхронізації DQS із системною тактовою частотою (CLK) мікросхеми мають убудовані схеми DLL (Delay Locked Loop). Цей протокол дозволяє зрушити в часі інтервал, протягом якого вихідні дані є Valid;

➢ Пропускна спроможність DDR DRAM – 1,6 Гб/с при частоті шини 100 МГц. На даний час вона розширюється до 2,4 Гб при частоті 150 МГц. Напруга живлення більш низька в порівнянні з раніше розглянутими типами DRAM, а саме 1,8 В.

11.4.5.7. SLDRAM

SLDRAM розробляється консорціумом виробників модулів пам'яті – SLDRAM Consortium. Вважається, що застосування SLDRAM економічно вигідно при ємності ОЗП не менше 256 Мбайтів. Цей тип пам'яті "увібрав" усі прогресивні технології, закладені в його попередниках – SDRAM і DDR RAM. Ще більше підвищення продуктивності досягається за рахунок поширення пакетного протоколу передачі даних на сигнали управління (звідки і пішла назва цього типу пам'яті – Linked SDRAM). У SLDRAM адреси, команди, а також сигнали управління передаються в пакетному режимі по однонаправленій шині Command Link.

Одночасно з ними по іншій, двонаправленій шині Data Link, і теж у пакетному режимі, передаються дані, причому передача відбувається на обох фронтах тактових імпульсів, як і у випадку з DDR SDRAM. Величина всього пакету даних може дорівнювати цілій сторінці (рядку ядра). Оскільки пропускна спроможність обох шин (команд і даних) однакова, можна переключатися на будь-яку сторінку пам'яті без втрати продуктивності.

У порівнянні з SDRAM набір команд у SLDRAM значно збільшений, що дуже полегшує роботу контролера. Команда являє собою чотири 10-бітних пакети і містить всю інформацію для проведення наступної операції. Таким чином, зростає ефективність управління пам'яттю – всього за 4 такти передається вся інформація, що описує цілий масив даних.

Максимально досяжна нинішнім поколінням SLDRAM швидкість передачі перевищує 1 Гб/с на кожен розряд при частоті 400 МГц. Треба відмітити, що при такій частоті дуже важливо, щоб усі сигнали точно синхронізувалися з тактовими імпульсами системної шини, і щоб усі мікросхеми пам'яті в межах одного модуля мали близькі часові затримки. Для цього контролер програмує всі чипи модуля пам'яті так, щоб вони видавали дані на шину одночасно, незалежно від розкиду їхніх параметрів і віддаленості мікросхем від контролера. У результаті сама вилучена мікросхема видає дані без затримки, а найближча —

через проміжок часу, потрібний для того, щоб сигнал поширився від самої дальньої до найближчої. Ці значення визначаються на момент подачі живлення на ІС і постійно коректуються під час роботи.

11.4.5.8. RDRAM. Основні типи технологій RDRAM

На даний момент існує тільки один спосіб підвищення пропускної спроможності (BW – BandWidth) будь-якої підсистеми – це збільшення або частоти комутації шини, або її "ширини" (розрядності).

Спільне збільшення цих параметрів досить проблематичне і має швидке "насичення", оскільки вплив електромагнітної інтерференції і емісій у цьому випадку зростає нелінійно. Ця обставина змушує розроблювачів йти на компроміс. У противагу технології SDRAM, де використовується 64 бітна магістраль і частота до 133 МГц, сучасна технологія RDRAM (Rambus) надає 16 бітну шину даних і результуючу частоту до 800 МГц відповідно. Вузька шина і колосальна частота значно підвищують ефективність використання пам'яті і завантаження, максимально звільняючи протокол від часових затримок.

Мікросхеми типу RDRAM названі за назвою фірми розроблювача – **Rambus** (RDRAM – Rambus DRAM). Вони представляють собою байт-послідовну пам'ять з дуже високим темпом передачі байтів. Основними нововведеннями архітектурного плану є синхронізація обома фронтами тактових імпульсів і спеціальний новий інтерфейс Rambus Channel. Синхронізація принципово подібна застосованій в SDRAM.

Інтерфейс Rambus Channel має всього 13 сигнальних ліній, що значно менше, ніж у традиційних мікросхем пам'яті. В інтерфейсі немає спеціалізованих адресних ліній. Замість звичайної адресації по інтерфейсі посилаються пакети, що включають у себе команди й адреси. Спочатку надходить пакет запитів, на який пам'ять відповідає пакетом підтвердження, після чого йде пакет даних. Через такий процес перший доступ до даних значно запізнюється. У першій розробці запізнювання складало 128 нс. Середня частота передачі байтів залежить від довжини пакета даних. При обміні пакетами по 256 байтів середня частота буде 400 МГц (до 2 не додається 0,5 нс на байт), при пакетах по 64 байти – 250 МГц і т.д.

RDRAM ідеально підходить для графічних і мультимедійних прикладень з типовим для них процесом – швидкою видачею довгої послідовності слів для формування зображення на екрані або подібних з цим задач.

Уперше пам’ять RDRAM з’явилася в 1995 р., працювала на частоті 150 МГц і забезпечувала пропускну спроможність 600 Мбайт/с. Вона використовувалась в станціях SGI Indigo2 IMPACTtm, у приставках Nintendo64, а також як відеопам’ять. Наступне покоління RDRAM з’явилося в 1997 р. за назвою Concurrent RDRAM. Нові модулі були цілком сумісні з першими. Подальшим розвитком інтерфейсу став фірмовий (Rambus) стандарт DRDRAM (Direct Rambus DRAM). Технологія Direct Rambus являє собою високошвидкісну замкнуту систему, що має свою адаптовану логіку управління і точно розраховані параметри. Direct Rambus дозволяє досягти дуже великих швидкостей передачі даних: до 1,6 Гбайт/с на один канал і до 6,4 Гбайт/с при чотирьох каналах.

У табл. 11.3 наведені основні типи технології RDRAM.

Таблиця 11.3

Основні типи технології RDRAM

Параметр	Base RDRAM	Concurrent RDRAM	Direct RDRAM
Частота синхронізації	250 – 300 МГц	300 – 350 МГц	400 МГц
Результуюча частота	500 – 600 МГц	600 – 700 МГц	800 МГц
Пікова пропускна спроможність	500 – 600 Мбайт/с	600 – 700 Мбайт/с	1,6 Гбайт/с
Шина даних (базова/ECC)	8/9 біт	8/9 біт	16/18 біт
Завантаження 32 бітів протоколу	60%	80%	97 – 100 %
Інтерфейс загального живлення	3.3 В	3,3 В	2,5 В
Розмах активних рівнів сигналів	1,0 В	1,0 В	0,8 В
Діапазон напруг “точка-точка”	1,5 – 2,5 В	1,5 – 2,5 В	1,0 – 1,8 В
Опорна напруга	2,0 В	2,0 В	1,4 В
Число високошвидкісних сигналів	13	13	30
Число виводів для кожного з каналів	32	32	72
Тип корпуса мікросхеми RDRAM	SHP/SVP	SHP/SVP	CSP (EBD/CBD)

Уся підсистема складається з таких компонентів: основний контролер (RMC – Rambus Memory Controller), канал (RC – Rambus Channel), роз'єм для модулів (RRC – Rambus RIMM Connector), модуль пам'яті (RIMM – Rambus In-line Memory Module), генератор диференціальних імпульсів (DRCG – Direct Rambus Clock Generator) і самі мікросхеми пам'яті (RDRAM – Rambus DRAM). У порівнянні з DDR SDRAM при тій же продуктивності DRDRAM має більш компактний інтерфейс і гнучку масштабованість. Розрядність ОЗП DRDRAM (16 байтів) не залежить від числа встановлених мікросхем, а число банків, доступних контролеру, і ємність пам'яті додається по всіх мікросхемах каналу. При цьому в каналі можуть бути присутніми мікросхеми різної ємності в будь-яких поєднаннях.

11.4.5.9. CDRAM

У структурах CDRAM (Cached DRAM, кешована DRAM) на одному кристалі з DRAM розміщена статична кеш-пам'ять (кеш першого рівня). При цьому кеш забезпечує швидкий обмін із процесором, якщо інформація знаходиться в кеші, а також швидке відновлення свого вмісту. Остання можливість пов'язана з тим, що розміщення кеша на одному кристалі з DRAM робить зв'язки між ними внутрішніми (реалізованими усередині кристала), а в цьому випадку розрядність шин може бути великою і обмін може здійснюватися великими блоками даних. Наприклад, у CDRAM фірми Ramtron застосована 2048-розрядна шина для відновлення вмісту кеша.

Як синонім позначення CDRAM іноді використовується позначення EDRAM (Enhanced DRAM). Кешування, як і завжди, ефективне при виконанні програм, для яких промахи відносно кеша бувають досить рідко.

11.4.5.10. Virtual Channel SDRAM

Virtual Channel SDRAM підноситься як революційне нововведення для створення нових типів ОЗП, розрахованих на багатозадачні системи, у яких чергуються звертання різних процесів до різних ділянок пам'яті та побудований на основі технології VCM.

VCM (Virtual Channel Memory) – розроблена NEC і Siemens технологія, що дозволяє оптимізувати доступ до оперативної пам'яті декількох "процесів" (наскільки можна судити, як окремі процеси можуть бути розглянуті, наприклад, запис даних центральним

процесором, перенесення вмісту оперативної пам'яті на твердий диск, звертання графічного процесора і т.п.) таким чином, що переключення між процесами не призводить до падіння продуктивності. На відміну від традиційної схеми, коли всі процеси поділяють ту саму шину вводу-виводу, у технології VCM кожен з них використовує "віртуальну" шину. Організована на рівні чипа взаємодія "віртуальних" і реальної шини дозволяє досягти приросту продуктивності системи до 25%. Схема VCM може бути реалізована в межах вже існуючої технології.

VCM була анонсована компанією NEC Electronics на виставці "Comdex/Fall'97". Фірма визначила новий протокол і схемні рішення, що дозволяють підсистемам, які звертаються до пам'яті, управляти віртуальними каналами (VC) – незалежними інтерфейсними блоками DRAM (рис. 11.36).

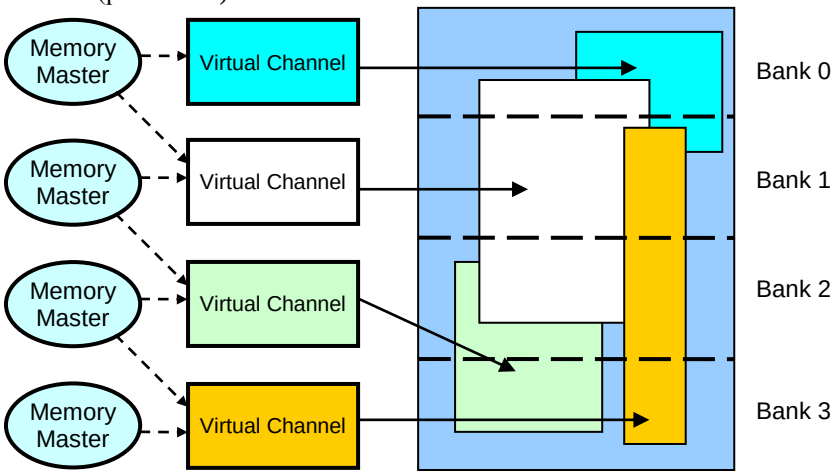


Рис. 11.36. Схема управління віртуальними каналами в Virtual Channel SDRAM

Будь-який прилад, наприклад, L2-контролер або графічний процесор, повинен мати свій віртуальний канал. Кожен канал містить статичний буфер сторінок. Прилад може зчитувати або записувати в буфери, копіювати їхній або завантажувати з накопичувача DRAM. Операційна система, що розпізнає архітектуру VCM, могла б призначити і власний віртуальний канал.

При розробці VCM основними цілями були зниження тривалості затримки, а також зниження енергоспоживання модулів пам'яті. Зрозуміло, що при декількох пристроях, одночасно виконуючих запити в різні області пам'яті (причому доступ в один момент часу може мати тільки один з них), про високу ефективність роботи говорити не доводиться.

Кожному пристрою призначається свій високошвидкісний віртуальний канал, що враховує специфічні характеристики його запитів. У тому числі, функцією віртуальних каналів є і кешування – Memory Master посилає каналу наказ на запис або читання, а той уже займається його виконанням із усіма супутніми деталями. У результаті чого зовнішні та внутрішні операції абсолютно незалежні одна від одної і можуть виконуватись паралельно.

Віртуальний канал може відфільтрувати більшість пропусків сторінок. Коли процес звертається до каналу, він усе ще знаходить SRAM-буфер сторінок незмінним після останнього звертання до нього. Встановлені безпосередньо в периферію DRAM буферизовані контролери віртуального каналу дозволять зменшити паузи в роботі системи при звертанні її підсистем до різних сторінок пам'яті.

У результаті, ефективність доступу до пам'яті значно підвищується, особливо якщо врахувати, що ніщо не заважає, наприклад, тій же відеокарті відкрити, допустимо, три таких канали – один для завантаження вершин трикутників, другий для завантаження текстур, третій для системного обміну з пам'яттю.

За даними NEC збільшення ефективності можна досягти до 90%, а взагалі за тестами VCM133 SDRAM перевищує PC133 на 10 – 30%. Це і затримки, що зменшилися, і більш висока пропускна спроможність, і зменшення енергоспоживання (приблизно на ті ж 30%) за рахунок того, що на той момент, коли відбувається передача результатів наказу системному пристрою, вся фонова активність по інший бік віртуального каналу може бути заморожена.

Чипи VCM цілком аналогічні звичайним чипам SDRAM, сумісні вони з ними і за використовуваним інтерфейсом, BIOS може легко розпізнати модулі VCM SDRAM шляхом використання SPD. Модулі, природно, абсолютно взаємозамінні із звичайними SDRAM DIMM, причому всі останні чипсети від Si, ALI, VIA VCM цілком підтримують. Збільшення площі чипа в порівнянні з тим же SDRAM складає всього 1 – 3%, при цьому використовується те ж виробниче і тестове устаткування, що і для звичайного SDRAM, а за рахунок

незначно збільшеного виходу, собівартість, скажемо, VCM133 для виробників повинна бути приблизно рівною собівартості PC133 SDRAM. Причому VCM цілком незалежна від типу пам'яті, і з легкістю може бути надалі убудована, наприклад, у DDR SDRAM.

Динамічна пам'ять фірми NEC вимагає, щоб контролер пам'яті явно замовив усі переміщення даних між SRAM-буферами і DRAM-масивом. Це дозволить досить "розумному" контролеру цілком сховати більшість тактів чекання, виконуючи переміщення даних, на той час як інші канали використовують буфери вводу-виводу. У результаті DRAM поліпшує продуктивність системи без збільшення тактової частоти роботи шини.

Відповідно до технології VCM у VCSDRAM будь-який системний пристрій (Memory Master) може зробити запит, що володіє унікальними характеристиками – адресою, розміром блоку даних до пам'яті, за допомогою віртуальних каналів. По ідеї, системний контролер пам'яті асоціює канали з процесами, що прискорює роботу системи, начебто кожному процесу виділявся окремий ресурс доступ до пам'яті. Кожен канал може виконати обмін даними з будь-яким рядком будь-якого банку ядра.

За цією технологією при записі дані не відразу заносяться в ядро, а містяться в буфер – віртуальний канал – і зберігаються там доти, поки ядро не буде готове їх прийняти (воно, наприклад, може бути зайнято регенерацією або обміном з іншим пристроєм).

Запис даних у VCSDRAM виконується таким способом: спочатку дані записуються у віртуальний канал, а потім у міру звільнення контролера DRAM відбувається запис безпосередньо в комірки пам'яті SDRAM.

Читання даних здійснюється шляхом запиту даних у віртуального каналу, що відповідно до запиту прямо вказує вміст блоку осередків SDRAM, що містить необхідні дані, і додатково виконає читання, що випереджає.

Щоб при одночасному звертанні до пам'яті декількох процесів не знизилася продуктивність, число каналів доведене до 16 по 1024 біти кожен (у модулях по 256 Мбітів кожен канал може передавати до 2048 бітів). Працює VC SDRAM при частоті аж до 143 МГц. Тип корпуса – стандартний, сумісний за контактами і набором команд ("зверху вниз") із SDRAM.

11.5. КЕШ-ПАМ'ЯТЬ

11.5.1. Загальне представлення про кеш-пам'ять

Кеш-пам'ять запам'ятовує копії інформації, переданої між пристроями (насамперед між процесором і основною пам'яттю). Вона має невелику ємність у порівнянні з основною пам'яттю і більш високу швидкодію (реалізується на тригерних елементах пам'яті).

При читанні даних спочатку виконується звертання до кеш-пам'яті (рис. 11.37).

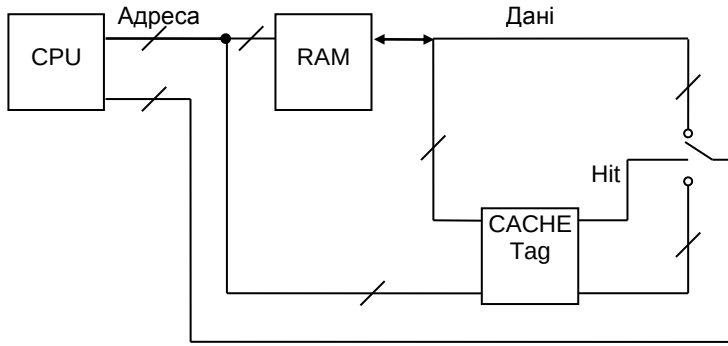


Рис. 11.37. Структура кешованої пам'яті

Якщо в кеші є копія даних адресованого осередку основної пам'яті, то кеш виробляє сигнал Hit (улучення) і видає дані на загальну шину даних. У противному разі сигнал Hit не виробляється і виконується читання з основної пам'яті й одночасне переміщення врахованих даних у кеш.

Ефективність кешування обумовлюється тим, що більшість прикладних програм мають циклічний характер і багаторазово використовують одні і ті ж дані. Тому після першого використання даних з відносно повільної основної пам'яті повторні звертання вимагають менше часу. До того ж при використанні процесором кеш-пам'яті основна пам'ять звільняється і може здійснюватися регенерація даних у динамічному ЗП або використання пам'яті іншими пристроями.

Ємність кеш-пам’яті набагато менше ємності основної пам’яті і будь-яка одиниця інформації, що поміщається в кеш, повинна супроводжуватися додатковими даними (тегом), що визначають, копією вмісту якого осередку основної пам’яті є ця одиниця інформації.

11.5.2. Види кеш-пам’яті

У цілком асоціативній кеш-пам’яті (FACM, Fully Associated Cache Memory), структура якої показана на рис. 11.38, кожен осередок зберігає дані, а в полі “тег” – повна фізична адреса інформації, копія якої записана.

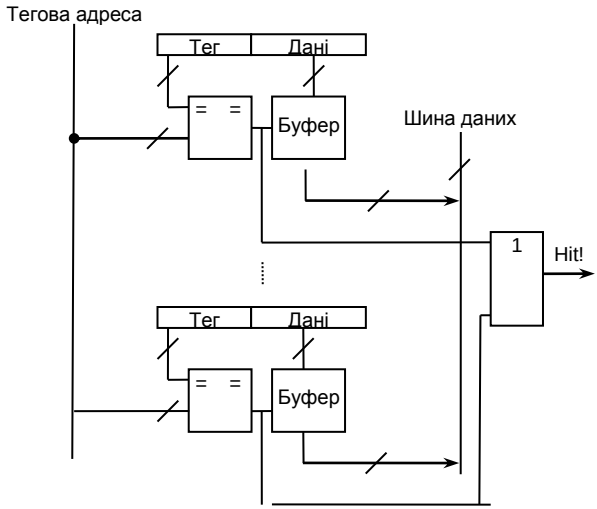


Рис. 11.38. Структура цілком асоціативної кеш-пам’яті

При будь-яких обмінах фізична адреса запитуваної інформації порівнюється з полями “тег” всіх осередків і при збігу їх у будь-якому осередку встановлюється сигнал Hit.

При читанні і значенні сигналу Hit = 1 дані видаються на шину даних, якщо ж збігів немає (Hit = 0), то при читанні з основної пам’яті дані разом з адресою розміщуються у вільний або найдовше невикористовуваний осередок кеш-пам’яті.

При записі дані разом з адресою спочатку, як правило, розміщуються в кеш-пам'яті (у виявлений осередок при Hit = 1 і вільний при Hit = 0). Копіювання даних в основну пам'ять виконується під управлінням спеціального контролера, коли немає звертань до пам'яті.

Пам'ять типу FACM є досить складним пристроєм і використовується тільки при малих ємностях, головним чином у спеціальних прикладеннях. На той же час цей вид кеш-пам'яті забезпечує найбільшу функціональну гнучкість і безконфліктність адрес, тому що будь-яку одиницю інформації можна завантажити в будь-який осередок кеш-пам'яті.

Складність FACM змушує шукати інші структури кеш-пам'яті, більш економічні за витратами апаратних засобів на їхню реалізацію. До числа таких структур відносяться *кеш-пам'ять із прямим розміщенням* і *кеш-пам'ять з набірно-асоціативною архітектурою* (з асоціацією за декількома напрямками). Для конкретного розгляду цих структур укажемо, що головними параметрами кеш-пам'яті є розмір рядка (Cache Line) та їхнє число (рис. 11.39). Рядок представляє собою деякий набір слів.

Рис. 11.39. Представлення кеш-пам'яті у вигляді сукупності рядків

Її ємність будемо вважати відповідною сторінці основної пам'яті.

У структурі FACM, названою також структурою з довільним завантаженням, будь-яку сторінку можна завантажити в будь-який рядок кеш-буфера (рис. 11.40, *а*). У якості тега використовується повна фізична адреса, якщо мова йде про адресацію окремих слів.

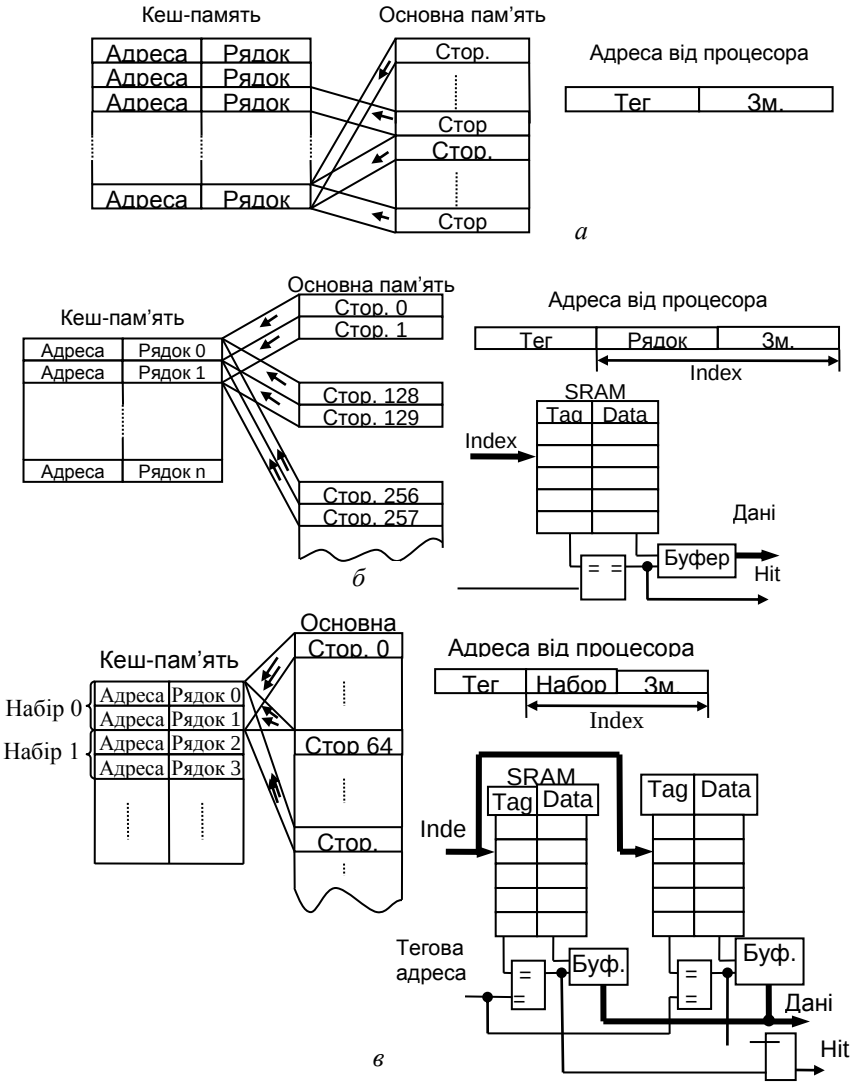


Рис. 11.40. Пояснення до організації кеш-пам'яті: *а* – з довільним завантаженням *б* – із прямим розміщенням, *в* – із набірно-асоціативною архітектурою

Іншими словами, у цьому випадку старші розряди адреси розглядаються як тег, тоді як молодші використовуються для адресації в межах рядка.

У кеш-пам'яті з прямим розміщенням (із прямим відображенням) кілька сторінок основної пам'яті строго відповідають одному рядку кеша (рис. 11.40, *б*). Тому що займати рядок на той саме час може тільки одна сторінка, потрібна спеціальна її ознака – тег. Адреса від процесора поділяється на три частини. Молодші розряди (зміщення) визначають положення слова в рядку. Середні дозволяють вибрати одну з рядків кеш-пам'яті. Старші розряди утворюють тег. За адресою рядка здійснюється зчитування. Поле адрес рядка порівнюється з теговою адресою і якщо є збіг, виробляється сигнал Hit видачі інформації і потім мультиплексуванням з рядка даних вибирається слово. При завантаженні з зовнішньої пам'яті заміняється весь рядок. Тут слід зазначити, що блокові передачі в сучасних системах здійснюються досить швидко.

Тег для кеш-пам'яті з прямим розміщенням значно скорочується за розрядністю. Звичайно номер рядка є адреса сторінки за модулем, який дорівнює цілому степеню двійки. На рис. 11.40, *б* це 128. Достоїнство кеша з прямим розміщенням – економічність за апаратурними витратами. Недолік – обмеження на розміщення сторінок у кеші, що може не дозволити сформувати в ньому оптимальний набір сторінок, тому що передача в кеш сторінки викликає видалення з нього іншої, яка може бути потрібна для формування оптимального набору сторінок.

Проміжним за складністю й ефективністю варіантом між структурами FACM і з прямим розміщенням є *кеш-пам'ять з асоціацією за декількома напрямками (набірно-асоціативна).* У цьому варіанті кілька рядків кеша поєднуються в набори, а середні розряди адреси пам'яті визначають уже не один рядок, а набір (рис. 11.40, *в*). Кеш-пам'ять поділяється на набори з невеликим числом рядків, кратних двійці, тобто 2, 4, 8 ... і т.д. (на рис. 11.40 – це 2). Сторінку основної пам'яті можна помістити тільки в той набір, номер якого дорівнює адресі сторінки за модулем (у даному випадку модуль дорівнює 64). Місце сторінки в наборі може бути довільним. Порівняння тегів зі старшими розрядами адреси здійснюється тільки для рядків, що входять у набір.

За числом рядків у наборі кеш-пам'яті розрізняють різноманітні структури: 2-входові, 4-входові і т.д.

Для даного прикладу використовуються два окремих блоки пам'яті для парних і непарних рядків. Одночасно вибираються парні і непарні рядки (слова в них). Зчитування йде від того блоку, де є збіг тега і тегової адреси. При цьому з рядка через зсув вибирається адресоване слово. При відсутності збігів відбувається звертання до основній пам'яті та заміщення рядка в одному з блоків кеша.

У порівнянні з кешем із прямим розміщенням кеш набірно-асоціативного типу має трохи подовжений тег (у даному прикладі всього на один розряд). Можливість вільного розміщення сторінок у наборі дозволяє сформувати в кеші кращий склад сторінок, тому що є можливість вибрати ту або іншу сторінку. У сучасних мікропроцесорних системах кеш першого рівня, що позначається L1 (від англійського слова Level (внутрішньопроцесорний)), звичайно має набірно-асоціативну структуру, а кеш другого рівня L2 (зовнішній) – структуру з прямим розміщенням.

Ряд фірм випускають мікросхеми асоціативної пам'яті. Наприклад, одна з мікросхем фірми Cypix має 4 к рядків, 15-розрядну тегову адресу і 16-розрядний вихід. Для побудови кеш-пам'яті використовують найчастіше звичайні SRAM у поєднанні з кеш-контролерами. У високопродуктивному мікропроцесорі Power 3 фірми IBM використаний кеш набірно-асоціативного типу ємністю 32 кбайти для команд і 64 кбайти для даних на 128 напрямків. Для зв'язків з кешем другого рівня L2 у системі Power 3 застосовується 256-розрядна шина. Ємність кеша L2 від 1 до 16 Мбайтів.

11.6. Програмувальні логічні пристрої. Програмувальні логічні матриці

У загальному випадку програмувальні логічні матриці (ПЛМ) і програмувальні логічні пристрої (ПЛП) являють собою логічну схему для перетворення комбінацій $X = x_1x_2\ldots x_n$ вхідного двійкового коду на відповідні комбінації $Y = y_1y_2\ldots y_m$ вихідного двійкового коду. Правило перетворення кодів задається звичайно таблицею істинності. Розряди вихідного коду $y_1y_2\ldots y_m$ можуть розглядатись як система булевих функцій від двійкових змінних $x_1x_2\ldots x_n$.

Програмувальні логічні матриці – найбільш традиційний тип ПЛІС, що має програмувальні матриці "І" та "АБО". У закордонній

літературі відповідними цьому класу абревіатурами є FPLA (*Field Programmable Logic Array*) і FPLS (*Field Programmable Logic Sequensers*). Прикладами таких ПЛІС можуть служити вітчизняні схеми К556РТ1, РТ2, РТ21.

Побудова ПЛМ заснована на тому, що будь-яка комбінаційна функція може бути представлена у вигляді логічної суми (операція АБО) логічних добутків (операцій І). Недолік такої архітектури – слабке використання ресурсів програмувальної матриці "АБО", тому подальший розвиток одержали мікросхеми, побудовані за архітектурою програмувальної матричної логіки (PAL – *Programmable Array Logic*) – це ПЛІС, що мають програмувальну матрицю "І" і фіксовану матрицю "АБО". До цього класу відноситься більшість сучасних ПЛІСів невеликого степеня інтеграції. Як приклади можна навести вітчизняні ІС КМ1556ХП4, ХП6, ХП8, ХЛ8, ранні розробки (1980-і роки) ПЛІС фірм INTEL, ALTERA, AMD, LATTICE та ін. Різновидом цього класу є ПЛІС, що мають тільки одну (програмувальну) матрицю "І", наприклад, схема 85С508 фірми INTEL. Наступний традиційний тип ПЛІС – програмувальна макрологіка. Вони містять єдину програмувальну матрицю "І-НІ" або "АБО-НІ", але за рахунок численних інверсних обернених зв'язків здатні формувати складні логічні функції. До цього класу відносяться, наприклад, ПЛІС PLHS501 і PLHS502 фірми SIGNETICS, що мають матрицю "І-НІ", а також схема XL78C800 фірми EXEL, заснована на матриці "АБО-НІ".

Програмованість матриць варто сприймати в тому розумінні, що ми можемо вибирати (програмувати) деякі з'єднання в них. У ПЛП матриця диз'юнкції жорстко задана; у ПЛМ можливо оперувати обома матрицями.

Програмуючи ПЛП, можна реалізувати потрібні системи булевих функцій.

Досить часто допускається можливість використання виходів як додаткових множників у вхідних змінних, тим самим розширюючи безліч реалізованих логічних функцій. Можлива комбінація, що включає використання таких елементів, як тригери.

При підключенні до ПЛМ дешифратора отримана схема може виконувати функції ПЗП. Таке поєднання вигідне застосовувати при побудові пристроїв пам'яті невеликої ємності, у яких ємність ПЗП використовуються не цілком, і тому витрати на ПЗП не виправдовуються. ПЛМ можна також застосовувати як фіксовану схему управління. Саме для цієї мети і призначалися перші ПЛМ.

Використовуючи фіксовану схему управління на ПЛМ, можна значно збільшити швидкодію всієї системи. Це пояснюється тим, що ПЛМ є комбінаційною схемою з високою швидкодією. Ще ПЛМ знаходять застосування як перетворювачі кодів, генераторів логічних функцій.

Розглянемо використання ПЛМ як генератора логічних функцій (рис. 11.41).

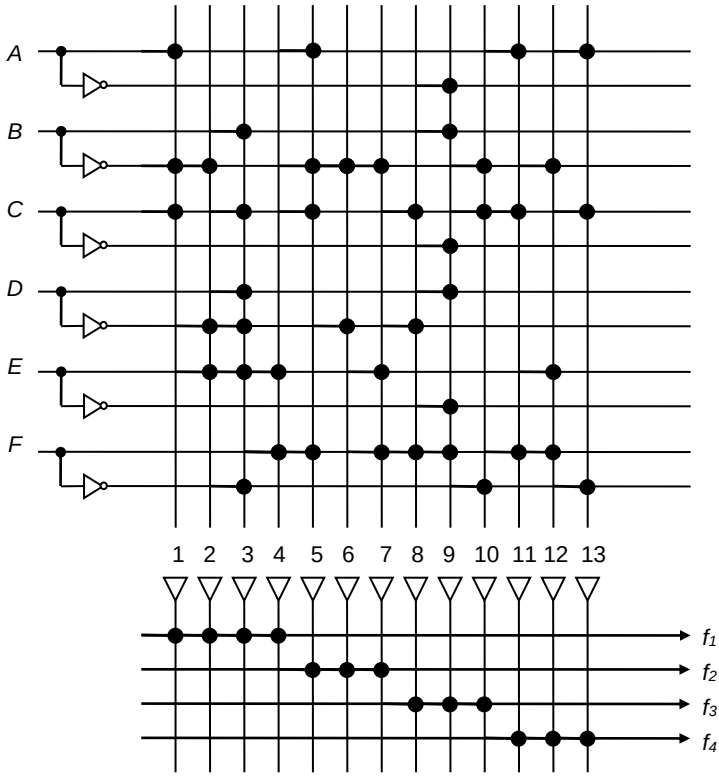


Рис. 11.41. Приклад використання ПЛМ як генератора логічних функцій

Зокрема розглянемо конкретну програмувальну логічну матрицю з 6-ма входами і 4-ма виходами. Нехай потрібно реалізувати таку систему перемикальних функцій:

$$f_1 = A\overline{B}C + \overline{B}\overline{D} + BC\overline{D}E\overline{F} + EF;$$
$$f_2 = A\overline{B}CF + \overline{B}\overline{D} + \overline{B}EF;$$
$$f_3 = C\overline{D}F + \overline{A}B\overline{C}D\overline{E}F + \overline{B}C\overline{F};$$
$$f_4 = ACF + \overline{B}EF + \overline{B}D + AC\overline{F}$$

Таблиця істинності наведена в табл. 11.4.

Таблиця 11.4

Таблиця істинності до рис. 11.41

Входи							Виходи			
№	A	B	C	D	E	F	F1	F2	F3	F4
1	1	0	1	x	x	x	1	0	0	0
2	x	0	x	0	x	x	1	0	0	0
3	x	1	1	0	1	0	1	0	0	0
4	x	x	x	x	1	1	1	0	0	0
5	1	0	1	x	x	1	0	1	0	1
6	x	0	x	0	x	x	0	1	0	1
7	x	0	x	x	1	1	0	1	0	0
8	x	x	1	0	x	1	0	0	1	0
9	0	1	0	1	0	1	0	0	1	0
10	x	0	1	x	x	0	0	0	1	0
11	x	1	x	1	x	1	0	0	0	1
12	x	0	x	x	1	1	0	0	0	1
13	1	x	1	x	x	0	0	0	0	1

Зауваження 1

У випадку з програмувальними логічними матрицями немає необхідності мінімізувати вихідну систему перемикальних функцій з метою зменшення кількості термів, тому що воно не грає для ПЛМ істотної ролі. Саме зменшення необхідно здійснювати для того, щоб задовольнити обмеження, що накладаються на ПЛМ у вигляді кількості її входів і виходів.

Зауваження 2

Реалізуючи управляючу логіку на стандартних БІСА, ми маємо випадок з великими витратами площі кристала на взаємні з'єднання

(неупорядкована логіка). Реалізація ПЛМ у вигляді ВІС – це упорядкована логіка і матрична організація, яка дозволяє досить ефективно використовувати площу кристала.

Промисловістю освоєні ПЛМ із різною схемотехнологією: з діодами в матриці І і біполярними транзисторами в матриці АБО, з *n*-МОП-транзисторами в обох матрицях та ін. При цьому властивості ПЛМ визначаються властивостями застосованих у ній елементів. Основними параметрами ПЛМ є число входів m, число перехідних ланцюгів (термів) l і число виходів n.

Складність ПЛМ прийнято оцінювати сумарною інформаційною ємністю (число вузлів – перетинань ліній, у яких розміщаються елементи зв'язку). Інформаційна ємність ПЛМ A = (2m + n) l, де коефіцієнт 2 перед m відбиває наявність прямих та інверсних значень вхідних змінних у матриці І.

11.7. ПЛІС. FPGA

На даний час перспективною тенденцією в області розробки мікросхем наступного покоління є так звані системи на кристалі (SOC – system on chip). У основі ідеї SOC лежить інтеграція всієї електронної системи в одному кристалі. Компоненти цих систем розробляються окремо і зберігаються у вигляді файлів параметризованих модулів. Остаточна структура SOC-мікросхеми виконується на базі цих "віртуальних компонентів", названих також "блоками інтелектуальної власності" за допомогою програм автоматизації проектування електронних пристроїв. Завдяки стандартизації в одне ціле можна поєднувати "віртуальні компоненти" від різних розроблювачів.

Існує декілька підходів до реалізації Soc'ів. Перший шлях – створення замовлених мікросхем (ASIC – Application Specific Integrated Circuit). Це деяка тверда логіка, реалізована в кристалі. Такий підхід економічно доцільний при випуску великої кількості однотипних мікросхем. Інший шлях – FPGA. Наявність великого числа макроелементів у кристалі FPGA дозволяє, зокрема, реалізувати на ньому ядро мікропроцесора, пам'ять, логічну частину.

11.7.1. Архітектура FPGA

FPGA (field-programable gate array) – великі швидкодіючі програмувальні логічні матриці. Це апаратні схеми, що можуть бути

модифіковані практично в будь-який момент у процесі їхнього використання. Вони складаються з конфігурованих логічних блоків, подібних до перемикачів з безліччю входів і одним виходом (gate). У цифрових схемах такі перемикачі реалізують базові двійкові операції AND, NAND, OR, NOR і XOR. Принципова відмінність FPGA полягає в тому, що і функції блоків, і конфігурація з'єднань між ними можуть змінюватись за допомогою спеціальних сигналів, що посилаються схемі.

Як приклад спробуємо знову реалізувати систему перемикальних функцій для двох змінних, але вже на FPGA (рис. 11.42).

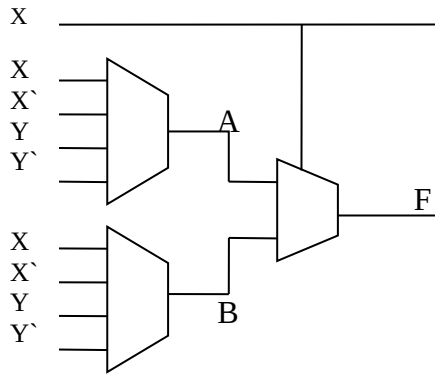


Рис. 11.42. Приклад реалізації функцій для двох змінних

Таблиця функціонування схеми реалізації для двох змінних наведена в табл. 11.5.

Таким чином, поклавши в основу мультиплексор, ми маємо можливість запрограмувати його на реалізацію кожної з 16-ти функцій для 2-х змінних.

Проектуючи матрицю з FPGA, можна не тільки задавати функції осередків, але і з'єднання між осередками, тобто вирішувати задачу маршрутизації. Зокрема, якщо в нас є осередок з функцією f, то n (north), w (west), e (east), s (south) задають напрямок передачі.

Можливість комутації вихідних сигналів одних осередків із входами інших подібних осередків реалізується в різних FPGA по-

різному, але типовим є випадок, коли вихід кожного осередку надходить на входи 4-х сусідніх за схемою.

Таблиця 11.5

Таблиця істинності до рис. 11.42

a	b	F
X	Y	X + Y
X	Y`	X + Y`
Y	X`	X` + Y
Y`	X`	X` + Y`
Y	X	X * Y
Y`	X	X * Y`
X`	Y	X` * Y
X`	Y`	X`* Y`
Y`	Y	X ⊕ Y
Y	Y`	X ⊕ Y`
X	X`	1
X`	X	0
X	X	X
X`	X`	X`
Y	Y	Y
Y`	Y`	Y`

FPGA – програмувальні вентильні матриці (ПВМ) – один з типів архітектури ПЛІС. ПЛМ складається з логічних блоків (ЛБ) (рис. 11.43) і комутуючих шляхів – програмувальних матриць з'єднань. Логічні блоки таких ПЛІС складаються з одного або декількох дещо простих логічних елементів, в основі яких лежить таблиця перекодування (ТП, *Look-up table* – LUT), програмувальний мультиплексор, D-тригер, а також ланцюги управління (рис. 11.44).

Таких простих елементів може бути досить багато, наприклад, у сучасних ПЛІС ємністю до 1 млн. вентилів число логічних елементів досягає декількох десятків тисяч.

За рахунок такого великого числа логічних елементів вони містять значне число тригерів. Безліч конфігурованих логічних блоків таких простих елементів може бути досить великою, наприклад, у сучасних ПЛІС ємністю до 1 млн. вентилів число логічних елементів

досягає декількох десятків тисяч. *Configurable Logic Blocks* – CLBs) поєднується за допомогою матриці з'єднань.

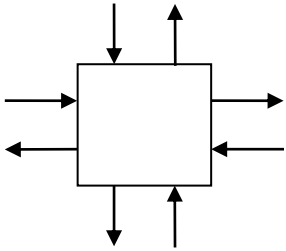


Рис. 11.43. Схема елементарного логічного блока

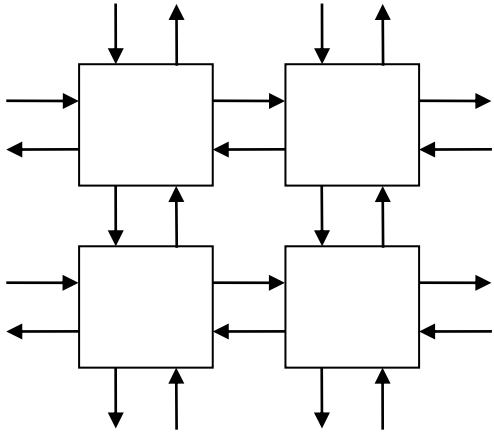


Рис. 11.44. Схема одного з варіантів з'єднання елементарних логічних блоків

Характерними для FPGA-архітектур є елементи вводу-виводу (input/output blocks – IOBs), що дозволяють реалізувати двонаправлений ввід-вивід, третій стан і т.п. Деякі сімейства ПЛІС мають убудовані реконфігуровані модулі пам'яті (РМП, *embedded array block* – EAB), що робить ПЛІС даної архітектури досить зручним засобом реалізації алгоритмів цифрової обробки сигналів, основними операціями в яких є перемножування, множення на константу, підсумовування і затримка сигналу. Разом з тим, можливості комбінаційної частини таких ПЛІС обмежені, тому разом із ПВМ застосовують ПКМБ (CPLD – Complex Programmable Logic Device) для реалізації управляючих та інтерфейсних схем.

CPLD (ще один варіант ПЛІС) – це програмувальні матричні блоки, що комутуються (ПКМБ). Вони містять кілька матричних логічних блоків (МЛБ), об'єднаних комутаційною матрицею. Кожен МЛБ являє собою структуру типу ПМЛ, тобто програмувальну матрицю "І", фіксовану матрицю "АБО" і макроосередок ПЛІС типу ПКМБ, як правило, мають високий степінь інтеграції (до 10000 еквівалентних вентилів, до 256 макроосередків).

До цього (CPLD) класу відносяться ПЛІС сімейства MAX5000 і MAX7000 фірми ALTERA, схеми XC7000 і XC9500 фірми XILINX, а

також велике число мікросхем інших виробників (Atmel, Vantis, Lucent та ін.).

До FPGA (ПВМ) класу відносяться ПЛІС XC2000, XC3000, XC4000, Spartan, Virtex фірми XILINX; ACT1, ACT2 фірми ACTEL, а також сімейства FLEX8000 фірми ALTERA, деякі ПЛІС Atmel і Vantis.

Особливістю сучасних ПЛІС є можливість тестування вузлів за допомогою порту JTAG (B-scan), а також наявність внутрішнього генератора (Osc) і схем управління послідовною конфігурацією.

Створення файлу конфігурації сучасних ПЛІС неможливо без автоматизованих систем проектування. Такі системи випускають як усі ведучі виробники ПЛІС (ALTERA, XILINX_), так і інші компанії.

При роботі в подібних системах конфігурація схеми, що повинна бути отримана "усередині" ПЛІС або алгоритм її роботи задається або текстовою мовою описів (ADHL, VDHL або Verilog), що нагадує мову програмування високого рівня (наприклад Сі), або на графічному рівні – у вигляді електричної схеми (у форматах OrCAD або PCAD), чи за допомогою блок-схем алгоритмів або графіків вхідних і вихідних сигналів. Надалі всі етапи роботи, включаючи програмування або завантаження ПЛІС, виконує автоматизована система.

11.7.2. Типи FPGA. Можливості FPGA

Існують два основних типи FPGA – так звані coarse-grained і fine-grained, що можна перевести як низький і високий рівень деталізації. Конфігурована матриця з низьким рівнем деталізації складається з невеликого числа потужних логічних блоків, кожен з яких може виконувати операції додавання і порівняння над двома числами. FPGA з високим рівнем деталізації має безліч простих блоків, що виконують тільки елементарні операції порівняння двох двійкових знаків. Вибір того або іншого типу програмувальної матриці залежить головним чином від прикладень, для яких вона буде використовуватися.

Можливості застосування FPGA досить різноманітні. Найбільш проста з них – переключення між функціями – являє собою апаратний еквівалент переходу від виконання однієї програми до виконання іншої. Для подібних прикладень цілком підходить матриця з великим, порядку декількох секунд, часом реконфігурації.

FPGA з більш швидким перепрограмуванням може виконувати послідовність задач, реконфігуруючись між ними. Така схема працює в режимі поділу часу і змінює конфігурацію настільки швидко, що

виникає враження, начебто виконуються відразу всі функції. Приклад використання подібних схем – система передачі відеозображень на базі однієї мікросхеми, що при обробці відеокадру встигає реконфигурироватись чотири рази. Спочатку FPGA зберігає відеосигнал у пам'яті, потім виконує два різних перетворення зображення і, нарешті, схема "трансформується" на модем для подальшої передачі сигналу. При використанні звичайних ASIC з фіксованою конфігурацією для рішення такої задачі треба було б у чотири рази більше апаратного забезпечення.

Найбільший потенціал мають конфігуровані мікросхеми, здатні змінюватися в режимі on the fly ("на льоту") у процесі виконання задачі. Наприклад, FPGA для розпізнавання образів буде перебудовуватись у відповідь на спробу ідентифікації розглянутого об'єкта: якщо це легковий автомобіль або трейлер, схема, спочатку призначена для відстеження високошвидкісних літаків, змінить свою конфігурацію і зможе визначати наземні види транспорту.

11.7.3. Сучасні тенденції розвитку FPGA

Еволюція FPGA призвела до значного прискорення власне процесу реконфігурації. Якщо в перших конфігурованих матрицях на перепрограмування потрібно було декілька секунд, то новітні FPGA-процесори можуть змінюватись за одну мілісекунду. За прогнозами, за найближчі два роки цей часовий відрізок скоротиться до 100 мкс. Зрештою комп'ютер з конфігурованим процесором одержить можливість практично безупинно адаптувати самого себе до будь-яких змін вхідних даних або середовища обробки.

Ідея конфігурованих мікросхем була висловлена Джеральдом Естрином з Каліфорнійського університету наприкінці 60-х років минулого століття. Однак перші програмувальні матриці з'явилися лише кілька років тому. Сьогодні деякі фірми, наприклад Xilinx і Altera, випускають FPGA з 100 тис. логічних елементів, однак цілком реалізувати потенціал цієї технології поки не вдалося.

Сучасні конфігуровані схеми підходять не для всіх типів обчислень. Вони ефективно реалізують алгоритми, що використовують бітові операції, такі як пошук відповідності шаблону або цілочислова арифметика, але в операціях з числами, наприклад при множенні з високим ступенем точності або обчисленнях із плаваючою точкою, FPGA ще недостатньо потужні. Крім того, у сучасних FPGA мало

пам’яті на чипі для збереження проміжних результатів обчислень. Тому для багатьох прикладень потрібна додаткова зовнішня пам’ять, звертання до якої збільшує витрату енергії та сповільнює процес обчислень.

Зараз з’являються вдосконалені FPGA з пам’яттю, арифметичними модулями й іншими спеціальними блоками на кристалі. Андре Дехон і Томас Найт із Массачусетського технологічного інституту запропонували FPGA, безліч конфігурацій якої зберігається в серії банків пам’яті. За один такт чип може заміняти одну конфігурацію іншою без втрати оброблюваних даних. Схожа ідея реалізована Бредом Хатчингсом з Університету Брайам Янг, який розробив комп’ютер з динамічним набором команд (dynamic instruction set computer, DISC). DISC ефективно поєднує кращі властивості універсального мікропроцесора і матриці FPGA та демонструє потенціал автоматичної реконфігурації за допомогою збережених у пам’яті банків конфігурацій. DISC дозволяє розроблювачу створювати і зберігати безліч конфігурацій апаратної схеми й активувати їх аналогічно виклику підпрограми на мікропроцесорі.

В Університеті Берклі розробляється система, що поєднує можливості універсального мікропроцесора і FPGA на одному кристалі. Спеціальний компілятор автоматично транслює програмний код у комбінацію машинних команд і конфігурацій FPGA.

Підхід, що використовує FPGA, послужив основою розробки прототипів комп’ютера HAL, що за твердженням розроблювачів має швидкодію 12,84 трлн. оп/с. Найвища швидкість обчислень обумовлена винятковою гнучкістю й ієрархічною архітектурою комп’ютера, у якому пам’ять і окремі логічні частини всіх процесорів можна в будь-якій комбінації з’єднувати один із одним. Перепрограмування логічних матриць можна здійснювати незліченне число разів. На даний час аналогічний підхід використовується фірмою IBM у схемах управління до АТМ-пристроїв (банкоматів). Оскільки протоколи АТМ досить часто змінюються, застосування FPGA дозволяє дистанційно перепрограмувати логіку їхньої роботи. Ще в 1995 році одним із перших застосувань мікросхем FPGA для робочої станції фірми SUN, що включали 32 FPGA-мікросхеми, була реалізація систолічного алгоритму для поєднання одновимірних образів при дослідженні молекул ДНК. Така спеціалізована плата показала швидкодію в 325 разів вище, ніж розрахунок цієї ж задачі на супер-ЕОМ Cray-2.

11.8. САМ-пам'ять

Корені концепції асоціативності ідуть від часів Арістотеля. Асоціація розглядалось як встановлення відповідності між об'єктами. Арістотель у праці "Про пам'ять і спогади" визначив три типи асоціації: за подібністю, за контрастом і за близькістю.

На дослідження і розробки в області асоціативних ЗП були витрачені значні зусилля. Мається багато сотень робіт, присвячених цій проблемі, і в даний час число публікацій з даного питання складає декілька десятків за рік.

У липні 1945 року Бушем була запропонована асоціативна архітектура гіпотетичної машини MEMEX. Передбачаючи в майбутньому лавиноподібне зростання знань, Буш запропонував MEMEX як допомогу людині в переробці величезних об'ємів інформації. Мозок, вважав Буш, працює асоціативно, і тому в MEMEX був реалізований вибір по асоціації, на відміну від відомих прийомів пошуку за покажчиками і т.д. MEMEX (механізований асоціативний доступ до даних), за твердженням Буша, служив розширенням людської пам'яті. У MEMEX окремі порції даних поєднувалися одна з одною за допомогою асоціації "аналогічно тому, як ми думаємо". Будь-яка окрема інформація викликає негайний автоматичний вибір іншої інформації.

11.8.1. Асоціативний принцип пошуку

Асоціативний принцип пошуку полягає в тому, що пошук кожного об'єкта інформації здійснюється за деякою сукупності внутрішньо властивих ознак. При цьому задана сукупність ознак порівнюється з відповідною сукупністю ознак, збережених у системі об'єктів інформації. Факт збігу порівнюваної сукупності ознак деякого об'єкта інформації з заданою сукупністю ознак вказує на те, що шуканий об'єкт інформації знайдено.

Існує два основних шляхи реалізації асоціативного принципу пошуку інформації: програмний і схемний. Перший шлях передбачає побудову пристрою програмного асоціативного пошуку, що містить схему порівняння двох слів і звичайний ЗП. При цьому вміст комірки пам'яті ЗП послідовно зчитується на схему порівняння, у якій ознаки слова порівнюються з заданими, і виробляється сигнал збігу або розбіжності. Характерною рисою пристрою програмного асоціативного

пошуку є те, що операція порівняння ознак виконується поза ЗП. Це вимагає у випадку неупорядкованого розміщення інформації в ЗП послідовного перегляду вмісту всіх комірок пам'яті ЗП.

Найбільш ефективно асоціативний принцип пошуку інформації реалізується схемним шляхом, що полягає в побудові асоціативних ЗП (АЗП). Асоціативна пам'ять може бути визначена як система для запису, збереження, пошуку, обробки і зчитування інформації, у якій дані (знання про об'єкт) можуть бути ініціалізовані за заданим фрагментом цих даних (знань), використовуваних в якості пошукових. Відмінною рисою АЗП є здатність реалізації операції порівняння безпосередньо в ЕП ЗП. Це дає можливість виконати операцію порівняння заданих ознак із ознаками всіх збережених в ЕП слів одночасно і незалежно від ємності АЗП. Для того щоб мати можливість знайти факт відповідності ознак збережених у ЕП слів заданим ознакам, кожна ЕП АЗП, реалізуючи операцію порівняння, повинна виробляти сигнал відповідності, одиничне значення якого вказує на наявність заданої відповідності зазначених ознак, а нульове – на його відсутність. Сигнали відповідності дозволяють встановити місце розташування і здійснити вибірку тих комірок пам'яті, у яких зберігаються слова для пошуку.

Найбільш загальним випадком пристрою, що допускає звертання до інформації із вмісту, є так званий цілком асоціативний пристрій, у якому ознака пошуку може бути довільно розташованою у всіх розрядах слова.

Можна виділити також три найбільш характерні типи частково-асоціативних ЗП, у яких пошук проводиться за деякою обмеженою частиною слова:

- пам'ять, що має властивості цілком асоціативного пристрою, але в межах обмеженої за довжиною частини слова;
- пам'ять з декількома асоціативними ознаками, будь-яка комбінація яких може брати участь в опитуванні;
- пам'ять з однією ознакою фіксованої довжини, всі розряди якої одночасно використовуються для опитування (при цьому інформацією для пошуку є інша частина слова).

11.8.2. Архітектура і функціонування АЗП

Блок-схема АЗП наведена на рис. 11.45, в якій прийнято такі позначення:

COM – блок місцевого управління;
RGDI – регістр збереження коду ознаки опитування;
DI – індикатор даних;
M – маска;
RGM – регістр маски;
AM – асоціативний накопичувач;
RGWS – регістр вибору слів;
RGSR – регістр результату пошуку;
RGDO – регістр збереження вихідної інформації;
CRI – блок індикаторів числа збігів;
MRR – розподільник багатокритеріальних збігів;
INS – команда;
NR – вихід, який означає, що немає збігів.

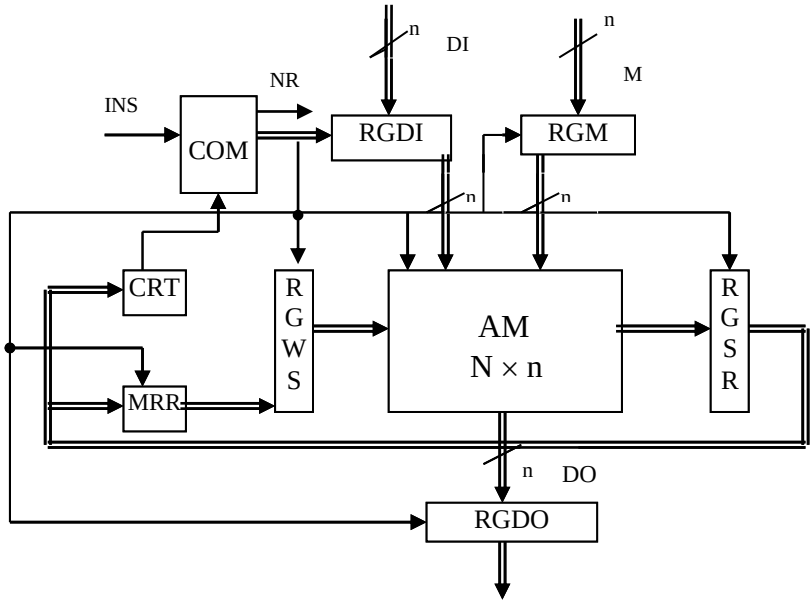


Рис. 11.45. Блок-схема АЗП

У елементах пам’яті асоціативного накопичувача АМ АЗП зберігаються M n-розрядних двійкових слів

$$Q_i = g_{i1} \dots g_{in}, \; i = 1 - M.$$

У відповідні розрядні шини, що поєднують однойменні розряди всіх слів у стовпці матриці АМ, з регістра RGDI задається код ознаки опитування $X = x_1 \ldots x_n$.

Код може мати довільне число розрядів, від одного до n (n – макси-мальне число розрядів). Якщо код ознаки використовується цілком, то він без зміни надходить на схему порівняння. Якщо ж необхідно використовувати тільки частину коду, то непотрібні розряди маскуються за допомогою регістра маски. Для порівняння вмісту елемента пам'яті і коду ознаки кожен елемент пам'яті матриці АМ доповнюється схемою еквівалентності. В АЗП з логікою збігу в кожному елементі пам'яті j-го розряду слів виконується операція рівнозначності між q_{ij} і x_j з наступною кон'юнкцією всіх розрядів кожного рядка від 1 до n:

$$\left(\bigcap_{j=1}^{n}\left(q_{ij} \sim x_j\right)\right).$$

В АЗП з логікою розбіжності виконується операція нерівнозначності з наступною диз'юнкцією результатів пошуку за кожним розрядом:

$$\left(\bigcup_{i=1}^{n}\left(q_{ij} \oplus x_i\right)\right).$$

Якщо при адресній організації ЗП вибір слів за номером є однозначним, то асоціативний принцип побудови пам'яті призводить до можливості неоднозначного вибору. Виникає специфічна для АЗП проблема поділу багатозначної відповіді — послідовна вибірка слів з виділеної підмножини. У загальному випадку регістр результату пошуку може містити довільну двійкову інформацію (всі нулі, одна або декілька одиниць), що відповідає відсутності або наявності в пам'яті одного або декількох слів інформації, що відповідає критерію опитування.

У деяких випадках для ефективного використання АЗП часто потрібен точний підрахунок кількості збігів. Цю функцію виконує вузол CRI. Крім підрахунку знайдених при пошуку слів часто виникає необхідність одержати їх з пам'яті у визначеній послідовності. Для організації обробки багатозначного результату пошуку (опитування) вводиться в АЗП блок MMR, що визначає черговість вибору з безлічі

опитування слів відповідно до обраної системи їхніх пріоритетів. Протягом кожного циклу звертання до АЗП розподільник багаторазових збігів указує слово, що має найвищий пріоритет при виборі, на той час як інші слова, які виявилися відзначеними при асоціативному опитуванні рядків матриці АМ, тимчасово ігноруються. Обрані слова з подальшого перегляду виключаються.

З формальної точки зору можна сказати, що для безлічі всіх підмножин слів у асоціативній пам'яті розглянуті спеціальні засоби являють собою механізм, що забезпечує виконання аксіоми вибору. Звідси випливає, що зазначені засоби, яким треба постачати асоціативну пам'ять, повинні бути такі, щоб задавати на безлічі її числових лінійок (рядків) відношення упорядкування. Це можна зробити або шляхом введення ланцюжка черговості, що зв'язує слова асоціативної пам'яті, або шляхом використання адресного дешифратора, що здійснює відображення безлічі числових лінійок на деякий відрізок натурального ряду.

Недоліками ланцюжка черговості є мала швидкість роботи і велика кількість елементів. У зв'язку з цим в АЗП вводять адресну систему, наявність якої є одним з найважливіших властивостей швидкодіючих АЗП великої ємності. Це додає АЗП усі властивості звичайної адресної пам'яті та може бути ефективно використано не тільки при поділі багатозначної відповіді, але й у цілому ряді інших асоціативних операцій.

Якщо конструкція дешифратора в адресних ЗП звичайно допускає звертання тільки до одного слова, то у випадку АЗП дешифратор повинен забезпечити можливість одночасного звертання до групи слів; при цьому адресу можна розглядати як змінну асоціативну ознаку, окремі розряди якої можуть бути замасковані. АЗП з таким дешифратором є еквівалентним цілком асоціативному ЗП без дешифратора, що має для кожного слова збільшене число розрядів, у які внесений номер даного слова.

Таким чином, проблема поділу багатозначної відповіді зводиться до проблеми упорядкованої вибірки слів за змістом розрядів, що зберігають номери цих слів. Хоча звичайно вибірка розглядається в порядку зростання або убування номерів, через взаємну незалежність розрядів, її можна робити в будь-якому лексикографічному порядку.

На основі методів упорядкованої вибірки слів в асоціативній пам'яті можна здійснювати пошук інформації з критеріїв більш складним шляхом, ніж простий збіг; наприклад, пошук максимального

(мінімального) числа, пошук найближчого більшого (меншого) у заданих межах і т.п.

Для побудови АЗП використовуються головним чином ЕП, що реалізують операцію порівняння без руйнування збереженої в них інформації, хоча в принципі можлива побудова АЗП також на елементі пам'яті з руйнуванням інформації. Використання ЕП без руйнування інформації дозволяє виконувати операцію порівняння одночасно у всіх розрядних перетинах ЕП АЗП та не вимагає побудови складних ланцюгів відновлення вихідної інформації.

Поряд з ланцюгами управління асоціативним пошуком АЗП містить ланцюги запису і зчитування. Звичайно один з розрядів у кожному осередку використовується для вказівки його зайнятості, тобто якщо осередок вільний для запису, то в цьому розряді записаний "0". У цьому випадку при записі в АЗП нової інформації встановлюється ознака "0" у відповідному розряді регістра асоціативних ознак і визначаються всі осередки запам'ятовуючого масиву, що вільні для запису інформації. Регістр вибору слів RGWS забезпечує звертання до потрібного рядка матриці на час циклу запису або читання.

Вимоги до елемента пам'яті ВІС АЗП:

- Для здійснення асоціативного пошуку елемент пам'яті повинен забезпечити збереження інформації і її обробку (у нашому випадку виконати операцію логічного порівняння).
- Так як звертання йде до всіх слів одночасно, то споживання потужності кожним елементом повинно бути мінімальним.
- Для об'єднання асоціативних елементів пам'яті в матрицю АМ, у схемі кожного елемента пам'яті повинні бути передбачені ланцюги управління по розрядних і словникових шинах, а також ланцюги видачі відповідних сигналів у загальну для елементів пам'яті кожного слова шину результатів порівняння.

Створення ЗП асоціативного типу можливо на основі найрізноманітніших фізичних принципів. Для побудови оперативних швидкодіючих АЗП великої ємності (ємністю > 10^7 бітів і з часом асоціативного звертання < 1 мкс) найбільш сприятлива перспектива – використання надпровідникових елементів.

11.8.3. Застосування АЗП і тенденції розвитку асоціативних засобів збереження й обробки інформації

Перелік задач, для рішення яких бажане використання АЗП, займає, по Е. Хенлону, близько трьох сторінок і відноситься практично до всіх як економічних, так і військових областей.

Якщо використовувати АЗП як оперативну пам'ять для обчислювальних задач, то варто враховувати, що істотне прискорення рішення може бути досягнуте лише тоді, коли питома вага операцій асоціативного типу досить висока, тому що в противному разі виграш у швидкості може бути загублений на тлі великої кількості дій іншого типу або обмінів із зовнішніми пристроями. При цьому досягнення ефективності при використанні АЗП не виходить просто саме по собі, а вимагає деяких зусиль у справі вдосконалювання організації обчислювального процесу.

Проведені дослідження показали, що для більшості звичайних алгоритмів задач лінійної алгебри і лінійного програмування використання АЗП не призводить до значного збільшення швидкості рішення, оскільки ці алгоритми в більшості випадків використовують елементні перетворення інформаційних масивів; АЗП в даному випадку може забезпечити невеликий виграш у швидкості, що практично мало залежить від розмірів задачі. Розглянуто питання про ефективність використання АЗП для ряду екстремальних комбінаторних задач. У задачах цього типу найбільш часто повторювані обчислювальні процедури пов'язані з витратою великої кількості дій на різні досить складні операції пошуку при відносно малому обсязі арифметичних операцій. Виявляється, що в даних задачах у всіх основних процедурах виграш у швидкості при використанні АЗП пам'яттю складає $\sim n/r$, де n – розмірність задачі, а r – кількість розрядів у машинному представленні чисел, так що для розглянутого класу задач ця величина є характерним показником алгоритмічної ефективності АЗП.

На даний час основними потребами асоціативних засобів є інтелектуальні системи, призначені для рішення таких класів задач:

1. Виконання паралельних високопродуктивних обчислень. Як приклад можна навести процесори CAPP. Вони базуються на природному паралелізмі АЗП. Першим застосуванням CAPP був STARAN'74м (мав 32 блоки АЗП, розрядність кожного 256 × 256).

2. Задачі розпізнавання й аналізу образів. Приклад: інтелектуальні роботи, розпізнавання супутникової, радіоінформації, обробка зображень, розпізнавання і синтезу мови, обробка нечіткої інформації.

3. Прийняття рішень в умовах невизначеності.

4. Використання в базах даних і знань, у системах машинного перекладу і логічного висновку.

Для того, щоб ефективно вирішувати зазначені області, ємність АЗП повинна складати 1 ÷ 8 Гбайтів. Одне з можливих рішень щодо досягнення такої ємності – це використання квазі- або частково асоціативних ЗП.

Існують такі підходи до реалізації АЗП великої і надвеликої ємності:

1. Модуль пам'яті розміщається на одному кристалі.
2. Асоціативна пам'ять виконується з великого числа асоціативних модулів, підключених до загальної шини даних.
3. Асоціативні модулі виконуються на кремнієвих пластинах максимального розміру у вигляді компонування тривимірних областей (об'ємна логіка).

На даний час розробка надвеликих інтегральних схем йде в таких напрямках:

- пошук структурних і технологічних способів поліпшення характеристик АЗП для побудови високоефективних інтелектуальних систем;
- розробка спеціалізованих схем АЗП, за допомогою яких досягається значний ефект при обробці інформації в інтелектуальних системах.

11.9. Пам'ять FeRAM

Розвиток оперативної пам'яті – нескінченний процес, метою якого є збільшення частоти роботи, ширини шини, а також зменшення часу доступу, а, у деяких випадках, і збільшення стійкості до впливів ззовні, наприклад, радіації. На даний момент вимоги до пам'яті не змінились, але до перерахованого додалися нові, задовольнити які набагато складніше, ніж це було раніше. Серед них однією з найважливіших є енергонезалежність і можливість побудувати пам'ять величезної ємності в процесорі. Розробки у цьому напрямку розпочато

вже досить давно, і як результатом стала поява таких технологій, як флеш-пам'ять, FeRAM, а також MRAM. Кожна з них має свої переваги і недоліки. Флеш-пам'ять дозволяє практично нескінченно зберігати записану інформацію без необхідності проведення циклів регенерації, що необхідні для DRAM. MRAM, новий вид магніторезистивної пам'яті, що просувається на ринок фірмою Infineon Technologies AG, спрямований на просування енергонезалежної пам'яті в комп'ютерах і мобільних пристроях. Це може надати користувачам численні переваги, аж до моментального завантаження операційної системи після включення живлення комп'ютера. Принцип дії цього виду пам'яті заснований на залежності опору матеріалу від прикладеного магнітного поля. Усі ці і безліч інших переваг увібрав у себе новий вид фероелектричної пам'яті. FeRAM уміло поєднує простоту і надійність в експлуатації DRAM, енергонезалежність MRAM і час збереження інформації флеш-пам'яті. Оскільки вона включає практично всі переваги перерахованих видів пам'яті, то по праву може називатись майбутнім сучасних технологій пам'яті, на чому, власне, і роблять акцент виробники. FeRAM – це пам'ять не тільки мобільних телефонів і персональних комп'ютерів. Вона може нормально функціонувати при впливі значних доз радіації, що є імпульсом для використання такого виду пам'яті на космічних кораблях і станціях.

Процес виробництва на даний час відрізняється тим, що від моменту розробки технології до моменту масового випуску продуктів проходить дуже незначний період часу. Виробники фероелектричної пам'яті розділилися на два конкуруючих між собою табори. Одна частина, очолювана постійними партнерами у виробництві пам'яті Infineon Technologies AG і Toshiba, розробляє технологію пам'яті, засновану на виробничому процесі 1T/1C (1 транзистор /1 конденсатор) для ринку мобільних пристроїв, персональних комп'ютерів і PDA. І друга частина – це фірми, які головним чином займаються розробкою конкурентноспроможних продуктів із вбудованої FeRAM для ринку процесорів і мікроконтролерів. Основні гравці цієї групи – це NEC і Matsushita, що ведуть розробки на основі технологічного процесу 2T/2C. Це більш ресурсомістка технологія виробництва пам'яті в порівнянні з 1T/1C. Введення додаткового транзистора і конденсатора в комірку пам'яті обумовлює велику стабільність роботи, але, на той же час, збільшує площу мікросхеми тієї ж конфігурації і піднімає ціну продуктів, виготовлених за таким процесом (рис. 11.46).

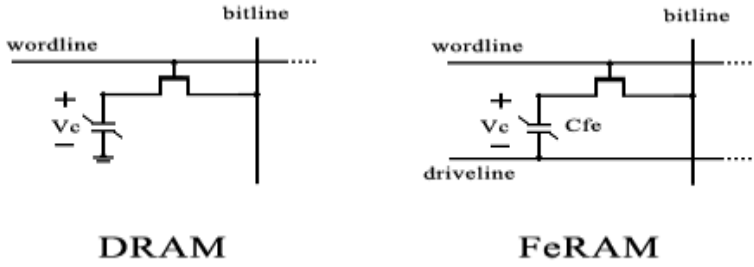


Рис. 11.46. Осередки DRAM і FeRAM

Запис у комірку пам'яті DRAM відбувається таким способом: на лінію даних (bitline) виставляється значення сигналу, яке варто записати в конденсатор. Для запису 1 на лінію даних подається напруга живлення V_{dd}. Після цього на управляючу лінію (wordline) подається сигнал, що відкриває польовий транзистор. Конденсатор заряджається, і ми маємо збережений біт інформації. В осередку FeRAM запис 1 відбувається іншим чином. Для цього на передавальну лінію подається напруга живлення, лінія даних заземлюється, а польовий транзистор знаходиться у відкритому стані.

Запис 0 відбувається подібним чином. У DRAM лінія даних підключається до землі, а транзистор відкривається. У цьому випадку конденсатор цілком розряджається, що і відповідає бінарному 0. У FeRAM запис 0 відбувається після подачі напруги живлення на лінію даних. У цьому випадку передавальна лінія підключається до землі, а транзистор тримається у відкритому стані.

Підводячи підсумок під розходженнями в роботі осередків DRAM і FeRAM, можна сказати, що інший принцип роботи осередку феромагнітної пам'яті є результатом того, що бінарним 1 і 0 відповідають різні значення поляризації, а не нульовий і одиничний заряд конденсатора, як це відбувається у випадку з DRAM.

Значення комірки пам'яті FeRAM можна визначити після подачі напруги живлення + V_{dd} на передавальну лінію. Якщо початкова поляризація феромагнетика негативна (позитивна), то читання осередку повертає менше (більше) значення сигналу на лінії даних. Одним з негативних властивостей осередку феромагнітної пам'яті є те, що, після читання вмісту, дані в ньому перестають зберігатися. Тобто,

після читання осередку, у ньому необхідно обновити значення поляризації.

11.10. Побудова модулів пам'яті

11.10.1. Побудова модулів RAM-пам'яті

Завдання 1. Побудувати модуль RAM-пам'яті з організацією 32 × 12 на мікросхемах з організацією 32 × 4.

1 етап. Визначення кількості мікросхем (МС):

Кількість БІС ЗП, яка необхідна для побудови накопичувача модуля ЗП, визначається поза залежністю від способу його побудови:

$$Q_{нк} = (N_{мод}/N_{мс})(n_{мод}/n_{мс}),$$

де $Q_{нк}$ – кількість БІС ЗП, необхідна для організації накопичувача модуля; $N_{МОД}$ – кількість слів у накопичувачі модуля ЗП; N_m – кількість слів у БІС ЗП; n – число розрядів у накопичувачі; n_m – число розрядів у БІС ЗП.

У нашому випадку:

$$Q_{нк} = (32 \times 12)/(32 \times 4) = 3 \text{ мс}.$$

2 етап. Визначення кількості адресних ліній МС:

$$n_{адр}^{МС} = \lceil \log_2 N_{мс} \rceil = \lceil \log_2 32 \rceil = 5.$$

3 етап. Визначення кількості адресних ліній модуля ЗП:

$$n_{адр}^{МОД} = \lceil \log_2 N_{мод} \rceil = \lceil \log_2 32 \rceil = 5.$$

4 етап. Побудова модуля ЗП. Схема розширення розрядності статичного ЗП (32 × 12) наведена на рис. 11.47.

Завдання 2. Побудувати модуль RAM-пам'яті з організацією 96 × 4 на мікросхемах з організацією 32 × 4.

1 етап. Визначення кількості мікросхем (МС):

Кількість БІС ЗП, яка необхідна для побудови накопичувача модуля ЗП, визначається поза залежністю від способу його побудови:

$$Q_{нк} = (N_{мод}/N_{мс})(n_{мод}/n_{мс}),$$

де $Q_{нк}$ – кількість БІС ЗП, необхідна для організації накопичувача модуля; $N_{МОД}$ – кількість слів у накопичувачі модуля ЗП; N_m – кількість слів у БІС ЗП; n – число розрядів у накопичувачі; n_m – число розрядів у БІС ЗП.

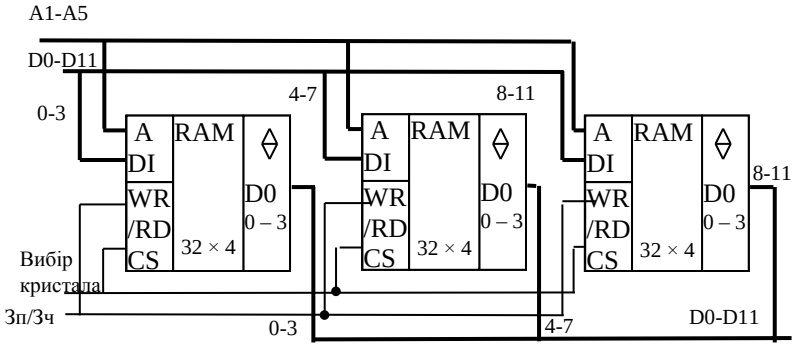


Рис. 11.47. Схема розширення розрядності статичного ЗП (32 × 12)

У нашому випадку:

$$Q_{нк} = (96 \times 4)/(32 \times 4) = 3 \text{ мс}.$$

2 етап. Визначення кількості адресних ліній МС:

$$n_{адр}^{МС} = \lceil \log_2 N_{мс} \rceil = \lceil \log_2 32 \rceil = 5.$$

3 етап. Визначення кількості адресних ліній модуля ЗП:

$$n_{адр}^{МОД} = \lceil \log_2 N_{мод} \rceil = \lceil \log_2 96 \rceil = 7.$$

4 етап. Побудова модуля ЗП. Схема розширення ємності статичного ЗП (96 × 4) наведена на рис. 11.48.

Завдання 3. Побудувати модуль пам'яті з організацією 1 к × 16 на мікросхемах з організацією 1 к × 4 та з двонаправленою шиною даних.

1 етап: визначення кількості мікросхем (МС):

$$Q_{нк} = (N_{мод}/N_{мс})(n_{мод}/n_{мс}),$$

де $Q_{нк}$ – кількість БІС ЗП, необхідна для організації накопичувача модуля; $N_{МОД}$ – кількість слів у накопичувачі модуля ЗП; N_m – кількість

слів у БІС ЗП; n – число розрядів у накопичувачі; n_m – число розрядів у БІС ЗП.

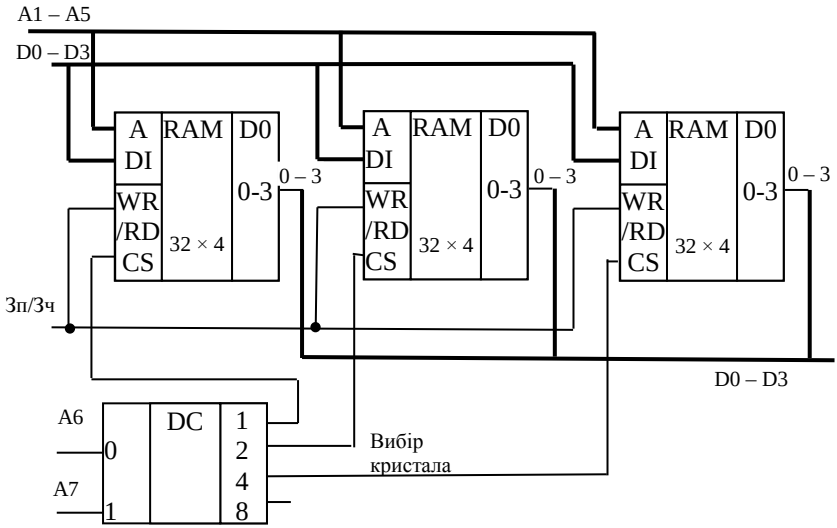


Рис. 11.48. Схема розширення ємності статичного ЗП (96 × 4)

У нашому випадку: $Q_{нк} = (1\text{ к} \times 16)/(1\text{ к} \times 4) = 4$ мс.

2 етап. Визначення кількості адресних ліній МС:

$$n_{адр}^{МС} = \lceil \log_2 N_{мс} \rceil = \lceil \log_2 1\text{ к} \rceil = 10.$$

3 етап. Визначення кількості адресних ліній модуля ЗП:

$$n_{адр}^{МОД} = \lceil \log_2 N_{мод} \rceil = \lceil \log_2 1\text{ к} \rceil = 10.$$

4 етап. Побудова модуля ЗП. Схема розширення розрядності статичного ЗП (1к × 16) наведена на рис. 11.49.

Завдання 4. Побудувати модуль пам'яті з організацією 4 к × 4 на мікросхемах з організацією 1к × 4 та з двонаправленою шиною даних.

1 етап. Визначення кількості мікросхем (МС):

$$Q_{нк} = (N_{мод}/N_{мс})(n_{мод}/n_{мс}),$$

де $Q_{нк}$ – кількість БІС ЗП, необхідна для організації накопичувача модуля; $N_{МОД}$ – кількість слів у накопичувачі модуля ЗП; N_m – кількість слів у БІС ЗП; n – число розрядів у накопичувачі; n_m – число розрядів у БІС ЗП.

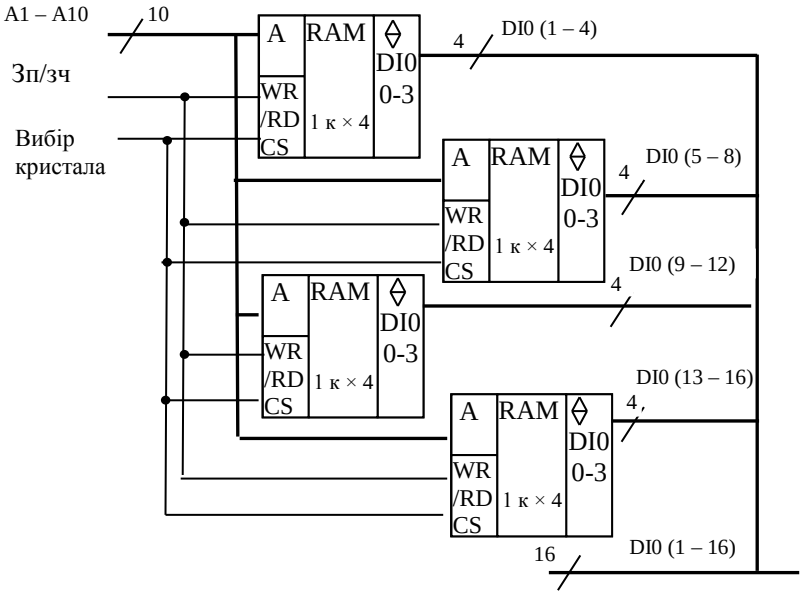


Рис. 11.49. Схема розширення розрядності статичного ЗП (1 к × 16)

У нашому випадку: $Q_{нк} = (1 к \times 16)/(1 к \times 4) = 4$ мс.

2 етап. Визначення кількості адресних ліній МС:

$$n_{адр}^{МС} = \lceil \log_2 N_{мс} \rceil = \lceil \log_2 1 к \rceil = 10.$$

3 етап. Визначення кількості адресних ліній модуля ЗП:

$$n_{адр}^{МОД} = \lceil \log_2 N_{мод} \rceil = \lceil \log_2 4 к \rceil = 12.$$

4 етап. Побудова модуля ЗП. Схема розширення розрядності статичного ЗП (4 к × 4) наведена на рис. 11.50.

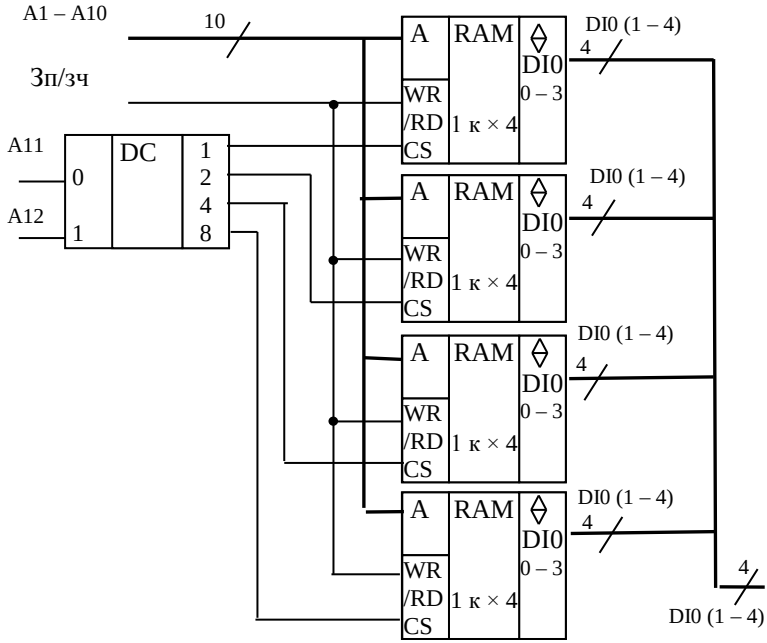


Рис. 11.50. Схема розширення розрядності статичного ЗП (4 к × 4)

11.10.2. Побудова модулів ROM-пам'яті

При побудові постійних ЗП на МС слід пам'ятати, що умовно-графічна ознака елемента постійного ЗП відрізняється від оперативного тим, що в ній відсутні поля: вхідних даних та сигналів запису-читання.

Завдання 5. Побудувати модуль ROM-пам'яті з організацією 96 × 12 на мікросхемах ПЗП з організацією 32 × 4.

1 етап. Визначення кількості мікросхем (МС):

Кількість БІС ЗП, необхідна для побудови накопичувача модуля ЗП, визначається поза залежністю від способу його побудови:

$$Q_{нк} = (N_{мод}/N_{мс})(n_{мод}/n_{мс}),$$

де $Q_{нк}$ – кількість БІС ЗП, яка необхідна для організації накопичувача модуля; $N_{МОД}$ – кількість слів у накопичувачі модуля ЗП; N_m – кількість

слів у БІС ЗП; n – число розрядів у накопичувачі; n_m – число розрядів у БІС ЗП.

У нашому випадку: $Q_{нк} = (96 \times 12)/(32 \times 4) = 9$ мс.

2 етап. Визначення кількості адресних ліній МС:

$$n_{адр}^{МС} = \lceil \log_2 N_{мс} \rceil = \lceil \log_2 32 \rceil = 5.$$

3 етап. Визначення кількості адресних ліній модуля ЗП:

$$n_{адр}^{МОД} = \lceil \log_2 N_{мод} \rceil = \lceil \log_2 96 \rceil = 7.$$

4 етап. Побудова модуля ЗП. Схема розширення розрядності та збільшення слів ПЗП (96 × 12) наведена на рис. 11.51.

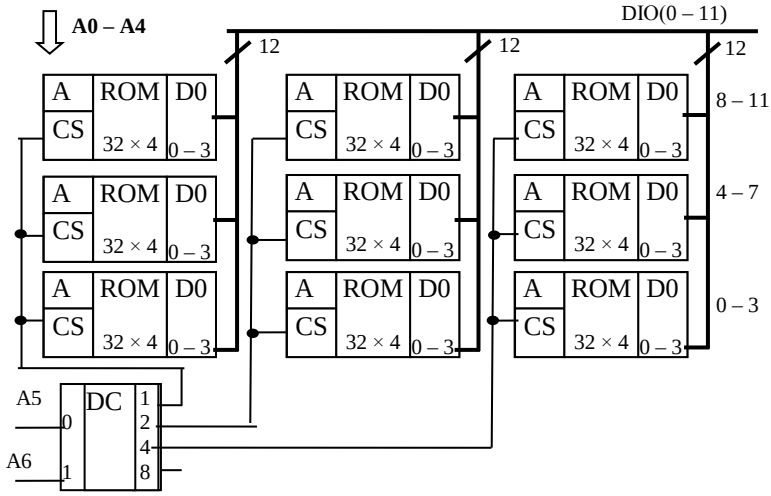


Рис. 11.51. Схема розширення розрядності та збільшення слів ПЗП (96 × 12)

Схема побудованого модуля ROM-пам’яті складається з 9 мікросхем пам’яті та дешифратора з організацією 2 × 4 для послідовного вибору стовпців підмодулів.

Варіанти завдань до вирішування наведені в табл. 11.6.

Таблиця 11.6

№	Тип МС	Орг. МС	Орг. модуля	P_{max}, мВт	№	Тип МС	Орг. МС	Орг. модуля	P_{max}, мВт
1	К531РУ8П	16 × 4	32 × 16	580	15	К1500РУ470	4 к × 1	12 к × 4	855
2	К531РУ9П	16 × 4	48 × 12	580	16	К541РУ34	8 к × 1	16 к × 8	565
3	К500РУ145	16 × 4	48 × 8	825	17	КР537РУ8А	2 к × 8	4 к × 16	6
4	К185РУ2	64 × 1	128 × 8	310	18	КР132РУ6А	16 к × 1	32 к × 4	440
5	КР556РТ14	2 к × 4	8 к × 8	740	19	К565РУ3А	16 к × 1	32 к × 8	460
6	К1500РУ073	64 × 4	256 × 12	1100	20	КР565РУ6Б	16 к × 1	48 к × 4	150
7	К1500РУ073	64 × 4	256 × 16	1100	21	К565РУ5Б	64 к × 1	128 к × 4	20
8	К500РУ410	256 × 1	1 к × 4	770	22	К568РЕ1	2 к × 8	8 к × 16	450
9	КР556РТ14	2 к × 4	8 к × 12	740	23	КР1610РЕ1	2 к × 8	4 к × 16	300
10	К500РУ415	1 к × 1	2к × 8	14	24	КР568РЕ2	8 к ×8	16 к × 16	450
11	К537РУ1А	1к × 1	4к × 4	525	25	КР556РТ12	1 к × 4	4 к × 8	740
12	К541РУ2А	1к × 4	4к × 8	525	26	КР556РТ14	2 к × 4	8 к × 12	740
13	К541РУ2А	1к × 4	4к × 12	28	27	КР556РТ16	8 к × 8	16 к × 16	1000
14	К537РУ2А	4 к ×1	8 к × 4	385	28	К573РФ33	1 к × 16	4 к × 32	700

У таблиці P_{max}, мВт позначено максимальну споживчу потужність однієї мікросхеми пам’яті.

У таблиці 11.7 наведений початковий ряд чисел N та його скороченого запису N’ на підставі ступеня двійки 2^n.

Таблиця 11.7

N	2^n	N	N’	2^n
–	–	1024	1 к	2^{10}
1	2^0	2048	2 к	2^{11}
2	2^1	4096	4 к	2^{12}
4	2^2	8192	8 к	2^{13}
8	2^3	16384	16 к	2^{14}
16	2^4	32768	32 к	2^{15}
32	2^5	65536	64 к	2^{16}
64	2^6	131072	128 к	2^{17}
128	2^7	262144	256 к	2^{18}
256	2^8	524288	512 к	2^{19}
512	2^9	1048576	1024 к	2^{20}

11.11. Завдання до самостійних досліджень

ОПЕРАТИВНА ТА ПОСТІЙНА ПАМ'ЯТІ ПК

1. МЕТА

Закріплення теоретичного матеріалу, отримання практичних навичок роботи з програмою схемотехнічного моделювання аналогових та цифрових радіоелектронних пристроїв Electronics Workbench фірми Interactive Image Technologies (Канада).

2. ТЕМИ ДЛЯ ПОПЕРЕДНЬОГО ОПРАЦЮВАННЯ

2.1. Ієрархічна структура пам'яті ПК. Типи пам'яті ПК.

2.2. Статична та динамічна пам'ять ПК.

2.3. Структурна організація комірок статичної та динамічної пам'яті.

2.4. Засоби нарощування ємності запам'ятовуючих пристроїв.

3. ПОСТАНОВКА ЗАДАЧІ

Дослідити особливості побудови та налагодження оперативних та постійних пристроїв ПК. Ознайомитися з програмою схемотехнічного моделювання аналогових та цифрових радіоелектронних пристроїв Electronics Workbench.

4. ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

4.1. Створити комірку оперативної пам'яті. Підключити на вхід схеми генератор слова, а на вихід – світлодіод чи лампочку. Оформити її у вигляді підсхеми (субблока).

4.2. Створити, в залежності від варіанта, матрицю n-бітного ОЗП. Для дослідження функціонування матриці підключити відповідні елементи.

4.3. Створити, в залежності від варіанта, матрицю запам'ятовуючого масиву n-бітного ПЗП. Для дослідження функціонування матриці підключити відповідні елементи.

4.4. Створити схему ПЗП. Реалізувати схему дешифратора

5. ВАРІАНТИ ЗАВДАНЬ

Лабораторна робота виконується по групах. Кожна група повинна побудувати комірки пам'яті статичного та постійного ЗП, а на основі

цих субблоків зібрати та дослідити модулі пам'яті згідно з варіантами за табл. 11.8.

Таблиця 11.8

№ з/п	ОЗП		ПЗП	
	Ряд.	Стовп.	Ряд.	Стовп.
1	2	2	5	2
2	2	3	4	4
3	2	4	4	3
4	3	2	4	2
5	3	3	3	4
6	3	4	3	3
7	4	2	3	2
8	4	3	2	4
9	4	4	2	3
10	5	2	2	2

6. ПОРЯДОК ВИКОНАННЯ РОБОТИ

6.1. Порядок проведення роботи для розробки принципової електричної схеми.

6.1.1. Запустіть Electronics Workbench.

6.1.2. Підготуйте новий файл для роботи. Для цього необхідно виконати такі операції з меню: File/New і File/Save as. При виконанні операції Save as буде необхідно вказати ім'я файла і каталог, у якому буде зберігатися схема. Рекомендується називати схему на прізвище виконавця.

6.1.3. Перенесіть необхідні елементи з заданої викладачем схеми на робочу область Electronics Workbench. Для цього необхідно вибрати розділ на панелі інструментів (Sources, Basic, Diodes, Transistors, Analog Ics, Mixed Ics, Digital Ics, Logic Gates, Digital, Indicators, Controls, Miscellaneous, Instruments), у якому знаходиться потрібний вам елемент, потім перенести його на робочу область.

6.1.4. З'єднайте контакти елементів і розташуйте елементи в робочій області для одержання необхідної вам схеми. Для з'єднання двох контактів необхідно клацнути по одному з контактів основною кнопкою миші і, не відпускаючи клавішу, довести курсор до другого контакту. У разі потреби можна додати додаткові вузли (розгалуження). Натисканням на елементі правою кнопкою миші можна одержати швидкий доступ до найпростіших операцій над

положенням елемента, таких як обертання (rotate), розворот (flip), копіювання/вирізання (copy/cut), вставка (paste).

6.1.5. Проставте необхідні номінали і властивості кожному елементу. Для цього потрібно двічі клацнути мишею на елементі.

6.1.6. Коли схема зібрана і готова до запуску, натисніть кнопку ввімкнення живлення на панелі інструментів. У випадку серйозної помилки в схемі (замикання елемента живлення накоротко, відсутність нульового потенціалу в схемі) буде видане попередження.

6.1.7. Зробіть аналіз схеми, використовуючи інструменти індикації. Виведення термінала здійснюється подвійним натисканням клавіші миші на елементі. У випадку потреби можна скористатися кнопкою Pause.

6.1.8. При необхідності зробіть доступні аналізи в розділі меню Analysis.

6.2. Порядок побудови елемента пам'яті статичного типу.

6.2.1. Виберіть на панелі інструментів піктограму з накресленням елемента 2I-HI, після натискання на яку випадає підменю з назвою Logic Gates. Знайдіть потрібні елементи та шляхом натискання на ліву клавішу миші пересуньте їх на аркуш документа.

6.2.2. Виберіть на панелі інструментів піктограму з накресленням RS-тригера, після натискання на яку випадає підменю з назвою Digital. Знайдіть потрібні елементи та шляхом натискання на ліву клавішу миші пересуньте їх на аркуш документа.

6.2.3. Знайдіть панель інструментів з зображенням семисегментного елемента. Шляхом натискання на ліву клавішу миші пересуньте зображення лампочки на аркуш документа.

6.2.4. З піктограми з зображенням резистора пересуньте зображення точки. Зв'яжіть елементи шляхом утримання лівої клавіші миші. Схема комірки пам'яті з елементами керування наведена на рис. 11.52.

6.2.5. Для зображення підписів елементів треба виділити елемент (клацнути лівою клавішею миші). Виділений елемент повинен стати червоного кольору. Навести стрілку курсора на елемент та натиснути праву клавішу миші. Вибрати підменю Component Properties. В ньому вибрати піктограму з назвою Lebel. Ввести назву елемента. Вікно програми EWB 5.12 зі схемою комірки статичної пам'яті зображене на рис. 11.53.

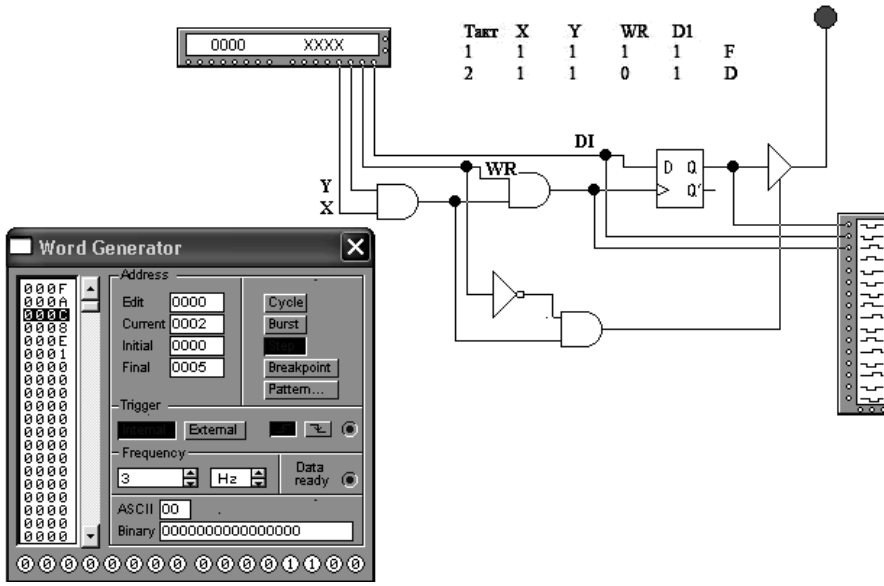


Рис. 11.52. Схема комірки пам’яті з елементами керування

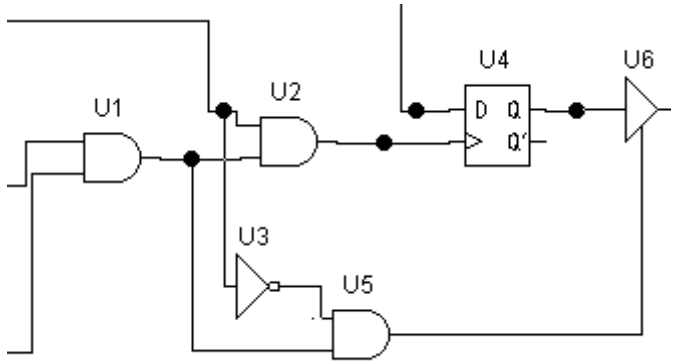


Рис. 11.53. Вікно програми EWB 5.12 зі схемою комірки статичної пам’яті

Для запису в комірку пам'яті на відповідній розрядній шині даних встановлюється 1 або 0, на вході WR/RD – сигнал 1 та після стробування лічильника або дешифратора адреси сигналом CS включаються елементи 2I: U1, U2. Перепад сигналу з елемента U2 поступає на тактовий вхід D-тригера U4, в результаті чого в нього записується 1 або 0 в залежності від рівня сигналу на його D-вході.

При читанні з комірки пам'яті на вході WR/RD встановлюється 0, при цьому спрацьовують елементи U1, U3, U5 та на вхід ДОЗВІЛ ВИХОДУ буферного елемента U6 поступає сигнал дозволу, в результаті чого сигнал Q-виходу D-тригера передається на розрядну шину даних DO0 ... DO3. Для перевірки функціонування комірки пам'яті використовується генератор слова.

6.3. Відповідно до варіанта завдання побудуйте модуль n-бітного ОЗП. Для цього потрібно виділити з допомогою лівої клавіші миші таку область схеми, в яку б не потрапили елементи, які не належать до неї. В результаті виконання команди викликається діалогове вікно, в стрічці Name якого треба ввести ім'я підсхеми. Після цього можливі варіанти:

Copy from Circuit – підсхема копіюється з назвою в бібліотеку Custom;

Move from Circuit – виділена частина вирізається з схеми та у вигляді підсхеми копіюється в бібліотеку Custom;

Replace in Circuit – виділена частина заміняється в схемі підсхемою та копіюється в бібліотеку Custom.

На рис. 11.54 наведена комірка статичної пам'яті з використанням підсхеми.

Для перегляду редагування підсхеми потрібно двічі клацнути мишею по її значку. Редагування підсхеми здійснюється за загальними правилами редагування схем. При створенні додаткового виводу необхідно з відповідної точки підсхеми курсором миші протягнути провідник до краю її вікна до появи незафарбованої прямокутної контактної площадки, після чого відпустити ліву кнопку миші. Для видалення виводу необхідно курсором миші схопитися за його прямокутну площадку в краю вікна підсхеми і винести її за межі вікна.

Конструктивно будь-яке ОЗП складається з двох блоків — матриці запам'ятовуючих елементів і дешифратора адреси. З технологічних міркувань матриця найчастіше має двокоординатну дешифрацію адреси – за рядками і за стовпцями.

На рис. 11.55 показана матриця 16-бітового статичного ОЗП: вона складається з 16 комірок пам’яті.

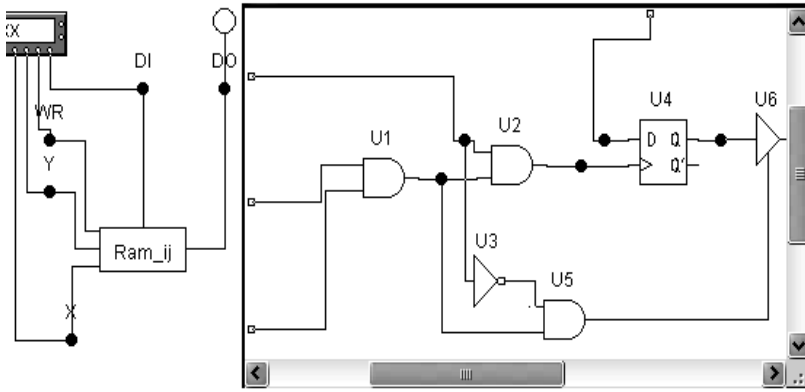


Рис. 11.54. Комірка статичної пам’яті з використанням підсхеми

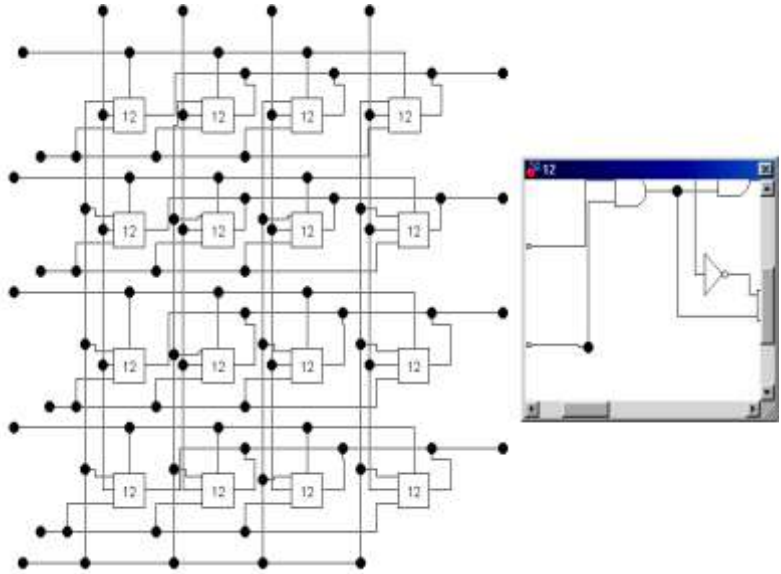


Рис. 11.55. Матриця 16-бітового ОЗП

Кожна комірка пам’яті адресується по входах X, Y шляхом вибору дешифраторами адресних ліній за рядками Ax0 – Ax3 і за стовпцями Ay0 – Ay3 і подачі по обраних лініях сигналу логічної одиниці.

При цьому в обраній комірці пам’яті спрацьовує двовходовий елемент І (U1), який підготовлює ланцюг читання-запису інформації на вхідних DI0...DI3 чи вихідних DOO...D03 розрядних шинах. Сигналом для видачі адреси є CS (chip select — вибір кристала), який подається на вхід дозволу лічильника адреси (Addr_cnt) або такий же вхід дешифраторів, підключених до виходів лічильника.

6.4. Порядок побудови запам’ятовуючого масиву пам’яті постійного типу.

Порядок побудови модуля ПЗП аналогічний до порядку побудови модуля ОЗП.

Схема нагромаджувача ППЗП наведена на рис. 11.56.

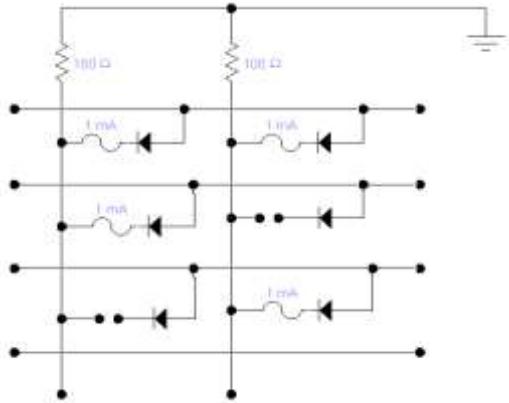


Рис. 11.56. Схема нагромаджувача ППЗП

6.5. Порядок створення схеми ПЗП.

На першому етапі створення схеми ПЗП необхідно створити схему дешифратора, та налагодити його роботу. Один із варіантів реалізації схеми дешифратора наведено на рис. 11.57.

Створення субблока дешифратора аналогічне до створення субблока комірки статичної пам’яті та наведене на рис. 11.58.

6.6. Порядок передачі зображення в текстовий редактор WORD.

Для передачі зображення або її частини в WORD необхідно:

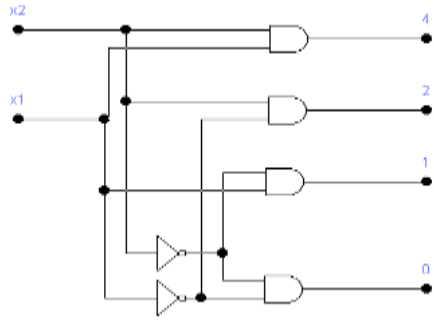


Рис. 11.57. Схема дешифратора з організацією 2 × 4

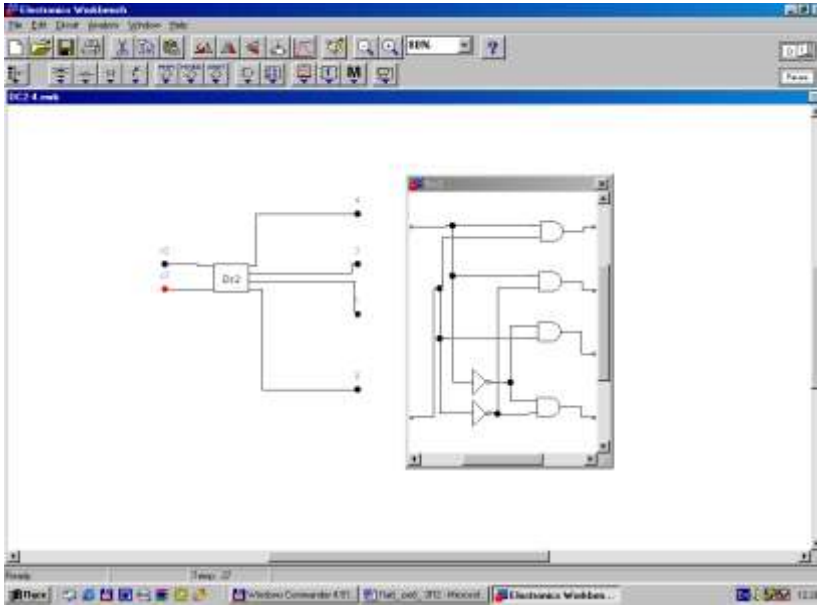


Рис. 11.58. Схема субблока дешифратора з організацією 2 × 4

6.6.1. Виділити елемент або групу елементів шляхом натискання лівої клавіші миші та її переміщення до потрібного місця.

6.6.2. Ввійти в меню Edit та натиснути кнопку Copy.

Для передачі зображення екрану в текстовий редактор WORD необхідно:

– натиснути на клавіатурі кнопку Print Screen.

Для запам'ятовування в буфері зображення активної сторінки в WORD необхідно натиснути одночасно дві клавіші Alt та Print Screen.

7. ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

7.1. Які бувають типи пам'яті?

7.2. Де в сучасних комп'ютерах використовується пам'ять статичного типу?

7.3. Чим відрізняється динамічна пам'ять від статичної?

7.4. Які типи динамічної пам'яті використовуються в сучасних комп'ютерах?

7.5. Що таке відеопам'ять та як вона пов'язана з характеристиками відображуваної на дисплеї інформації?

7.6. Що таке BIOS для комп'ютера?

ЛІТЕРАТУРА

1. Бродин В.Б., Шагурин И.И. Микроконтроллеры. Архитектура, программирование, интерфейс. – М.: Эком, 1999. – 400 с.

2. Гребнев В.В. Однокристальные микроЭВМ (микроконтроллеры) семейства MCS-9. – Псков: Коммерческая палата, 1995. – 142 с.

3. Дискретные устройства автоматизированных систем управления. Учебник. МО СССР, 1990. Под ред. Г.Н. Тимонькина, В.С. Харченко. – 512 с.

4. Зельдин Е.А. Цифровые интегральные микросхемы в информационно-измерительной аппаратуре, Л.: Энергоатомиздат, 1986. – 280 с.

5. Карлащук В.М. Электронная лаборатория на IBM PC. Программа Elektronics Workbench и ее применение, 2001. “Солон – Р”- 736 с.

6. Каспер Э. Программирование на языке Ассемблера для микроконтроллеров семейства i8051.– М.: Горячая линия – Телеком, 2003. – 191 с.

7. Козаченко В.Ф. Микроконтроллеры. Руководство по применению 16-разрядных микроконтроллеров Intel MCS-196/296 во встроенных системах управления. – М.: Эком, 1997. – 688 с.

8. Корнейчук В.И. и др.. Вычислительные устройства на микросхемах: Справочник / В.И. Корнейчук, В.П. Тарасенко, Ю.Н. Мишинский. – К.: Техніка, 1986. – 264 с.

9. Липовецкий Г.А. и др. Семейство МК48, МК51: Техническое описание и руководство по применению. – М.: Бином, 1992. – 243 с.

10. Мікропроцесорна техніка: Підручник / Ю.І. Якименко, Т.О. Терещенко, Є.І. Сокол, В.Я. Жуйкою, Ю.С. Петергеря/ За ред. Т.О. Терещенко. – К.: Політехнік, 2003. – 440 с.

11. Прикладная теория цифровых автоматов/ К.Г. Самофалов, А.М. Романкевич, В.Н. Валуйский, Ю.С. Каневский, М.М. Пиневич. – К.: Вища шк., 1987. – 375 с.

12. Применение интегральных микросхемы электронной вычислительной технике: Справочник / Р.В. Данилов, С.А. Ельцова,

Ю.П. Иванов и др..; Под ред. Б.Н. Файзулаева, Б.В. Тарабрина. – М.: Радио и связь, 1987. – 384 с.

13. Рисований О.М., Грушенко М.В. Мікроконтролерні системи. Мікроконтролери сімейства MCS-51/ За ред. О.М. Рисованого. Навчальний посібник. МО України – Х.: ХУПС, 2005. – 352 с.

14. Рисований О.М., Грушенко М.В. Цифрові пристрої та мікропроцесори. Архітектура та програмне забезпечення: Навчальний посібник. – Х.: ХУПС, 2005. – 384 с.

15. Рисований О.М., Стасєв Ю.В. Комп'ютерна схемотехніка / За ред. О.М. Рисованого: Навчальний посібник. – Х.: ХУПС, 2007. – 332 с.

16. Сергеев Н.П., Вашкевич Н.П. Основы вычислительной техники: Учеб. пособие для електротехн. спец. вузов. – М.: Высш. шк., 1988. – 311 с.

17. Сташин В.В., Урусов А.В., Мологонцев О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. – М.: Энергоатомиздат, 1990. – 224 с.

18. Фрунзе А.В. Микроконтроллеры? Это же просто! Т.1. – М.: ООО "ИД СКИМЕН", 2000. – 336 с.

19. Фрунзе А.В. Микроконтроллеры? Это же просто! Т.2. – М.: ООО "ИД СКИМЕН", 2000. – 392 с.

20. Фурман І.О., Краснобаєв В.А., Далека В.Д., Корольова Н.А., Рисований О.М. Моделі та структури даних у системах автоматизованого керування: Підручник для ВНЗ. – К., 2004. – 253 с.

21. Фурман И.А., Краснобаев В.А., Скороделов В.В., Рысованый А.Н. Организация и программирование микроконтроллеров: Учебник. – Х.: Эспада, 2005. – 248 с.

22. Цифрові пристрої та мікропроцесори. Організація та функціонування: Навчальний посібник / Рисований О.М., Соколов С.О., Зиков І.С., Скородєлов В.В. / За ред. О.М. Рисованого. – Х.: ХВУ, 2002. – 328 с.

23. Цифрові пристрої та мікропроцесори. Архітектура та програмування мікроконтролерів: Навчальний посібник / Скородєлов В.В., Рисований О.М., Даниленко О.Ф., Ліпчанський М.В. / За заг. ред. М.П. Деменка.. – МОУ, Х.: ХВУ, 2004. – 318 с.

24. Шагурин И.И. Микропроцессоры и микроконтроллеры фирмы Motorola: Справочное пособие. – М.: Радио и связь, 1998. – 560 с.

НАВЧАЛЬНЕ ВИДАННЯ

СОКОЛ Євген Іванович
КРАВЕЦЬ Валерій Олексійович
РИСОВАНИЙ Олександр Миколайович

Комп’ютерна схемотехніка

Підручник

Роботу до видання рекомендував *В.Д. Дмитрієнко*

Редактори *В.М. Баранов, Л.Л. Яковлева*

План 2007 р., п. 2

Підписано до друку 18.06.2007 Формат 60x84 1/16. Папір офісний.
Друк – ризографія. Гарнітура Times New Roman. Ум. друк. арк. 27,8.
Обл.-вид. арк. 29,8. Наклад 300 прим. Зам №. 323. Ціна договірна.
Видавничий центр НТУ’’ХПІ’’61002, Харків, вул. Фрунзе, 21.
Свідоцтво про державну реєстрацію ДК № 116 від 10.07.2000 р.

Друкарня НТУ’’ХПІ’’. 61002, Харків, вул. Фрунзе, 21.