Рябенький В.М. Жуйков В.Я. Ямненко Ю.С. Заграничний А.В.

Схемотехніка: Пристрої цифрової електроніки

ТОМ 2

Рекомендовано Методичною радою НТУУ «КПІ»
як електронний підручник для студентів,
що навчаються за спеціальністю «Електроніка»

Київ 2016

Рецензенти: *М.М. Юрченко*, д-р техн. наук, проф. (Інститут електродинаміки НАН України)

В.В. Каплун, д-р. техн. наук, проф. (Київський національний університет технологій та дизайну)

Рябенький В.М. Жуйков В.Я. Ямненко Ю.С. Заграничний А.В. „Схемотехніка: Пристрої цифрової електроніки". Електронний підручник для вищих навчальних закладів.

Електронний підручник "СХЕМОТЕХНІКА: ПРИСТРОЇ ЦИФРОВОЇ ЕЛЕКТРОНІКИ" містить в собі фундаментальний матеріал призначений для вивчення методів перетворення та обробки цифрових сигналів на основі апаратних засобів – мікросхем невисокого рівня інтеграції, які є основою для побудови складних цифрових електронних систем різноманітного функціонального призначення- від комунікаційних і комп'ютерних до пристроїв побутової техніки. Він призначений для студентів, що навчаються за напрямком "Електроніка" (06.08.09.00). Підручник може бути повністю використаний студентами, що навчаються за напрямками "Електротехніка", "Електромеханіка", "Прилади", "Комп'ютерні науки", "Комп'ютерна інженерія" для вивчення відповідних розділів електронної техніки.

Дисципліна "ЦИФРОВА ЕЛЕКТРОНІКА" має не тільки важливе самостійне практичне значення, а і є базою при вивченні мікропроцесорної техніки. Мектодологічно підручник побудований таким чином, щоб значно полегшити студентам вивчення мікропроцесорної техніки.

Автори:

Рябенький Володимир Михайлович, д-р. техн. наук
Жуйков Валерій Якович, д-р техн. наук
Ямненко Юлія Сергіївна, д-р. техн. наук
Заграничний Артур Володимирович

З М І С Т

ВСТУП

Розділ 6. ЛІЧИЛЬНИКИ ІМПУЛЬСІВ

6.1. Типи лічильників та особливості їх роботи

6.1.1. Загальна характеристика лічильників……... 9

6.1.2. Асинхронні лічильники…… 10

6.1.3. Лічильники з довільним модулем рахунку…….. 16

6.1.4. Синхронні лічильники…… 20

6.2. Серійні мікросхеми лічильників та їх використання

6.2.1. Лічильники ТТЛ…….. 26

6.2.2. Лічильники КМОН…… 37

6.3. Області використання лічильників…… 44

6.4. Скінченні автомати на основі лічильників……... 54

Контрольні питання……... 56

Вправи і завдання…….. 58

Розділ 7. РЕГІСТРИ

7.1. Загальне поняття про регістри…… 63

7.2. Регістри пам'яті……... 64

7.3. Конвеєрні пристрої…….. 73

7.4. Регістри зсуву…… 75

7.5. Приклади мікросхем регістрів та особливості їх використання

7.5.1. Паралельні регістри……..... 82

7.5.2. Регістрові файли…….. 86

7.5.3. Послідовні регістри……..... 87

7.6. Напрямки (області) використання регістрів

7.6.1. Забезпечення обміну інформацією у послідовному форматі…… 98

7.6.2. Регістрові лічильники імпульсів (розподілювачі)……... 100

7.6.3. Лічильники Джонсона……..... 105

7.6.4. Поліноміальні пристрої кодування та фільтрації……..... 111

7.6.5.Системи контролю цифрової апаратури……... 113

7.6.6. Використання регістрів для обчислення контрольної суми…… 115

Контрольні питання……... 118

Вправи і завдання……... 120

Розділ 8. ЗАПАМ'ЯТОВУЮЧІ ПРИСТРОЇ

8.1. Постійні запам'ятовуючі пристрої (принципи побудови, типи, характеристики)

8.1.1. Одновимірні ПЗП…… 123

8.1.2. Двовимірне декодування в ПЗП……..... 129

8.1.3. Мікросхеми ПЗП……. 131

8.2. Репрограмовані ПЗП

8.2.1. Принципи побудови репрограмованих ПЗП……..... 135

8.2.2. Мікросхеми РПЗП……... 140

8.2.2.1. Паралельні EEPROM……..... 141
8.2.2.2. Послідовні EEPROM……. 145
8.2.2.3. EEPROM з трьохпровідною послідовною шиною……. 152
8.2.3. Флеш-пам'ять…….. 156
8.2.3.1. Основи побудови флеш пам'яті……... 156
8.2.3.2. Мікросхеми флеш-пам'яті……..... 160
8.3. Використання ПЗП
8.3.1. Використання ПЗП як універсальних комбінаційних схем…… 174
8.3.2. ПЗП як нелінійні функціональні перетворювачі……. 177
8.3.3. Перетворювачі кодів для матричних індикаторів…… 178
8.3.4. Використання ПЗП для реалізації складних способів модуляції……... 180
8.3.5. Використання ПЗП у генераторах періодичних послідовностей……... 182
8.3.6. Використання ПЗП у скінченних мікропрограмних автоматах……. 185
8.4. Оперативні запам'ятовуючі пристрої
8.4.1. Статичні ОЗП…….. 191
8.4.2. Динамічні ОЗП (DRAM)……..... 195
8.4.3. Використання ОЗП……. 199
8.4.3.1. Використання ОЗП як інформаційного буфера……..... 203
Контрольні питання…….. 208
Вправи і завдання…….. 210

Розділ 9. ОСНОВИ ПРОЕКТУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ НА БАЗІ ПРОГРАМОВАНИХ ЛОГІЧНИХ МАТРИЦЬ (ПЛІС)
9.1. Основи побудови структур простих ПЛІС…… 212
9.1.1. ПЛМ (програмовані логічні матриці)…….. 213
9.1.2. ПМЛ (програмована матрична логіка)……. 216
9.1.3. Мікросхеми програмованої макрологіки……. 218
9.1.4. БМК (базові матричні кристали)…….. 218
9.2. Сучасні ПЛІС……... 219
9.3. Основні параметри ПЛІС…….. 222
9.4. Основи проектування цифрових пристроїв на ПЛІС в САПР MAX+plus II... 229
9.4.1. Загальний опис САПР MAX+plus II……. 229
9.4.2. Початок роботи з САПР MAX+plus II…….. 231
9.4.3. Реалізація простих логічних пристроїв на базі ПЛІС……. 234
9.4.4. Використання стандартних мікросхем в MAX+plus II……..... 270
9.4.5. Використання нестандартних елементів в MAX+plus II…… 278
Контрольні питання……... 287

Розділ 10. ІМПУЛЬСНІ ПРИСТРОЇ НА БАЗІ ЦИФРОВИХ ІНТЕГРАЛЬНИХ СХЕМ
10.1. Пристрої формування імпульсів
10.1.1. Пристрої часових перетворень на основі логічних елементів……. 289
10.1.2. Використання зовнішніх *RC*-ланок…… 291
10.1.3. Пристрої перетворення форми імпульсів…….. 303
10.2. Одновібратори
10.2.1. Одновібратори на основі логічних елементів…… 305

10.2.2. Спеціалізовані мікросхеми одновібраторів…… 308
10.2.3. Одновібратори на основі інтегрального таймера КР1006ВИ1……. 310
10.2.4. Одновібратори на основі тригерів……... 313
10.3. Генератори прямокутних імпульсів…….. 316
10.3.1. Мультивібратори на основі логічних елементів…… 319
10.3.2. Мультивібратори на основі тригерів……... 322
10.3.3. Мультивібратори на основі мікросхем одновібраторів…… 324
10.3.4. Мультивібратори на основі таймера КР1006ВИ1……. 326
10.3.5. Кварцові генератори……. 328
10.4. Універсальні генераторні мікросхеми
10.4.1. Мікросхема К1108ПП1 та її використання…… 333
10.4.2. Генератор з системою ФАПЧ К564ГГ…… 339
Контрольні питання……... 343
Вправи і завдання……... 345

Список умовних скорочень……... 352
Література……. 355

В С Т У П

Ера цифрової напівпровідникової електроніки почалась у 60-ті роки минулого століття з появою резистивно-транзисторної логіки (РТЛ), розробленої американською фірмою Fairchild. При використанні РТЛ виявилися переваги цифрової форми обробки сигналів. Але недоліки цієї технології досить швидко привели до витіснення РТЛ більш надійною діодно-транзисторною логікою (ДТЛ). Остання у вітчизняній практиці була реалізована в серії “Логіка Т” та інших. Досить швидко на її зміну прийшла транзисторно-транзисторна логіка (ТТЛ) як більш надійна і маюча значно менші масо-габаритні показники. Авторство ТТЛ належить фірмі Silvania, але масове використання ТТЛ інтегральні схеми знайшли після створення стандартних серій. Перша масова стандартна серія 74хх була розроблена фірмою Texas Instrument. Технічні характеристики різних серій ТТЛ розкрили безмежний простір у розвиткові цифрової схемотехніки, що в свою чергу приводило до подальшого зростання вимог до логічних елементів на біполярних транзисторах, до їх схемотехніки.

Наприкінці 60-х років з’явилась також логіка на польових транзисторах. Спочатку це були р-МОН і n-МОН технології, які мали важливі технологічні переваги, завдяки чому почали використовуватись при виготовленні складних і великих інтегральних схем (ВІС). За ними досить швидко з’явилась КМОН-технологія, яка в статичних режимах практично не споживала електроенергії (розробник – фірма RCA). Перші КМОН ІС мали низькі робочі частоти, були досить чутливими до статичної електрики і мали несумісні з ТТЛ рівні логічних сигналів. Але мала споживана потужність на низьких робочих частотах робили її привабливою в апаратурі з батарейним живленням, що і спонукало до її розвитку.

Протягом 70-х років основні напрямки розвитку ІС на біполярних і польових транзисторах інтенсивно розвивались. Були розроблені більш розвинуті ТТЛ ІС на транзисторах Шоткі (ТТЛШ), з'явились і інтенсивно розвивались ІС емітерно-зв'язаної логіки (ЕЗЛ), розвивалась інжекційна логіка (ІІЛ – інтегральна інжекційна логіка) та інші. Розвиток інтегральної схемотехніки відбувався на якісному рівні – узгоджувались рівні вхідних і вихідних сигналів, перешкодостійкість, підвищувались робочі частоти, але КМОН ІС значно відставали за частотними властивостями від логіки на біполярних транзисторах.

У 80-ті роки вперше з'явилась серія КМОН ІС (74НС), яка за своїми частотними властивостями не відставала від логіки на біполярних транзисторах. За нею виникли ще більш досконалі, з більшими робочими частотами, більш технологічні, що привело до використання КМОН-технологій для виготовлення мікроконтролерів, однокристальних ЕОМ, програмованих логічних матриць, які в останньому десятиріччі витіснили з практики дискретні компоненти невисокого рівня інтеграції.

Сучасний рівень електронної техніки в значній мірі визначається розвитком технологій цифрової схемотехніки. Зменшення розміру дискретного транзистора і збільшення площі використованих кремнієвих пластин дають можливість забезпечувати схемотехнічну реалізацію алгоритмів досить високої складності. Це дало поштовх у розвитку комп'ютерної техніки, телекомунікацій, мікропроцесорної техніки для управління складними технологічними процесами, побутової електроніки, пристроїв енергетичної електроніки. Поряд з розвитком технологій, інтенсивно вдосконалюються програмні засоби, які значно полегшують задачі проектування і моделювання складних електронних пристроїв на базі мікроконтролерів, ОЕОМ, програмованих логічних матриць.

Оволодіння цими досягненнями можливе лише на основі знань фундаментальних основ побудови пристроїв і алгоритмів цифрової

електроніки, базових алгоритмів функціонування цифрових систем і умінь грамотно і коректно їх використовувати у взаємозв'язку.

Пропонована книга призначена для вивчення основ цифрової схемотехніки, її базових алгоритмів з різноманітних аспектів їх використання для розв'язання практичних задач.

Автори висловлюють щиру подяку асистенту кафедри теоретичної електротехніки та електронних систем Національного університету кораблебудування ім. адмірала Макарова (м. Миколаїв)

Буряку Володимиру Святославовичу

за надану допомогу у створенні підручника.

Розділ 6
ЛІЧИЛЬНИКИ ІМПУЛЬСІВ

6.1. Типи лічильників та особливості їх роботи

6.1.1. Загальна характеристика лічильників

З попереднього розділу відомо, що *лічильники імпульсів* – це своєрідні скінченні автомати, які можуть проектуватись у відповідності до описаних у **Розділі 5** правил і необхідного алгоритму функціонування.

У загальному плані *лічильниками* називаються цифрові пристрої (автомати), призначені для підрахунку і фіксації кількості імпульсів, що подаються на їх інформаційні входи або синхровходи. Назва *"лічильники"* використовується до будь-яких послідовнісних цифрових пристроїв із замкнутим циклом діаграми станів.

Як пристрої цифрової схемотехніки, лічильники характеризуються наступними основними параметрами. Статичний параметр – *коефіцієнт перерахування* (*модуль перерахування*) M – характеризує максимальну кількість імпульсів, яка може бути подана на лічильник, щоб привести його до початкового стану. Динамічні параметри лічильників характеризують їх швидкодію. Основний динамічний параметр – *час установлення вихідного коду* ($t_{уст}$): це інтервал часу з моменту подачі вхідного імпульсу до моменту встановлення коду на виходах лічильника. Другий важливий динамічний параметр – *дозволяюча спроможність лічильника* ($t_{сд}$), яка визначається як мінімальний інтервал часу між двома вхідними імпульсами. Величина

$$f_{max} = t_{сд}^{-1} \tag{6.1}$$

називається *максимальною частотою роботи лічильника*.

Оскільки скінченні автомати можуть бути синхронними і асинхронними, то і, відповідно, ті групи автоматів, які спеціально призначені для підрахунку кількості імпульсів, можуть також бути синхронними і асинхронними.

У синхронних лічильниках, приклади яких приведені у **Розділі 5**, синхросигнал на всі тригери подається одночасно, а зміна стану тригера відбувається лише тоді, коли на інформаційних входах будь-якого тригера підготовлені відповідні дані. В асинхронних лічильниках, які часто називають *послідовними*, вхідна послідовність імпульсів подається лише на перший тригер, а решта тригерів спрацьовує в залежності від зміни стану попереднього.

6.1.2. Асинхронні лічильники

Прикладом асинхронних (послідовних) лічильників є схема, що приведена на рис. 6.1, *а*.

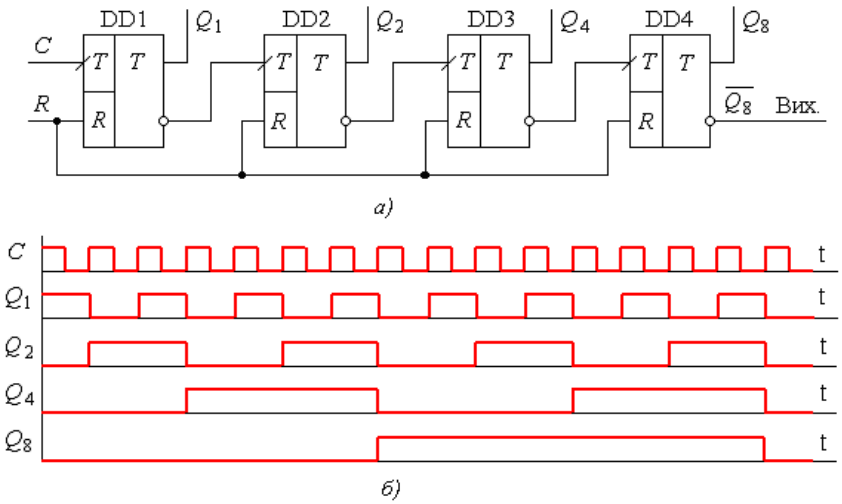


Рис. 6.1

Такі лічильники в своїй роботі використовують властивості *T*-тригера, оскільки вони можуть як зберігати свій стан, так і додавати за модулем **2** вхідний сигнал до інформації, записаної попередньо. Одиночний *T*-тригер ділить на **2** частоту вхідної послідовності імпульсів. Послідовне включення *m* таких тригерів дає можливість поділити частоту вхідних імпульсів у $M = 2^m$ разів, або утворює лічильник з коефіцієнтом перерахунку *M* (модуль рахунку, ємність лічильника).

Робота найпростішого двійкового лічильника (рис. 6.1, *а*) пояснюється даними, приведеними у табл. 6.1 і часовими діаграмами, зображеними на рис. 6.1, *б*.

Перед подачею вхідних імпульсів тригери лічильника обнуляються. При подачі першого імпульсу на вхід *C* тригер, виконаний на мікросхемі DD1, у відповідності до його алгоритму роботи, змінює свій стан на протилежний. При цьому на його прямому виході з'явиться сигналеквівалентний логічній **"1"**, а на інверсному виході встановлюється логічний **"0"**.

Таблиця 6.1

Вхідні імпульси (N)	Q_8	Q_4	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

У відповідності до схеми, тригери з динамічною синхронізацією спрацьовують за фронтом вхідного імпульсу. Стани решти тригерів не зміняться, оскільки другий тригер DD2 по своєму тактовому входу сприйме перепад з **"1"** в **"0"**, що матиме місце на інверсному виході першого тригера, а тригери DD3 і DD4 на своїх входах не матимуть ніяких змін. У результаті після першого вхідного імпульсу значення виходів тригерів відповідатимуть значенням, приведеним у табл. 6.1 в другому рядку. Другий вхідний імпульс призведе до повторної зміни стану тригера DD1. Тепер на його інверсному виході матиме місце зміна стану з логічного **"0"** на логічну **"1"**, що сприймається тригером DD2 як фронт вхідного імпульсу. В результаті тригер DD2 змінить свій стан на протилежний, і на його виході Q_2 з'явиться сигнал логічної **"1"**, у той час як вихід Q_1 тригера DD1 перейде в нульовий стан (в умовних позначеннях виходів лічильників прийнято номери виходів поєднувати з їх ваговими коефіцієнтами). Цей стан відповідатиме третьому рядку таблиці, відповідно до якого виходи тригерів зафіксують число **2** у двійковому коді. Третій імпульс знову змінить лише стан тригера DD1, прямий вихід якого відповідає молодшому розряду двійкового коду, що відображає кількість імпульсів, що були подані на вхід *C*-.

Рис. 6.1, *б* ілюструє часові співвідношення між станами тригерів, що зафіксовані їх прямими виходами. Зміни станів тригерів прийняті миттєвими.

З табл. 6.1 і часової діаграми (рис. 6.1, *б*) бачимо, що виходи тригерів лічильника в двійковому коді відображають кількість поданих на вхід імпульсів *N*, якщо ця кількість менша числа *M*.

Якщо на вхід лічильника подано *N* імпульсів, то їх кількість, підрахована за допомогою лічильника, відповідає формулі:

$$N = K \cdot M + a_0\ 2^0 + a_1\ 2^1 + \ldots + a_{m-1}\ 2^{m-1} = K \cdot M + \sum_{i=0}^{m-1} a_i\ 2^i\ , \qquad (6.2)$$

де K – кількість вихідних імпульсів лічильника; $a_i \in 1, 0$ – рівень сигналу на i-ому виході лічильника; 2^i – вагові коефіцієнти кожного прямого виходу. Для урахування вагових коефіцієнтів виходи лічильника нумерують відповідними індексами. Це дає можливість з послідовності значень виходів $Q_8\ Q_4\ Q_2\ Q_1$, рівних, наприклад, **1011**, одразу ж зчитувати вміст лічильника, тобто кількість імпульсів, яка менша *M*.

При $N = M$ усі тригери лічильника обнуляються, і такий перехід фіксується зміною стану інверсного виходу тригера DD4 з **“0”** в **“1”**, що наступними аналогічними схемами повинно сприйматись як фронт вихідного імпульсу. Це дає можливість безпосереднього нарощування розрядності лічильників шляхом прямого з’єднання входу наступного лічильника з виходом попереднього. При однотипних лічильниках кількість тригерів подвоюється, і загальний коефіцієнт перерахунку визначатиметься формулою:

$$M_З = 2^{2m} = 2^m \cdot 2^m = M \cdot M\ .$$

Звідси витікає, що при безпосередньому нарощуванні кількості лічильників загальний коефіцієнт перерахунку визначається добутком відповідних коефіцієнтів окремих лічильників.

Зазвичай код, що визначається вихідними станами тригерів, змінюється у зростаючому напрямку (*додаючі лічильники*). Операція збільшення вмісту лічильника на одиницю називається *інкрементуванням*. При зворотній зміні

станів лічильники звуться *віднімаючими*, а якщо напрямок рахунку може змінюватися, то лічильник має назву *реверсивного*. Операція зменшення вмісту лічильника на одиницю називається *декрементуванням*. Напрямок рахунку визначається як динамічними властивостями тригерів (перемикання за фронтом чи за спадом), так і способом з'єднання виходів попереднього тригера зі входом наступного. Якщо, наприклад, у лічильнику використовуються тригери, які змінюють свій стан за фронтом синхроімпульсу, то для організації додаючого лічильника необхідно з'єднати вхід наступного тригера з інверсним виходом попереднього, а для створення віднімаючого лічильника для зв'язку використовуються прямі виходи тригерів. При застосуванні тригерів, що змінюють свій стан за спадом синхроімпульсу, вказані зв'язки повинні бути протилежними. Прикладом віднімаючого лічильника може слугувати схема, що приведена на рис. 6.2.

Якщо всі тригери лічильника встановлені в одиничний стан, то при подачі першого вхідного імпульсу тригер DD1 змінить свій стан на протилежний, і на виході Q_1 запишеться значення $Q_1 = 0$. Тригер DD2 при цьому не змінить свій стан, оскільки на його вході матиме місце спад імпульсу , на який він не реагує.

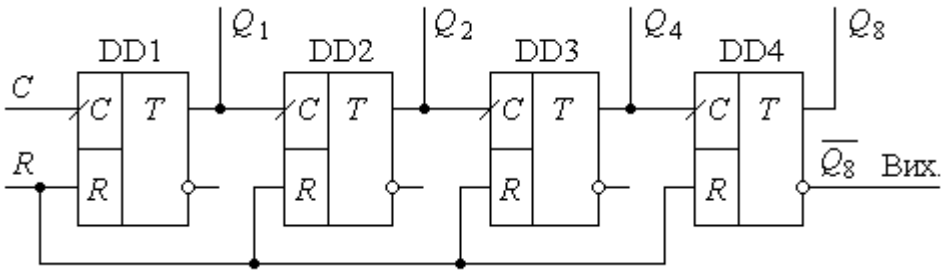


Рис. 6.2

Наступний вхідний імпульс призведе до появи на виході Q_1 сигналу одиничного рівня, при цьому тригер DD2 змінить свій стан, і на виходах тригерів зафіксується число $Q_8\,Q_4\,Q_2\,Q_1 = 1101_2 = 13_{10}$. Подібний процес продовжуватиметься до повного обнуління тригерів, а шістнадцятий імпульс забезпечить встановлення всіх тригерів в одиничний стан.

Для чотирьохрозрядного додаючого лічильника з $M = 2^4 = 16$ початковим кодом, записаним на прямих виходах тригерів, є **0000**, а кінцевим, після якого настає переповнення, – **1111**. Останній код, звичайно, є початковим для віднімаючого лічильника. У випадку, коли в якості початкового коду віднімаючого лічильника прийнятий стан **0000**, то поточні стани тригерів лічильника відображають від'ємне число зчитаних імпульсів, що представлене в допоміжному коді (див. **Розділ 1**).

Порівнюючи схеми, приведені на рис. 6.1, ***а*** і рис. 6.2, легко дійти висновку щодо можливості побудови реверсивних лічильників, які завдяки керуючому входу могли б забезпечувати додавання чи віднімання вхідної послідовності імпульсів. Для цього між тригерами необхідно встановити схему, яка б забезпечувала керовану комутацію одного з двох виходів попереднього тригера на вхід наступного. Такою схемою, як відомо, є мультиплексор **2:1.**

На рис. 6.3 приведена схема асинхронного реверсивного лічильника. Керуючий вхід V за допомогою двохвходових мультиплексорів DD2, DD4, DD6 забезпечує комутацію виходів тригерів DD1, DD3, DD5 на входи наступних – відповідно, DD3, DD5, DD7.

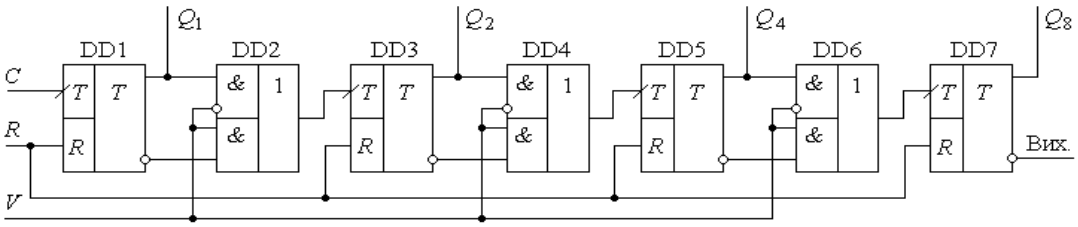


Рис. 6.3

При значенні сигналу на цьому вході $V = 1$ забезпечуватиметься підключення інверсних виходів і лічильник працюватиме на інкрементування вмісту, при $V = 0$ забезпечуватиметься режим декрементування.

R – асинхронний статичний вхід загального скидання, за яким прямі виходи всіх тригерів встановлюються в **"0"**.

За один цикл роботи лічильник приймає 2^m станів. Тому, з точки зору теорії скінченних автоматів, розглянутий лічильник – це асинхронний автомат з

замкнутим циклом роботи, в якому перехід з одного стану до іншого забезпечується вхідними інформаційними імпульсами.

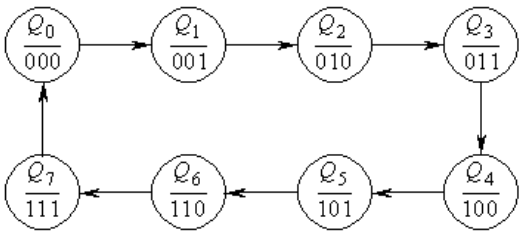


Рис. 6.4

Кожен стан кодується *m*-розрядним двійковим словом. Граф-схема лічильника за модулем 8 як скінченного автомата (діаграма станів) приведена на рис. 6.4.

Особливість розглянутих схем лічильників полягає в тому, що для зміни значення одного з старших розрядів кодового слова повинні змінюватись і всі молодші розряди. Фактично це означає, що перед зміною стану тригера, що зберігає інформацію про вміст старших розрядів кодового слова, повинні змінити свій стан і всі попередні тригери. Якщо прийняти, що зміна стану одного тригера проходить за інтервал часу затримки $t_З$, то між моментом подачі вхідного імпульсу і зміною стану тригера старшого розряду має місце часова затримка $t_{ЗМ} = m\, t_З$. Іноді цей параметр називається *часовою спроможністю лічильника*. Фізично параметр $t_{ЗМ}$ визначає мінімальний інтервал часу між двома вхідними імпульсами, при якому лічильник в найгіршому режимі перемикання встигає їх відпрацювати. Ця максимально можлива величина часової затримки визначає швидкодію лічильників при перерахунку імпульсних послідовностей. Максимально можлива частота вхідної послідовності імпульсів визначається за формулою

$$f_М = \frac{1}{m \cdot t_З}. \qquad (6.3)$$

Тому послідовний спосіб забезпечення перерахунку суттєво обмежує швидкодію двійкових лічильників. Пристрої такого типу називаються *лічильниками з послідовним перенесенням*.

Існують лічильники з іншими видами кодування станів виходів тригерів – наприклад, *унітарним* – коли стан лічильника представляється кількістю записаних в ньому одиниць; *одинарним* – коли стан лічильника визначається розташуванням однієї одиниці.

6.1.3. Лічильники з довільним модулем рахунку

Лічильники з довільним модулем рахунку мають значення M, що відрізняється від цілого ступеню числа **2**. Прикладами таких лічильників можуть служити пристрої з $M = 10$; $M = 12$; $M = 24$; $M = 60$ і т. д. На практиці доводиться мати справу з лічильниками, призначеними для ділення частоти вхідних послідовностей імпульсів у сотні, тисячі і десятки тисяч разів, і далеко не завжди коефіцієнт ділення може бути кратним 2^m (m – ціле число).

При побудові лічильників цього типу використовують такі способи:

- виключення зайвих станів;
- зворотного зв’язку;
- кратних модулів.

Найбільшого розповсюдження набув спосіб виключення зайвих станів. Його реалізація переважно здійснюється наступним чином:

- застосуванням попередньої установки лічильника;
- використанням примусового його обнуління.

На рис. 6.5 приведена схема лічильника, в якому попереднє завантаження початкового стану забезпечується через асинхронні *S*-входи *T*-тригерів за допомогою логіки DD1…DD4 і керуючого входу *PE* (паралельного завантаження). Через входи D_1, D_2, D_4, D_8 у лічильник може бути записаний будь-який двійковий код у діапазоні **0000....1111**, значення якого буде зафіксоване на виходах Q_1, Q_2, Q_4, Q_8. Запис коду забезпечується до початку подачі вхідної послідовності імпульсів на *C*-вхід. Тому з моменту подачі вхідних імпульсів лічильник рахуватиме, починаючи не з нуля, а з занесеного коду. Кількість імпульсів N, що може бути подана на *C*-вхід лічильника до

переповнення, обчислюється за формулою: $N = M - D$, тобто з M станів лічильника виключається D перших станів.

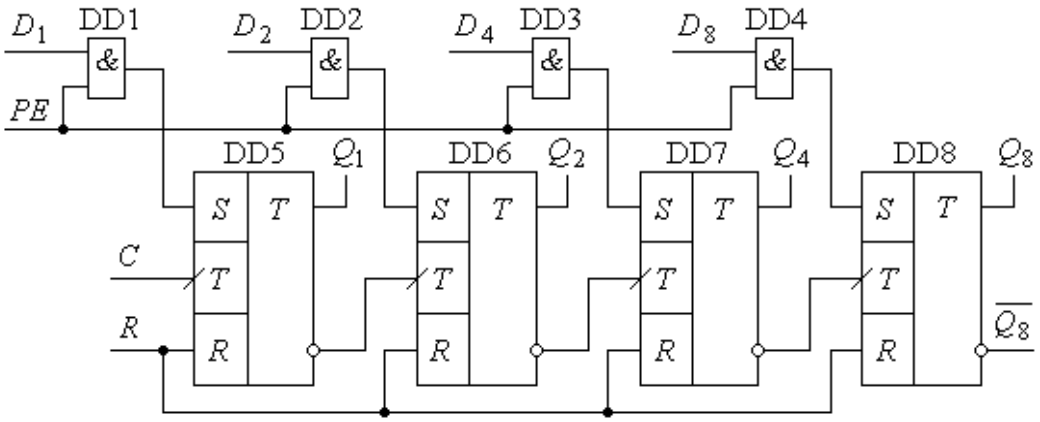


Рис. 6.5

Приклад 6.1. Забезпечити коефіцієнт перерахунку лічильника з попередньою установкою рівним 10 на одному циклі роботи.

Розв'язання. Для забезпечення коефіцієнта перерахунку 10 необхідно на входах $D_1 \ldots D_8$ встановити в двійковому коді число $16_{10} - 10_{10} = 6_{10}$ і подати короткочасний одиничний імпульс на вхід *PE*. В результаті такої дії в лічильник запишеться код, еквівалентний послідовній подачі шести імпульсів. Після дії 10 вхідних імпульсів лічильник обнулиться і без перезапису коду $0110_2 = 6_{10}$ почне відлік з нуля.

На рис. 6.6 приведені умовні позначення двійкового додаючого лічильника (рис. 6.6, *а*) та лічильника з попередньою установкою (рис. 6.6, *б*).

На відміну від двійкового лічильника, в позначенні лічильника з попередньою установкою зображені входи D ($D_1 \ldots D_8$) і допоміжний вхід паралельного завантаження *PE*. Особливість таких лічильників полягає в тому, що при циклічній роботі після завершення кожного циклу необхідно перезаписувати початковий код. Крім того, необхідно враховувати, що двійковий вихідний код у такому лічильнику при $D \neq 0$ не відповідає кількості імпульсів, поданих на вхід *C*.

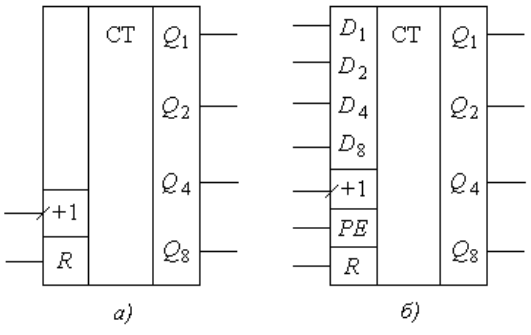


Рис. 6.6

Незважаючи на це, лічильники цього класу знаходять широке використання. Як приклад можна назвати блок адресації у мікропроцесорах. Вихідний двійковий код лічильника задає адресний простір, за яким процесор звертається до зовнішніх пристроїв та пам'яті. При необхідності переходу на нову програму в лічильник завантажується початкова адреса необхідної програми, і він починає формувати послідовно її адреси.

При використанні в якості віднімаючого, вихідний код лічильника зменшуватиметься від записаного, що широко використовується в різноманітних таймерах. При зменшенні коду до нуля наступним імпульсом лічильник встановиться в код **1111**, якщо перед цим знову не буде записаний початковий код.

Приклад 6.2. Обґрунтувати можливість побудови таймера з програмованою затримкою від 0 до 99 секунд і відображенням поточної затримки за допомогою семисегментних індикаторів.

*Пояснення***.** Для побудови таймера необхідно використати віднімаючий лічильник з попередньою установкою. Лічильник повинен мати 7 тригерів ($2^7 = 128$). Але відображати двійковий код за допомогою семисегментних індикаторів складно, оскільки необхідно встановлювати перетворювач двійкового коду в двійково-десятковий.

Задача розв'язується простіше, якщо використовувати два чотирьохрозрядні десяткові лічильники і забезпечувати в них циклічний перезапис встановленого коду після кожного обнуління.

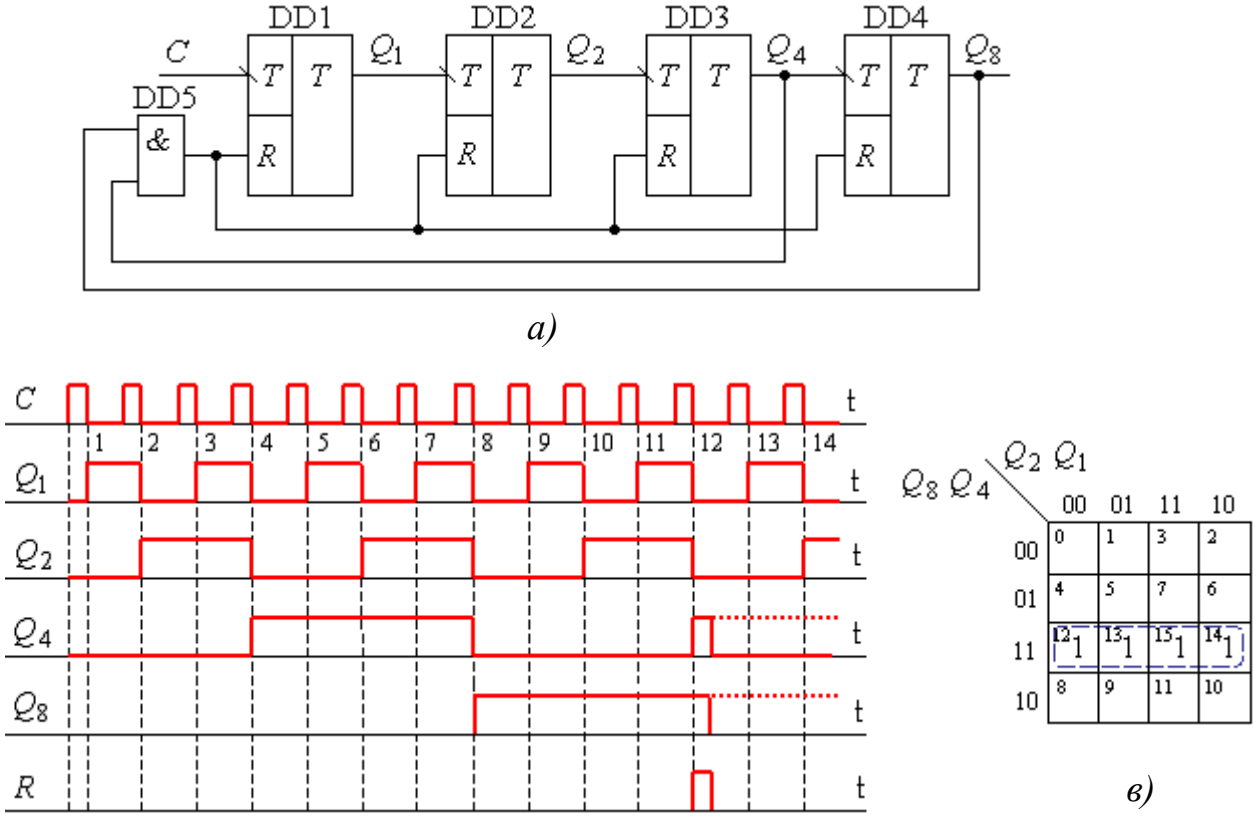
Примусове обнуління характеризується тим, що із загальної кількості станів M виключаються ті, які своїми значеннями перевищують встановлений модуль перерахунку $M_{\text{В}}$. Якщо, наприклад, лічильник з $M = 16$ повинен мати $M_{\text{В}} = 12$, то всі значення, котрі перевищують 12, повинні бути виключені. Тобто створення лічильника з модулем перерахунку $M_{\text{В}}$ досягається виконанням двох умов:

- обнуління лічильника при досягненні поточним кодом значення $M_{\text{В}}$;
- виключення всіх станів, що перевищують $M_{\text{В}}$.

Перша умова забезпечується, якщо сигнал обнуління: $R = M_{\text{В}}$.

Для лічильника з $M_{\text{В}} = 12$ маємо: $R = Q_8\, Q_4\, \overline{Q_2}\, \overline{Q_1}$.

На рис. 6.7, *а* приводиться приклад схеми двійкового лічильника з $M_{\text{В}} = 12$, а на рис. 6.7, *б* – часові діаграми, що пояснюють його роботу. Як витікає з часових діаграм, за зрізом дванадцятого вхідного імпульсу на виходах Q_4, Q_8 з'являються одиничні сигнали, які створюють сигнал обнуління



а)

б)

в)

Рис. 6.7

R усіх тригерів. Враховуючи те, що стан лічильника Q_1, Q_2, Q_3, Q_4 = 0011 є короткочасним (він існує протягом часу обнуління тригерів лічильника), його можна вважати перехідним і віднести до неробочих.

Таким чином функція виключення зайвих станів, яка забезпечує обнуління лічильника, виглядає наступним чином:

$$Y_{\text{В}} = \vee 12, 13, 14, 15,$$

яка після мінімізації за допомогою карти Карно (рис. 6.7, *в*) прийме вигляд:

$$Y_{\text{В}} = R = Q_8\, Q_4\ .$$

Спосіб зворотного зв'язку використовується досить рідко. Особливість цього способу полягає у наступному. Припустимо, що лічильник містить у собі m тригерів. Виділимо з цієї групи меншу – з m_1 тригерами, тоді $m = m_1 + m_2$. У виділеній групі заводиться зворотній зв'язок так, щоб при появі одиниць на виходах всіх тригерів перший тригер скидався в нуль. Тоді коефіцієнт перерахунку в цій групі становитиме $2^{m_1} - 1$, а загальний коефіцієнт перерахунку:

$$M = (2^{m_1} - 1) \cdot 2^{m_2} = 2^m - 2^{m_2}. \qquad (6.4)$$

Спосіб кратних модулів пояснюється схемою, що приведена на рис. 6.8.

Лічильники СТ1, СТ2, СТ3 з модулями рахунку, відповідно, M_1, M_2, M_3 з'єднані по входу паралельно, а їх виходи об'єднані елементом **3І**. Імпульс на виході з'явиться лише тоді, коли одночасно на виходах всіх лічильників з'являться вихідні імпульси. У такій схемі частота імпульсів вхідної послідовності $f_{вх}$ буде ділитися на коефіцієнт $M = M_1 M_2 M_3$, і частота вихідних імпульсів матиме значення:

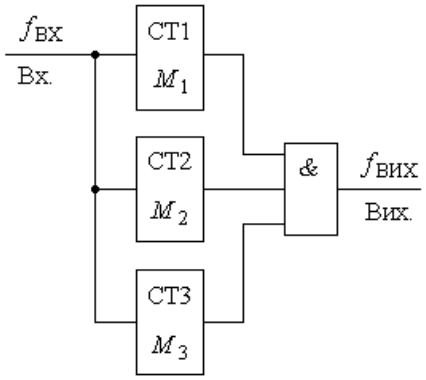


Рис. 6.8

$$f_{вих} = \frac{f_{вх}}{M}. \qquad (6.5)$$

6.1.4. Синхронні лічильники

У **Розділі 5** в якості скінченних автоматів розглядалися лічильники, в яких синхросигнали та інформаційні сигнали, що записуються в тригери, розділені по входам. В таких схемах інформаційний сигнал, що записується у тригер, підготовляється одразу після встановлення попереднього стану, а синхросигнал подається паралельно на всі тригери. Час підготовки $t_П$ – це час, який необхідно враховувати як час розповсюдження сигналу через допоміжні логічні елементи, і він завжди має величину, значно меншу, ніж час перемикання одиночного

тригера. Тому швидкодія подібного типу лічильників має бути значно більшою, ніж швидкодія лічильників з послідовним переносом. Лічильники такого типу називаються *лічильниками з паралельним переносом*.

Виходячи з цього, розглянемо синтез синхронного лічильника з паралельним переносом з використанням *JK*-тригерів, який забезпечить коефіцієнт $M = 8$.

Таблиця станів лічильника, а також сигнали управління J та K входів тригерів приведені у табл. 6.2 (умовним позначенням * зображений байдужий стан відповідного входу).

Таблиця 6.2

C_n	Q_{4n}	Q_{2n}	Q_{1n}	$Q_{4(n+1)}$	$Q_{2(n+1)}$	$Q_{1(n+1)}$	J_4	K_4	J_2	K_2	J_1	K_1
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	1	0	1	*	0	0	*	1	*
5	1	0	1	1	1	0	*	0	1	*	*	1
6	1	1	0	1	1	1	*	0	*	0	1	*
7	1	1	1	0	0	0	*	1	*	1	*	1

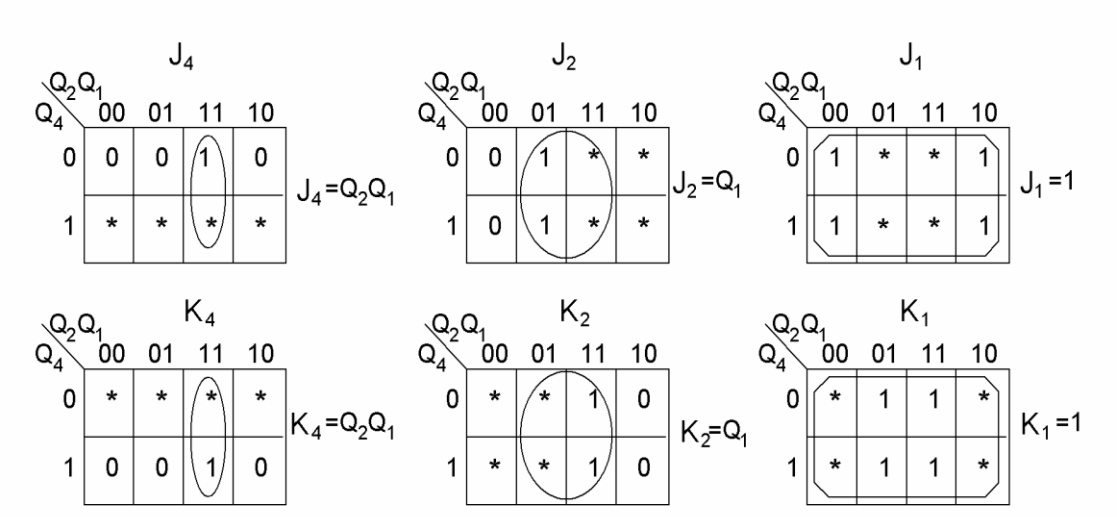


Рис. 6.9

Використовуючи карти Карно для входів J_4, K_4, J_2, K_2, J_1, K_1 (рис. 6.9), знаходимо вирази для сигналів їх збудження. Відповідно, схема лічильника приведена на рис. 6.10.

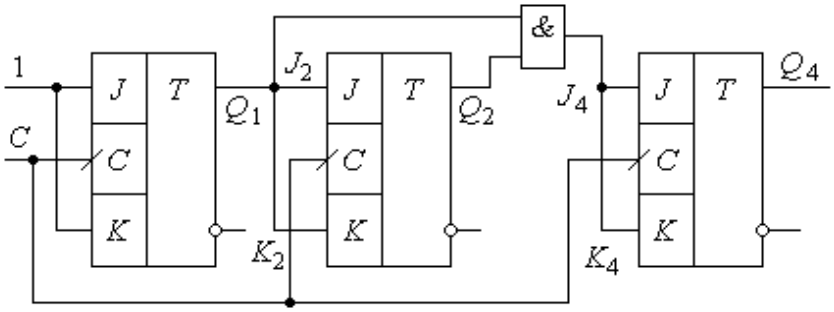


Рис. 6.10

Узагальнюючи отриманий результат, можна стверджувати, що, якщо для лічильника з $M = 2$: $J_1 = 1$; $K_1 = 1$;

для лічильника з $M = 4$: $J_2 = Q_1$; $K_2 = Q_1$;

для лічильника з $M = 8$ $J_4 = Q_1 Q_2 = J_2 Q_2$; $K_4 = Q_1 Q_2 = K_2 Q_2$;

то для лічильника з $M = 16$ матимемо:

$$J_8 = Q_1 Q_2 Q_4 = J_4 Q_4; \qquad K_8 = Q_1 Q_2 Q_4 = K_4 Q_4.$$

Відповідно, для лічильника з $M = N$:

$$J_N = Q_1 Q_2 \ldots Q_{(N-1)} = J_{(N-1)} Q_{(N-1)}; \qquad K_N = Q_1 Q_2 \ldots Q_{(N-1)} = K_{(N-1)} Q_{(N-1)}.$$

Для синхронних лічильників зворотного рахунку (віднімаючих) аналогічно можемо записати:

$$\begin{cases} J_1 = K_1 = 1; \\ J_2 = K_2 = \overline{Q_1}; \\ J_4 = K_4 = \overline{Q_1}\ \overline{Q_2} = J_2\ \overline{Q_2}; \\ J_8 = K_8 = \overline{Q_1}\ \overline{Q_2}\ \overline{Q_4} = J_4\ \overline{Q_4}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ J_N = K_N = \overline{Q_1}\ \overline{Q_2} \ldots \overline{Q_{(N-1)}} = J_{(N-1)}\ \overline{Q_{(N-1)}}. \end{cases}$$

За аналогією з асинхронними лічильниками, з використанням допоміжної логіки синхронні лічильники можуть бути реверсивними. Прикладом

лічильника з паралельним переносом є мікросхема КМОН К1561ИЕ10, спрощена схема якої приведена на рис. 6.11.

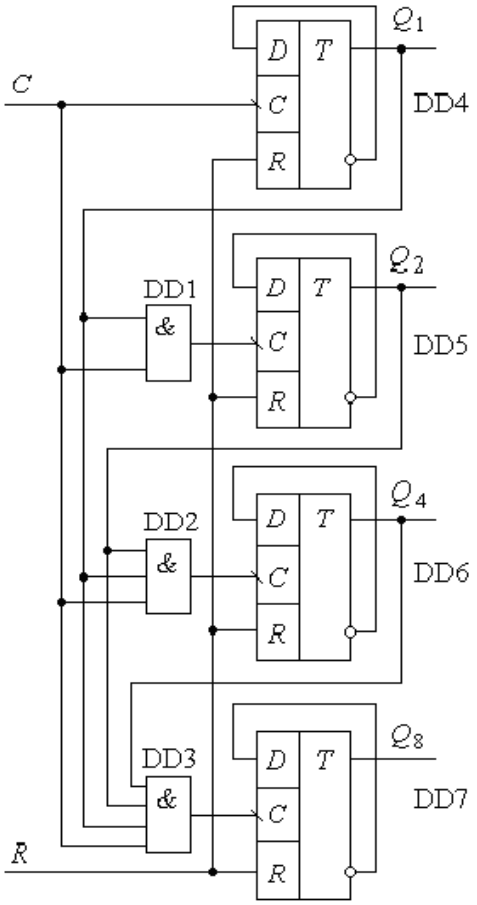


Рис. 6.11

Лічильник виготовлений на *D*-тригерах DD4…DD7, які працюють у режимі *T*-тригерів. Допоміжна логіка DD1…DD3 забезпечує швидкий доступ послідовності імпульсів, що підраховуються, до синхровходів. При подачі сигналу високого рівня на *R*-вхід усі прямі виходи тригерів Q_1, Q_2, Q_4, Q_8 встановлюються у нульовий стан. За зрізом першого тактового сигналу вихід Q_1 тригера DD4 встановлюється в **“1”** і цим підготовлює доступ синхросигналу до синхровходу тригера DD5. Другий імпульс своїм зрізом встановить тригер DD4 у початковий стан, а DD5 – в **“1”**.

Цей процес триватиме доти, поки всі тригери не будуть встановлені у стан, при якому $Q_1 = Q_2 = Q_4 = Q_8 = 1$. Наступний за цим (шістнадцятий) синхроімпульс встановить всі тригери в нуль, і цикл закінчиться.

Як випливає з роботи лічильника, всі тригери сприймають імпульси вхідної послідовності практично одночасно, аналогічно відбувається зміна їх станів, що забезпечує високу швидкодію лічильника. Фактично вона визначається часом перемикання тригера і незначною, порівняно з ним, затримкою вхідної логіки.

З аналізу роботи лічильника бачимо, що перемикання кожного наступного тригера при приході чергового імпульсу має місце лише тоді, коли всі попередні тригери встановлені в **“1”**.

Математично це можна виразити наступною формулою:

$$Q_{i(n+1)} = \overline{Q_{in}}\ p_i + Q_{in}\ \overline{p_i} = Q_{in} \oplus p_i\ , \qquad (6.6)$$

де Q_{in} – стан i-го тригера до подачі імпульсу; $p_i = Q_{0n} \cdot Q_{1n} \cdot \ldots \cdot Q_{(i-1)n}$ – узагальнений сигнал перенесення.

У розглянутих лічильниках, на відміну від пристроїв з послідовним перенесенням, напрямок рахунку не залежить від типу динамічного входу тригера, а визначається виключно тим, який з виходів тригера (прямий чи інверсний) використовується для формування сигналу перенесення. Оскільки ми впевнились, що лічильник, схема якого приведена на рис. 6.11, є додаючим, то при використанні інверсних виходів тригерів отримаємо лічильник віднімаючий. Таким чином алгоритм перемикання в реверсивних лічильниках (додавання/віднімання) залишається незмінним.

Недоліки лічильників з паралельним перенесенням полягають, перш за все, у критичності до тривалості вхідних імпульсів та їх фронтів. Тому при використанні таких пристроїв слід звертати увагу на довідкові дані по цим параметрам. Обумовлено це тим, що, незважаючи на те, що тригери лічильника перемикаються майже одночасно і час установки двійкового коду $t_{КУ}$ на виході лічильника фактично визначається часом установки тригера $t_Т$, для підготовки до наступного перемикання необхідно, щоб пройшов деякий час, який називається часом підготовки $t_П$. За цей час послідовно формуються сигнали перенесення на всіх входах елементів **І**. Величина $t_П$ залежить від встановленого в попередньому такті коду.

Складність практичної реалізації лічильника з великим значенням модуля M є другим їх недоліком. Це обумовлено ускладненням логіки переносу з великою кількістю входів.

Задача нарощування лічильників розв’язується декількома шляхами. Наприклад, у мікросхемі К1561ИЕ10 синхровхід запаралелюється через елемент **АБО** допоміжним інверсним входом $\overline{ES}$. Цей вхід дає можливість організувати рахунок як за фронтом імпульсу, так і за зрізом, а при

послідовному нарощуванні використовується як вхід асинхронного вводу від виходу Q_8 попереднього лічильника. Такий спосіб нарощування називається *комбінованим* (*послідовно-паралельним*).

На практиці часто використовуються структури з *комбінованим паралельно-паралельним перенесенням*. Особливість такого перенесення ілюструється рис. 6.12.

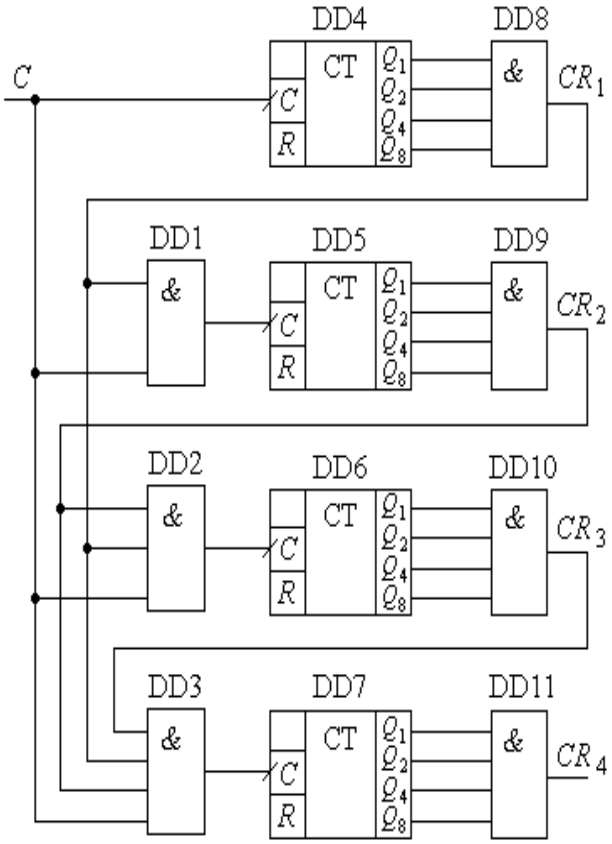


Рис. 6. 12

За аналогією зі схемою рис. 6.11, логічні елементи DD1…DD3 забезпечують паралельне перенесення від лічильників DD4…DD7. Формування сигналів перенесення $CR_1 \ldots CR_4$ з кожного лічильника виконують логічні елементи DD8…DD11 при умові, що виходи $Q_1 \ldots Q_8$ кожного з лічильників встановлені в **“1”.** У такій структурі лічильника швидкодія визначається часом установки коду $t_{КУ}$. Ця величина практично визначається часом установки коду в межах одного лічильника.

Час підготовки $t_{\text{П max}}$ у лічильнику визначається сумою затримок у логічних елементах **І**. Тому максимальна частота перемикання визначатиметься як:

$$f_{\max} = \frac{1}{t_{\text{КУ}} + t_{\text{П max}}}. \qquad (6.7)$$

6.2. Серійні мікросхеми лічильників та їх використання

Властивості лічильників в основному визначаються тими зв'язками, які закладені в них для передачі вхідної послідовності імпульсів на входи тригерів, а також для передачі сигналів стану тригерів молодших розрядів на інформаційні входи наступних. Найчастіше використовують такі типи зв'язків: *безпосередній*; *за допомогою кіл послідовного перенесення*; *за допомогою кіл паралельного перенесення*.

Методологія побудови лічильників ТТЛ- і КМОН-технологій майже однакова, але більші можливості інтеграції КМОН-технологій дозволяють розширити функціональні можливості цієї елементної бази.

Розглянемо особливості побудови і використання лічильників обох технологій.

6.2.1. Лічильники ТТЛ

Прикладом пристрою з безпосереднім зв'язком є лічильник КР1533ИЕ5, схема якого приведена на рис. 6.13.

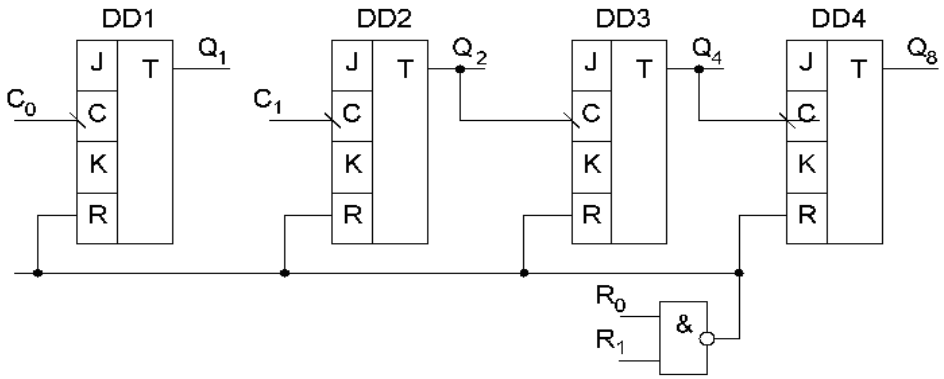


Рис. 6.13

Даний лічильник, який відноситься до ряду найбільш простих , складається з чотирьох *JK*-тригерів DD1…DD4, які працюють в режимі *T*-тригерів. Тригер DD1 має вільний вихід і може використовуватись для ділення на **2** частоти імпульсів, що поступають на вхід C_0. Тригери DD1…DD4 утворюють лічильник з коефіцієнтом перерахунку $M = 8$ для імпульсної послідовності, що подається на вхід C_1. При з'єднанні виходу Q_1 тригера DD1 з входом C_1 тригера DD2 утворюється лічильник з $M = 16$. Входи R_0 і R_1 забезпечують два можливі режими: *режим блокування* та *режим перерахунку*.

Лічильник КР153ЗИF5 являється асинхронним за структурою, особливістю якого є неодночасність спрацювання тригерів (всіх або хоча б двох). Загальний недолік асинхронних імпульсних лічильників – послідовне в часі спрацювання тригерів і, в результаті, значний час реакції останнього тригера на зміну вхідних сигналів.

Таблиця 6.3

Номер імпульсу (*N*)	Q_8	Q_4	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Таблиця 6.4

Входи		Виходи			
R_0	R_1	Q_1	Q_2	Q_4	Q_8
1	1	0	0	0	0
0	0/1	Підрахунок			
0/1	0	Підрахунок			

Таблиця 6.5

Вхід	Вихід	Час розповсюдження	
		При вмиканні, нс	При вимиканні, нс
C_0	Q_1	18	16
C_0	Q_8	70	70
C_1	Q_2	21	16
C_1	Q_8	51	51

У довідковій літературі для пояснення режимів роботи лічильника приводяться таблиці станів. Перша з них (табл. 6.3) відображає залежність станів виходів лічильника від кількості вхідних імпульсів при $M = 16$. Друга таблиця (табл. 6.4) визначає залежність виходів тригерів лічильника $Q_1 \ldots Q_8$ від значень входів R_0, R_1.

У якості динамічних параметрів задається час розповсюдження сигналу від входу до виходу. Для лічильника з двома входами, що розглядається, часові затримки приводяться у табл. 6.5.

Нарощування таких лічильників з метою збільшення коефіцієнта M досягається шляхом з'єднання виходу Q_8 одного лічильника з входом C_0 або C_1 іншого.

Приклад 6.3. Використовуючи лічильник КР1533ИЕ5, розробити схему лічильника з $M = 128$.

Розв'язання. Враховуючи, що $M = M_1 * M_2$, знаходимо M_2:

$$M_2 = \frac{M}{M_1} = \frac{128}{16} = 8 \ .$$

Отже, другий лічильник повинен мати коефіцієнт перерахунку, рівний 8. Тому загальна схема лічильника прийме вигляд, приведений на рис. 6.14. Перший лічильник DD1 має $M_1 = 16$, а другий DD2 – $M_2 = 8$.

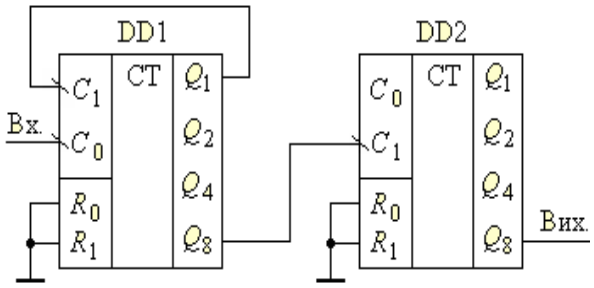


Рис. 6.14

Для обнуління станів лічильника використовуються входи R_0 і R_1, які також є асинхронними, і при високому рівні сигналу на них всі виходи лічильника приймають нульовий (низький) рівень.

Використовуючи об'єднані по **І** входи обнуління R_0 і R_1, маємо можливість реалізувати будь-який модуль перерахунку від двох до шістнадцяти за рахунок кіл зворотнього зв'язку.

Аналогічну структуру мають лічильники ИЕ2 та ИЕ4. Різниця полягає в тому, що в ИЕ2 три тригери об'єднані в схему, що дає $M = 5$, а в ИЕ4 – аналогічно, три тригери забезпечують модуль $M = 6$. Перша мікросхема легко реалізує схему ділення на 10, а друга – схему ділення на 12.

Близькі за принципом роботи двійкові асинхронні лічильники – мікросхеми ИЕ19. В них розміщується два лічильники з динамічним входом

(відлік за зрізом) і потенційним низькорівневим асинхронним входом обнуління.

Лічильник ИЕ14 має структуру, аналогічну лічильнику ИЕ2 (тригери розділені на дві групи: на одному тригері забезпечується коефіцієнт ділення на **2**, а на решті трьох – коефіцієнт ділення на **5**). У той же час, ИЕ14 має значно більші функціональні можливості завдяки використанню *JK*-тригерів та допоміжної керуючої логіки.

На рис. 6.15, *а* приведена схема модулю з $M = 2$. Умовне зображення лічильника приводиться на рис. 6.15, *б*. Вибір режимів лічильника забезпечується за допомогою керуючих входів (табл. 6.6).

Таблиця 6.6

Режими роботи	Вхід				
	R	$\overline{PE}$	C	D_n	Q_n
Установка	0	x	x	x	0
Завантаження	1	0	x	0/1	0/1
Режим лічильника	1	1	⌐	x	Рахунок

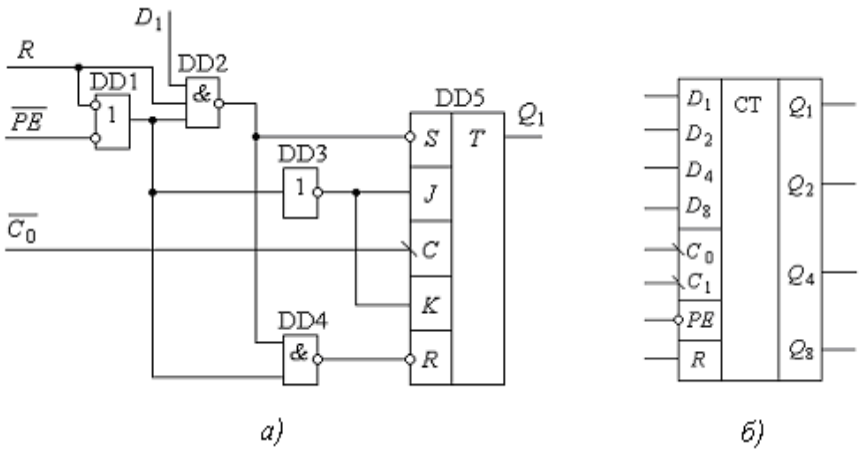


Рис. 6.15

Як і в попередніх розглянутих схемах, для забезпечення модулю перерахунку $M = 10$ необхідно з'єднати вихід Q_1 із входом C_1, а імпульси для підрахунку подавати на вхід C_0. Коефіцієнт $M = 5$ забезпечується при подачі вхідних імпульсів на C_1. Оскільки лічильник декадний, то він часто використовується в схемах для перетворення кількості вхідних імпульсів у

двійковий код, який через дешифратори подається на семисегментні індикатори. Тому при побудові таких схем слід пам'ятати, що лічильник асинхронний і значення його станів $Q_1 \ldots Q_8$ встановлюються неодночасно. Застосовані в таких схемах дешифратори повинні мати вхід дозволу, щоб не передавати на індикатори хибні коди.

Лічильник ИЕ15 – двійковий лічильник з попередньою установкою, асинхронний, виконаний за структурою, подібною до ИЕ14.

Лічильники, які виготовляються за схемами, подібними до ИЕ5, характеризуються простотою, легкістю нарощування, високою надійністю при змінній частоті і тривалості вхідних імпульсів. Їх недолік полягає у значній затримці перемикання, величина якої залежить від коефіцієнта перерахунку.

Прикладом ***синхронних лічильників*** з попередньою установкою є мікросхеми ИЕ6 та ИЕ7. Обидва вони реверсивні, ИЕ6 – двійково-десятковий, а ИЕ7 – двійковий. Умовні позначення цих лічильників приводяться на рис. 6.16, *а – б* відповідно. Структурно вони побудовані за принципом, аналогічним рис. 6.13, з одночасним перемиканням всіх підготовлених тригерів.

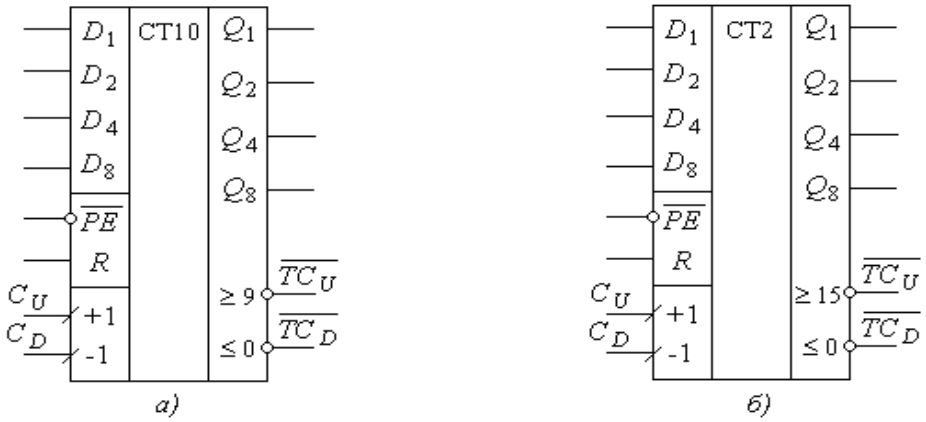


Рис. 6.16

Тактові входи C_U – для відліку на збільшення та C_D – на зменшення вихідного коду лічильника є прямими, динамічними та розділені між собою. Це суттєво розширює їх можливості, наприклад, для використання в якості

частотних інтеграторів. Напрямок відліку (на збільшення чи зменшення) визначається тим, на який з тактових входів подаються позитивні ($0 \rightarrow 1$) перепади вхідних імпульсних послідовностей.

Робота лічильників пояснюється відповідною таблицею станів, що приведена в табл. 6.7.

Для забезпечення попереднього запису за допомогою входів $D_1 \ldots D_8$ необхідно забезпечити подачу низьких рівнів сигналів на входи R і $\overline{PE}$. Значення сигналів на динамічних входах – довільні.

Для спрощення побудови лічильників шляхом з'єднання декількох мікросхем з метою збільшення коефіцієнта M мікросхеми мають виводи сигналів закінчення відліку на збільшення $\overline{TC_U}$ та на зменшення $\overline{TC_D}$.

Таблиця 6.7

Режими роботи	Вхід			
	R	$\overline{PE}$	C_U	C_D
Установка в стан логічного нуля	H	x	x	x
Запис інформації	L	L	x	x
Відлік на збільшення	L	H	⎍	H
Відлік на зменшення	L	H	H	⎍

Інверсні значення цих сигналів забезпечують можливість отримання фронту імпульсу на цих виходах при обнулінні вмісту лічильника після переповнення. Завдяки цьому в наступний лічильник запишеться **"1"**. З іншого боку, при появі 9-го (15-го) імпульсу є можливість без допоміжної логіки забезпечити перезапис даних, що подаються на D-входи.

Приклад 6.4. Використовуючи лічильник ИЕ7, розробити схему для забезпечення циклічного режиму роботи з коефіцієнтом перерахунку $M = 12$.

Розв'язання. Для забезпечення коефіцієнта $M = 12$ необхідно організувати циклічний перезапис по входах $D_1 \ldots D_8$ числа $4_{10} = 0100_2$. Тому схема матиме вигляд, приведений на рис. 6.17.

Приклад 6.5. Використовуючи попередній приклад, розробити схему лічильника з коефіцієнтом перерахунку $M = 144$.

Розв'язання. Виходячи з того, що $M = 144 = M_1 M_2 = 12 \cdot 12$, необхідно послідовно з'єднати два лічильники з $M = 12$, тобто приєднати вихід $\overline{TC_U}$ першого лічильника до входу C_U другого лічильника, забезпечивши для кожного з них з'єднання входів і виходів у відповідності до рис. 6.17.

Згідно з табл. 6.7, входи R та $\overline{PE}$ є асинхронними, потенційними і пріоритетними, порівняно з входами C_U і C_D, тобто при дії сигналу R (високий рівень) всі виходи лічильника $Q_1 \ldots Q_8$ встановлюються в нуль. Аналогічно, при дії низького рівня на вході $\overline{PE}$ значення виходів встановлюються в відповідності зі значеннями сигналів на входах $D_1 \ldots D_8$.

При нарощуванні мікросхем лічильників входи $\overline{PE}$ і R об'єднуються загальними шинами і використовуються як загальні входи для всіх мікросхем.

Приклад 6.6. Використовуючи лічильник ИЕ7, розробити схему лічильника з коефіцієнтом перерахунку $M = 177$.

Розв'язання. Враховуючи, що $M = 177$ не може бути записаний у вигляді добутку M_1 і M_2 двох окремих лічильників, схема може бути побудована як лічильник з $M = 256 = 16 \times 16$ з попередньою установкою числа $(256 - 177)_{10} = 79_{10} = 1001111_2$ з циклічним перезаписом після переповнення. Внаслідок цього отримуємо схему, що приведена на рис. 6.18.

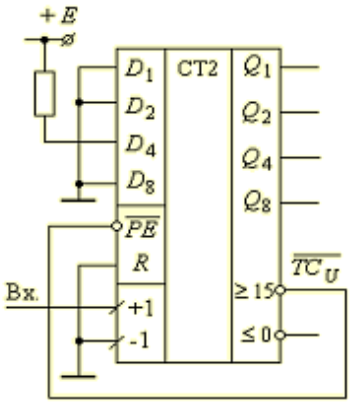


Рис. 6.17

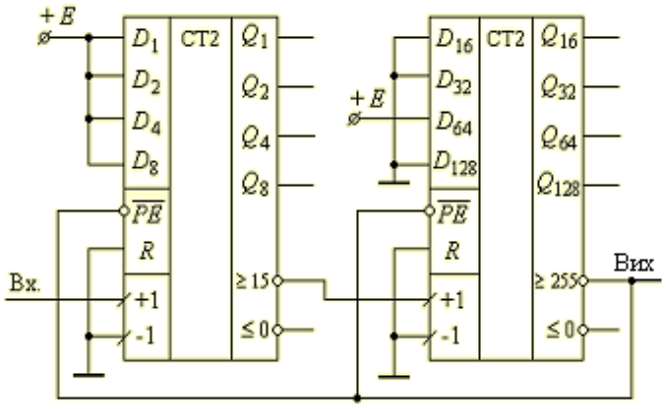


Рис. 6.18

Близькими до розглянутих ИЕ6, ИЕ7 є відповідно лічильники ИЕ16, ИЕ17. Вони мають дещо іншу логіку керування, її особливість відображена в умовному позначенні (рис. 6.19) та таблиці станів мікросхем (табл. 6.8).

Мікросхеми мають один тактовий вхід C, а зміна напрямку відліку забезпечується статичним сигналом $U/\overline{D}$.

Паралельне завантаження даних по входах $D_1 \ldots D_8$ дозволяється низьким рівнем сигналу на вході $\overline{PE}$. При цьому відлік зупиняється, і за фронтом синхроімпульсу C дані з шини D записуються в лічильник.

Для каскадного з'єднання мікросхем використовуються два входи – $\overline{CET}$ і $\overline{CEP}$. Вхід $\overline{CET}$ наступного лічильника з'єднується з виходом $\overline{TC}$ попереднього. Входи $\overline{CEP}$ об'єднуються в загальну шину допоміжного дозволу. За більш детальною інформацією по каскадному з'єднанню таких лічильників слід звернутися до спеціальної літератури.

Таблиця 6.8

Режими роботи	Входи						Виходи	
	C	$U/\overline{D}$	$\overline{CEP}$	$\overline{CET}$	$\overline{PE}$	D_n	Q_n	$\overline{TC}$
Паралельне завантаження	↑	x	x	x	0	0	0	1*
					0	1	1	1*
Відлік на збільшення	↑	1	0	0	1	x	Збільшення	1*
Відлік на зменшення	↑	0	0	0	1	x	Зменшення	1*
Зберігання		x	1	x	1	x	Q_n	1*
			x	1				1

У практиці проектування цифрових пристроїв мікросхеми ИЕ16 та ИЕ17 знаходять значно менше використання, ніж ИЕ6 та ИЕ7.

Своєрідним лічильником є мікросхема ИЕ8, яка має спеціальне призначення – програмоване ділення частоти вхідних імпульсних послідовностей. Умовне позначення мікросхеми приводиться на рис. 6.20.

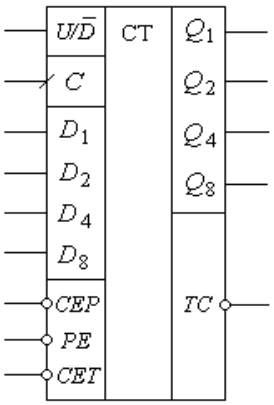


Рис. 6.19

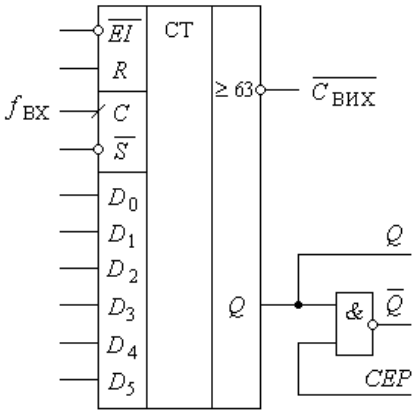


Рис. 6.20

Вихідними виводами є вихід переносу $\overline{C}_{\text{вих}}$, прямий вихід Q і комплементарний $\overline{Q}$ – виходи для виводу кількості імпульсів у відповідності до коефіцієнта перерахунку. Частота імпульсів на виході Q при подачі на вхід C імпульсної послідовності з частотою $f_{\text{вх}}$ обчислюється за формулою:

$$f_{\text{вих}} = \frac{f_{\text{вх}}}{64} \cdot \left(D_5 \cdot 2^5 + D_4 \cdot 2^4 + D_3 \cdot 2^3 + D_2 \cdot 2^2 + D_1 \cdot 2^1 \cdot D_0 \cdot 2^0 \right) = \frac{f_{\text{вх}}}{64} \sum_{i=0}^{5} D_i \cdot 2^i , \quad (6.8)$$

де D_i можуть приймати значення **1** або **0**.

Зрозуміло, що найбільш широко використовується кратне ділення частоти на 2^i, оскільки в такому випадку отримується симетрична вихідна послідовність. Але мікросхема забезпечує довільний (у заданих межах) коефіцієнт ділення, при цьому вихідна послідовність матиме різні інтервали часу між імпульсами.

Для обнуління лічильника використовується асинхронний вхід R, високий рівень на якому при одночасному високому рівні на вході $\overline{S}$ забезпечує обнуління всіх тригерів лічильника, а також переведення виходу $\overline{C}_{\text{вих}}$ і Q у стан високого потенціалу.

Сигнал дозволу, що подається на вхід $\overline{EI}$ (*Enable Input*), має низький рівень, але для забезпечення режиму ділення частоти необхідно, щоб одночасно і на вхід $\overline{S}$ також подавався сигнал низького рівня. Зупинка роботи мікросхеми забезпечується високим рівнем сигналу на вході $\overline{S}$. Вихід *CEP* використовується для нарощування мікросхем.

Функціонально мікросхема складається з двох вузлів – безпосередньо синхронного лічильника з попередньою установкою на шести *D*-тригерах та логічної схеми вибору імпульсів, побудованої на багатовходових логічних елементах **І** та **АБО**. Алгоритми побудови програмованих лічильників досить широко використовуються в задачах мікропроцесорної техніки, тому, не вдаючись у детальний аналіз роботи мікросхеми, розглянемо його на прикладі схеми пристрою, що приведена на рис. 6.21, *а*. Часові діаграми, які пояснюють роботу приведені на рис. 6.21, *б*.

За фронтом синхроімпульсу C вихід Q тригера, підготовленого по входу D, встановлюється в **“1”** і цим забезпечує можливість при наявності дозволяючих входів S і D_i отримати на виході Q_f вихідної послідовності імпульсів, частота яких співпадає з частотою зміни сигналів високого рівня на прямому виході тригера. Такий алгоритм формування дозволяючих інтервалів для передачі поділених вхідних імпульсів може бути легко реалізований для широкої гами задач.

Для цього необхідно побудувати часові діаграми на прямих та інверсних виходах тригерів на одному циклі перерахунку і на їх основі встановити логічні залежності, які необхідно використати для задач вибору вхідних імпульсів.

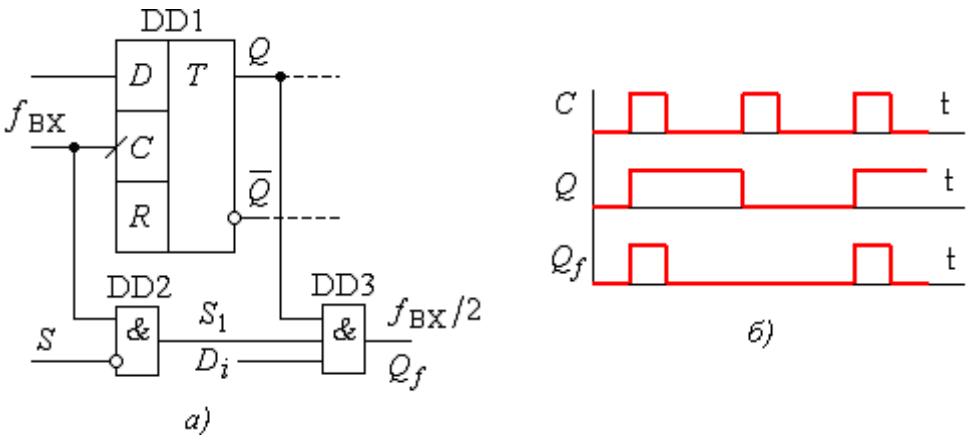


Рис. 6.21

Звертаючись, наприклад, до часових діаграм, що приведені на рис. 6.1, *б*, можемо встановити, що для забезпечення коефіцієнта ділення на **4** (тобто отримання на виході чотирьох імпульсів з шістнадцяти) необхідно, щоб на вхід логічного елемента вибору (DD3 на рис. 6.21, *а*) був поданий дозволяючий сигнал $S_1 = \overline{Q}_1\,\overline{Q}_2$ або $S_1 = Q_1\,Q_2$. Таким шляхом можна забезпечити широкий діапазон коефіцієнтів ділення з симетричним або несиметричним формуванням вихідної імпульсної послідовності.

Група лічильників ИЕ9, ИЕ10, ИЕ11, ИЕ13, ИЕ18 – це синхронні лічильники, близькі по своїй структурі. Лічильник ИЕ9 – двійково-десятковий, решта – двійкові. Різниця в логіці їх функціонування досить незначна, тому

розглянемо лише особливості роботи мікросхеми ИЕ9. Умовне позначення її приведене на рис. 6.22, а таблиця станів – у табл. 6.9.

Внутрішня логіка лічильників побудована таким чином, що забезпечується висока швидкодія, а наявність допоміжних входів дозволу *CEP* (паралельний) та *CET* (допоміжний) дає можливість використання схем прискореного переносу та нарощування розрядності лічильників на базі цих мікросхем.

Установка тригерів лічильника (режим паралельного завантаження) відбувається синхронно і незалежно від значень сигналів на входах *CEP* і *CET*. Процедура завантаження полягає в тому, що при низькому рівні сигналу $\overline{PE}$ режим відліку зупиняється, і за фронтом синхросигналу дані з шини *D* завантажуються в тригери лічильника.

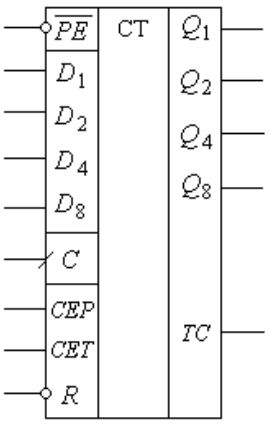


Рис. 6.22

Таблиця 6.9

Режими роботи	Входи						Виходи	
	$\overline{R}$	C	CEP	CET	$\overline{PE}$	D_n	Q_n	TC
Паралельне завантаження	1	↑	x	x	0	0	0	0
						1	1	1
Відлік	1	↑	1	1	1	x	Відлік	1
Зберігання	1	x	0	x	1	x	Q_n	1
	1	x	x	0				
Обнуління	0	x	x	x	x	x	0	0

На рис. 6.23 приводиться схема з’єднання чотирьох мікросхем ИЕ9 у швидкодіючий синхронний 16-розрядний лічильник.

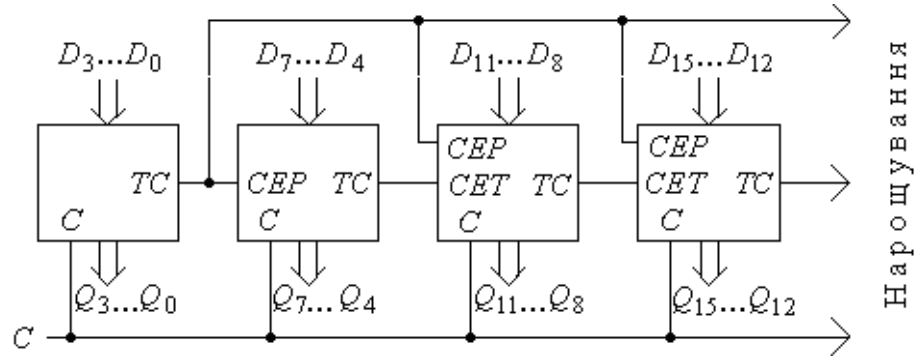


Рис. 6.23

При роботі з мікросхемою слід дотримуватись ряду обмежень. На входах *CEP* і *CET* не допускаються перепади з високого рівня до низького, якщо на вході *C*- низький потенціал. Недопустимо подавати позитивний перепад (0 → 1) на вхід $\overline{PE}$, якщо на *C* маємо сигнал низького рівня, а на *CEP* і *CET* – високого. Сигнали на входах *CEP* і *CET* можна змінювати, якщо на *C* присутній потенціал низького рівня [Шило]. Більш детально особливості використання мікросхем приводяться в довідковій літературі.

6.2.2. Лічильники КМОН - серій

Лічильники КМОН-серій, як і ТТЛ-серій, призначені для підрахунку кількості імпульсів та ділення частоти. Лічильники для підрахунку імпульсів умовно розділяють на *спеціалізовані* – призначені для використання в електронних годинниках, таймерах, пристроях для організації часових затримок, а також *універсальні* – пристрої загального призначення.

Мікросхеми серії 176 (ИЕ1…ИЕ8) були замінені більш пізніми серіями КР561 (CD4000A) та КР1554 (74AC), хоча група мікросхем ИЕ2, ИЕ3, ИЕ4, ИЕ5 знаходить досить широке використання в радіолюбительській практиці. Лічильники останніх серій знаходять досить широке використання в цифровій схемотехніці.

Слід зазначити, що, починаючи з серії 1554, функціональне призначення мікросхеми визначається її номером і є спільним як для ТТЛ, так і для КМОН. Тому, наприклад, лічильники ИЕ6 серій КР1533 і КР1554 є повністю еквівалентними за своїми функціональними можливостями. У мікросхемах основних виробників зарубіжжя така особливість закладалась з самого початку.

Найширше використання у промислових розробках знайшла серія 561 (564), яка мала свої аналоги серед мікросхем CD4000A. У цій серії мікросхеми лічильників виготовлялись як на основі схемотехніки, описаної вище, так і на основі використання лічильників Джонсона (ИЕ8, ИЕ9, ИЕ19), які будуть розглянуті пізніше.

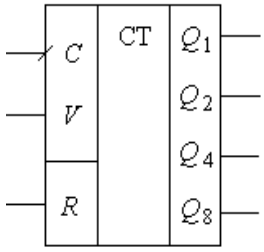


Рис. 6.24

Мікросхема 561ИЕ10 містить два незалежні чотирьохрозрядні двійкові лічильники з паралельним переносом. Умовне позначення одного з них приведене на рис. 6.24, а на рис. 6.25 зображена структурна схема, що пояснює особливості його функціонування.

Лічильник побудований на основі *D*-тригерів, які працюють в режимі *T*-тригера. Тактовий вхід *C* і дозволяючий *V*, об'єднані елементом **АБО**, взаємно інверсні. Це дає можливість взаємозаміняти їх, а також організовувати відлік за фронтом чи спадом тактового імпульсу. Режими роботи лічильника приведені у табл. 6.10.

Таблиця 6.10

Входи			Режими роботи
C	*V*	*R*	
↑	1	0	Режим відліку
0	↓	0	
x	x	1	Обнуління

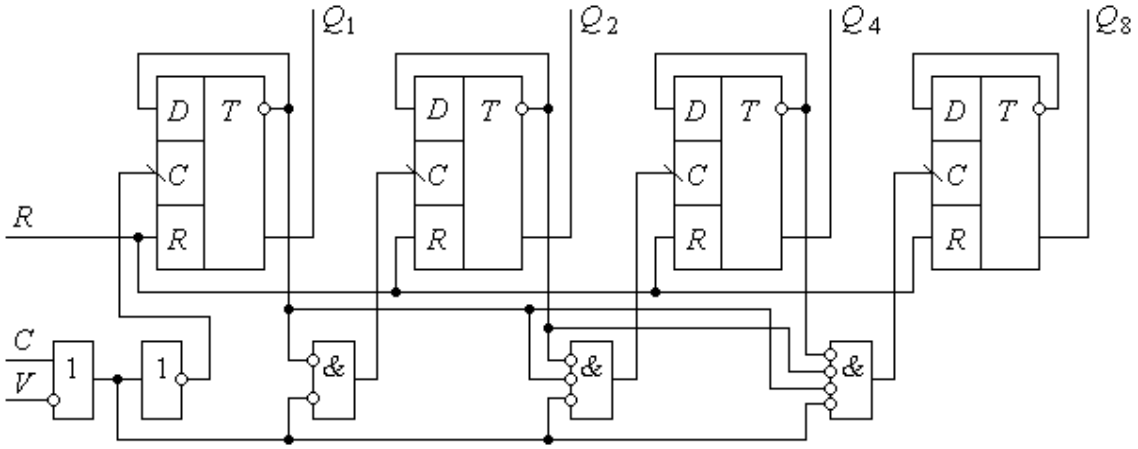


Рис. 6.25

При необхідності нарощування лічильників можна використовувати як послідовне, так і паралельне формування переносу. При послідовному вихід Q_8 попереднього лічильника необхідно з'єднати з входом *V* наступного, а вхід *C* приєднати до потенціалу низького рівня.

Лічильники ИЕ10 мають можливість зменшення модулю M. Але для забезпечення $M < 2^4$ необхідно використовувати зовнішні елементи і враховувати, що лічильник обнуляється сигналом високого рівня на вході R. Обнуління при заданому M можна забезпечити як автоматичним (подібно до рис. 6.7, *а*), так і керованим. На рис. 6.26 приводиться приклад керованого обнуління для $M = 10$ за допомогою зовнішнього сигналу високого рівня M_0. Такий спосіб, як відмічалося вище, дає можливість змінювати коефіцієнти перерахунку лічильника відповідно до вимог. У приведеній схемі маємо $M = 16$ при $M_0 = 0$ і $M = 10$ при $M_0 = 1$.

Для забезпечення паралельного переносу декількох лічильників використовується схема, що приведена на рис. 6.27. Кожен лічильник разом із чотирьохвходовим елементом **І-НІ** створює каскад, який з'єднується з аналогічними на основі паралельного переносу.

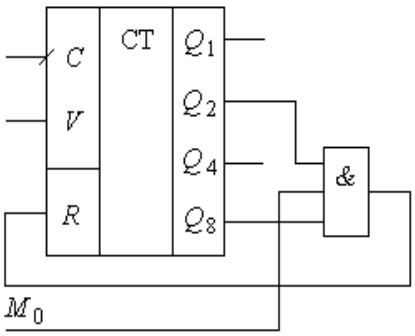


Рис. 6.26

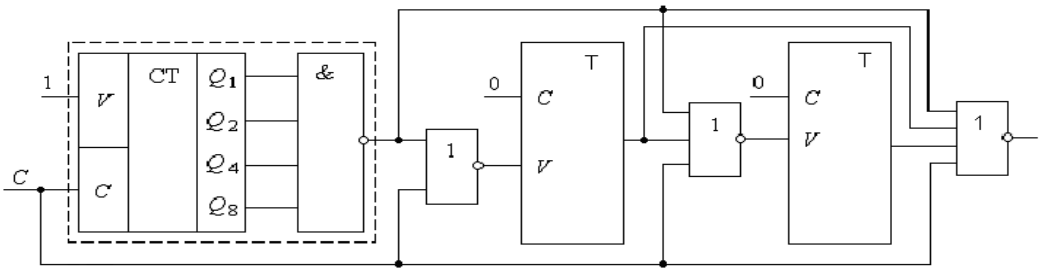


Рис. 6. 27

Мікросхеми 561ИЕ11 і 561ИЕ14 є чотирьохрозрядними реверсивними лічильниками з паралельним переносом і мають багато спільного як з мікросхемами ТТЛ (ИЕ16 і ИЕ17), так і між собою. Різниця між ними полягає лише в тому, що 561ИЕ11 має $M = 16$ і вхід R загального обнуління, а 561ИЕ14 має вхід перемикання $M = 16 / M = 10$, але вхід R відсутній. Умовне позначення мікросхеми ИЕ11 приведене на рис. 6.28, а режими роботи і відповідні їм вхідні сигнали – у табл. 6.11.

Запис попередньої інформації з входів D в усі тригери забезпечується одночасно високим рівнем сигналу на вході WR. Запис виконується до подачі сигналів на вхід C. Вхід $\overline{V}$ – дозволяючий. Високий рівень сигналу на цьому вході забезпечує зупинку режиму відліку, але інформація в лічильнику зберігається. Вихід переносу $\overline{P}$ використовується для нарощування розрядності лічильників. Поточний рівень сигналу на цьому виході – високий. Низький рівень на ньому виникає в режимі відліку на збільшення при $Q_1 = Q_2 = Q_4 = Q_8 = 1$, а в режимі відліку на зменшення – при $Q_1 = Q_2 = Q_4 = Q_8 = 0$.

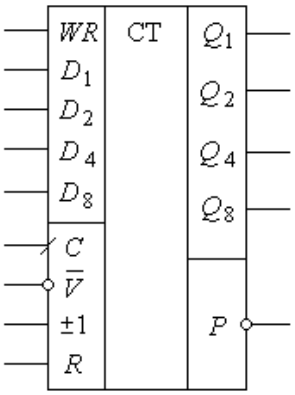


Рис. 6. 28

Таблиця 6.11

Входи				Режими роботи
R	$\overline{V}$	±1	WR	
1	x	x	x	Обнуління
0	1	x	0	Зберігання
0	0	1	0	Відлік на збільшення
0	0	0	0	Відлік на зменшення
0	x	x	1	Попередній запис

Використання виходу $\overline{P}$ дає можливість нарощувати розрядність мікросхеми, використовуючи як послідовний, так і паралельний переноси. Забезпечити циклічний режим роботи з використанням попереднього запису без допоміжної логіки неможливо.

Мікросхема 561ИЕ15 (564ИЕ15) не має аналогів в ТТЛ ІС. Це програмований лічильник з одним виходом, який працює лише в режимі ділення. Коефіцієнт ділення $K_д$ задається відповідними установками рівнів вхідних сигналів і може мати будь-яке значення в межах від **3** до **21327** з кроком **1**. Лічильник може працювати у двох режимах:

- *безперервної дії*, коли на виході отримується послідовність імпульсів частотою $f_{вих} = \frac{f_{вх}}{K_д}$ (з тривалістю імпульсів $t_i = \frac{1}{f_{вх}}$);

- *одноразового відліку*, при якому після подачі на вхід лічильника $K_д$ імпульсів вихідний сигнал змінюється з низького рівня на високий.

Умовне зображення мікросхеми приведене на рис. 6.29.

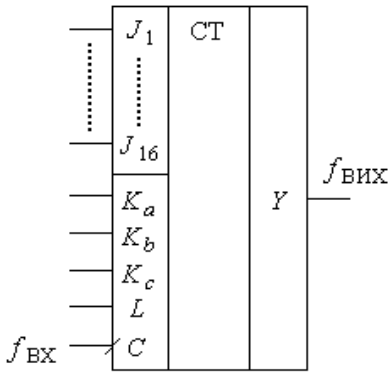


Рис. 6.29

Для визначення рівнів сигналів, що подаються на входи $J_1 \ldots J_{16}$ мікросхеми, використовується формула:

$$K_д = M \cdot (10^3\ P_Т + 10^2\ P_С + 10^1\ P_Д + 10^0\ P_О) + P_З\ , \qquad (6.9)$$

де M – модуль, який може мати наступні значення: **2**, **4**, **5**, **8**, **10**; $P_Т$ – множник тисяч, приймає значення від **0** до **7**; $P_С$, $P_Д$, $P_О$ – множники сотень, десятків, одиниць; $P_З$ – залишок від ділення.

Числа $P_С$, $P_Д$, $P_О$ при використанні формули (6.9) можуть бути задані як у десятковій, так і шістнадцятковій системах числення, але для введення у мікросхему вони повинні попередньо бути перетворені в двійковий код (див. **Розділ 1**).

Входи мікросхеми $J_1 \ldots J_{16}$ використовуються для установки необхідного коефіцієнта ділення. Входи K_a, K_b, K_c використовуються для формування значення модуля M. Вхід L – для вибору одного з зазначених вище режимів (**0** – безперервної дії; **1** – одноразовий відлік).

Входи $J_1 \ldots J_{16}$ використовуються у такій послідовності:

- на входах $J_1 \ldots J_4$ встановлюється двійковий код залишку $P_З$;
- на входах $J_5 \ldots J_8$ – множник одиниць $P_О$;
- на входах $J_9 \ldots J_{12}$ – множник десятків $P_Д$;
- на входах $J_{13} \ldots J_{16}$ – множник сотень $P_С$;
- на входах $J_2 \ldots J_4$ – множник тисяч $P_Т$.

Для кожного двійкового набору молодшого розряду числа відповідає вхід з меншим індексом. Розподіл входів $J_2 \ldots J_4$ між множниками $P_Т$ і $P_З$ регламентується табл. 6.12.

Для установки мікросхеми в початковий стан необхідно забезпечити $K_b = K_c = 0$ протягом не менше трьох періодів вхідних імпульсів. Режими роботи лічильника і відповідні значення сигналів на входах приведені у табл. 6.13.

Діапазони зміни коефіцієнта $P_\text{Т} \ldots P_\text{О}$ приводяться у табл. 6.13. Діапазони зміни коефіцієнтів $P_\text{Т}$ і $P_\text{З}$ залежать від вибору M. Якщо встановити $P_\text{Т} = P_\text{З} = 0$, то для будь-яких значень $P_\text{С}$, $P_\text{Д}$, $P_\text{О}$ при зміні модулю M формується сітка частот з постійним відношенням їх значень: f/N; $1{,}25\,f/N$; $2\,f/N$; $2{,}5\,f/N$; $5\,f/N$. Така особливість може бути корисною при реалізації синтезатора частот, в якому при зміні частоти задаючого генератора необхідно зберігати співвідношення між синтезованими частотами.

Таблиця 6.12

K_a	K_b	K_c	M	$P_\text{О\ max}$	Входи установки $P_\text{О}$	$P_\text{Т\ max}$	Входи установки $P_\text{Т}$	$K_\text{Д}$ при $P_\text{С}$, $P_\text{Д}$, $P_\text{О}$, рівних	
								0÷9	0÷25
1	1	1	2	1	J_1	7	$J_2 \div J_4$	3÷15999	3÷17 331
0	1	1	4	3	J_1, J_2	3	$J_3 \div J_4$	3÷15999	3÷18 663
1	0	1	5	4	$J_1 \div J_3$	1	J_4	3÷9 999	3÷13 329
0	0	1	8	7	$J_1 \div J_3$	1	J_4	3÷15999	3÷21 327
x	1	0	10	9	$J_1 \div J_4$	0	—	3÷9 999	3÷16 659

Таблиця 6.13

L	K_a	K_b	K_c	M	N	Режими роботи
0	1	1	1	2	Var	Безперервної дії
0	0	1	1	4	Var	
0	1	0	1	5	Var	
0	0	0	1	8	Var	
x	0	1	0	10	Var	
0	1	1	0	10	10^4	
1	1	1	1	2	Var	Одноразовий відлік
1	0	1	1	4	Var	
1	1	0	1	5	Var	
1	0	0	1	8	Var	
1	1	1	0	10	Var	
x	x	0	0	—	—	Заборона відліку. Установка

Приклад 6.7. Вибрати рівні логічних сигналів на входах лічильника для забезпечення $K_д = 8479$.

Розв'язання. Виходячи з формули (6.9), можемо стверджувати, що лічильник можна розглядати як послідовне з'єднання двох віднімаючих лічильників, один з яких має коефіцієнт перерахунку, рівний M, а другий визначається значеннями в дужках. Тому, виходячи з можливих комбінацій коефіцієнтів в дужках, приймемо $M = 5$ з ряду його значень. Значення коефіцієнта в дужках:

$$\frac{8479}{5} = 1695 \text{ і залишок } 4 \rightarrow P_З = 4.$$

Представляємо необхідний коефіцієнт ділення у вигляді (6.9):

$$8479 = 5 \times (1000 \times 1 + 100 \times 6 + 10 \times 9 + 1 \times 5) + 4 .$$

Як наслідок, маємо: $P_Т = 1$; $P_С = 6$; $P_Д = 9$; $P_О = 5$.

Знаходимо відповідні значення рівнів сигналів на входах:

$$M = 5_{10} = 101_2 = K_a\, K_b\, K_c \Rightarrow K_a = 1,\quad K_b = 0,\quad K_c = 1 ;$$

$$P_Т = 1 \Rightarrow J_4 = 1 ;$$

$$P_с = 6_{10} = 0110_2 = J_{16}\, J_{15}\, J_{14}\, J_{13} \Rightarrow J_{16} = 0,\quad J_{15} = 1,\quad J_{14} = 1,\quad J_{13} = 0 ;$$

$$P_д = 9_{10} = 1001_2 = J_{12}\, J_{11}\, J_{10}\, J_9 \Rightarrow J_{12} = 1,\quad J_{11} = 0,\quad J_{10} = 0,\quad J_9 = 1 ;$$

$$P_о = 5_{10} = 0101_2 = J_8\, J_7\, J_6\, J_5 \Rightarrow J_8 = 0,\quad J_7 = 1,\quad J_6 = 0,\quad J_5 = 1 ;$$

$$P_з = 4_{10} = 100_2 = J_3\, J_2\, J_1 \Rightarrow J_3 = 1,\quad J_2 = 0,\quad J_1 = 0 .$$

Основний режим роботи лічильника – режим безперервної дії, в якому по закінченню циклу у внутрішні лічильники мікросхеми перезаписуються дані з входів, встановлюючи кожен з них у відповідний коефіцієнт перерахунку.

Режим одноразового відліку характеризується тим, що його обов'язково упереджує режим установки в початкове значення. Після виконання одного циклу на виході Y встановлюється високий рівень сигналу, який зберігається незалежно від наявності вхідних імпульсів. Якщо встановити $L = 0$, то лічильник перейде в циклічний режим з частотою вихідних імпульсів $f_{вих} = f_{вх} / N$. Щоб виконати повторний одиничний запуск, необхідно знову встановити лічильник у початковий стан, після чого повернутися до режиму одноразового відліку.

6.3. Області використання лічильників

Найбільш широке використання лічильників – ділення частоти вхідної імпульсної послідовності і підрахунок кількості імпульсів. Як у ТТЛ, так і в КМОН-серіях ІС є спеціально для цього призначені мікросхеми лічильників. Маються на увазі мікросхема ТТЛ ИЕ8 – програмована мікросхема для ділення частоти вхідних імпульсів – і КМОН ІС ИЕ15.

Розглянемо особливості роботи з мікросхемою ИЕ8 (див. рис. 6.20). Як витікає з (6.8), вихідна частота залежить від кодової комбінації на входах D_i.

Як відмічалося вище, при ряді кодових комбінацій інформаційних сигналів на входах D_i вихідна послідовність імпульсів буде розподілена на періоді нерівномірно, тому вихідна частота $f_{\text{вих}}$ характеризуватиме лише осереднену частоту на ряді циклів. У такому плані розглянута мікросхема забезпечує виконання функцій *перетворювача код-частота* (ПКЧ). При послідовному з'єднанні двох мікросхем загальний коефіцієнт ділення має діапазон від 4096:1 до 4096:4095.

Мікросхема ИЕ8 може використовуватись також при необхідності перетворення в частоту двійково-десяткового коду. Для цього на входи D_0 і D_1 подаються низькі рівні сигналу, а двійково-десятковий код подається на чотири старші розряди. При цьому на виході буде отримана імпульсна послідовність з осередненою частотою $f_{\text{вих}} = \frac{f_{\text{вх}}}{16} \cdot D$, де D – кодова комбінація двійково-десяткового коду.

Якщо виникає необхідність перетворення в частоту багаторозрядного двійково-десяткового коду, то необхідно, щоб синхронний лічильник, на базі якого будується ПКЧ, був також двійково-десятковим з тим, щоб частота вхідних імпульсів, яка подається на слідуючий десятковий розряд, була б у 10 разів нижче, ніж на попередньому. Тому такі ПКЧ слід будувати на основі єдиної логічної керуючої системи, яка повинна порозрядно забезпечити таку відповідність.

Ще більші можливості має мікросхема 564ИЕ15. В її програмуванні закладені такі властивості, які дозволяють легко отримати дискретну сітку частот з постійним відношенням їх значень. Це дозволяє простими програмними засобами створювати на її базі генератори дискретної сітки частот, октавні дільники.

У [7] приводиться приклад синтезу частот звукоряду для електромузичних інструментів. Точні значення частот звукоряду створюють ряд ірраціональних чисел з відношенням двох сусідніх частот, рівним $\sqrt[12]{2}$. На практиці це відношення замінюється ірраціональними дробовими числами за умови, що відхилення, які виникають при цьому, не перевищують 0,2% від необхідного значення частоти. Це відповідає дозволяючій спроможності людського вуха. Звідси витікає, що для отримання 12 значень частот звукоряду коефіцієнти ділення повинні бути не менше 1000. При вибраній частоті задаючого генератора 2 кГц і $K_д$ у межах 1000…2000 можна отримати звукоряд з 12 дискретних частот третьої октави. Для ділення частоти часто використовується мікросхема 561ИЕ15. Частоти для більш низьких октав можна отримати шляхом використання більш простих пристроїв ділення частоти. Так, у складі мікросхем серії 145 є спеціалізовані мікросхеми (К145ИК14 і К145ИК15) для використання в музичних інструментах для октавного ділення частоти.

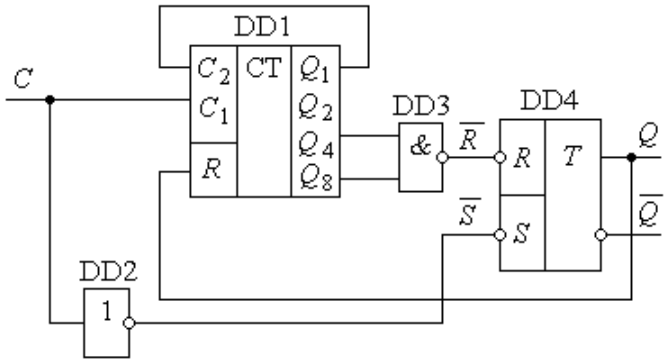
Синтезатори частот, які будуються за структурою систем фазової автопідстройки частоти (ФАПЧ), також містять в собі цифрові лічильники для ділення частоти еталонного генератора.

Лічильники з фіксованим коефіцієнтом ділення частоти стабільних кварцових генераторів використовуються в різноманітних датчиках часу (таймерах), годинниках, календарях (мікросхеми 176ИЕ5, 176ИЕ12). Якщо в лічильнику передбачені пристрої, що за сигналом переповнення завантажують змінювані дані по входах D_i, то відповідно змінюватиметься коефіцієнт перерахунку. Такі лічильники використовуються при цифровому керуванні

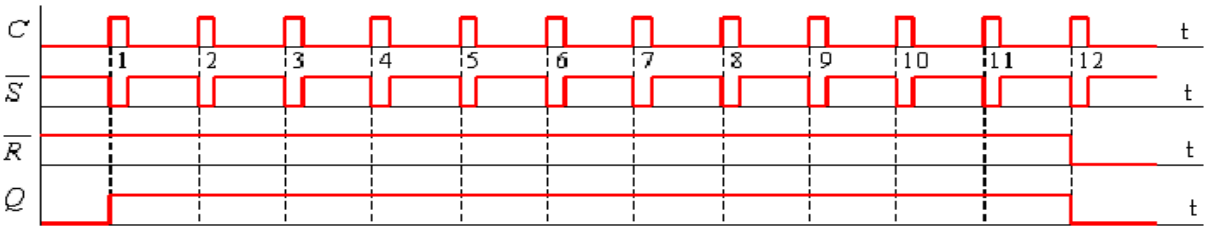
швидкістю крокових двигунів, перетворювачів частоти. Важливий недолік цифрових пристроїв ділення частоти – нерівноінтервальність потоку імпульсів у циклі роботи лічильника – компенсується спеціальними пристроями, призначеними для створення постійного інтервалу.

Для пристроїв розподілення імпульсів, що використовуються в системах промислової автоматики, – наприклад, у системах вибіркового контролю, діагностики, у системах керування кроковими двигунами, напівпровідниковими перетворювачами – перевага часто надається лічильникам, побудованим на основі схем Джонсона, реалізованих, наприклад, на мікросхемах К561ИЕ8, К561ИЕ9, К561ИЕ19. Перевага їх полягає в тому, що інтервал активного стану на кожному з виходів дорівнює періоду вхідних імпульсів, помноженому не на число, яке дорівнює кількості тригерів.

Широке використання знаходять лічильники при створенні таймерів різного функціонального призначення. Схема одного з варіантів таймера-одновібратора з використанням лічильника ТТЛ ИЕ5 приведена на рис. 6.30, *а*. Часові діаграми роботи пристрою приведені на рис. 6.30, *б*.



а)



б)

Рис. 6. 30

Лічильник зв'язком $Q_1 \rightarrow C_2$ встановлений у режим роботи з модулем перерахунку $M = 16$. Перший синхроімпульс на вході C таймера через інвертор DD2 встановлює тригер DD4 у стан $Q = 1$ і змінює вміст лічильника. Наступні десять імпульсів не змінюють стану тригера. При появі дванадцятого імпульсу на виходах Q_4 і Q_8 лічильника DD1 встановляться високі логічні рівні, а на виході ЛЕ DD3 – навпаки, низький рівень сигналу $\overline{R}$, який встановить тригер у стан $Q = 0$. Тому на виході Q *RS*-тригера буде отриманий сигнал високого рівня, тривалість якого задається коефіцієнтом перерахунку лічильника DD1.

Аналогічно створюються одновібратори з програмованою затримкою, які знаходять використання як у пристроях цифрової автоматики, так і в ряді інших пристроїв – наприклад, в пристроях побутової техніки.

Якщо використати вихід Q мікросхеми DD4 як джерело дозволяючого сигналу для того, щоб вибирати з безперервної послідовності визначену кількість імпульсів, то отримаємо *дозатор* – пристрій, який можна використовувати, наприклад, для визначення розміру пакетів при послідовних форматах обміну інформацією. Часто в якості інтервалів часу, формованих дозатором, визначається інтервал, необхідний для передачі одного байту інформації.

Інший варіант схеми дозатора приводиться на рис. 6.31.

Сигналом R лічильник обнуляється, потенціал на виході переносу P приймає низький рівень, внаслідок чого імпульсна послідовність C через ЛЕ DD1 передаватиметься на вихід **Вих.** і на вхід лічильника. При переповненні лічильника потенціал виходу P зміниться на високий і, як результат, призведе до заборони подальшої передачі вхідних імпульсів на вихід. В якості лічильника DD2 можна взяти мікросхему для ділення частоти з програмованим коефіцієнтом ділення. Якщо в такому дозаторі використовується двійковий лічильник і підрахований ним пакет

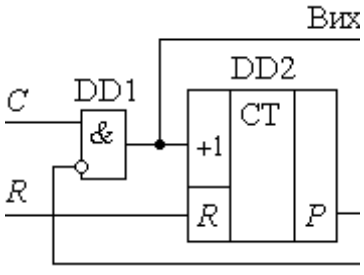


Рис. 6.31

імпульсів подати на вхід іншого лічильника – десяткового, то після закінчення циклу підрахунку число, що було введене в дозатор у якості модулю перерахунку, буде перетворене у двійково-десятковий код, який можна зняти паралельно з виходів десяткового лічильника. Якщо ж двійковий і десятковий лічильники поміняти місцями, то буде отриманий зворотний перетворювач двійково-десяткового коду в двійковий.

Вимірювання ряду електричних величин зводиться до вимірювання інтервалів часу: період гармонічного сигналу, інтервал часу одного оберту вала, зсув гармонічних сигналів, інтервал часу між посилкою та прийомом імпульсу радіолокатора, інтервал часу розряду конденсатора в аналого-цифрових перетворювачах з подвійним інтегруванням і т.п.

На рис. 6.32 приведена функціональна схема цифрового пристрою для вимірювання частоти.

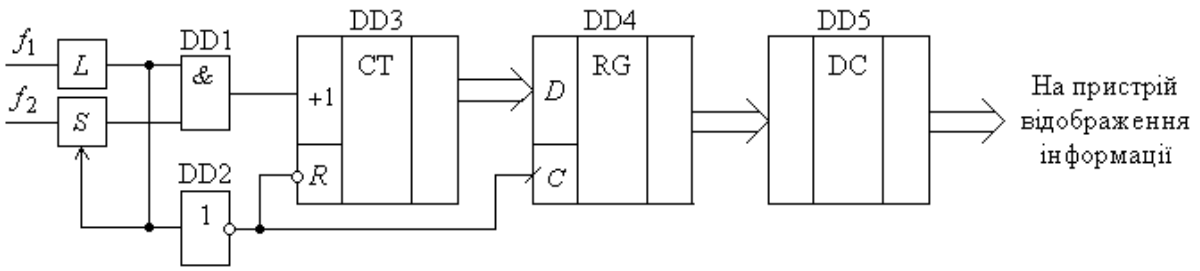


Рис. 6.32

Пристрій L формує інтервал часу вимірювання T. Це може бути один період вимірюваного сигналу, його частина або значно більший інтервал часу. Наприклад, для вимірювання частоти мережі 220/380 В можна взяти один період. Вибраний інтервал часу пристроєм L формується у вигляді високого рівня сигналу. Високочастотна послідовність імпульсів f_2, частота яких задається допустимою помилкою вимірювання, подається через пристрій синхронізації S, принципи побудови яких описані в попередніх розділах, і через елемент **2І** DD1 – на вхід лічильника. Кількість імпульсів, яка підрахована лічильником за інтервал часу T, в кінці інтервалу перезаписується в

запам’ятовуючий пристрій *RG* у вигляді двійкового коду. Отриманий код дешифратором *DC* дешифрується і передається на пристрій відображення інформації. Регенерація зображення у пристрої, побудованому за приведеною схемою, може відбуватись досить часто, що необхідно враховувати при виборі інтервалу *T*.

На рис. 6.33 приводиться варіант використання лічильників у пристроях віднімання частоти. Подібні пристрої широко використовуються у вимірювальній техніці; у приладах, призначених для роботи з частотними датчиками; у системах частотного керування електроприводами та ін.

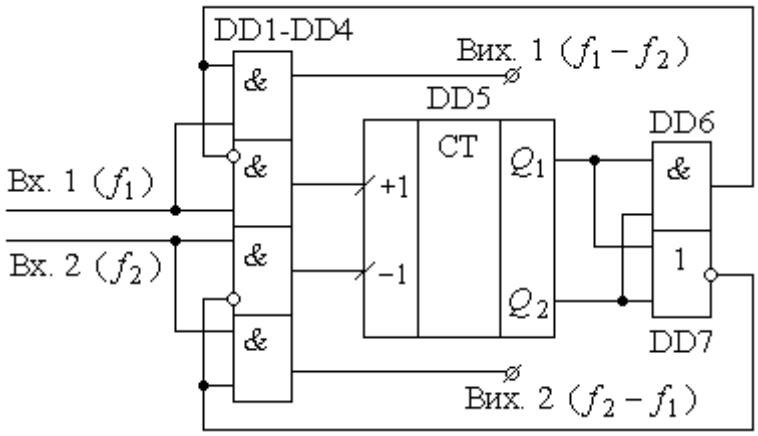


Рис. 6.33

Імпульси з частотами f_1 та f_2 подаються, відповідно, на входи **Вх. 1** і **Вх. 2**. Логічна схема на елементах DD1…DD4 забезпечує дозовану подачу імпульсів на додаючий і віднімаючий входи лічильника DD5. У найпростішому випадку це може бути двохрозрядний лічильник з виходами розрядів Q_1 та Q_2. Коли лічильник DD5 імпульсами частотою f_1 переводиться у стан $Q_2\,Q_1 = 11$, високий рівень сигналу з елемента **2I** DD6 блокує подальшу подачу імпульсів на вхід **+1** і відкриває шлях до подачі їх на вихід **Вих. 1** ($f_1 - f_2$). Імпульси частотою f_2, що поступають на віднімаючий вхід лічильника DD5, змінюють його вихідний стан. Якщо його виходи приймуть стан $Q_2\,Q_1 = 00$, то аналогічно через елемент DD7 буде заблокований віднімаючий вхід лічильника, і імпульси частотою f_2 подаються на вихід **Вих. 2** ($f_2 - f_1$). Якщо $f_1 = f_2$, то код,

відображаючий кількість поданих в лічильник імпульсів, коливатиметься біля одного рівня: імпульс частотою f_1 збільшить його на **1**, а імпульс частотою f_2 – відповідно, встановить попереднє значення.

У розглянутій схемі реверсивний лічильник виконує функцію низькочастотного фільтру інтегрального типу для частотних сигналів. Якщо різниця частот коливається біля нульового рівня, то ці коливання осереднюються лічильником і не проходять на вихід. Дійсно, якщо знак різниці зміниться, то потрібен деякий час, перш ніж лічильник перейде в новий крайній стан, і тільки після цього імпульси можуть перейти на вихід. Ємність лічильника можна збільшити і задати значно більший діапазон варіації кількості імпульсів між двома крайніми значеннями. Таке збільшення еквівалентно збільшенню постійної часу інтегратора. Робота описаної схеми у відповідності до приведеного опису відбувається при неспівпаданні імпульсів з частотами f_1 та f_2 у часі, що може бути забезпечено шляхом використання двофазної синхронізації (див. **Розділ 4**). Приведений пристрій може використовуватись в якості фазового детектора в схемах фазової автопідстройки частоти, причому від пристроїв іншого типу відрізняється однозначністю фазової характеристики.

На рис. 6.34 приводиться приклад спрощеної структурної схеми керування частотою обертання вала двигуна постійного струму.

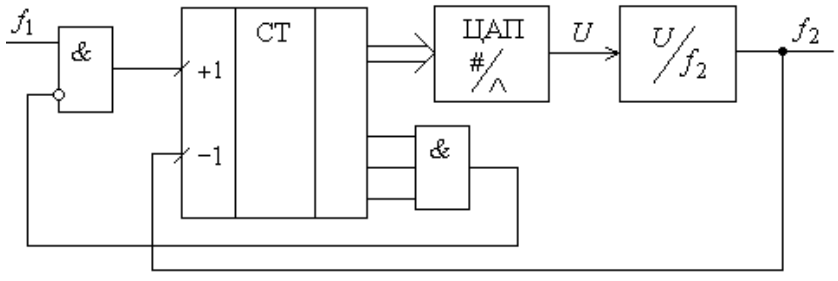


Рис. 6.34

Лічильник у цій схемі виконує функцію цифрового інтегратора-суматора з обмеженням. Він є повним еквівалентом аналогового інтегратора на операційному підсилювачі з обмеженням напруги на конденсаторі за

допомогою стабілітронів.

Двигун постійного струму умовно зображений у вигляді перетворювача керуючого аналогового сигналу в частоту f_2, еквівалентну частоті обертання ротора. При подачі керуючого частотного сигналу f_1 вміст лічильника зростає, зростає його вихідний двійковий код, який за допомогою цифро-аналогового перетворювача перетворюється в напругу керування двигуном. Зростання напруги приводить до зростання частоти f_2 сигналу зворотного зв'язку, який подається на віднімаючий вхід лічильника. Зростання частоти обертання ротора двигуна приводить до зростання кількості імпульсів, що подаються на вхід **–1.** У сталому режимі частоти f_1 та f_2 вирівняються і створюють близьке до стабільного значення двійкового коду на виході лічильника.

У мікропроцесорній техніці лічильники використовуються, наприклад, у блоках адресації для організації звернень до пристроїв пам'яті та зовнішніх пристроїв. Просте інкрементування лічильника забезпечує покрокове звернення за адресами з командами та даними, які потім за допомогою керуючих сигналів *RD* і *WR* зчитуються в процесор та перезаписуються повторно в пам'ять. При необхідності переходів іншого типу в лічильник завантажується двійковий код початкової адреси іншої підпрограми, яка потім виконується в покроковому режимі.

Приблизно такі ж функції лічильник виконує в мікропрограмних автоматах, в яких кожне значення вихідного двійкового коду лічильника декодується в групу сигналів, які реально представляють собою мікрокоманду.

Сумісне використання перетворювачів код-частота і пристрою віднімання частот дозволяє будувати арифметичні віднімаючі пристрої. У [Ланцов] описаний арифметичний пристрій, який виконує операції додавання, віднімання, множення та ділення двійкових чисел, але, на жаль, досить повільно. В [Зельд] описаний пристрій, що виконує операції ділення за допомогою віднімаючих лічильників.

Дещо інші принципи закладені в арифметичні пристрої, що описані в

[Оберман]. Вони реалізують функції нормуючих перемножувачів, а також різноманітні функціональні перетворення – наприклад, $\log_2 x$, $\sqrt[m]{x}$, $1/x$, $\sin x$, $\cos x$. У [Ланц, Оберман] приводяться приклади реалізації рекурсивних лічильників, лічильників зі спеціальними вихідними кодами – наприклад, такими, що відповідають числам Фібоначі.

Розглянемо ще одне використання лічильників у пристроях, які в залежності від характеру використання називаються *інтерполяторами*, *цифро-частотними перемножувачами*, *цифро-частотними інтеграторами* [Потемк.].

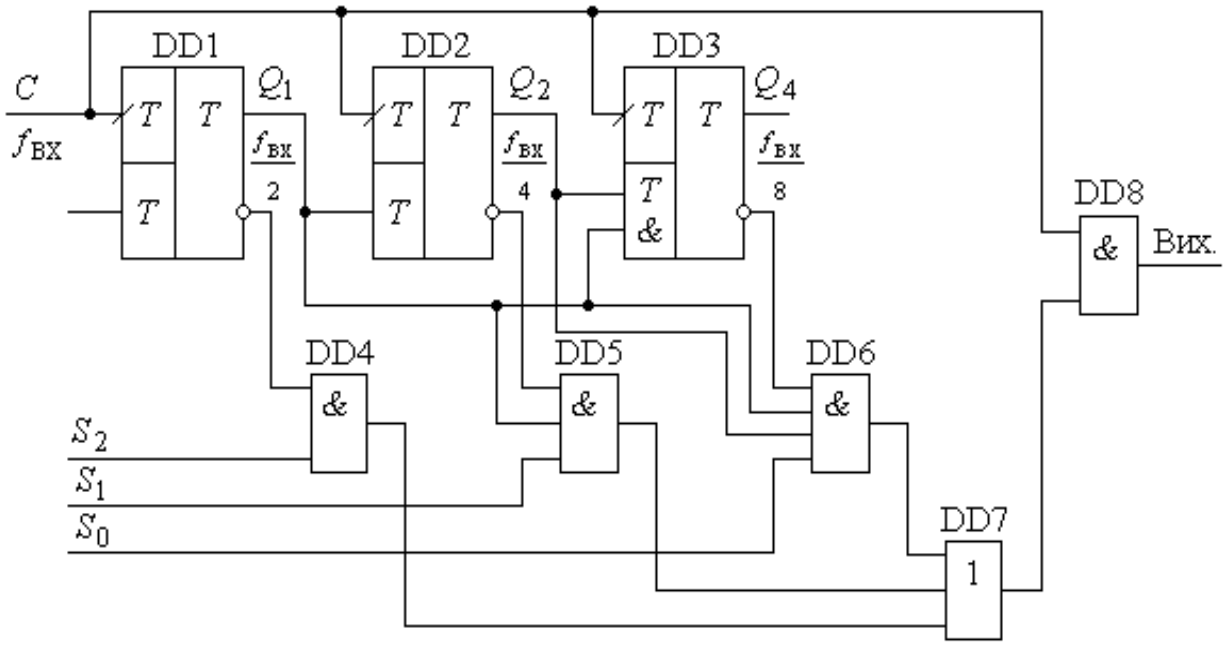


Рис. 6.35

Основою схеми пристрою є синхронний лічильник з паралельним перенесенням (рис. 6.35), виготовлений на тригерах DD1…DD3. Лічильник має модуль $M = 8$. Частота імпульсів на прямому виході кожного тригера зменшується вдвічі порівняно із вхідною. При виконаній схемі з'єднань високі рівні сигналів на виходах тригерів не співпадають у часі. Тому у вихідному сигналі DD7 кожна частотна складова буде представлена самостійно і незалежно від наявності чи відсутності інших (рис. 6.36).

Часові діаграми сигналів на виходах DD4, DD5, DD6 зображені при умові,

що сигнали на їх входах – відповідно, S_2, S_1, S_0 – мають високі рівні, тобто відповідають кодовій комбінації **111**. З часової діаграми витікає, що кількість імпульсів, які передаються на вихід пристрою на одному циклі роботи, однозначно визначається значенням двійкового коду на керуючих входах S_2, S_1, S_0.

Кількість імпульсів, відповідну значенню керуючого двійкового коду, забезпечує на своєму виході і дозатор, але особливість інтерполятора полягає в тому, що вихідні імпульси передаються не в пакетній формі, а приблизно рівномірно розподілені по інтервалу часового циклу.

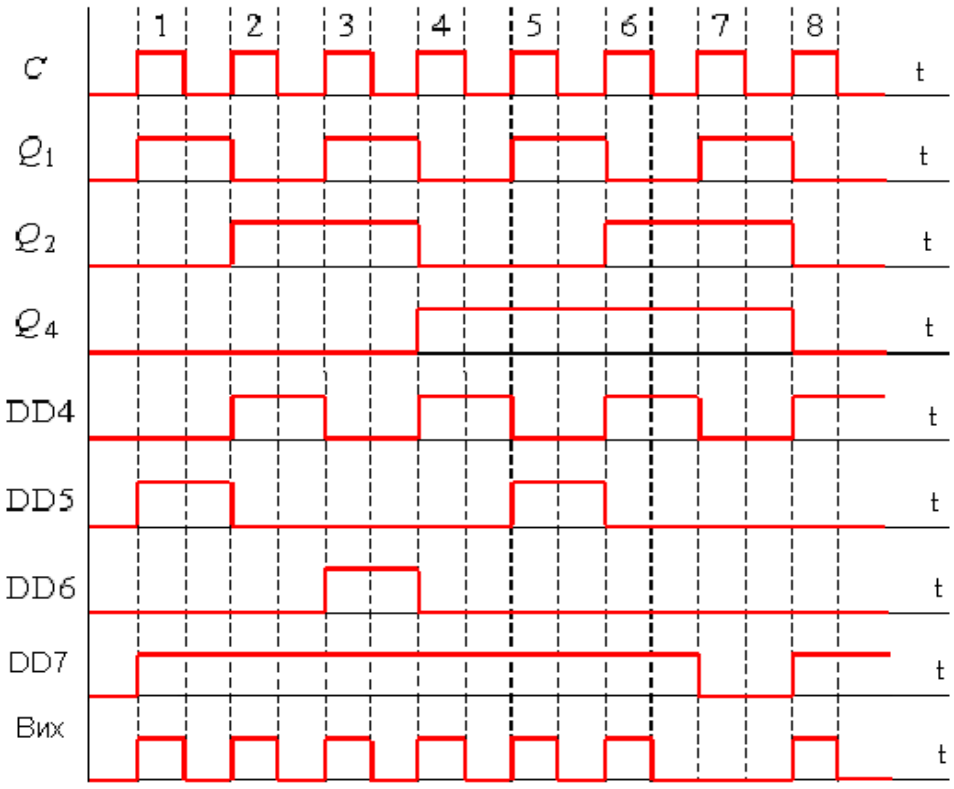


Рис. 6.36

Інтерполятори використовують для керування кроковими двигунами, коли їх швидкість задається двійковим кодом $S_2 S_1 S_0$. Вони можуть використовуватись як пристрій для перемноження сигналу, що задається частотою вхідних імпульсів $f_{вх}$, на сигнал, що задається керуючим двійковим кодом $S_2 S_1 S_0$. При цьому результат відображається частотою

вихідних імпульсів.

Якщо забезпечити такий режим, при якому на вхід S подаватиметься у двійковому коді деяка функція $F(x)$, а на вхід C – приріст аргументу цієї функції Δx, то кількість імпульсів на виході буде пропорційною інтегралу $\int F(x)\,dx$.

Окрім зазначених областей використання, лічильники знаходять широке застосування в інших електронних пристроях. Відоме широке використання лічильників в аналого-цифрових перетворювачах різних типів [під ред. Якубовського], для реалізації нелінійних часових функцій [Зельд], для керування запам'ятовуючими пристроями і т.п.

6.4. Скінченні автомати на основі лічильників

На початку розділу відмічалося, що лічильники можуть розглядатись як скінченні автомати з циклічним режимом роботи. Але іноді виникає необхідність побудови автомата, що працює в майже циклічному режимі. У таких випадках для побудови автоматів корисно використати готові мікросхеми лічильників з базовим циклічним режимом роботи і доповнити комбінаційною логікою для забезпечення нециклічних переходів.

На рис. 6.37, як приклад, приводиться діаграма станів автомата з майже циклічним режимом роботи. В такому випадку необхідно вибрати мікросхему лічильника і за допомогою логічних елементів забезпечити необхідні переходи.

Функціональна схема автомата, що відповідає приведеній діаграмі станів, показана на рис. 6.38.

Вхід автомата R фактично призначений для примусового встановлення автомата в стан **0**, а сигнал S – аналогічно, для забезпечення примусового встановлення автомата в стан **6.**

Розглянемо особливості проектування таких автоматів на прикладі лічильника з $M = 8$, який за сигналом керування G повинен працювати у двійковому коді ($G = 0$) або в режимі відліку відповідно до коду Грея ($G = 1$).

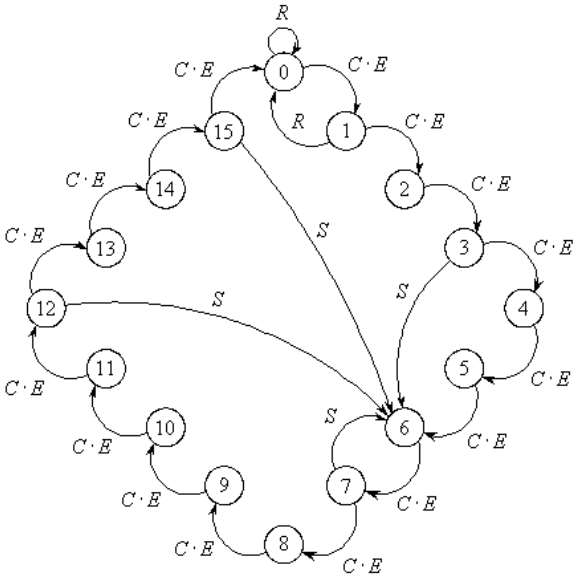


Рис. 6.37

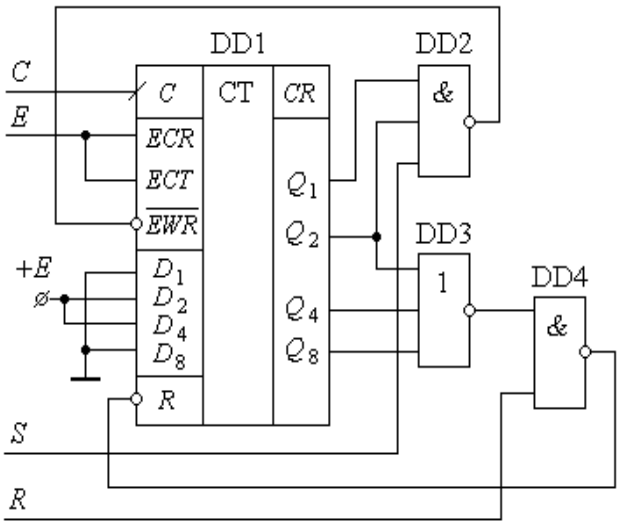


Рис. 6.38

Відповідність між значеннями двійкового коду і коду Грея була приведена у **Розділі 1** (табл. 1.1), а граф-схема автомата матиме вигляд, зображений на рис. 6.39.

Виходячи з граф-схеми, можемо визначитись, що для побудови такого автомата необхідно використати двійковий лічильник з попередньою асинхронною установкою.

Оскільки зв'язок між розрядами бінарного коду b_0, b_1, b_2 і коду Грея g_0, g_1, g_2 встановлюється залежностями:

$$g_0 = b_0\,\overline{b_1} + \overline{b_0}\,b_1 = b_0 \oplus b_1\,;$$
$$g_1 = b_2\,\overline{b_1} + \overline{b_2}\,b_1 = b_2 \oplus b_1\,;$$

то матимемо функціональну схему автомата (рис. 6.40).

Із функціональної схеми витікає, що для побудови автомата для $M = 16$ досить зняти подачу сигналу на вхід R і додати, аналогічно попередньому, ще один логічний елемент **"ВИКЛ. АБО"**.

Використання лічильників при побудові цифрових автоматів дозволяє значно спрощувати апаратну реалізацію, але вимагає від інженерів-розробників досить високої кваліфікації.

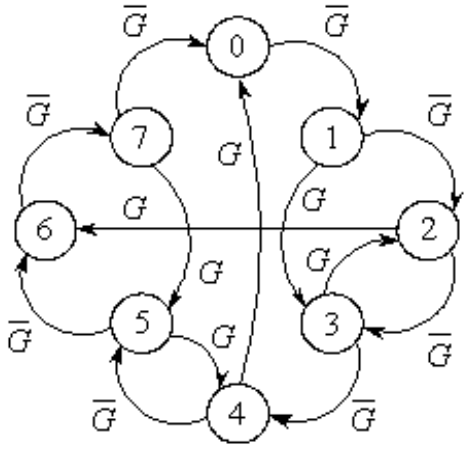


Рис. 6.39

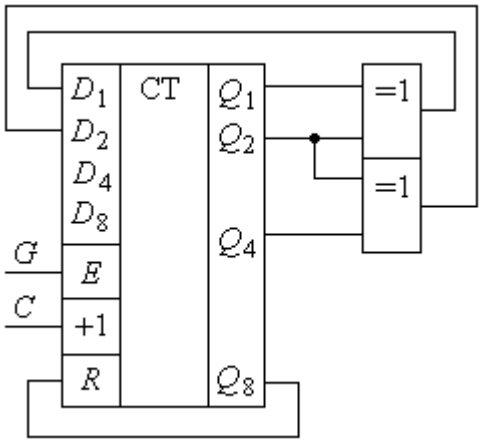


Рис. 6.40

КОНТРОЛЬНІ ПИТАННЯ

1. Дайте визначення терміну *“лічильник”*.

2. Які параметри характеризують лічильник як цифровий пристрій? Назвіть їх і дайте фізичну інтерпретацію.

3. У чому полягає різниця між *синхронними* і *асинхронними* лічильниками?

4. Поясніть суть операцій *інкрементування* та *декрементування* вмісту лічильника.

5. Чим обмежується швидкодія лічильників з послідовним переносом?

6. Які способи використовуються для побудови лічильників з довільним модулем перерахунку?

7. Які недоліки має лічильник з попередньою установкою модуля перерахунку?

8. У чому полягає перевага синхронних лічильників перед асинхронними?

9. Перерахуйте та обґрунтуйте недоліки двійкового лічильника з паралельним переносом.

10. Якими шляхами розв’язується задача нарощування розрядності синхронних лічильників з паралельним переносом?

11. У чому полягає недолік асинхронних лічильників при їх використанні з дешифраторами та семисегментними індикаторами для відображення інформації?

12. Як на основі мікросхеми ИЕ8 розробити перетворювач чотирьохрозрядного двійкового коду в частоту вихідних імпульсів; восьмирозрядного двійкового коду в частоту вихідних імпульсів?

13. Поясність особливості нарощування розрядності лічильників ИЕ9 та ИЕ10.

14. Чим мікросхема 564ИЕ15 принципово відрізняється від інших типів мікросхем лічильників?

15. Приведіть порівняльну характеристику переваг та недоліків двійкових лічильників і лічильників Джонсона.

16. Поясніть, на якому принципі будуються цифрові одновібратори та різноманітні таймери.

17. На якому принципі будуються дозатори? Що це за пристрої і де вони можуть використовуватися?

18. Поясніть, як можна використати дозатор для перетворення двійкового коду в двійково-десятковий та навпаки.

19. Поясніть принцип роботи цифрових частотних інтеграторів.

20. Обґрунтуйте можливість використання двійкових лічильників у блоках адресації мікропроцесорів.

21. Поясніть принцип роботи та призначення інтерполяторів. Які функції вони можуть виконувати?

ВПРАВИ І ЗАВДАННЯ

1. Скільки тригерів необхідно для побудови двійкового послідовного асинхронного лічильника з $M = 1024$?

2. На вхід двійкового асинхронного додаючого лічильника з $M = 16$ подано $N = 87$ імпульсів. Який код встановиться на виходах Q лічильника?

3. Повторити вправу 2, але для віднімаючого лічильника з початковою установкою **0000**.

4. Розробити функціональну схему додаючого двійкового лічильника з $M = 4$, в якого стан виходів визначається кількістю записаних в ньому одиниць.

5. Розробити функціональну схему лічильника до прикладу 6.2.

6. Використовуючи мікросхему ТТЛ ИЕ5, розробити схему лічильника з коефіцієнтом перерахунку **10**.

7. Використовуючи мікросхему ИЕ5 і мультиплексор КП7 ТТЛ, розробити схему для перетворення однобайтового паралельного формату даних у послідовний з паузою в 2 біти між словами.

8. Спроектувати синхронні лічильники за модулем **12** на основі довільно взятої логіки та наступних елементів: *а*) *T*-тригерів; *б*) *RS*-тригерів; *в*) *JK*-тригерів; *г*) *D*-тригерів. Розробити логічну схему для декодування результату відліку.

9. Використовуючи *JK*-тригери та допоміжну логіку, розробити циклічний генератор послідовності, що задана у табл. 6.14.

Таблиця 6.14

N	Q_2	Q_1	Q_0
1	0	0	1
2	1	0	0
3	0	1	0
4	1	0	1
5	1	1	0
6	0	1	1

10. На рис. 6.42 приведена схема цифрового пристрою на основі лічильника ИЕ17. Проаналізувати і пояснити роботу пристрою.

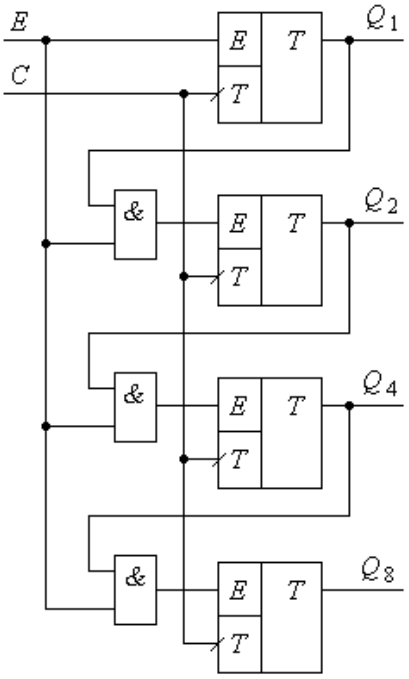


Рис. 6.41

11. Лічильник ИЕ18 має зовнішні з'єднання відповідно до схеми, приведеної на рис. 6.41. Проаналізувати особливість роботи пристрою, якщо у початковому стані маємо $Q_8\ Q_4\ Q_2\ Q_1 = 0000$. Привести часові діаграми для одного циклу роботи лічильника.

12. На рис. 6.43 приведена схема синхронного двійкового лічильника з логікою послідовного зсуву. Привести формулу для обчислення максимальної частоти синхросигналів. У формулі повинні бути присутніми час затримки *T*-тригера від *T*-входу до *Q*-виходу, а також час між моментом подачі сигналу *E* та моментом початку фронту синхросигналу. Як називаються вказані інтервали часу?

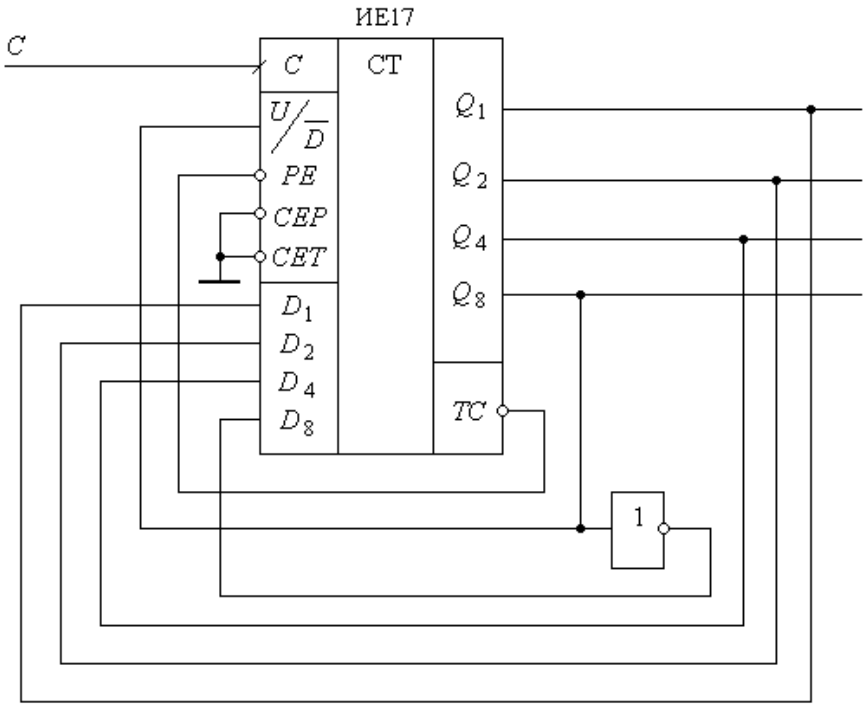


Рис. 6.422

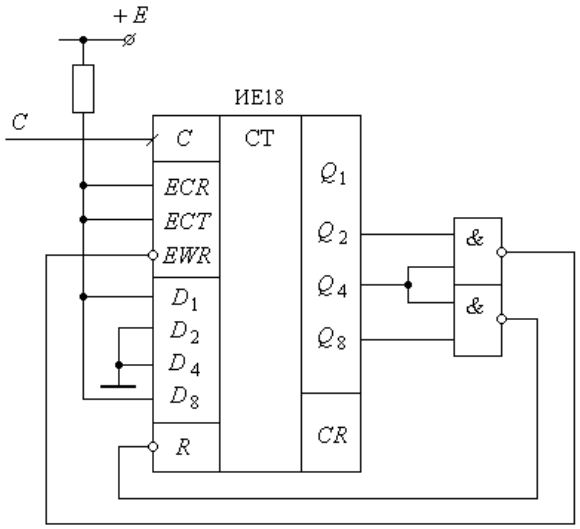


Рис. 6.43

13. Повторити вправу попередньої задачі для синхронного паралельного двійкового лічильника і порівняйте результати з попередніми.

14. Повторити вправу для послідовного лічильника з модулем M.

15. Використовуючи лічильник ИЕ18, спроектуйте лічильник із $M = 11$ з відліковою послідовністю: 3, 4, 5, …, 12, 13, 3, 4, …для роботи в циклічному режимі.

16. Використовуючи мікросхему ИЕ18, розробити схему лічильника з $M = 129$. Пояснити її роботу.

17. Використовуючи відомі мікросхеми, розробити схему лічильника з бінарним чотирьохрозрядним керуючим входом $N_3\ N_2\ N_1\ N_0$ і одним виходом Y. Лічильник повинен видавати в циклі із 16 вхідних синхроімпульсів таку кількість вихідних, яка дорівнює десятковому значенню вхідного двійкового коду.

18. Використовуючи мікросхему ИЕ17 і, якщо необхідно, допоміжну логіку, спроектувати лічильник з M = 16 із наступною відліковою послідовністю: 7, 6, 5, 4, 3, 2, 1, 0, 8, 9, 10, 11, 12, 13, 14, 15, 7, 6, …

19. Розробити 8-бітний самокоригований кільцевий лічильник, в якому циркулює лише один нуль і сім одиниць підряд.

20. Використовуючи типовий двійковий лічильник з $M = 16$ і логічні елементи **“ВИКЛ. АБО”**, розробити схему лічильника, вихідні сигнали якого змінюватимуться відповідно до коду Грея.

21. Побудувати часові діаграми у контрольних точках (C, Q_1, Q_2) схеми лічильника з $M = 3$ (рис. 6.44).

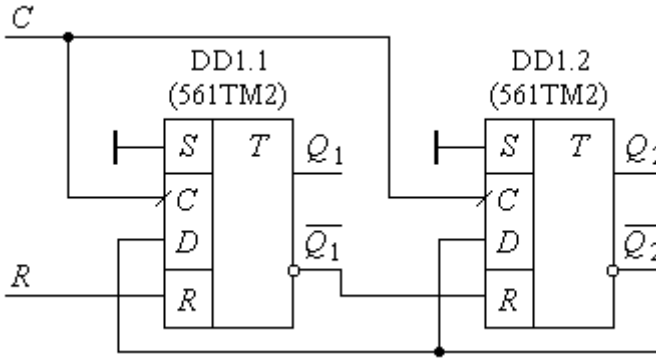


Рис. 6.44

22. Використовуючи мікросхему 561(564)ИЕ11 та допоміжну логіку, розробіть схему ділення частоти вхідних імпульсів на **60**.

23. Умова задачі 15. Розробити двійковий лічильник з $M = 161$.

24. Використовуючи мікросхему 564ИЕ15, розробити схему пристрою для ділення вхідної частоти в **12 122** рази.

25. Розробити схему керування дешифратором-демультиплексором для забезпечення перетворення однобайтового слова послідовного формату в паралельний формат. Між інформаційними словами послідовного формату повинна бути встановлена пауза в два такти.

26. Розробити функціональну схему пристрою для контролю опору ізоляції електричної мережі у 20 точках. Датчик опору ізоляції забезпечує високий рівень логічного сигналу, якщо величина опору знизиться до величини, меншої мінімально допустимої. Пристрій повинен видати інформацію у вигляді номеру аварійної точки і подати сигнал оповіщення.

27. У практиці кабельного електромонтажу виникають задачі маркування жил провідників кабелю, кінці якого рознесені у різні віддалені приміщення. Пропонується розробити електронний пристрій і технологію, які б дозволили автоматизувати задачі маркування.

Підказка: Використайте результати попередньої задачі.

28. Розробити функціональну схему приладу для вимірювання частоти мережі з відображенням двох значущих цифр на семисегментних індикаторах.

Підказка: Формується меандр періоду або півперіоду, що заповнюється високочастотними імпульсами, які потім зчитуються лічильниками.

29. Розробити функціональну схему для вимірювання ємності конденсатора частотним способом.

30. Розробити функціональну схему годинника з відображенням годин та хвилин на семисегментних індикаторах.

31. Виконати задачу 31, але з відображенням годин за допомогою світлодіодів, розміщених на круглому циферблаті.

32. Застосувавши один з двійкових реверсивних лічильників, розробити принципову схему пристрою, який би при подачі послідовності синхросигналів міг автоматично забезпечувати реверс при повному заповненні (обнулінні) лічильника. Використати лічильник з $M = 16$.

Підказка: Імпульси на входи інкрементування та декрементування необхідно подавати через дозволяючі елементи **2І**, сигнали на інших входах яких задаються *RS*-тригером. Тригер, у свою чергу, може керуватися від дешифратора **4:16**, що перетворює двійковий код лічильника в сигнал **“1 з 16”**.

33. Розробити функціональну схему віднімаючого таймера-лічильника, в який можна задавати в двійково-десятковому коді час в інтервалі 1 хвилина з точністю до 1 секунди і який при обнулінні видавав би на своєму виході сигнал високого рівня тривалістю 1 секунда.

34. Розробити функціональну схему дозатора для формування слів ємністю 10 біт з паузою між ними в 1 такт.

Розділ 7

РЕГІСТРИ

7.1. Загальні поняття про регістри

Регістрами називають цифрові пристрої, призначені для тимчасового зберігання та перетворення інформації, у вигляді багаторозрядних двійкових чисел. Основою будь-якого регістра є елемент пам'яті – *тригер*. Кількість тригерів, розміщених паралельно, або з'єднаних послідовно, визначає *розрядність* регістрів. Але, на відміну від лічильників, до розрядів регістрів поняття *"ваговий коефіцієнт"* не застосовується, оскільки кожен з них незалежний.

У регістрах використовуються *RS*-, *D*-, *JK*-тригери. Для забезпечення керування записом інформації у тригери та її зчитуванням використовуються комбінаційні пристрої, які закладають алгоритми керування регістрами.

Регістри можуть класифікуватися за різними ознаками, але основною є спосіб введення та виведення інформації. Виходячи з цього, вони розподіляються на дві групи: *паралельні* (або *регістри пам'яті*) та *послідовні* (або *регістри зсуву*). У свою чергу, послідовні регістри можуть забезпечувати *послідовний*, *паралельний* та *комбінований* способи введення та виведення інформації.

Записана у тригери інформація може зчитуватись у прямому коді, оберненому або одночасно в прямому та оберненому кодах.

Регістри зсуву можуть бути *однонаправленими*, тобто забезпечувати зсув лише в одному напрямку, або *реверсивними* (*двонаправленими*).

У мікропроцесорній техніці широко використовуються регістри, в яких вхідні та вихідні лінії даних об'єднані в одну групу (порт даних). У залежності від керуючого сигналу, такий порт може налаштовуватись або на введення інформації, або на її вивід.

7.2. Регістри пам’яті

Регістри памяті є порівняно простими структурами, призначеними для запису, тимчасового зберігання та передачі невеликих об’ємів інформації, представленої в цифровому вигляді.

Елементами пам’яті в них виступають здебільшого *D*-тригери, тому регістри подібного типу можуть виготовлятись з використанням мікросхем, що містять набори синхронних тригерів, – наприклад, мікросхеми КР1533ТМ2 (SN74ALS74A), ТМ7, ТМ8, ТМ9. Керування режимами запису-зчитування може бути забезпечено вхідною та вихідною логікою, побудованою, наприклад, з використанням ЛЕ **2І** (ЛИ1 (ALS08)).

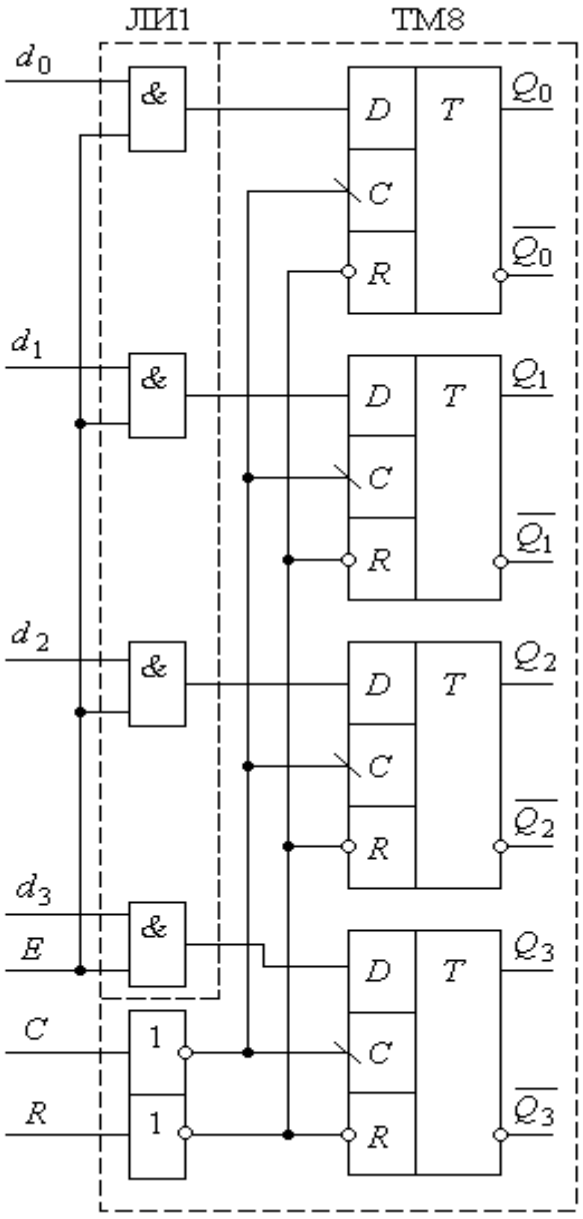


Рис. 7.1

На рис. 7.1 приводиться приклад реалізації функціональної схеми регістра пам’яті з дозволяючим входом запису *E*, прямими та інверсними виходами. Запис інформації в *D*-тригери може виконуватись при високому рівні сигналу на вході *E* за фронтом синхросигналу *C*. Зчитування інформації в прямому або оберненому кодах виконується безпосередньо з виходів тригерів Q_i та $\overline{Q_i}$. Для забезпечення керування процесом зчитування може бути використана та ж мікросхема, або створений режим мультиплексування виходів тригера на один вихід регістра.

При використанні такого пристрою у структурах з великою кількістю регістрів вхід E може використовуватись для адресного звернення до одного з них через відповідні дешифратори адреси.

Прикладом такої структури є схема, приведена на рис. 7.2.

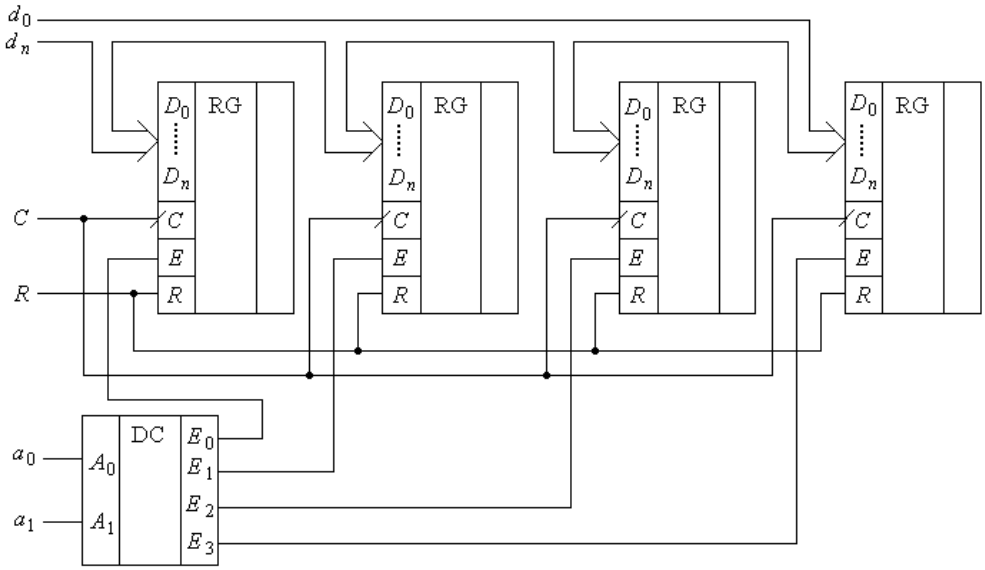


Рис. 7.2

За допомогою адресних сигналів $a_0\, a_1$ вибирається один із регістрів, на який подається високий рівень дозволяючого сигналу. Внаслідок цього дані з шини $d_0 \ldots d_n$ за фронтом синхросигналу C будуть записані у вибраний регістр. Така структура дозволяє демультиплексувати потік даних по декількох окремих шинах.

Подібним шляхом може бути розв'язана зворотна задача, коли необхідно з різних шин подавати інформацію на одну – режим мультиплексування потоків даних, їх ущільнення. Читачам пропонується самостійно розробити таку схему.

Прикладом регістра пам'яті є мікросхема ИР5. Це чотирьохрозрядний паралельний регістр даних, умовне зображення якого приведене на рис. 7.3.

Він має по два входи D_{iA} та D_{iB} для одного запам'ятовуючого елемента. Вибір входу забезпечується рівнем сигналу, що подається на вхід S, а запис – зрізом синхросигналу. Враховуючи наявність логіки на входах та виходах тригерів, режими роботи регістрів описуються таблицями станів.

Приклад таблиці станів для регістра ИР5 приводиться у табл. 7.1., в якій відображені всі режими роботи регістра, і для кожного з них вказана однозначна відповідність між рівнями логічних сигналів на його вході та виході.

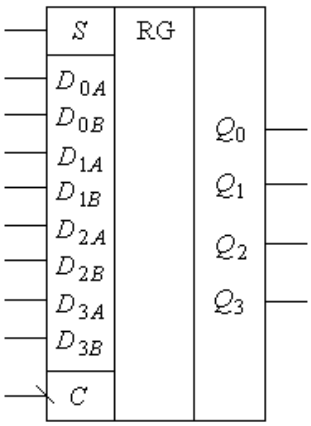


Рис. 7.3

Таблиця 7.1

Входи				Вихід
D_{iA}	D_{iB}	S	C	Q_i
0/1	x	0	⌊	0/1
x	0/1	1	⌊	0/1

Для забезпечення запису та зчитування інформації по одній шині даних може бути запропонована схема, приведена на рис. 7.4 для одного розряду даних.

При високому рівні сигналу на вході $R/\overline{WR}$ інформація з виходу *D*-тригера (мікросхема DD2) зчитується через елемент DD3 на шину даних d_i. При низькому рівні сигналу $R/\overline{WR}$ інформація з шини даних записуватиметься в тригер регістра.

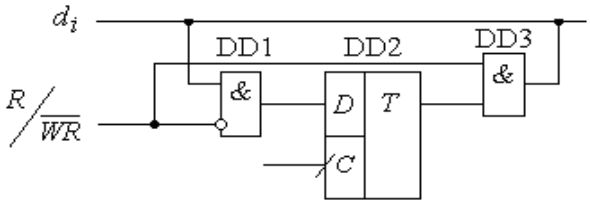


Рис. 7.4

Звичайно, це досить спрощена структура портів вводу / виводу інформації у паралельному форматі. Реальні порти значно складніші апаратно і вимагають відповідної ініціалізації. Але подібна схемотехніка використовується у

багаторегістрових структурах для забезпечення міжрегістрових пересилок.

На рис. 7.5 приводиться функціональна схема, що містить чотири регістри, які об'єднані єдиною шиною даних ($d_0 \ldots d_3$).

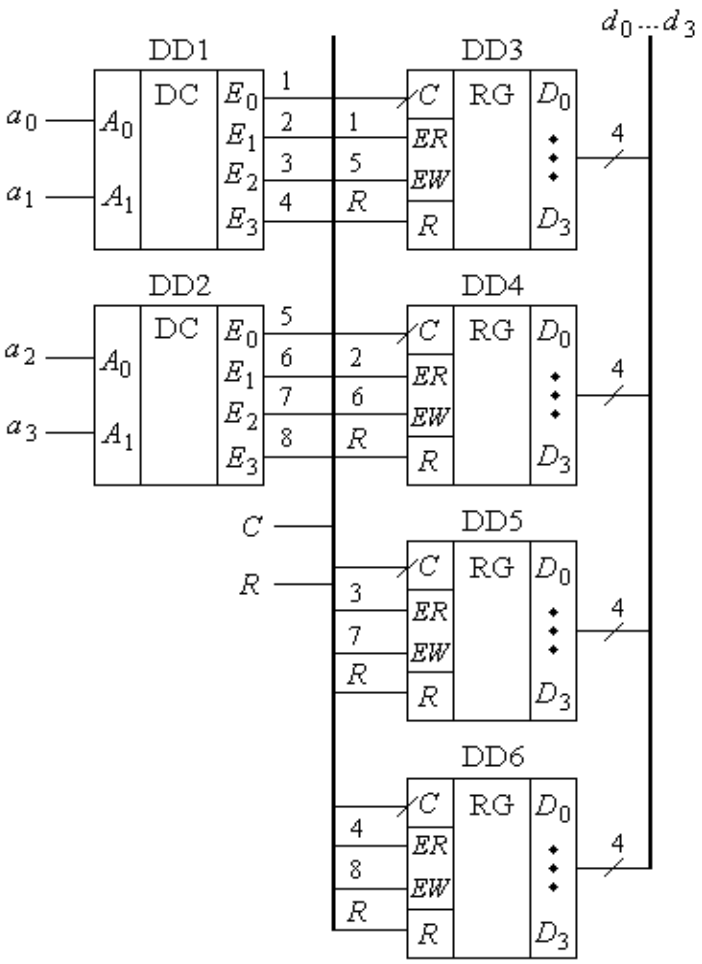


Рис. 7.5

Дешифратори DD1 і DD2 на основі адресних сигналів ($a_0 \ldots a_3$) генерують сигнали дозволу на читання інформації з регістра *ER* та запису *EW*. При виборі двох регістрів за їх адресами на відповідних входах *ER* і *EW* встановлюються сигнали високого рівня, що дозволяє за фронтом синхросигналу *C* зчитати інформацію з одного регістра та записати в інший.

Приклад 7.1. Сформувати код двійкової команди для перезапису інформації з регістра DD4 в регістр DD6.

Розв'язання. Створимо таблицю адресації регістрів при зчитуванні та запису (табл. 7.2). Зрозуміло, що для перезапису інформації з регістра DD4 у регістр DD6 її необхідно спочатку зчитати з регістра DD4, а потім записати в регістр DD6. Виходячи з даних таблиці, двійковий код команди матиме вигляд: $EW \cdot ER = a_3\, a_2\, a_1\, a_0 = 1101$.

Таблиця 7.2

a_1	a_0	ER	a_3	a_2	EW
0	0	DD3	0	0	DD3
0	1	DD4	0	1	DD4
1	0	DD5	1	0	DD5
1	1	DD6	1	1	DD6

Міжрегістрові пересилки досить широко використовуються у мікропроцесорній техніці. Своєрідними запам'ятовуючими пристроями є *регістрові файли*. Вони призначені для зберігання декількох слів, причому процеси запису одного слова та зчитування іншого відбуваються одночасно і незалежно.

На рис. 7.6 приводиться функціональна схема регістрового файлу на чотири чотирьохрозрядні слова. Файл побудований у вигляді матриці 4×4, де кожен стовпець тригерів призначений для запам'ятовування одного слова, а кожен рядок – відповідно, розряди кожного з слів.

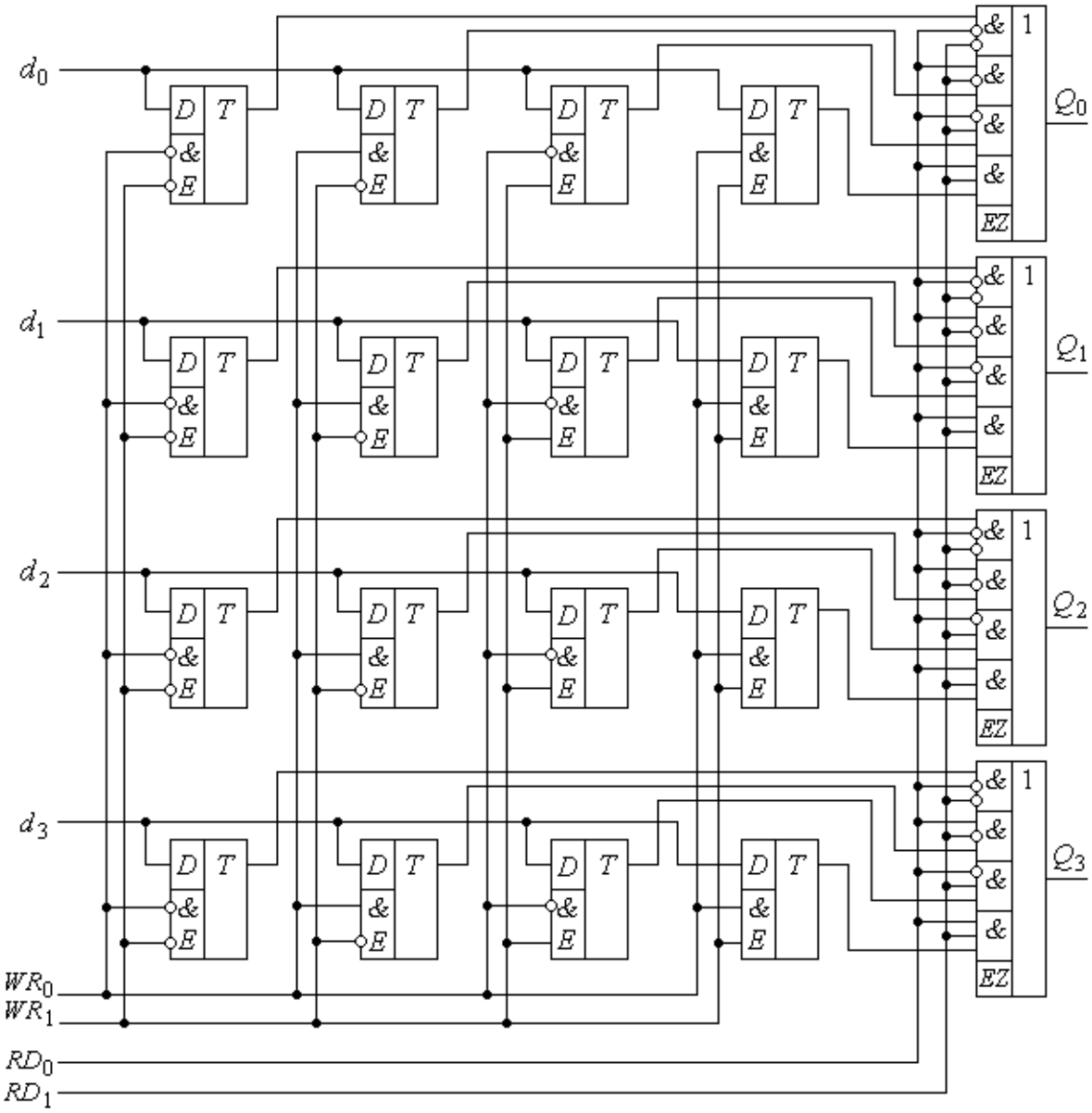


Рис. 7.6

Запис у будь-який стовпець забезпечується набором його адреси на входах $WR_1\ WR_0$. При цьому на кожному з тригерів стовпця з'явиться сигнал дозволу запису E, за яким в асинхронному режимі слово з шини даних $d_0 \ldots d_3$ запишеться у тригери стовпця.

Зчитування інформації забезпечується незалежно за допомогою логічних елементів **4(3І-АБО)**. Вибір будь-якого стовпця забезпечується вхідною логікою елементів **3І**, оскільки два входи кожного з них – адресні і приєднані до входів $RD_1\ RD_0$, які задають адресу слова, що зчитується.

Виходи регістрового файлу мають *Z*-стан, що дає можливість з'єднувати їх в одній точці. Для мікросхем КР1533 допускається з'єднання до 128 виходів, що дозволяє створювати запам'ятовуючі пристрої на 512 слів. Збільшення ємності такої пам'яті можливе з використанням допоміжних елементів. Регістрові файли дозволяють також збільшувати розмір слова до 8, 12, 16 розрядів шляхом паралельного з'єднання входів дозволу.

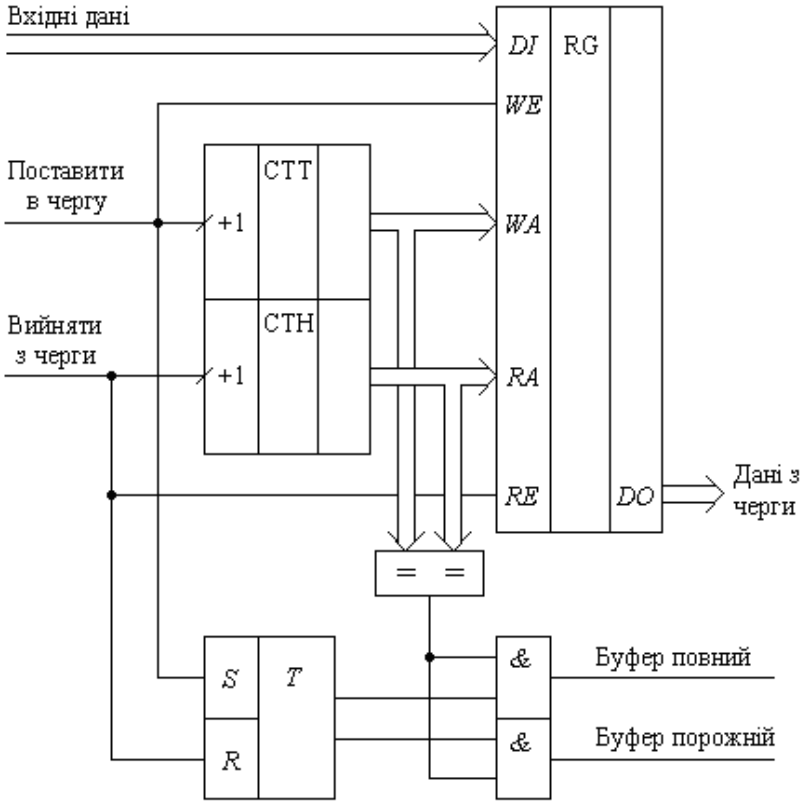
Регістрові файли часто використовуються для створення швидкодіючих запам'ятовуючих пристроїв малої ємності.

Окрім адресованих регістрів загального призначення, у цифровій апаратурі використовуються допоміжні запам'ятовуючі пристрої з "неявно" вираженою адресацією, які служать для зберігання черг і називаються іноді *"буферами даних"*. Ці пристрої часто будують на основі регістрової пам'яті.

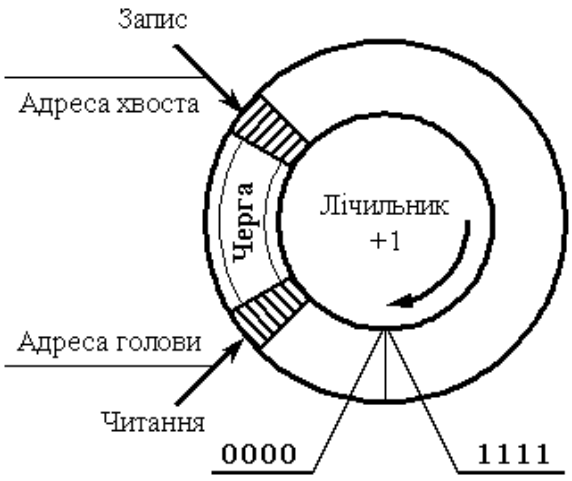
Необхідність у такому буфері виникає тоді, коли джерело даних постачає приймачеві слова, що розподілені у часі нерівномірно, причому інтервали часу між двома словами можуть іноді бути меншими, ніж час, необхідний приймачеві для обробки одного слова. Подібна ситуація може приводити до того, що частина даних втрачатиметься. Якщо такі втрати недопустимі, то між джерелом та приймачем включається *буфер "черги"* для створення і зберігання черги слів для їх наступної послідуючої обробки, або *буфер типу FIFO* (***F****irst* ***I****n* - ***F****irst* ***O****ut* – перший ввійшов - перший вийшов).

Функціональна схема буфера, побудованого на основі регістрової пам'яті, зображена на рис. 7.7, *а*. На рис. 7.7, *б* набір регістрів пам'яті, точніше адреси регістрів пам'яті, зображені у вигляді кільця.

Частина регістрів заповнена словами черги – решта вільні. Адреса запису при постановці в чергу задається лічильником хвоста черги (СТТ).



а)



б)

Рис. 7.7

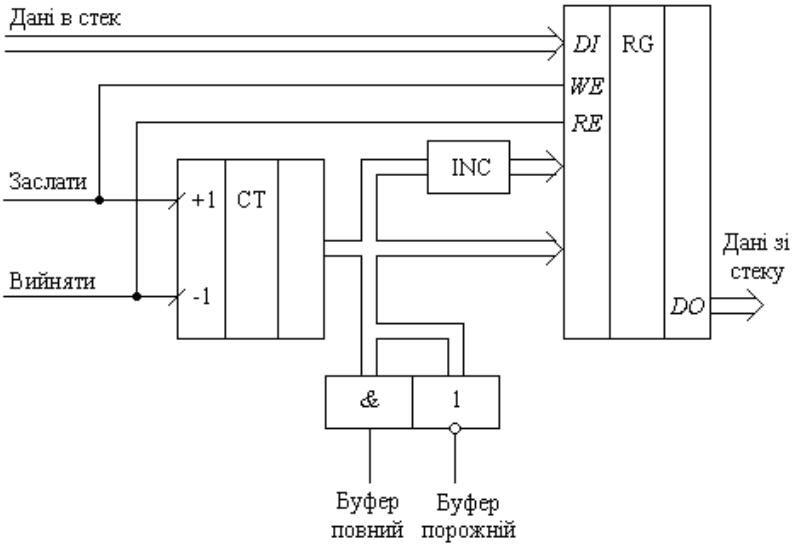
Сигнал “Поставити в чергу” подається на вхід *WE* дозволу запису і забезпечує запис слова даних, що поступили по вхідній шині *DI* (***D**ata **I**n*) у той регістр пам’яті, номер якого зберігається в СТТ. За зрізом сигналу “Поставити в чергу” вихідний код СТТ збільшується на **1**, готовлячи адресу запису для чергового слова. При появі сигналу “Вийняти з черги” на вихідній шині *DO*

(***D**ata **O**ut*) з’явиться слово, яке зберігалося в тому регістрі пам’яті, номер якого задається кодом лічильника голови СТН. За зрізом сигналу вихідний код лічильника збільшується на **1**, готовлячи тим самим до видачі наступне слово, яке стало першим у черзі. Переповнення лічильника СТТ не призводить до ускладнення ситуації, оскільки після максимально можливого коду **1111…11** у ньому автоматично з’явиться код **0000…00**. Черга в кільці переміститься «хвостом» через нульову відмітку лічильника. Так само з часом матиме місце переміщення «голови». В процесі нормальної роботи черга рухатиметься по кільцю значень адрес за часовою стрілкою «хвостом» вперед, зростаючи чи скорочуючись у відповідності до швидкості передавача. Схема буфера *FIFO* повинна сигналізувати про появу двох особливих ситуацій. Перша – “буфер повний”. Це означає, що в нього неможливо більше записувати і необхідно зупинити передачу даних. Друга – “буфер порожній” (неможливо брати дані і необхідно зупинити приймач). Обидві ситуації мають спільну ознаку: рівність кодів обох лічильників після зникнення вхідного сигналу, яка виявляється компаратором. Якщо коди лічильників зрівнялись після чергової вибірки з черги, то це означає, що черги немає і буфер порожній. Якщо ж коди лічильників зрівнялись після чергового запису в чергу, то буфер повний. Характер останнього звернення до буферу запам’ятовується в *RS*-тригері, який при появі особливих ситуацій через елементи **2І** видає відповідні інформаційні сигнали, призначені для подальшого прийняття рішень.

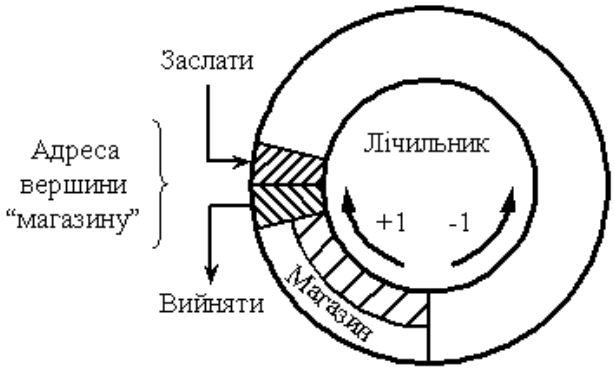
Іншим буфером, що широко використовується, є *буфер типу “магазин”*, або *“стек”* (*Stack*), або *буфер типу LIFO* (***L**ast **I**n - **F**irst **O**ut* – останній ввійшов - перший вийшов).

Стекові структури даних використовуються у цифрових пристроях, в яких процес виконання завдання переривається більш терміновим, тому всі дані, що були пов’язані з перерваним завданням, розміщуються у буфері типу “магазин” для тимчасового зберігання. Виконання термінового завдання також може бути перервано появою більш термінового, і т. п. Внаслідок таких переривань у

буфері накопичуватимуться нові й нові дані, створюючи чергу слів, які очікують обробки. Вибірка з буфера слів відбувається у відповідності до вимог їх черговості, внаслідок чого першими виймаються слова, які були занесені в буфер останніми.



а)



б)

Рис. 7.8

На рис. 7.8, а приведена функціональна схема буфера типу “магазин”. Основою його є регістрова пам’ять, а також реверсивний лічильник адреси, в якому зберігається номер регістра вершини стеку (рис. 7.8, б). Коли подається команда “Заслати у стек” (Push), вміст лічильника інкрементується на 1; при команді “Вийняти зі стеку” (Pop) його вміст декрементується. Як видно з діаграми, адреса, за якою засилається слово у стек, завжди на 1 більше адреси, за якою слово зчитується зі стеку. Постійну різницю на 1 між адресами запису

та зчитування підтримує пристрій, який називається інкременатором (INC). Два особливі стани – “буфер повний” та “буфер порожній” – визначаються за максимально можливим і нульовим значенням вмісту лічильника.

Інші варіанти побудови стеку будуть розглянуті пізніше.

7.3. Конвеєрні пристрої

При реалізації складних логічних задач складність комбінаційної схемотехніки суттєво зростає. Зростають також величини часових затримок у комбінаційних схемах, що, як відмічалося вище (див. **Розділ 3**), може приводити до критичних змагань (“гонок”). Для боротьби з ними використовується синхронізація, яка забезпечується тригерними пристроями. Якщо кількість виходів КС зростає, то зростає і кількість розділюючих синхронних тригерів. У такому випадку замість тригерів можуть бути встановлені паралельні регістри пам’яті, внаслідок чого принцип обробки інформації комбінаційними схемами може бути принципово змінений, оскільки з’являється можливість конвеєрної обробки інформації.

Структурна схема конвеєра для обробки інформації приведена на рис. 7.9. Конвеєр складається з буферних регістрів RG1, RG2 та послідуючими за ними виконавчими схемами, які можуть бути комбінаційними або послідовнісними і разом з вхідними регістрами називаються *стадіями*. Стадія виконує конкретну операцію і видає проміжний результат на наступну за нею стадію.

Завершує схему вихідний буферний регістр. Сигнал таймера подається одночасно на кожен буферний регістр, і за кожним імпульсом таймера кожна стадія передає свої проміжні результати на вхідний регістр наступної стадії. Кінцевий результат отримується після того, як вхідні дані будуть послідовно оброблені в кожній стадії.

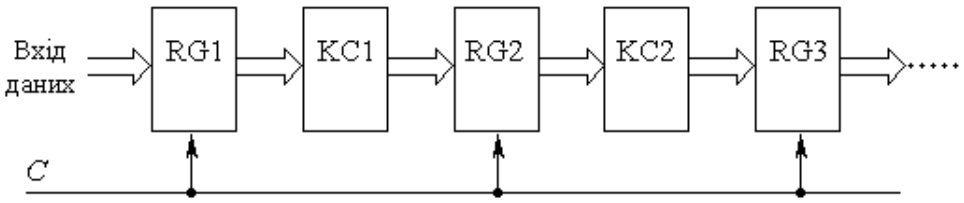


Рис. 7.9

Період імпульсів таймера повинен бути достатнім для проходження оброблюваними даними найповільнішої стадії, включаючи сюди інтервал часу для запису результатів у проміжний регістр. Тобто тактовий інтервал часу T_k складається з часу $T_В$ – часу виконання операції та $T_З$ – часу запису та зберігання в буфері, тобто:

$$T_k = T_В + T_З \approx T_З , \qquad \text{оскільки } T_З >> T_В .$$

Час виконання n задач T_n, які подаються послідовно на вхідну стадію конвеєра, що має m стадій, теоретично знаходиться за формулою:

$$T_n = m \times T_k + (n - 1) \times T_k ,$$

де $m \times T_k$ – час, необхідний для виконання першої вхідної задачі;

$(n - 1) \times T_k$ – час, необхідний для виконання решти задач.

З формули витікає, що після завантаження конвеєра він оброблятиме кожну задачу протягом одного такту T_k, і тому при розв’язанні типових задач швидкість виконання операцій буде в m разів більшою, порівняно з безконвеєрною обробкою.

Конвеєрна обробка інформації використовується в тих ситуаціях, коли необхідна однотипна обробка вхідної інформації. Конвеєри широко використовуються як у спеціалізованих процесорах (наприклад, сигнальних), так і в процесорах загального призначення.

Сучасні конвеєрні засоби обробки інформації є досить розвиненими. Упроцесорах часто встановлюється декілька конвеєрів, які можуть гнучко переналагоджуватись у процесі роботи.

7.4. Регістри зсуву

Регістрами зсуву називаються регістри, вміст яких при подачі керуючих сигналів може зміщуватись у бік старших чи молодших розрядів. Такі пристрої складаються з ряду динамічних тригерів, спрацьовування яких виконується за фронтом або спадом синхронізуючого імпульсу *C*-. У ряді літературних джерел останніх років активно використовується розподіл тригерів на “прозорі” та “непрозорі”.

Під терміном *“прозорість”* мається на увазі властивість тригера при активному рівні сигналу на вході *C*- адекватно з його типом відслідковувати на виході зміни інформаційних сигналів керуючих входів; якщо це *D*-тригер, то просто повторюється стан *D*-входу. Іншими словами, цим терміном охоплюються тригери зі статичним входом.

“Непрозорість” тригерів – властивість навіть при активному рівні сигналу на *C*-вході не передавати на вихід змін сигналів керуючих входів, що з’являються вслід за перемикаючим фронтом. Цією властивістю охоплюються всі тригери, які перемикаються за фронтом або зрізом синхроімпульсу.

Складність застосування статичних тригерів у регістрах зсуву обумовлена тим, що при активному рівні сигналу на синхронізуючих входах всіх тригерів перемикання одного з них призведе до перемикання наступних, а їх кількість визначатиметься тривалістю синхросигналу та швидкодією тригерів. Однак під цим не слід розуміти, що застосування статичних тригерів у схемах з послідовним зсувом принципово неможливе. Така можливість з’являється, наприклад, при використанні двофазної синхронізації, яка дозволяє розділити у часі процеси перемикання на окремих ділянках схеми. Прикладом ефективного застосування двофазної синхронізації може служити мікропроцесорний комплект ВІС серії К580. В інтегральних схемах середнього ступеню інтеграції використовується однофазна синхронізація.

На рис. 7.10 наведена схема регістра зсуву, побудованого на *D*-тригерах.

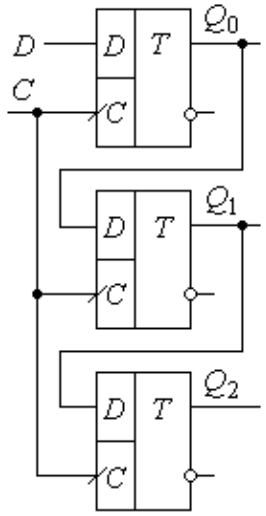


Рис. 7.10

Інформація, призначена для запису в регістр і представлена у послідовному форматі, поступає на *D*-вхід першого тригера. Її запис у перший та наступні тригери здійснюється за фронтом синхроімпульсу, що поступає одночасно на входи *C*- всіх тригерів. Зчитування інформації виконується також у послідовному форматі з виходу *Q* останнього тригера шляхом подання на синхровходи серії імпульсів. У паралельному форматі вона може бути зчитана з виводів *Q* тригерів, якщо такі виводи передбачені в мікросхемі регістра. За подібною схемою реалізована МС 564ИР2, що містить два незалежні чотирьохрозрядні однонаправлені регістри.

На рис. 7.11 приводиться варіант схеми регістра зсуву на *JK*-тригерах, кожен з яких працює в режимі *D*-тригера.

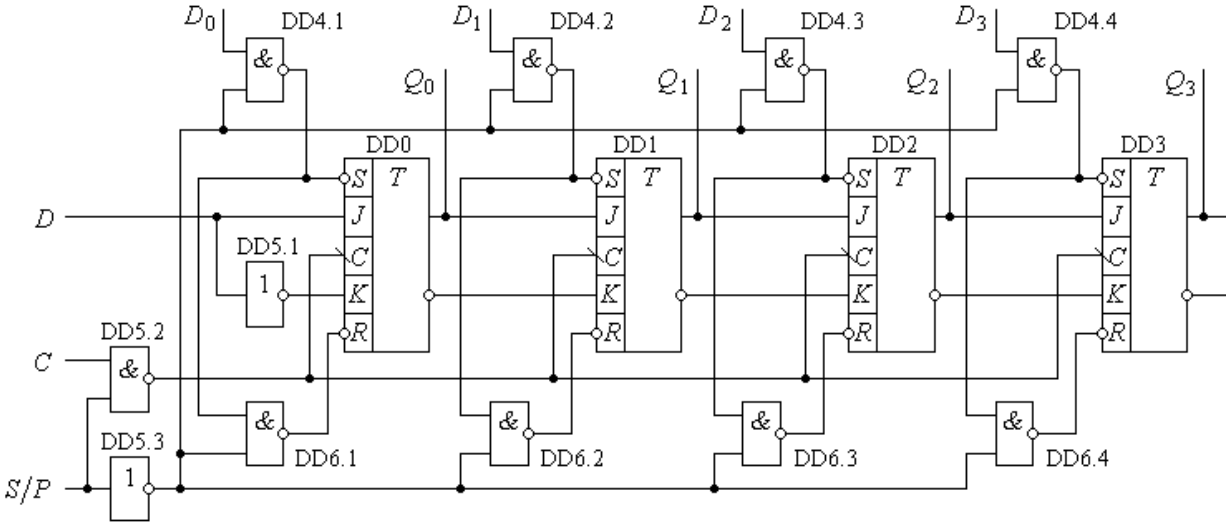


Рис. 7.11

Цей режим забезпечується для тригерів DD1…DD3 протифазними виходами попередніх, а *D*-режим тригера DD0 забезпечується інвертором DD5.1. Доповнений логічними елементами DD4.1…DD4.4, він дозволяє виконувати послідовне та паралельне введення інформації, а також послідовне або паралельне її виведення.

Часова діаграма, яку наведено на рис. 7.12, ілюструє процедуру послідовного введення по входу *D*- за наявності тактових імпульсів *C*- та логічної одиниці на вході вибору режимів введення: $S/P = 1$. Розглянемо особливості роботи регістра при послідовному введенні інформації. Припустимо, що в регістр послідовно вводиться, починаючи зі старшого розряду, двійковий код **1101**, який подається синхронно з тактовими імпульсами *C*-.

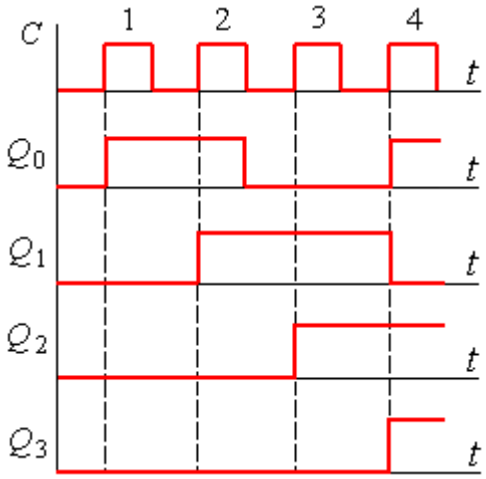


Рис. 7.12

Враховуючи, що тригери повинні спрацьовувати за зрізом вхідногосигналу, який подається на їх *C*-входи, а синхросигнал регістр а інвертується елементом DD5.2, запис інформації забезпечуватиметься за фронтом синхросигналу регістра.

При цьому інформація, що записується у тригери, повинна бути підготовленою до подачі синхросигналу (див. **Розділ 4**). Наприклад, для тригерів серії КР1533 інтервал часу підготовки повинен бути не меншим 20 нс. Оскільки всі тригери регістра сприймають синхросигнал одночасно, то у кожний з них запишеться інформація, що мала місце на інформаційних входах безпосередньо перед зрізом синхросигналу тригера (фронтом синхросигналу регістра). Через час затримки розповсюдження сигналу (для КР1533 вона складає 15 нс) інформаційний сигнал буде записаний і з'явиться на *Q*-виході тригера.

Таким чином, за фронтом першого тактового імпульсу *C*- в DD0 буде записана одиниця молодшого розряду, а у решту тригерів перезапишуться нулі з прямих виходів попередніх тригерів (див. рис. 7.12). Перед приходом другого синхросигналу на входах DD0 і DD1 будуть присутні сигнали високого рівня, а на входах DD2 і DD3 – низького. Тому за фронтом другого синхросигналу в перший тригер DD0 знову перезапишеться **"1"**, в другий DD1 – також **"1"** з

виходу Q_0, а в DD2 і DD3 будуть записані нулі відповідно з виходів Q_1 і Q_2. Аналогічно, за третім синхросигналом **“1”** з виходів Q_0 і Q_1 перемістяться на виходи тригерів DD1 і DD2, а в DD0 буде записаний **“0”** з *D*-входу регістра. Так за кожним синхросигналом у регістрі матиме місце зсув інформації на один розряд вправо. Після чотирьох тактових імпульсів код на виходах $Q_0 \ldots Q_3$ відповідатиме вхідному коду і може бути зчитаний зовнішнім пристроєм. Розрядність регістра може бути підвищена нарощуванням подібних пристроїв, яке здійснюється шляхом під’єднання виходу Q_3 попереднього регістра до входу *D*- наступного з відповідним об’єднанням синхронізуючих та керуючих входів.

У регістрі, виготовленому відповідно до розглянутої схеми, є можливість перетворювати інформацію, задану в послідовному форматі, у паралельний, тобто виконувати функцію демультиплексора. Але перевага регістра при виконанні такої функції полягає в тому, що інформація в паралельному форматі може зчитуватись одночасно з усіх розрядів, а не рознесена у часі, як у демультиплексорах. До того ж, інформація запам’ятовується і може бути зчитана багаторазово.

При наявності в регістрі *JK*-тригерів з асинхронними *R*- і *S*- входами можна забезпечити за допомогою допоміжної логіки (ЛЕ DD4) паралельний запис інформації $D_0 \ldots D_3$ у кожен з тригерів. Запис забезпечується асинхронно за низьким рівнем керуючого сигналу S/P (*Serial*/*Parallel*). Таким чином записати інформацію в паралельному форматі можна в інтервалі часу між двома синхроімпульсами.

Зчитування інформації у послідовному форматі забезпечується подачею на *C*-вхід серії синхросигналів у кількості, що дорівнює розрядності регістра. Зі сказаного витікає, що регістр зсуву може виконувати функцію мультиплексора з запам’ятовуванням.

Запис інформації в паралельному форматі може бути забезпеченим не тільки при використанні у регістрах *JK*-тригерів, а також при використанні

RS- та *D*-тригерів. Прикладом такого регістра може бути мікросхема ИР1, функціональна схема якої приведена на рис. 7.13, *а*, а умовне зображення – на рис. 7.13, *б*.

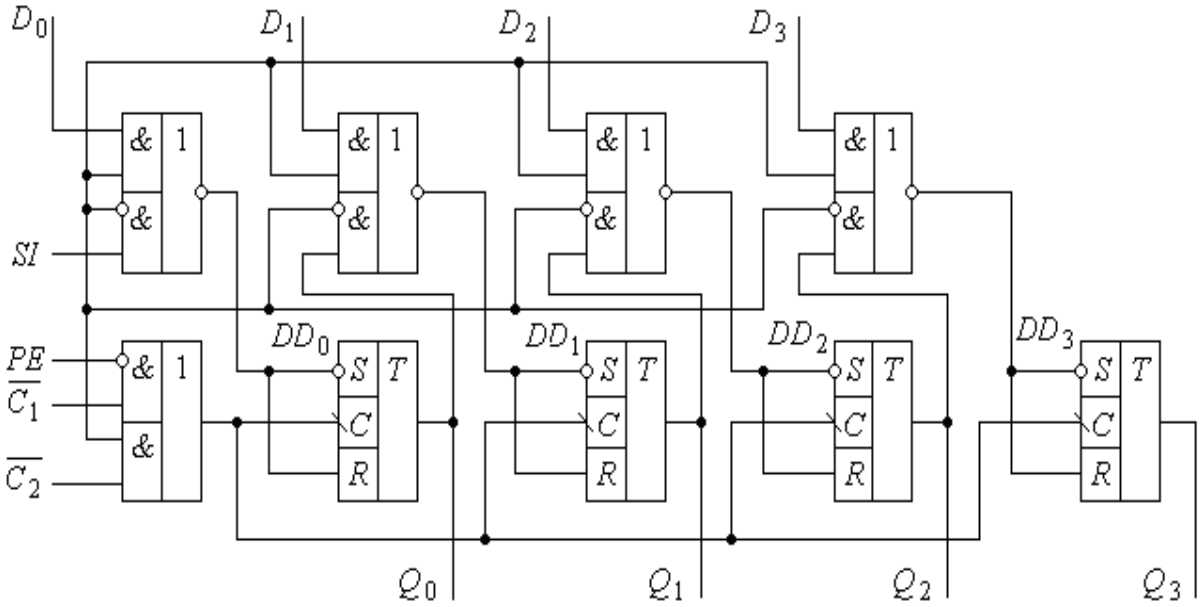


Рис 7.13 а)

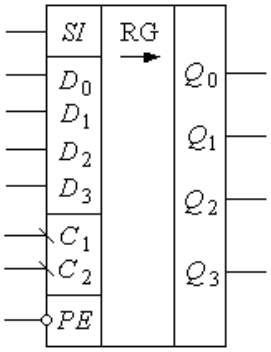


Рис. 7.13 б)

Кожен розряд регістра виготовлений на основі синхронного *RS*-тригера (DD0…DD3), що працює в режимі *D*-тригера з динамічним входом синхронізації *C*-. Регістр має чотири входи даних $D_0 \ldots D_3$ для завантаження інформації в паралельному форматі і один *SI* – для завантаження у послідовному. Регістр має два синхровходи – $\overline{C_1}$ і $\overline{C_2}$. Синхровхід $\overline{C_1}$ працює при *PE* = 0 і забезпечує синхронне послідовне завантаження регістра з входу *SI*, а також зсув вправо. При *PE* = 1 регістр працює за тактовим сигналом $\overline{C_2}$ і за його зрізом завантажує дані з паралельних входів $D_0 \ldots D_3$. Режими роботи регістра задаються його таблицею станів. Ступінь деталізації таблиці може бути різною, але вона повинна бути достатньою для того, щоб зрозуміти всі особливості взаємодії сигналів для кожного з режимів роботи регістра.

Оскільки в регістрі, що розглядається, є лише три керуючі сигнали, взаємозв'язані між собою, то достатньою є скорочена таблиця станів (табл. 7.3).

Таблиця 7.3

Входи			Режим
PE	C_1	C_2	
0	⌊	x	Запис послідовним форматом коду Зсув вправо
1	x	⌊	Запис паралелним форматом коду

Для забезпечення зсуву інформації вліво необхідно в регістрі створити зв'язки, які могли б дозволити переміщувати інформацію з правого тригера в лівий, а також передбачити два послідовних входи: *DR* – вхід для прийому інформації, що поступає в регістр зі сторони молодшого розряду та зсувається вправо, та *DL* – вхід для прийому інформації, що поступає в регістр зі сторони молодшого розряду та зсувається вліво. Ці входи використовуються і при нарощуванні регістрів: *DR* приєднуються до входу старшого розряду сусідньої молодшої секції загального регістру; *DL* – приєднується до виходу Q_0 молодшого розряду сусідньої старшої секції загального регістра.

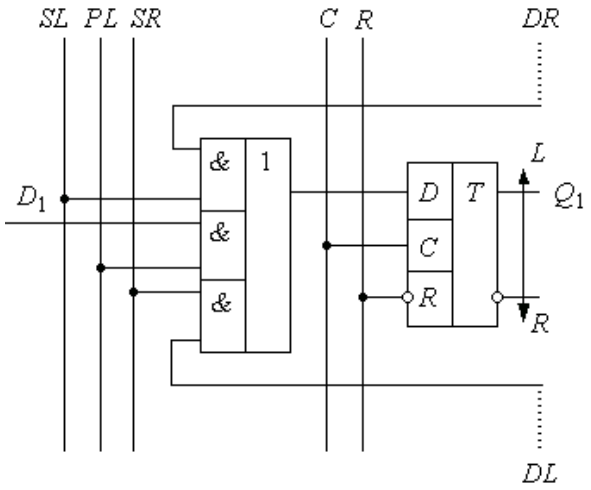


Рис. 7.14

Функціональна схема одного розряду двонаправленого регістра зсуву з можливістю паралельного завантаженні приведена на рис. 7.14.

До *D*-входу кожного тригера регістра приєднується вихід мультиплексора, який за допомогою керуючих сигналів *SL* (*Shift Left* – зсув ліворуч), *SR* (*Shift Right* – зсув праворуч) і *PL* (*Parallel Load* – паралельне завантаження) приєднує вхід *i*-го тригера відповідно до виходу наступного тригера, до виходу

попереднього тригера або до *i*-го розряду паралельного завантаження. За сигналом синхронізації у тригер завантажуються відповідні дані. Вихід *i*-го тригера приєднується до відповідних входів мультиплексорів сусідніх розрядів (напрямки *L*, *R*). У мікросхемах регістрів входи *SL*, *PL*, *SR* не виводяться з корпусу мікросхеми, а керування ними здійснюється через дешифратор режимів.

На рис. 7.15 приводиться спрощений приклад з'єднання трьох тригерів для забезпечення зсуву вліво та вправо відповідно до керуючих сигналів *SL* та *SR*.

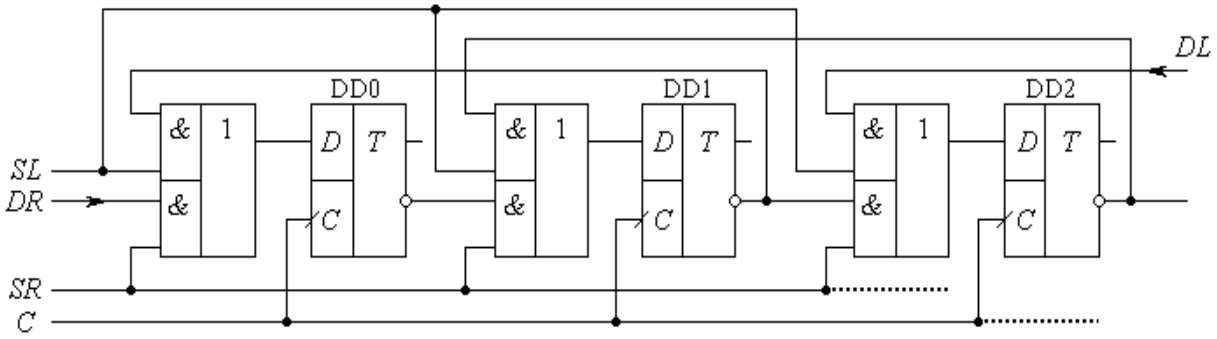


Рис. 7.15

З'єднання виходу наступного тригера з входом попереднього може бути виконано і за допомогою зовнішнього монтажу виводів у мікросхемах регістрів зсуву з паралельним введенням і виведенням інформації. Наприклад, у розглянутому регістрі ИР1 для забезпечення зсуву ліворуч необхідно зробити такі зовнішні зв'язки: Q_3 з'єднати з D_2; Q_2 з D_1; Q_1 з D_0. Вхід D_3 необхідно використовувати як вхід *DL* для введення інформації з старшого розряду зі зсувом вліво. Зсув ліворуч забезпечуватиметься за синхросигналом *C*-.

Забезпечення можливості зсувати записані дані ліворуч - праворуч відкриває простий спосіб ділення та множення даних на коефіцієнт (дільник / множник), кратний **2**. Дійсно, зміщення числа на один розряд ліворуч у межах розрядної сітки еквівалентно множенню на **2**, а зміщення на один розряд праворуч – діленню на **2** (наприклад, 1010 ↔ 0101).

7.5. Приклади мікросхем регістрів та особливості їх використання

7.5.1. Паралельні регістри

Найпростішими регістрами паралельного типу є набори тригерів *D*-типу – наприклад, мікросхеми КР1533ТМ7 (SN74LS75), що містять чотири *D*-тригери; ТМ8 (ALS175), що місять чотири *D*-тригери з прямими та інверсними виходами; ТМ9 (ALS174) з шістьма синхронними *D*-тригерами.

У групі регістрів паралельного типу близькими структурами до вказаних наборів тригерів є мікросхеми КР1533ИР22, КР1533ИР23, КР1533ИР37. Вони орієнтовані для роботи на низькоомне навантаження або навантаження з великою ємністю.

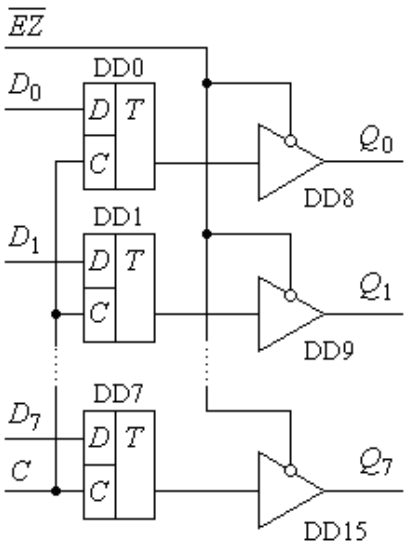


Рис. 7.16

Функціональна схема мікросхеми ИР22 приведена на рис. 7.16.

Регістр призначений для зберігання восьмирозрядного слова, що записується та зчитується у паралельному форматі. Мікросхема регістра містить вісім статичних синхронних *D*-тригерів, запис інформації в які забезпечується через паралельну шину даних $D_0 \ldots D_7$ за імпульсом синхросигналу (тобто при $C = 1$). При записі коду та його зберіганні на вході $\overline{EZ}$ повинен бути сигнал низького рівня. Вихідна шина даних $Q_0 \ldots Q_7$ приєднана до виходів тригерів через буферні підсилювачі з трьома станами виходів, що забезпечує високу навантажувальну здатність. Оскільки при високому рівні синхросигналу інформація з *D*-входів передається безпосередньо на вихід, то регістр забезпечує високу швидкодію при записі інформації та керуванні її передачею. Тому ці регістри знаходять широке використання в мікропроцесорних системах з магістральною організацією. При $C = 0$ регістр знаходиться у режимі зберігання. Для переводу виходів

мікросхеми у *Z*-стан необхідно на вхід $\overline{EZ}$ подати високий рівень сигналу, при цьому стан інших входів байдужий. Система керування *Z*-станом побудована так, що при зниженні напруги живлення до 3 В вона автоматично переводить мікросхему у третій стан незалежно від інформації на вході $\overline{EZ}$. Така особливість виключає можливість появи крізних струмів в інформаційних магістралях. Завдяки цьому, високій швидкодії та навантажувальній здібності дана мікросхема, знаходить широке використання в мікропроцесорних системах у якості буферного регістра, магістрального пристрою для прийому і передачі інформації та за іншими призначеннями.

Особливість регістрів ИР23 та ИР37 полягає в тому, що запис інформації в тригери забезпечується за фронтом синхросигналу. Оскільки розрахункова тривалість фронту синхроімпульсу менша інтервалу часу передачі інформаційного сигналу з *D*-входу тригера на його вихід, то регістр ИР23 часто використовується в якості регістра зсува. Для цього вихід попереднього розряду елемента пам'яті (наприклад, Q_0) підє'днаний до входу наступного (D_1). При цьому регістр має можливість забезпечувати послідовний запис інформації (через D_0) і зчитування (через Q_7). Як послідовний, регістр ИР23 широко використовується в апаратно-програмних засобах відображення інформації (світлодіодних панно, панелях, біжучих рядках тощо). Завдяки тому, що струм виходів мікросхеми $I^1_{\text{вих}} = 19$ мА, $I^0_{\text{вих}} = 29$ мА, світлодіоди можуть бути приєднані безпосередньо до виходів $Q_0 \ldots Q_7$.

Мікросхема КР1533ИР27 (SN74LS377) є близьким малопотужним аналогом до ИР23. Вона не має *Z*-стану, але для забезпечення запису інформації за фронтом *C*-сигналу необхідно на вхід $\overline{EWR}$ подати сигнал низького рівня. Вона також може працювати в режимі однонаправленого регістра зсуву, але малопотужні виходи *Q* (струм $I^0_{\text{вих}} = 0{,}2$ мА) обмежують його використання.

Мікросхема КР1533ИР33 (SN74ALS573) є повним аналогом регістра ИР22. Мікросхема КР1533ИР35 близька до мікросхем ИР23 та ИР27, але, на відміну

від них, має загальний вхід обнуління $\overline{R}$, який є асинхронним і пріоритетним перед іншими входами. Виходи мікросхеми малопотужні.

Мікросхема КР1533ИР15 (SN74LS173A) – чотирьохрозрядний регістр *D*-типу, який має по виходу три стани, а також прямий вхід обнуління *R*. Умовне позначення мікросхеми приведено на рис. 7.17, а таблиця станів, що відображає режими його роботи, – у табл. 7.4.

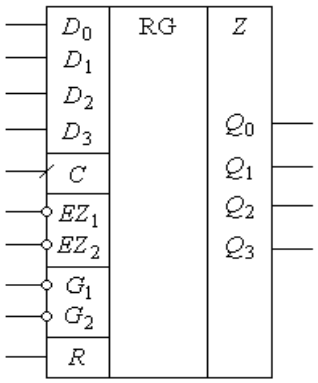


Рис. 7.17

Таблиця 7.4

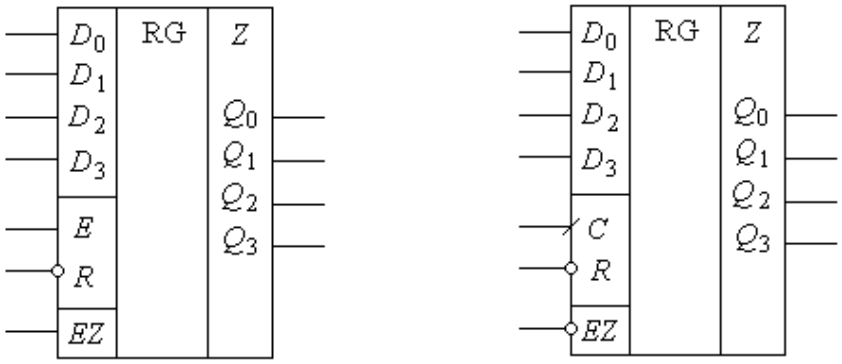
Входи							Q_i	Режим
R	C	G_1	G_2	EZ_1	EZ_2	D		
H	x	x	X	x	x	x	L	Установка в **"0"**
L	⌠	L	L	x	x	1/0	1/0	Запис
L	L	x	x	x	x	x	d_i	Зберігання
L	⌠	H	x	x	x	x	d_i	
L	⌠	x	H	x	x	x	d_i	
L	x	x	x	0	0	x	d_i	Вивід

Установка тригерів у нуль забезпечується високим рівнем сигналу *R* незалежно від рівнів інших сигналів. Виведення інформації на шину *Q* можливе лише за умови $\overline{EZ_1}\ \overline{EZ_2} = 00$. Високий рівень сигналу на будь-якому з цих входів переводить виходи мікросхеми у *Z*-стан, але при цьому режими запису і обнуління працюють незалежно. Входи G_1 і G_2 є входами дозволу запису. Високий рівень сигналу на будь-якому з цих входів блокує режим запису.

Чотирьохрозрядними є також буферні регістри у мікросхемах КР1533ИР34 (SN74ALS873) та КР1533ИР38 (SN74ALS874). У кожній з мікросхем містяться по 2 незалежні регістри. Умовне позначення одного регістра першої мікросхеми приведене на рис. 7.18, *а* ; другої – на рис. 7.18, *б*.

Буферизовані виходи з *Z*-станом в обох мікросхемах майже ідентичні. Вони мають підвищену навантажувальну здібність ($I^0_{\text{вих}} = 29$ мА), достатню для роботи в якості магістральних приймачів / передавачів.

Єдина різниця полягає в тому, що регістри мікросхеми ИР34 переводяться у *Z*-стан високим рівнем сигналу *EZ*, а ИР38 – низьким.



а) ***б)***

Рис. 7.18

Регістри обох мікросхем спроектовані так, що при пониженні напруги живлення до 3 В виходи переводяться у *Z*-стан з метою виключення появи в магістралях наскрізних струмів. Входи регістрів обох мікросхем також близькі за функціональними властивостями. Різниця полягає лише в тому, що у ИР34 вхід дозволу запису інформації статичний і запис інформації забезпечується високим рівнем сигналу, а у мікросхемі ИР38 вхід динамічний, а запис забезпечується за фронтом. Обидві мікросхеми мають можливість шляхом нарощування їх розрядів створити один регістр на вісім розрядів. У мікросхемі ИР38, за аналогією з раніше описаним способом, можна створити режим послідовного по входу регістра на вісім розрядів, фактично повторюючи описані вище режими регістра ИР23.

Мікросхема КР1533ИР30 –восьмирозрядний регістр зберігання з адресацією. Він містить вісім *D*-тригерів з прямими виходами, але запис інформації в кожен з них відбувається через єдиний *D*-вхід шляхом адресного звернення по шині *A* ($A_2\,A_1\,A_0$).

Умовне позначення регістра приведене на рис. 7.19.

Адресація до кожного з тригерів, що мають виходи $Q_0 \ldots Q_7$, відбувається у відповідності до послідовності мінтермів трьохрозрядного двійкового коду.

Режими роботи регістра пояснюються таблицею станів (табл. 7.5). Вхід $\overline{E}$ – вхід дозволу запису.

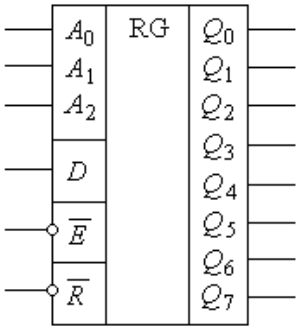


Рис. 7.19

Таблиця 7.5

Входи		Вихід адресованого тригера	Вихід неадресованих тригерів	Режим
$\overline{R}$	$\overline{E}$			
H	L	D	Q_{i0}	Адресація
H	H	Q_{i0}	Q_{i0}	Зберігання
L	L	D	L	Демультиплексор
L	H	L	L	Обнуління

Для виключення похибок у роботі мікросхеми на вході $\overline{E}$ необхідно тримати сигнал високого рівня при зміні адресних сигналів.

У режимі дешифратора / демультиплексора на адресний вхід поступає інформація зі входу *D*-. При цьому решта виходів знаходиться у стані низького рівня напруги. Така особливість можлива тому, що, по-перше, у режимі демультиплексування одночасно з адресними сигналами і сигналом дозволу $\overline{E}$ діє низький рівень сигналу на вході $\overline{R}$, а, по-друге, *D*-тригери побудовані так, що вхід $\overline{R}$ блокується при подачі безпосередньо на тригер сигналу дозволу завантаження $EWR = \overline{E} \cdot A$.

7.5.2. Регістрові файли

До регістрових файлів відносяться дві мікросхеми з близькими структурами і технічними характеристиками – КР1533ИР26 (SN74LS670) і КР1533ИР32 (SN74LS170).

Умовне зображення мікросхеми ИР26 приведено на рис. 7.20. Мікросхема має чотири інформаційні входи $D_0 \ldots D_3$, які використовуються для запису чотирьох чотирьохрозрядних слів, і чотири виходи $Q_0 \ldots Q_3$ з *Z*-станом.

За допомогою входів WR_1 WR_2 та RD_1 RD_2 забезпечується можливість незалежних запису та зчитування слів за індивідуальною адресою.

Адресація як запису, так і зчитування окремих слів двійкова і описується незалежно мінтермами від **0** до **3**. Запис слова відбувається за вибраною адресою при низькому рівні сигналу на вході дозволу запису *EWR*. Аналогічно, зчитування інформації забезпечується за адресами, заданими на входах RD_1 та RD_2 при низькому рівні сигналу на вході дозволу зчитування *ERD*. Високий рівень сигналу *EWR* забезпечує режим зберігання інформації, а високий рівень сигналу на вході *ERD* переводить виходи мікросхеми у *Z*-стан. Мікросхема має порівняно високу потужність виходів, що забезпечує їй можливість працювати безпосередньо у магістралях мікропроцесорних систем.

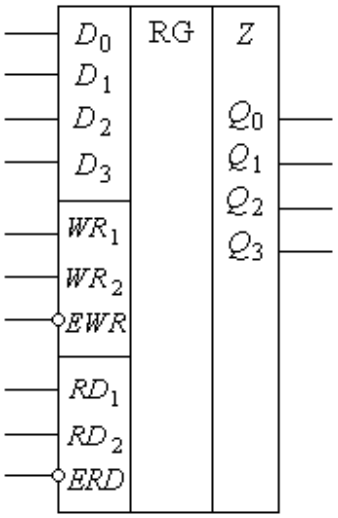


Рис. 7.20

Відмінність мікросхеми ИР32 полягає лише в тому, що її виходи $Q_0 \ldots Q_3$ виготовлені у вигляді відкритого колектора, що дозволяє використовувати її при роботі у магістралях, а також об'єднувати декілька виходів для отримання функції **"Монтажне І"**.

7.5.3. Послідовні регістри

Мікросхема КР1533ИР8 (SN74ALS164) – восьмирозрядний зсувний регістр з послідовним завантаженням і паралельним виводом інформації. Функціональна схема регістра зображена на рис. 7.21. Запис інформації забезпечується в послідовному форматі від двох взаємно незалежних входів *A* і *B* за фронтом синхросигналу *C*-. Вхід $\overline{R}$ – асинхронний. Він дозволяє встановлювати всі тригери у початковий нульовий стан. Послідовне виведення інформації може бути забезпечене з виводу Q_7 при подачі серії з восьми синхроімпульсів на вхід *C*-.

Регістри ИР9 та ИР10 тієї ж серії, навпаки, забезпечують можливість паралельного завантаження одного байту інформації. Умовне позначення

регістра ИР9 приведене на рис. 7.22, а особливості його роботи розкриває табл. 7.6.

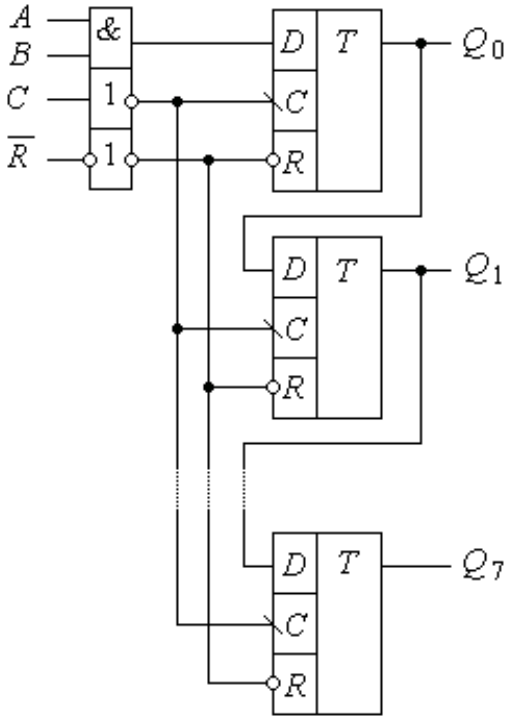


Рис. 7.21

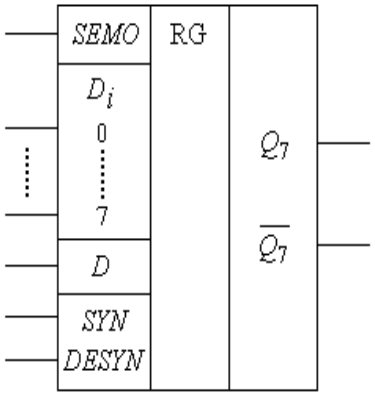


Рис. 7.22

Таблиця 7.6

Входи			Режим
SEMO	*SYN*	*DESYN*	
L	x	x	Паралельне завантаження
H	H	x	Зберігання
H	x	H	Зберігання
H	L	⎍	Зсув
H	⎍	L	Зсув

Як витікає з табл. 7.6, завантаження інформації в паралельному форматі забезпечується при низькому рівні сигналу на вході *SEMO*. Інформація для завантаження подається на входи даних D_i (0…7) і зберігається у прямому коді.

Завантаження у послідовному форматі через вхід *D*- забезпечується в режимі зсуву за фронтом синхросигналу, що подається на вхід *SYN* або блокування *DESYN*. Високий рівень потенційного сигналу на одному з цих входів блокує роботу регістра. Інформація виводиться лише в послідовному форматі в прямому або інверсному кодах. Особливість ИР9, порівняно з ИР10, полягає в наявності асинхронного обнуління $\overline{R}$, наявності виходу лише для прямого коду і забезпеченні зсуву за фронтом синхросигналу *SYN*.

Мікросхема КР1533ИР13 (SN74ALS198), на відміну від попередніх, є повнофункціональним реверсивним регістром, який забезпечує як паралельне введення і виведення інформації, так і послідовне, а також, зсув вправо або вліво. Умовне позначення мікросхеми приведено на рис. 7.23, а таблиця станів – у табл. 7.7. Тригери регістра встановлюються в нуль за асинхронним низьким рівнем сигналу на вході $\overline{R}$.

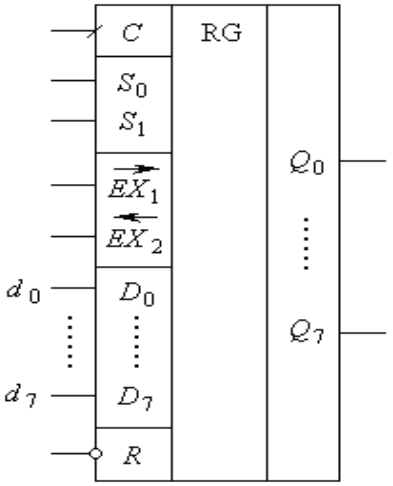


Рис. 7.23

Таблиця 7.7

Входи							Виходи		
$\overline{R}$	Вибір режиму		C	Зсув		Дані $D_0 \div D_7$	Q_0	$Q_1 \div Q_6$	Q_7
	S_1	S_0		EX_1 (вправо)	EX_2 (вліво)				
L	x	x	x	x	x	x	L	L	L
H	x	x	L	x	x	x	Зберігання		
H	H	H	⎍	x	x	$d_0 \div d_7$	d_0	$d_1 \div d_6$	d_7
H	L	H	⎍	H/L	x	x	H/L	$d_0 \div d_5$	d_6
H	H	L	⎍	x	H/L	x	d_1	$d_2 \div d_7$	H/L
H	L	L	x	x	x	x	Зберігання		

Високий рівень сигналу на цьому вході не впливає на режим роботи і вміст регістра. Низький рівень сигналу на синхровході при $\overline{R} = 1$ забезпечує режим зберігання інформації без зміни її положення. При цьому, відповідно до табл. 7.7, рівні сигналів на інших входах не мають значення. Для забезпечення синхронного паралельного завантаження за фронтом синхроімпульсу C- необхідно на входи $d_0 \ldots d_7$ подати у паралельному форматі восьмирозрядне слово для запису і встановити на входах S_0 і S_1 вибору режиму сигнали високого рівня. Дані завантажуються у тригери і появляються безпосередньо на вихідних шинах $Q_0 \ldots Q_7$. При паралельному завантаженні можливість зсуву в будь-який бік блокується. При $S_1 S_0 = 01$ маємо режим зсуву вправо, причому в молодший розряд і далі завантажуються дані з входу $\overrightarrow{EX_1}$. При $S_1 S_0 = 10$

забезпечується режим зсуву вліво, а в старший розряд Q_7 завантажується сигнал з входу $\overleftarrow{EX_2}$. При забезпеченні зсуву на декілька розрядів вправо або вліво дані у регістр подаються відповідно на кожному такті з входів $\overrightarrow{EX_1}$ або $\overleftarrow{EX_2}$. Виведення інформації у послідовному форматі забезпечується відповідно через виходи Q_7 або Q_0. Режим $S_1 S_0 = 00$ забезпечує зберігання інформації у тригерах регістра. Зміна рівнів сигналів на $S_1 S_0$ повинна забезпечуватись тільки при високому рівні напруги на тактовому вході C-.

Приклад 7.2. Побудувати часові діаграми керуючих сигналів для забезпечення циклічного запису інформації у паралельному форматі та її передачі через вихід Q_7 у зовнішню лінію в послідовному форматі.

Розв'язання. Для забезпечення такого циклічного режиму необхідно забезпечити $\overline{R} = 1$; $S_0 = 1$; $\overrightarrow{EX_1} = 0$; $\overleftarrow{EX_2} = 1/0$. Для паралельного запису необхідно $S_1 = 1$, а для зсуву вправо – $S_1 = 0$. Перемикання $S_1 \rightarrow 1/0$ повинно забезпечуватись при $C = 1$. Для запису необхідний 1 такт синхросигналу, для виведення інформації у послідовному форматі – 8 тактів. Забезпеченню вказаних режимів відповідає часова діаграма, приведена на рис. 7.24. На такті **0** забезпечується перемикання S_1 на режим паралельного запису, а за синхросигналом **1** буде записана інформація з шини даних $d_0 \ldots d_7$ у регістр. На тому ж такті S_1 перемикається на режим забезпечення зсуву вправо, і за синхросигналами **2 ... 9** інформація буде виведена з регістру в послідовному форматі. Тобто для забезпечення перетворення одного байту інформації з паралельного формату в послідовний необхідно **10** тактів синхросигналу.

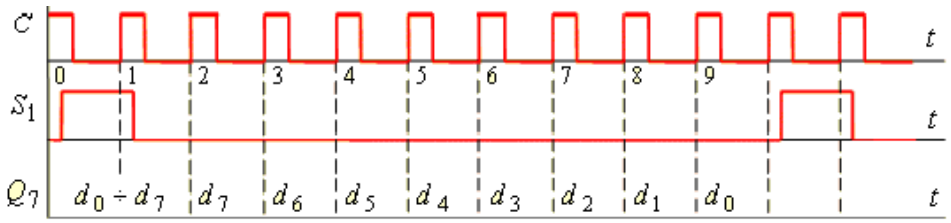


Рис. 7.24

Приклад 7.3. Одночасно (синхронно) з двох каналів інформація в паралельному форматі з частотою 8 кГц зчитується в два регістри ИР13, з'єднані послідовно. Визначити частоту тактових сигналів для забезпечення циклічного зчитування інформації в регістри і передачі в зовнішні пристрої у послідовному форматі.

Розв'язання. Незалежно від кількості паралельних каналів передачі даних, запис у регістри повинен забезпечуватись за 2 такти, а зчитування у послідовному форматі двох слів – за 16 тактів. Тобто 18 тактів повинні вкластися в 1 період частоти 8 кГц. Отже, частота тактових сигналів дорівнюватиме: $f_C = 18 \times 8 = 144$ кГц.

Приклад 7.4. Використовуючи таблицю станів мікросхеми КР1533ИР13 (табл. 7.7), пояснити, як можна забезпечувати періодичну затримку в передачі записаної інформації на *N* тактів.

Пояснення. Створення будь-якої затримки на *N* тактів може бути забезпечено шляхом переводу регістра після запису в нього інформації у стан зберігання. Оскільки режим зберігання забезпечується умовою $\overline{C} \vee \overline{S_1}\,\overline{S_2}$, то на вказаний інтервал часу необхідно за допомогою допоміжної логіки заблокувати подачу синхросигналів або встановити нульовий рівень сигналу S_0.

Мікросхема КР1533ИР16 є близьким і спрощеним аналогом ИР13. Це чотирьохрозрядний реверсивний регістр зсуву з можливістю за допомогою входу $\overline{EZ}$ забезпечувати високоімпедансний стан виходів. Зсув вліво схемотехнічно не передбачений, тому для його забезпечення необхідно виконувати відповідні зовнішні з'єднання входів і виходів.

Мікросхеми КР1533ИР24 (SN74ALS299) та КР1533ИР29 є універсальними регістрами зсуву, що забезпечують практично всі можливі режими роботи.

Умовне позначення мікросхем приводиться на рис. 7.25. Суттєва їх особливість полягає в тому, що запис та зчитування інформації забезпечується по одній шині вводу / виводу.

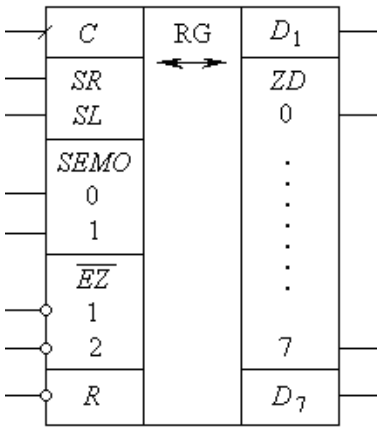


Рис. 7.25

Таблиця станів мікросхеми (табл. 7.8) відображає взаємозв'язок між сигналами для різних режимів роботи мікросхеми. Установка тригерів регістра в **"0"** забезпечується в асинхронному режимі при подачі сигналу низького рівня на вхід $\overline{R}$ незалежно від рівня інших сигналів. Решта режимів забезпечується при високому рівні сигналу на вході $\overline{R}$.

Таблиця 7.8

Режим	Входи								Вхід / вихід								Виходи	
	$\overline{R}$	*SEMO*		*EZ*		*C*	*SL*	*SR*	*ZD*								D_0	D_7
		0	1	$\overline{1}$	$\overline{2}$				0	1	2	3	4	5	6	7		
Установка **"0"**	L	L	x	L	L	x	x	x	L	L	L	L	L	L	L	L	L	L
	L	x	L	L	L	x	x	x	L	L	L	L	L	L	L	L	L	L
	L	H	H	x	x	x	x	x	x	x	x	x	x	x	x	x	L	L
Зберігання	H	L	L	L	L	x	x	x	$d_0 \ldots d_7$								d_0	d_7
	H	x	x	L	L	L	x	x	$d_0 \ldots d_7$								d_0	d_7
Зсув вправо	H	H	L	L	L	⌡	x	H	H	$d_0 \ldots d_6$							H	d_6
	H	H	L	L	L	⌡	x	L	L	$d_0 \ldots d_6$							L	d_6
Зсув вліво	H	L	H	L	L	⌡	H	x	$d_1 \ldots d_7$							H	d_1	H
	H	L	H	L	L	⌡	L	x	$d_1 \ldots d_7$							L	d_1	L
Завантаження	H	H	H	x	x	⌡	x	x	$d_0 \ldots d_7$								d_0	d_7

Завантаження інформації у паралельному форматі забезпечується при високому рівні сигналу на входах вибору режиму $SEMO_0$ і $SEMO_1$ за фронтом синхросигналу *C*-. При цьому інформація, що була попередньо подана на входи / виходи $ZD(0\ldots7)$, завантажується в тригери регістра. Перехід шини входу / виходу у високоімпедансний стан забезпечується умовою $EZ_1 + EZ_2 = 1$, тобто високим рівнем сигналу на будь-якому з входів дозволу стану високого імпедансу. Режим зберігання інформації у тригерах регістра забезпечується при виконанні умови: $\overline{SEMO} \cdot \overline{EZ_1} \cdot \overline{EZ_2} + \overline{C} \cdot \overline{EZ_1} \cdot \overline{EZ_2} = 1$.

Звертаємо увагу на той факт, що керування регістром у відповідності до першої кон'юнкції забезпечує високу універсальність та гнучкість при переході з одного режиму до іншого. Зміна рівня сигналу на одному з входів вибору режиму, тобто $SEMO_0 \cdot \overline{SEMO_1}$ або $\overline{SEMO_0} \cdot SEMO_1$, приводить до переходу в режим зсуву вправо або вліво, задавши лише рівень сигналу на тригері, що звільнюється по входу зсуву вправо *SR* або входу зсуву вліво *SL*. Ці входи використовуються при нарощуванні розрядності регістрів, при введенні інформації, заданої у послідовному форматі, одночасно з виходами D_1 і D_7.

Приклад 7.5. Використовуючи мікросхему ИР24, розробити схему універсального регістра для обробки 16-розрядних слів.

Пояснення. При нарощуванні розрядності регістрів входи *SEMO*, $\overline{EZ}$, *C* і *R* з'єднуються паралельно. Вхід *SR* першого регістра залишається послідовним входом обох мікросхем. Вихід D_7 першого регістра з'єднується з входом *SR* другого.

Для забезпечення зсуву вліво необхідно виконати аналогічні з'єднання. Вхід *SL* першого регістра залишається вільним. Вихід D_1 першого регістра з'єднується з входом *SL* другого.

Серед регістрів інших серій слід виділити ІМС 564ИР13, 155ИР17, призначені для використання в приладах цифро-аналогового перетворення, а також 564ИР6, з широкими функціональними можливостями. Остання ІМС є восьмирозрядним регістром, що дозволяє забезпечити виконання наступних операцій: перетворення послідовної форми представлення інформації в паралельну та передача останньої у будь-який з двох каналів; зберігання інформації, виведення її у паралельному форматі; прийом паралельної інформації від будь-якого з двох каналів та перетворення її у послідовний формат. На рис. 7.26, *а* наведене умовне зображення мікросхеми 564ИР6, а на рис. 7.26, *б* – її функціональна схема.

Регістр складається з логічної схеми керування та восьми модулів пам'яті, кожен з яких містить двоступінчатий *D*-тригер та керуючий мультиплексор. Кожен модуль забезпечує запам'ятовування інформації, а також двонаправлену її передачу від шини ***A*** до шини ***B*** і навпаки. Виводи шини ***A*** мають *Z*-стан, який встановлюється при нульовому потенціалі на керуючому вході *AE*. При *AE* = 1 шини ***A*** переходять в активний стан та можуть як приймати, так і передавати двійкову інформацію, подану у паралельному форматі.

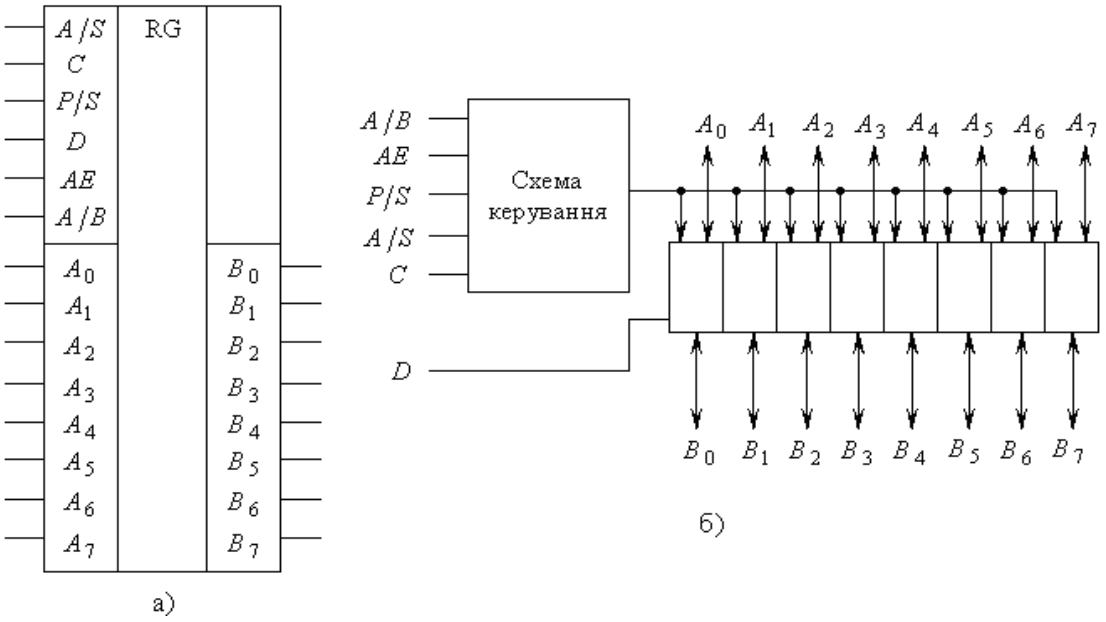


Рис. 7.26

Для прийому регістром інформації, поданої у послідовному форматі, служить вхід *D*-. Режимам асинхронний / синхронний (*A*/*S*), паралельному / послідовному (*P*/*S*) способам введення інформації, забезпеченню напрямку передачі інформації від ***A*** до ***B*** або навпаки відповідають логічні значення керуючих сигналів 1/0. Це означає, що високий рівень сигналу на відмічених керуючих входах відповідає: *A*/*S* = 1 – асинхронний обмін інформацією; *P*/*S* = 1 – паралельний спосіб введення/виведення; *A*/*B* = 1 – передача інформації забезпечується від шини ***A*** до шини ***B*** (шина ***A*** – введення; шина ***B*** – виведення). Протилежні значення рівнів сигналів забезпечують виконання зворотних операцій. Режими роботи регістра у залежності від станів входів відображені у табл. 7.9.

При нарощуванні розрядності регістрів на мікросхемі 561(564)ИР6 керуючі входи запаралелюються. Для організації послідовного введення вихід B_7 регістру молодших розрядів з'єднуються з *D*-входом регістру старших розрядів, а послідовне виведення інформації здійснюється з виходу B_7 старших розрядів. У тих випадках, коли окремі функції регістром не виконуються, відповідні керуючі входи підключаються до напруги джерела живлення або до загальної шини.

Таблиця 7.9

Входи				Режим роботи
AE	P/S	A/B	A/S	
0	0	X	X	Послідовне синхронне введення Шини А і В відключені
0	1	0	0	Паралельне синхронне введення через шину В Щина А відключена
0	1	0	1	Паралельне асинхронне введення через шину В Щина А відключена
0	1	1	X	Зберігання інформації Виходи Аі В відключені
1	0	0	1	Послідовне синхронне введення Вихід – шина А
1	0	1	1	Послідовне синхронне введення Вихід – шина В
1	1	0	0	Паралельне синхронне введення Входи – шина В, виходи шина - А
1	1	0	1	Паралельне асинхронне введення Входи – шина В, виходи шина - А
1	1	1	0	Паралельне синхронне введення Входи – шина А, виходи шина - В
1	1	1	1	Паралельне асинхронне введення Входи – шина А, виходи шина - В

Описаний регістр придатний для використання у цифрових приладах різного функціонального призначення. На рис. 7.27 наведений варіант його застосування в якості фазового компаратора.

На виході компаратора з’явиться напруга, відповідна **лог. “1”**, якщо $f_1 < f_2$, а **лог. “0”**, якщо $f_1 > f_2$. При $f_1 = f_2$ на виході присутній симетричний меандр.

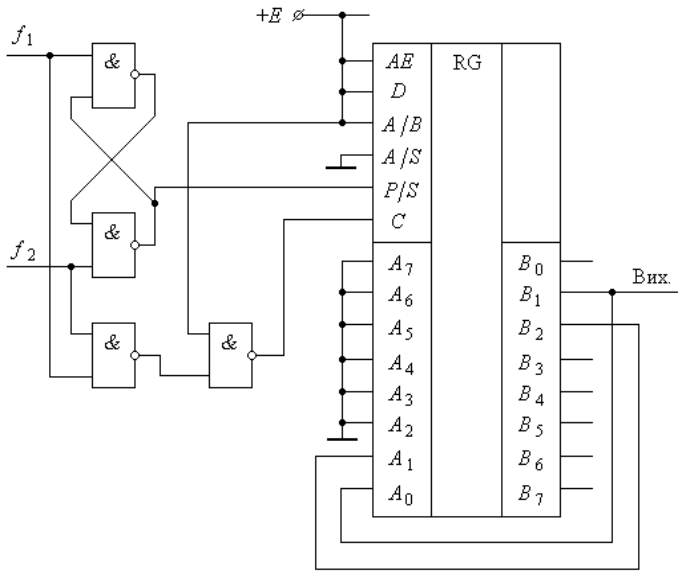


Рис. 7.27

Компаратор зручний для використання у цифрових системах з фазовою автопідстройкою.

Основне призначення мікросхеми – організація магістрального обміну інформацією.

У найпростішому випадку такого обміну ІМС використовується як буферний регістр між абонентом та магістраллю (рис. 7.28, *а*). Однак у цьому випадку необхідно враховувати специфіку регістра, яка полягає в тому, що порт (шина) ***B*** у режимі запису з боку магістралі *L* стає передаючим і не може бути ввімкненим подібно порту ***A***.

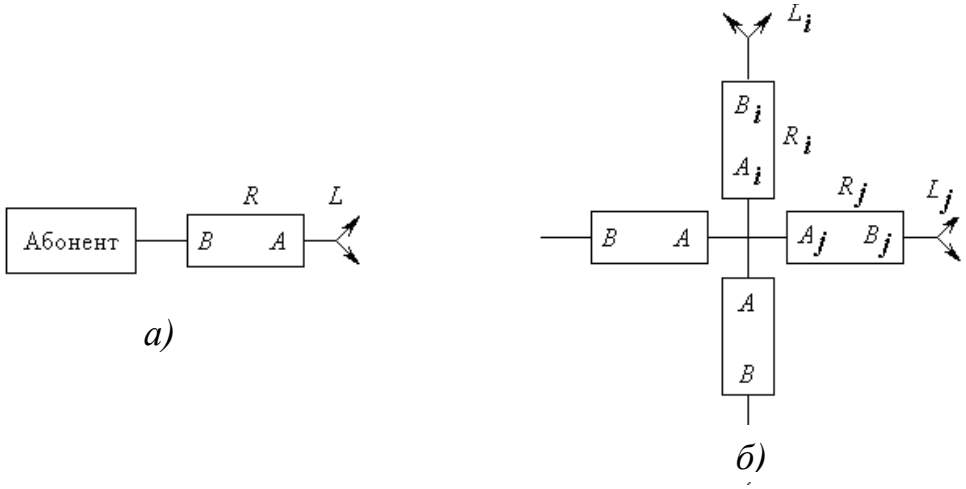


Рис. 7.28

Це призводить до того, що абонент ***A*** при записі в регістр *R* з боку магістралі *L* повинен або синхронно перемикатися на прийом, або мати можливість відмикатися від порту ***B***, переходячи самостійно у *Z*-стан.

Найбільш повно функціональні можливості ІМС реалізуються у різних *магістральних комутаторах*, варіант одного з яких зображений на рис. 7.28, *б*. Робота наведеного радіального комутатора полягає в наступному. Припустимо, що виникла необхідність передачі даних від абонента, розташованого на магістралі L_i, до абонента, підключеного до магістралі L_j. Передача починається із запису інформації в регістр R_i через порт ***B***. Протягом цієї операції порт ***A*** перебуває у *Z*-стані. Після закінчення операції запису абонент магістралі L_i відключається від регістра R_i. Далі здійснюється синхронний

перезапис інформації з регістра R_i у регістр R_j через відповідні порти A_i та A_j, після чого порти ***A*** обох регістрів відмикаються. Режим роботи регістра R_j задається відповідними керуючими сигналами, поданими на його входи, тому він може як запам'ятовувати прийняту інформацію, так і передавати її в магістраль L_i транзитом.

Мікросхема 564ИР1 є своєрідним регістровим файлом. Вона складається з чотирьох окремих секцій, поєднаних загальною динамічною синхронізацією за зрізом синхроімпульсу (рис. 7.29).

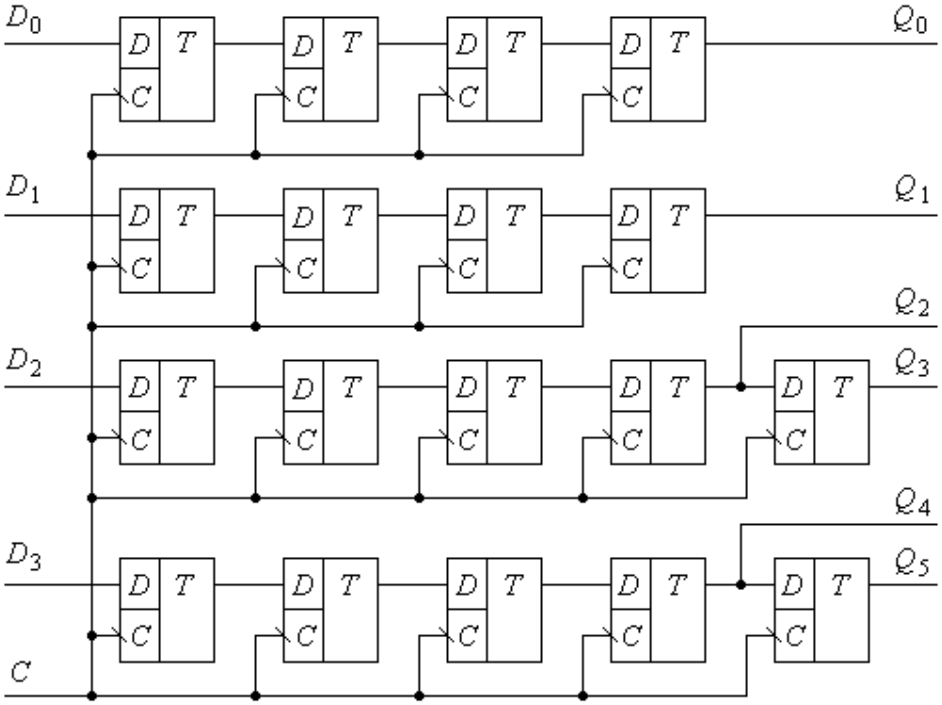


Рис. 7.29

Кожна з секцій являє собою 4(5)-розрядний регістр зсуву з послідовним введенням і виведенням інформації. Шляхом зовнішніх з'єднань входів $D_0 \ldots D_3$ і виходів $Q_0 \ldots Q_5$ можна забезпечити організацію регістрів зсуву, починаючи з чотирьох розрядів до вісімнадцяти у відповідності до дискретності секцій. Нарощуванням таких регістрів можна отримати регістрові схеми із значно більшою кількістю розрядів. Мікросхема використовується для створення пристроїв часової затримки, а також для забезпечення дискретного ділення частоти вхідної послідовності імпульсів.

7.6. Напрямки (області) використання регістрів

Регістри пам’яті знаходять широке використання в арифметичних пристроях, у пристроях пам’яті. Основні області використання приводились вище. Регістри зсуву мають ширше практичне використання. Напрямки використання – запис та зчитування інформації у послідовному та паралельному форматах, перетворювання послідовного коду у паралельний і навпаки, зсув вправо та вліво – були розглянуті вище. Розглянемо деякі інші можливості використання регістрів зсуву.

7.6.1. Забезпечення обміну інформацією у послідовному форматі

Задача забезпечення обміну інформацією у послідовному форматі є досить актуальною, оскільки у цьому форматі передається більшість цифрової інформації.

Вхідний регістр DD1 (рис. 7.30) забезпечує перетворення інформації, поданої в паралельному форматі, у послідовний формат і з виходу Q_3 передає її в лінію зв’язку. Вихідний регістр DD2 виконує зворотне перетворення. У регістр-приймач необхідно передавати не тільки послідовний код інформації, що передається, а також і синхроімпульси, необхідні для керування зсувом на приймальній стороні лінії зв’язку. Така система зв’язку є *однофазною трипровідною*, і їй властиві недоліки однофазних цифрових систем.

Найсерйозніший недолік полягає у розфазуванні сигналів, що передаються.

Забезпечення стійкості до розфазування досягається різними шляхами. Один з них полягає в застосуванні вирівнюючого регістру, що вмикається перед входом регістра-приймача DD2.

Особливість використання вирівнюючого регістра наступна. Якщо інформація, що передається, відновлюється за зрізом синхроімпульсу *С*-, то в якості вирівнюючого використовується регістр, що перемикається за фронтом, який виставляє на своєму виході інформацію за онтом синхросигналу

лінії зв’язку, тобто інформацію, яка з гарантією відноситься саме до даного такту. Можливі й інші засоби синхронізації.

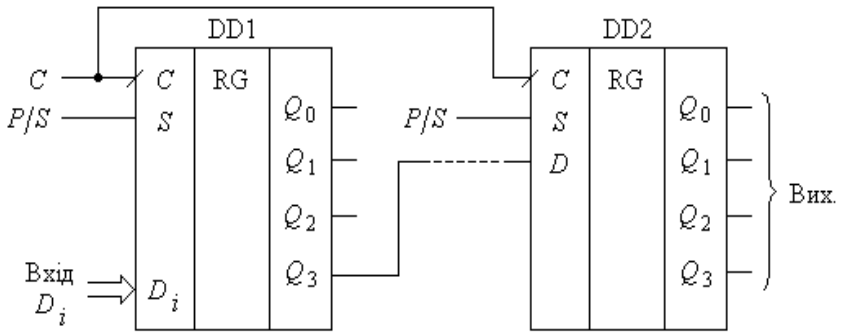


Рис. 7.30

Також використовуються й асинхронні способи послідовної передачі даних, в яких синхросигнал не передається по окремому провіднику, а за рахунок спеціального кодування передається по тому ж каналу, що й дані, розмежовуючись з ними у часі. Канал зв’язку спрощується за рахунок зниження швидкодії інформаційного обміну і ускладнення передавальної та приймальної апаратури. Деякі аспекти такої передачі розглядались у попередньому розділі. Такий канал зв’язку є *двопровідним* (в практиці зв’язку називається *однопровідним*, оскільки один із провідників – загальна шина).

Використовуються різні способи такої однопровідної передачі. За першим з них синхронізуючий перепад супроводжує кожен біт інформації (*фазове* і *частотне кодування*). За іншим синхронізуючий перепад, який зветься стартовим бітом, супроводжує один байт. Це стандартний послідовний інтерфейс, який використовується у COM-портах ЕОМ, в USB-шині. У послідовному форматі забезпечується запис інформації на жорсткі носії. Для передачі та прийому інформації у послідовному форматі випускаються спеціальні мікросхеми. У більшості мікроконтролерів і мікропроцесорів для обробки інформації вмонтовується спеціальний пристрій – *UART* (*Universal Asynchronous Receiver / Transmitter*). Передача інформації у послідовному форматі з підвищенням швидкодії лінії зв’язку стає домінуючою в промислових системах автоматики.

7.6.2. Регістрові лічильники імпульсів (*розподілювачі*)

Розподілювачами називають функціональні вузли, які циклічно розподіляють потік імпульсів по декількох каналах. Прикладом розподілювача може служити схема двійкового лічильника, на вхід якого подаються імпульси для підрахунку, а виходи приєднані до дешифратора. Збільшуваний або зменшуваний вихідний двійковий код лічильника дешифрується на ряд каналів, створюючи в кожному з них послідовність імпульсів, що задається циклом роботи схеми. Аналіз такої схеми без урахування динамічних процесів показує, що створювані імпульсні послідовності можуть бути симетричними, якщо лічильник має модуль 2^m. Якщо ж модуль перерахунку не відповідає такій кратності, то не всі вихідні послідовності матимуть симетричний вид. При врахуванні динамічних режимів перемикання лічильника на виході дешифратора виникають короткі сплески напруги – гонки, які ще більше спотворюють вихідні послідовності.

Простіше функцію розподілювача може виконувати регістр зсуву, якщо лише в один його розряд записати одиничний сигнал. При подачі синхроімпульсів записана одиниця переміщуватиметься по розрядах регістра, відображаючи на лінійній шкалі кількість поданих синхроімпульсів (наприклад, загоряння лампочки номеру поверху в ліфті). Значно більші можливості мають регістри зсуву, що з'єднані в кільце, а точніше, мають зворотний зв'язок з виходів на вхід. У літературі такі схеми, у залежності від характеру зворотних зв'язків, називають *кільцевими лічильниками*, *лічильниками Джонсона*, *лічильниками Мебіуса*, *лічильниками в коді Лібау-Крейга*, *кільцями Реженера*.

Загальна структура кільцевого лічильника з використанням регістра зсуву приведена на рис. 7.31.

Особливість її полягає в тому, що допоміжно вона має комбінаційну схему, яка обробляє вихідні сигнали кожного тригера і генерує черговий сигнал на вхід регістра молодших розрядів: $Y = f(Q_0 \ldots Q_N, \overline{Q_0} \ldots \overline{Q_N})$.

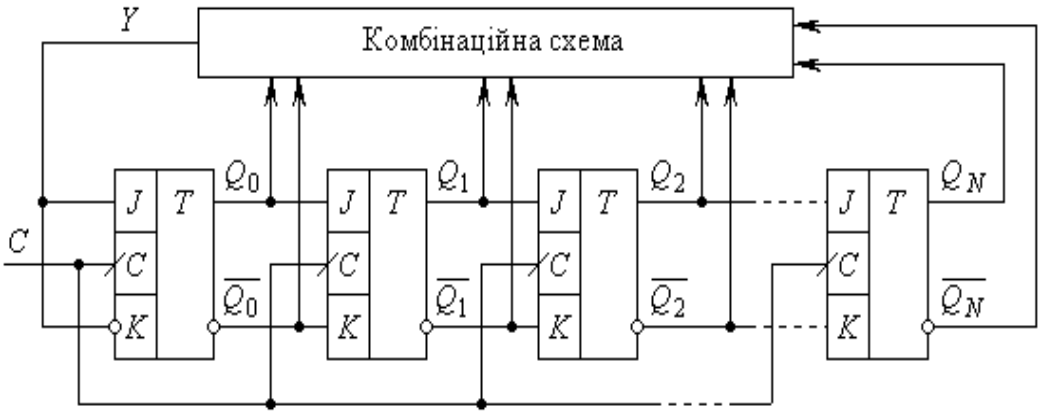


Рис. 7.31

Тому можемо взаємозв’язок між входами/виходами кожного розряду зобразити у вигляді систем рівнянь:

$$Q_{0\,(n+1)} = f(Q_0 \ldots Q_N, \overline{Q_0} \ldots \overline{Q_N})_n ;$$

$$Q_{1\,(n+1)} = Q_{0n} ;$$

……

$$Q_{N\,(n+1)} = Q_{(N-1)\,n} .$$

Ланка зворотного зв’язку формує сигнал **0** або**1**, задаючи тим самим черговий стан схем лічильника.

У загальному плані аналіз таких схем досить складний, тому розглянемо ряд конкретних широко використовуваних схем лічильників. Найпростіший варіант зворотного зв’язку має схема кільцевого лічильника, в якій виходи тригера старших розрядів безпосередньо з’єднані з входами тригера молодших розрядів (рис. 7.32).

Схема доповнена входом сигналу початкової установки *S*, високий рівень на якому встановлює тригер DD0 в одиничний стан, а решту – в нульовий. Часова діаграма роботи лічильника при подачі серії синхроімпульсів матиме вигляд, приведений на рис. 7.33. Записана в DD0 одиниця переміщується в черговий розряд (старший або молодший у залежності від напрямку зсуву) і затримується в ньому на інтервал часу, що дорівнює періоду синхронізуючого імпульсу T_C. Решта тригерів знаходиться в нульовому стані.

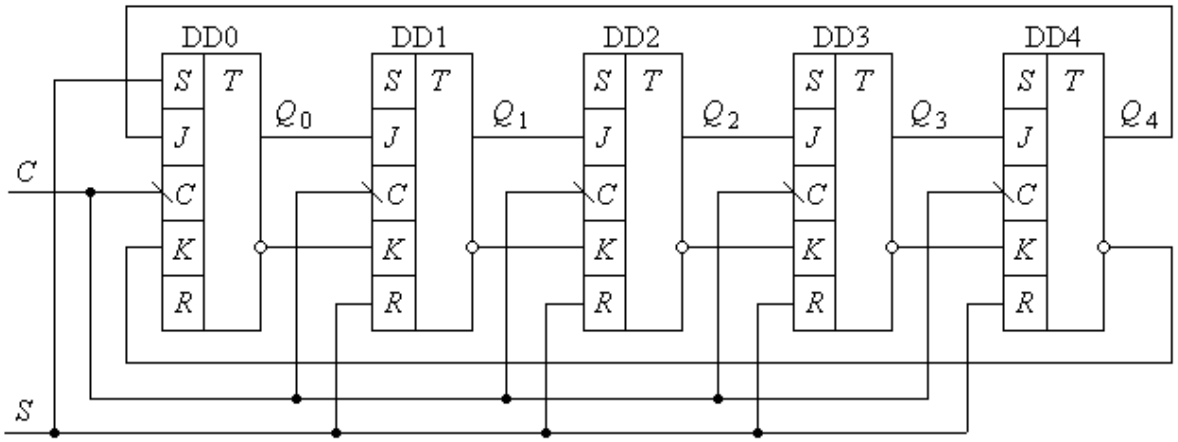


Рис. 7.32

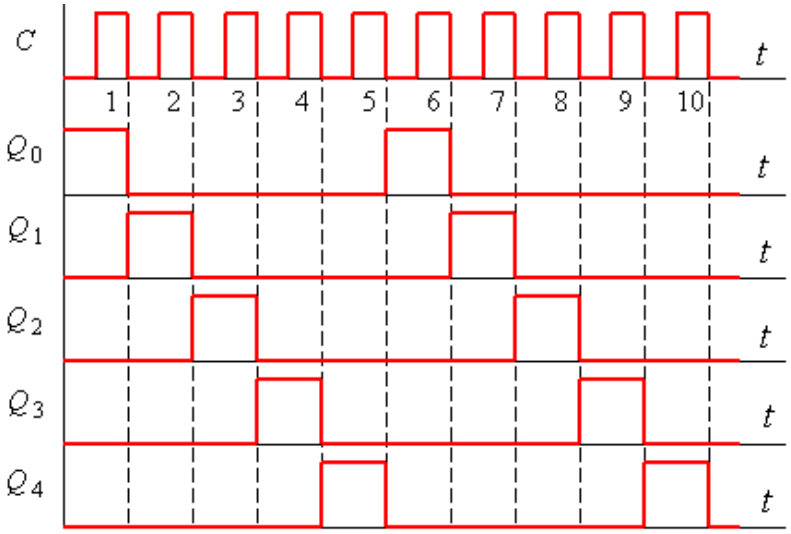


Рис. 7.33

При використанні серійних регістрів слід враховувати, що при вмиканні живлення тригери регістра можуть встановлюватись довільним чином, тому необхідно мати можливість попередньої установки лічильника по входу *S*.

Розглянута схема може повністю замінити описану вище пару “двійковий лічильник - дешифратор” при перетворенні кількості вхідних імпульсів у код, що задається кількістю тригерів регістра. Вихідна частота імпульсів по будь-якому з виходів дорівнює $f_{вих} = f_{вх}/m$, де m – кількість тригерів. Оскільки в схемі, що розглядається, відсутні зовнішні логічні елементи, то кільцеві лічильники мають високу швидкодію. Але одним з недоліків кільцевих лічильників є велика кількість використовуваних тригерів.

Розглянута схема має досить широке використання. Так вона може виконувати функцію розподілення імпульсів по декількох каналах, кількість яких рівна *m*.

Функції розподілення широко використовуються в керуванні пристроями енергетичної електроніки, керування кроковими двигунами. За допомогою кільцевих лічильників і допоміжної логіки можна створювати пристрої автоматичного формування сигналів тривоги, оповіщення та ін.

Приклад 7.6. Розробити схему, яка б автоматично і циклічно генерувала сигнал типу, що зображений на рис. 7.34.

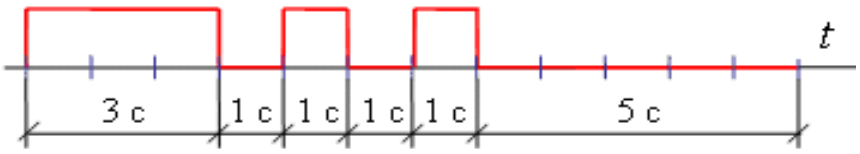


Рис. 7.34

Розв'язання.

1. Підраховуємо цикли повторення сигналу. Він складає 12 секунд.

2. Вибираємо зсувний регістр, за допомогою якого можемо створити кільцевий лічильник з кількістю тригерів 12, що має виходи з прямого виходу для тригера. В якості такого регістра вибираємо набір *D*-тригерів КР1533ИР23.

3. Вибираємо дві мікросхеми і з'єднуємо в кільце 12 тригерів. Розробляємо схему початкової установки тригерів у регістрі. В якості такої можемо вибрати елементи **2І**, що входами приєднуються до виходів $Q_0 \ldots Q_{11}$.

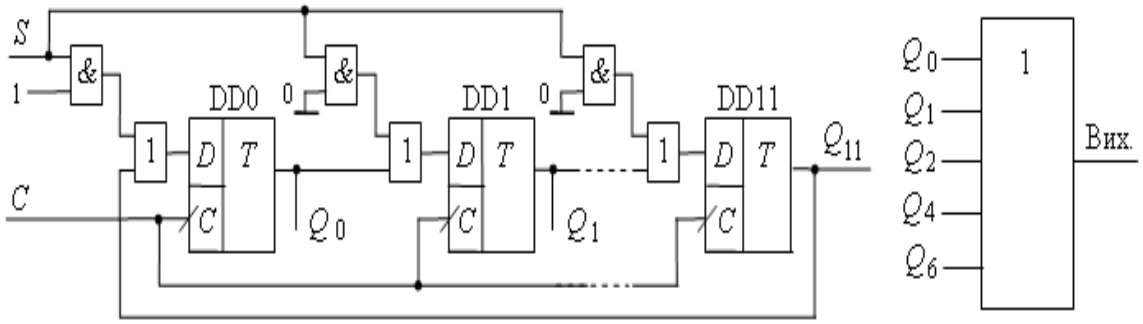


Рис. 7.35

Фрагмент схеми приводиться на рис. 7.35. Перед запуском усі тригери, за винятком першого, встановлюються в нуль. При подачі періодичної послідовності імпульсів на вхід *C*-

з періодом 1 секунда записана у перший тригер одиниця перемішуватиметься по кільцю. Сигнал, що знімається з виходів Q_0, Q_1, Q_2, Q_4, Q_6, на ЛЕ **АБО** дасть на виході необхідну імпульсну послідовність.

Схема кільцевого лічильника може бути розроблена і з самозапуском. Ідея самозапуску полягає в тому, що одиничний сигнал на вхід першого тригера повинен бути поданий при умові, коли всі виходи послідуючі тригерів знаходяться в нульовому стані, тобто:

$$D_0 = \overline{Q_1}\,\overline{Q_2}\,\overline{Q_3}\ldots\overline{Q_{11}}.$$

Це означає, що замість безпосереднього зворотного зв'язку необхідно подати сигнал на вхід D_0, об'єднавши всі інверсні виходи тригерів через елемент **І**. Якщо після вмикання схеми на деяких тригерах будуть встановлені одиничні рівні, то через декілька тактів вони заміняться нулями, після чого в перший триггер буде записаний сигнал високого рівня.

Звернемось знову до недоліків кільцевого лічильника. Другий його недолік полягає у неможливості виведення інформації у двійковому коді. Окрім цього, його послідовність відліків може значно змінитись, якщо внаслідок хибного спрацювання схеми він перейде в один із станів, що не використовується. Акцентуємо на цьому більш детально нашу увагу.

Коли ми розглядали двійкові лічильники, то визначились, що його граф-схема має 2^m станів, і для чотирьохрозрядного лічильника їх було $2^4 = 16$. Якщо порівняти, наприклад, два десятирозрядні лічильники – двійковий і кільцевий, то можемо сказати, що двійковий матиме $2^{10} = 1024$ стани, а кільцевий, який працює в коді «1 із 10», має 10 робочих станів в циклі. Звідси витікає, що кільцевий лічильник має (1024 – 10) = 1014 заборонених до використання станів. Отже, ймовірність можливого переходу кільцевого лічильника в один з заборонених станів досить висока, тому внаслідок дії завад він може взагалі не перейти до правильної послідовності відліку без сторонньої допомоги.

При роботі кільцевого лічильника можливі дві основні помилкові ситуації. Перша з них полягає в тому, що внаслідок дії завад у будь-якому з тригерів може з'явитись ще один сигнал логічної одиниці, який циркулюватиме у схемі. Друга обумовлена переходом у нуль (скиданням) єдиної одиниці в кільці.

Тому, щоб забезпечити працездатність лічильника, його необхідно доповнити допоміжними засобами, які повинні виявляти зайву одиницю, а також виявляти ситуацію, при якій одиниця в тригерах лічильника взагалі не існує. Такі задачі легко розв'язуються для простих лічильників. Наприклад, для лічильника з $m = 4$ і виходами Q_0, Q_1, Q_2, Q_3 функція виявлення зайвої одиниці матиме вигляд:

$$y_1 = Q_0\,Q_1 + Q_0\,Q_2 + Q_0\,Q_3 + Q_1\,Q_2 + Q_1\,Q_3 + Q_2\,Q_3 =$$
$$= Q_0\,(Q_1 + Q_2 + Q_3) + Q_1\,(Q_2 + Q_3) + Q_2\,Q_3\,.$$

Використання такої функції для забезпечення попередньої установки дає можливість відреагувати і виправити помилку на першому такті. Якщо ж використовувати більш прості функції, котрі не мають повного перебору кон'юнкцій, то швидкодія виправлення помилки може затягнутись на декілька тактів. Для того, щоб визначити ситуацію, коли всі тригери встановлені у нуль, можна використати функцію:

$$y_0 = \overline{Q_0 + Q_1 + Q_2 + Q_3} = \overline{Q_0}\;\overline{Q_1}\;\overline{Q_2}\;\overline{Q_3}\,.$$

7.6.3. Лічильники Джонсона

Недолік кільцевого лічильника, обумовлений великою кількістю тригерів, потрібних для реалізації необхідного коефіцієнта перерахунку, може бути усунений, якщо один із зв'язків між тригерами зробити перехресним, тобто вхід одного з тригерів з'єднати з інверсним виходом попереднього. При цьому принципово не має значення той факт, між якими з тригерів установлений перехресний зв'язок (*лічильник Мебіуса*).

Після установки всіх тригерів у нульовий стан на вході першого тригера матимемо сигнал логічної одиниці, який з кожним синхроімпульсом передаватиметься вправо, не зникаючи в попередньому розряді до заповнення всіх розрядів. За рахунок зворотного зв'язку у наступному циклі в лічильнику нарощуватиметься кількість нулів.

На рис. 7.36 приводиться приклад такого лічильника, що має п’ять тригерів.

Прийнявши, що в нульовому стані всі тригери лічильника встановлюються в нуль, отримаємо послідовність відліку, що приводиться у табл. 7.10.

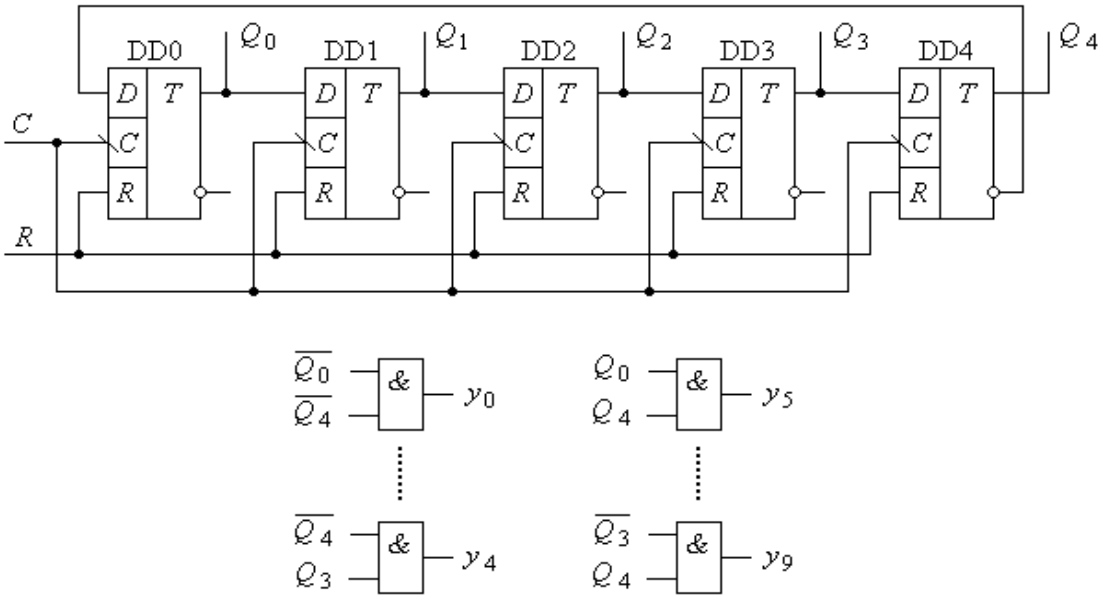


Рис. 7.36

Отримана послідовність має 10 станів, які можна вибрати з табл. 7.10. Логічна схема ланки зворотного зв’язку *DO* може бути отримана таким чином: у стовпець таблиці *DO* заносяться необхідні значення сигналу *DO* для забезпечення наступного стану.

В результаті отримуємо: $DO = \bigvee_{0}^{9} 0, 1, 2, 3, 4.$

Переходячи до логічних змінних $Q_0 \ldots Q_4$ й мінімізуючи їх за допомогою карт Карно (рис. 7.37), отримуємо: $DO = \overline{Q_4}$, що підтверджує відповідність схеми рис. 7.36 і табл. 7.10.

Організація десяткового відліку може бути забезпечена допоміжною логікою, яка призначена для декодування станів лічильника. Для визначення і мінімізації логіки декодування станів карти Карно, що зображені на рис. 7.37, дещо змінимо, вписавши до них номери станів (рис. 7.38).

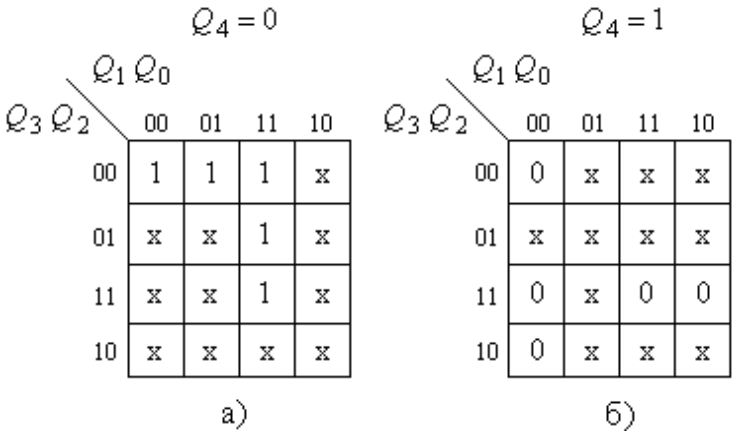


Рис. 7.37

Таблиця 7.10

N тактового імпульсу	Q_0	Q_1	Q_2	Q_3	Q_4	Спосіб дешифрації стану	*DO*
0	0	0	0	0	0	$\overline{Q_0}\cdot\overline{Q_4}=\overline{Q_0\vee Q_4}$	1
1	1	0	0	0	0	$Q_0\cdot\overline{Q_1}=\overline{\overline{Q_0}\vee Q_1}$	1
2	1	1	0	0	0	$Q_1\cdot\overline{Q_2}=\overline{\overline{Q_1}\vee Q_2}$	1
3	1	1	1	0	0	$Q_2\cdot\overline{Q_3}=\overline{\overline{Q_2}\vee Q_3}$	1
4	1	1	1	1	0	$Q_3\cdot\overline{Q_4}=\overline{\overline{Q_3}\vee Q_4}$	1
5	1	1	1	1	1	$Q_4\cdot Q_0=\overline{\overline{Q_4}\vee\overline{Q_0}}$	0
6	0	1	1	1	1	$\overline{Q_0}\cdot Q_1=\overline{Q_0\vee\overline{Q_1}}$	0
7	0	0	1	1	1	$\overline{Q_1}\cdot Q_2=\overline{Q_1\vee\overline{Q_2}}$	0
8	0	0	0	1	1	$\overline{Q_2}\cdot Q_3=\overline{Q_2\vee\overline{Q_3}}$	0
9	0	0	0	0	1	$\overline{Q_3}\cdot Q_4=\overline{Q_3\vee\overline{Q_4}}$	0

Використовуючи відомі засоби мінімізації, для функцій, що залежить від Q_4, маємо:

$$y_0 = \overline{Q_0}\;\overline{Q_4}\,; \qquad y_5 = Q_0\,Q_4\,;$$
$$y_4 = Q_3\,\overline{Q_4}\,; \qquad y_9 = \overline{Q_3}\,Q_4\,.$$

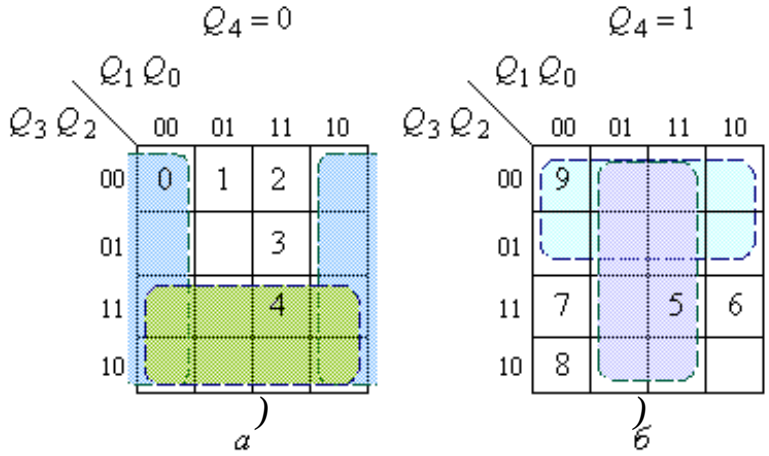


Рис. 7.38

Аналогічно отримуються і функції, які не залежать від Q_4. Результати мінімізації представлені у табл. 7.10.

Для лічильника Джонсона, що розглядається, існують також три послідовності, відповідно до яких він може змінювати свої стани після дії перешкод:

1. $S_2 \to S_5 \to S_{11} \to S_{23} \to S_{14} \to S_{29} \to S_{26} \to S_{20} \to S_8 \to S_{17} \to S_2$;

2. $S_4 \to S_9 \to S_{19} \to S_6 \to S_{13} \to S_{27} \to S_{22} \to S_{12} \to S_{25} \to S_{18} \to S_4$;

3. $S_{10} \to S_{21} \to S_{10}$.

Вказані послідовності можуть бути відображені за допомогою карт Карно (рис. 7.39), цифри на яких відповідають вказаним послідовностям.

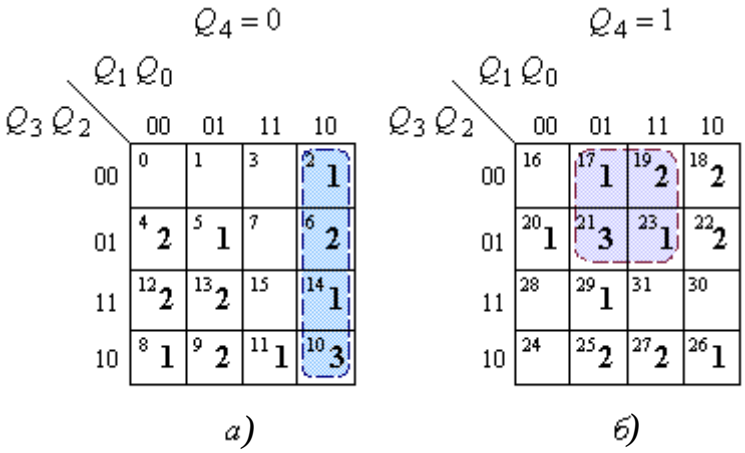


Рис. 7.39

З карти, що приведена на рис. 7.39, *а*, бачимо, що чотири суміжні клітини зі станами S_2, S_6, S_{14}, S_{10} охоплюють стани всіх небажаних послідовностей. Такою ж групою клітин на рис. 7.39, *б* є клітки з номерами S_{17}, S_{19}, S_{21}, S_{23}. Функції, що описують ці групи кліток, мають вигляд:

$$f_1 = \overline{Q_4}\, Q_1\, \overline{Q_0}\,; \qquad f_2 = Q_4\, \overline{Q_3}\, Q_0.$$

Є і інші варіанти таких функцій. Якщо використати одну з таких функцій для установки всіх тригерів, то через один цикл роботи лічильник відновить правильну послідовність зміни станів.

Лічильник Джонсона має парну довжину циклу відліку, яка дорівнює $2m$. Але, як ми бачили з аналізу роботи лічильника і табл. 7.10, можна змінити

функцію зворотного зв'язку з тим, щоб зменшити величину коефіцієнта перерахунку – наприклад, зробити її непарною. Такі задачі читачам пропонується розв'язати самостійно.

Приклад типової задачі. Знайти таку функцію зворотного зв'язку *DO*, щоб лічильник не мав станів **00000** і **11111**.

Деякі мікросхеми лічильників побудовані на основі схеми Джонсона (наприклад, 564ИЕ8, 564ИЕ9).

На рис. 7.40 приводиться функціональна схема одного з таких лічильників, в якому функція зворотного зв'язку має вигляд: $D_0 = Q_2 \oplus Q_3$.

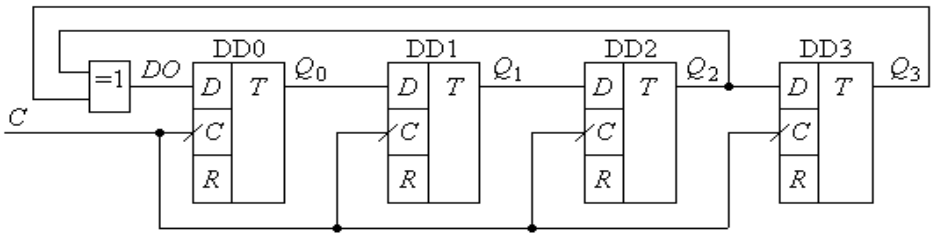


Рис. 7.40

Розглянемо особливості роботи лічильника. Оскільки на вхід *DO* не поступатимуть інформаційні сигнали при $Q_3\,Q_2\,Q_1\,Q_0 = 0000 = S_0$, то задамо початкову умову: $Q_3\,Q_2\,Q_1\,Q_0 = 0001 = S_1$.

У цьому випадку $DO = Q_2 \oplus Q_3 = 0 \oplus 0$ і при появі тактового імпульсу отримаємо новий стан $0010 = S_2$.

У табл. 7.11 приведена послідовність зміни станів протягом одного циклу. У циклі маємо 15 різних станів, за винятком стану $S_0 = 0000$, при якому лічильник є непрацездатним. Тобто максимальна довжина послідовності станів $l_д = 2^m - 1$. *Послідовність максимальної довжини* (ПМД) є однією з характеристик лічильника, що розглядається. Тільки конкретно визначені зв'язки функції зворотного зв'язку можуть дати ПМД.

Приклад типової задачі. Визначити максимальну довжину циклу, якщо в схемі, приведеній на рис. 7.40, одним з входів суматора за модулем **2** взяти не Q_2, а Q_1. Побудувати таблицю станів.

У табл. 7.12 ПМД приводяться вирази для функції зворотного зв’язку, використання яких дозволяє сформувати ПМД для деяких лічильників з кількістю розрядів від 1 до 18 [Голдсуорт].

Таблиця 7.11

Номер тактового імпульсу	S	Q_3	Q_2	Q_1	Q_0	DO
	S_1	0	0	0	1	0
2	S_2	0	0	1	0	0
3	S_4	0	1	0	0	1
4	S_9	1	0	0	1	1
5	S_3	0	0	1	1	0
6	S_6	0	1	1	0	1
7	S_4	1	1	0	1	0
8	S_{10}	1	0	1	0	1
9	S_5	0	1	0	1	1
10	S_{11}	1	0	1	1	1
11	S_7	0	1	1	1	1
12	S_{15}	1	1	1	1	0
13	S_{14}	1	1	1	0	0
14	S_{12}	1	1	0	0	0
15	S_8	1	0	0	0	1
16	S_1	0	0	0	1	0

Таблиця 7.12

Кількість розрядів m	Рівняння зворотного зв’язку
1	Q_0
2	$Q_0 \oplus Q_1$
3	$Q_1 \oplus Q_2$
4	$Q_2 \oplus Q_3$
5	$Q_2 \oplus Q_4$
6	$Q_4 \oplus Q_5$
7	$Q_5 \oplus Q_6$
10	$Q_6 \oplus Q_8$
11	$Q_9 \oplus Q_{10}$
17	$Q_{13} \oplus Q_{16}$
18	$Q_{10} \oplus Q_{17}$

Можливо також створити інші ПМД, використовуючи не тільки двовходові логічні елементи **ВИКЛ. АБО**, а й ЛЕ з більшою кількістю входів.

Розглянута схема може бути використаною для генерації двійкових послідовностей. Причому елементи послідовності можуть формуватися безпосередньо на виході одного з тригерів. Статистичні характеристики послідовності одиниць та нулів, отримуваних з виходу будь-якого тригера, близькі до характеристик випадкової послідовності і тим ближче, чим більшої довжини регістр. Такі лічильники називають *генераторами псевдовипадкових послідовностей*.

У табл. 7.13 приводяться дані по кількості розрядів і типах зв’язків, які забезпечують вказані ПМД. Для більшості значень розрядності m існує декілька різних, але майже еквівалентних способів встановлення зворотних зв’язків для забезпечення ПМД.

Вихідна послідовність може бути знята з будь-якого тригера, оскільки статистично всі розряди еквівалентні. Паралельний псевдовипадковий код можна зняти одночасно з ряду тригерів.

Таблиця 7.13

Кількість розрядів m	$DO = Q_i \oplus Q_j$	$l_д$
3	2, 0	7
6	5, 4	63
10	9, 2	1 023
15	14, 0	32 767
16	15, 11, 8, 6	65 535
20	19, 2	1 048 575

Генератори псевдовипадкових послідовностей використовуються для імітації вхідних сигналів при діагностиці та налагодженні апаратури, цифровому моделюванні, у радіотехніці і радіолокації для підвищення захищенності зв'язку. Використовуючи пристрої аналого-цифрового перетворення, можна отримати аналоговий випадковий сигнал. Перемішуючи розряди тригерів, можна отримати ряд корельованих випадкових сигналів. Шуми квантування можуть бути відфільтровані фільтрами низьких частот.

7.6.4. Поліноміальні пристрої кодування та фільтрації

Поліноміальні пристрої кодування та фільтрації призначені для апаратної реалізації ряду операцій над поліномами, що мають бінарні коефіцієнти при ступінях змінної x. Такі операції дозволяють будувати, наприклад, циклічні коди, які широко використовують для виявлення і виправлення помилок при передачі та зберіганні даних.

У **Розділі 1** ми вже мали справу з поліномами, що мають бінарні коефіцієнти. Якщо записати такий поліном за зростаючими ступенями змінної x зліва направо і уявити, що бінарні коефіцієнти розміщені у регістрі зсуву, то, наприклад, для конкретного поліному $P = a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots + a_n x^n$ зсув коефіцієнтів на один розряд праворуч відповідатиме множенню поліному на x: $P \cdot x = x \cdot (a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots + a_n x^n)$, а, відповідно, зсув ліворуч – діленню поліному на x.

Якщо, наприклад, код значень коефіцієнтів деякого поліному дорівнює $P = 011101$, то код поліному $P \cdot x = 0011101$. Звідси витікає, що на основі

правил двійкової арифметики можна виконувати операцію множення поліномів.

Приклад 7.7. Виконати операцію множення двох поліномів:

$$P_1 = 1 + x^3\,; \qquad P_2 = 1 + x^2 + x^3\,.$$

Розв'язання. Визначимо коди значень коефіцієнтів в обох поліномах. Для цього обидва поліноми запишемо у вигляді: $P_1 = 1\cdot x^0 + 0\cdot x^1 + 0\cdot x^2 + 1\cdot x^3; P_2 = 1\cdot x^0 + 0\cdot x^1 + 1\cdot x^2 + 1\cdot x^3$.

В результаті маємо: $P_1 = 1001$; $P_2 = 1011$.

Множення поліномів з бінарними коефіцієнтами має свою особливість, яка полягає в тому, що показники ступеню множених членів додаються за правилами арифметики, а коефіцієнти при однакових ступенях x – за модулем **2**. Тому, наприклад, якщо потрібно знайти суму однакових ступеней, то маємо: $x^3 \oplus x^3 = x^3\,(1 \oplus 1) = 0 \cdot x^3 = 0$.

Тому процедура множення коефіцієнтів матиме вигляд:

			×	1	0	0	1
				1	0	1	1
				1	0	0	1
⊕		1	0	0	1		
	0	0	0	0			
1	0	0	1				
1	0	1	0	0	1	1	

В результаті поліном, що знаходиться як добуток двох поліномів, матиме вигляд:

$$P = P_1 \cdot P_2 = 1 + 0 \cdot x + 1 \cdot x^2 + 0 \cdot x^3 + 0 \cdot x^4 + 1 \cdot x^5 + 1 \cdot x^6 = 1 + x^2 + x^5 + x^6.$$

З розглянутого прикладу витікає особливість побудови апаратних засобів для виконання операції множення поліномів. Найпоширеніші з них полягають у тому, що коефіцієнти поліному P_1 послідовно, починаючи зі старших розрядів, вводяться в структуру, що відображає інший поліном. Вказана структура забезпечує поетапне знаходження суми за модулем **2**, внаслідок чого на виході отримуються коефіцієнти поліному-добутку. Важливим моментом у такій процедурі є те, що поліном P_1 може бути будь-яким, у той час як поліном P_2 реалізований апаратно і змінюватись не може.

За аналогією з множенням двійкових поліномів, виконується і операція ділення. Для ділення використовується операція віднімання, але віднімання за модулем **2** співпадає з сумою за модулем **2**. При апаратній реалізації операції

ділення поліном P_1 послідовно, починаючи зі старших розрядів, вводиться в структуру, в якій забезпечується покроковий зсув і порозрядне віднімання. В результаті на виході буде отриманий поліном $P = P_1 / P_2$.

Пристрої множення та ділення на поліноми широко використовується при контролі правильності передачі інформації у послідовному форматі. Це пристрої дистанційного керування через COM-порт та USB-шини, пристрої запису та зчитування інформації на жорсткі диски, системи телекомунікацій, техніка кабельного та радіозв'язку, окремі комп'ютерні мережі.

Методика такого контролю полягає в тому, що передаваний код P_1 у пристрої, який називається *кодером,* множиться на поліном P_2, який називається *породжуючим поліномом*, і в лінію передається добуток $P = P_1 \cdot P_2$. На приймальній стороні у пристрої, який називається *фільтром*, відбувається ділення коду P на породжуючий поліном P_2. Якщо операція ділення виконується без залишку, то інформація прийнята без пошкоджень.

Використовуються і інші прийоми контролю правильності передачі інформації. Сучасна теорія завадостійкого кодування дає можливість цілеспрямовано підбирати вигляд породжуючого поліному для забезпечення необхідних корегуючих властивостей. Коди, які побудовані на використанні операцій з породжуючими поліномами, відносяться до класу циклічних кодів (коди Хемінга, Файра).

Використання операцій з породжуючими поліномами дає можливість забезпечувати автоматичне шифрування повідомлень.

7.6.5. Системи контролю цифрової апаратури

Вузли ділення поліномів використовують також при контролі правильності роботи логічних схем методом сигнатурного аналізу. На входи схеми, що перевіряється, з генератора псевдовипадкових чисел подається визначеної довжини та завжди однакова послідовність псевдовипадкових чисел. Вихідна тестова послідовність проходить через поліноміальний розподілювач, який

ділить її на достатньо складний поліном *G*. Поліноміальний розподілювач називають *сигнатурним аналізатором*. Залишок, який залишається в регістрі поліноміального розподілювача після закінчення вихідної тестової послідовності, називають *сигнатурою*. На виході справної цифрової системи сигнатура від заданої цифрової тестової послідовності заздалегідь відома і може бути розпізнана, наприклад, за допомогою компаратора. Сигнатура несправної системи інша, причому, завдяки процедурі ділення та складному поліному *G*, будь-яка незначна відмінність значно посилюється. На теперішній час широке використання отримав шістнадцятирозрядний аналізатор. Частка помилок, котра ним не виявляється, оцінюється величиною 2^{-16}.

Сигнатурні аналізатори використовуються як при стендовому контролі цифрових систем, так і при створенні вбудованих пристроїв самоконтролю. В останньому випадку додатково застосовують так званий принцип *“наскрізного регістра”*.

Суть цього методу полягає у наступному. Для повного контролю складних цифрових схем необхідно мати доступ до великої кількості контрольних точок, які повинні бути виведені на роз’єми відповідних плат, або до виводів ВІС. Однак для цього повинна бути велика кількість виводів, що у більшості випадків забезпечити неможливо. У зв’язку з цим в якості контрольних точок зручно використовувати виводи тригерів. З’єднуючи всі тригери аналізованої схеми (а у ряді випадків – і додаткові зовнішні тригери) в один регістр зсуву, можливо послідовним кодом вводити тестову інформацію, що аналізує контрольні точки. Зазвичай для вирішення цієї задачі використовується третій стан мікросхем, тригери відключаються від основної схеми та заводяться спеціальні серії синхросигналів. Перевірка працездатності наскрізного регістра здійснюється за допомогою генераторів псевдовипадкових послідовностей та сигнатурних аналізаторів.

7.6.6. Використання регістрів для обчислення контрольної суми

При передачі даних у послідовному форматі широко використовується перевірка достовірності передачі-прийому масивів за допомогою *контрольних сум.* Суть даного способу перевірки полягає в тому, що разом з масивом передається невеликий контрольний код обсягом до 32 розрядів, що утворюється шляхом додавання всіх інформаційних слів (байтів) масиву як беззнакових чисел і містить у собі в згорнутому вигляді інформацію про весь масив. При читанні отриманого масиву за тим же алгоритмом знову обчислюється той самий контрольний код, і при його збіганні з кодом, приєднаним до масиву, з достатньо великою ймовірністю вважається, що масив переданий без помилок; у протилежному випадку – що масив прийнятий або відтворений з похибкою.

Оскільки розміри контрольного коду (контрольної суми) набагато менші розмірів масиву даних, то ймовірність її спотворення значно менша, порівняно з можливістю спотворення контрольованого масиву. Однак підраховано, що ймовірність спотворення масиву так, щоб контрольна сума залишилася незмінною, дуже мала. Взагалі, при виборі засобів контролю достовірності передачі інформації завжди аналізують всі можливі типи спотворень, обчислюють їх ймовірності і, виходячи з допустимої ймовірності, вибирають компроміс між надійністю системи та її вартістю.

Описаний спосіб контролю передачі інформації широко використовується в комп’ютерних мережах різного рівня, а також у комп’ютерних системах при обміні інформацією з оперативною та постійною пам’яттю, що розміщується на жорстких носіях та флеш-картках. Але, на жаль, він дозволяє лише фіксувати наявність помилки, не надаючи інформації про місце її виникнення та характер. Тому при неспівпаданні контрольних сум процес прийому або відтворення масиву, якщо це можливо, повторюють.

Серед способів обчислення контрольної суми найбільшого поширення набув *циклічний спосіб контролю за надлишковістю* (*CRC – Cycle Redundancy*

Check), при якому використовується циклічна контрольна сума. Обчислюється вона у такий спосіб. Весь контрольований масив інформації розглядається як одне *N*-розрядне двійкове число. Це число ділиться за модулем **2** на постійний поліном. Залишок від ділення використовується в якості контрольної суми.

Приклад 7.8. Обчислити контрольну суму для масиву 101111001110 при використанні поліному 10011.

Розв'язання. Ділення за модулем **2** виконується в тій самій послідовності, що і звичайне ділення двійкових чисел (див. **Розділ І**) з тією лише різницею, що замість операції віднімання виконується операція додавання за модулем **2**. Операцію ділення виконаємо в стовпчик.

```
  1 0 1 1 1 1 0 0 1 1 1 0 | 1 0 0 1 1      Поліном
⊕ 1 0 0 1 1               |
-----------               |
  0 0 1 0 0 1 0           |
⊕   1 0 0 1 1             |
    -------               |
    0 0 0 0 1 0 1 1 1     |
  ⊕   1 0 0 1 1           |
      -------             |
      0 0 1 0 0 0         Залишок (контрольна сума)
```

Оскільки результат виконання операції не використовується, то отриманий залишок 1000 і є необхідна контрольна сума.

Вибір поліному для обчислення контрольної суми однозначно пов'язаний із заданою ймовірністю виявлення помилок. Розглянутий спосіб виявляє одиночні помилки у контрольованому масиві з ймовірністю 1, а будь-яку іншу кількість помилок – з ймовірністю, що приблизно визначається формулою:
$P = 1 - 2^{-n}$, де n – кількість розрядів контрольної суми.

Як приклад, для $n = 8$ $P = 0{,}996$; для $n = 16$, $P = 0{,}999985$, а для $n = 32$, $P = 0{,}999999999767$.

З приведеного прикладу можна відмітити таку особливість залишку: він має розмірність на один розряд меншу, ніж розмірність поліному. Це не випадковість. Можна взяти масив будь-якої довжини, і результат (залишок) також матиме довжину на один розряд меншу довжини поліному. Таким чином, вибравши ймовірність виявлення помилок, обчислюється розмір контрольної суми і, відповідно, довжину поліному. Другою вимогою до поліному є те, щоб

він був простим числом з точки зору ділення за модулем **2** (вибраний поліном повинен ділитися лише на 1 і на самого себе).

Оскільки розглянутий спосіб здебільшого використовується при передачі інформації у послідовному форматі, то, відповідно, і обчислення контрольної суми необхідно вести послідовно, біт за бітом, після чого передавати її у послідовному форматі вслід за масивом.

Послідовний обчислювач контрольної суми виконується на регістрі зсуву зі зворотними зв'язками від деяких розрядів через суматори за модулем **2.** Повна кількість розрядів регістра зсуву повинна бути рівною розрядності контрольної суми, що обчислюється. Місце підключення зворотних зв'язків однозначно визначається вибраним поліномом (згадайте генератори псевдовипадкових послідовностей). Кількість з'єднань для створення зворотного зв'язку визначається кількістю одиниць в поліномі (одиниця молодшого розряду не враховується), а номери розрядів регістра зсуву, з яких беруться сигнали зворотних зв'язків, визначаються номерами одиничних розрядів у коді поліному. На відміну від генератора псевдовипадкових послідовностей, у даному випадку необхідно змішувати по функції **ВИКЛ. АБО** не тільки сигнали зворотних зв'язків, а й вхідний сигнал даних у послідовному коді.

На рис. 7.41 приводиться приклад схеми пристрою для послідовного обчислення шістнадцятирозрядної циклічної контрольної суми при поліномі **1 0001 0000 0010 0001**. Оскільки в коді поліному наявні три одиниці (за виключенням молодшого розряду), то необхідно вибрати три лінії зворотних зв'язків з розрядів регістра, відповідних положенням одиничних бітів у поліномі.

Перед початком роботи регістр встановлюється у нуль (**Уст. "0"**). Біти масиву супроводжуються синхросигналом. По закінченню масиву в регістр буде записана контрольна сума, яка серією синхросигналів буде виведена вслід за бітами масиву. Швидкодія даного пристрою обмежується

сумарною затримкою регістра та елементів **ВИКЛ. АБО**, яка повинна бути меншою періоду синхросигналів.

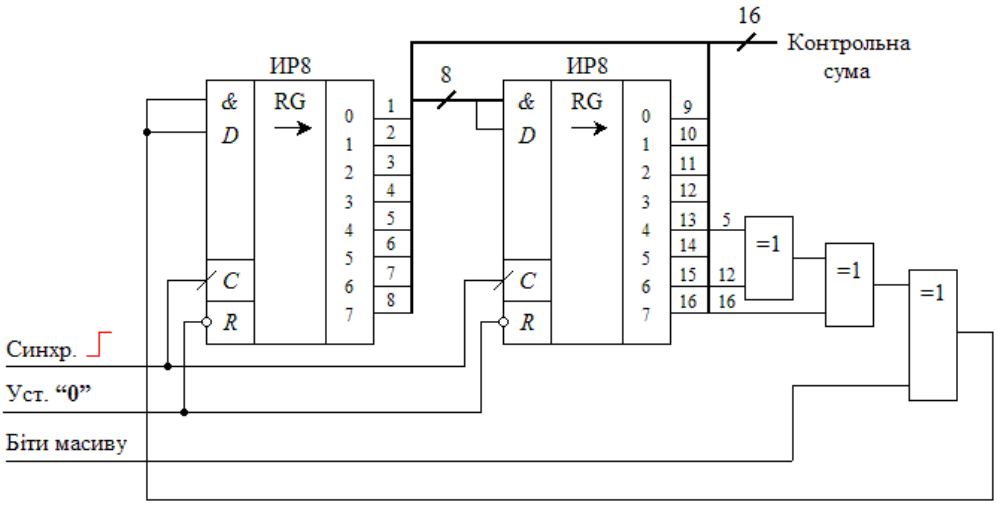


Рис. 7.41

КОНТРОЛЬНІ ПИТАННЯ

1. Приведіть визначення терміну *"регістр"*. Якими параметрами характеризуються регістри?
2. Приведіть основні класифікаційні ознаки регістрів.
3. Які функції виконує вхідна і вихідна логіка регістрів пам'яті?
4. Чим принципово відрізняються регістри пам'яті і регістри зсуву?
5. Поясніть призначення регістрових файлів та особливості їх побудови.
6. Поясніть призначення та функції буфера *FIFO*. Яке відношення він має до регістрів?
7. Поясніть призначення та функції буфера "стек".
8. У чому полягає суть конвеєрної обробки інформації? Які функції в ньому виконують регістри?
9. Приведіть визначення терміну *"регістр зсуву"*.
10. Чи можливо використовувати статичні тригери в регістрах зсуву? Якщо ні, чому? Якщо можливо, то приведіть принципову або функціональну схему такого регістра і часові співвідношення між вхідними сигналами.

11. Який з пристроїв – демультиплексор чи регістр зсуву – має переваги при перетворенні послідовного формату двійкових слів у паралельний?

12. Як виконуються операції ділення та множення за допомогою регістрів зсуву?

13. Поясніть, яким шляхом набори тригерів можуть бути перетворені у регістри пам’яті; у регістри зсуву.

14. Ви маєте умовне зображення мікросхеми регістра з позначеннями виводів. Яка ще потрібна інформація, щоб визначитись з режимами її роботи?

15. Чи можливо приєднувати виходи кількох регістрів, що мають виходи з *Z*-станом, на одну шину? Якщо неможливо, чому? Якщо можливо, то при яких умовах?

16. Поясніть особливості синхронних і асинхронних послідовних способів передачі інформації.

17. Приведіть приклади використання кільцевих лічильників.

18. Перелічіть і поясніть недоліки кільцевих лічильників.

19. Поясніть призначення і особливості використання схеми самозапуску кільцевого лічильника.

20. Як отримати кільцевий лічильник, в якому циркулює 0, а не 1?

21. Як виправляються помилки при роботі кільцевих лічильників?

22. У чому полягає особливість лічильника Мебіуса? Приведіть приклади використання такого лічильника.

23. Поясніть методологію дешифрації станів лічильників Джонсона та Мебіуса.

24. Поясніть спосіб усунення небажаних послідовностей у лічильнику Джонсона.

25. У чому полягає особливість кільцевого лічильника зі зворотнім зв’язком через логічні елементи **ВИКЛ. АБО**?

26. Приведіть приклади використання кільцевих лічильників-генераторів псевдовипадкових послідовностей.

27. Поясніть особливість використання регістрів зсуву для кодування інформації.

28. Поясніть особливість контролю цифрової апаратури на основі регістрів зсуву.

ВПРАВИ І ЗАВДАННЯ

1. Використовуючи мікросхеми з наборами *RS*-, *D*- та *JK*-тригерів, а також допоміжну логіку, розробити:

а) асинхронний регістр пам’яті з дозволяючим входом запису і відкритим виходом на чотири розряди;

б) синхронний регістр пам’яті з одним входом дозволу запису та дозволу зчитування на чотири розряди;

в) синхронний регістр пам’яті з дозволом запису та дозволом зчитування у прямому або інверсному кодах;

г) синхронний регістр пам’яті з буферизованим виходом та *Z*-станом.

2. На прикладі регістра SN74LS173A (чотири розряди) розробити схему регістра пам’яті на 12 розрядів.

3. Використовуючи регістри пам’яті SN74LS173A, розробити схему модуля на вісім чотирьохрозрядних слів, в якому запис та зчитування даних забезпечувалися б по одній шині даних.

4. Використовуючи мікросхему КР1533ИР30 та допоміжні елементи, розробити схему пристрою для перетворення однобайтового слова, заданого у послідовному форматі, у паралельний. Побудувати часові діаграми.

5. Використовуючи мікросхему ИР5, розробити схему пристрою для мультиплексування чотирьохрозрядних даних з чотирьох шин на одну. Сформувати адресні сигнали.

6. Використовуючи регістри SN74LS173A та допоміжні мікросхеми, розробити функціональну схему регістрового файлу для запису та зчитування восьми чотирьохрозрядних слів.

7. Розширити обсяг пам’яті регістрового файлу до чотирьох восьмирозрядних слів; до восьми восьмирозрядних слів.

8. Використовуючи мікросхеми SN74LS670 та допоміжні елементи, розробити схему модуля пам’яті *FIFO*.

9. Використовуючи мікросхеми SN74LS670, розробити схему буфера *LIFO* (“стек”): *а*) на чотири чотирьохрозрядні слова; *б*) на вісім чотирьохрозрядних слів; *в*) на чотири восьмирозрядних слова.

10. Вміст регістру зсуву з послідовним входом і послідовним виходом $Q_3\ Q_2\ Q_1\ Q_0 = 0101$. Послідовність 1011 вводиться зліва направо, починаючи зі старшого розряду, за чотири такти. Побудувати часові діаграми зміни вмісту тригерів регістру.

11. Дати порівняльну характеристику швидкодії регістру зсуву при переміщенні інформації зліва направо та справа наліво.

12. Розробити схему регістру зсуву на мікросхемі КР1533ИР23.

13. Використовуючи набір *D*-тригерів ИР23, розробити регістр зсуву справа наліво.

14. Розробити стек на вісім чотирьохрозрядних слів на основі регістру зсуву.

15. На основі регістрів зсуву та допоміжної логіки розробити лічильники:

а) з $M = 10$; *б*) з $M = 12$; *в*) з $M = 24$.

16. Використовуючи регістри зсуву, розробити схему пристрою для циклічної генерації послідовності 01110101.

17. Побудувати таблицю станів для чотирьохрозрядних регістрів зсуву з елементами **ВИКЛ. АБО** в колі зворотного зв’язку при наступних умовах:

а) $DO = Q_0 \oplus Q_1$; *б*) $DO = Q_2 \oplus Q_3$.

18. Знайти функцію зворотного зв’язку *DO*, при якій лічильник не матиме станів: *а*) 00000; *б*) 11111.

19. Для генерації двох послідовностей, довжина яких складає 7 біт і 5 біт, використовується трьохрозрядний регістр зсуву. Для переходу від однієї

послідовності до іншої використовується керуючий сигнал d. Спроектувати на основі регістра зсуву такий генератор, щоб при $d = 0$ генерувалася б послідовність довжини 7 біт, а при $d = 1$ – відповідно, 5 біт.

20. Розробити імітатор секундної стрілки на круглому циферблаті за допомогою кільцевого лічильника і кільця світлодіодів.

21. Розробити восьмибітний кільцевий самокорегований лічильник з циклічним переміщенням одного нуля.

22. Визначити максимальну довжину циклу для схеми на рис. 7.40, якщо один з входів суматора за модулем **2** взяти не Q_2, а Q_1. Побудувати таблицю станів.

23. Проаналізувати таблиці станів для трьохбітного лічильника зі зворотним зв'язком через елементи ВИКЛ. АБО і знайти рівняння зворотного зв'язку для послідовності максимальної довжини, яке відрізняється від приведеного у табл. 7.12.

24. На рис. 7.42 приведена схема автомату для гри на угадування, в основу якої покладено регістр ИР11. Проаналізувати і описати роботу автомату.

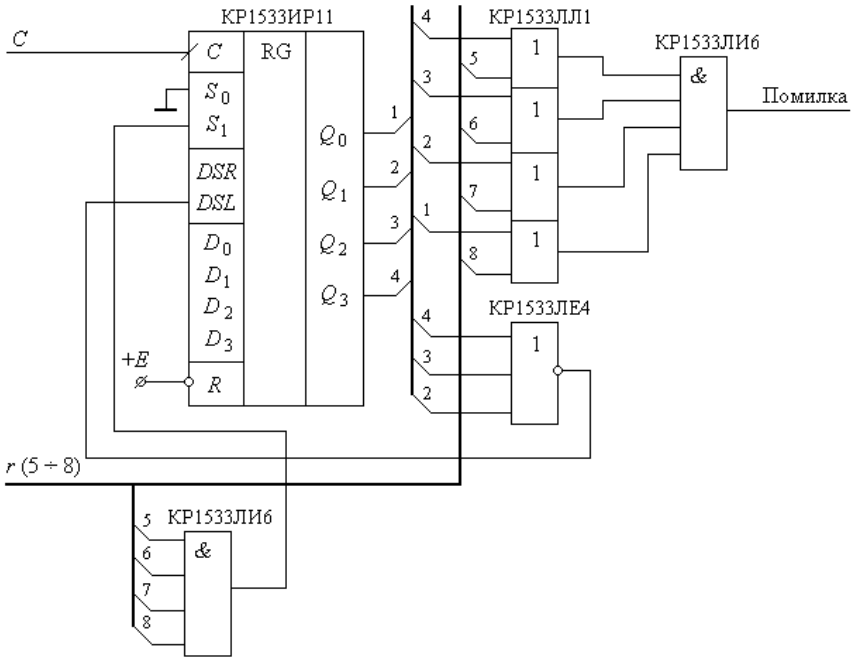


Рис. 7.42

Розділ 8

ЗАПАМ'ЯТОВУЮЧІ ПРИСТРОЇ

У попередніх розділах розглянуті пристрої тимчасового запам'ятовування невеликих обсягів інформації (тригери, регістри, регістрові файли). Але сучасні інформаційні технології вимагають оперативної роботи із значно більшими об'ємами інформації, яка може досить швидко змінюватись.

З великого обсягу різноманітних пристроїв запам'ятовування і зберігання інформації у цьому розділі розглянемо лише напівпровідникові, які широко використовуються як у мікропроцесорних системах, так і в якості складових різноманітних систем електроніки, автоматики та ін.

Використання напівпровідникових пристроїв пам'яті широке та різноманітне, завдяки наявності на ринку електронних компонентів великої кількості різноманітних запам'ятовуючих пристроїв (ЗП). В літературі використовуються різні способи їх класифікації, але в цілому вони можуть розглядатись як дві великі групи – *енергозалежні* та *енергонезалежні* ЗП, тобто визначальним фактором виступає те, зберігається записана в них інформація при відключенні живлення чи ні.

8.1. Постійні запам'ятовуючі пристрої (принципи побудови, типи, характеристики)

Розглянемо спочатку енергонезалежну пам'ять, яка в найбільш загальній формі називається *ПЗП – постійні запам'ятовуючі пристрої* (***ROM*** – ***R****ead* ***O****nly* ***M****emory*).

8.1.1. Одновимірні ПЗП

ПЗП – це пристрої пам'яті, вміст яких не може бути змінений процесором під час виконання програми і зберігається при відсутності живлення. Приклад структури ПЗП приводиться на рис. 8.1. Вона складається з дешифратора адреси $A_2 A_1 A_0$ і діодної матриці. У відповідності до значень сигналів на

адресних входах, один з виходів дешифратора активізується (у даному випадку – на одному з виходів 0…7 з'являється сигнал високого рівня), який через відповідні діоди передається на виходи $D_0 \ldots D_3$, фактично відображаючи у двійковому коді слово, що створене в матриці за допомогою приєднаних діодів.

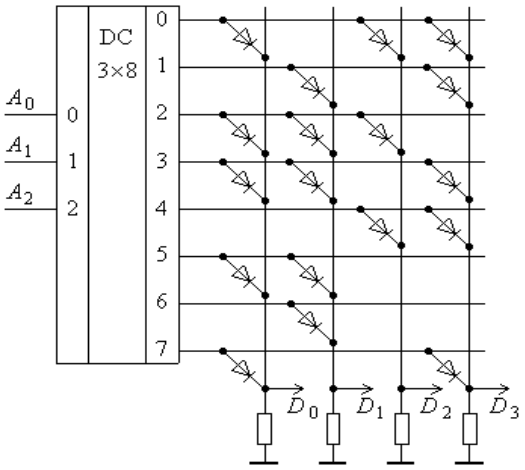


Рис. 8.1

Приклад 8.1. Пояснити, яка двійкова комбінація з'явиться на виході ПЗП (рис. 8.1), якщо на вхід подається двійковий код $A_2\, A_1\, A_0 = 011$.

Розв'язання. Приведеному двійковому коду на адресних входах відповідає активний вихід дешифратора з номером **3,** високий рівень сигналу якого буде переданий на виходи D_0, D_1 і D_3. Вихід D_2 матиме низький рівень вихідного сигналу, оскільки він через резистор приєднаний до загальної шини. Звідси знаходимо слово, записане за вказаною адресою:

$$D_3\, D_2\, D_1\, D_0 = 1011.$$

Аналогічно можна скласти таблицю станів для кожного з виходів як функцію адресних входів, яка для приведеної на рис. 8.1 схеми представлена в табл. 8.1.

ПЗП можна вважати комбінаційною схемою з n входами $A_0 \ldots A_{n-1}$, які називаються *адресними* і m-виходами $D_0 \ldots D_{m-1}$, які є виходами даних. Іншими словами, ПЗП зберігає в матричному пристрої пам'яті таблицю станів n-входової і m-виходової комбінаційної функції. Виходячи з даних табл. 8.1, можемо стверджувати, що в даному ПЗП зберігається 2^3 слів, кожне з яких має розмірність $m = 4$ розряди. Інформаційна ємність його складає $2^3 \times 4 = 32$ біти.

Умовне позначення ПЗП приводиться на рис. 8.2.

Таблиця 8.1

N	Входи			Виходи			
	A_2	A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	1	1	0	1
1	0	0	1	1	0	1	0
2	0	1	0	0	1	1	1
3	0	1	1	1	0	1	1
4	1	0	0	1	1	0	0
5	1	0	1	0	0	1	1
6	1	1	0	0	0	1	0
7	1	1	1	1	0	0	1

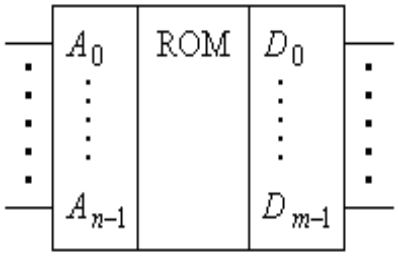


Рис. 8.2

Оскільки ПЗП є комбінаційною схемою, то справедливим буде твердження, що це взагалі не пам’ять, а група комбінаційних логічних елементів. Але, у той же час, якщо у ПЗП записується інформація, то зрозуміло, що вона звідти зчитуватиметься неодноразово і зберігатиметься навіть при відсутності живлення. Це і мається на увазі, коли говорять, що ПЗП є енергонезалежною пам’яттю.

Аналізуючи стани табл. 8.1, можемо її розглядати як таблицю чотирьох функцій трьох змінних:

$$D_0 = f_0(A_0 \dots A_2); \qquad D_1 = f_1(A_0 \dots A_2);$$
$$D_2 = f_2(A_0 \dots A_2); \qquad D_3 = f_3(A_0 \dots A_2).$$

Як витікає з **Розділу 3**, така система функцій може бути реалізована на основі дешифратора 3 × 8 і чотирьох логічних елементів **АБО**, кількість входів кожного з яких відповідає кількості мінтермів відповідної логічної функції:

$$D_0 = \bigvee_0^7 0, 2, 3, 5, 7; \qquad D_1 = \bigvee_0^7 1, 2, 3, 5, 6;$$
$$D_2 = \bigvee_0^7 0, 2, 4; \qquad D_3 = \bigvee_0^7 0, 1, 3, 4, 7.$$

Таке представлення записаної в ПЗП інформації забезпечує спосіб їх побудови на основі дешифраторів та багатовходових логічних елементів **АБО**.

З іншого боку, за кожною з адрес $A_2 \dots A_0$ маємо чотири біти даних, які можуть зчитуватись як окремі записані слова. Представивши їх у шістнадцятковій системі числення, табл. 8.1 можемо зобразити в іншій формі

(див. табл.8.2), де кожному значенню адреси, яка зображена у вигляді впорядкованого ряду мінтермів, відповідає конкретне значення вихідного слова, що записане в структурі пам'яті.

Таблиця 8.2

$A_2 \div A_0$	0	1	2	3	4	5	6	7
$D_3 \div D_0$	D	A	7	B	C	3	2	9

Табл. 8.2 називається *таблицею* (*картою*) *прошивок* або *гексадецимальним лістінгом*.

Таблиця станів (табл. 8.1) може мати будь-які значення двійкового коду, тому вона може розглядатися ще з однієї позиції. Позначимо, наприклад, входи $A_1 A_0$ як входи одного слова, а $A_2 = B_0$ – вхід іншого слова. Враховуючи, що у таблицю станів можна записати будь-які значення, котрі відповідають взаємодії цих двох слів, створимо таблицю, яка відповідає арифметичній дії додавання дворозрядного слова A і однорозрядного слова B. Тоді значення виходів можемо позначити як P_2, S_1, S_0, де P_2 – значення переносу в третій розряд; $S_1 S_0$ – розряди суми. Отримуємо відповідну таблицю (табл. 8.3).

Приклад 8.2. Використовуючи ПЗП формату 256 × 8, розробити таблицю прошивок для виконання операції множення двох чотирьохрозрядних слів.

Таблиця 8.3

B_0	A_1	A_0	P_2	S_1	S_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	0	1	1
1	1	1	1	0	0

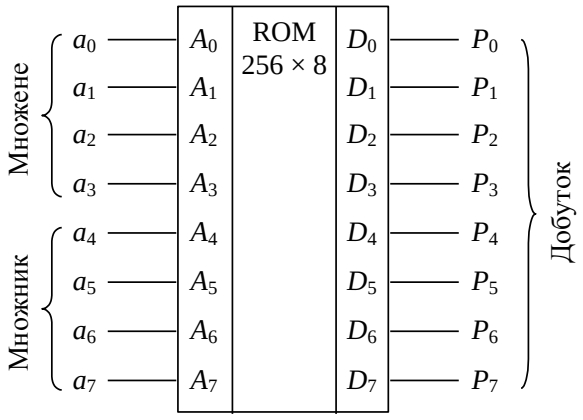


Рис. 8.3

Розв'язання. На рис. 8.3 приведена схема, яка пояснює особливості використання виводів мікросхеми при побудові перемножувача.

Таблиця прошивок мікросхеми приведена у табл. 8.4.

Множене *Таблиця 8.4*

Множник	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
20	00	02	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
30	00	03	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
40	00	04	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
50	00	05	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
60	00	06	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
70	00	07	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
80	00	08	10	18	20	28	30	38	40	48	50	58	60	68	70	78
90	00	08	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A0	00	0A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B0	00	0B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C0	00	0C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D0	00	0D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E0	00	0E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F0	00	0F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

У ній крайній зліва стовпець (ліві знаки) відображає адресний простір старших розрядів ($A_4 \ldots A_7$), що записані у шістнадцятковій системі числення. Решта стовпців визначаються молодшими розрядами адресного простору. На перетині відповідного стовпця та рядка у відповідній клітині записується шістнадцяткове значення добутку.

Якщо, наприклад, задамо множник $7_{10} = 7_{16}$, а множене $8_{10} = 8_{16}$, то на перетині рядка 70 і стовпця 08 маємо результат $38_{16} = 56_{10}$. Це означає, що з шини $D_0 \div D_7$ повинно бути зчитане число 38_{16}. Запис такого числа в пам'ять забезпечується в такій послідовності. Оскільки вибрана адреса визначається кодом $78_{16} = 01111000_2$, то це означає, що в мікросхемі пам'яті вибраний $01111000_2 = 120_{10}$ рядок. В цей рядок необхідно записати слово $38_{16} = 56_{10} = 00111000_2$.

Внутрішня структура ПЗП реально дещо відрізняється від приведеної на рис. 8.1 – перш за все, тим, що внутрішній дешифратор має інверсні виходи, а тому при активізації відповідного рядка його потенціал стає близьким до нуля (рис. 8.4). Кожен з виходів дешифратора називається *рядком слова*, оскільки ним вибирається слово з матриці діодів. Вертикальні шини через резистори приєднані до джерела живлення. Вони називаються *бітовими стовпцями*, оскільки відповідають значенню одного біта записаного слова.

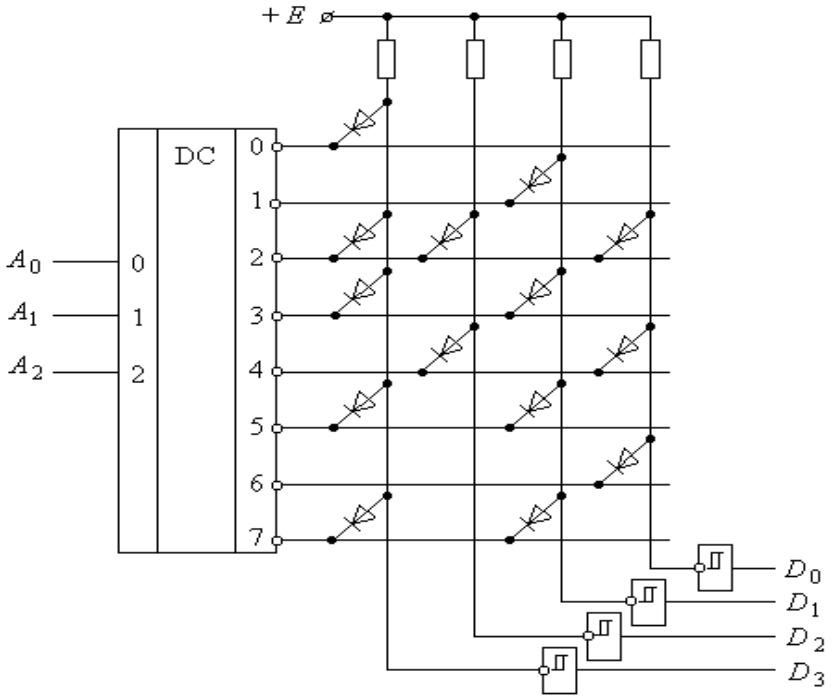


Рис. 8.4

У схемі, що розглядається, з’єднання через діод відповідних ліній слова і бітового стовбця забезпечує запис логічної одиниці.

Наявність інверторів з тригерами Шмідта покращує рівень захисту від перешкод, оскільки падіння напруги на діоді складає приблизно 0,5…0,6 В, і падіння напруги на відкритому виході дешифратора підвищують стійкість до перешкод, що виникають на загальній шині живлення.

Мікросхеми ПЗП виготовляються також із використанням МДН-технології, де в якості елементів пам’яті використовуються транзистори з n- або p- каналами, а також комплементарні ключі. При використанні МДН-транзисторів під час виготовлення ПЗП запис логічних одиниць або нулів забезпечується за рахунок величини порогової напруги стокозатворної характеристики транзистора. Збільшення або зменшення величини порогової напруги встановлюється товщиною ізоляційного прошарку між затвором та напівпровідником. У матрицях з такими транзисторами всі транзистори, що відносяться до одного записаного слова, приєднуються затворами до вибраного рядка, а стоки кожного транзистора з’єднуються з відповідним розрядним

стовпцем матриці. Витоки всіх транзисторів об'єднані і приєднуються до джерела живлення. При виборі відповідного рядка транзистори з низьким пороговим рівнем напруги відкриваються, і на відповідних стовпцях матриці отримується сигнал логічної одиниці. У транзисторах з високою порогової напругою буде записаний логічний нуль.

8.1.2. Двовимірне декодування в ПЗП

Припустимо, що нам необхідно побудувати ПЗП з організацією 128 × 1 за структурою, що була описана вище. Зрозуміло, що в такому випадку необхідно мати дешифратор **"з 7 на 128"**, котрий, як відомо, не так легко створити. Тому виникла ідея створення ПЗП такого типу – на основі діодної матриці з розділеною дешифрацією рядків та стовпців. Приклад її реалізації приводиться на рис. 8.5.

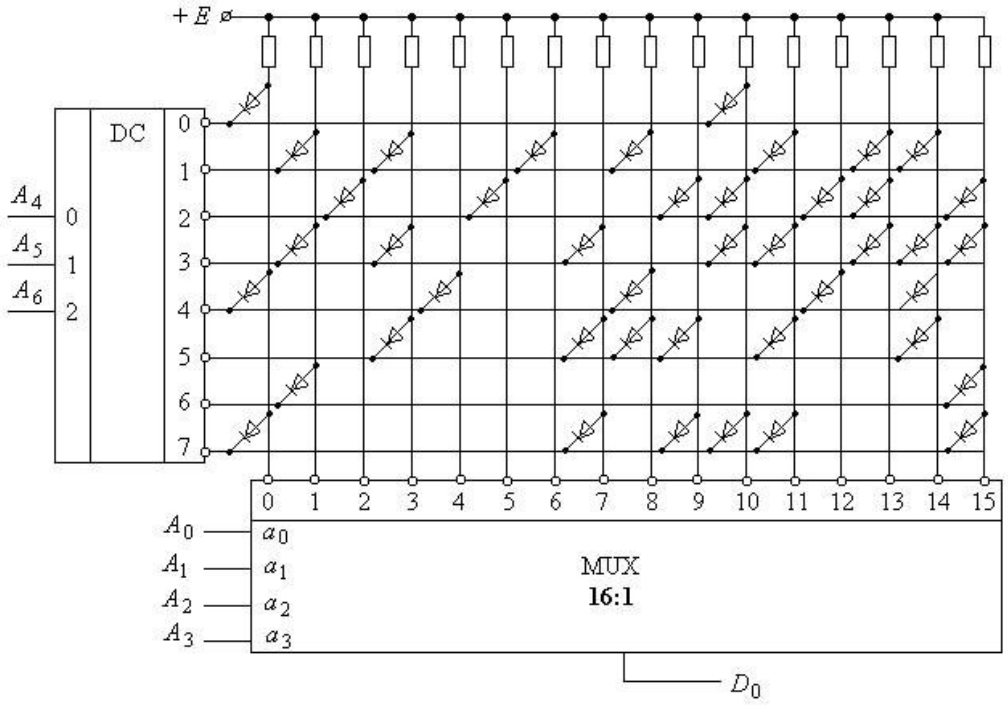


Рис. 8.5

Вся шина адресного простору розбивається у даному випадку на дві нерівні частини. Три старші розряди $A_6 A_5 A_4$ дешифруються у вісім рядків, а

молодші розряди $A_3\,A_2\,A_1\,A_0$ використовуються для керування мультиплексором, що вибирає один з 16 бітових стовпців.

Така структура близька за формою до квадратної матриці, що забезпечує найбільш ефективне використання площі кристалу. Матрична структура побудови ПЗП відповідає ідеології фізичної реалізації на кристалах квадратної або майже квадратної форми.

У ПЗП з великою кількістю розрядів даних однобітна матриця може бути більш вузькою з тим, щоб структура всієї мікросхеми була наближена до квадрату.

Як приклад, на рис. 8.6 приводиться можливий варіант ПЗП структури 64K × 8. Кожна матриця має 1024 рядки, які вибираються дешифратором старших розрядів адреси і 64 стовпці, кожен з яких вибирається мультиплексором **64:1**. Звідси маємо 1024 × 64 = 65 536 слів з розрядністю 8 біт.

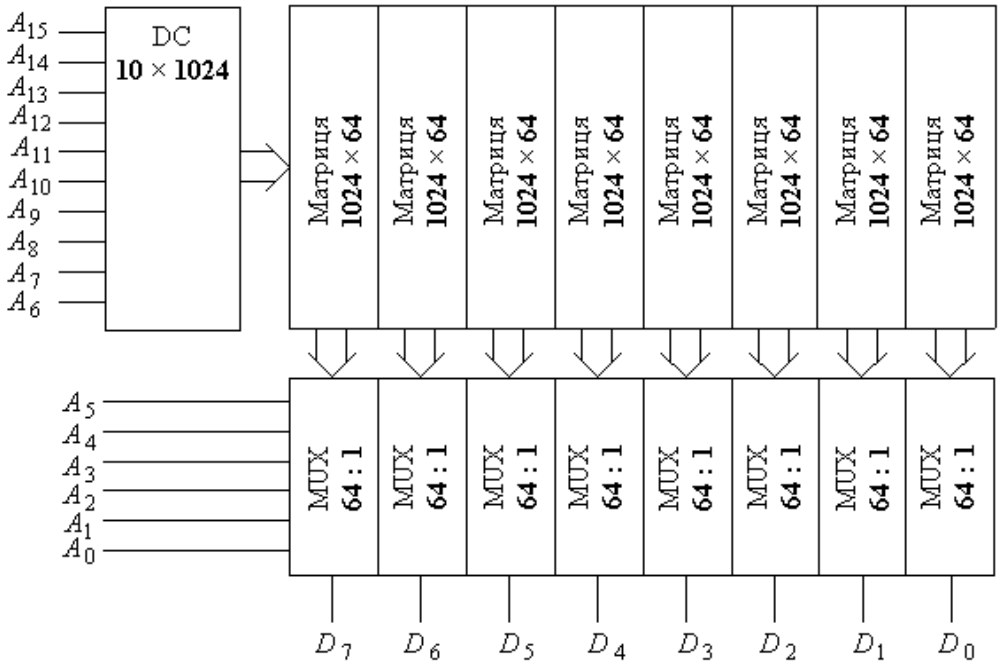


Рис. 8.6

У більш загальному плані структура ПЗП має вигляд, приведений на рис. 8.7.

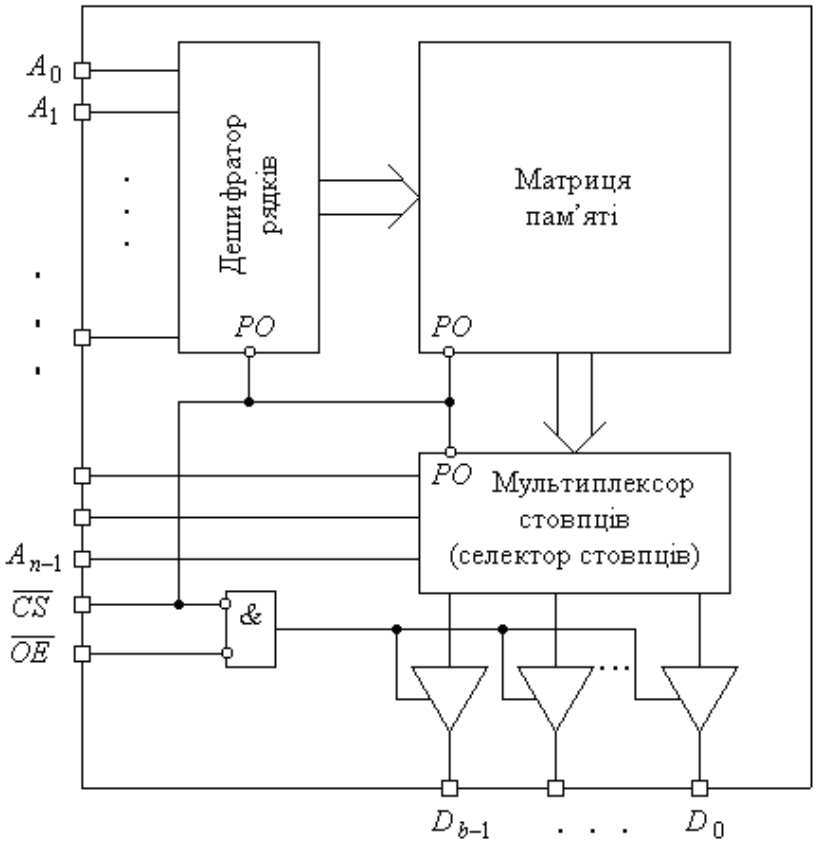


Рис. 8.7

Більшість ПЗП, оскільки до мікропроцесорних систем вони приєднуються в великій кількості, мають вхід вибору мікросхеми $\overline{CS}$. Цей вхід керує процесом енергозбереження, адже при високому рівні сигналу на ньому внутрішні елементи ПЗП відключаються від живлення, і забезпечує переведення виходів шини даних у *Z*-стан. При $\overline{CS}=1$ енергоспоживання зменшується майже на 90%. Переведення виходів із *Z*-стану і виведення інформації з мікросхеми забезпечується входом дозволу виведення даних $\overline{OE}$ (***O**utput **E**nable*).

8.1.3. Мікросхеми ПЗП

Структура ПЗП, що приведена на рис. 8.7, є загальною для всіх серій мікросхем. За такою структурою будувались вітчизняні ПЗП серій К568, КР568, К596, КР1656, а також зарубіжні серії 27ххх (2764, 27128, 27256, 27512).

Незначна різниця має місце у блоках формування вихідних даних та в організації дешифраторів рядків і селекторів стовпців.

Більш розвинену архітектуру мають ПЗП, які виготовляються за n-МОН та КМОН-технологіями (мікросхеми серій КР1801РЕ2, К1809РЕ1, КР588РЕ1, КА1603РЕ1). Не вдаючись у деталі архітектури, розглянемо її особливості на прикладі мікросхеми КР1801РЕ2, умовне позначення якої приведено на рис. 8.8.

У ній шина даних і шина адреси суміщені (за винятком входу D_0). Керування режимом зчитування забезпечується входами синхронізації $\overline{C}$, вибору мікросхеми $\overline{CS}$ та зчитування $\overline{RD}$. Робота мікросхеми описується таблицею істинності (див. табл. 8.5).

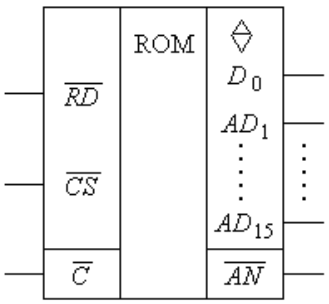


Рис. 8.8

Таблиця 8.5

Входи			Виходи / входи		
$\overline{C}$	$\overline{CS}$	$\overline{RD}$	D_0	$AD_1 \dots AD_{15}$	$\overline{AN}$
x	H	x	*Z*	*Z*	H
H	x	x	*Z*	*Z*	H
⌞	L	H	x	*A*	H
L	L	L	D_0	*D*	L

З неї витікає, що вибір даних за заданою адресою забезпечується шляхом запису адреси за зрізом синхросигналу при $\overline{CS}=0$ та $\overline{RD}=1$. Зчитування даних відбувається у наступний інтервал часу при $\overline{C}=\overline{CS}=\overline{RD}=0$. При цьому на виході $\overline{AN}$ формується сигнал низького рівня, який інформує зовнішні пристрої про зчитування інформації.

Зрозуміло, що архітектура такого ПЗП повинна бути доповнена, порівняно зі схемою на рис. 8.7, регістром тимчасового зберігання адресних сигналів, блоком керування вихідними шинами, блоком формування сигналу $\overline{AN}$.

Розглянемо ще одну особливість даної мікросхеми. Її інформаційна ємність 64К, а організація – 4К × 16. Для такої організації достатньо 12 адресних розрядів, а мікросхема має 15. Пояснюється це тим, що старші три розряди

A_{13} …A_{15} адресного коду використовуються для формування сигналу вибірки, який створюється при співпадінні коду на входах A_{13} … A_{15} з внутрішнім кодом ПЗП, що “зашивається” при виготовленні. Для формування сигналу вибірки в архітектурі ПЗП передбачений спеціальний модуль.

Подібну архітектуру мають мікросхеми ПЗП К1809РЕ1, КР588РЕ1.

Незважаючи на значну динаміку розвитку мікросхем пам’яті, появу високоякісних репрограмованих ПЗП та флеш-пам’яті, при побудові функціональних цифрових пристроїв невисокої складності широко використовуються ПЗП на основі ТТЛШ-технології – мікросхеми КР556РТ…, які виготовляються з інформаційною ємністю від 1К до 64К і організацією від 256 × 4 до 8192 × 8. До особливостей цих мікросхем слід віднести той факт, що в початковому стані в матрицю пам’яті записані нулі. Мікросхеми КР556РТ4, КР556РТ6 мають вихідні каскади з відкритим колектором, решта – з трьома станами.

Наявність відкритого колектора приводить до необхідності використання допоміжних колекторних резисторів, які приєднуються між виходами мікросхеми та джерелом живлення.

Мікросхеми ПЗП мають необхідне забезпечення для нарощування інформаційної ємності як за довжиною слова, так і за кількістю слів. Нарощування кількості слів досягається шляхом збільшення адресного простору. В результаті зростає інформаційна ємність пристрою пам’яті, яка визначається як сума об’ємів окремих мікросхем. Усі мікросхеми працюють по черзі, а вибір мікросхеми забезпечується через вхід вибору $\overline{CS}$ за допомогою допоміжного дешифратора. Адресні входи всіх мікросхем об’єднуються паралельно і створюють молодші розряди адресного коду. В якості старших розрядів адресного коду використовуються входи допоміжного дешифратора. Виходи всіх мікросхем з’єднуються паралельно.

При нарощуванні довжини слова всі входи мікросхем з’єднуються паралельно, а виходи створюють формат слова необхідної довжини.

При нарощуванні кількості слів час вибору слова зростатиме, оскільки дешифратор і мікросхеми пам’яті працюватимуть послідовно за часом.

Технічні характеристики ПЗП, аналогічно до вище розглянутих мікросхем, характеризуються *статичними електричними* та *динамічними часовими параметрами*. Статичні параметри повністю визначаються технологією виготовлення ПЗП і співпадають з відповідними характеристиками ТТЛ та КМОН ІС.

У той же час, запам’ятовуючі пристрої мають свої характерні технічні параметри. До них відносяться:

- *інформаційна ємність* – максимально можливий об’єм інформації, що може зберігатися. Виражається в бітах, кілобітах, мегабітах (наприклад, 64K = 2^{16} біт);
- *організація ЗП* – добуток кількості слів, що зберігаються в пам’яті, на їх розрядність.

Основні динамічні параметри пояснюються часовими діаграмами, що приводяться на рис. 8.9 і мають наступну фізичну інтерпретацію:

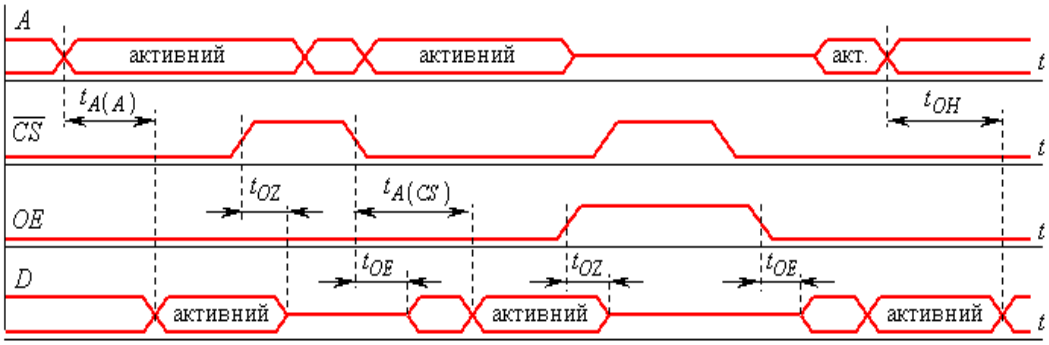


Рис. 8.9

- $t_{A(A)}$ – час вибірки за адресою – планована затримка від моменту встановлення активного рівня адресного коду до моменту отримання дійсних значень коду даних;
- $t_{A(CS)}$ – час вибірки (доступу) входу вибору мікросхеми – планована (прогнозована) затримка з моменту встановлення активного рівня сигналу на

вході $\overline{CS}$ до моменту отримання дійсних значень коду даних;

- t_{OE} – інтервал часу, за який після відповідного встановлення рівнів сигналів на входах OE і $\overline{CS}$ мікросхема виходить із *Z*-стану. Цей інтервал часу значно менший, ніж час доступу, тому часто при встановлених значеннях коду адресних сигналів сигнал OE забезпечує швидкий доступ до даних;
- t_{OZ} – інтервал часу, за який вихід мікросхеми перейде у *Z*-стан при відповідній зміні входів $\overline{CS}$ і (або) OE.
- t_{OH} – час витримки – інтервал часу, протягом якого дані ще можуть бути зчитані після того, як код адресного сигналу стане неактивним.

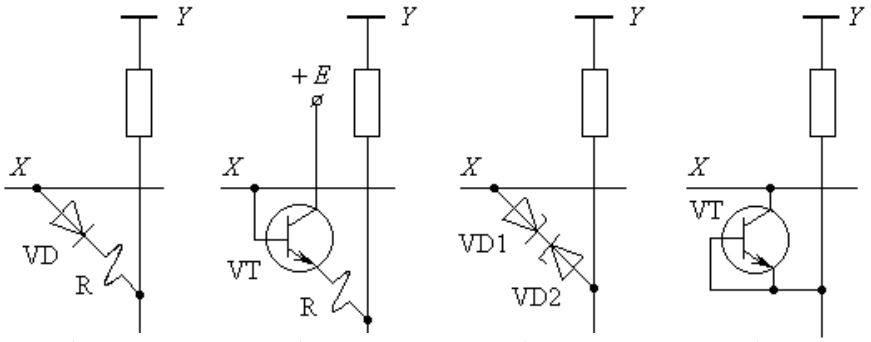
8.2. Репрограмовані ПЗП

Репрограмованими називають ПЗП, які є енергонезалежними приладами і інформація в яких може неодноразово перезаписуватись. Ця група запам'ятовуючих пристроїв привертає до себе дуже велику увагу розробників, оскільки забезпечує високу надійність зберігання записаних у них даних. Тому вони мають високу динаміку розвитку і оновлення.

8.2.1. Принципи побудови репрограмованих ПЗП

Історично першими з них були розроблені *одноразово програмовані РПЗП* (***PROM*** – ***P****rogrammable* ***ROM***, ***OTP ROM*** – ***O****ne* ***T****ime* ***P****rogrammable* ***ROM***) на основі запам'ятовуючих матриць, елементами пам'яті в яких використовувались перепалювані ніхромові перемички (рис. 8.10, *а – б*) або p-n- переходи напівпровідникових приладів (рис. 8.10, *в – г*).

В елементах пам'яті, приведених на рис. 8.10, *а – б*, у початковому стані біт інформації відповідає логічній одиниці. Це пояснюється тим, що при дешифрації рядка X матриці, тобто при $X = 1$, сигнал високого рівня передається на шину Y через діод VD та перемичку R. Запис логічного нуля забезпечується подачею короткочасного і потужного імпульсу напруги, який перепалює перемичку.



а) *б)* *в)* *г)*

Рис. 8.10

В елементах пам’яті, зображених на рис. 8.10, *в – г*, у початковому стані зберігається нуль, адже шини *X* та *Y* з’єднані зміщеним у зворотному напрямку переходом стабілітрона VD1 (рис. 8.10, *в*) та переходом колектор-база транзистора VT (рис. 8.10, *г*). Якщо перехід зруйнувати імпульсом високої напруги, то в елемент пам’яті буде записана одиниця.

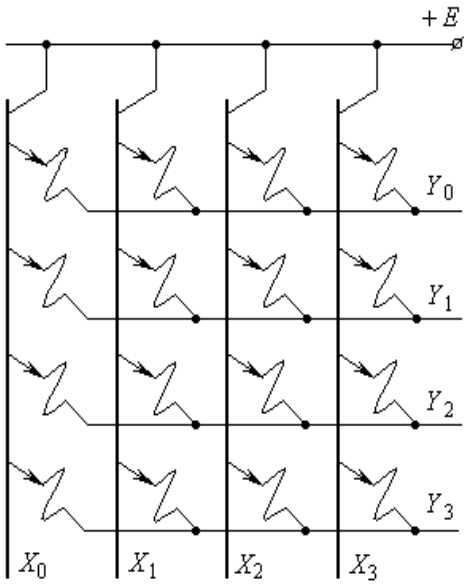


Рис. 8.11

При використанні ТТЛШ-технології матриця пам’яті створювалась на основі багатоемітерних транзисторів (рис. 8.11).

Вона вигідно відрізнялась від діодних структур тим, що перепалення необхідних перемичок забезпечується струмом, який протікає через емітер транзистора. Керування струмом емітера забезпечується значно меншим струмом бази, а, відповідно, дешифратор рядків може виготовлятися значно меншої потужності. На основі ТТЛШ-технології виготовлялися програмовані ПЗП серії 556 (556РТ4…РТ18) з інформаційною ємністю до 64 Кбіт і часом вибору даних до 70 нс.

Сучасні РПЗП будуються на основі МОН-структур і мають значно більші можливості, оскільки інформація в них може бути замінена багаторазово.

Вони розділяються на такі групи:

- *EPROM* (***E**lectrically **P**rogrammable **ROM***) або РПЗП-УФ (репрограмовані ПЗП з ультрафіолетовим стиранням записаної інформації);
- *EEPROM* (***E**lectrically **E**rasable **PROM***) або **РПЗП-ЕС** (репрограмовані ПЗП з електричним стиранням записаної інформації);
- пам'ять типу *flash* (флеш).

Особливість цих типів мікросхем полягає в тому, що запис інформації, який забезпечується електричними імпульсами, проходить значно повільніше, ніж читання, що є принциповою різницею між ними та пристроями оперативної пам'яті. Запам'ятовуючими елементами є спеціальні транзистори МОН-типу, в яких підзатворний ізоляційний діелектрик складається з двох шарів – тонкого (двоокис кремнію $Si\,O_2$) та товстого (нітрид кремнію $Si_3\,N_4$) (рис. 8.12) (завдяки появі допоміжного шару на основі нітриду кремнію ці транзистори отримали назву МНОН - метал-нітрид-окисен-напівпровідник).

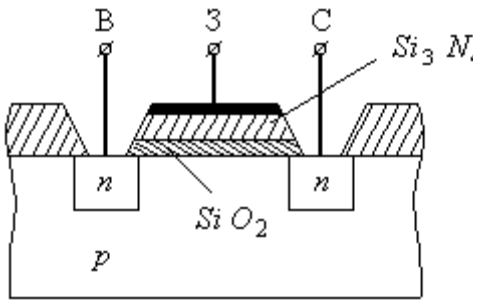


Рис. 8.12

Особливість цих транзисторів проявляється у появі на межі розподілу двох ізоляційних шарів центру захоплення заряду та його зберігання. Накопичення заряду відбувається за рахунок створення явища тунелювання основних носіїв з р- структури при прикладанні до затвору негативної напруги по відношенню до підкладки. Накопичений позитивний заряд між шарами приводить до зниження величини порогової напруги. При відсутності заряду або при наявності заряду протилежної полярності величина порогової напруги залишається незмінною або зростає. Накопичені заряди зберігаються в описаній структурі як при наявності зовнішньої напруги, так і при її відсутності. РПЗП на основі МНОН-транзисторів можуть зберігати записану інформацію десятиріччями. Робота матриці пам'яті на основі МНОН-структур пояснюється рис. 8.13.

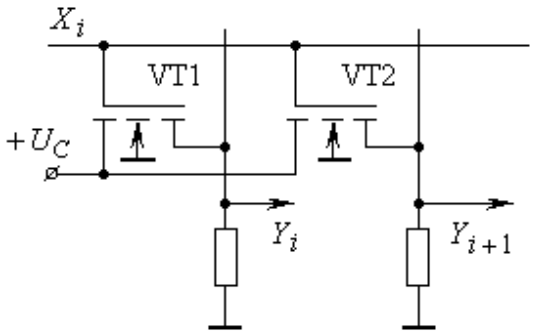


Рис. 8.13

Високий рівень сигналу на шині вибору X_i приводить до того, що в транзисторах з низьким пороговим рівнем створюється канал для протікання струму, і на відповідних виходах Y_i буде високий рівень сигналу, еквівалентний логічній одиниці. Якщо ж у транзисторі при запису створений високий рівень порогу, то канал буде відсутнім, і на відповідному виході матимемо низький рівень сигналу.

Для створення заряду у міжшаровому проміжку на затвор транзистора необхідно подавати напругу, що значно перевищує напругу зчитування. Для того, щоб записати нову інформацію, раніше записана інформація має бути попередньо стертою.

Іншим типом транзисторів, які широко використовується при побудові сучасних репрограмованих ПЗП, є *транзистори МОН з лавинною інжекцією заряду* (ЛІЗМОН), які відрізняються від інших тим, що мають так званий *плаваючий затвор* (ПЗ) (рис. 8.14). У мікросхемах пам'яті зі стиранням інформації шляхом ультрафіолетового опромінення області затвора транзистора (*EPROM*) він є єдиним. У мікросхемах з електричним стиранням записаної інформації (*EEPROM*) він є допоміжним, на нього наноситься заряд, що призводить до підвищення величини порогової напруги транзистора.

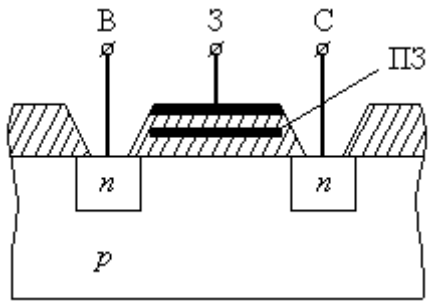


Рис. 8.14

Оскільки плаваючий затвор ізольований з усіх сторін, то нанесений на нього заряд може зберігатись протягом тривалого часу. Запис інформації забезпечується тим, що при подачі на затвор, витік та стік високої позитивної напруги по відношенню до підкладки, у зворотно-зміщених p-n- переходах

створюється лавинний пробій з високою концентрацією і енергією електронів. Плаваючий затвор розміщується так, що частина електронів, енергія яких перевищує визначений рівень, долають діелектричний бар'єр шару діелектрика і потрапляють на нього. Цим створюється постійний заряд плаваючого затвору, що збільшує порогову напругу транзистора і може зберігатись десятки років.

При подачі високого рівня напруги на фіксований затвор у транзисторі з плаваючим затвором канал між витоком та стоком не створюється, і цим забезпечується запис логічного нуля.

Стирання інформації, записаної в мікросхему пам'яті, яку виготовлено на основі ЛІЗМОН, базується на створенні умови для стікання заряду з плаваючого затвору. Це досягається одним з двох шляхів, що раніше згадувались. Перший з них передбачає наявність в мікросхемі спеціального віконця, через яке опромінюється напівпровідникова структура. Оскільки вибрані ізоляційні матеріали є прозорими для ультрафіолетового опромінення, то створені в приладі фотоструми, а також теплові струми забезпечують стікання заряду. Така процедура стирання займає десятки хвилин, але забезпечує одночасне стирання інформації по всьому кристалу. Оскільки ультрафіолетове опромінення призводить до зміни властивостей матеріалу, то кількість циклів перепрограмування обмежується десятками разів.

Спосіб електричного стирання забезпечується подачею на стоки транзисторів високої напруги при низькому рівні напруги на затворах. При цьому створюються умови для стікання (витіснення) заряду з плаваючого затвора на стік.

Переваги електричного способу стирання суттєві. Перш за все, значно збільшується швидкість стирання, яка у сучасних мікросхемах відноситься до циклу запису і не перевищує декількох десятків мілісекунд. Електричний спосіб стирання не впливає на структуру напівпровідника та ізоляційного матеріалу, тому кількість циклів перезапису зростає до 10^6, а інтервал часу зберігання даних – до 100 років.

З’являються нові можливості при роботі з мікросхемами пам’яті. Можливо здійснювати вибірковий перезапис – за окремими адресами або окремими сторінками. Оскільки в сучасних мікросхемах висока напруга для перезапису створюється безпосередньо в корпусах на основі ємнісних накопичувачів, то при цьому проблеми перезапису зводяться до задач керування і алгоритмів керування цими процесами. Мікросхеми пам’яті при цьому можуть залишатись на своєму місці у друкованій платі.

Суттєво розширюються можливості зберігання інформації, обмеження доступу до зчитування записаної інформації. З’явилась можливість використання режиму енергозбереження, яка полягає в тому, що живлення до модулів мікросхеми подається лише на інтервали часу, коли до мікросхеми йде звернення від зовнішніх пристроїв, подібно до ПЗП (на рис. 8.7 режим енергозбереження забезпечується підключенням живлення через входи *PO* (*Power On*) модулів при зверненні до мікросхеми по входу $\overline{CS}$).

8.2.2. Мікросхеми РПЗП

Мікросхеми EEPROM вітчизняного виробництва виготовлялись в серіях КР558 (КР558РР1…РР3), КР1601 (КР1601РР1…РР3), КР1609 (КР1609РР1…РР2). Порівняно з сучасними, вони мали низьку швидкодію і високу напругу програмування. Організація цих мікросхем (1…8) К × 8, тобто їх інформаційна ємність невелика.

Мікросхеми EPROM з ультрафіолетовим стиранням виготовлялись в серії 573 (РФ1…РФ9) з організацією (1…128) К × 8. На сучасному технологічному рівні виготовлення мікросхем пам’яті вони майже повністю витіснені мікросхемами EEPROM.

Як приклад, розглянемо особливості структурних схем, а також роботу мікросхем EEPROM фірми ATMEL, які широко використовуються при побудові мікропроцесорних систем різноманітного функціонального призначення як на основі контролерів цієї ж фірми, так і інших виробників.

8.2.2.1. Паралельні EEPROM

Прикладом паралельних EEPROM являються мікросхеми сім'ї AT28C фірми ATMEL.

Особливості приладів цієї сім'ї полягають у наступному:

- використана одна напруга живлення для всіх операцій;
- 8-розрядна організація;
- вбудовані фіксатори адреси та даних;
- вбудований автомат програмування;
- забезпечується побайтове і секторне програмування;
- типова кількість циклів програмування – не менш 10000;
- виконання як для побутової (діапазон робочих температур 0…+ 70° С), так і промислової (– 40…+ 85° С) апаратури.

Сім'я 8-розрядних EEPROM AT28C має діапазон інформаційних ємностей від 16 Кбіт до 4 Мбіт. В усіх режимах мікросхеми працюють від одного джерела живлення, що забезпечує просте внутрисистемне перепрограмування. Частина МС забезпечує побайтове програмування як одного (кількох) байтів, так і всього обсягу основної пам'яті, інші мають секторну організацію і за одну операцію програмуються як в межах одного сектора, так і декількох байтів. Програмування забезпечується вбудованим автоматом, не вимагає зовнішнього тактування і операцій попереднього стирання інформації. Звернення до пристроїв при операціях читання / запису аналогічне зверненню до статичних ОЗП. Мікросхеми містять вбудовані регістри на весь об'єм даних – байт або сторінку в залежності від типу приладу. На час циклу запису адреса і дані фіксуються вбудованими регістрами-фіксаторами, звільнюючи шину для інших операцій керуючого процесора. Закінчення циклу запису може бути виявлено опитуванням виходу $\overline{DATA}$ і опитуванням біту чергування або за станом виходу $RDY/\overline{BUSY}$ (у залежності від типу МС). Функції МС розширені додатковими можливостями забезпечення високої якості та технологічності – застосована вбудована корекція помилок для підвищення надійності

збереження даних. Додатковий механізм програмного захисту даних зберігає дані від випадкового перепрограмування. У приладах сім'ї виділена особлива область пам'яті ємністю від 32 до 128 байт для зберігання користувальницьких даних ідентифікації.

Одним з представників сім'ї AT28C з паралельною адресною організацією вводу-виводу є EEPROM AT28C64 та її модифікації для різних використань з організацією 8K × 8 ємністю 64K. Умовне позначення мікросхеми приведено на рис. 8.15.

Призначення виводів приводиться у табл. 8.6.

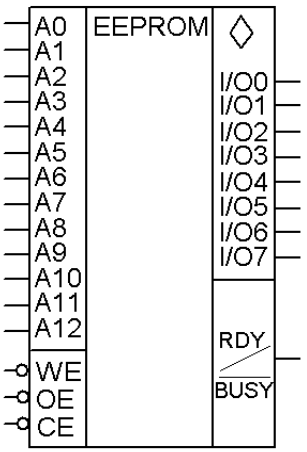


Рис. 8.15

Таблиця 8.6

Назва виводу	Призначення
$A_0 \dots A_{12}$	*Address Inputs* (адресні входи)
$\overline{CE}$	*Chip Enable* (пристрій активований)
$\overline{OE}$	*Output Enable* (вихід дозволено)
$\overline{WE}$	*Write Enable* (запис дозволено)
$I/O_0 \dots I/O_7$	*Data Inputs / Outputs* (інформаційні входи/виходи)
$RDY/\overline{BUSY}$	*Ready*/$\overline{Busy}$ *Output* (вихід готовності/зайнятості)

Мікросхема має час доступу до даних (час зчитування), що не перевищує 120 нс, а інтервал часу запису коливається у межах від 200 мкс до 1 мс. Цикли запису та зчитування забезпечуються сигналами мікропроцесора без допоміжних елементів.

Оскільки інтервал часу запису набагато більший, ніж часу читання, то при записі даних передбачено, що код адреси і дані попередньо запам'ятовуються, звільнюючи шини процесора для іншого використання. Після ініціалізації циклу запису мікросхема переходить у стан зайнятості, і перезапис введених даних забезпечується за допомогою внутрішнього керуючого таймера. Для визначення кінця циклу запису використовуються два способи. Перший забезпечується рівнем сигналу на виході $RDY/\overline{BUSY}$, а другий – рівнем

сигналу на I/O_7. Мікросхема має режим енергозбереження. Споживаний струм в активному режимі не перевищує 30 мА при напрузі живлення 5 В. У режимі енергозбереження величина споживаного струму обмежується величиною 100 мкА. Мікросхеми фірми ATMEL забезпечують внутрішню корекцію даних (див. *"Корекція за Хемінгом"*, **Розділ 3**), що дає можливість підвищити надійність і покращити якість даних, які зберігаються.

Блок-схема мікросхеми EEPROM AT28C64 приведена на рис. 8.16.

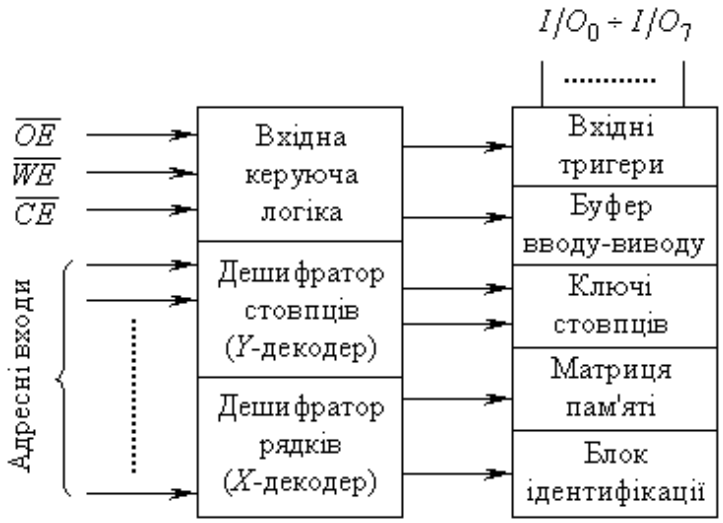


Рис. 8.16

Нижче описані режими роботи мікросхеми.

Режим зчитування. AT28C64 має доступ до читання даних, подібно статичним ОЗП. Коли виводи $\overline{CE}$ і $\overline{OE}$ мають низький рівень напруги, а $\overline{WE}$ – високий, - дані, що зберігаються в пам'яті, зчитуються у відповідності до адресного коду на виходи $I/O_0 \ldots I/O_7$. Виходи переходять у *Z*-стан, коли будь-який з входів $\overline{CE}$ або $\overline{OE}$ отримає високий рівень сигналу. Така подвійна лінія керування надає більшої гнучкості розробникам у попередженні виникнення конфліктів ("гонок") по шині даних.

Запис байту. Запис даних в AT28C64 також подібний до статичних ОЗП. Низький рівень сигналу на входах $\overline{WE}$ та $\overline{CE}$ з високим рівнем сигналу на $\overline{OE}$ ініціює запис байту. Адресний код фіксується за зрізом сигналу на $\overline{WE}$ або $\overline{CE}$.

Внутрішня структура мікросхеми самоочищення пам’яті перед записом і протягом циклу запису t_{WC} видає низькорівневий інформаційний сигнал на відповідний вихід (I/O_7 або $RDY/\overline{BUSY}$).

У мікросхемі передбачений захист від несанкціонованого запису даних, функції запису забороняються при зниженні напруги живлення до величини, меншої ніж 3,8 В. Забороняється запис в інтервалі 5 мс після подачі живлення.

Очищення пам’яті. Повне очищення пам’яті забезпечується установкою $\overline{CE} = 0$ і подачею на $\overline{OE}$ напруги 12 В. Мікросхема очиститься від вмісту протягом 10 мс після подачі на $\overline{WE}$ низького рівня сигналу.

Точні часові співвідношення між сигналами керування процесами запису і зчитування можна знайти в технічній документації на відповідну мікросхему, де приводяться часові діаграми з можливими допусками.

Враховуючи велику різницю між циклами запису та зчитування, з метою збільшення обсягу інформації, що записується за час циклу, розроблені EEPROM з паралельною формою запису даних сторінкового формату. Прикладом таких мікросхем є ATMEL EEPROM AT18C64B інформаційною ємністю 64K (8K × 8). Особливість цієї мікросхеми, порівняно з попередньою, полягає в наявності режиму запису сторінки (*PAGE WRITE*) даних обсягом від 1 до 64 байт. Цей об’єм даних записується протягом одного внутрішнього програмного циклу запису. Ініціалізація режиму запису сторінки відбувається тією ж групою сигналів і в тій же послідовності, що й режим запису байту. Після першого байту, що поданий для запису, кожний наступний подається в інтервалі часу 150 мкс після подачі попереднього. Після завантаження сторінки внутрішня логіка зупиняє подальший прийом даних і продовжує внутрішню процедуру запису. Всі байти під час запису сторінки повинні розміщуватись у розмірах тієї ж сторінки, яка з початку запису задана кодом старших адресних розрядів $A_6 \ldots A_{12}$. Адресний код, що подається на входи $A_0 \ldots A_5$, визначає адреси записаних байтів усередині сторінки. В рамках сторінки байти можуть бути завантажені в будь-якому порядку.

У мікросхемі AT28C64B передбачені контроль розміру сторінки даних і індикація кінця циклу запису для забезпечення якості запису інформації. До того ж, у мікросхемі використані апаратні та програмні засоби захисту даних. Апаратні засоби з невеликим розширенням практично повторюють засоби, застосовані в попередній мікросхемі. Програмні засоби захисту даних (*SDP – Software Data Protection*) призначені для упередження ненавмисного запису. Ці програмні засоби є внутрішнім алгоритмом роботи мікросхеми і користувачем можуть не використовуватись.

У табл. 8.7 приведені основні параметри мікросхем сім'ї AT28C.

Таблиця 8.7

Тип приладу	Ємність, біт	Організація	Обсяг сторінки, байт	Час вибірки, нс	Тривалість циклу запису байту {сторінки}, мкс	I_{AC}, мА	I_{SB}, мА
AT28C16-T	16 К	2К x 8			1	30	0,1
AT28C64 AT28C64E	64 К	8К x 8		120	1 0.2	30	0,1
AT28C256F	256 К	32К x 8	64		{3}	50	0,2
AT28C010 AT28C010E	1 М	128К x 8	128	150	{10}	40	0,2
AT28C040	4 М	512К x 8	256	280	{10}	80	0,3

Примітки:

1*. Споживання струму в активному режимі приладів промислового виконання складає 45 мА.

2*. Індекс "E" присвоєно приладам з кількістю циклів перезапису 100 000.

8.2.2.2. Послідовні EEPROM

Серед мікросхем пам'яті все більшою популярністю користуються мікросхеми з послідовним протоколом запису та зчитування даних. Це мікросхеми з двохпровідною шиною типу IIC (I^2C) та з трьохпровідною шиною типу MICROWARE (SPI).

У табл. 8.8 приведені дані деяких типів мікросхем з послідовним протоколом обміну даними, що виготовляються фірмами ATMEL та MICROCHIP.

Розглянемо більш детально особливості архітектур і організації обміну інформації в мікросхемах з різними типами шин. Умовне позначення мікросхем з шиною I^2C (AT24C) приведені на рис. 8.17.

Призначення виводів МС приведено у табл. 8.9.

Таблиця 8.8

Шина **I^2C**			Шина **Microware (SPI)**		
Типи мікросхем, фірми		Організація	Типи мікросхем, фірми		Організація
ATMEL	MICROCHIP		ATMEL	MICROCHIP	
24C01	85C72 24C01A 24LC01		93C06		16 × 6
24C02 24LLC02	24C02	256 × 8	93C46	93C46 93LC46	64 × 16
24C04A 24LC04	24C04	2 × 256 × 8	93C46	93LC46	128 × 8
	24C08	4 × 256 × 8	93C56	93C56	128 × 16
24C16 24LC16	24C16	8 × 256 × 8	93C56	93C56 93LC56	256 × 8
			93C66	93C66 93LC66	256 × 16

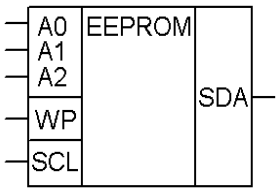


Рис. 8.17

Таблиця 8.9

Назва виводу	Призначення
$A_0 \ldots A_2$	*Address Inputs* (адресні входи)
SDA	*Serial Data* (дані в послідовному форматі)
SCL	*Serial Clock Input* (вхід тактового генератора)
WP	*Write Protect* (захист від запису)

Блок-схема мікросхеми приводиться на рис. 8.18.

У доповнення до вже відомих блоків (матриці пам'яті, дешифраторів), маємо ряд допоміжних, які забезпечують описані нижче протоколи обміну інформації у послідовному форматі через вхід *SDA*.

Старт-стопна логіка (*START STOP LOGIC*) призначена для формування сигналів початку й кінця обміну в залежності від характеру обміну.

Компаратор вхідної адреси (*DEVICE ADDRESS COMPARATOR*) порівнює молодші розряди адресного коду вибору мікросхеми з внутрішнім кодом.

Якщо коди співпадають, то мікросхема генерує сигнал підтвердження низького рівня на вході *SDA* за допомогою блоку логіки вихідних сигналів. Якщо ж коди не співпадають, то вона переходить у режим енергозбереження.

Лічильник слова даних і адреси (*DATA WORD ADDR / COUNTER*) виконує функцію генератора внутрішніх адрес даних при посторінковій формі запису та зчитуванні.

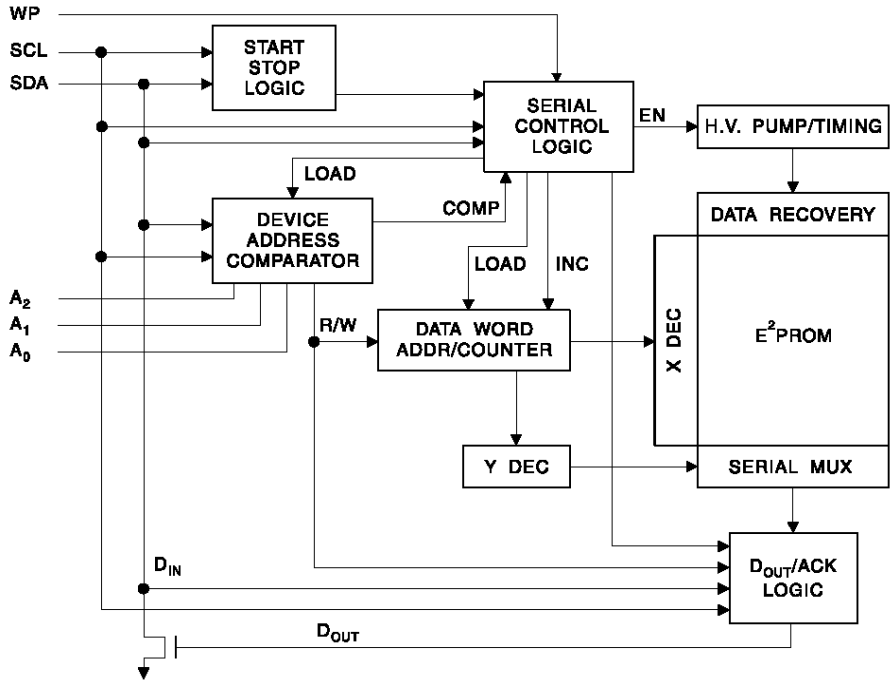


Рис. 8.18

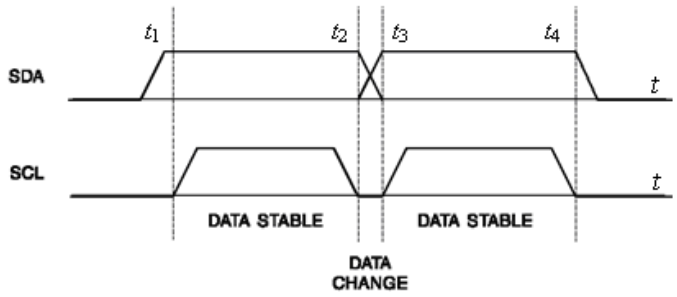
SCL – вхід, який використовується як для забезпечення синхронізації, так і для визначення напряму передачі даних по шині *SDA*. За фронтом сигналу дані записуються в мікросхему пам’яті, а за зрізом – читаються.

Вхід *SDA* використовується для двонаправленої передачі даних. Цей вивід з відкритим стоком може використовуватися для з’єднання декількох мікросхем за схемою монтажного **АБО**.

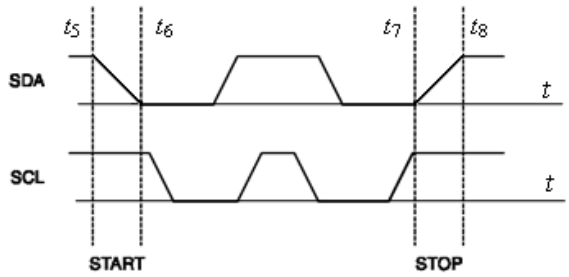
Взаємодія між сигналами на входах *SCL* і *SDA* забезпечує всі основні режими роботи мікросхеми пам’яті.

Прийом і передача даних по входу *SDA* забезпечується лише при умові, коли на вході *SCL* маємо сигнал низького рівня (рис. 8.19, *а*, інтервал часу $t_2 \dots t_3$). Більш точно умова початку обміну інформацією пояснюється на рис. 8.19, *б*, тобто перехід сигналу з високого рівня в низький на вході *SDA* при підтримці високого рівня на вході *SCL* забезпечує умови початку обміну (інтервал часу $t_5 \dots t_6$).

Умова закінчення обміну забезпечується переходом з низького рівня на високий на вході *SDA* при підтримці високого рівня на вході *SCL* (інтервал часу $t_7 \ldots t_8$).



а)



б)

Рис. 8.19

WP – вхід, що дозволяє захищати дані. При *WP* = 0 мікросхема працює у режимі запису / читання, якщо ж *WP* = 1, то перезапис інформації неможливий.

При обміні інформацією наступним за умовою початку обміну передається байт адресного коду, який містить у собі незмінну частину в чотирьох старших бітах, як зображено на рис. 8.20, однакову для мікросхем ємністю від 1K до 16K.

Наступними 3 бітами можуть бути адреси, за якими розпізнається мікросхема шляхом порівняння з внутрішнім кодом або адреси сторінок пам’яті в EEPROM.

	7	6	5	4	3	2	1	0
1K/2K	1	0	1	0	A_2	A_1	A_0	*R/W*
4K	1	0	1	0	A_2	A_1	P_0	*R/W*
8K	1	0	1	0	A_2	P_1	P_0	*R/W*
16K	1	0	1	0	P_2	P_1	P_0	*R/W*

Рис. 8.20

Наймолодший біт адресного коду визначає тип операції, що виконується над операндом, який поступає вслід за адресою його розміщення. Це операції запису або читання (високий рівень – операція читання; низький – запис). Якщо адресний код прийнятий мікросхемою пам’яті, то вона видає на 9-му такті сигнал низького рівня як сигнал підтвердження; якщо ж адресний код не фіксується , то МС переходить у режим енергозбереження.

Для виконання операції запису необхідно вслід за кодом адреси вибору мікросхеми, отримавши низькорівневий сигнал підтвердження, відправити адресу розміщення слова даних. На рис. 8.21 приводиться послідовність сигналів, що має шина (лінія) *SDA.*

Рис. 8.21

Запис даних у послідовних EEPROM фірми ATMEL може виконуватися сторінками. Сторінки в EEPROM ємністю 1K; 2K мають розмір 8 байт, а в мікросхемах ємністю 4K; 8K; 16K – відповідно, 16 байт.

Послідовність запису сторінки мало чим відрізняється від послідовності запису одного байту. Різниця полягає у тому, що по завершені передачі слова мікроконтролер не передає стоповий сигнал, а, отримавши сигнал підтвердження (9-й такт), повинен передати послідовно чергові 7 сигналів (для 1K; 2K) або 15 (для 4K; 8K; 16K). Після отримання кожного слова EEPROM генерує на 9-му такті сигнал підтвердження (див. рис. 8.21). Під час запису сторінки адреси розміщення байтів сторінки формуються шляхом інкрементування молодших трьох розрядів адреси слова даних для EEPROM ємністю 1K; 2K і чотирьох розрядів для мікросхем ємністю 4K; 8K; 16K. Формування внутрішніх адрес таким шляхом забезпечується внутрішнім

лічильником (див. блок-схему рис. 8.18). Якщо більш ніж 8 (16) слів будуть подаватися при записі, то адреси циклічно повторюватимуться, перезаписуючи попередні дані.

Операція читання близька до операції запису з тією лише різницею, що в молодшому біті адреси мікросхеми стоятиме не **“0”**, а **“1”.**

При використанні мікросхем пам’яті мають місце три способи зчитування даних: *читання за поточною адресою*; *послідовне зчитування*; *зчитування за довільною адресою*.

Внутрішній лічильник формує адреси даних по черзі їх читання шляхом інкрементування на одиницю. Ці адреси залишаються активними між операціями зчитування. “Прокручування” адрес при читанні має місце від останнього байта останньої сторінки до першого байта першої сторінки; а при операції запису – з останнього байта поточної сторінки до першого байту тієї ж сторінки.

На рис. 8.22 приводиться послідовність сигналів на вході *SDA* при зчитуванні даних поточної адреси.

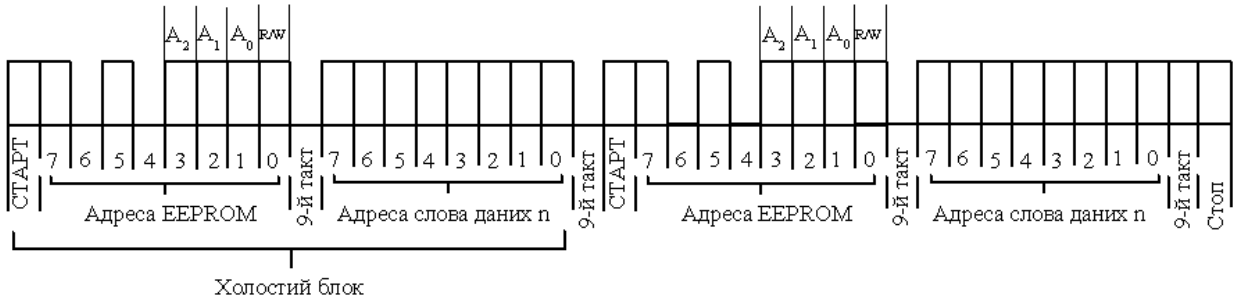


Рис. 8.22

Після прийому байту адреси мікросхеми з високим рівнем нульового біту йде сигнал підтвердження на 9-му такті, після чого послідовно зчитується байт даних. Після зчитування на 9-му такті сигнал підтвердження низького рівня не з’являється, а контролер повинен сформувати сигнал завершення обміну.

При довільному зчитуванні необхідно спочатку сформувати холостий блок сигналів з сигналом запису за заданою адресою. Послідовність сигналів на *SDA* приводиться на рис. 8.23.

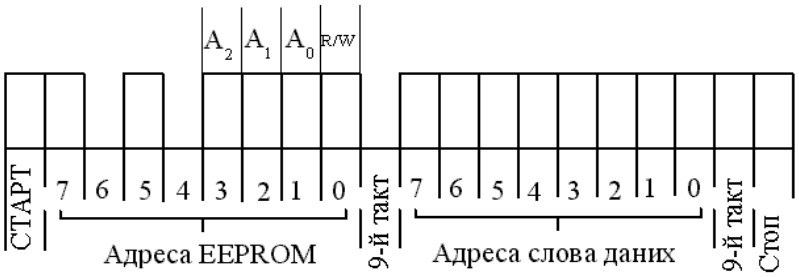


Рис. 8.23

Після отримання сигналу підтвердження контролер повинен знову згенерувати умову початку обміну “старт”, повторити адресу мікросхеми з умовою зчитування і, після підтвердження, прийняти байт даних. Мікросхема пам’яті після зчитування не генерує сигнал підтвердження, а контролер видає сигнал кінця обміну “стоп”.

При послідовному зчитуванні контролером спочатку ініціюється початкова адреса послідовності, яка зчитується, згідно з одним з вище описаних способів. Після отримання адресного слова контролер формує сигнал підтвердження, після чого в EEPROM лічильник генерує наступну адресу, за якою зчитуватиметься наступне слово. Після цього контролер знову генерує сигнал підтвердження до того часу, доки не буде сформований сигнал підтвердження на черговому 9-му такті, а на наступному такті сформує сигнал завершення обміну “стоп”. Послідовність сигналів на лінії *SDA* приводиться на рис. 8.24.

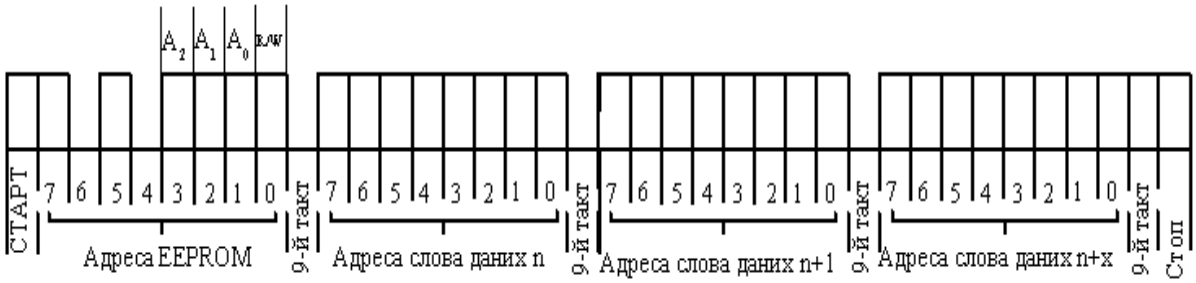


Рис. 8.24

Мікросхеми пам’яті AT24C01…AT24C16 орієнтовані на роботу з контролерами AT28CX051, для яких розроблене програмне забезпечення, сумісне з розглянутими EEPROM.

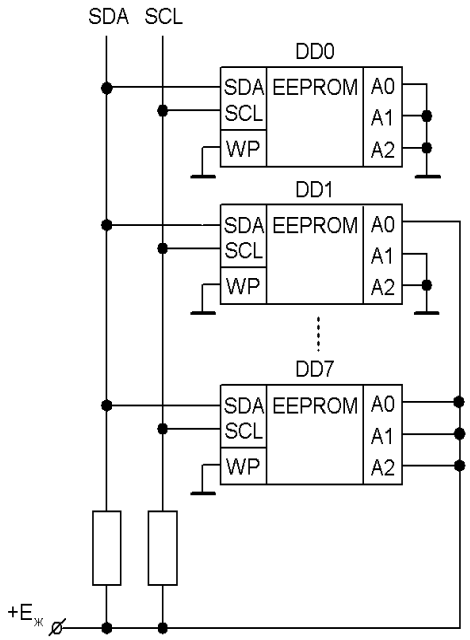


Рис. 8.25

Функціональна схема побудови модуля пам'яті з використанням мікросхем AT24C01A приводиться на рис. 8.25. Адресні входи $A_0 \ldots A_2$ використовуються для апаратного запису адреси кожної мікросхеми, які розподіляються в діапазоні значень 000…111.

Шина обміну має два провідники, до яких паралельно приєднані виводи *SDA* і *SCL*. Порядок взаємодії між сигналами на цих провідниках і їх послідовність, що називається *протоколом обміну*, задається контролером, до якого і приєднуються виводи шини.

8.2.2.3. EEPROM з трьохпровідною послідовною шиною

Типовими представниками таких мікросхем пам'яті виступають EEPROM фірми ATMEL AT93C46, AT93C56, AT93C57, AT93C66. Умовне позначення мікросхеми приводиться на рис. 8.26, а призначення виводів – у табл. 8.10.

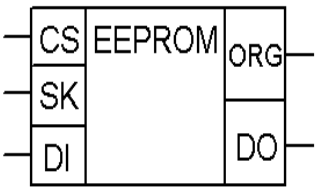


Рис. 8.26

Таблиця 8.10

Умовні позначення	Призначення
CS	*Chip Select*
SK	*Serial Data Clock*
DI	*Serial Data Input*
ORG	*Internal Organization*
DO	*Serial Data Output*

Мікросхеми можуть бути організовані як 64/128/256 16-розрядних слів при *ORG* = 1, або як 128/256/512 однобайтних слів при *ORG* = 0. Доступ до мікросхем цієї групи забезпечується попереднім вибором мікросхеми (вхід *CS*) і забезпечується по трьохпровідній шині у послідовному форматі через виводи *DI, DO* та *SK*. Блок-схема мікросхеми пам'яті приводиться на рис. 8.27.

Матриця пам'яті (*MEMORY ARRAY*) шляхом зміни рівня сигналу на вході *ORG* може переналагоджуватись на запис і читання слів довжиною в

1 або 2 байти. Організація пам'яті адресна, і доступ до елементів пам'яті забезпечується адресним кодом, який подається через вхід *DI* у послідовному форматі і декодується в модулі декодування.

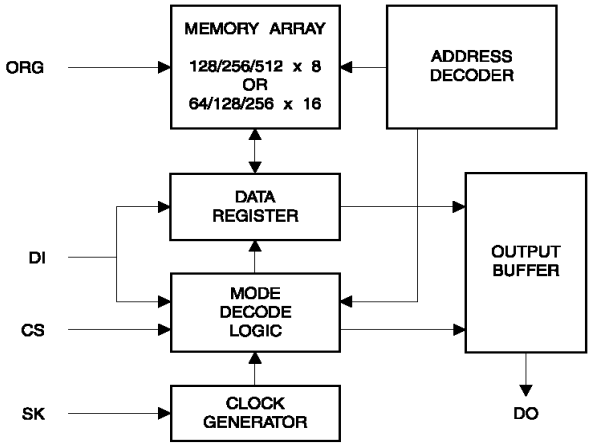


Рис. 8.27

Вхідна шина забезпечує гнучке керування мікросхемою за допомогою сімох команд, які подаються на зазначенні вище виводи (*DI*, *DO*, *SK*) від контролера. Дія команд починається за фронтом сигналу, що подається на вхід *CS* і містить у собі стартовий біт (*SB* – **лог. "1"**), відповідний код операції (*Operation Code*) *OC* розміром в 2 біти і необхідну адресу доступу до пам'яті.

Постійні кодові комбінації для кожної з команд, як приклад, для мікросхеми AT93C46 приведені у табл. 8.11.

Команда *READ* містить у собі адресний код тих байтів, які повинні зчитатись. Після того, як команда декодується в модулі декодуючої логіки (*MODE DECODE LOGIC*), послідовний код адреси перетворюється у паралельний у модулі декодування адреси (*ADDRESS DECODER*) і подається на дешифратори рядків та стовпців.

Дані з вибраних елементів пам'яті у послідовному форматі будуть передані через регістр даних (*DATA REGISTER*) і вихідний буфер (*OUTPUT BUFFER*) на вихід *DO*. Вихідні дані синхронізуються за фронтом синхросигналів, поданих на вхід *SK*. Вихідні дані 8-ми чи 16-ти бітного формату упереджує "холостий" біт низького рівня.

Таблиця 8.11

Команда	*SB*	*OC*	Адреси		Дані		Коментарі
			× 8	× 16	× 8	× 16	
READ	1	10	$A_6 \dots A_0$	$A_5 \dots A_0$			Читання даних, що зберігаються в пам'яті за визначеними адресами
EWEN	1	00	11xxxxx	11xxxx			Забезпечення стирання перед записом
ERASE	1	11	$A_6 \dots A_0$	$A_5 \dots A_0$			Стирання пам'яті за адресами
WRITE	1	01	$A_6 \dots A_0$	$A_5 \dots A_0$	$D_7 \dots D_0$	$D_{15} \dots D_0$	Запис пам'яті за адресами
ERAL	1	00	10xxxxx	10xxxx			Стирання всієї пам'яті
WRAL	1	00	01xxxxx	01xxxx	$D_7 \dots D_0$	$D_{15} \dots D_0$	Запис всієї пам'яті
EWDS	1	00	00xxxxx	00xxxx			Заборона всіх команд

Часові діаграми процедури читання приведені на рис. 8.28.

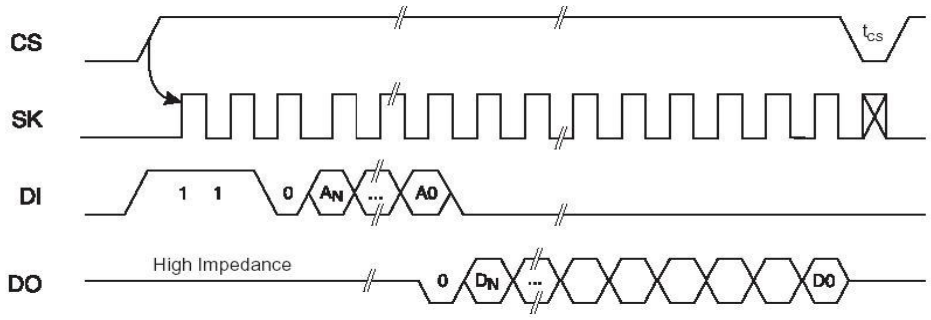


Рис. 8.28

Команда *ERASE* забезпечує стирання всіх бітів в області пам'яті, що визначена адресами. Вона переводить всі елементи пам'яті в стан логічної "**1**". Операція стирання здійснюється протягом циклу, що задається внутрішнім генератором одразу після того, як адреса і команда декодуються у відповідних модулях. Поява сигналу на виході *DO* відображає, що операція стирання завершена і мікросхема готова до прийому нової команди.

Команда *ERAL* (*ERase ALl*) забезпечує запис у кожен елемент пам'яті матриці сигналу логічної "**1**". Вона часто використовується для тестування пам'яті.

Команда *WRITE* містить у собі 8 або 16 біт даних, що записуються за визначеними адресами. Часові діаграми сигналів на шині при виконанні команди приведені на рис. 8.29.

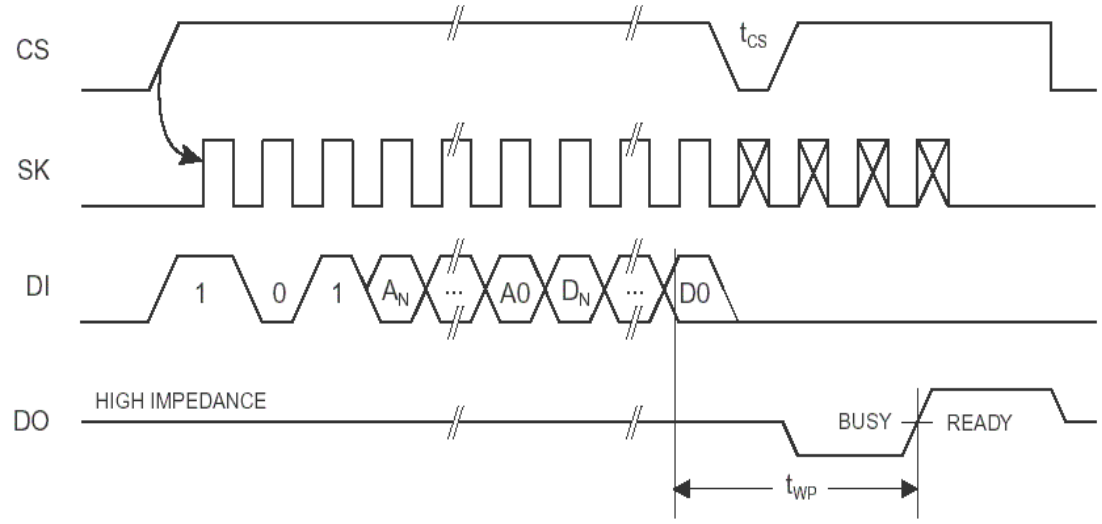


Рис. 8.29

Програмний цикл запису даних починається після отримання у послідовному форматі даних через вхід *DI* і триває протягом інтервалу часу t_{WP}. Протягом цього циклу, якщо матиме місце звернення до мікросхеми по входу *CS*, на виході *DO* з'явиться сигнал низького рівня, який інформує про зайнятість її у режимі запису. Якщо звернення до мікросхеми буде тривати, то по закінченню циклу запису на виході *DO* з'явиться сигнал високого рівня, що інформує про готовність мікросхеми до чергового обміну.

Команда *WRAL* (*WRite ALl*) програмує всю пам'ять мікросхеми за зразком, що визначений в інструкції. Часові діаграми сигналів на шині при виконанні команди приведені на рис. 8.30.

Команди *EWEN* (*Erase/Write ENable*) і *EWEDS* (*Erase/Write DiSable*) призначенні для забезпечення гарантованого зберігання даних та їх запису при подачі напруги живлення, при зміні команд.

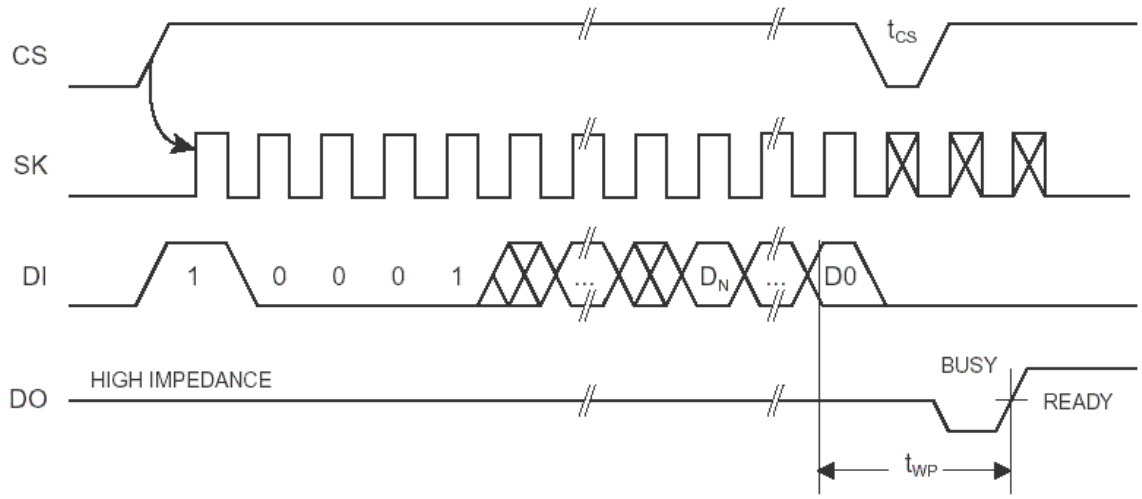


Рис. 8.30

Мікросхеми AT93 сумісні з контролерами, що працюють з частотою 2 МГц, мають цикл запису t_{WP} = 1 мс, витримують до 1 млн. циклів запису, тривалість зберігання даних до 100 років.

При використанні мікросхеми з контролерами вхід *ORG* повинен бути приєднаним до загальної шини або шини живлення в залежності від формату даних. Вона може використовуватись як з чотирьохпровідною шиною обміну з контролером (*CS, SK, DI, DO*), так і з трьохпровідною. В останньому випадку вхід *DI* об'єднується з входом *DO* і розглядається з боку контролера як один провідник з двонаправленої шиною.

8.2.3. Флеш-пам'ять

8.2.3.1. Основи побудови флеш пам'яті

Запам'ятовуючі пристрої, що мають назву *флеш-пам'яті* (*Flash memory*), побудовані подібно до EEPROM, але окремі архітектурні і експлуатаційні властивості обумовили необхідність її розгляду як окремої групи репрограмованих ПЗП. Головною особливістю флеш-пам'яті є те, що стирання записаної в неї інформації забезпечується не окремими словами (байтами), а в цілому для всієї мікросхеми або блоками досить великих розмірів. Це дає можливість суттєво спростити схемотехніку, підвищити інформаційну ємність і швидкість.

Одночасне стирання всієї інформації РПЗП має і свої недоліки. Головний з них полягає в тому, що заміна навіть одного слова у флеш-пам'яті вимагає стирання і нового перезапису всієї інформації. Оскільки такий недолік не допустимий для великої кількості задач, то, поряд із флеш-пам'яттю з загальним стиранням (*BULK ERASE*) пам'яті, були розробленні мікросхеми на основі блочної структури, в яких весь об'єм пам'яті ділиться на блоки, що стираються незалежно один від іншого.

Розвиток архітектур флеш-пам'яті йде двома основними напрямками. Перший з них розвиває мікросхеми пам'яті для зберігання даних і програм, які не дуже інтенсивно змінюються – наприклад, у мобільних телефонах, модемах, BIOS персональних комп'ютерів, системах керування автомобільними двигунами. Такі дані часто називають *параметричними*. Для цього напрямку, в зв'язку з порівняно рідким оновленням вмісту, параметри циклів стирання та запису менш суттєві, порівняно з інформаційною ємністю та швидкістю зчитування. В пристроях пам'яті цього типу використовують як загальне стирання, так і поблочне. Пам'ять з параметричними блоками використовується для зберігання телефонних номерів, обліку часу, використання у *SIM*-картках, для зберігання кодів помилок та параметрів оптимальних режимів роботи та ін.

Серед мікросхем з поблочним стиранням слід виділити мікросхеми з спеціалізованими блоками, які призначені для зберігання програм ініціалізації системи, тобто програм, що вводять систему у робочій стан після вмикання живлення. Такі блоки називають *блоками завантаження* (*Boot Blocks*), а мікросхеми з такими блоками мають назву *Boot Block Flash Memory*. У *Boot Block* зберігаються також програми для програмування та стирання флеш-пам'яті.

Мікросхеми, що використовують *Boot Block*, здебільшого мають несиметричну структуру. Приблизний вигляд карти пам'яті, приведений на рис. 8.31 (для мікросхеми ємністю 4 Мбіт).

3FFFFH 3E000H	**16K BYTE BOOT BLOCK**
3DFFFH 3D000H	**8K BYTE PARAMETER BLOCK**
3CFFFH 3C000H	**8K BYTE PARAMETER BLOCK**
3BFFFH 30000H	**96K BYTE PARAMETER BLOCK**
2FFFFFH 20000H	**128K BYTE MAIN BLOCK**
1FFFFH 10000H	**128K BYTE MAIN BLOCK**
0FFFFH 00000H	**128K BYTE MAIN BLOCK**

Рис. 8.31

Блок (*BLOCK*) – це область пам’яті з фіксованим діапазоном адресних кодів. При стиранні блоку пам’яті стираються всі елементи пам’яті всередині блоку. При цьому інформація в інших блоках залишається незмінною. Оскільки флеш-пам’ять не допускає перезапису окремих слів, а тим більше окремих біт без попереднього стирання всього блоку пам’яті, то для розв’язання вказаних задач використовуються програмні методи емуляції перезапису байта з використанням двох параметричних блоків ємністю по 8 Кбайт, які показані на рис. 8.31.

У *блоках параметрів* (*PARAMETER BLOCK*) зберігаються дані та програми, що змінюються відносно часто, – наприклад, діагностичні програми, коди ідентифікаторів і т. п. В *основних блоках* (*MAIN BLOCK*) зберігаються основні дані. Різниця між блоками полягає також у тому, що інформація, записана у *BOOT BLOCK*, апаратно захищена від випадкового стирання.

Окрім мікросхем з несиметричною картою пам’яті, виготовляються мікросхеми і з симетричним розміщеними *BOOT*-блоками на початку та в кінці адресного простору, що забезпечує симетричне зчитування даних як по зростанню адресного коду, так і по його зменшенню.

Другий напрямок розвитку флеш-пам’яті – це мікросхеми з великою ємністю (64…512) М, які використовуються для зберігання значних обсягів

інформації, тобто для заміни носіїв на магнітних дисках. Такі пристрої пам’яті містять у собі більш розвинені засоби перезапису інформації і побудовані на основі симетричних карт. До такого типу відноситься файлова система флеш-пам’яті, яка в останні роки все більше замінює дискові запам’ятовуючі пристрої, оскільки має меншу потужність споживання, зберігаючи при цьому повну сумісність з існуючими засобами керування пам’яттю. Блочна структура файлових пристроїв флеш-пам’яті є аналогом секторів магнітних дисків. Тому ідеологія обміну даними між секторами жорстких дисків дала можливість розробити програми, що забезпечують обмін між флеш-блоками. Оскільки організація обміну між оперативною пам’яттю та жорсткими дисками побудована на сторінковій основі, то для організації швидкого запису інформації у файлових флеш-пристроях передбаченні сторінкові буфери значної ємності, які дозволяють на високій швидкості сприймати деякі обсяги інформації, які потім перезаписується у блоки основної пам’яті з меншою швидкістю. Для файлових пристроїв пам’яті розроблені програми-драйвери, які дозволяють легко узгодити різноманітні апаратні засоби.

Ще одним напрямом розвитку флеш-пам’яті є мікросхеми, які називаються *Strata Flash* і зберігають в одному елементі пам’яті не один, а два біти. Нова якість обумовлена значним підвищенням точності інтегральних технологій, які дозволили у транзисторі з плаваючим затвором фіксувати не тільки наявність чи відсутність заряду, а й встановлювати його величину. На сучасному рівні розвитку технології чітко можуть фіксуватись чотири рівні заряду, що й забезпечує зберігання в одному елементі пам’яті два біти. Використовуючи багаторівневі компаратори та джерела еталонного струму можна визначити один з чотирьох рівнів струму транзистора, який описується двохрозрядним двійковим кодом у межах 00…11.

Оскільки зберігання двох бітів інформації здійснюється практично в тих же запам’ятовуючих пристроях, це дало можливість удвічі збільшити інформаційну ємність флеш-пам’яті на одному кристалі.

Сучасні мікросхеми флеш-пам’яті виготовляються за технологією, яка називається *Smart Voltage*, що дозволяє використовувати єдину напругу програмування і живлення модулів, забезпечуючи при цьому високу швидкодію мікросхем при зчитуванні інформації, високу надійність запису і найшвидший спосіб програмування. Ряд мікросхем, як приклад, при напрузі живлення 5 В забезпечують зчитування даних за 60 нс, а запис – за 13 мкс.

8.2.3.2. Мікросхеми флеш-пам’яті

Мікросхеми з паралельним доступом. Мікросхеми серії AT29C були спеціально розроблені для розв’язання задач оновлення невеликих обсягів інформації в діапазоні від 64K до 4M – як програм, так і даних – при живленні однією з напруг ряду 5 В; 3 В; 2,7 В.

Особливості архітектури і використання мікросхем даної серії розглянемо на прикладі мікросхеми AT29C256 ємністю 256K з організацією 32K × 8.

Мікросхеми мають час доступу на зчитування, що не перевищує 70 нс, а час програмування сторінки – не більше 10 мс. Кожен з секторів може репрограмуватись не менш ніж 10 000 разів.

Спрощена архітектура мікросхеми приведена на рис. 8.32, а на рис. 8.33 приводиться умовне позначення мікросхеми.

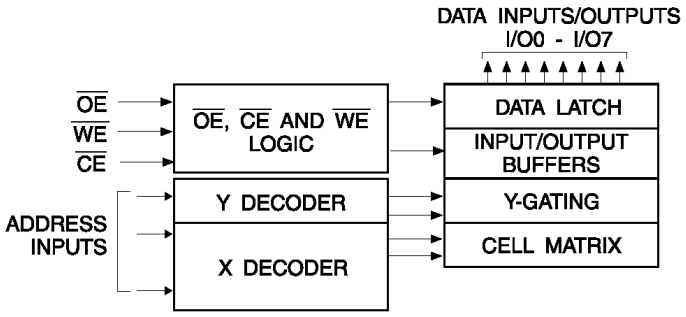


Рис. 8.32

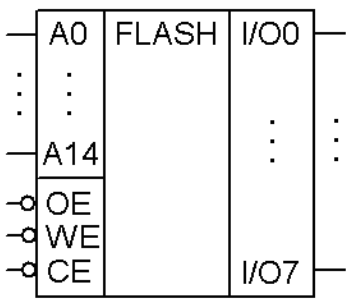


Рис. 8.33

У табл. 8.12 пояснюється призначення виводів.

Таблиця 8.12

Назва виводу	Призначення
$A_0 \dots A_{14}$	*Address Inputs* (адресні входи)
$\overline{CE}$	*Chip Enable* (пристрій активізовано)
$\overline{OE}$	*Output Enable* (вихід дозволено)
$\overline{WE}$	*Write Enable* (запис дозволено)
$I/O_0 \dots I/O_7$	*Data Inputs/Outputs* (інформаційні входи/виходи)

Порівнюючи рис. 8.32 і рис. 8.16 (блок-схема паралельної EEPROM), бачимо, що їх структури дуже близькі. Читання даних з мікросхеми подібне до читання зі статичної пам'яті. Репрограмування AT29C256 забезпечується на сторінковій основі, 64 байти даних завантажуються в мікросхему, а потім одночасно записуються. Вміст мікросхеми може бути стертим одночасно шестибайтним програмним кодом, хоча стирання перед програмуванням необов'язково. Протягом циклу репрограмування адреси розміщення 64 байтів записуваних даних завантажуються у внутрішні регістри і звільняють адресну шину і шину даних для інших операцій.

При наступній ініціалізації програмного циклу в мікросхемі флеш-пам'яті автоматично стирається адресована сторінка і завантажуються дані для наступного запису, при якому використовується внутрішній таймер. Закінчення програмного циклу може бути визначено по значенню I/O_7. Якщо кінець програмного циклу визначений, то може бути розпочатий новий доступ до мікросхеми на зчитування інформації чи її перезапис.

Розглянемо операції з флеш-пам'яттю.

Операція читання (READ). Для читання мікросхема доступна подібно попередньо розглянутим пристроям пам'яті. При $\overline{CE} = \overline{OE} = 0$ і $\overline{WE} = 1$ дані, які зберігаються в елементах пам'яті за адресним кодом, що задається на адресних входах $A_0 \dots A_{14}$ передаються на шину *I/O*. Перехід шини у *Z*-стан забезпечується умовою: $\overline{CE} \vee \overline{OE} = 1.$

Рекомендована послідовність подачі сигналів на входи мікросхеми: $A_0 \dots A_{14}$; $\overline{CE} = 0$; $\overline{OE} = 0$; читання даних. Після зчитування даних за вказаною адресою зміна адресного коду приведе до зчитування даних за новою адресою.

Операція програмування (PROGRAMMING). Мікросхема флеш-пам'яті програмується сторінками, які визначаються старшими розрядами $A_6 \dots A_{14}$ адресного коду. Молодші розряди $A_0 \dots A_5$ визначають розміщення байтів усередині сторінки. При необхідності зміни одного байту у межах сторінки необхідно виконувати перезапис цілої сторінки. Для програмування цілої сторінки спочатку інформація побайтово завантажується при високому рівні сигналу на вході $\overline{OE}$ і низьких рівнях сигналів на входах $\overline{CE}$ і $\overline{WE}$. Адресний код, який забезпечує запис байта, фіксується за зрізом останнього з сигналів $\overline{CE}$ або $\overline{WE}$ ($t_{\text{ф}A}$). Дані фіксуються за першим з фронтів $\overline{CE}$ або $\overline{WE}$ ($t_{\text{ф}D}$). Приклад часової діаграми завантаження байта приводиться на рис. 8.34.

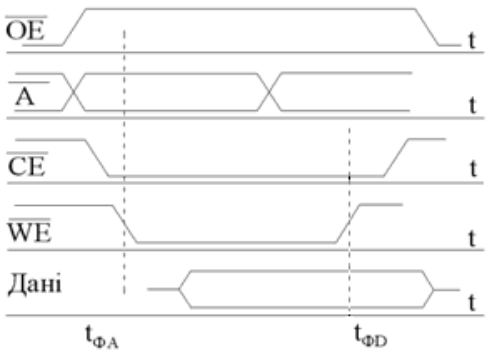


Рис. 8.34

Після завантаження одного байту, наступний повинен бути завантажений протягом інтервалу часу, що не перевищує 150 мкс. Якщо ж в інтервалі 150 мкс черговий байт не завантажиться, період завантаження припиниться і почнеться внутрішній період програмування сторінки. Протягом інтервалу запису сторінки має місце заборона операції читання, що контролюється, подібно до *EEPROM*, через I/O_7.

При такій організації запису перезапис окремого байту можливий лише на основі перевантаження сторінки з байтом, що замінюється в оперативну пам'ять, заміни байту в оперативній пам'яті з наступним перезаписом у флеш-пам'ять.

Дещо іншу, більш розвинену архітектуру мають мікросхеми ємністю понад 1 Мбіт. Особливість таких архітектур розглянемо на прикладі

мікросхеми флеш-пам’яті AT29C040A ємністю 4 Мбіт з організацією 512K × 8. Умовне позначення мікросхеми подібне до рис. 8.33 з розширеним адресним кодом в інтервалі $A_0 \ldots A_{18}$. Блок-схема мікросхеми приведена на рис. 8.35.

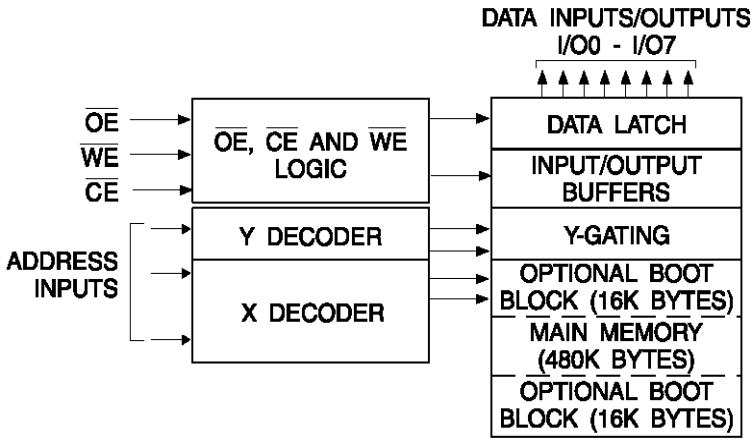


Рис. 8.35

Окрім базової системи команд, аналогічної раніше приведеним мікросхемам, мікросхема, що розглядається, має програмний механізм використання *BOOT*-блоків для блокування доступу до окремих розділів пам’яті. Мікросхема має два апаратно організованих блоки пам’яті, що мають механізм блокування. При використанні такого механізму виключається можливість програмування даних у визначені блоки. Кожен з таких блоків (рис. 8.35) має ємність 16 Кбайт. Режим роботи механізму блокування може бути встановленим незалежно для кожного з блоків або для двох одночасно. Такі блоки називають *BOOT*-блоками, оскільки в них здебільшого знаходяться коди, що завантажують всю систему пам’яті і які повинні бути надійно захищені. В даній мікросхемі *BOOT*-блоки розміщені у перших і в останніх шістнадцяти кілобайтах пам’яті. Завдяки цьому механізм блокування може підтримувати систему, що завантажує пам’ять з нижніх або верхніх адресних кодів.

При включенні механізму блокування дані в цих блоках ніколи не можуть бути стерті або перепрограмовані, у той час як дані основної пам’яті можуть бути змінені шляхом типового перепрограмування. Активізація механізму

блокування досягається серією з 7 команд для визначення адресних кодів специфічних даних, які повинні бути використані.

Визначити, чи включений механізм блокування відносно *BOOT*-блоків, можна програмними засобами. Для цього в довідковій літературі на мікросхему приводиться відповідний алгоритм зчитування. Суть його полягає в тому, що у режимі програмної ідентифікації йде звернення за адресою 00002H для визначення стану нижнього *BOOT*-блоку, а за адресою FFFF2H – відповідно, верхнього. Якщо зчитані дані FE, то відповідний блок можна програмувати; якщо ж за вказаними адресами зчитується FF, то механізм блокування включений і відповідний блок не може бути запрограмованим. Механізм ідентифікації стану *BOOT*-блоків можливо використати для повернення їх у звичайний режим роботи програмування.

Мікросхеми флеш-пам’яті з більшою інформаційною ємністю зведені в серію AT49F (AT49F008A, AT49F008AT, AT49F8192A, AT49F8192AT). Усі вони мають ємність 8 Мбайт з організацією 1M × 8 (512K × 16).

Для використання у персональних комп’ютерах випускаються флеш-карти ємністю 64, 128, 256 і 512 МБайт. Детальну інформацію про них можна знайти у довідковій літературі.

Серед мікросхем флеш-пам’яті з послідовним доступом розглянемо МС AT45D011 серії AT45D інформаційною ємністю 1 МБайт і організацією 512 сторінок по 264 байти кожна (до серії входять також МС AT45DB011, AT45DB081A, AT45DB321). У доповнення до основної пам’яті, мікросхема AT45D011 містить один буфер даних послідовної оперативної пам’яті ємністю 264 байти. На відміну від звичайної *flash*-пам’яті, що забезпечує доступ до даних через паралельну адресну шину і паралельний інтерфейс, мікросхема використовує послідовний доступ до даних і послідовний інтерфейс. Умовне позначення мікросхеми приведено на рис. 8.36, а призначення виводів – у табл. 8.13.

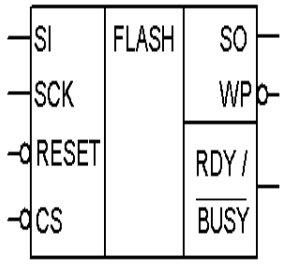


Рис. 8.36

Таблиця 8.13

Назва виводу	Призначення
$\overline{CS}$	*Chip Select* (вхід вибору мікросхеми)
SCK	*Serial Clock* (вхід тактового генератора)
SI	*Serial Input* (послідовний вхід)
SO	*Serial Output* (послідовний вихід)
$\overline{WP}$	*Hardware Page Write Protect Pin* (вхід апаратного захисту запису сторінки)
$\overline{RESET}$	*Chip Reset* (перевантаження пристрою)
$RDY/\overline{BUSY}$	*Ready/Busy* (готовність/зайнятість пристрою)

Мікросхема оптимізована для використання у різноманітних комерційних і промислових задачах, де суттєвими є малі розміри, невелика кількість виводів, низька напруга живлення і потужність споживання. Типове використання мікросхеми – зберігання звукових, голосових сигналів і різноманітних даних. Мікросхема має частоту синхронізації до 15 МГц. Блок-схема мікросхеми пам’яті зображена на рис. 8.37.

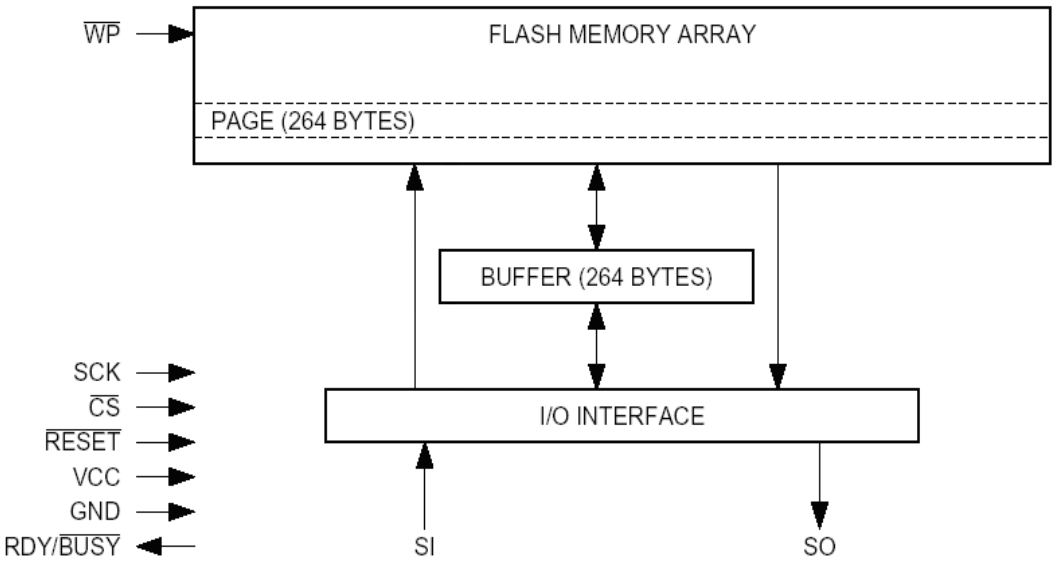


Рис. 8.37

З рис. 8.37 витікає загальний алгоритм функціонування МС AT45D011. Він полягає у тому, що весь обмін між матрицею флеш-пам’яті і зовнішніми пристроями забезпечується за допомогою керуючих сигналів інтерфейсу вводу-виводу через буфер оперативної пам’яті. Це забезпечує можливість розв’язання задач читання і особливо запису у послідовному форматі з високою швидкістю. Сторінка даних ємністю 256 байт записується за 7 мс за допомогою внутрішньої системи організації запису без використання окремої операції стирання через 167рьох провідну інтерфейсну шину, що складається з провідників *CS*, *SO*, *SI*.

Матриця пам’яті для забезпечення високої гнучкості розділена на три рівні: *сектори*, *блоки* та *сторінки*. Архітектура її з урахуванням взаємозв’язків між секторами, блоками і сторінками приводиться на рис. 8.38.

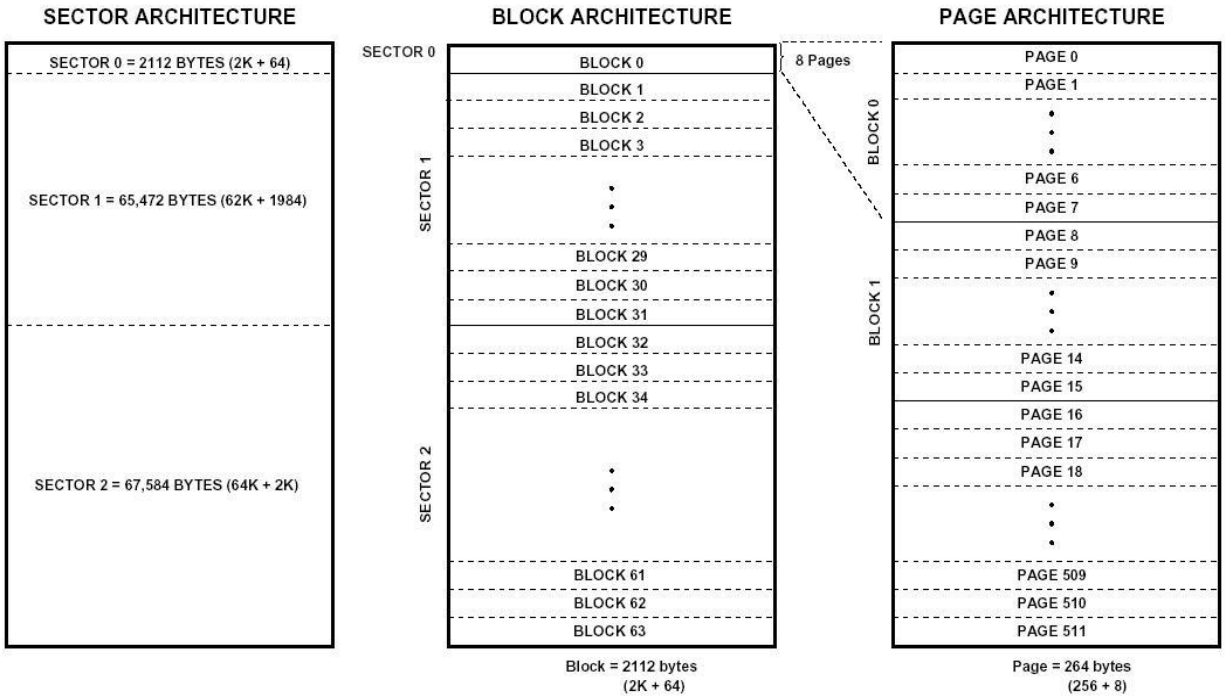


Рис. 8.38

Усі сторінки мають однаковий розмір – 256 байт, але кількість сторінок в блоках різна. Усі операції обміну з матрицею пам’яті виконуються на основі сторінок, але операції вибіркового стирання можуть виконуватись як сторінками, так і поблоково.

Усі операції обміну інформації забезпечуються через процесор за допомогою невеликої кількості спеціальних команд, які будуть висвітлені по ходу пояснень. Команди, адреси і дані передаються, починаючи зі старшого розряду.

Існують декілька операцій, які забезпечують зчитування інформації.

Зчитування інформації посторінково дозволяє користувачу читати дані з будь-якої з 512 сторінок основної пам'яті, минаючи буфер даних і залишаючи вміст буфера незмінним. Щоб почати посторінкове зчитування, формується така послідовність сигналів. Спочатку подається 8-бітний машинний код 52h. За ним подається адресний код довжиною 24 біти, а потім 32 біти байдужого стану (рис. 8.39).

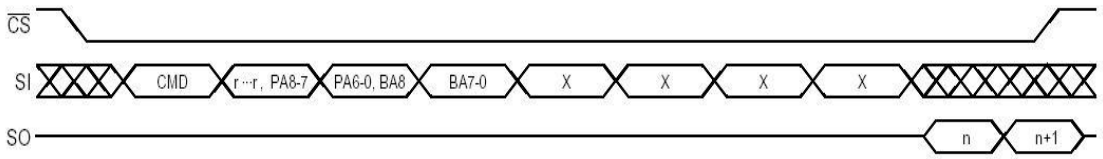
63 32	31 23	22 14	13 8	7 0
x … x	BA0 … BA8	PA0 … PA8	r … r	52h

Рис. 8.39

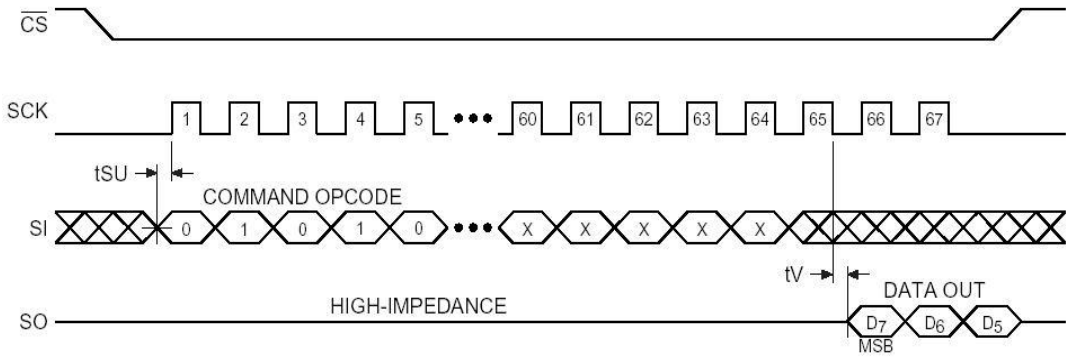
У приведеній послідовності 6 адресних бітів (8…13) зарезервовані (r…r) для можливості організації роботи з мікросхемами пам'яті, що містять більший обсяг інформації. Наступні 9 адресних бітів (14…22) вказують на адресу сторінки (PA0…PA8), а адресні біти BA0…BA8 (23…31) вказують на початкову адресу байта на сторінці. Решта 32 біти потрібні для ініціалізації операції зчитування. Уся ця послідовність сигналів подається на вхід *SI* з одночасною подачею синхроімпульсів на *SCK*. Після завершення описаної послідовності через 1 такт на вихід *SO* будуть видаватись дані синхронно з роботою тактового генератора. Описана операція зчитування ілюструється часовими діаграмами, що приведені на рис. 8.40, *а*, а більш детально деякі особливості пояснює рис. 8.40, б.

Вихід $\overline{CS}$ повинен мати низький рівень сигналу протягом всього часу звернення до мікросхеми і зчитування даних. Якщо при $\overline{CS} = 0$ досягається

кінець зчитування сторінки, то керуючий інтерфейс знову сформує початкову адресу цієї ж сторінки і продовжить зчитування. Перехід сигналу $\overline{CS}$ до високого рівня приведе до закінчення операції зчитування, і вихід *SO* перейде у *Z*-стан.



а)



б)

Рис. 8.40

Дані можуть бути зчитані і з буфера даних. Для цього формується послідовність сигналів, що зображена на рис. 8.41.

39 32	31 23	22 8	7 0
x … x	BFA0 … BFA8	x … x	54h

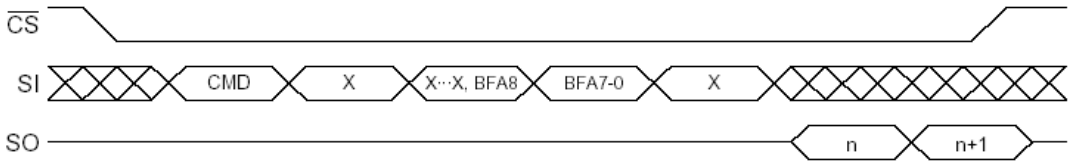
Рис. 8.41

Після машинного коду 54h подається 15 бітів байдужого стану, потім 9 бітів адресних (BFA8…BFA0) і знову 8 бітів байдужого стану.

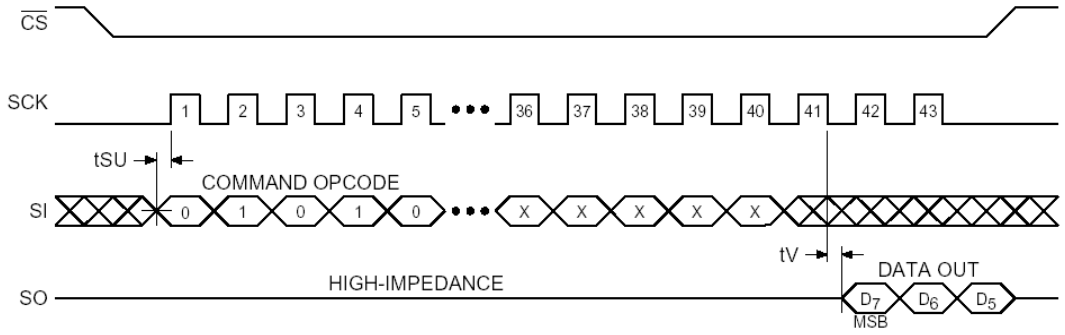
Дія сигналу $\overline{CS}$ у розгляданому випадку аналогічна попередній.

Деталізовані часові діаграми процесу зчитування даних з буфера приводяться на рис. 8.42.

Однією з операцій читання є *операція переміщення сторінки з основної пам'яті до буфера*. Така процедура починається після встановлення $\overline{CS} = 0$ завантаженням машинного коду 53h, за яким слідують 6 резервних бітів (r), 9 адресних (PA8…PA0), що вказують на сторінку пам'яті, котру необхідно перемістити, і 9 бітів байдужого стану (рис. 8.43).



а)



б)

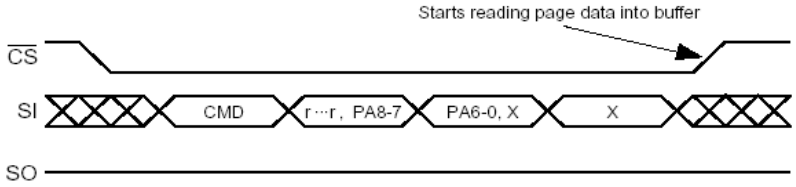
Рис. 8.42

31 23	22 14	13 8	7 0
x … x	PA8 … PA0	r … r	53h

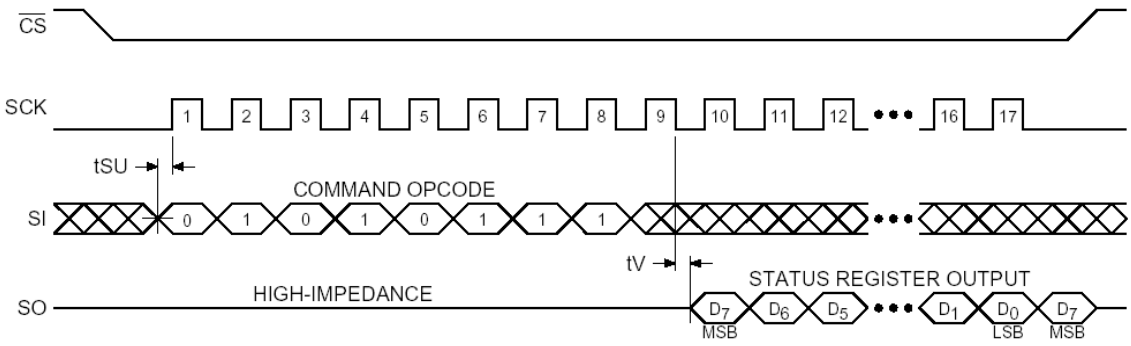
Рис. 8.43

В інтервалі дії вказаних 32-х тактів $\overline{CS} = 0$. Переміщення сторінки даних розпочинається після того, як по закінченню 32-го такту рівень сигналу на вході $\overline{CS}$ зміниться на високий. Під час переміщення даних за допомогою режимного регістра (*Status Register*) можна визначити, чи закінчилась передача даних. Часові діаграми завантаження коду для переміщення даних зображені на рис. 8.44.

До операції читання відноситься також *операція порівняння сторінки основної пам'яті з буфером*. Після машинного коду 60h йдуть 24 адресні біти, що містять 6 резервних (r…r), 9 адресних (PA8…PA0), які вказують на сторінку в основній пам'яті, порівнювану з даними буфера, і 9 бітів байдужого стану. Завантаження цієї послідовності відбувається аналогічно попередньому. При переході сигналу $\overline{CS}$ з низького рівня на високий, 264 байти обраної сторінки порівнюватимуться з відповідними байтами буфера.



а)



б)

Рис. 8.44

Результат порівняння виводиться у 6-й розряд режимного регістра ("**0**" у цьому розряді позначає відповідність даних).

Режимний регістр пристрою пам'яті використовується для визначення режиму мікросхеми (вільна або зайнята), коду обсягу пам'яті мікросхеми, результату порівняння сторінки пам'яті з буфером та деякої іншої інформації. Для визначення коду режимного регістра завантажується машинний код 57h. Після введення останнього з 8 бітів режимного регістра, починаючи зі старшого, можуть бути зчитані послідовно з *SO* за 8 тактів *SCK*. П'ять старших бітів регістра несуть інформацію про пристрій пам'яті, а решта 3

використовуються за індивідуальними призначеннями. Після зчитування нульового біта режим зчитування регістра повторюватиметься до того часу, поки $\overline{CS} = 0$.

Інформація про зайнятість мікросхеми (вільна або зайнята) міститься у 7-му біті. Якщо значення цього біта дорівнює **“1”**, то пристрій готовий отримати нову команду, тобто він вільний. Користувач може вибирати 7-й біт з режимного регістра, зупиняючи *SCK*, як тільки 7-й біт поданий на вихід.

Особливості операцій програмування. Запис даних у буфер через вхід *SI* забезпечується після завантаження через цей же вхід кодової послідовності, що складається з машинного коду 84h, 15 бітів байдужого стану та 9-ти адресних (BFA8…BFA0), що визначають перший байт буфера. Дані вводяться слідом за адресним кодом. Якщо при записі даних буфер буде заповнений, то при $\overline{CS} = 0$ почнеться повторний запис у буфер з початкової адреси. Запис припиняється при переході $\overline{CS}$ з **“0”** в **“1”**. Часові діаграми запису приведені на рис. 8.45, *а*, а структура кодової послідовності для виконання операцій запису в буфер – відповідно, на рис. 8.45, *б*.

а)

31 23	22 8	7 0
BFA8 … BFA0	x … x	84h

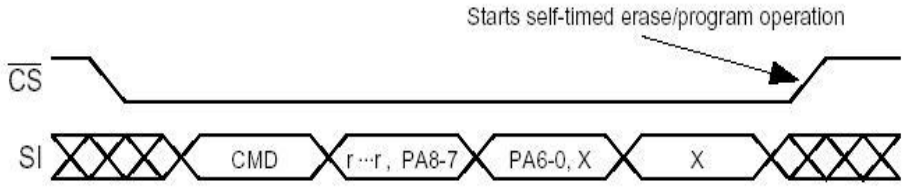
б)

Рис. 8.45

Перезапис в основну пам’ять з буфера забезпечується кодовою послідовністю, що починається з коду 83h. Структура кодової послідовності для перезапису приведена на рис. 8.46, *а*, а на рис. 8.46, *б* – відповідні часові діаграми.

31 23	22 14	13 8	7 0
x … x	PA8 … PA0	r … r	83h

а)



б)

Рис. 8.46

При переході рівня сигналу на $\overline{CS}$ з “**0**” на “**1**” пристрій спочатку знищить попередній запис на вибраній сторінці (запишуться всі одиниці), а потім завантажить дані з буфера. Режими стирання і запису в основну пам’ять супроводжуються внутрішнім режимом синхронізації. При цьому відповідний біт режимного регістра показуватиме зайнятість мікросхеми.

Якщо ж мікросхема програмується з попередньо очищеними сторінками пам’яті, то використовується попередня кодова послідовність (див. рис. 8.46, *а*), в якій машинний код 83h заміняється кодом 84h. Час запису у такому випадку буде зменшений.

Для очищення сторінки виконується команда *PAGE ERASE*, кодова послідовність якої повторює попередню з тією лише різницею, що вона має машинний код 81h. При переході рівня сигналу на $\overline{CS}$ з **“0”** на **“1”** пристрій очистить вказану сторінку і встановить у ній всі **“1”**. Операція стирання забезпечується внутрішньою синхронізацією.

Подібно до очищення сторінки основної пам’яті, може бути очищений цілий блок. Така операція виконується для зменшення часу перезапису великих обсягів інформації. Для виконання команди *BLOCK ERASE* використовується машинний код 50h. Кодова послідовність очищення блоку приведена на рис. 8.47.

31 20	19 14	13 8	7 0
x … x	PA8 … PA0	r … r	50h

Рис. 8.47

Шість адресних регістрів PA8…PA3 вказують на блок з 8 сторінок, який необхідно очистити. При переході рівня сигналу на $\overline{CS}$ з “**0**” на “**1**” за внутрішньою самосинхронізацією очиститься весь блок.

Програмування безпосередньо основної пам’яті виконується кодовою послідовністю, що приведена на рис. 8.48.

Після того, як всі адресні біти будуть завантажені, пристрій прийме дані з *SI* і запише їх в буфер даних. При заповнені буфера і зміні рівня сигналу на $\overline{CS}$ з “**0**” на “**1**” спочатку буде очищена адресована сторінка, а потім перепишуться дані з буфера. Очищення сторінки та її перезапис забезпечуються внутрішньою синхронізацією.

31 23	22 14	13 8	7 0
PA8 … PA0	BFA8 … BFA0	r … r	82h

Рис. 8.48

На завершення, розглянемо призначення тих виводів мікросхеми, які раніше не розглядалися. Вхід *WP* – апаратний захист запису. При нульовому значенні сигналу на ньому перші 256 сторінок основної пам’яті не можуть бути репрограмовані. Для репрограмування цих сторінок необхідно встановити сигнал «1» на *WP* і використати вище описані команди.

Вхід *RESET* при низькому рівні сигналу на ньому перериватиме виконання будь-якої операції і зупинятиме роботу мікросхеми. Нормальна робота забезпечуватиметься високим рівнем сигналу, який автоматично встановлюється на вході *RESET* після подачі напруги живлення. В подальшому рекомендується підтримувати високий рівень сигналу на ньому за допомогою зовнішніх кіл.

Вихід $RDY/\overline{BUSY}$ низьким рівнем сигналу сигналізує про зайнятість мікросхеми внутрішніми режимами запису. Стан зайнятості показує, що масив флеш-пам’яті не може використовуватись.

8.3. Використання ПЗП

Широка номенклатура постійних і репрограмованих запам’ятовуючих пристроїв дає можливість широкого їх використання за різними призначеннями.

Вже з першого параграфу поточного розділу зрозуміло, що мікросхеми пам’яті можуть безпосередньо використовуватись для реалізації систем логічних функції, для виконання лінійних і нелінійних перетворень вхідної цифрової послідовності, для виконання апаратного перемноження двох сигналів, заданих у цифровій формі і, зрозуміло, для зберігання послідовностей команд та даних.

8.3.1. Використання ПЗП як універсальних комбінаційних схем

Використання ПЗП як універсальних комбінаційних схем є одним з найпоширеніших напрямків, оскільки дає можливість замінити велику кількість корпусів простих логічних елементів з неповним (здебільшого) їх використанням, одним ПЗП, ємність якого може бути підібрана під задану систему комбінаційних функцій. Такий підхід водночас дає можливість суттєво підвищити швидкодію розробленого комбінаційного пристрою та зменшити величину струму споживання.

Виходячи з § 8.1, принцип використання ПЗП для реалізації системи комбінаційних схем зводиться до того, що адресні входи мікросхем ПЗП розглядаються як входи комбінаційної схеми, тобто входи логічних змінних, а виходи – відповідно, як функції.

Приклад 8.3. Пояснити, скільки логічних функцій можуть бути реалізовані на мікросхемі ПЗП К155РЕ3 (82S23); функції скількох змінних можуть бути на ній реалізовані.

Пояснення. Мікросхема К155РЕ3 – це електрично програмований шляхом перепалення плавких перемичок ПЗП ємністю 256 біт з організацією 32×8. У початковому стані за всіма адресами в усі розряди записані нулі. Умовне позначення мікросхеми приведено на рис. 8.49.

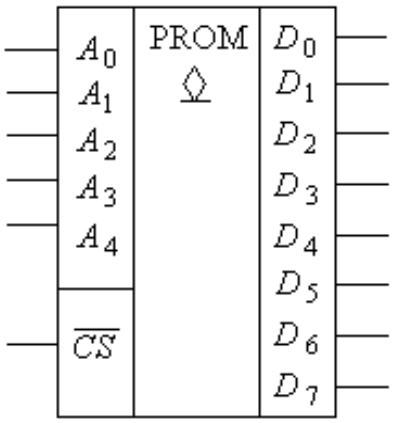


Рис. 8.49

Виходячи з електричних даних мікросхеми, її можна запрограмувати для реалізації системи восьми логічних функцій, кожна з яких має п’ять змінних.МП має виходи $D_0 \dots D_7$ з відкритим колектором, тому вони повинні бути приєднані до джерела живлення через резистори.

Приклад 8.4. Використовуючи мікросхему ПЗП КР556РТ4 (82S126), реалізувати систему логічних функцій:

$$\begin{cases} y_0 = \bigvee\limits_{0}^{31} 0, 2, 5, 11, 13, 17, 20, 23, 25, 31; \\ y_1 = \bigvee\limits_{0}^{31} 1, 2, 5, 8, 11, 14, 18, 21, 22, 28, 30; \\ y_2 = \bigvee\limits_{0}^{31} 0, 3, 8, 9, 10, 12, 19, 24, 25, 27, 31; \\ y_3 = \bigvee\limits_{0}^{31} 4, 5, 8, 11, 13, 15, 17, 21, 24, 26, 29. \end{cases}$$

Розв’язання. Оскільки система логічних функцій має п’ять логічних змінних, а мікросхема має 8 адресних входів, тому невикористовувані входи A_5, A_6, A_7, $\overline{CS_1}$ та $\overline{CS_2}$ необхідно заземлити.

Складаємо карту прошивки ПЗП. У початковому стані, відповідно до довідкової літератури, у ній записані всі нулі. Оскільки старші розряди заземлені, то це означає, що максимальна адреса, яка може бути використана в ПЗП – це 00011111_2, тобто $1F_{16}$. Як наслідок, таблиця прошивок матиме лише два рядки з початковими адресами 00 та 10, по 16 клітин кожний, з адресами від 00 до 0F і від 10 до 1F. В кожну клітину повинен бути записаний чотирьохрозрядний код, який потім може бути зчитаний з виходів мікросхеми. Присвоюємо виходам мікросхеми $D_0 \dots D_3$ відповідні значення функцій $y_0 \dots y_3$. Тоді значення функцій, які повинні бути зчитаними, запишемо у вигляді таблиці (табл. 8.14).

Таблиця 8.14

N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y_0	1	0	1	0	0	1	0	0	0	0	0	1	0	1	0	0
y_1	0	1	1	0	0	1	0	0	1	0	0	1	0	0	1	0
y_2	1	0	0	1	0	0	0	0	1	1	1	0	1	0	0	0
y_3	0	0	0	0	1	1	0	0	1	0	0	1	0	1	0	1
N	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
y_0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	1
y_1	0	0	1	0	0	1	1	0	0	0	0	0	1	0	1	0
y_2	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	1
y_3	0	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0

Тепер табл. 8.14 можемо скористатись для заповнення таблиці прошивок (див. табл. 8.15).

Таблиця 8.15

Адреса	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	5	2	3	4	8	B	0	0	E	4	4	B	4	9	2	8
01	0	9	2	4	1	A	2	1	C	5	8	4	0	8	2	5

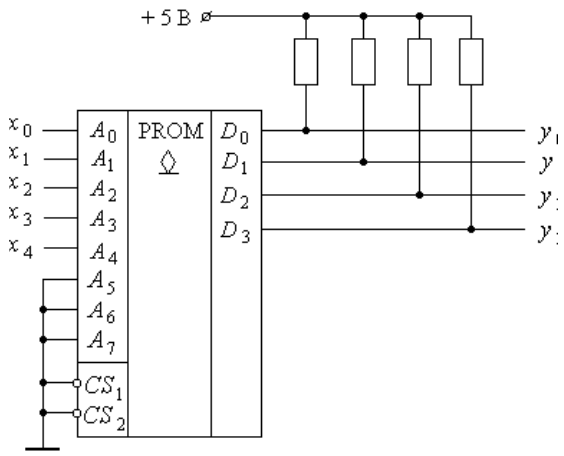


Рис. 8.50

Значення верхніх 16 стовпців табл. 8.14 (*N=0…15*), представлені у шістнадцятковому коді, переписуються у рядок табл. 8.15 з адресою 00. Наприклад, значення нульового стовпця табл. 8.12 $y_3\,y_2\,y_1\,y_0 = 0101_2 = 5_{16}$ записується у нульовий стовпець рядка 00 табл. 8.15 у вигляді 5_{16} і т.д. Значення нижніх 16 стовпців табл. 8.16 (*N=16…31*) записуються у рядок табл. 8.15 за адресою 01 (рис. 8.50).

Конфігурація схеми показує, наскільки спрощується схемотехніка, зменшується складність друкованих плат, потужність споживання. У той же час, використання ПЗП для заміни комбінаційних схем має свої недоліки. Вони полягають у тому, що мікросхеми ПЗП дуже чутливі до моментів зміни вхідних сигналів, тобто адресних розрядів, в які можлива поява коротких важко прогнозованих імпульсів. Тому при таких замінах слід врахувати подібні особливості. Вони обумовлені тим, що при проектуванні пристроїв комбінаційної схемотехніки на дискретних компонентах майже завжди враховується явище гонок (див. **Розділ 3**), а при

програмуванні на ПЗП подібний аналіз, як правило, відсутній. Тому при замінах складної комбінаційної схемотехніки програмованими ПЗП слід врахувати зазначені особливості. Це робиться, наприклад, використанням синхронізації через входи $\overline{CS}$. У даному випадку низький рівень синхросигналу на вході $\overline{CS}$ слід подавати лише після того, як всі перехідні процеси в мікросхемі, обумовлені зміною логічних сигналів на адресних входах, будуть закінчені.

Приведений приклад показує, що за допомогою ПЗП можливо створити будь-який комбінаційний пристрій, але в ряді випадків краще не використовувати таку можливість, оскільки подібний шлях не завжди корисний. Реально є сенс використовувати ПЗП у тих випадках, коли необхідні комбінаційні пристрої відсутні в бібліотеках компонентів.

8.3.2. ПЗП як нелінійні функціональні перетворювачі

Оскільки в загальному плані ПЗП можуть розглядатись як перетворювачі вхідного коду у вихідний, що задається розробником, то їх можна використовувати як нелінійні функціональні перетворювачі, швидкодіючі обчислювачі, тощо. Такі табличні перетворювачі мають високу швидкодію, порівняно з іншими, дають можливість забезпечувати високу точність, порівняно легко створюються. Прикладом є перемножувач, приведений на початку розділу.

Недоліком обчислювачів табличного типу, реалізованих на ПЗП, є значне збільшення обсягу пам’яті при збільшенні розрядності вхідного коду. Для зменшення необхідної інформаційної ємності ПЗП часто їх використання поєднують зі спеціальними математичними перетвореннями, які надають можливість зменшити апаратні затрати. Наприклад, для множення двох восьмирозрядних чисел A_8 та B_8 їх зображають у вигляді суми двох чотирьохрозрядних:

$$A_8 = A_4 + \Delta A_4 ; \qquad B_8 = B_4 + \Delta B_4 .$$

Внаслідок виконання операції множення отримуємо:

$$A_8 B_8 = A_4 B_4 + A_4 \Delta B_4 + \Delta A_4 B_4 + \Delta A_4 \Delta B_4 .$$

Це дає можливість використати чотирьохрозрядні перемножувачі та невелику кількість суматорів з використанням відповідних схем з'єднання.

Аналогічно операції множення можна зобразити і операцію ділення. Наприклад, ділення восьмирозрядного числа A_8 на чотирьохрозрядне число B_4 можна зобразити у вигляді алгоритму:

$$\frac{A_8}{B_4} = \frac{A_4 + \Delta A_4}{B_4} = \frac{A_4}{B_4} + \frac{\Delta A_4}{B_4},$$

що дає можливість використовувати ПЗП невеликої інформаційної ємності та чотирьохрозрядні суматори.

Подібні алгоритми дозволяють за допомогою ПЗП будувати обчислювачі і більш складних функцій.

8.3.3. Перетворювачі кодів для матричних індикаторів

Одним з найпоширеніших напрямів використання ПЗП є побудова перетворювачів кодів для матричних індикаторів відображення різноманітних знаків, цифр, букв (знакогенераторів), а також перетворювачів кодів для семисегментних індикаторів. Формування матриць знакомісць використовується у табло "біжучий рядок", на екранах моніторів, великих рекламних табло та ін. Кожен знак розміщується на прямокутній матриці – знакомісці, що містить декілька рядків та стовпців крапкових елементів зображення, які можуть засвічуватись незалежно один від іншого. Якість зображення тим вища, чим більше елементів використовується у знакомісці. Мінімальний розмір знакомісця – 5 стовпців і 7 рядків, тобто 35 елементів, що створюють зображення.

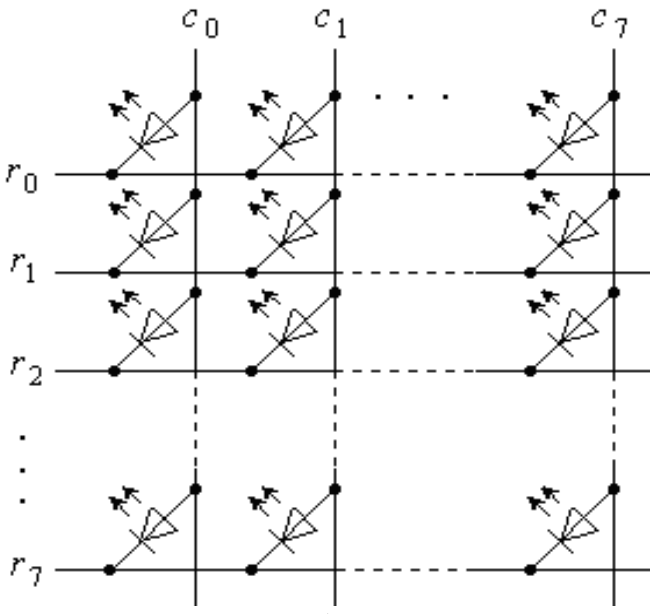
ПЗП для таких задач містить у собі інформацію про зображення всіх використовуваних знаків. Оскільки вихідний код ПЗП має невелику кількість

розрядів, тому кожен розряд повинен містити інформацію про зображення не всього символу, а лише одного стовпця або рядка. Приклад схеми синтезу знаків у знакомісці приводиться на рис. 8.51, *а – б*.

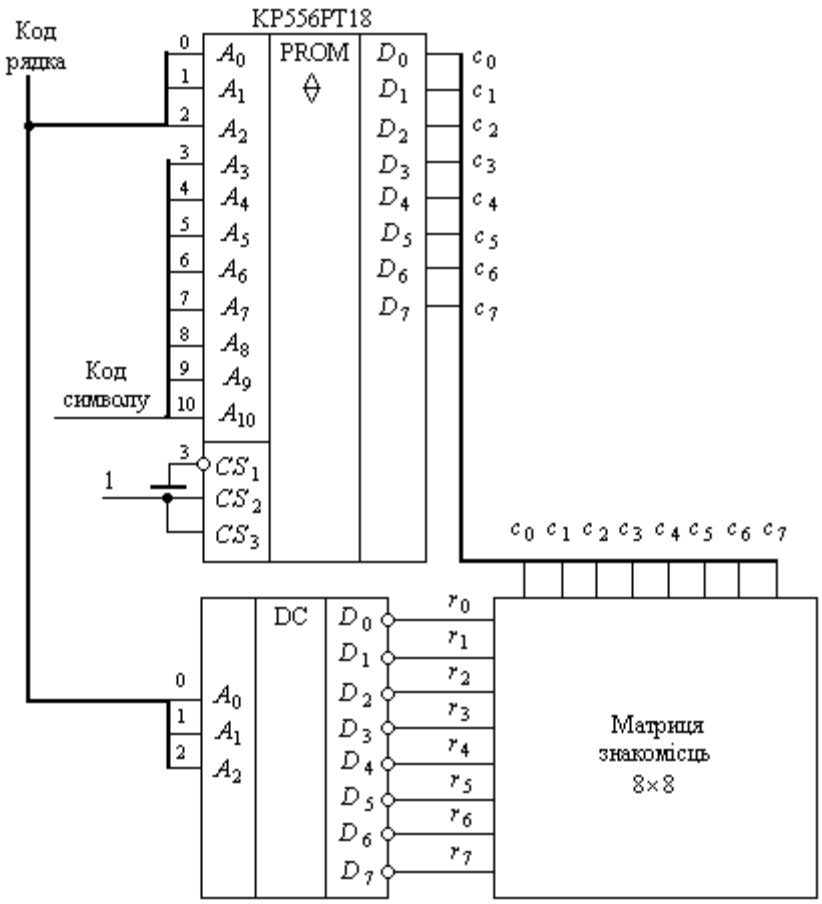
На рис. 8.51, *а* приведена схема з'єднання світлодіодів у матриці 8 × 8, а на рис. 8.51, *б* – схема керування цією матрицею.

У даному випадку матриця ПЗП КР556РТ18 (НМ76161) дає можливість за допомогою адресних розрядів $A_3 \ldots A_{10}$ вибирати один з 256 символів. Три молодші розряди призначені для сканування рядків. Вибір рядка забезпечується за допомогою дешифратора DC.

Побудова карти прошивки такого ПЗП є досить складною задачею і на практиці реалізується за допомогою спеціальних програм. Але методологія її побудови подібна до попередніх прикладів. Якщо, наприклад, для деякого знаку необхідно, щоб в першому рядку засвітились перший і останній світлодіоди, то за відповідною адресою цього рядку повинен бути записаний код 81_{16}(10000001). У такий же спосіб записуються коди відображаючі стани світлодіодів інших семи рядків.



а)



б)

Рис. 8.51

8.3.4. Використання ПЗП для реалізації складних способів модуляції

Все більш широке використання ПЗП знаходять у задачах забезпечення складних способів модуляції (модеми, CD-ROM), у компресорах та експандерах кодових послідовностей.

У попередньому розділі наводився приклад типової цифрової телефонної системи, в яких використовується восьмибітне шифрування аналогового сигналу частотою 8 кГц з наступним перетворенням у послідовний формат і компресією при передачі. Спосіб кодування, в якому фіксується амплітуда і знак сигналу, називається *лінійним кодуванням*. Такий спосіб кодування має кількість рівнів $2^8 = 256$, що відповідає динамічному діапазону $20\lg 256 \approx 48$ дБ (для порівняння, в аудіо-компакт-дисках використовується 16-бітне лінійне

кодування з динамічним діапазоном $20 \lg 2^{16} \approx 96$ дБ; зрозуміло, наскільки якість звуку з телефонних ліній буде гіршою, ніж якість звуку з компакт-дисків).

Щоб забезпечити більш широкий динамічний діапазон без збільшення розрядності і, відповідно, значних матеріальних витрат, використовується 8-бітні розширюючі кодери, які називаються μ-ІКМ (μ-імпульсно-кодова модуляція), а в англомовних джерелах – *PCM* (*Pulse-Code Modulation*).

На рис. 8.52 зображено формат 8-бітного кодового слова по типу зображення чисел з "плаваючою комою", де S – знак числа, E – область порядку, M – область мантиси.

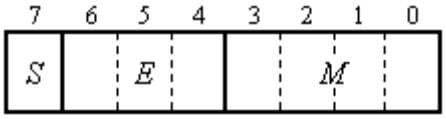


Рис. 8.52

Число U, представлене байтом такого формату, обчислюється за формулою:

$$U = (1 - 2 S) [2^E (2 M - 33) - 33].$$

Аналоговий сигнал, зображений у такому форматі, приймає значення від - 8159 k до + 8159 k, де k – довільний масштабний коефіцієнт. Кількістно дискретний ряд сигналів дорівнює $2 \cdot 8159$, а найменша різниця між членами ряду дорівнює 2 (коли $E = 0$), так що маємо динамічний діапазон $20 \lg 8159$, або близько 78 дБ, що значно більше, ніж при лінійному кодуванні.

Тепер подивимось на деякі особливості використання μ-ІКМ при кодуванні звукового сигналу.

У багатьох типах телефонного зв'язку голос навмисно послаблюють на декілька децибел з метою покращення роботи апаратури. В аналогових телефонних мережах послаблення забезпечується пасивними аналоговими колами, що неможливо зробити у світі цифр. Тому створений за допомогою μ-ІКМ байт необхідно перетворити в інший, послаблений цифровим

атенюатором. Така операція досягається шляхом перемноження μ-ІКМ-байту на визначений коефіцієнт послаблення. Приклад подібного множення приводиться на початку розділу.

Всі описані задачі повинні розв'язуватись у реальному масштабі часу, тому їх реалізація виконується з використанням ПЗП, оскільки така реалізація має практично найбільшу швидкодію.

8.3.5. Використання ПЗП у генераторах періодичних послідовностей

Використання ПЗП у генераторах періодичних послідовностей є одним з важливих напрямків їх використання. ПЗП дають можливість генерувати складні послідовності імпульсів, які широко використовуються у різноманітних вимірювальних системах, у пристроях автоматики, телевізійних системах тощо. Особливість таких генераторів полягає в тому, що за їх допомогою можна створювати взаємозв'язані послідовності, сигнали, які можуть мати постійну амплітуду в широкому частотному діапазоні, постійні кутові співвідношення і т. п.

Одна з найбільш розповсюджених структур генераторів послідовностей імпульсів приводиться на рис. 8.53.

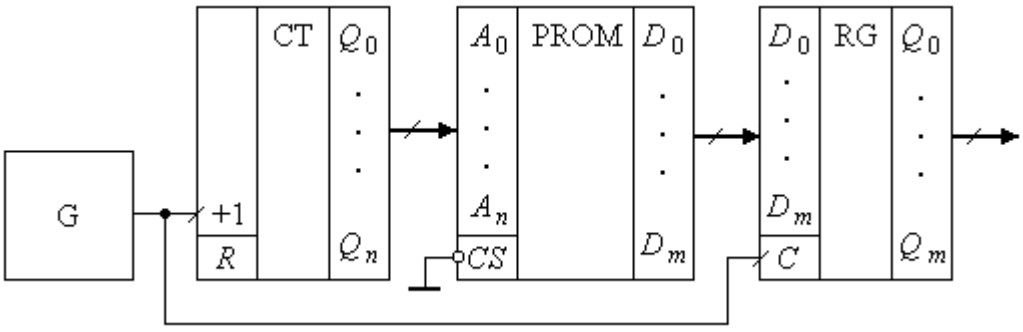


Рис. 8.53

Генератор і двійковий лічильник виконують функцію генератора двійкового коду і задають кількість тактів на періоді та тривалість такту. ПЗП забезпечує характер вихідної послідовності або декількох послідовностей

вихідних імпульсів. Вихідний регістр призначений для того, щоб відвернути можливість появи в вихідних сигналах паразитних імпульсів, обумовлених перемиканням виходів ПЗП, а також для забезпечення одночасного перемикання всіх вихідних сигналів.

Головна задача при проектуванні генераторів – це розробка карти прошивки ПЗП. У той же час, методологія її розробки багато в чому схожа із задачею реалізації системи логічних функцій, описаною вище.

Приклад 8.5. Розробити таблицю прошивок генератора шести імпульсних послідовностей, часові діаграми яких приведені на рис. 8.54.

Розв'язання. Оскільки маємо шість сигналів, то ПЗП повинен мати не менше шести виходів. Він може бути створений або шляхом нарощування по виходах, або використовується ПЗП на 8 виходів.

Періодичність повторення сигналів дорівнює 13 тактам. Це дає можливість використати ПЗП з кількістю адресних входів, що дорівнює 4. При більшій кількості входів старші розряди необхідно заземлити. Створимо таблицю для потактного кодування вихідних послідовностей (див. табл. 8.16).

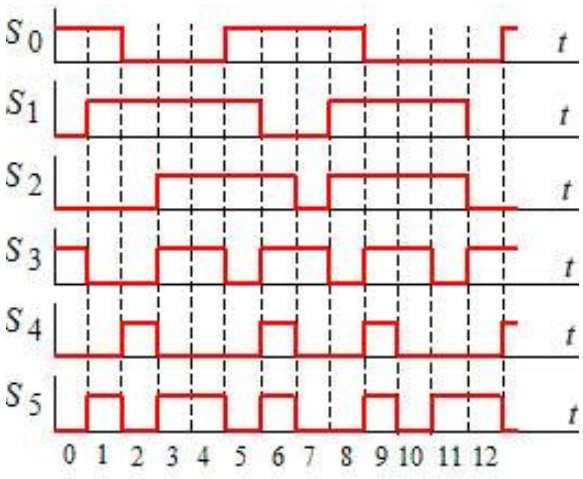


Рис. 8.54

Таблиця 8.16

N такту	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Шістнадцятковий код
0	0	0	0	0	1	0	0	1	09
1	0	0	1	0	0	0	1	1	23
2	0	0	0	1	0	0	1	0	12
3	0	0	1	0	1	1	1	0	2E
4	0	0	1	0	1	1	1	0	2E
5	0	0	0	0	0	1	1	1	03
6	0	0	1	1	1	1	0	1	3D
7	0	0	0	0	1	0	0	1	09
8	0	0	0	0	0	1	1	1	07
9	0	0	1	1	1	1	1	0	3E
10	0	0	0	0	1	1	1	0	0E
11	0	0	1	0	0	1	1	0	26
12	0	0	1	0	1	0	0	0	28

Приймаємо, що сигнали $S_0 \dots S_5$ знімаються відповідно з виходів ПЗП $D_0 \dots D_5$, а старші розряди D_6, D_7 залишаються незадіяними. Створена таблиця і відповідний їй шістнадцятковий код дають можливість створити таблицю прошивок вибраного ПЗП (див. табл. 8.17).

Таблиця 8.17

Адреса	0	1	2	3	4	5	6	7	8	9	A	B	C
00	09	23	12	2E	2E	03	3D	09	07	3E	0E	26	28

Вибір лічильника, його коефіцієнту перерахунку, частоти тактового генератора, а також тривалості тактового імпульсу – це задачі, які розв'язуються на основі попередніх прикладів.

ПЗП часто використовують для генерації сигналів аналогового типу – наприклад, періодичних тригонометричних функцій ($\sin x$, $\cos x$), імпульсів спеціальної форми, функціональних залежностей періодичного типу, які неможливо реалізувати на аналогових схемах. Для цього після регістра або ПЗП встановлюються цифро-аналогові перетворювачі, які забезпечують на виході сигнал, близький до аналогового. Подібні генератори мають свої переваги. Наприклад, синусоїда, створена подібним шляхом, матиме незмінну амплітуду в широкому діапазоні частот, який задаватиметься тактовим генератором.

Приклад 8.6. Розробити таблицю прошивок ПЗП для генератора синусоїдальної функції $y = 15 \sin \varphi$ на інтервалі $[\,0; \pi\,]$, базуючись на ПЗП, що має 4 адресні входи і 4 вихідні розряди.

Розв'язання. За умовою задачі, півперіод можемо розбити на 16 дискретних кутових інтервалів, а амплітуду – на 16 дискретних рівнів. Функцією зобразимо у вигляді:

$$N = 15 \sin\left(i \cdot \frac{\pi}{16} \right) \qquad (0 \le i \le 15 - \text{ціле число}).$$

Таблиця істинності генератора функції матиме вигляд табл. 8.18, де $A_3 \dots A_0$ – адресні входи ПЗП; $D_3 \dots D_0$ – виходи ПЗП.

Значення i та N таблиці істинності відповідатимуть таблиці прошивок ПЗП.

Форма сигналу, який буде отриманий на виході цифро-аналогово перетворювача, має вигляд, приведений на рис. 8.55.

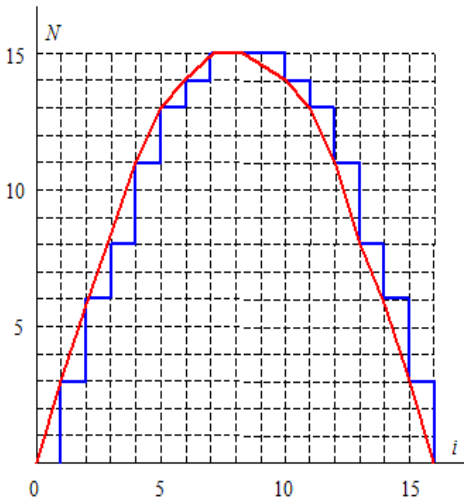


Рис. 8.55

Таблиця 8.18

i	A_3	A_2	A_1	A_0	y	N	D_3	D_2	D_1	D_0
0	0	0	0	0	0,000	0	0	0	0	0
1	0	0	0	1	2,926	3	0	0	1	1
2	0	0	1	0	5,740	6	0	1	1	0
3	0	0	1	1	8,334	8	1	0	0	0
4	0	1	0	0	10,610	11	1	0	1	1
5	0	1	0	1	12,472	13	1	1	0	1
6	0	1	1	0	13,860	14	1	1	1	0
7	0	1	1	1	14,712	15	1	1	1	1
8	1	0	0	0	15,000	15	1	1	1	1
9	1	0	0	1	14,712	15	1	1	1	1
10	1	0	1	0	13,860	14	1	1	1	0
11	1	0	1	1	12,472	13	1	1	0	1
12	1	1	0	0	10,610	11	1	0	1	1
13	1	1	0	1	8,334	8	1	0	0	0
14	1	1	1	0	5,740	6	0	1	1	0
15	1	1	1	1	2,926	3	0	0	1	1

8.3.6. Використання ПЗП у скінченних мікропрограмних автоматах

Оскільки мікропрограмні автомати працюють за програмою, яка зчитується і виконується керуючим автоматом, то використання ПЗП у них дає можливість суттєво розширити обсяги розв’язуваних задач.

Розглянемо одну з розповсюджених структур цифрового автомату, що складається з ПЗП (*ROM*), регістра (*RG*) та генератора тактових імпульсів *C* (рис. 8.56), який призначений для генерації імпульсних послідовностей керування системи контролю та діагностики.

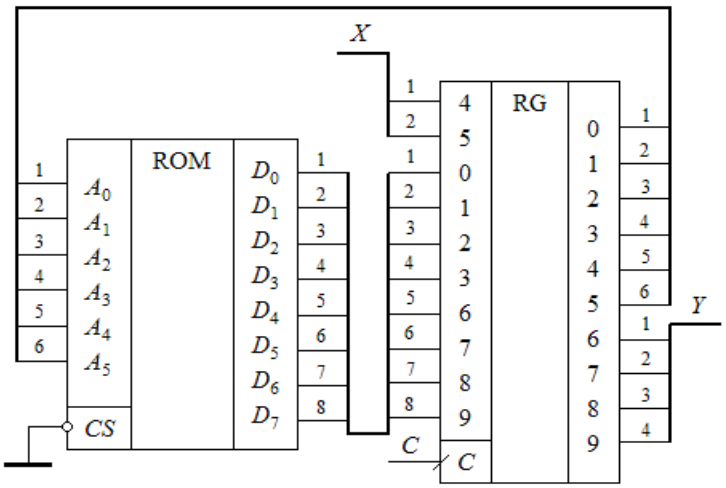


Рис. 8.56

У даній структурі ПЗП має 6 адресних входів і 8 виходів, тобто його організація 64 × 8. Усі адресні сигнали формуються безпосередньо з молодших розрядів регістра *RG*. Вихідні сигнали *Y* формуються безпосередньо на чотирьох старших розрядах регістра.

Особливість ЦА полягає у тому, що вхідні керуючі сигнали *X* ($x_1\, x_0$) подаються на 4-й та 5-й розряди регістра, які потім безпосередньо використовуються як адресні сигнали ПЗП.

Вихідні розряди ПЗП розділені на дві групи, по 4 розряди кожна: молодші використовуються для створення чергових адрес ПЗП, а старші використовуються як вихідні сигнали автомату. Тобто безпосередньо по коду даних з таблиці прошивок можна бачити, якою буде наступна адреса ПЗП і якими будуть вихідні сигнали автомату на черговому такті. Вхідні сигнали використовуються для генерації старших адресних сигналів ПЗП. Така особливість досить чітко видима при розгляді карти прошивки, адже 4 молодші розряди адресного коду задають клітини ПЗП, що розміщуються у першому рядку за адресою 00. Два старші розряди, тобто розряди, що задаються вхідними сигналами, дають можливість вибирати один з чотирьох рядків, що мають адреси 00, 01, 02, 03.

Приклад 8.7. Провести аналіз роботи автомату, карта прошивок ПЗП якого приведена у табл. 8.19.

Таблиця 8.19

Адреса	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	11	22	33	44	55	66	77	88	99	AA	BB	CC	DD	EE	FF	00
01	11	22	33	44	55	66	77	88	99	AA	BB	CC	DD	EE	FF	00
02	10	21	32	43	54	65	76	87	98	A9	BA	CB	DC	ED	FE	0F
03	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	00

Розв'язання. Рядок з адресою 00 демонструє послідовний перебір адрес пам'яті при нульових вхідних сигналах. Якщо у початковому стані автомат знаходиться за адресою 00, то в клітині за цією адресою стоїть шістнадцятковий код 11, молодший розряд якого показує, що на наступному такті він перейде за адресою 01. Розглядаючи клітину за адресою 01, бачимо, що чергова адреса переходу – 02, і так до клітини з адресою OE, в якій стоїть чергова адреса OF. У клітині OF вказана початкова адреса 00.

Чотири біти старших розрядів повторюють адресний код, який на кожному такті інкрементується на **1**, тобто схема працює, подібно двійковому накопичувальному лічильнику з коефіцієнтом перерахунку $M = 16$.

Другий рядок демонструє іншу особливість роботи автомату – циклічне повторення групи тактів. Якщо робота автомату почнеться з адреси 10, то автомат послідовно інкрементуватиметься на чергові адреси 11, 12, 13, 14, 15, 15, 16, 17, 18, 19. З адреси 19 автомат повернеться на адресу 15 і циклічно повторюватиме групу адрес 15, 16, 17, 19. Якщо ж автомат почне роботу з адреси 1A, то він послідовно перейде на адресу 19 і знову почне циклічно працювати в інтервалів адрес 15 – 19.

Розглянемо тепер ситуацію, коли вхідний сигнал $X = x_1\, x_0$ при інкрементуванні адресного коду змінює своє значення – наприклад, з $\overline{x_1}\ \overline{x_0}$ в $\overline{x_1}\ x_0$. Фактично це означає, що автомат переходить з однієї мікропрограми на іншу. Якщо перехід відбувається в інтервалі адрес 00…08 (10…18), то як по адресних, так і по вихідних сигналах це неможливо помітити, оскільки адреси дублюються, і вихідні сигнали теж. Фактично це означає, що такою прошивкою нейтралізується реакція автомату на зміну вхідного сигналу x_0.

Третій рядок з адресою $x_1\ \overline{x_0} = 10$ показує приклад зупинки автомату за кожною адресою і очікування приходу вхідного сигналу. Наприклад, автомат знаходиться за адресою 23 (клітина, в якій записано число 43). Наступною адресою в коді даної клітини вказано 3, тобто при незмінних вхідних сигналах автомат залишатиметься за цією адресою постійно. Але якщо вхідний сигнал переведе автомат на адресу 03, то він почне виконувати мікропрограму нульового рядку. Звідси витікає, що дублювання адреси на одному з рядків може не тільки зупиняти виконання програм, а й створювати затримку в виконанні програми на час, що задається вхідним кодом.

Останній рядок демонструє створення переходу з будь-якої адреси рядка на нульову адресу того ж рядка. Наприклад, виконується мікропрограма нульового рядка, і за адресою 06 обидва вхідні сигнали змінюють свої значення на 11 ($x_1\, x_0 = 11$). Автомат при цьому попадає у робочу точку клітки за адресою 37, з якої переходить за адресою 30. У цій точці автомат зупиниться і знаходитиметься в ній до зміни вхідних сигналів. Якщо значення вхідних сигналів зміняться на $\overline{x_1}\ \overline{x_0} = 00$, то він почне відлік з нульового рядка.

Іноді виникає необхідність переходу автомату на деяку визначену адресу. Наприклад, при вмиканні живлення в регістр повинен записатись будь-який довільний код. У таких випадках можна створити мікропрограму, за допомогою якої автомат за декілька тактів завжди перейде на задану адресу, незалежно від початкової адреси.

Приклад 8.8. Розробити мікропрограмний автомат, який генеруватиме послідовність з трьох вихідних сигналів за фронтом вхідного сигналу у відповідності до рис. 8.57. На інтервалі дії даної послідовності вхідний сигнал x_0 може довільно змінюватись без впливу на характер зміни вихідних сигналів. Після закінчення вказаної послідовності автомат знову переходить у режим очікування фронту вхідного сигналу. Тривалість одного такту 10 мкс.

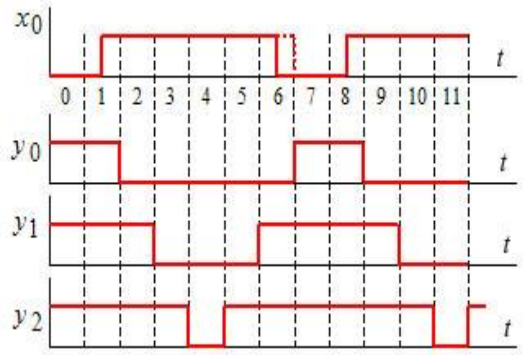


Рис. 8.57

Розв'язання. Оскільки тривалість одного такту дорівнює 10 мкс, то частота генератора тактових сигналів становитиме 100 кГц.

Мінімальна тривалість одного періоду генерованої послідовності з урахуванням того, що повинен бути як мінімум 1 такт, при якому на всіх виходах маємо високий рівень сигналу, дорівнює 6 тактам. Для формування 6 тактів необхідно не менш ніж 3 розряди ПЗП ($2^3 = 8 > 6$). Один допоміжний розряд необхідно мати для фіксації вхідного сигналу, тому ПЗП повинен мати 4 розряди.

Розмір вихідного слова визначається, виходячи з наступних умов: три розряди необхідно мати для запису вихідної послідовності; три розряди потрібні для задання чергової адреси при відпрацюванні вихідної адреси. Загальна кількість розрядів вихідного слова ПЗП – 6.

Розрядність регістра визначається сумою вихідних розрядів ПЗП і розрядів вхідного керуючого сигналу, тобто (6 + 1) = 7.

В результаті отримаємо схему цифрового автомату, що приводиться на рис. 8.58.

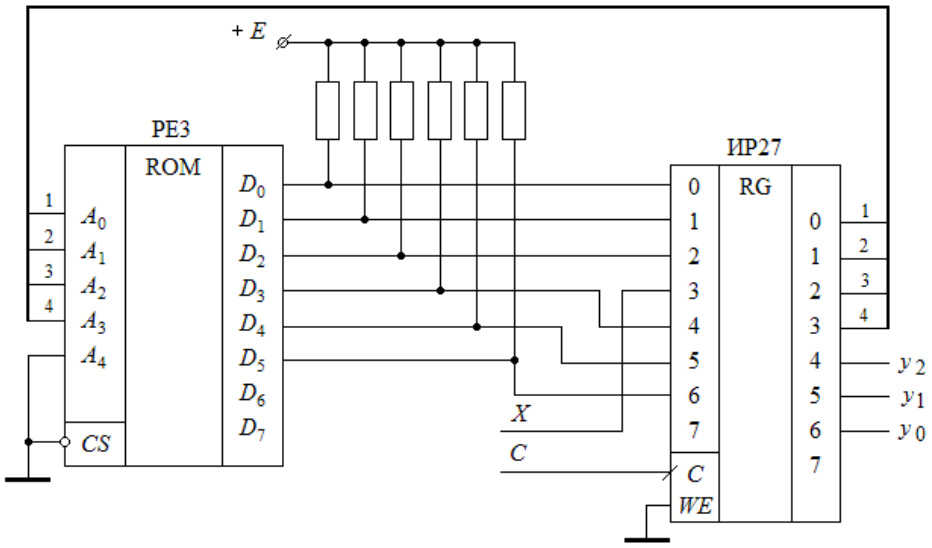


Рис. 8.58

Останній крок створення цифрового автомату – це розробка карти прошивки ПЗП. Ця робота вимагає високої уваги і дотримання певної послідовності. Перш за все, складемо детальну таблицю формування кодових послідовностей на виході автомату, на виході ПЗП та адресних кодів на вході ПЗП. Вона може мати надлишок рядків, в яких можуть дублюватись деякі данні, але, у той же час, містити в собі всю необхідну інформацію для формування кодів прошивок.

Оскільки кількість адресних входів – лише 4, то карта прошивок повинна мати лише один рядок.

Заповнення табл. 8.20 виконуватимемо крок за кроком, виходячи з аналізу роботи схеми у відповідності до часових діаграм.

Перед початком роботи вхідний сигнал $x_0 = 0$, і автомат повинен видавати на виході $y_2\, y_1\, y_0 = 111$. Приймемо, що виходи регістра вказують на нульову адресу ПЗП, тобто $A_2\, A_1\, A_0 = 000$. Вільні виходи, незалежно від адреси, повинні показувати нульові значення. Відповідно, у першому стовпці за адресою 0 запишемо $A_2\, A_1\, A_0 = 000$, $D_7\, D_6 = 00$, $D_5\, D_4\, D_3 = 111$. Код адреси ПЗП – 0.

Біти $D_2\, D_1\, D_0$ – це молодші адресні входи тієї адреси, за якою в наступному такті перейде автомат. Приймемо, що будемо послідовно інкрементувати їх за законом зміни бінарного коду, але, оскільки наступна адреса на наступному такті 1 задаватиметься вхідним сигналом x_0, то залишимо їх незмінними, тобто приймемо $D_2\, D_1\, D_0 = 000$. В результаті отримаємо код прошивки за нульовою адресою 38_{16} ($D_7 \ldots D_1 = 00111000$).

Таблиця 8.20

Адреса			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Код прошивки			38	0A	03	0C	18	38	38	38	19	0A	03	0C	1D	3D	38	38
Виходи ПЗП	A_0	D_0	0	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0
	A_1	D_1	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0
	A_2	D_2	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0
	y_2	D_3	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1
	y_1	D_4	1	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1
	y_0	D_5	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
	Вільні	D_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	виходи	D_7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Адреси ПЗП	$D_0 \rightarrow$	A_0	0	0	1	0	1	0	0	0	1	0	1	0	1	1	0	0
	$D_1 \rightarrow$	A_1	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0
	$D_2 \rightarrow$	A_2	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
	$x_0 \rightarrow$	A_3	0	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0
Код адреси			0	8	1	2	3	0	0	0	9	A	B	C	D	5	0	0

На такті 1 $x_0 = 1$ і адресний код, за яким буде йти звернення до ПЗП, $A_3\, A_2\, A_1\, A_0 = 1000_2 = 8_{16}$, тому необхідно заповнити стовпчик за адресою 8_{16}. Вихідний код

має бути $y_2\ y_1\ y_0 = 110 = D_5\ D_4\ \overline{D_3}$, а молодші адреси інкрементуємо на 1. Тобто за адресою 8_{16} записуємо код прошивки 19_{16}. За адресою 9_{16} записуємо бінарний код чергової адреси – A_{16} – і переносимо в клітини $D_2\ D_1\ D_0 = 010$, а вихідний код $y_2\ y_1\ y_0 = 110$, внаслідок чого маємо код прошивки ПЗП – 0A. Інкрементуючи на 1 молодші адреси $A_2\ A_1\ A_0 = 011$ і зчитуючи значення виходів з часової діаграми $y_2\ y_1\ y_0 = 000$, отримаємо черговий код прошивки ПЗП – 03_{16}, потім $0C_{16}$, $1D_{16}$. Наступний такт (шостий) забезпечує $y_2\ y_1\ y_0 = 111$ вже при $x_0 = 0$. Але оскільки в адресному коді змінюється старший розряд, то інкрементувати молодші розряди не слід. Як результат, за адресою D_{16} запишемо код $3D_{16}$ і перейдемо на п'яту адресу. Оскільки за цією адресою починає повторюватись режим, еквівалентний нульовій адресі, то, заповнивши стовпець, отримуємо код 38_{16}, і автомат знову очікуватиме зміни вхідного сигналу.

За умовою задачі, автомат, розпочавши цикл, не повинен реагувати на зміну вхідного сигналу, тому, змінивши в адресі старший розряд з 1 на 0, заповнюємо стовпці 1…3 кодами, що розміщені за адресами 9, A, B. Невикористовувані стовпці 6, 7 і E, 8 кодуємо прошивкою 38 на той випадок, щоб автомат, попавши на такі адреси, перейшов на нульову адресу.

З останнього прикладу витікає, що при наявності надлишку ємності РПЗП є можливість досить вільно використовувати його карту прошивок.

8.4. Оперативні запам'ятовуючі пристрої

Назва *"оперативні запам'ятовуючі пристрої"* (ОЗП) фактично відображає функціональне призначення такого роду пристроїв пам'яті, які призначені для швидкого запису, зчитування та тимчасового зберігання обсягів інформації, які використовуються оперативно при виконанні програм. В англомовній літературі початкова назва такої пам'яті була *Read/Write Memory* (*RWM*), тобто, знову ж таки, пам'ять для читання і запису, і зберігання, в будь-який час. Більш пізня і сучасна її назва – *Random Access Memory* (*RAM*) (пам'ять з довільним доступом) –означає, що такі пристрої пам'яті мають можливість зчитувати або записувати біт інформації незалежно від його розміщення в пам'яті. З цієї точки зору *ROM* (*Read Only Memory*) (або ПЗП – постійні запам'ятовуючі пристрої) також відносяться до *RAM*. Але, незважаючи на таке непорозуміння, абревіатура *RAM* широко використовується як англомовна назва ОЗП.

У цілому ОЗП розділяють на дві групи – *статичні* та *динамічні*. Різниця між ОЗП кожної з груп досить суттєва, і проявляється у різноманітних характеристиках – від областей застосування до вартості. *Статичні ОЗП* в якості елементів пам'яті використовують *D*-тригери, інформація в яких при наявності напруги живлення зберігатиметься від запису до перезапису. *Динамічні ОЗП* в якості елементів пам'яті використовують конденсатори ємністю порядку 0,5 пФ. Тривалість зберігання інформації у статичних ОЗП необмежена. У динамічних вона обмежується часом саморозряду конденсаторів, що вимагає спеціальних засобів відновлення (регенерації) даних і додаткових витрат часу на цей процес. Звідси і витікає суттєва різниця як у схемотехніці обох типів пам'яті, так і в принципах роботи з нею.

8.4.1. Статичні ОЗП

Сучасні ОЗП статичного типу відносяться до пристроїв пам'яті високої вартості, і тому у мікропроцесорних системах використовуються досить обмежено. Основною перевагою статичних ОЗП є висока швидкодія, що визначає їх використання у швидкодіючих системах обробки інформації. Як КЕШ-пам'ять вона широко використовується для суттєвого підвищення швидкості взаємодії процесора з основною (динамічною) пам'яттю у персональних комп'ютерах, а також є основною пам'яттю у потужних мультипроцесорних системах.

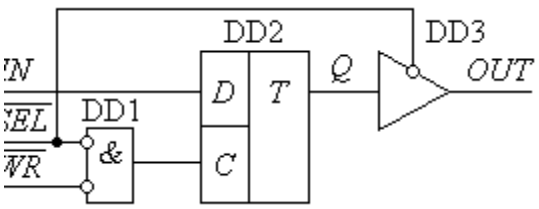


Рис. 8.59

На сучасному рівні розвитку технології напівпровідникової техніки статичні ОЗП виготовляються на основі КМОН-елементів. Основою статичних ОЗП є елемент пам'яті, побудований на синхронних статичних *D*-тригерах, який структурно має вигляд, приведений на рис. 8.59.

При виборі елемента пам'яті ($\overline{SEL} = 0$) дані, що зберігаються в *D*-тригері, передаються на вихід через вихідний підсилювач з *Z*-станом. При цьому дані в

D-тригері залишаються незмінними. При необхідності змінити вміст *D*-тригера подається сигнал $\overline{WR} = 0$, внаслідок чого дані, що знаходяться на вході *IN*, будуть занесені в елемент пам'яті.

Однобітні елементи пам'яті поєднуються в матрицю, в якій керуючі сигнали кожного з елементів об'єднуються. Матриці пам'яті будуються на основі схемотехніки, подібної до репрограмованих запам'ятовуючих пристроїв. На рис. 8.60 як приклад статичного ОЗП приводиться блок-схема мікросхеми SRM2114 фірми SU WA SEIKOSHA (Японія) з організацією 1024 слова по 4 біти (4 Кбіт). Порівняння приведеної блок-схеми ОЗП мікросхеми пам'яті з блок-схемою РПЗП показує їх значну схожість, але, незважаючи на це, ОЗП має ряд особливостей при експлуатації, які будуть розглянуті нижче.

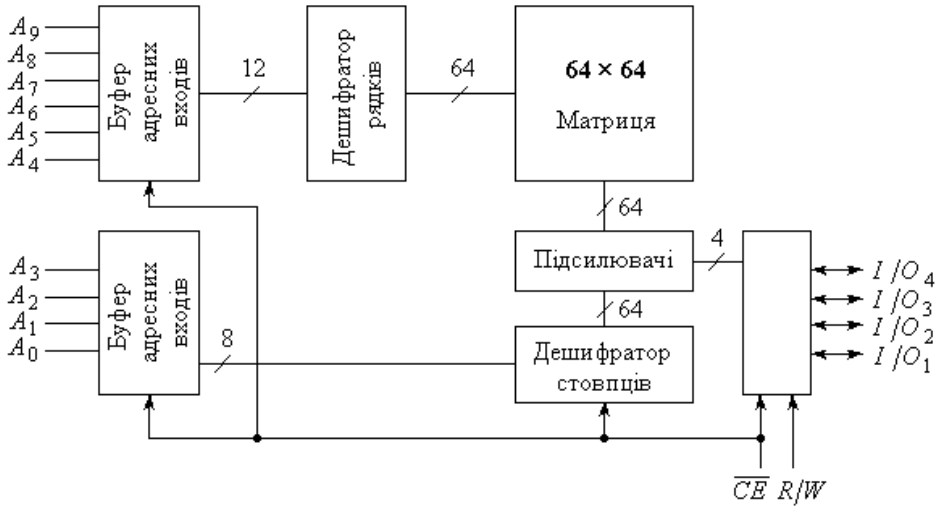


Рис. 8.60

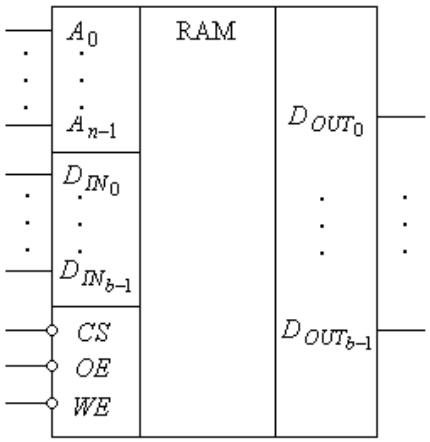


Рис. 8.61

Умовне зображення мікросхеми ОЗП приводиться на рис. 8.61. Призначення виводів відоме з попереднього матеріалу, операції запису та зчитування у загальному плані зрозумілі.

Для забезпечення операції зчитування необхідно забезпечити низькі рівні сигналів на входах $\overline{CS}$ та $\overline{OE}$ і за заданою адресою

вибрати необхідні дані, які з матриці пам'яті будуть передані на виводи $D_{OUT\,0}\ldots D_{OUT\,(b-1)}$.

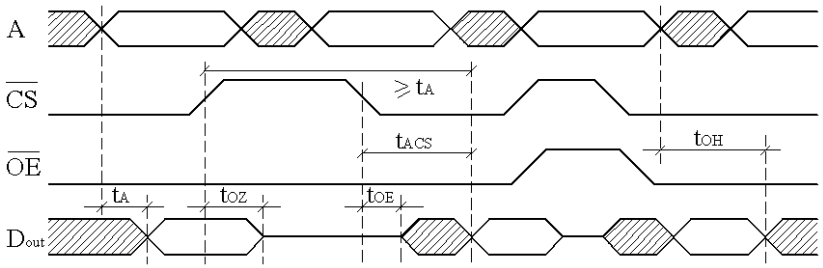
Для забезпечення операції запису вибирається необхідна адреса $A_0\ldots A_{n-1}$, необхідна мікросхема $\overline{CS}=0$, необхідні для зберігання дані подаються на входи $D_{IN\,0}\ldots D_{IN\,(b-1)}$ і за сигналом $\overline{WE}=0$ заносяться у тригери матриці пам'яті.

У загальному плані робота ОЗП описується таблицею станів табл. 8.21, яка характеризує статичні значення сигналів для кожного з режимів.

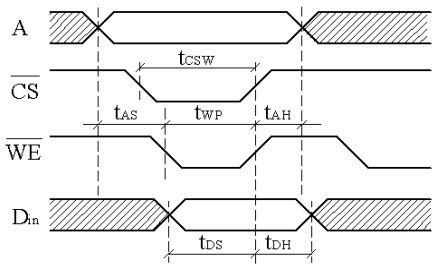
Таблиця 8.21

$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	A	D_{IO}	Режим
1	x	x	x	Z	Зберігання
0	x	0	A	DI	Запис
0	0	1	A	DO	Зчитування

Але для забезпечення надійного запису, так само як і зчитування, при роботі з ОЗП необхідно чітко дотримуватися послідовності подачі сигналів та часових співвідношень між ними, які задаються часовими діаграмами та відповідними часовими співвідношеннями, подібними до відповідних параметрів ПЗП (рис. 8.62).



а)



б)

Рис. 8.62

На рис. 8.62, *а* приведені часові діаграми для режиму зчитування, а на рис. 8.62, *б* – для режиму запису.

Часові параметри мають наступну інтерпретацію:

- t_A (*Access time from address*) – час доступу до даних після подачі установлених адресних сигналів (час перехідних процесів зображено заштрихованими ділянками). Це мінімальний інтервал часу між установленими значеннями адресних сигналів до моменту початку зчитування даних за умови, що сигнали $\overline{CS}$ і $\overline{OE}$ підготовлені раніше. Цей інтервал часу мається на увазі, коли говорять про час доступу до даних конкретного типу статичної пам'яті;
- t_{ACS} (*Access time from chip select*) – час доступу до даних після подачі і установлення $\overline{CS}$. Це мінімальний інтервал часу між установленням сигналу $\overline{CS}$ і моментом початку зчитування даних при установлених попередньо адресних сигналах і сигналі $\overline{OE}$;
- t_{OE} (*Output enable time*) – час переходу вихідних підсилювачів із *Z*-стану в активний. Визначається як інтервал часу з моменту переходу одного з керуючих сигналів $\overline{CS}$ або $\overline{OE}$ в активний стан (низький рівень) до моменту появи даних (неустановлений режим) на вихідних шинах при установлених значеннях адресного сигналу;
- t_{OZ} (*Output disable time*) – час переходу вихідних підсилювачів з активного у *Z*-стан. Він визначається як інтервал часу між переходом одного з керуючих сигналів $\overline{CS}$ або $\overline{OE}$ у пасивний стан (високий рівень) та моментом, з якого виходи мікросхеми перейдуть у *Z*-стан при установлених значеннях адресного сигналу;
- t_{OH} (*Output hold time*) – час підтримки виходу. Цей параметр визначає, як довго можна знімати дані з виходів мікросхеми після зміни адресного коду;
- t_{AS} (*Address setup time before write*) – час установлення адресного коду перед записом. Цей інтервал часу важливий з точки зору надійного і непошкодженого запису даних за установленим адресним кодом. Визначається з моменту установлення коду адреси до моменту активізації останнього.

- t_{AH} (*Address hold time after write*) – час підтримки адреси після запису. Цей параметр аналогічний t_{AS} і визначає інтервал часу, протягом якого адресний код повинен залишатися без зміни після переводу керуючих сигналів $\overline{CS}$, $\overline{OE}$ у пасивний стан (високий рівень);
- t_{CSW} (*Chip-select setup before end of write*) – час підтримки активного рівня керуючого входу $\overline{CS}$ при виконані операції запису;
- t_{WP} (*Write-pulse width*) – тривалість активного рівня керуючого входу $\overline{WE}$ для забезпечення надійного запису;
- t_{DS} (*Data setup time before end of write*) – інтервал часу, протягом якого всі сигнали повинні бути незмінними для забезпечення надійного запису;
- t_{DH} (*Data hold time after end of write*) – аналогічно t_{DS}, тобто дані на вході повинні залишатись незмінними по закінченню циклу запису.

Виробники статичних мікросхем оперативної пам'яті визначають два типи циклів запису: керований сигналом $\overline{WE}$ та керований сигналом $\overline{CS}$. Різниця між ними полягає лише у тому, який з сигналів активізується. При цьому несуттєво, який з цих сигналів першим переходить у пасивний стан, оскільки після цього рівень сигналу на іншому вході не має значення.

Стандартні мікросхеми статичної пам'яті мають ємність і організацію, подібні до ПЗП і РПЗП. Широко використовувані мікросхеми мають ємність до 4 Мбіт і мінімальний час доступу 2…3 нс.

Найширше використання статичні ОЗП знаходять у малих мікропроцесорних системах, вмонтованих системах керування, телефонії та ін. Швидкодіючі статичні ОЗП використовуються як КЕШ-пам'ять у високопродуктивних обчислювальних машинах і системах.

8.4.2. Динамічні ОЗП (*DRAM*)

Складність запам'ятовуючих елементів у статичних ОЗП обмежує можливості досягнення високої щільності і, відповідно, високої ємності

виготовлення мікросхем пам'яті. Підвищення цих параметрів досягається тим, що в якості елемента пам'яті використовується конденсатор малої ємності $C_{ЗЕ}$, заряд якого забезпечується через МОН-транзистор VT (рис. 8.63).

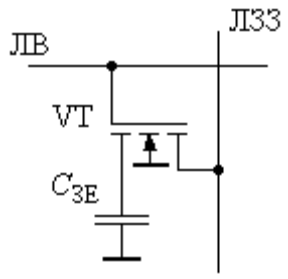


Рис. 8.63

Транзистор забезпечує комутацію конденсатора з *лінією запису/зчитування* (ЛЗЗ), або *бітовою лінією* (*Bit line*).

У режимі зберігання транзистор закритий. Заряджений конденсатор зберігає **лог. "1"**, а розряджений – **лог. "0"**.

При виборі даного ЗЕ на затвор по лінії ЛВ (*лінії вибірки*, або *лінії слова* (*Word line*)) подається напруга, що відкриває транзистор, і заряд конденсатора передається на ЛЗЗ. Процес запису-зчитування більш детально пояснюється рис. 8.64.

При необхідності запису одиниці або нуля у відповідний ЗЕ він активізується подачею сигналу на відповідну ЛВ. Лінія ЛЗЗ ключами K_1 або K_0 приєднується до джерела напруги високого рівня E або до загальної шини. Внаслідок такої комутації в конденсатор $C_{ЗЕ}$ записується відповідна інформація через низькоомний канал транзистора.

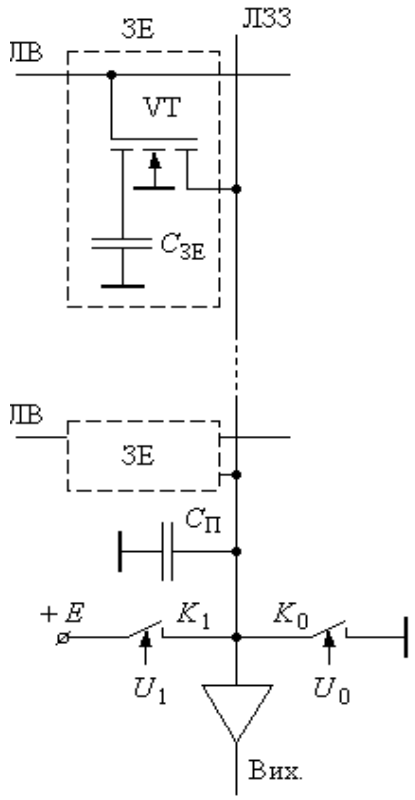


Рис. 8.64

Процес зчитування дещо складніший, і ця складність пояснюється тим, що ємність $C_{ЗЕ}$ має малу величину – значно меншу, ніж паразитна ємність $C_П$ лінії ЛЗЗ. Обумовлено це тим, що ємність $C_{ЗЕ}$ створюється на основі структури польового транзистора, а $C_П$ – це ємність лінії ЛЗЗ досить великої протяжності з великою кількістю підключених ЗЕ.

Перед операцією зчитування забезпечується

попередній заряд конденсатора $C_П$ до напруги, яка дорівнює половині напруги E. Якщо при зчитуванні заряд конденсатора $C_{ЗЕ}$ був нульовим, то конденсатор $C_П$ лінії ЛЗЗ частково розрядиться на $C_{ЗЕ}$ до вирівнювання напруги на них, і результуюча напруга на них стане дещо меншою, ніж $E/2$, на величину ΔU, яка і є сигналом про зчитування нуля.
Якщо ж конденсатор $C_{ЗЕ}$ попередньо зберігав одиницю, то, аналогічно, напруга на $C_П$ зросте на ΔU, що буде служити сигналом про зчитування одиниці. В обох випадках операція зчитування змінює напругу на конденсаторі $C_{ЗЕ}$.

Для того щоб виділити незначні відхилення потенціалу ЛЗЗ, використовуються спеціальні підсилювачі – регенератори, які забезпечують перетворення малих відхилень ΔU в сигнали логічної одиниці і нуля і в той же час відновлюють потенціал на конденсаторі $C_{ЗЕ}$.

Незначна ємність конденсатора $C_{ЗЕ}$ призводить до того, що, незважаючи на великий опір закритого каналу транзистора, заряджений до потенціалу логічної одиниці, він розряджається протягом декількох мікросекунд. Тому системи динамічної пам’яті будуються з використанням циклів регенерації. Наявність і необхідність циклів регенерації, яка забезпечується адресним зверненням до кожного ЗЕ, ускладнює взаємодію такої пам’яті з процесором і зовнішніми пристроями.

Ще однією особливістю динамічних ОЗП є *мультиплексування адресної шини*. Необхідність мультиплексування обумовлена двома причинами: перша з них – це потреба у зменшенні кількості виводів мікросхеми, особливо для пристроїв з великою ємністю пам’яті, друга – те, що адресні входи мікросхеми використовуються по-різному (наприклад, при регенерації адресу стовпця не використовується). Ідеологія мультиплексування полягає у тому, що весь адресний простір розділяється на дві частини – так звані *напівадреси*. Ними є окремо адреси рядків і адреси стовпців, які подаються на одні й ті ж самі виводи мікросхеми пам’яті, але супроводжуються відповідним стробом – $\overline{RAS}$ (***R**ow **A**ddress **S**trobe*) і $\overline{CAS}$ (***C**olumn **A**ddress **S**trobe*).

Умовне зображення мікросхеми динамічної пам’яті серії 4000 (4164 з організацією 64K × 1) приведене на рис. 8.65.

На рис. 8.66 показана схема, яка пояснює метод адресації з мультиплексуванням. ЗЕ мікросхем динамічної пам’яті організовані у вигляді прямокутної (зазвичай квадратної) матриці. У приведеній схемі для звернення до ЗЕ використовуються дешифратори стовпців і рядків. На вході кожного дешифратора встановлений регістр-фіксатор. Спочатку на адресні входи подається молодша частина розрядів адреси ($A_0 \dots A_5$), після чого поступає строб адреси рядка $\overline{RAS}$, за яким ці розряди адреси записуються у регістр-фіксатор дешифратора адреси рядка. Потім подається старша частина розрядів адреси ($A_6 \dots A_{11}$), яка за стробом адреси стовпця $\overline{CAS}$ записується у регістр-фіксатор дешифратора адреси стовпця. Після цього у регістрах-фіксаторах зберігатиметься повна адреса, яка надає можливість звертатися до кожного ЗЕ. Для зменшення часу регенерації при зчитуванні одного ЗЕ рядка матриці регенерується весь рядок.

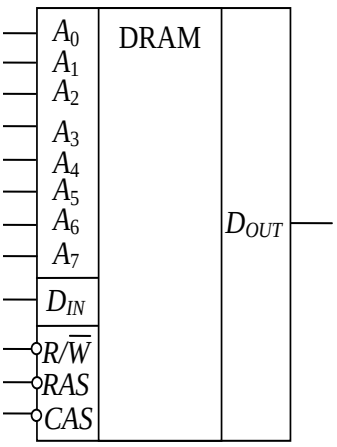


Рис. 8.65

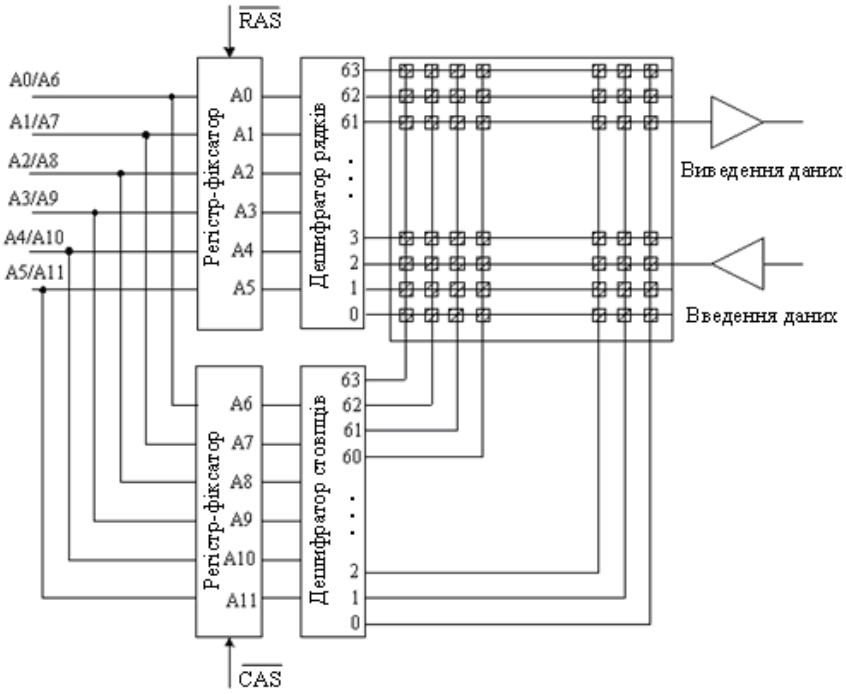


Рис. 8.66

8.4.3. Використання ОЗП

На ринку електронних компонентів існує великий вибір ОЗП з різною ємністю, кількістю розрядів, різними методами керування і технічними характеристиками. Для ко жної інженерної задачі обробки інформації необхідно підбирати мікросхеми пам'яті, які найкраще відповідають вимогам поставленої задачі. У цьому плані динамічні ОЗП мають досить обмежені області використання в якості системної оперативної пам'яті комп'ютерів.

Статичні ОЗП за характером доступу до пам'яті можуть бути розподілені на:

- ОЗП з паралельним (або довільним) доступом;
- ОЗП з послідовним доступом.

Особливість ОЗП з довільним доступом полягає у можливості довільної зміни порядку подачі адресних кодів на входи мікросхеми без будь-яких наслідків для її роботи. Заміна адресних розрядів, яка може бути використана для полегшення розводки друкованих плат, не має ніякого значення для функціонування мікросхеми. У цьому режимі можливо записувати інформацію за будь-якою адресою та зчитувати у довільному порядку. Але такий спосіб доступу вимагає формування складних послідовностей вхідних сигналів мікросхем пам'яті. Тобто для запису або зчитування необхідно сформувати код адреси необхідних ЗЕ, підготувати дані (при запису) і виконання операції супроводжувати сигналами *CS*, *WR*/*RD*, які повинні бути повністю узгодженими в часі. Такий режим в основному використовується в комп'ютерах і мікропроцесорних системах, де вимагається висока універсальність і гнучкість використання пам'яті, у тому числі і декількома процесорами.

Послідовний спосіб доступу до пам'яті передбачає більш простий порядок звернення. У цьому випадку відпадає необхідність задавати код адреси використовуваного ЗЕ, оскільки він формується схемою автоматично.

Для запису даних необхідно лише мати їх двійковий код та подати сигнал запису. Аналогічно забезпечується і зчитування. Автоматичне формування коду адреси забезпечується лічильником.

Виділяють три основні типи ОЗП з послідовним доступом:

- пам’ять типу *FIFO*;
- пам’яті типу *LIFO*;
- пам’яті для зберігання машинних даних.

Вище вже розглядалися перші два типи на основі регістрів, тому розглянемо лише приклади побудови на основі ОЗП.

На рис. 8.67 зображена функціональна схема пам’яті типу *FIFO*. Адреси ОЗП формуються двома лічильниками, вихідні коди яких мультиплексуються на адресний вхід ОЗП. Потік даних поступає на вхід ОЗП по шині *DI*, а виводиться на шину *DO*.

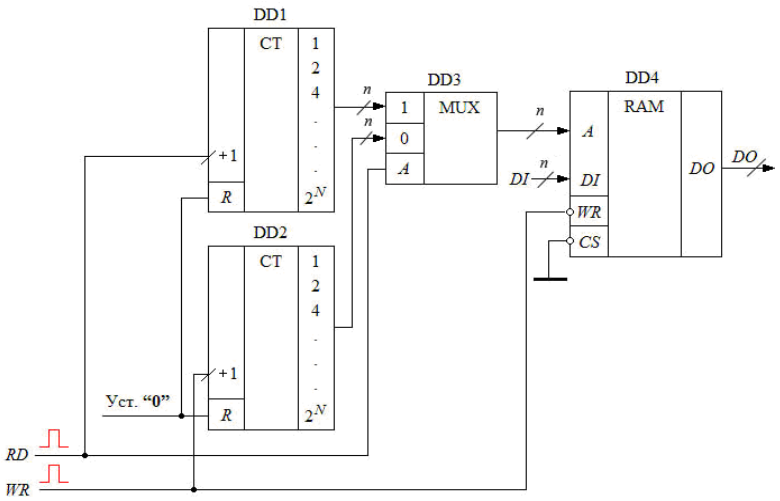


Рис. 8.67

Перед початком роботи лічильники DD1 і DD2 встановлюються в нуль. При виконані операції запису сигнал *RD* має низький рівень і перемикає мультиплексор DD3 на передачу адресного коду, сформованого лічильником DD2 за фронтом сигналу *WR*. Підготовленні дані з шини *DI*, за сигналом запису *WR*, записуються в ЕП за адресами, які задані кодом лічильника DD2. Запис починається з нульової адреси і виконується у відповідності до

зростаючого двійкового коду.

Зчитування інформації також починається з нульової адреси, яка формується лічильником DD1. При подачі сигналу зчитування *RD* за його фронтом інкрементується вміст лічильника DD1, а потім за високим потенціалом мультиплексор DD3 передає адресний код на ОЗП DD4, і відповідні дані зчитуються на шину *DO*.

Умови правильної роботи наступні:

- тривалість сигналу *WR* не повинна бути меншою мінімально допустимої тривалості сигналу *WR* мікросхеми пам’яті;
- тривалість сигналу *RD* не повинна бути меншою суми затримок на перемикання мультиплексора і часу вибірки мікросхеми пам’яті за адресним сигналом;
- інтервал подачі сигналу *WR* не повинен бути меншим суми затримок на перемикання лічильника запису, затримки на перемикання мультиплексора і мінімально допустимої тривалості сигналу *WR* мікросхеми пам’яті;
- інтервал подачі сигналу *RD* не повинен бути меншим суми затримок на перемикання лічильника читання, тривалості затримки мультиплексора і мінімального інтервалу часу на зчитування мікросхеми пам’яті.

Приклад функціональної схеми пам’яті типу *LIFO* приводиться на рис. 8.68. Вона має значно простішу структуру і використовує реверсивний лічильник.

Запис підготовлених даних *DI* забезпечується низьким рівнем сигналу $\overline{WR}$. Після виконання операції запису вміст лічильника за фронтом сигналу $\overline{WR}$ інкрементується на одиницю, формуючи тим самим адресу для чергових даних.

Зчитування забезпечується за фронтом імпульсу *RD*, який декрементує вміст лічильника.

Третій тип пам’яті, призначений для тимчасового зберігання масивів даних, пояснюється функціональною схемою, що приведена на рис. 8.69.

Особливість такого модуля пам’яті полягає в тому, що спочатку масив

даних визначеного розміру послідовно, починаючи з молодшої адреси, що задається лічильником DD2, записується з шини *DI* в елементи пам'яті ОЗП DD3.

Для зчитування лічильник встановлюється в нуль зовнішнім сигналом, а потім за сигналом *RD* зчитується у тій самій послідовності на зовнішню шину *DO*.

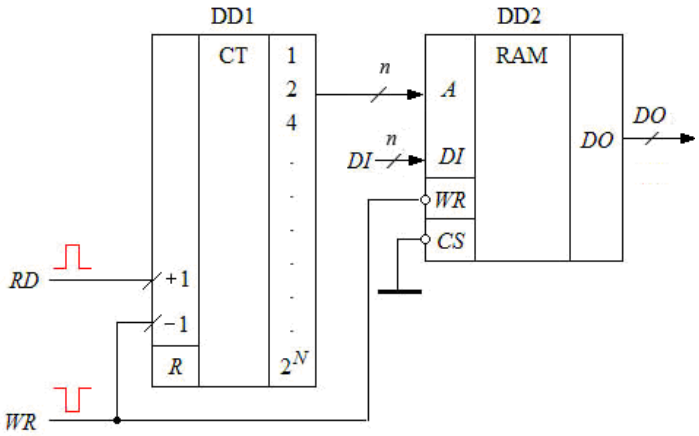


Рис. 8.68

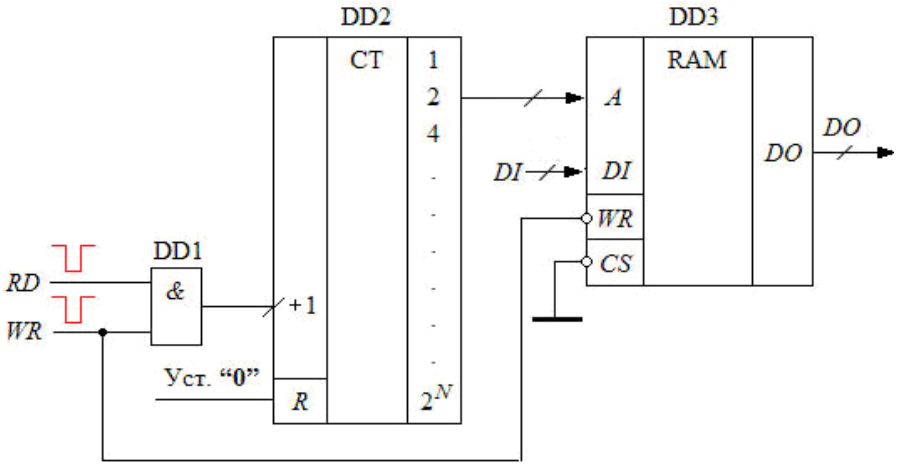


Рис. 8.69

В усіх розглянутих випадках ОЗП використовується подібно до регістрів великої ємності.

8.4.3.1. Використання ОЗП як інформаційного буфера

Буферна пам’ять використовується як модуль пам’яті, призначений для узгодження роботи декількох цифрових пристроїв, що мають різну швидкодію (рис. 8.70).

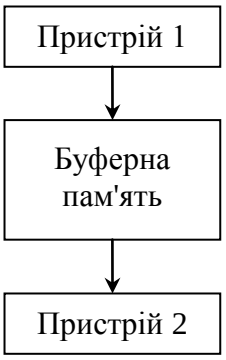


Рис. 8.70

Швидкодіючий **Пристрій 1** відправляє дані в буферну пам’ять, де вони накопичуються протягом деякого інтервалу часу, а потім **Пристрій 2** з іншою (здебільшого, меншою) швидкістю зчитує прийняті дані з буфера. Така технологія обміну не тільки дозволяє узгодити швидкості передачі та прийому даних, а й надає можливість розв’язувати ряд інших задач взаємодії двох пристроїв – наприклад, підвищити рівень їх незалежності один від одного, рознести у часі передачу і прийом, та ін. Такий взаємозв’язок можливий в організації обміну швидкодіючого процесора з повільними периферійними пристроями – наприклад, з пристроями зовнішньої пам’яті, принтерами, модемами, комп’ютерними мережами. У комп’ютерах ряду версій в якості такої буферної пам’яті використовувалась частина ОЗП комп’ютера з жорстко виділеним адресним простором.

Відмінність інформаційних буферів, що розглядаються, полягає в тому, що до пам’яті через адресні коди мають доступ як мінімум два пристрої, що суттєво ускладнює розділення потоків інформації та взаємодію між пристроями.

Інформаційні буфери можуть бути однонаправленими (сканери, принтери) і двонаправленими (комп’ютерні мережі, модеми та ін.).

Буфери можуть забезпечувати безперервний обмін між пристроями або періодичний. Прикладом безперервного обміну можна назвати пам’ять відеокарти комп’ютера. Приклад періодичних обмінів – робота комп’ютера із зовнішньою пам’яттю.

Інформаційні буфери з періодичним режимом роботи можуть бути організовані за типом *FIFO* або *LIFO*.

Розглянемо декілька типових схем інформаційних буферів.

Функціональна схема першого з них – однонаправленого буфера з періодичним режимом обміну за принципом *FIFO* – приведена на рис. 8.71.

Перед початком роботи виконується установка лічильника DD4 і тригера DD5 в початкові стани. Дані в послідовному форматі подаються на вхід *DI* ОЗП DD7. Сигнал з інверсного виходу тригера DD5 забороняє запис інформації в регістр DD8 і, у той же час, через елемент **АБО** DD6 дозволяє проходження сигналу $\overline{WR}$ на вхід лічильника DD4.

При передачі інформації зовнішній пристрій подає імпульси $\overline{WR}$ низького рівня, за допомогою яких забезпечується запис даних в ОЗП DD7 за встановленою лічильником DD4 адресою.

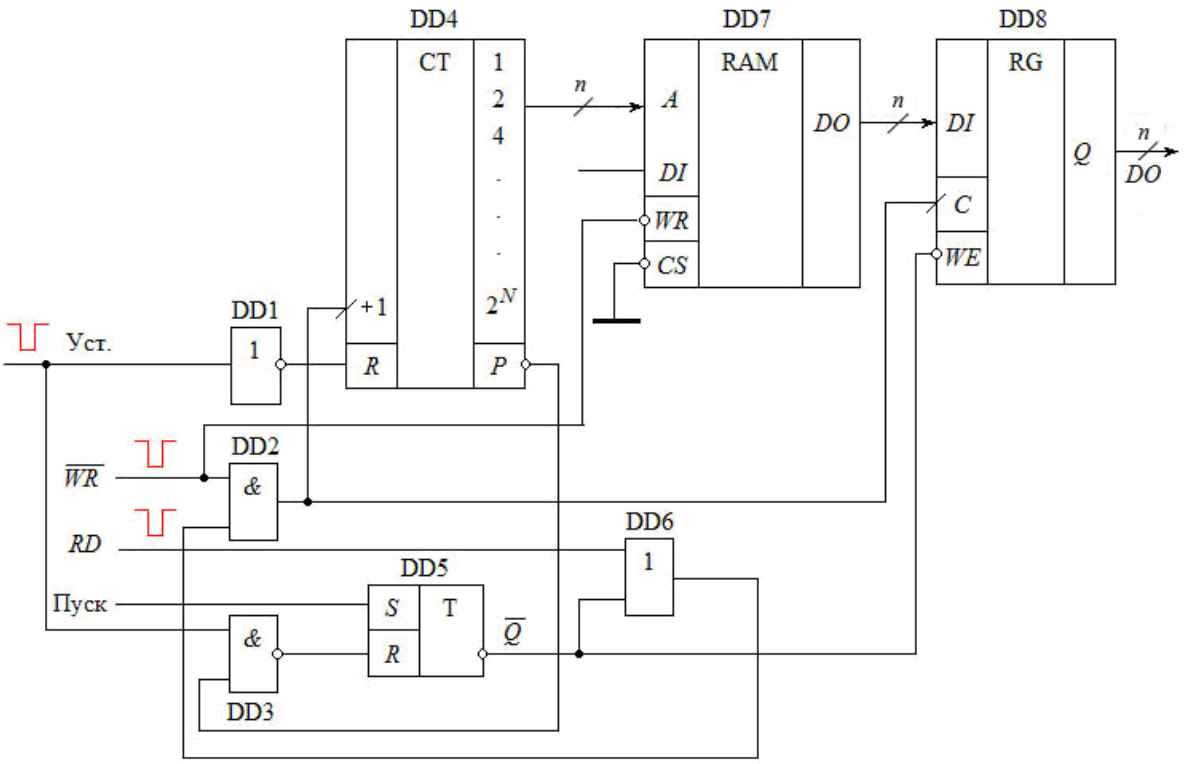


Рис. 8.71

За фронтом (0 → 1) сигналу $\overline{WR}$ вміст лічильника DD4 інкрементується, формуючи код наступної адреси. Запис інформації ведеться до повного

заповнення ЕП ОЗП і, відповідно, до повного перебору його адрес, що формуються лічильником.

По закінченню операції запису пристрій, що передавав інформацію, генерує сигнал “Пуск”, яким змінює стан тригера, а пристрій, що приймає інформацію, починає генерувати сигнал *RD* зі своєю частотою, яка може відрізнятись від частоти передаючого пристрою. Сигнал *RD* через елементи DD6 і DD2 подається на вхід лічильника для інкрементування його вмісту і одночасно на вхід регістру DD8 для перезапису вмісту ОЗП у регістр. Дозвіл на запис у регістр DD8 формується низьким рівнем інверсного виходу тригера.

По закінченню циклу зчитування сигнал переносу *P* лічильника приймає низький рівень, яким через ЛЕ DD3 встановлюється тригер DD5 у початковий стан (режим запису інформації).

Розглянемо тепер особливості побудови і роботи більш складної структури двонаправленого буфера з періодичним режимом обміну типу *LIFO,* функціональна схема якого приведена на рис. 8.72.

Такий буфер дозволяє приймати і відправляти масиви даних довільної довжини з заданою швидкістю, що характерно, наприклад, для адаптерів локальних мереж (мережевих карт). Зворотний порядок зчитування з буфера при взаємодії між собою двох буферів не матиме ніякого значення, оскільки кожен з них приймає інформацію з мережі у протилежному порядку, а комп’ютер передаватиме її у прямому.

Двонаправлений буфер має чотири режими роботи:

- *режим запису*. У цьому режимі дані, що подаються на вхід двонаправленого буферного підсилювача, записуються в буферну пам’ять;
- *режим передачі*. У цьому режимі дані з буферної пам’яті у зворотному порядку зчитуються з буферної пам’яті в регістр, який перетворює їх у послідовний формат і передає у мережу;
- *режим прийому*. У цьому режимі дані з мережі записуються в регістр, з якого у паралельному форматі перезаписуються в буферну пам’ять;

- *режим читання*. У цьому режимі дані з буферної пам'яті зчитуються у зворотній послідовності і через буфер передаються на шину даних.

Режим запису майже не відрізняється від аналогічних режимів у пристроях, розглянутих вище. Спочатку виконується установка лічильника в нульовий стан сигналом "Установка "0".

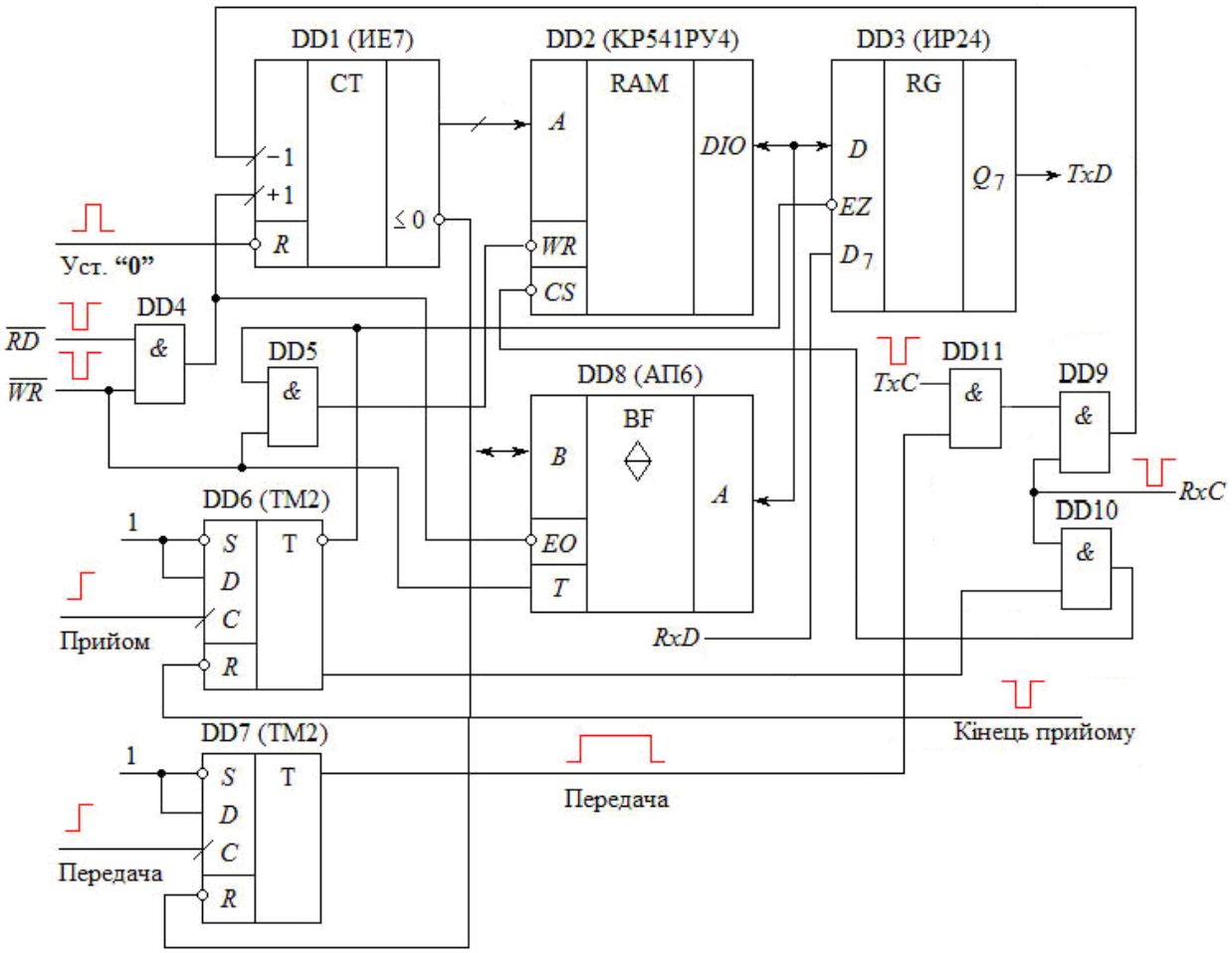


Рис. 8.72

Оскільки в режимі запису сигнал $\overline{RD}$ має високий рівень, то вихідний сигнал ЛЕ DD4 повторюватиме вхідний сигнал $\overline{WR}$, забезпечуючи низьким рівнем дозвіл на передачу інформації буфером DD8 у відповідності до його таблиці станів (див. табл. 8.22).

Таблиця 8.22

Входи		Виходи	
$\overline{EO}$	T	A_n	B_n
0	0	$B \rightarrow A$	Вх.
0	1	Вх.	$A \rightarrow B$
1	x	Z	Z

У той же час, низький рівень сигналу на вході T (*Transmit*) забезпечує передачу інформації від шини ***B*** до шини ***A***. Сигнал $\overline{WR}$

низького рівня на DD2 формується ЛЕ DD5, а сигнал $\overline{CS}$ – відповідно, ЛЕ DD10, оскільки прямий вихід тригера DD6 має низький рівень сигналу. Таким шляхом, узгоджена взаємодія сигналів $\overline{WR}$ з потоком даних дозволяє забезпечити послідовний запис даних в усі ЕП ОЗП DD2.

По закінченню масиву даних сигналом дозволу на передачу "Передача" тригер DD7 високим рівнем сигналу з прямого виходу забезпечує сигналом *TxC* (*Transmitted Clock*) декрементування адресного коду у зворотному порядку. Дані при кожній зміні адресного коду перезаписуються у паралельному форматі в регістр DD3, а потім у послідовному форматі зчитуються на шину *TxD* (*Transmitted Data*) (елементи схеми керування регістром DD3 для перетворення паралельного формату у послідовний не зображені). Коли вміст лічильника декрементується до нуля, формується сигнал низького рівня на виході його переносу (≤ 0), який встановлює лічильник DD7 у початковий стан. Таким чином забезпечується передача всього масиву даних в мережу в зворотному порядку.

Режим прийому інформації з мережі починається встановленням сигналом "Прийом" тригера DD6. Високий рівень сигналу на його прямому виході забезпечує прийом синхросигналу *RxC* (*Received Clock*), який через DD9 забезпечує декрементування адресного коду, сформованого лічильником DD1, і в той же час через DD10 формує сигнал для запису вмісту регістра. Інформація з мережі *RxD* (*Received Data*) записуються в регістр в послідовному форматі (схема керування операцією запису в регістр з мережі не зображена). По закінченню прийому сигналом "Кінець прийому" тригер DD6 встановлюється у нуль.

Режим читання інформації з пам'яті забезпечується через шинний буфер DD8, який сигналом високого рівня $\overline{WR}$ налаштований на передачу інформації з ОЗП (шина ***A***) на шину ***B*** за сигналом $\overline{RD}$, який забезпечує читання за зростаючим кодом адреси ОЗП, забезпечуючи запис у прямій послідовності

масиву даних.

У приведеному розділі не розглядалося використання пристроїв пам’яті у мікропроцесорних системах, оскільки такий матеріал вимагає досконалого знання режимів роботи мікропроцесорів. При бажанні читач може звернутись до спеціалізованих видань.

КОНТРОЛЬНІ ПИТАННЯ

1. Поясніть різницю між енергонезалежними і енергозалежними пристроями пам’яті.
2. Дайте визначення ПЗП.
3. Приведіть умовне позначення ПЗП та поясніть призначення виводів.
4. У чому полягає різниця між одновимірними і багатовимірними ПЗП?
5. Поясніть, як можна реалізувати систему логічних функцій на основі ПЗП?
6. Що означає термін *“гексадецимальний лістінг”*?
7. Поясніть, як створюється таблиця прошивок, на основі самостійно запропонованого Вами прикладу.
8. Наведіть структурну схему ПЗП та поясніть призначення її елементів.
9. У чому полягає перевага двовимірних ПЗП? Які особливості їх побудови?
10. Що означає термін *“режим енергозбереження”*? За рахунок чого він досягається в сучасних мікросхемах ПЗП?
11. Як визначається інформаційна ємність ПЗП? Як вона пов’язана з організацією ПЗП?
12. Доведіть, що в схемі ПЗП, яка приведена на рис. 8.4, наявність діода, що з’єднує лінії слова і біта, еквівалентна запису логічної одиниці.
13. Поясніть, як, використовуючи ПЗП з організацією 4×16, можна

реалізувати дві функції, які зчитуватимуться одночасно (наприклад, функції синуса і косинуса).

14. Наведіть основні динамічні параметри ПЗП та дайте їм характеристику.

15. Які ПЗП називають репрограмованими? Які засоби збереження інформації в них закладені?

16. Які типи репрограмованих ПЗП Вам відомі? У чому полягають їх особливості?

17. Наведіть умовне позначення РПЗП та поясніть призначення виводів.

18. Наведіть основні динамічні параметри РПЗП та дайте їх пояснення.

19. У чому полягає різниця між послідовними та паралельними РПЗП (*EEPROM*)?

20. Дайте характеристику основним режимам роботи паралельних РПЗП; послідовних РПЗП.

21. Наведіть приклади паралельних і послідовних РПЗП, дайте їм коротку характеристику.

22. Приведіть умовні позначення паралельних і послідовних РПЗП, поясніть призначення виводів.

23. Поясніть особливості кожного з трьох способів зчитування даних з послідовних РПЗП.

24. У чому полягають особливості РПЗП із трьохпровідною послідовною шиною?

25. У чому полягає особливість побудови *flash*-пам’яті?

26. Поясніть основні режими операцій із *flash*-пам’яттю.

27. У чому полягають особливості операцій програмування *flash*-пам’яті?

28. Які елементи використовуються в якості запам’ятовуючих в ОЗП? Поясніть переваги кожного з них.

29. Як нарощується ємність ОЗП?

30. Наведіть приклади використання ОЗП. Дайте пояснення на прикладах,

які наведені у розділі.

31. Дайте порівняльну характеристику статичних і динамічних ОЗП.

32. Поясніть особливості використання ОЗП у буферах типу *FIFO* та *LIFO*.

33. Як Ви розумієте термін *"інформаційний буфер"*? У чому полягає особливість його роботи з використанням ОЗП?

ВПРАВИ І ЗАВДАННЯ

1. Мікросхема КР554РТ4 має організацію матриці пам'яті 256×4. Два входи $\overline{CS_1}$ та $\overline{CS_2}$ об'єднані по &. Що необхідно зробити, щоб отримати мікросхему з організацією 128×4? Приведіть відповідну схему.

2. Розробити схему пристрою пам'яті на основі мікросхеми КР556РТ4 з організацією 512×4.

3. Розробити схему пристрою пам'яті на основі мікросхеми КР556РТ4 з організацією 256×16.

4. Розробити схему пристрою пам'яті на основі мікросхеми КР556РТ4 з організацією 1024×8.

5. Розробити схему пристрою для отримання квадрату чотирьохрозрядного двійкового числа.

6. Прочитати карту прошивки, що приведена у табл. 8.23. Пояснити призначення пристрою, що реалізований з відповідною ПЗП.

Таблиця 8.23

Адреса	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	01	04	09	10	19	24	31	40	51	64	79	90	A9	C4	E1

7. Побудувати карту прошивки для обчислення квадратів числа, що задається у двійково-десятковому коді і змінюється у межах від 0 до 15_{10}.

8. Побудувати карту прошивки перетворювача чотирьохрозрядного

двійкового коду в код семисегментних індикаторів при умові, що кожен сегмент засвічується при низькому рівні сигналу, який подається на нього.

9. Використовуючи ПЗП з організацією 16×4, розробити схему пристрою для реалізації системи логічних функцій:

$$\begin{cases} y_0 = \bigvee_0^{15} 0, 2, 4, 8, 11, 14, 15; \\ y_1 = \bigvee_0^{15} 1, 2, 3, 5, 6, 8, 11, 12, 15; \\ y_2 = \bigvee_0^{15} 0, 2, 4, 5, 7, 9, 10, 14; \\ y_3 = \bigvee_0^{15} 1, 3, 5, 8, 9, 10, 12, 14. \end{cases}$$

10. Використовуючи ПЗП, розробити схему пристрою для циклічної генерації імпульсів, тривалість яких змінюється лінійно від 10 тактів до одного з паузою між ними в один такт, тобто імпульси повинні генеруватись у послідовності 10*T*, 9*T*, 8*T*, …, *T*, 10*T*, … Підібрати тип ПЗП та розробити карту прошивки.

11. Розробити карту прошивки для ПЗП генератора імпульсів експоненціальної форми, амплітуда яких повинна змінюватись від 0 до 10,24 В (на виході цифро-аналогового перетворювача) при зміні вхідного шестирозрядного коду у межах 00…3F.

12. Розробити на основі ПЗП скінченні автомати, робота яких описується часовими діаграмами, що приведені на рис. 5.15 і рис. 5.18 (**Розділ 5**).

13. Розробити принципову схему буфера *FIFO*, використовуючи лічильник КР1533ИЕ7, мультиплексор КР1533КП11 і мікросхему пам’яті КР1533РУ7. Дані мають чотирьохрозрядну форму. Мікросхема КР1533РУ7 має організацію 1024×1 з розділеними вхідними і вихідними даними. Входи $\overline{CS}$ і $\overline{WR}$ – інвертовані (активний рівень – низький).

14. Сформулювати умови правильної роботи буфера *FIFO*, функціональна схема якого зображена на рис. 8.67.

15. Розробити буфер *LIFO* на 64 слова розміром в 1 байт.

16. Побудувати часові діаграми для режимів запису і зчитування інформаційного буфера, схема якого приведена на рис. 8.71.

Розділ 9
ОСНОВИ ПРОЕКТУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ НА БАЗІ ПРОГРАМОВАНИХ ЛОГІЧНИХ МАТРИЦЬ (ПЛІС)

9.1. Основи побудови структур простих ПЛІС

Розглянуті у попередніх розділах різноманітні цифрові пристрої відносяться до стандартних інтегральних схем невисокого рівня інтеграції. Вони навіть на сучасному рівні розвитку мікропроцесорної техніки знаходять дуже широке використання, виготовляються на основі простих технологій, а тому мають низьку вартість, що, знову ж таки, підштовхує розробників цифрової апаратури до їх використання.

Але, поряд із стандартними елементами, у будь-якому цифровому пристрої існують і нестандартні модулі, без використання яких часто не можна обійтися, але їх виготовлення на стандартних елементах є або неможливим, або надто затратним, оскільки приводить до використання великої кількості корпусів, ускладнення монтажу, друкованих плат, зростанню часових затримок і т. п. Виготовлення спеціалізованих ІС на замовлення приводить до значних витрат коштів і часу на їх розробку та виготовлення. Саме ці проблеми і дали поштовх до створення інтегральних схем з програмованою і репрограмованою структурами, призначених для побудови нестандартних модулів розроблюваної електронної системи. Незважаючи на те, що такі структури можуть мати досить значну кількість логічних елементів, їх універсальність та висока серійність будуть більш вагомими аргументами в плані застосування.

Використання програмованих логічних інтегральних схем відкриває широкий простір для розробника електронної апаратури. На їх основі можливе створення функціонального обладнання різного ступеня складності. Наприклад, протягом останнього часу провідні фірми-виробники ПЛІС почали рекламувати

свої вироби як повноправну заміну класичним процесорам цифрової обробки сигналів.

9.1.1. ПЛМ (*програмовані логічні матриці*)

Програмовані логічні матриці (ПЛМ) (*PLA – Programmable Logic Array*) історично були першими з програмованих інтегральних схем. Прикладами ПЛМ можуть служити вітчизняні мікросхеми К556РТ1, К556РТ2, К556РТ21.

Узагальнена їх структура приведена на рис. 9.1.

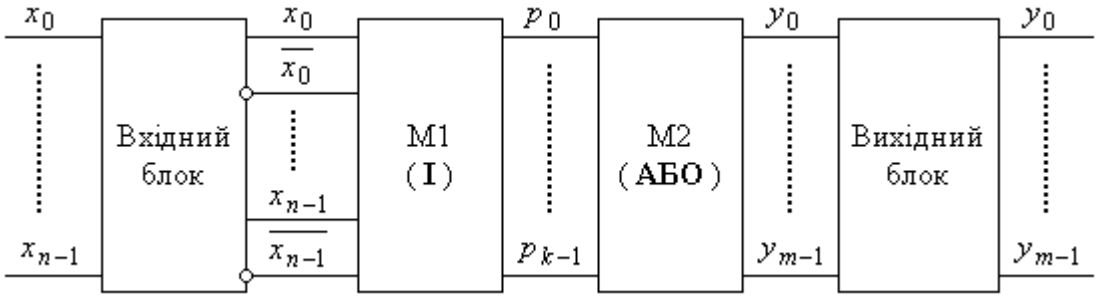


Рис. 9.1

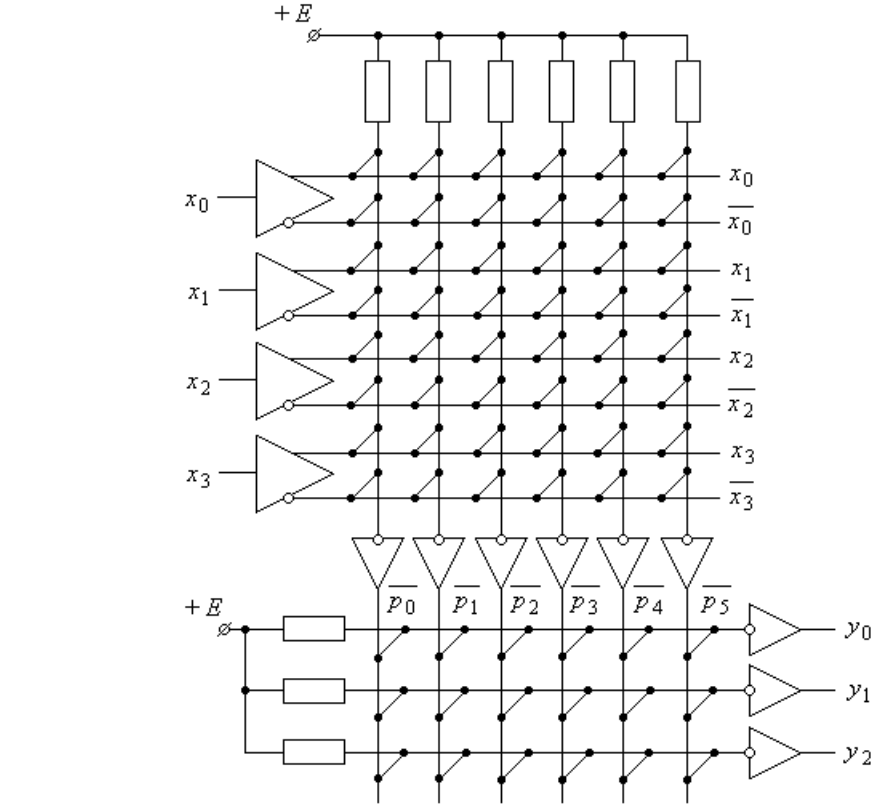
Матриця складається з послідовно з’єднаних вхідного блоку, призначеного для формування парафазних груп вхідних сигналів необхідної потужності; матриці M_1 багатовходових елементів **І**; матриці M_2 елементів **АБО** і вихідного блоку, який забезпечує необхідні характеристики вихідного сигналу (наявність *Z*-стану, відкритий колектор і т. п.).

Основними параметрами ПЛМ є кількість входів n, кількість мінтермів (виходів матриці M_1) k і кількість виходів m. Реально завжди виконується нерівність: $2^n >> k$, тому число мінтермів визначає складність логічних функцій, що реалізуються.

На рис. 9.2, *а* приводиться приклад принципової схеми з $n = 4$ і $m = 3$ для ТТЛ, а на рис. 9.2, *б, в* – схеми відповідних комутаційних вузлів, що складаються з діода і плавкої перемички, подібно до РПЗП.

При використанні КМОН-технології діоди замінюються польовими транзисторами. Схема матриці (рис. 9.2, *а*) залишається незмінною, а схеми комутаційних вузлів забезпечують реалізацію логічних операцій **І-НІ** та

АБО-НІ, що не впливає на кінцевий результат.



а)

б) *в)*

Рис. 9.2

У практиці проектування цифрових пристроїв на ПЛМ використовується спрощене зображення їх схемотехніки. Приклад такого зображення для $n = 4$; $k = 6$; $m = 3$ приводиться на рис. 9.3. Крапки, що встановлені на перетині рядків та стовпців матриці, характеризують відповідні з'єднання.

З приведеної на рис. 9.3 схеми матриці можемо записати функції, які реалізуються з її використанням:

$$y_0 = p_0 + p_1\,; \qquad y_1 = p_2 + p_3 + p_4\,; \qquad y_2 = p_0 + p_2 + p_5\,;$$

де $p_0 = x_0\ x_1$; $p_1 = \overline{x_0}\ \overline{x_1}\ \overline{x_2}\ \overline{x_3}$; $p_2 = x_0\ \overline{x_2}$;
$p_3 = \overline{x_0}\ x_2\ x_3$; $p_4 = x_1$; $p_5 = \overline{x_0}\ \overline{x_1}\ \overline{x_3}$.

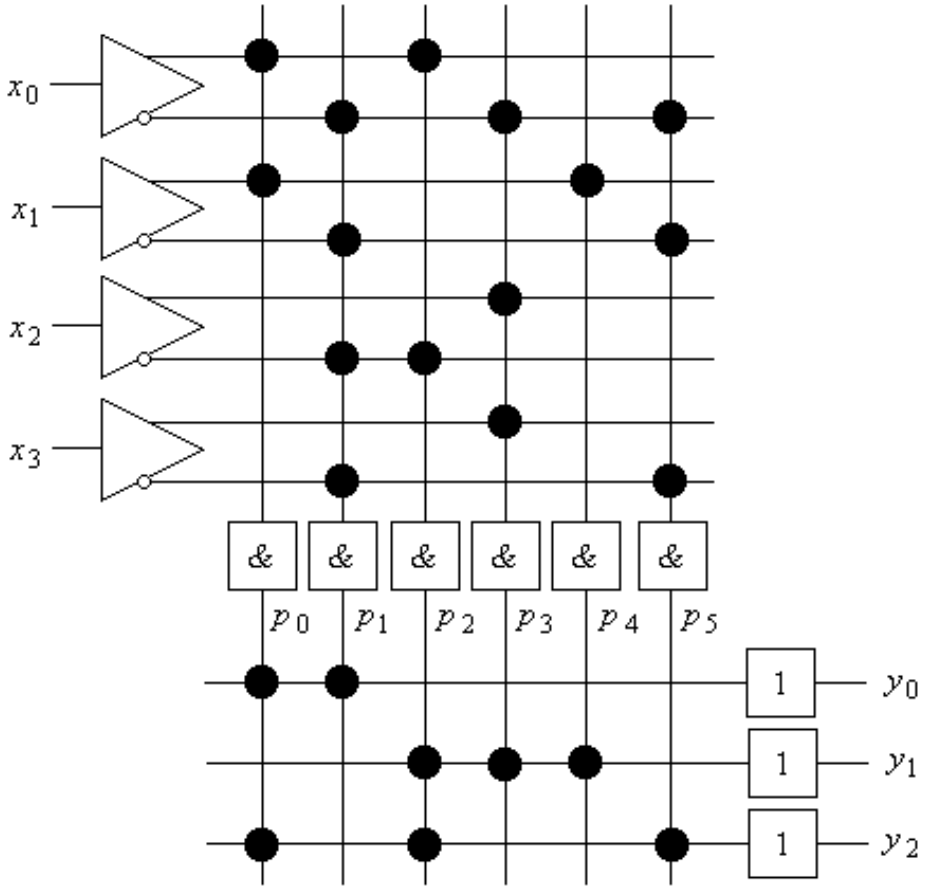


Рис. 9.3

Однією з особливостей, що відрізняють ПЛМ від ПЗП, є можливість використання свого роду зворотних зв'язків, тобто з'єднання виходів ПЛМ з її входами, що дає змогу реалізовувати більш складні логічні функціональні залежності. Приклад такої схеми приводиться на рис. 9.4, з якого маємо:

$$y_0 = x_0 \cdot x_1 + \overline{x_0} \cdot \overline{x_1};$$

$$y_1 = x_0 \cdot x_1 + y_0 \cdot x_2 = x_0 \cdot x_1 + x_2 \cdot \left(x_0 \cdot x_1 + \overline{x_0} \cdot \overline{x_1} \right).$$

Недоліком архітектури ПЛМ виявилося слабке використання ресурсів програмованої матриці **АБО**, тому подальший розвиток отримали мікросхеми, побудовані за архітектурою програмованої матричної логіки (ПМЛ).

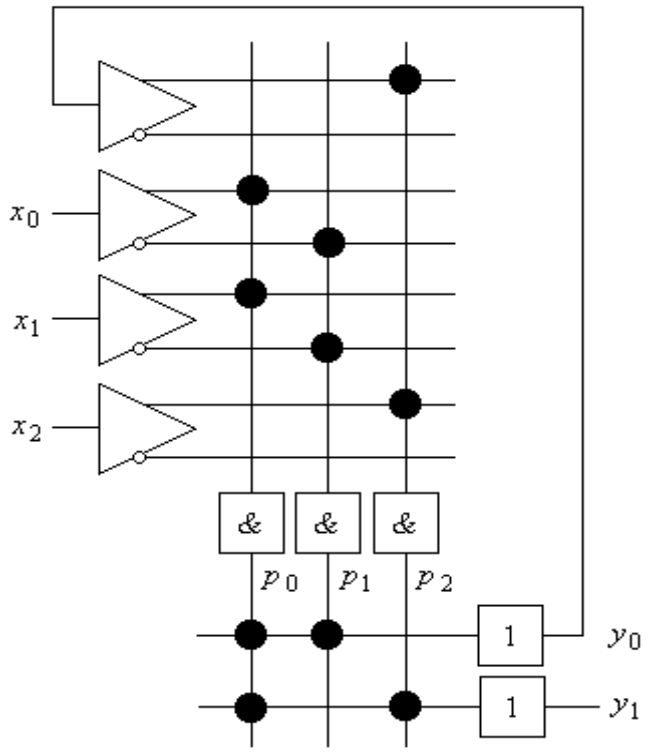


Рис. 9.4

9.1.2. ПМЛ (*програмована матрична логіка*)

Програмована матрична логіка (ПМЛ) (в англомовних літературних джерелах *PAL – Programmable Array Logic*) є спрощеним варіантом ПЛМ і характеризується тим, що має програмовану матрицю елементів **І** та фіксовану матрицю **АБО**. Поява цих пристроїв обумовлена тим, що в переважній більшості прикладних задач використовуються нескладні логічні функції, які можуть бути реалізовані на скінченній кількості елементів **АБО**. У ПМЛ відпадає необхідність програмування матриці **АБО**, внаслідок чого її використання суттєво спрощується.

Приклад організації ПМЛ представлений на рис. 9.5.

Прикладами ПМЛ такого типу є мікросхеми PAL16L8, PAL22V10 ТТЛ, а також GAL16V8-5 і PALCE16V8-5 КМОН-технологій. Серед мікросхем країн СНД подібними є мікросхеми серії КМ1556 (ХП4, ХП6, ХП8, ХЛ8). Різновидом цього класу є ПЛІС, які мають тільки одну програмовану матрицю **І**, –

наприклад, мікросхема 85C508 фірми Intel.

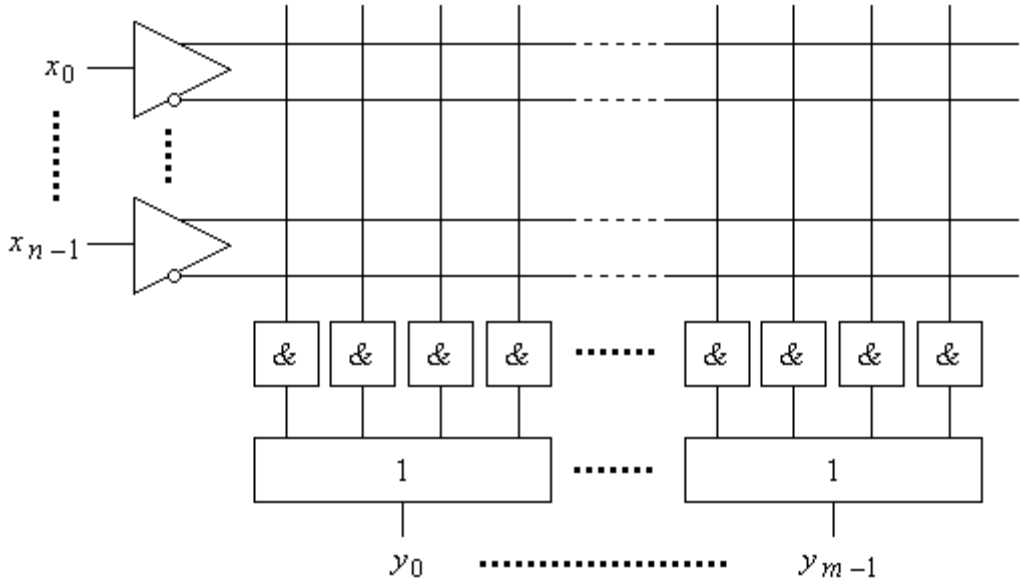


Рис. 9.5

Мікросхеми ПМЛ знайшли широке використання при реалізації нескладних логічних функцій і інтенсивно вдосконалювались в різних напрямках. Виникли ПМЛ з програмованим вихідним буфером, що дало можливість отримувати як прямі, так і інверсні значення функцій; мікросхеми з двонаправленими виводами, що можуть використовуватись як входи, так і як виходи; мікросхеми з елементами пам’яті на основі синхронних *D*-тригерів або цілих регістрів. Все це значно розширює області використання програмованої матричної логіки, відкрило можливості побудови синхронних цифрових автоматів.

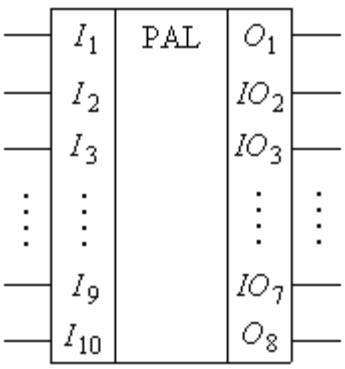


Рис. 9.6

На рис. 9.6 приводиться умовне зображення ПМЛ PAL16L8 – мікросхеми, що до останнього часу знаходила широке використання при побудові спеціалізованих цифрових пристроїв. Її програмована матриця **І** має 64 рядки і 32 стовпці і використовувані для програмування $64 \times 32 = 2048$ перепалювані перемички. Кожен з 64-х елементів матриці **І** має 32 входи, що охоплюють 16 вхідних змінних та їх заперечення; 10 з них приєднується до входів $I_1 \div I_{10}$, а решта 6 мають можливість приєднуватись до виводів $IO_2 \div IO_7$, що можуть програмуватись як входи, так і як виходи, а також мають *Z*-стан.

9.1.3. Мікросхеми програмованої макрологіки

Одним з напрямків побудови ПМЛ є ПЛІС, що мають лише одну програмовану матрицю **І** чи **І-НІ** (**АБО-НІ**), але за рахунок великої кількості інверсних зворотних зв'язків на їх основі можливо будувати складні логічні функції. До цього класу відносять мікросхеми PLHS501, PLHS502 фірми SIGNETICS, які побудовані на основі матриці **І-НІ**, а також мікросхему XL78C800 фірми EXEL, побудовану на основі матриці **АБО-НІ**.

Оскільки ПЛІС за рівнем інтеграції характеризуються кількістю елементів **2І-НІ**, то приведені вище мікросхеми характеризуються як ПЛІС низького рівня інтеграції з середньою кількістю ЛЕ до 1500…2000.

У цілому вони морально застарілі не тільки з точки зору їх технологій виготовлення, а й з точки зору програмування.

9.1.4. БМК (*базові матричні кристали*)

Ще одним напрямком розвитку ПЛІС є базові матричні кристали (БМК), призначені для реалізації нестандартних пристроїв ЕОМ без використання інтегральних схем низького і середнього рівня інтеграції, зорієнтовані на великі інтегральні схеми (ВІС) (в англомовній літературі для позначення ВІС такого призначення використовується термін *GA* (*Gate Array* – вентильна матриця).

Причини, що спонукали до створення ВІС такого призначення, полягають у наступному. Витрати на розробку ВІС, а також надвеликих ІС (НВІС) (у російській термінології – БИС і СБИС), дуже великі і визначаються сотнями мільйонів доларів, тому розробляти спеціалізовані ВІС нераціонально. Розв'язання проблеми знайшли в тому, що були розроблені НВІС, функціонування яких налаштовувалось під конкретні задачі лише на заключних стадіях їх виготовлення. Все це в цілому дозволило значно знизити їх вартість, час на проектування та виготовлення і дало можливість використовувати ВІС та НВІС для побудови унікальних або спеціалізованих пристроїв.

Фірмами-виробниками країн СНД виготовляються БМК з кількістю еквівалентних ЛЕ до 50 тисяч, з затримкою кожного з них 20 нс.

9.2. Сучасні ПЛІС

Розглянуті у § 9.1 мікросхеми ПЛІС послужили основою для розвитку двох сучасних напрямків ВІС і НВІС з програмованими і репрограмованими структурами. Перший продовжує напрямок розвитку програмованої матричної логіки в серіях *CPLD* (*Complex Programmable Logic Devices*), а другий напрямок розвиває БМК в серіях *FPGA* (*Field Programmable Gate Array*). Альтернативою обох напрямків стала змішана архітектура НВІС з назвою *FLEX* (*Flexible Logic Element matriX*). Зростання рівня інтеграції дало можливість розміщувати на одному кристалі схеми, складність яких відповідає цілим електронним системам.

Сучасний рівень інтеграції НВІС настільки високий, що на кристалі розміщуються мільйони логічних елементів. Реальне та функціональне знання схемотехніки НВІС для розробників цифрової апаратури, особливо фахівців невисокої кваліфікації, не має суттєвого значення. Тому розглянемо лише ті особливості, які можуть дати корисну інформацію.

Програмованість НВІС забезпечується наявністю в ній великої кількості двополюсників (ключів), які за допомогою програмних засобів можуть

переводитись користувачем в стан високої або низької (нульової) провідності. Тим самим задається необхідна конфігурація схеми, що формується на кристалі. Кількість програмованих ключів для сучасних ПЛІС досягає декількох мільйонів.

У сучасних ПЛІС використовуються такі типи програмованих ключів:

- перемички типу *antifuse* (тонкі діелектричні пробивані перемички);
- ЛІЗМОН-транзистори з подвійним затвором;
- ключові польові транзистори, керовані тригерами пам'яті конфігурації.

Програмування з перемичками типу *antifuse* є одноразовим. Перемички мають малі розміри і в початковому стані пропускають дуже малі струми ($\approx 10^{-15}$ А). Програмуючий імпульс напруги пробиває перемичку і створює провідний канал. Опір каналу досить точно залежить від величини пробивного струму, що дає можливість створювати перемички з заданою величиною опору. Параметри обох станів перемички можуть зберігатись близько 40 років.

Елементи *EEPROM* (*FLASH*) на ЛІЗМОН були розглянуті раніше.

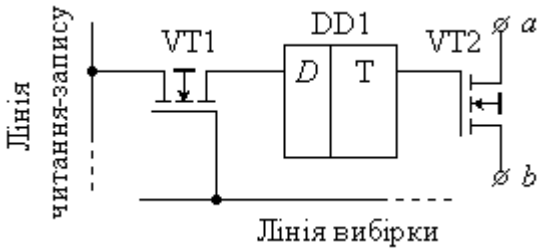


Рис. 9.7

Приклад схеми ключа, керованого тригером пам'яті конфігурації, приводиться на рис. 9.7.

Ключ VT2 забезпечує комутацію ділянки *ab* в залежності від стану тригера.

Тригер, у свою чергу, встановлюється високим потенціалом на лінії вибірки і відповідним сигналом (**1** або **0**) на лінії читання-запису.

Застосування такого ключа відкриває широкі можливості використання ПЛІС, оскільки час їх запису визначається десятками-сотнями мілісекунд, а витрати часу на стирання взагалі непотрібні.

ВІС, в яких схема створюється з використанням логічних елементів і програмованих ключів, називаються *ВІС* (*НВІС*) *програмованої логіки*.

Дещо іншу архітектуру мають програмовані користувачем вентильні

матриці (*FPGA*).

Вони побудовані на основі спеціалізованих блоків ідентичної конфігурації, в якості яких використовуються:

- прості логічні елементи;
- логічні модулі на основі мультиплексів;
- логічні модулі на основі ПЗП;

а також спеціалізовані блоки типу блока вводу-виводу і т. п. Блоки мають можливість об’єднуватись між собою, створюючи необхідну конфігурацію схеми.

Існує досить велика кількість архітектур *FPGA*, які мало чим можуть бути корисні для схемотехніків цифрових пристроїв.

Складні програмовані логічні схеми (*CPLD* і *FLEX*) архітектурно містять в собі центральну комутаційну матрицю та велику кількість функціональних блоків (макроелементів), периферійних блоків. Їх архітектура і схемотехніка досить складна і не має суттєвого значення для розробників цифрової апаратури. Класичними представниками *CPLD* є мікросхеми MAX7000 фірми ALTERA. На ринку НВІС представлена велика кількість *CPLD* і НВІС ПЛМ фірм ALTERA (сім’я MAX, FLEX, APEX), ATMEL (сім’я ATF1500), VANTIS (сім’я MACH), XILINX (сім’я XC9500) та ін. У рамках кожного сім’ї є ряд НВІС різних за складністю, вартістю та іншими показниками. У виборі необхідних мікросхем допомагають програмні засоби, призначені для їх програмування.

Наступним рівнем розвитку ПЛІС стали НВІС, які визначаються як *“система на кристалі”* (в англомовній літературі *SoC – System on Crystal*). Поява таких НВІС обумовлена зростанням рівня інтеграції та швидкодії, внаслідок чого з’явилась можливість створювати на кристалі спеціалізовані обчислювальні модулі та цілі процесори, оперативну та постійну пам’ять, периферійні пристрої, а також спеціалізовані пристрої тестування. Зрозуміло, такі ПЛІС мають велику кількість спеціалізованих блоків, але, у той же час, на

базі таких ПЛІС виникає можливість будувати спеціалізовані цифрові системи досить високої потужності. До таких ПЛІС відноситься ряд НВІС типу APEX20K/KE фірми ALTERA, а також сім'я НВІС типу VIRTEX фірми XILINX.

Суттєвою відмінністю ПЛІС останнього покоління є суттєве приближення їх до користувача з точку зору програмування та тестування. Для цього ПЛІС забезпечуються сукупністю засобів та операцій, які дозволяють забезпечити тестування ВНІС без фізичного доступу до кожного з її виводів. Комплекс вмонтованих апаратних і програмних засобів, призначених для розв'язання задач тестового контролю, називається *інтерфейсом JTAG* (*Joint Test Action Group* – група, що розробила стандарт тестування). Тестування за *JTAG*-стандартом має назву *"периферійного сканування"* (*Boundary Scan Testing* (*BST*)). Периферійне сканування дозволяє перевірити роботу мікросхем, монтажні з'єднання мікросхем між собою на друкованій платі, зчитувати сигнали на виводах мікросхеми під час її роботи або керувати цими сигналами.

Важливою властивістю ПЛІС останнього покоління є можливість забезпечення реконфігурації (програмування) в системі, що надає можливість виконувати зміни в логіці її роботи. Властивість програмованості безпосередньо в системі позначається скороченням *ISP* (*In-System Programmable*). Необхідність в використанні цієї властивості виникає при потребі в оновленні системи або при виявленні помилок в роботі.

9.3. Основні параметри ПЛІС

При виборі фірми-виробника ПЛІС, сім'ї і типу розробник цифрових систем повинен користуватися такими критеріями:

- логічна ємність, достатня для реалізації проекту, з можливістю майбутнього оновлення;
- швидкодія ПЛІС;

- схемотехнічні та конструктивні параметри;
- наявність або вартість засобів розробки, що включає в себе апаратні та програмні засоби;
- наявність технічної та методичної підтримки;
- вартість мікросхем та надійність каналів їх придбання.

Виходячи з цих критеріїв, для фахівців, які не мають достатнього досвіду роботи з ПЛІС, рекомендуємо розпочинати роботу з мікросхемами фірми ALTERA (www.altera.com). Ця фірма виготовляє широкий ряд ПЛІС *CPLD* сімейств MAX3000, MAX7000, MAX9000, а також ПЛІС *FPGA* сімейств FLEX10K, FLEX6000, FLEX8000. Важливою перевагою є те, що фірма ALTERA забезпечує користувачів безкоштовною версією програмного забезпечення – базовий пакет MAX+plus II BASELINE можна отримати на CD *"ALTERA Digital Library"* або на офіційному Інтернет-сайті компанії.

ПЛІС фірми ALTERA виготовлені з можливістю програмування безпосередньо на платі. Для програмування та завантаження конфігурації пристрою опубліковані схеми кабелю для завантаження – *ByteBlaster* та *ByteBlasterMV*.

Для пояснення параметрів ПЛІС, з якими користувачу доводиться мати справу при проектуванні цифрових пристроїв, розглянемо особливості функціональної схеми широко використовуваних ІС ряду MAX3000 (рис. 9.8).

Основними елементами функціональної схеми є:

- логічні блоки (ЛБ) (*Logic Array Blocks* (*LAB*));
- макроелементи МЕ (*Macrocells*);
- логічні розширювачі (*Expanders*): паралельний (*Parallel*) та розподілюваний (*Shareble*);
- програмована матриця з'єднань (ПМЗ) (*Programmable Interconnected Array* (*PIA*));
- елементи вводу-виводу (ЕВВ) (*I/O Control Block*).

Велика складність матриці обумовлює виділення групи електричних кіл,

які охоплюють всі схеми. Такі електричні кола називаються *глобальними*. За глобальними колами закріплюється частина виводів (*Dedicated Inputs*). Це глобальні кола синхронізації, обнуління, установки в *Z*-стан кожного макроелементу. Разом з цим, ці виводи можуть бути використані як входи або виходи користувача для “швидких “ сигналів, що обробляються ПЛІС.

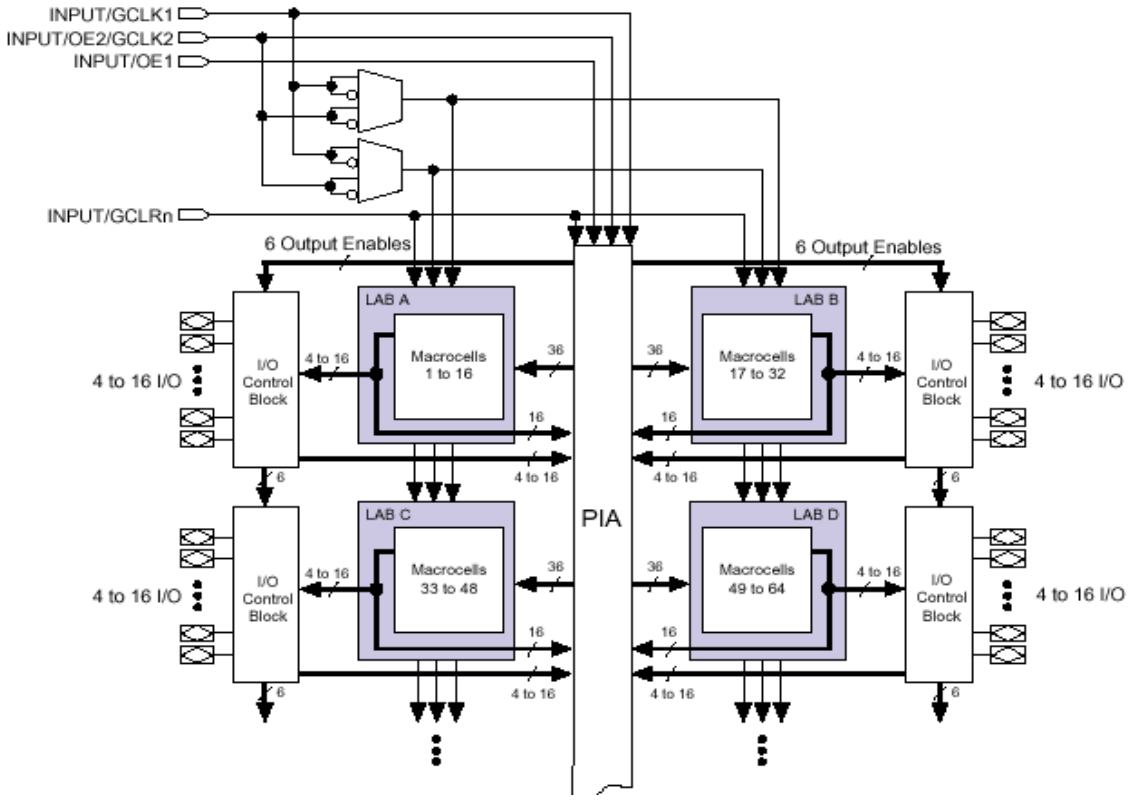


Рис. 9.8

Як видно з рис. 9.8, в основу архітектури ПЛІС сім’ї, що розглядається, покладені логічні блоки, які містять у собі по 16 макроелементів кожний. Логічні блоки з’єднуються за допомогою програмованої матриці з’єднань. Кожен логічний блок має 36 входів з ПМЗ.

На рис. 9.9 приводиться структурна схема макроелементу ПЛІС сім’ї MAX3000. Вона складається з трьох головних вузлів:

- локальної програмованої матриці (ЛПМ) (*LAB Local Array*);
- матриці розподілення термів (МРТ) (*Product-Term Select Matrix*);
- програмованого регістра (ПР) (*Programmable Register*).

Комбінаційні функції реалізуються на локальній програмованій матриці і

матриці розподілення термів, що дозволяє об'єднувати добутки по **АБО** (**OR**) або по **ВИКЛ. АБО** (**XOR**). Крім цього, матриця розподілення термів дозволяє забезпечити комутацію кола керування тригером макроелементу.

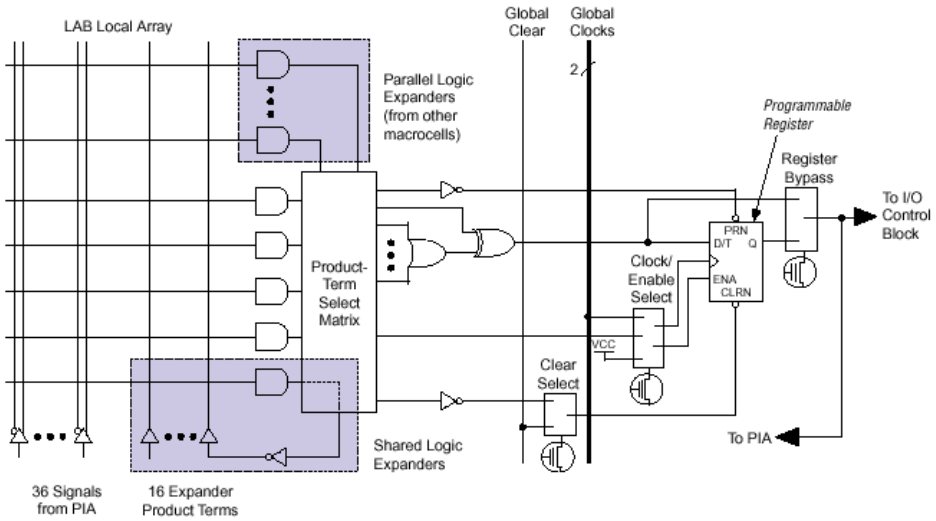


Рис. 9.9

У ПЛІС сім'ї MAX3000 використані 2 глобальні тактові сигнали, що дозволяє проектувати схеми з двофазною синхронізацією.

З точки зору ємності ПЛІС, вони характеризуються такими параметрами:

- логічною ємністю, тобто кількістю еквівалентних логічних елементів типу **2І-НІ**;
- кількістю макроелементів;
- кількістю логічних блоків;
- кількістю програмованих користувачем виводів.

У табл. 9.1 приводяться довідкові дані по ПЛІС фірми ALTERA різних сімейств.

Таблиця 9.1

Сім'я / Параметр	MAX3000 EPM3256A	FLEX600 EPF6024A	MAX9000 EPM9560	FLEX10K EPF10K250	APEX20K EP20K1000
Логічна ємність	5000	24000	5000	250000	2670000
Кількість макроелементів	256	1960	560	12160	4224
Кількість логічних блоків	16	196	772	20	264

Кількість виводів для користування	158	45	46	470	780

Для реалізації логічних функцій з великою кількістю змінних використовуються логічні розширювачі (див. рис. 9.10 - 9.11).

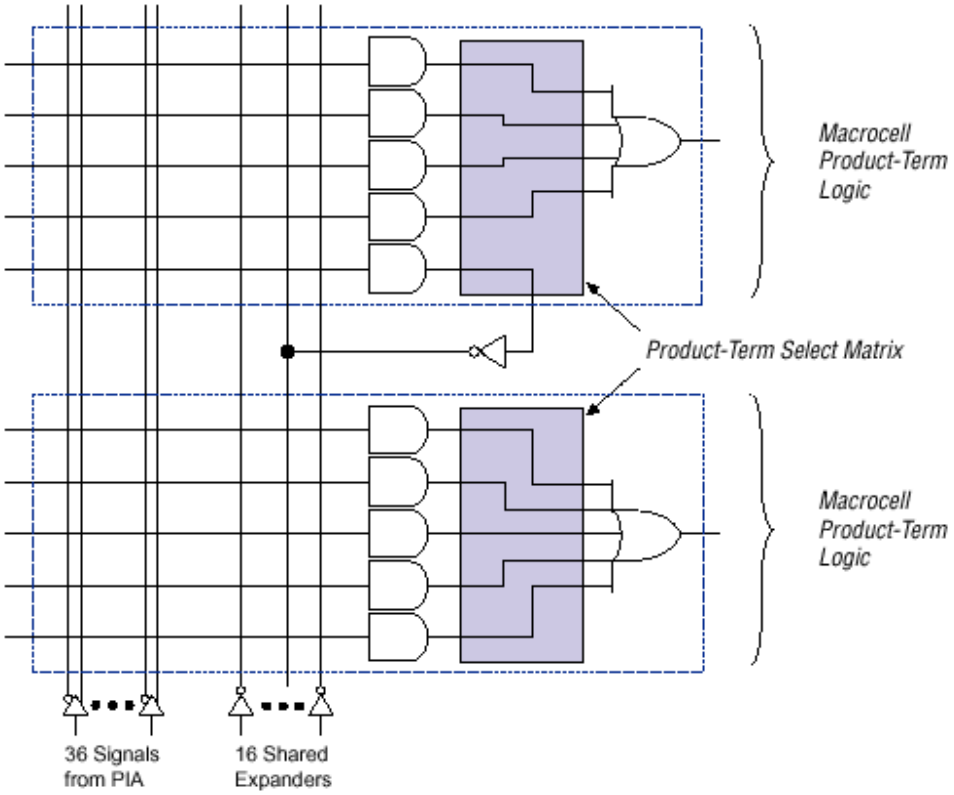


Рис. 9.10

Розподілюваний логічний розширювач (рис. 9.10) дозволяє реалізовувати логічні функції з великою кількістю входів, забезпечуючи можливості об’єднувати МЕ, що входять до складу одного ЛБ. У такий спосіб розподілюваний розширювач формує терм, інверсне значення якого передається матрицею розподілу термів в локальну програмовану матрицю і може бути використано у будь-якому МЕ даного ЛБ.

Як видно з рис. 9.10, наявні 36 сигналів ПМЗ, а також 16 інверсних сигналів з розподілюваних логічних розширювачів, дозволяють у межах одного ЛБ реалізувати логічну функцію до 52 термів рангу 1.

Паралельний логічний розширювач (рис. 9.11) дозволяє використовувати локальні матриці суміжних МЕ для реалізації функцій, в які входять більше

5 термів.

Одна ланка паралельних розширювачів може містити до 4 МЕ, реалізуючи функцію 20 термів.

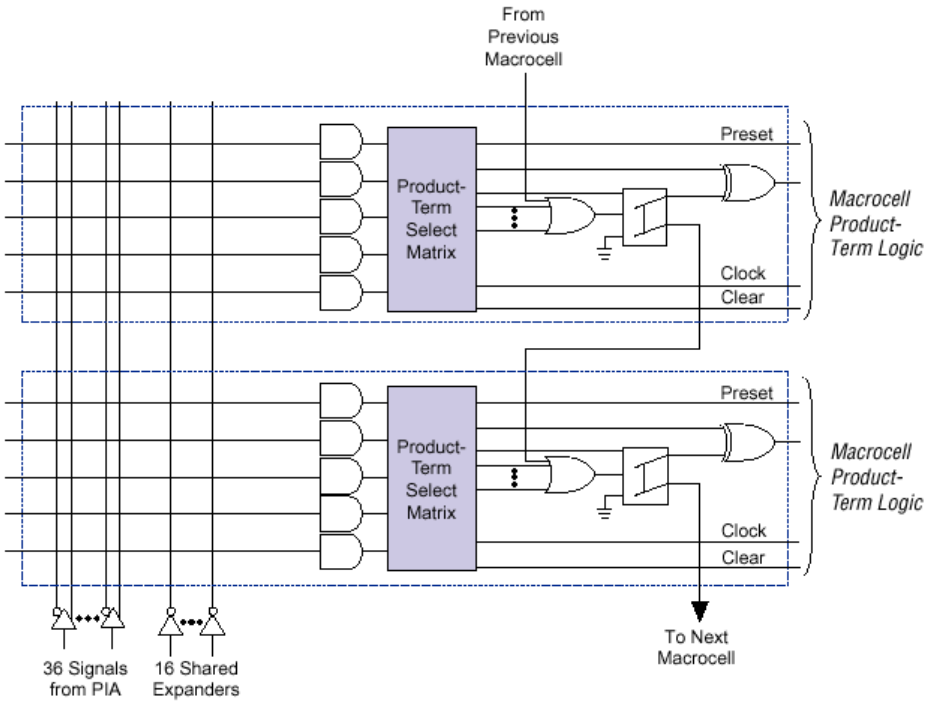


Рис. 9.11

Програмована матриця з’єднань (рис. 9.12) побудована так, що на неї виводяться сигнали від всіх можливих джерел – ЕВВ, сигналів зворотних зв’язків ЛБ, спеціалізованих виділених виводів. При програмуванні забезпечуються тільки необхідні зв’язки для передачі сигналів.

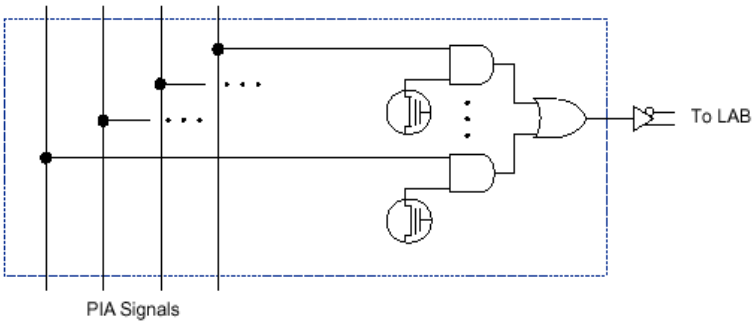


Рис. 9.12

Елементи вводу/виводу ПЛІС (рис. 9.13) дозволяють забезпечити режими роботи на ввід/вивід інформації, а також режими з відкритим колектором та

Z-станом.

Мікросхеми сім'ї MAX3000 повністю підтримують стандарт *JTAG*.

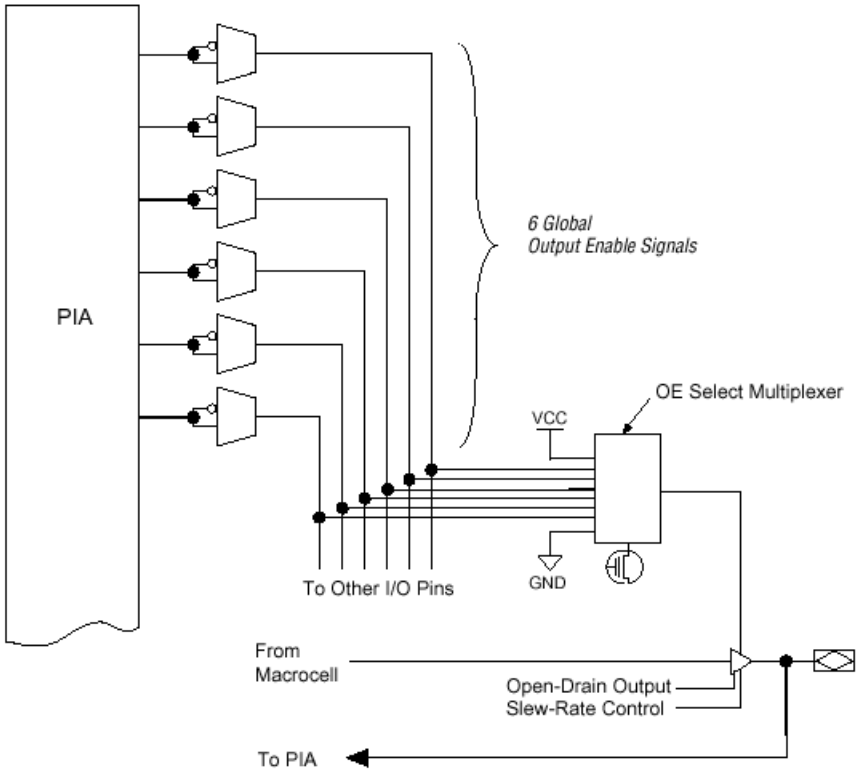


Рис. 9.13

Програмування забезпечується підвищеною, порівняно з живленням (3,3 В), напругою, яка створюється спеціалізованими схемами помножувачів напруги, що входять до складу ПЛІС.

Послідовність програмування, режим тактування тригерів, конфігурація окремих модулів повністю забезпечуються автоматично під час синтезу проекту в САПР MAX+plus II.

Наступною групою параметрів ПЛІС є часові параметри програмування. Але, оскільки ПЛІС програмуються за допомогою САПР, то їх компілятори повністю забезпечують необхідні часові співвідношення між сигналами і користувачу їх знання необов'язкове.

Для користувача важливими є затримки в розповсюдженні сигналів. До них відносять:

- затримки розповсюдження сигналу вхід/вихід;

- затримка глобального тактового сигналу до виходу;
- час установлення глобального тактового сигналу;
- максимальна глобальна тактова частота.

Для ПЛІС сім'ї MAX3000 часові затримки знаходяться у межах 3…5 нс, а максимальна тактова частота досягає 200 МГц.

ПЛІС, що розглядаються, мають передбачувану величину затримки у розповсюджені сигналу, тому при моделюванні в САПР MAX+plus II результати часового моделювання відповідатимуть реальним часовим затримкам у мікросхемі. Тому за результатами моделювання можна отримати більш детальні дані величин часових затримок сигналу між заданими входами та виходами мікросхеми.

9.4. Основи проектування цифрових пристроїв на ПЛІС в САПР MAX+plus II

9.4.1. Загальний опис САПР MAX+plus II

Система автоматизованого проектування MAX+plus II призначена для проектування електронних система середнього рівня складності на ПЛІС *ALTERA*. Програмне забезпечення системи MAX+plus II представляє собою єдине ціле, забезпечує керування користувачем середовищем проектування і допомагає досягти максимальної ефективності та швидкості в розробці електронних систем.

Під час інсталяції системи MAX+plus II створюються два каталоги: *maxplus*2 та *max*2*work*. Каталог *maxplus*2 містить системне програмне забезпечення та файли даних. Каталог *max*2*work* містить файли навчальної програми та приклади. Файли проектів бажано створювати саме в цьому каталозі, оскільки це дозволить запобігти плутанини і дозволить швидше відшукувати потрібні файли.

Система MAX+plus II пропонує повний спектр можливостей логічного

дизайну: різні засоби опису проекту з ієрархічною структурою, потужний логічний синтез, компіляцію з заданими часовими параметрами, розділення на частини, функціональне та часове тестування (симуляцію), тестування декількох пов'язаних пристроїв, автоматичну локалізацію помилок, а також програмування ПЛІС. У системі MAX+plus II можна як читати, так і записувати файли на мові AHDL, файли на мовах опису апаратури Verilog HDL та VHDL, а також схемні файли OrCAD. Є можливість опису проекту у вигляді файлу на мові опису апаратури, створеному у зовнішньому редакторі або у текстовому редакторі MAX+plus II, у вигляді електричної схеми, створеної у графічному редакторі *Graphic Editor*, у вигляді часової діаграми, створеної в редакторі *Waveform Editor*. Для роботи зі складними ієрархічними проектами з кожної частини схеми може бути створений символ, редагування якого виконується у графічному редакторі. Під *проектом* розуміється сукупність ієрархічно зв'язаних файлів проекту. Такі команди, як відкриття файлів, ввід призначень пристроїв, виводів та логічних елементів, компіляція проекту, схожі для багатьох редакторів системи. Редактори для розробки проекту дозволяють виконувати схожі задачі (наприклад, пошук символу) схожими засобами. Можна легко комбінувати різні типи файлів проекту в ієрархічному проекті, обираючи для кожного функціонального блоку той формат, який найбільше підходить.

Усі пакети працюють як на платформі IBM PC, так і на платформах SUN, IBM RISC/6000 і HP9000. Надалі розглядатимемо роботу на платформі IBM PC.

Для нормальної інсталяції і роботи САПР MAX+plus II (версія 9.4 вийшла в грудні 1999 року) необхідна IBM PC-сумісна ЕОМ із процесором не гірше Pentium, обсягом ОЗП не менше 16 МБ і вільним місцем на жорсткому диску порядку 150 – 400 МБ в залежності від конфігурації системи. Для розробки великих кристалів на ПЛІС FLEX10 і вище бажано мати не менш 64 МБ ОЗП (краще 128 МБ і вище) і процесор Pentium II. Звичайно, можна використовувати і більш слабкі машини, але тоді зростає час компіляції і збільшується

навантаження на жорсткий диск через свопінг. Збільшення обсягу оперативної пам'яті і КЕШ-пам'яті дає кращі результати в порівнянні зі збільшенням тактової частоти процесора. Якщо не передбачається трасування великих кристалів, то цілком вистачає 32 МБ ОЗП при нормальній швидкості компіляції проекту. Що стосується вибору операційної системи, то, без сумніву, краще використовувати Windows NT, Windows 98.

Назва системи MAX+plus II є абревіатурою від ***M**ultiple **A**rray matri**X** **P**rogrammable **L**ogic **U**ser **S**ystem* (користувацька система програмування логіки упорядкованих структур). Система MAX+plus II легко пристосовується до конкретних вимог користувача. Вона має засоби зручного ведення проекту, швидкого аналізу як логіки, так і часових параметрів, безпосереднього програмування пристроїв.

Процедуру розробки нового проекту можна представити наступним чином:

- створюється новий файл проекту або ієрархічна структура файлів проекту з використанням редакторів розробки проекту в системі MAX+plus II;
- задається ім'я файлу проекту в якості імені проекту;
- призначається сім'я ПЛІС для реалізації проекту. Користувач може сам призначити конкретний пристрій або запропонувати це компілятору для оцінки ресурсів; відкриваються вікна компілятора (*Compiler*) і виконується його запуск кнопки*Start* для початку компіляції проекту;
- у разі вдалої компіляції проводиться тестування і часовий аналіз;
- програмування ПЛІС, що виконується шляхом запуску модуля програматора (*Programmer*) з послідуючим розташуванням пристрою в програмуючому адаптері програматора.

9.4.2. Початок роботи з САПР MAX+plus II

Для початку роботи з системою MAX+plus II після установки на комп'ютер слід з меню **Пуск** (**Start**) вибрати підменю **Програми** (**Programs**), далі вибрати підменю **MAX+plus II 9.23 Baseline** і запустити файл

MAX+plus II 9.23 Baseline (рис. 9.14).

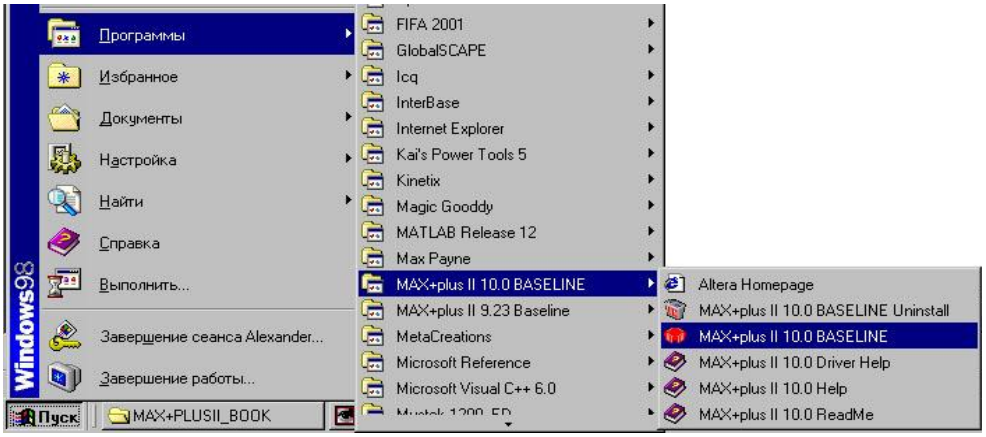


Рис. 9.14

Після запуску системи MAX+plus II автоматично відчиняється головне меню, яке охоплює всі її прикладні програми. У верхній частині вікна відображені назва проекту та ім'я поточного файлу проекту. Потім йде рядок меню, під ним панель основних інструментів системи, яка забезпечує швидкий виклик її компонентів. Виклик компонентів системи зручно проводити через вікно меню MAX+plus II, представленого на рис. 9.15.

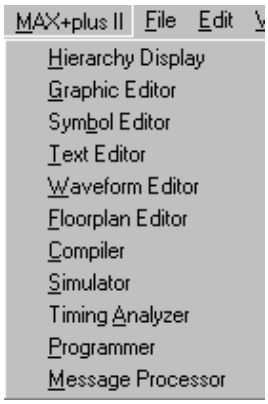


Рис. 9.15

У Табл. 9.2 приведено вигляд кнопок, функції яких відповідають пунктам меню MAX+plus II, а також описується їх призначення.

Таблиця 9.2

Додаток	Функція
Hierarchy Display 	**Огляд ієрархії** – відображає поточну ієрархічну структуру файлів у вигляді дерева з гілками, що представляють собою окремі файли проекту. Можна візуально визначити, чи є файл проекту схемним, текстовим чи сигнальним, які файли відкриті в даний момент, які допоміжні файли в проекті доступні користувачу для редагування. Можна також безпосередньо відкрити чи закрити один або кілька файлів дерева і ввести призначення ресурсів для них.
Graphic Editor 	**Графічний редактор** – дозволяє розробляти схемний логічний дизайн у форматі реального відображення на екрані WYSIWYG. Застосовуючи розроблені фірмою ALTERA примітиви, мегафункції і макрофункції як основні блоки розробки. Користувач може також використовувати власні символи.

Symbol Editor 	**Символьний редактор** – дозволяє редагувати існуючі символи і створювати нові.
Text Editor 	**Текстовий редактор** – дозволяє створювати і редагувати текстові файли проекту, написані на мовах опису апаратури AHDL, VHDL і Verilog HDL. Крім того, у цьому редакторі можна створювати, переглядати і редагувати інші файли формату ASCII, які використовуються іншими редакторами системи MAX+plus II. Можна створювати файли на мовах HDL і в інших текстових редакторах, однак даний текстовий редактор системи MAX+plus II дає переваги у виді контекстної довідки, виділення кольором синтаксичних конструкцій і готових шаблонів мов AHDL, VHDL і Verilog HDL.
Waveform Editor 	**Сигнальний редактор** – виконує подвійну функцію: інструмент для розробки дизайну та інструмент для введення тестових сигналів і спостереження результатів тестування.
Floorplan Editor 	**Порівневий планувальник** – дозволяє графічними засобами робити призначення виводам пристрою і ресурсів логічних елементів і блоків. Можна редагувати розташування виводів на кресленні корпуса пристрою і призначати сигнали окремим логічним елементам на більш докладній схемі логічної структури (LAB-View). Можна також переглядати результати останньої компіляції.
Compiler 	**Компілятор** – обробляє логічні проекти, розроблені для сімейств пристроїв Altera Classic, MAX 5000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000 і FLEX 10K. Більшість завдань виконується автоматично. Однак користувач може керувати процесом компіляції повністю чи частково.
Simulator 	**Симулятор** – дозволяє тестувати логічні операції і внутрішню синхронізацію спроектованої логічної схеми. Можливі три режими тестування: функціональне, часове і тестування декількох з'єднаних між собою пристроїв.
Timing Analyzer 	**Аналізатор часових параметрів** – аналізує роботу проектованого логічного пристрою після того, як він був синтезований і оптимізований компілятором. Дозволяє оцінити затримки, що виникають у схемі.
Programmer 	**Програматор** – дозволяє програмувати, конфігурувати, проводити верифікацію і випробувати пристрої фірми ALTERA.
Message Processor 	**Генератор повідомлень** – видає на екран повідомлення про помилки, що попереджають інформаційні повідомлення про стан проекту користувача і дозволяє користувачу автоматично знайти джерело повідомлення у вихідному чи допоміжному файлі (файлах) і в порівневому планувальнику.

Програмне забезпечення системи містить одинадцять прикладних програм та головну керуючу оболонку. Різні додаткові програми, що створюють файли проекту, можуть бути активізовані шляхом вибору відповідного пункту меню та натисканням лівої кнопки миші. *Файл проекту* – це графічний, текстовий або сигнальний файл, створений за допомогою відповідного редактора системи MAX+plus II. Проект складається з усіх файлів ієрархічної структури дизайну. Тому кожний новий проект бажано починати в новому підкаталозі з відповідною назвою. Схемні файли проекту створюються в графічному редакторі. Можна також відкрити, редагувати і зберігати схеми, створені схемним редактором OrCAD.

Для запуску графічного редактора слід вибрати пункт *Graphic Editor* з меню MAX+plus II або натиснути відповідну кнопку на панелі інструментів. Слід відмітити, що основна робота з розробки схем буде проводитись у графічному редакторі.

Проекти на мовах AHDL, VHDL та Verilog HDL створюється у текстовому редакторі *MAX+plus II Text Editor*, або в будь якому іншому. Сигнальні проекти створюються в сигнальному редакторі *MAX+plus II Waveform Editor*. Запуск будь-якого з цих редакторів виконується шляхом вибору відповідного пункту меню з назвою редактора та натисканням лівої кнопки миші.

9.4.3. Реалізація простих логічних пристроїв на базі ПЛІС

Зберемо схему, що зображена на рис. 9.16.

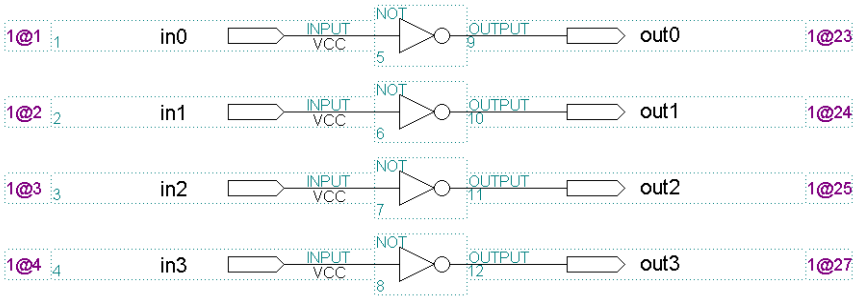


Рис. 9.16

Запускаємо систему MAX+plus II. В її меню вибираємо *Graphic Editor* або з меню *File* команду *New* і у вікні, що з'явилось (рис. 9.17), вибираємо *Graphic Editor*. Задаємо ім'я головного файлу проекту та записуємо його на жорсткий диск. Для цього з меню *File >> Project* вибираємо пункт *Name* (рис. 9.17).

На рис. 9.18 показане діалогове вікно, що з'явиться.

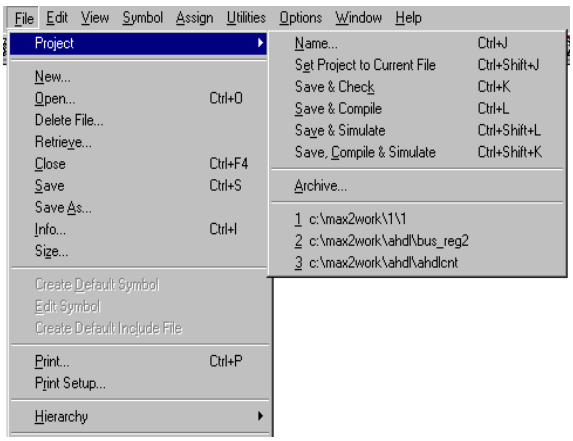


Рис. 9.17

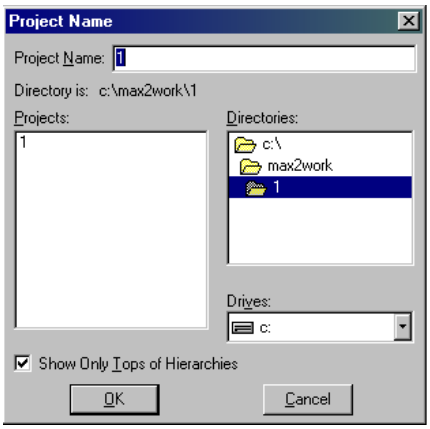


Рис. 9.18

Вводимо ім'я файлу та вказуємо каталог, в якому будуть розміщені файли проекту. Натискаємо кнопку ОК. В полі *Project Name* вводиться назва головного файлу проекту. У меню *Drives* вибирається логічний диск, а у меню *Directories* – каталог, в якому будуть міститися файли проекту.

Перед тим, як почати працювати в системі MAX+plus II, варто зрозуміти різницю між *файлами проекту* (*design-file*), *допоміжними файлами* і *проектами*.

Файл проекту – це графічний, текстовий чи сигнальний файл, створений за допомогою графічного, текстового чи сигнального редакторів системи MAX+plus II. Можна створювати ці файли і в інших редакторах, які використовують промислові стандарти схемного чи текстового редактора, або за допомогою програми *Netlist writer*, що є в пакетах, які підтримують EDIF, VHDL і Verilog HDL. Цей файл містить логіку для проекту MAX+plus II і обробляється компілятором.

Компілятор може автоматично обробляти наступні файли проекту:

- графічні файли проекту (**.gdf**);
- текстові файли проекту мовою AHDL (**.tdf**);
- сигнальні файли проекту (**.wdf**);
- файли проекту мовою VHDL (**.vhd**);
- файли проекту мовою Verilog (**.v**);
- схемні файли OrCAD (**.sch**);
- вхідні файли EDIF (**.edf**);
- файли формату Xilinx Netlist (**.xnf**);
- файли проекту Altera (**.adf**);
- файли цифрового автомата (**.smf**).

Допоміжні файли – це файли, що пов'язані з проектом MAX+plus II, але не є частиною ієрархічного дерева проекту. Більшість таких файлів не містить логіки дизайну. Деякі з них створюються автоматично додатком системи MAX+plus II, інші – користувачем. Прикладами допоміжних файлів є файли призначень і конфігурації (**.acf**), символьні файли (**.sym**), файли звіту (**.rpt**) і файли тестових векторів (**.vec**).

Проект складається з усіх файлів ієрархічної структури дизайну, в тому числі допоміжних і вихідних файлів. *Ім'ям проекту* є ім'я файлу верхнього рівня без розширення. Система MAX+plus II виконує компіляцію, тестування, часовий аналіз і програмування відразу цілого проекту, хоча користувач може в цей час редагувати файли цього проекту в рамках іншого проекту. Наприклад, під час компіляції проекту project1 користувач може редагувати дизайн його файлу TDF, що є також файлом проекту *project*2 і зберігати його; однак якщо він захоче компілювати його, потрібно буде спочатку задати ім'я *project*2 як ім'я проекту.

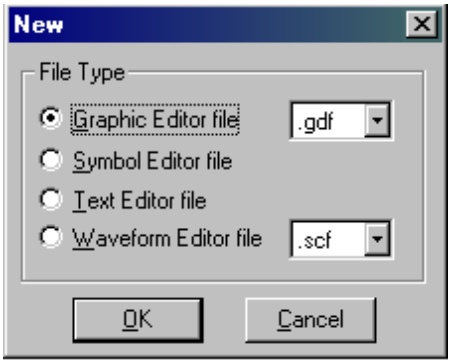


Рис. 9.19

При виборі пункту *New* з'являється вікно (рис. 9.27), в якому пропонується вибрати редактор, в якому буде створений головний файл проекту.

При виборі одного з пунктів і натисканні кнопки OK запускається відповідний редактор. Слід вибрати *Graphic Editor File*. Інші пункти меню *File* мають такі ж самі призначення, що й у багатьох програмах Windows. Це відкриття вже існуючого файлу проекту (*Open*), збереження створюваного (*Save, Save as*…), інформація про файл (*Info*), його розмір (*Size*), а також вихід з системи MAX+plus II (*Exit MAX+plus II*).

У меню *Edit* містяться команди копіювання (*Copy*), вирізання (*Cut*), вставки (*Paste*), знищення (*Delete*) елементів схеми, відміни останньої дії (*Undo*). Крім того, команди *Flip Vertical*, *Flip Horizontal*, *Rotate* дозволять повертати елементи схеми, що спрощує побудову логічної схеми. В меню *View* містяться команди з керування відображенням схеми: на весь екран, збільшення масштабу, зменшення масштабу, нормальний розмір та максимальний (рис. 9.28).

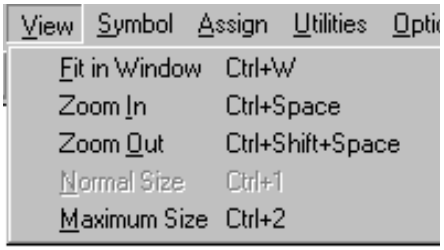


Рис. 9.20

У меню *Symbol* (рис. 9.28) містяться наступні команди:

- *Enter Symbol* – відкриває вікно вибору елемента, який треба додати в схему;
- *Edit Ports / Parameters* – відкриває вікно редагування виводів елементів в схемі;
- *Update Symbol* – оновлює символ в схемі.

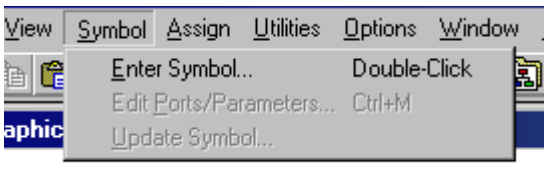


Рис. 9.21

За допомогою команд з меню *Assign* є можливість вводити, редагувати і знищувати типи призначень ресурсів, приладів та параметрів, котрі управляють компіляцією проекту. Вигляд меню приведено на рис. 9.29. Розміщуємо на схемі вхідні виводи.

Для цього з меню *Symbol* вибираємо команду *Enter Symbol* або двічі натискаємо ліву кнопку миші на білому полі графічного редактора. З'явиться вікно, показане на рис. 9.26.

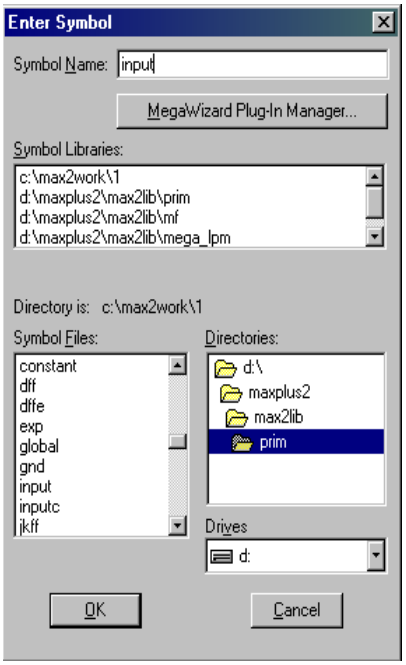


Рис. 9.22

У полі *Symbol Libraries* пропонується вибрати бібліотеки логічних функцій. У бібліотеці *prim* містяться найпростіші логічні елементи. У бібліотеці *mf* є елементи, які реалізують макрофункції (мультиплексори, дешифратори та ін.). Є бібліотека мегафункцій (*mega_lpm*) та бібліотека користувачів (*edif*), які містять символи, створені власне користувачем. Двічі натиснемо ліву кнопку миші навпроти бібліотеки примітивів. У полі *Symbol Files* з'явиться весь доступний список елементів. З цього списку треба вибрати елемент *input* і натиснути кнопку OK.

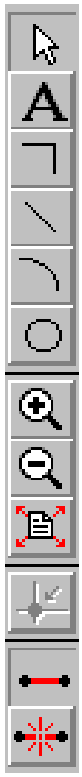
З'явиться вхідний вивід: 

.

Замість назви *PIN_NAME* слід ввести іншу – наприклад, *in*0. У подальшому це полегшить роботу і допоможе запобігти плутанини. Для цього треба натиснути ліву кнопку миші на *PIN_NAME* і відредагувати назву виводу. Додамо ще три виводи. Для цього треба повторити операції, описані вище, або скопіювати вже існуючий вивід. Для цього слід натиснути праву кнопку миші на виводі і з меню вибрати команду *Copy*. У потрібному місті схеми натиснути праву кнопку миші і вибрати команду *Paste*. Після цього слід відредагувати назву виводу. Переміщення елемента здійснюється шляхом виділення інструментом "стрілка" та переміщення в потрібне місце схеми з натиснутою лівою кнопкою миші.

Розглянемо основну панель інструментів. Вона містить дванадцять кнопок. Її вигляд показаний на рис. 9.23.

Стрілка має ту особливість, що при наведенні на вивід елемента або на кінець лінії вона автоматично перемикається на лінію. При наведенні на текст і після натискання лівої кнопки миші можна його редагувати.



– виділяє елементи схеми, лінії, вузли, текст

– дозволяє вставити текст в будь-якому місці схеми

– креслить лінії під прямим кутом

– креслить пряму лінію

– креслить дугу

– креслить коло

– збільшує масштаб схеми. Дозволяє бачити дрібні елементи схеми

– зменшує масштаб. Дозволяє побачити схему цілком

– змінює масштаб таким чином, що схема вміщується на весь екран

– ставить вузол в схемі

– вмикає прив’язку ліній до елементів схеми

– вимикає прив’язку ліній до елементів схеми

Рис. 9.23

За допомогою команд з меню *Assign* є можливість вводити, редагувати і знищувати типи призначень ресурсів, приладів та параметрів, котрі керують компіляцією проекту. Вигляд меню приведено на рис. 9.24.

Device – вибір пристрою. Призначає тип ПЛІС, на базі якої буде реалізовано проект. Якщо проект складається з декількох пристроїв, дана функція призначає мікросхеми конкретним проектам. Можна також обирати опцію *AUTO*. Тоді компілятор сам вибере пристрій із даного сім’ї. Взагалі, перед розробкою схеми слід обрати тип мікросхеми, за допомогою якої буде реалізована поставлена задача. З меню *Assign* слід вибрати команду *Device*. При виборі цієї команди з’являється вікно, вигляд якого показаний на рис. 9.25.

У полі *Device Family* вибирається сім’я ПЛІС. Для нашої задачі підійде FLEX8000.

Необхідно зняти позначку напроти *Show Only Fastest Speed Grades*. Тоді у полі *Devices* можна буде побачити всі мікросхеми сім'ї. Слід вибрати мікросхему EPF8282ALC84-4. У подальшому будемо працювати саме з цією мікросхемою.

Рис. 9.24

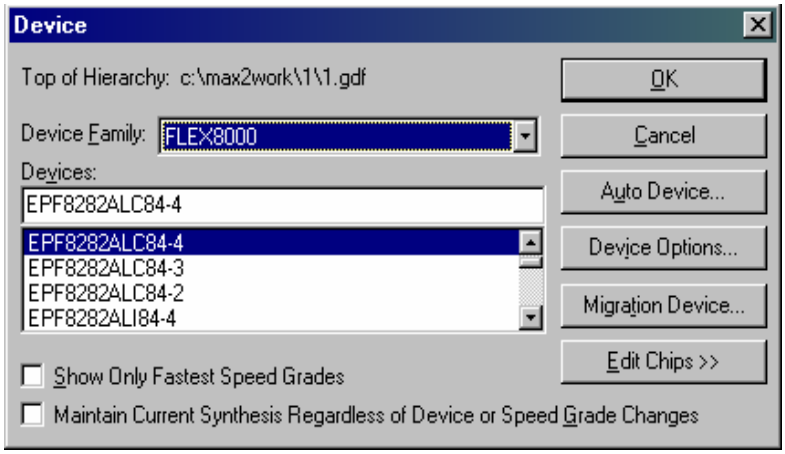


Рис. 9.25

Після вибору треба натиснути на ОК. *Assign >> Pin / Location / Chip …* – призначає вхід або вихід однієї логічної функції, такій, як примітив, мегафункція, конкретному контакту або вертикальному чи горизонтальному ряду. При виборі цієї команди з'явиться вікно, показане на рис. 9.26.

Назначити вивід на схемі будь-якому виводу на пристрої можна за допомогою команди *Assign >> Pin / Location / Chip …* або вибравши аналогічну команду з контекстного меню, яке з'являється при натисканні правої кнопки миші на виводі. Призначаємо вхідним виводам на схемі виводи на мікросхемі з номерами 1, 2, 3, 4. При досвіді проектування замість пошуків елементів по бібліотеках можна в полі *Symbol Name* (рис. 9.26) просто вводити назву останнього. Вхідний вивід має назву *input*.

Розмістимо на схемі чотири вихідні виводи. Слід виконати ті ж самі процедури, що описані вище. Вибирається елемент *output*. Потім треба змінити назви виводів на *out*0, *out*1, *out*2, *out*3. Добавимо у схему чотири інвертори. Цей елемент має назву *not*. З'єднаємо елементи, як показано на рис. 9.16, за допомогою інструмента "лінія".

Призначаємо вихідним виводам на схемі ніжки на мікросхемі з номерами 23, 24, 25, 27 відповідно. Це робиться наступним чином. З меню *Assign* вибирається команда *Pin* / *Location* / *Chip* …

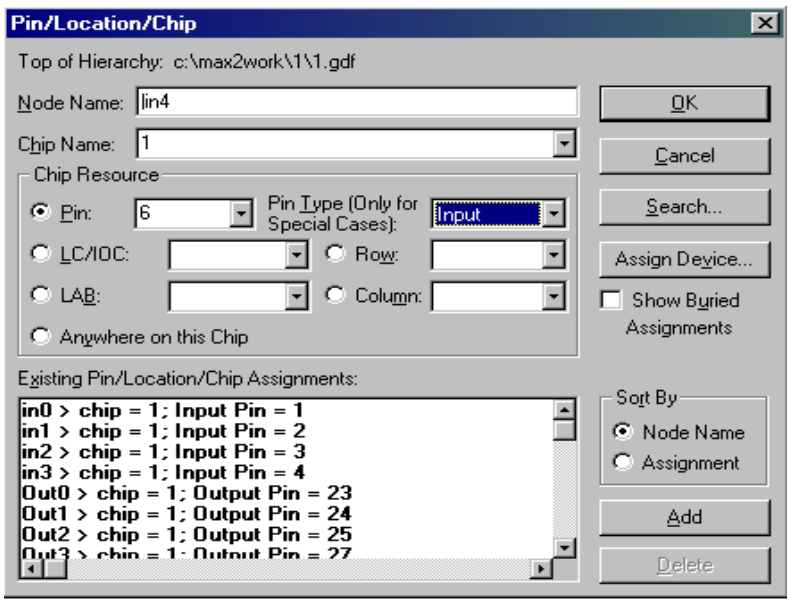
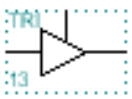


Рис. 9.26

У полі *Node Name* вводиться назва контакту. Назва повинна відповідати назві одного з контактів на схемі. Назви не повинні співпадати. У протилежному випадку компілятор видасть помилку. В полі *Chip name* вводиться назва чипу. Часто назва чипу співпадає з назвою головного файлу проекту. В полі *Pin* слід ввести номер виводу на мікросхемі. Слід зазначити, що не можна двом виводам у схемі призначати однакові виводи на мікросхемі. У полі *Pin Type* (*Only for Special Cases*) вводиться призначення виводу: вхід (*input*), вихід (*output*), двонаправлений (*bidir*) чи невизначений. **Уважно слідкуйте за призначенням типу виводу!** Не можна призначати виводам, на які подається вхідний сигнал, тип *output* – у найгіршому випадку може вийти з ладу мікросхема. Для двонаправлених виводів (*bidir*) слід в схемі встановлювати елемент TRI, схема якого і таблиця станів мають вигляд:

Inputs		Output
IN	OE	OUT
X	0	Z
1	1	1
0	1	0



Після запису всіх параметрів для виводу слід натиснути кнопку *Add*.

У полі *Existing Pin/Location/Chip Assigments* можна побачити виводи схеми, яким вже призначені виводи на ПЛІС. У полях *Row* і *Column* вказується, в якому рядку чи колонці відповідно повинні бути виводи на мікросхемі. Якщо не вказані параметри, то компілятор сам призначить виводам мікросхеми відповідні виводи схеми з розрахунку найменшої ресурсоємності та максимально можливої швидкодії (мінімальних часових затримок) мікросхеми.

Коли схема завершена і виводам на схемі призначені виводи на пристрої, можна компілювати проект. Для цього з меню *File* >> *Project* слід вибрати команду *Save* & *Compile*. Можна перевірити схему на помилки. Для цього обирається команда *Save* & *Check*, або натискається відповідна кнопка на панелі інструментів. При вдалій компіляції з'явиться повідомлення (рис. 9.27).

Якщо навпроти *errors* не буде стояти 0, то це означає, що в схемі є помилки, які для успішної компіляції проекту слід виправити. Виявити помилки допоможе *Message Processor*, вікно якого можна відкрити вибором цієї команди з меню MAX+plus II.

Рис. 9.27

Вікно компілятора зображене на рис. 9.28.

Сині прямокутники зверху вказують, яка саме дія виконується у поточний момент. Під ними вказані розширення файлів, які створюються в каталозі проекту. Червоний індикатор показує об'єм виконаної компілятором роботи.

Для початку компіляції слід натиснути кнопку *Start*. Зупинити компіляцію в будь-який момент часу можна натисканням кнопки *Stop*.

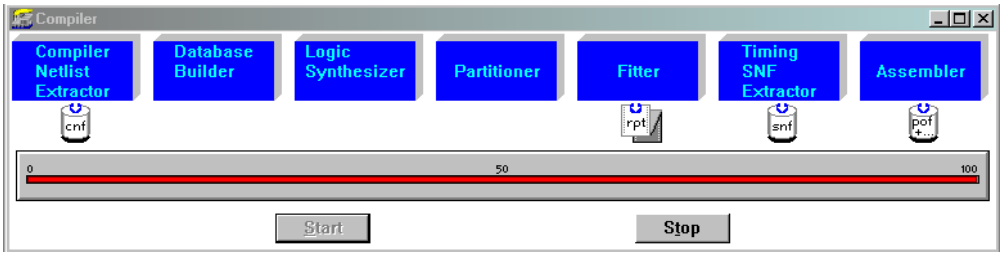


Рис. 9.28

Розглянемо інші команди меню *Assign*.

Timing Requirements (часові вимоги) – керує логічним синтезом і припасуванням окремих логічних функцій для одержання необхідних характеристик для часу затримки t_{PD} (вхід – вихід), t_{CO} (синхросигнал – вихід), t_{SU} (синхросигнал – час установки), f_{MAX} (частота синхросигналу). Користувач може також вирізувати з'єднання між шляхами поширення для конкретного сигналу (названого "вузлом" у системі MAX+plus II) і іншими вузлами проекту. Призначення часових параметрів вузла робиться по команді *Assign*/*Timing Requirements* (рис. 9.29).

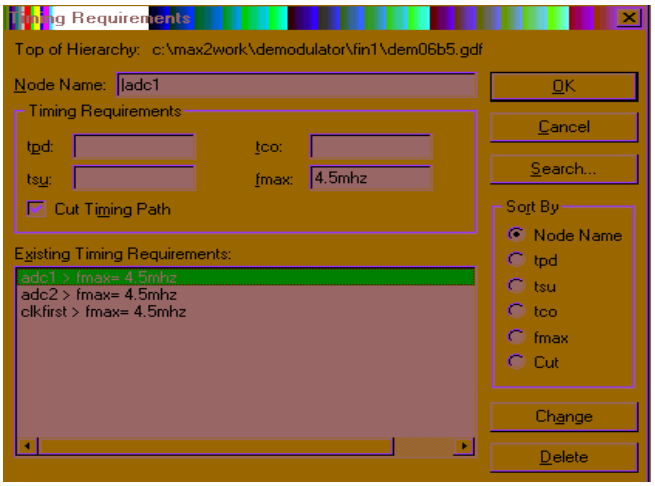


Рис. 9.29

Clique – призначає, які логічні функції повинні залишатися разом. Групування логічних функцій гарантує, що вони реалізуються в одному і тому ж блоці логічної структури. В полі *Clique* задають його ім'я В полі *Node name* задають ім'я вузла.

Chip – задає, які логічні функції мають бути реалізовані в одному пристрої у разі розділення проекту на частини.

Logic Options – керує синтезом окремих логічних функцій під час компіляції з використанням стилю логічного синтезу або окремих опцій логічного синтезу. Користувач може використовувати вже готові стилі, розроблені фірмою *ALTERA*, чи створювати власні.

Probe – привласнює ім’я, що легко запам’ятовуються, входу чи виходу логічної функції.

Connected Pins – задає зовнішнє з’єднання двох або більше виводів на схемі користувача.

Local Routing – привласнюється коефіцієнт розгалуження по виходу вузла логічного елемента, що знаходиться в одному блоці LAB або в сусідніх, суміжних з вибраним вузлом, з використанням загальних місцевих зв’язків. Слід відмітити, що ця команда активна лише при роботі з сім’єю FLEX6000. Вона ігнорується іншими сім’їми.

Timing assignment – керує логічним синтезом та підгонкою окремих логічних функцій для отримання необхідних значень для часу затримки t_{PD}, t_{CO}, t_{SU}, f_{MAX}. Можна задати опції пристрою для компілятора, для того, щоб він їх використовував для усіх пристроїв при обробці проекту. Для резервування додаткових можливостей логіки на майбутнє можна задати процентне відношення виводів і логічних елементів, котрі повинні залишатися невикористаними під час поточної компіляції. Глобальні часові вимоги до проекту вводяться у вікні *Global Project Timing Requirements*. Його вигляд зображено на рис. 9.30.

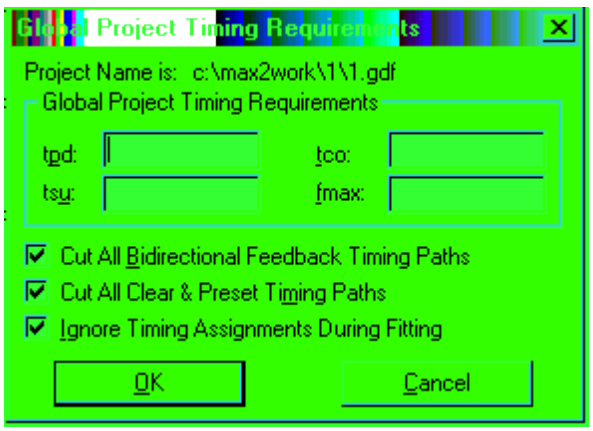


Рис. 9.30

Прапорець *Cut All Bidirectional Feedback Timing Paths* дозволяє виключити всі кола зворотного зв’язку для двонаправлених виводів. Прапорець *Cut All Clear&Preset Timing Paths* дозволяє знищити зв’язки між усіма колами скиду передустановки проекту. Останній прапорець дозволяє запустити трасувальник без урахування часових обмежень проекту. Слід зазначити, що на початку компіляції слід вибрати цей прапорець, а в подальшому скинути. Керований часовий синтез можливий лише для пристроїв FLEX.

Редактори Max+plus II мають загальні функції – такі, як, наприклад, збереження та відкриття файлів. Загальними для редакторів є наступні функції:

- створення файлів символів та файлів з прототипами функцій;
- пошук вузлів (*node location*);
- траверз ієрархічного дерева (*hierarchy traversal*);
- випливаючі вікна меню;
- часовий аналіз;
- пошук та заміна фрагментів тексту;
- відміна останньої дії.

На панелі інструментів є кнопки (рис. 9.28), що виконують функції меню *Utilites*, показаному на рис. 9.31.

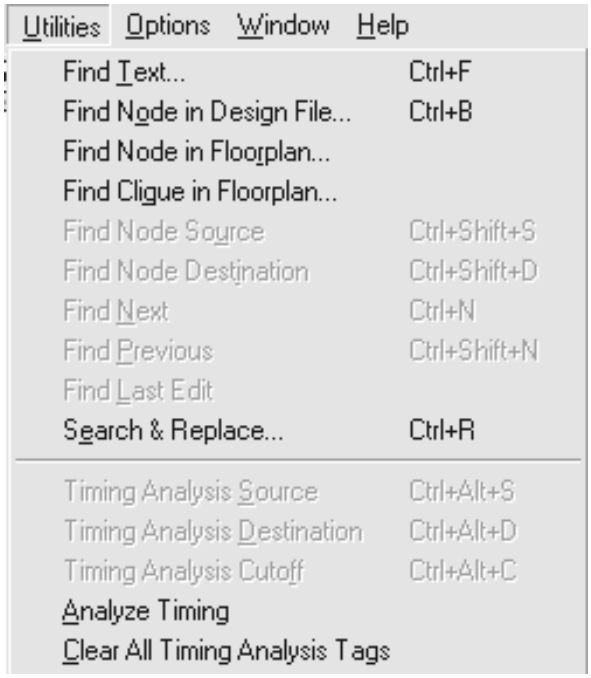


Рис. 9.31

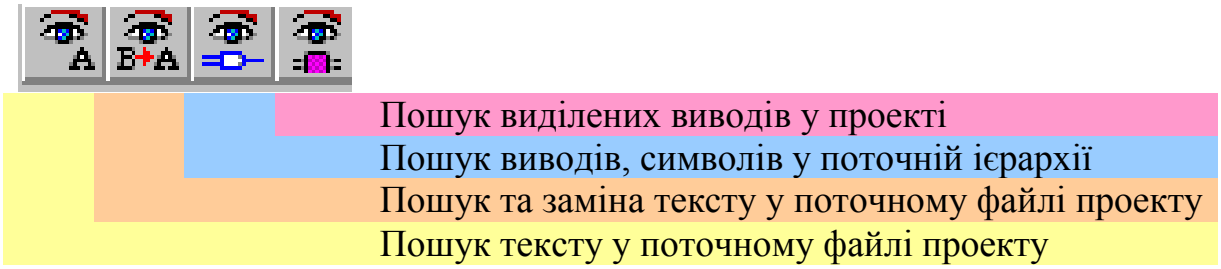


Рис. 9.32

Команда *Search & Replace* знаходить та замінює текст у поточному проекті, точніше відкритому файлі проекту.

Повернемося до нашого прикладу. Відкриємо вікно *Floorplan Editor*, натиснувши відповідну кнопку на панелі інструментів (Табл. 9.2) або обравши відповідну команду з меню MAX+plus II (рис. 9.15). Вигляд цього вікна показаний на рис. 9.33.

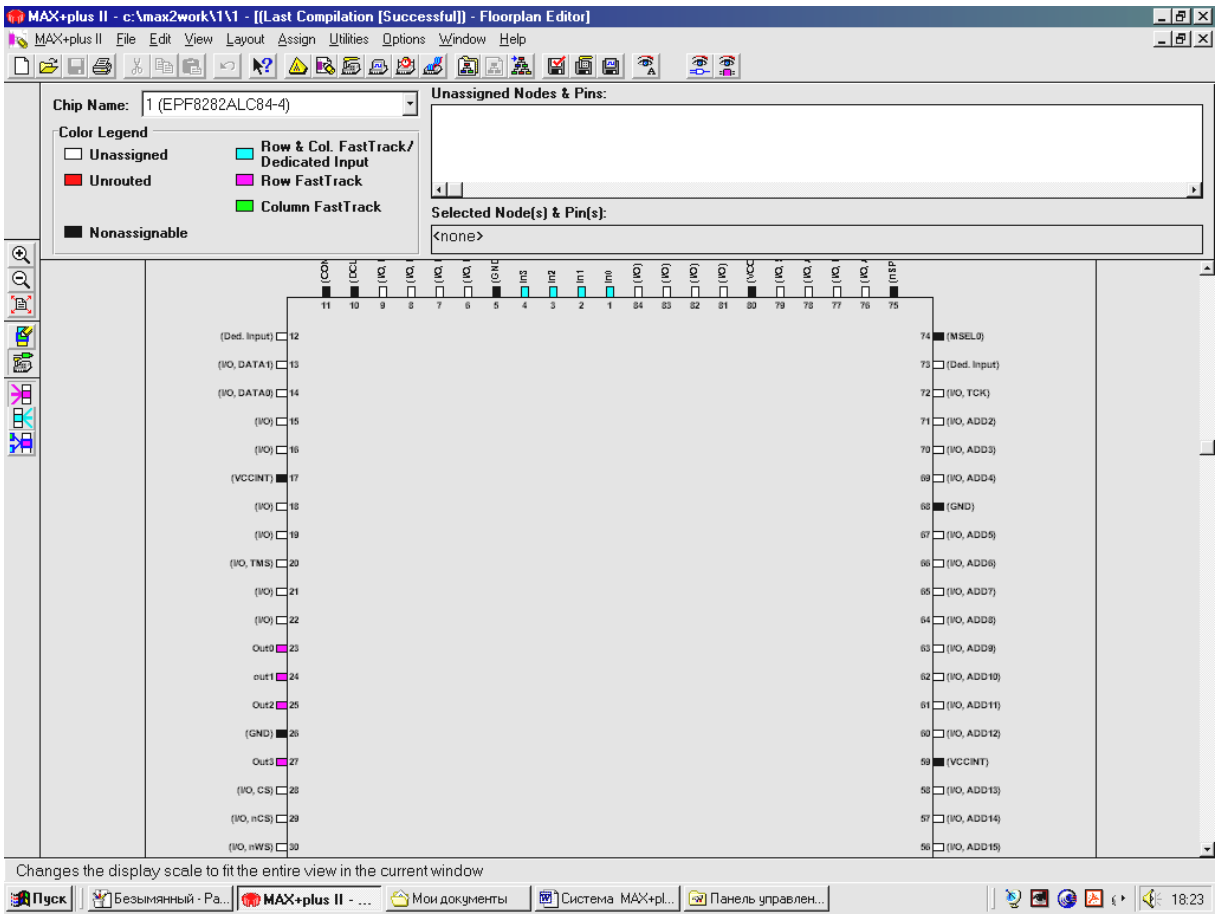


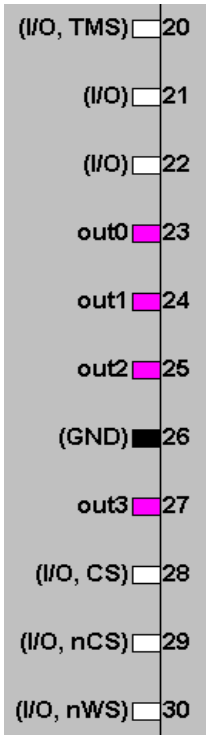
Рис. 9.33

Для перемикання вигляду мікросхеми слід двічі натиснути ліву кнопку миші на ній.

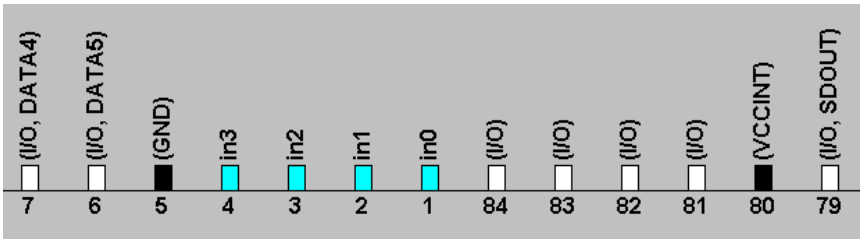
Кольором виділені виводи, яким призначені виводи у логічній схемі, створеної у графічному редакторі.

Як бачимо, фіолетовим кольором позначені вихідні виводи мікросхеми (рис. 9.34, *а*). Вони мають номери 23, 24, 25, 27. Вхідні виводи позначені зеленим кольором (рис. 9.34, *б*) і мають номери 1, 2, 3, 4.

Чорний колір мають виводи, використовувати які не дозволяється. Вони призначені для живлення мікросхеми, а також для її програмування. До цих виводів відноситься вивід номер 14. Тому його не слід використовувати при призначенні виводів у редакторі; у протилежному випадку компілятор видасть помилку.



а)



б)

Рис. 9.34

Зліва розміщені інструменти зміни масштабу. Вони працюють аналогічно до відповідних інструментів інших редакторів системи MAX+plus II. Білим кольором позначені виводи, які не мають призначень. Червоним – виводи, для яких не вдалося виконати призначення. У полі *Chip Name* вибирається тип мікросхеми. Це потрібно якщо проект розбивається на частини. Під час компіляції створюється файл, в якому міститься інформація про компіляцію. Це текстовий файл, що має розширення **.rpt**. Щоб його відкрити, треба в меню *File* вибрати команду *Open*. У вікні, що з'явилось, у полі *Files* слід вибрати файл з розширенням **.rpt**. Для вибору файлу слід в опції *Show file list* вибрати *All file*. Вигляд вікна схожий на той, що зображений на рис. 1.6. Відкриємо цей

файл. З’явиться вікно *Text Editor*, в якому можна побачити інформацію про проект. Можна побачити рядок:

```
***** Project compilation was successful
```

Це перше повідомлення, яке свідчить про вдалу компіляцію проекту.

Далі йде інформація про тип використовуваного пристрою та кількість виводів:

```
** DEVICE SUMMARY **

Chip/                    Input   Output   Bidir              LCs
POF      Device          Pins    Pins     Pins      LCs    % Utilized

1        EPF8282ALC84-4   4        4        0       4          1  %

User Pins:                4        4        0
```

Як бачимо, в нашому проекті застосовані по 4 вхідних та вихідних виводи. Це вказано у полях *Input Pins* та *Output Pins*. Використовується мікросхема EPF8282ALC84-4, яка вказана у полі *Device*.

Трохи нижче приведена інформація про використані виводи мікросхеми:

```
** PIN/LOCATION/CHIP ASSIGNMENTS **

                  Actual
    User       Assignments
Assignments   (if different)     Node Name

1@1                               in0
1@2                               in1
1@3                               in2
1@4                               in3
1@23                              out0
1@24                              out1
1@25                              out2
1@27                              out3
```

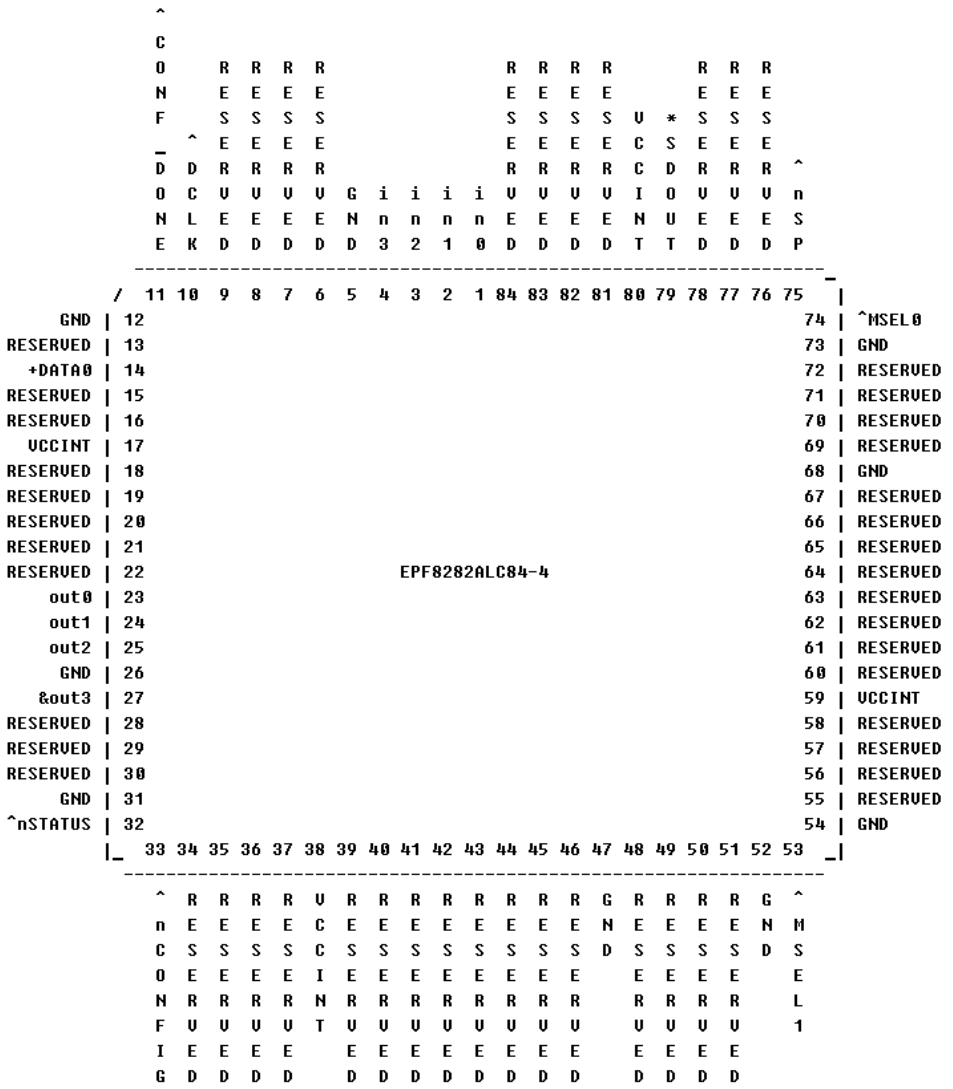
У таблиці вказується номер виводу мікросхеми та його назва.

Далі слідує інформація про логіку дизайну. Якщо помилок немає, можна побачити рядок:

```
Device-Specific Information:                          c:\max2work\1\1.rpt
1

***** Logic for device '1' compiled without errors.
```

Інформація про мікросхему приведена нижче. За допомогою символів у текстовому файлі створений зовнішній вигляд чипу.



На ньому позначені використані виводи:

N.C. – No Connect – вивід, який не має внутрішнього з’єднання у пристрої;

VCCINT – зарезервований вивід, котрий має бути приєднаним до джерела живлення напругою +5 В;

VCCIO – зарезервований вивід, котрий має бути приєднаним до джерела живлення напругою +5 В;

GND – зарезервований вивід, котрий приєднується до вузла з нульовим потенціалом (земля);

RESERVED – виводи, що не використовуються (повинні бути роз’єднані);

^ – вивід для конфігурації мікросхеми;

+ – вивід, що має три стани при використанні чипу;

***** – вивід, що використовується як вихідний при роботі з мікросхемою;

PDn – вивід зниженого енергопостачання;

@ – вивід спеціального призначення.

При розробці великих схем може знадобитися інформація про використання ресурсів. Приводиться інформація про кількість вхідних, вихідних або двонаправлених виводів, кількість тригерів, регістрів, осередків, ланцюгів повторювачів, кількість каскадних ланцюгів та осередків у них.

```
** RESOURCE USAGE **

Logic                Column       Row
Array                Interconnect Interconnect         Clears/     External
Block   Logic Cells  Driven       Driven       Clocks  Presets   Interconnect
B1       1/ 8( 12%)   0/ 8(  0%)   1/ 8( 12%)    0/2    0/2       1/24(  4%)
B2       1/ 8( 12%)   0/ 8(  0%)   1/ 8( 12%)    0/2    0/2       1/24(  4%)
B3       1/ 8( 12%)   0/ 8(  0%)   1/ 8( 12%)    0/2    0/2       1/24(  4%)
B4       1/ 8( 12%)   0/ 8(  0%)   1/ 8( 12%)    0/2    0/2       1/24(  4%)


Total dedicated input pins used:                 0/4      (  0%)
Total I/O pins used:                            10/64     ( 15%)
Total logic cells used:                          4/208    (  1%)
Average fan-in:                                 1.00/4    ( 25%)
Total fan-in:                                   4/832     (  0%)

Total input pins required:                       4
Total input I/O cell registers required:         0
Total output pins required:                      4
Total output I/O cell registers required:        0
Total buried I/O cell registers required:        0
Total bidirectional pins required:               0
Total reserved pins required                     2
Total logic cells required:                      4
Total flipflops required:                        0
Total logic cells in carry chains:               0
Total number of carry chains:                    0
Total logic cells in cascade chains:             0
Total number of cascade chains:                  0

Synthesized logic cells:                         4/ 208   (  1%)

Logic Cell Counts

Column:  01  02  03  04  05  06  07  08  09  10  11  12  13  Total
 A:       0   0   0   0   0   0   0   0   0   0   0   0   0      0
 B:       1   1   1   1   0   0   0   0   0   0   0   0   0      4

Total:    1   1   1   1   0   0   0   0   0   0   0   0   0      4
```

Взагалі, під час компіляції створюється багато файлів. Структура системного каталогу та розширення файлів приведена в табл. 9.3.

Таблиця 9.3

Підкаталог	Опис
.\drivers	Містить драйвери пристроїв для середовища WINDOWS (тільки для інсталяції на платформі PC у середовищі WINDOWS)
.\edc	Містить командні файли , що поставляються фірмою Altera, (**.edc**) і генерують вихідні файли (**.edo**) за замовленням користувача для заданих умов тестування
.\lmf	Містить файли макробібліотек (**.lmf**), що поставляються фірмою Altera. Установлюють відповідність між логічними функціями користувача й еквівалентними логічними функціями пакету MAX+plus II
.\max2inc	Містить Include-файли (файли "заголовків") з прототипами функцій для розроблених фірмою ALTERA макрофункцій. У прототипах функцій перелічуються порти (виводи) для макрофункцій, реалізованих у текстових файлах проекту (**.tdf**), написаних мовою AHDL
.\max2lib\edif	Містить примітиви і макрофункції, що використовуються для інтерфейсів EDIF
.\max2lib\mega_lpm	Містить мегафункції, у тому числі бібліотеку функцій параметризованих модулів (LPM) і Include-файли для них з відповідними прототипами, написаними мовою AHDL
.\max2lib\mf	Містить користувальницькі і застарілі (74-series) макрофункції.
.\max2lib\prim	Містить примітиви, що поставляються фірмою ALTERA
.\vhdlnn\altera	Містить бібліотеку altera із програмним пакетом MAX+plus II. У цей пакет входять усі примітиви, мегафункції і макрофункції системи MAX+plus II, підтримувані мовою VHDL
.\vhdlnn\ieee	Містить бібліотеку ieee пакетів VHDL, у тому числі std_logic_1164, std_logic_arith, std_logic_signed і std_logic_unsigned
.\vhdlnn\std	Містить бібліотеку std з пакетами стандартів і засобів вводу/виводу тексту, описаними в довіднику по стандартах інституту IEEE мовою VHDL IEEE Standard VHDL Language Reference Manual

Мікросхема програмується за допомогою програматора. Він запускається шляхом вибору відповідної команди з головного меню системи (рис. 9.15). Вікно програматора показане на рис. 9.35.

Тепер слід з меню *Options* вибрати команду *Hardware Setup*. З'явиться вікно, показане на рис. 9.36.

У табл. 9.4 приводиться структура робочого каталогу.

Таблиця 9.4

Підкаталог	Опис
.\ahdl	Містить файли прикладів, що ілюструють тему “Як використовувати мову AHDL” (How to Use AHDL) в електронному довіднику (MAX+ plus II Help) і в керівництві MAX+plus II AHDL.
.\chiptrip	Містить усі файли навчального проекту chiptrip, описаного в керівництві MAX+plus II AHDL.
.\edif	Містить усі файли прикладів, що ілюструють особливості EDIF в електронному довіднику (MAX+plus II Help).
.\tutorial	Містить інформаційний файл readme навчального проекту chiptrip. Усі файли, створені в проекті chiptrip, повинні знаходитися в цьому підкаталозі.
.\vhdl	Містить файли прикладів, що ілюструють тему “Як використовувати мову VHDL” (How to Use VHDL) в електронному довіднику (MAX+plus II Help) і в керівництві MAX+plus II VHDL.
.\verilog	Містить файли прикладів, що ілюструють тему “Як використовувати мову верифікаційного протоколу Verilog HDL” (How to Use Verilog HDL) в електронному довіднику (MAX+plus II Help) і в керівництві MAX+plus II Verilog HDL.

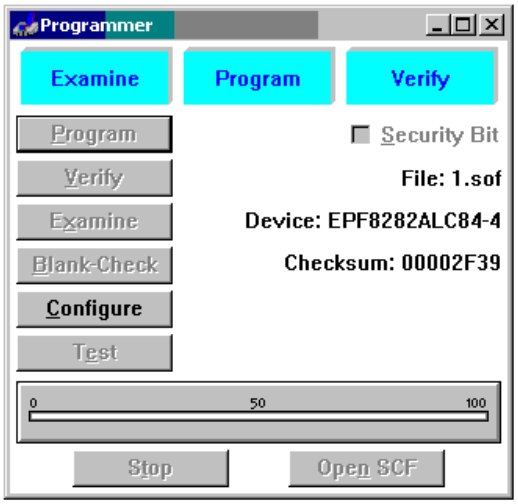


Рис. 9.35

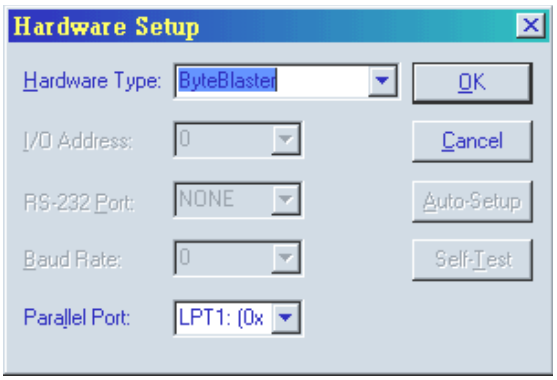


Рис. 9.36

У полі *Hardware Type* слід вибрати *ByteBlaster*, а в *Parallel Port* – LPT1:(0x378), після чого натиснути кнопку OK. У вікні програматора (рис. 9.31) слід натиснути кнопку *Configure*. Перед цим слід впевнитися, що стенд увімкнений та підключений до комп’ютера, червоні світлодіоди не світяться, а зелені горять усі. У разі успішного програмування мікросхеми з’явиться вікно повідомлення про успішність компіляції.

Слід звернути увагу на те, що в меню *Options* команда *Hardware Setup* з'явиться лише при активному вікні програматора. При спробі запрограмувати мікросхему може з'явитися повідомлення про невдалу компіляцію. В ньому говориться, що *ByteBlaster* відсутній, і пропонується перевірити живлення та шлейфи. Дійсно, найчастіше саме з цих причин спроби запрограмувати мікросхему невдалі.

Наш приклад ілюструє роботу в графічному редакторі системи. При розробці швидкодіючих систем важливими будуть часові параметри.

Проаналізувати часові затримки можна за допомогою сигнального редактора та симулятора, вікно якого можна відкрити, вибравши з меню MAX+plus II (рис. 9.11) відповідний пункт.

Спочатку розглянемо роботу з сигнальним редактором. Часові співвідношення досліджуються у сигнальному редакторі (*Waveform Editor*). Сигнальний редактор виконує дві ролі: він є інструментом вводу логіки за допомогою часових діаграм та аналізу результатів тестування. Користувач може створювати сигнальні файли проекту (**.wdf**), котрі містять логіку для проекту, а також файли каналів тестування (**.scf**), котрі містять вхідні вектори для тестування та функціонального налагодження. Новий файл можна створити командою *New* меню *File*. Але спочатку треба відкрити сигнальний редактор (*Waveform Editor*). Треба з меню MAX+plus II вибрати пункт *Waveform Editor*. Після цього з'явиться вікно сигнального редактора (рис. 9.37).

Сигнальний редактор має наступні можливості:

- можна створити або відредагувати вузол для отримання типу вхід/вихід, який представляє собою вхідний або вихідний контакт;
- при розробці WDF можна задати тип логіки, котра робить кожен вузол контактом;
- можна задавати значення по замовчуванню в логічному вузлі для активного логічного рівня: високий (1), низький (0), невизначений (X), з високим імпедансом (Z). Можна копіювати, вставляти, переміщувати та

знищувати вибрану частину форми сигналу або всю форму. Також є можливість відображати на екрані сітку, вводити коментарії, змінювати масштаб відображення.

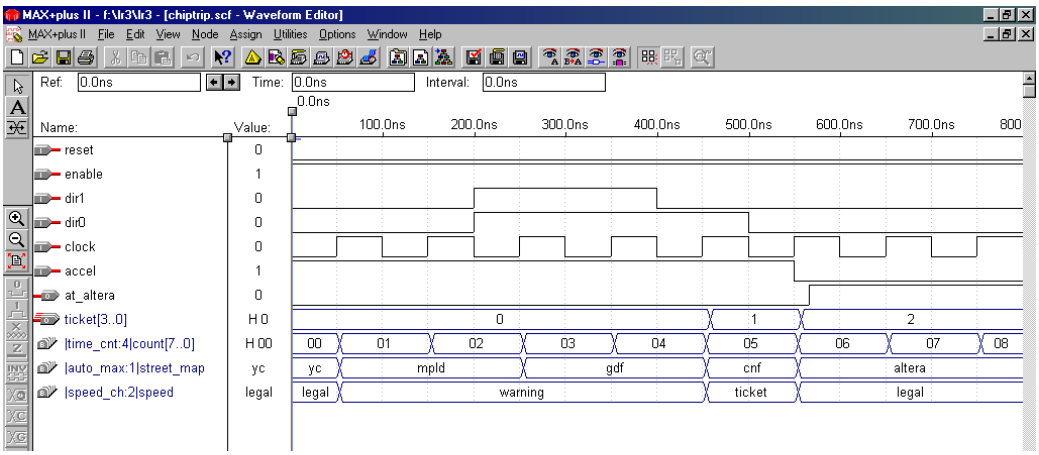


Рис. 9.37

Запустивши систему MAX+plus II і відкривши сигнальний редактор, треба записати цей файл на диск. У діалоговому вікні збереження файлу (*File >> Save As…*) слід встановити розширення **.wdf**.

У верхній частині редактора є декілька полів. У полі *Name* вказується назва виводу. Для назв виводів справедливі ті ж самі вимоги, як і до ідентифікаторів у мовах програмування високого рівня. У полі Type вказується тип виводу: вхід (*input*), вихід (*comb*). У полі *Value* вказується значення логічної змінної (0 або 1) (рис. 9.38).

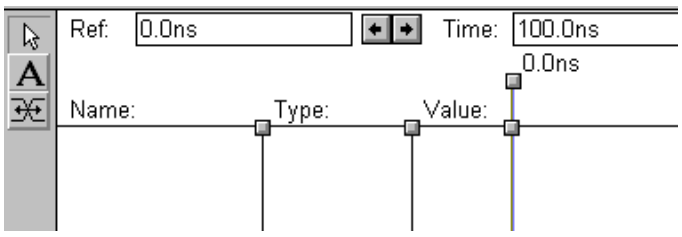


Рис. 9.38

Як приклад, реалізуємо в сигнальному редакторі інвертор. Двічі натискаємо ліву кнопку миші, попереднє встановивши курсор у полі *Name* або з меню *Node* вибираємо команду *Insert Node*. З'явиться вікно, показане на рис. 9.39.

У полі *Node Name* слід ввести назву виводу, а в *I/O Type* вказати його тип. Після цього натиснути кнопку OK.

У полі *Default Value* вказується початкове значення сигналу на цьому виводі.

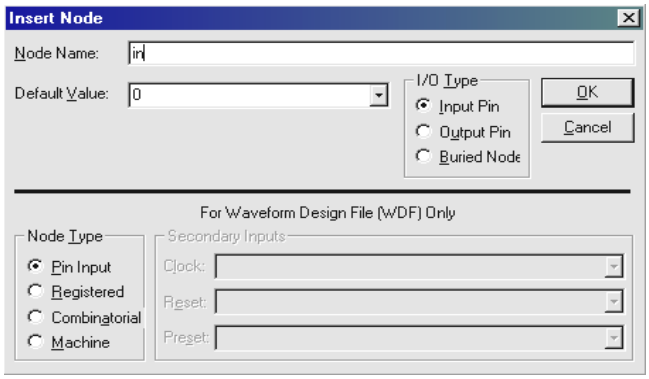


Рис. 9.39

Подібну процедуру слід провести для того, щоб добавити вихідний вивід. Для цього треба у *I/O Type* вказати *Output Pin*. На екрані у полі *Name* з'явиться два мініатюрні зображення виводів (рис. 9.40).

Рис. 9.40

Ця процедура може бути дещо спрощена, якщо у вікні *Insert Node* натиснути кнопку *List*. При цьому у вікні з'явиться список входів та виходів, які необхідно лише вибрати.

Тепер треба намалювати часові діаграми. Для того, щоб намалювати їх у сигнальному редакторі, треба зробити наступне: натискаємо ліву кнопку миші навпроти виводу *in* (рис. 9.41) та між значеннями часу 100ns та 200ns і тримаючи її трохи перемістимо курсор праворуч.

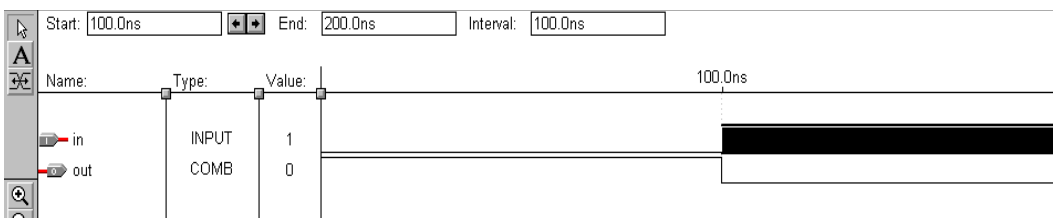


Рис. 9.41

З’явиться чорний прямокутник, що свідчить про виділення цього проміжку Для виділення більшого проміжку часу слід натиснути ліву кнопку миші, і не відпускаючи її протягнути курсор по необхідній зоні. Зліва екрану активізується панель інструментів (рис. 9.42), на якій розташовані кнопки.

	– Збільшення масштабу зображення часової діаграми
	– Зменшення масштабу зображення часової діаграм
	– Зміна масштабу таким чином, щоб вся діаграма розмістилася на екрані
	– Встановлення рівня логічного нуля
	– Встановлення рівня логічної одиниці
	– Невизначене значення
	– Рівень високого імпедансу
	– Інверсія сигналу
	– Послідовність імпульсів 0-1-0-1-0-1-0-1-0-1-0-1-0-1-…
	– Задає вільну послідовність імпульсів (наприклад 0-1-1-0-1-1-0-1-1-0-1-1-0-1-1-0-…)

Рис. 9.42

Коли область має виділення, треба натиснути на кнопку  “Встановлення рівня логічної одиниці”. Таку ж саму процедуру треба повторити для виводу *out*, виділивши відповідну область. На екрані з’явиться картинка, показана на рис. 9.43.

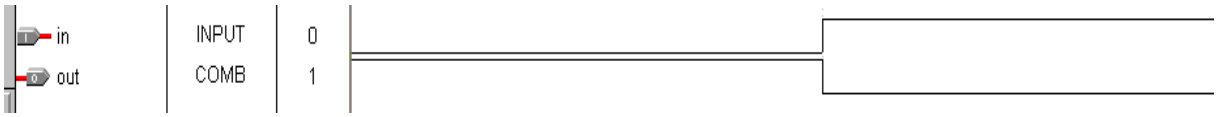


Рис. 9.43

Слід призначити виводам на схемі відповідні виводи безпосередньо на мікросхемі. Для цього потрібно з меню *Assign* вибрати команду *Pin/Location/Chip*. Виводу *in* призначити 1-й вивід, а виводу *out* призначити 23-й вивід мікросхеми.

Робота з діалоговим вікном, що з'явиться, детально описана трохи вище. Також необхідно задати часовий діапазон. Для цього в меню *File* слід вибрати команду *End Time…* З'явиться вікно, зовнішній вигляд якого показаний на рис. 9.44.

У полі *End Time* слід вказати час 200ns. В меню *View* є команда *Time Range*. Вона дозволяє задавати проміжок часу, що буде відображатися на екрані монітора. Якщо все це зроблено, можна компілювати проект. Для цього треба разом натиснути кнопки Ctrl та L. У разі вдалої компіляції проекту з'явиться стандартне повідомлення. Можна побачити зовнішній вигляд мікросхеми з виводами, які використовуються. Вони будуть позначені характерними кольорами. Для виводу на екран порівневого планувальника треба з меню MAX+plus II вибрати *Floorplan Editor*.

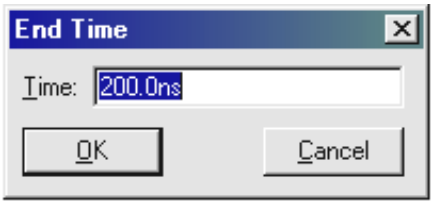


Рис. 9.44

Реалізуємо більш складну логічну функцію. Нехай логічна функція задана виразом:

$$Y = \overline{X}_4\overline{X}_3X_2X_1 + \overline{X}_4X_3\overline{X}_2X_1 + \overline{X}_4X_3X_2\overline{X}_1 + X_4\overline{X}_3\overline{X}_2X_1 + X_4\overline{X}_3X_2\overline{X}_1 + X_4X_3\overline{X}_2\overline{X}_1$$

У сигнальному редакторі спроектуємо схему, що реалізує цю логічну функцію. В схемі буде чотири вхідні виводи та один вихідний. Назвемо вхідні виводи *in*1, *in*2, *in*3, *in*4. Вихідний вивід *out*. Часова діаграма у Сигнальному редакторі має вигляд, зображений на рис. 9.45.

Слід зазначити, що не обов'язково кожен раз виділяти окрему область та задавати для неї логічний рівень. Цей процес можна дещо автоматизувати.

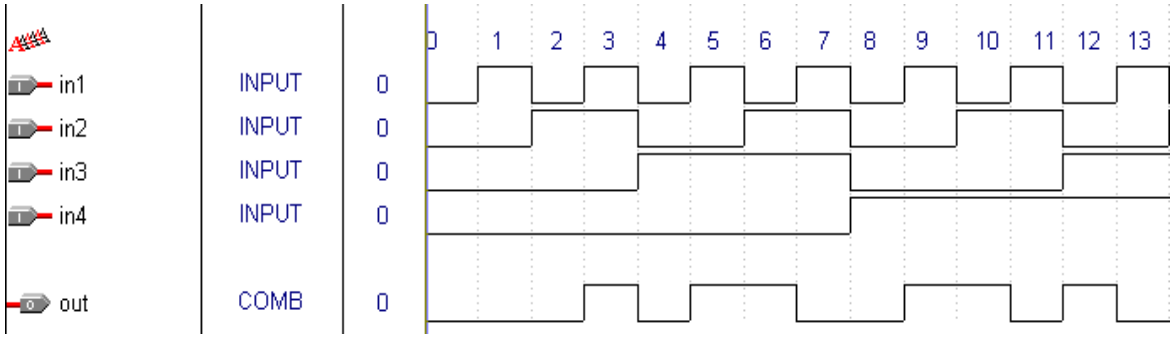


Рис. 9.45

Натиснемо ліву кнопку миші навпроти виводу *in*1 у полі *Type* на *INPUT*. При цьому виділиться весь рядок чорним кольором. Натискаємо на кнопку панелі інструментів . З'явиться вікно, показане на рис. 9.46.

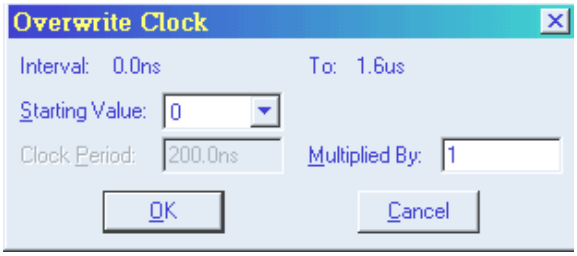


Рис. 9.46

У полі *Starting Value* встановлюємо початкове значення логічного рівня. У полі *Multiplied By* встановити 1 – для першої ніжки, 2 – для другої, 4 – для третьої, 8 – для четвертої. Після цього слід встановити відповідні рівні на виході. Слід виділити область під номерами (рис. 9.45), а потім натиснути кнопку панелі інструментів. Коли все зроблено, можна призначати виводи на схемі виводам мікросхеми. Слід з меню *Assign* вибрати команду *Pin/Location/Chip*. Детально ця процедура описана вище. Слід не забути вказати тип мікросхеми. У протилежному випадку елементи управління діалогового вікна будуть неактивними. Коли все зроблено, можна компілювати проект. Натискаємо комбінацію клавіш Ctrl+L. У разі вдалої компіляції з'явиться стандартне повідомлення. Відкриваємо вікно компілятора і, зробивши всі попередні настройки, записуємо інформацію в мікросхему.

Спробуємо тепер отримати часові характеристики нашого інвертора. Слід зберегти файл на диск. Назву файлу залишаємо незмінною. Тепер відкомпілюємо проект. Запустимо симулятор, вибравши його з меню системи (рис. 9.15). Вікно симулятора зображене на рис. 9.47.

Необхідно поставити мітки навпроти Setup/Hold, Oscillation. У полі Oscillation слід ввести значення 200ns. Після цього натискаємо Start. Тепер навпроти виводу Out з'явиться діаграма (рис. 9.48). Звернемо увагу на те, що рівень логічної одиниці дещо зміщений. Це пов'язано з часовою затримкою. Часові параметри схеми можна дослідити за допомогою Timing Analyzer.

Для початку аналізу слід натиснути кнопку Start.

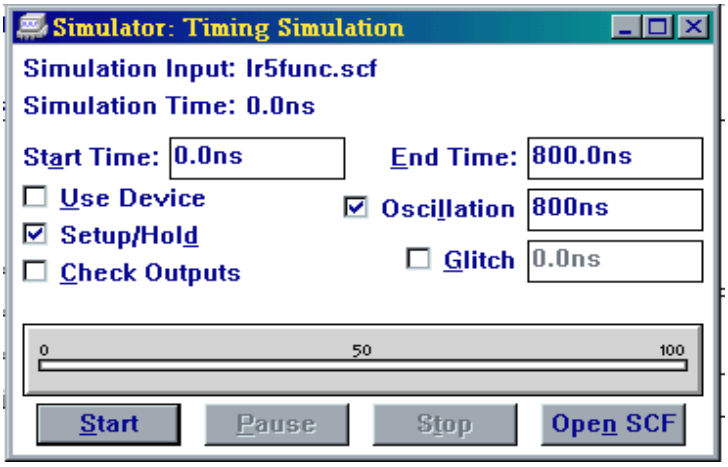


Рис. 9.47

Часові затримки, що виникають в схемі, зображені на рис. 9.49. Як бачимо, часова затримка складає близько 20 нс. Система враховує цю затримку і відображає її у часовій діаграмі (рис. 9.48).

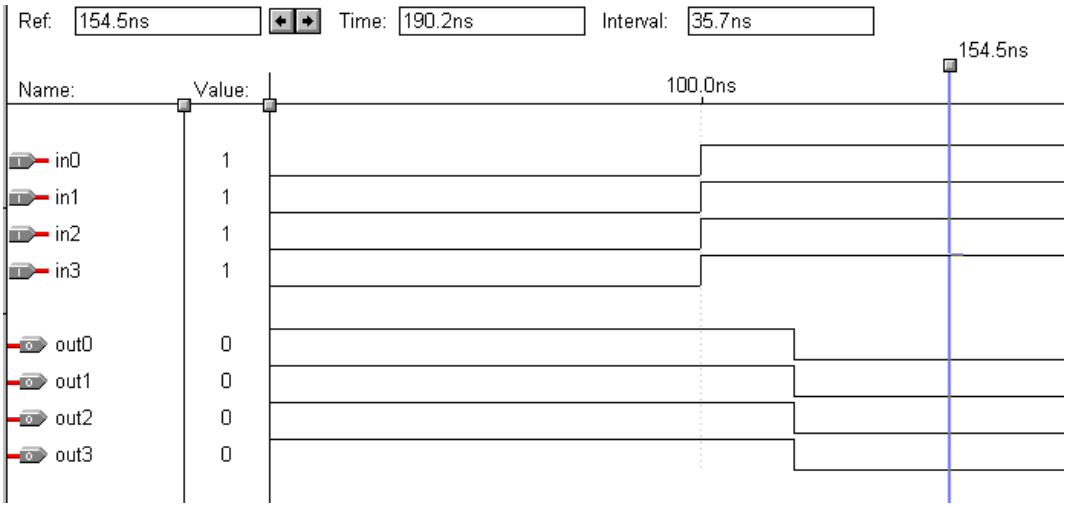


Рис. 9.48

	Destination			
	Out0	Out1	Out2	Out3
In0	20.3ns			
In1		20.3ns		
In2			20.3ns	
In3				20.3ns

Рис. 9.49

Можна і по-іншому оцінювати величини часових затримок та їх вплив на послідуючі схеми. Наприклад, для логічної функції $y = x_1\, x_2 + \overline{x_1}\, \overline{x_2}\, \overline{x_3}$,

реалізованої з використанням дешифратора 74138 та допоміжного елемента 3I-HI, часова діаграма приводиться на рис. 9.50.

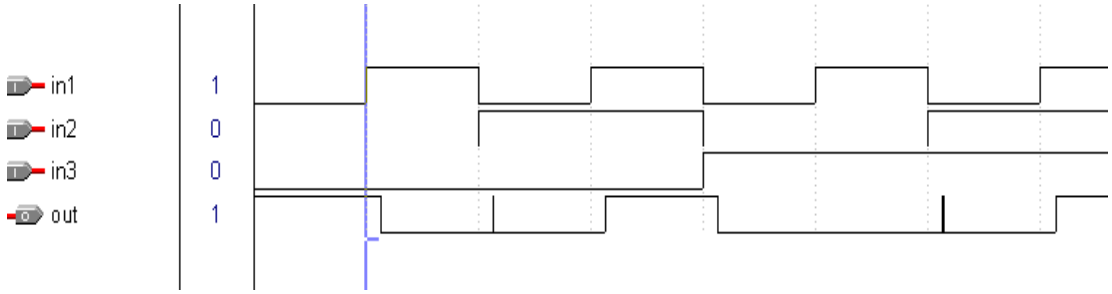


Рис. 9.50

На часовій діаграмі чітко бачимо затримки, що виникають у схемі. Проаналізувавши їх за допомогою редактора *Timing Analyser* (рис. 9.51), легко можна підрахувати максимальну частоту, при якій схема буде працювати.

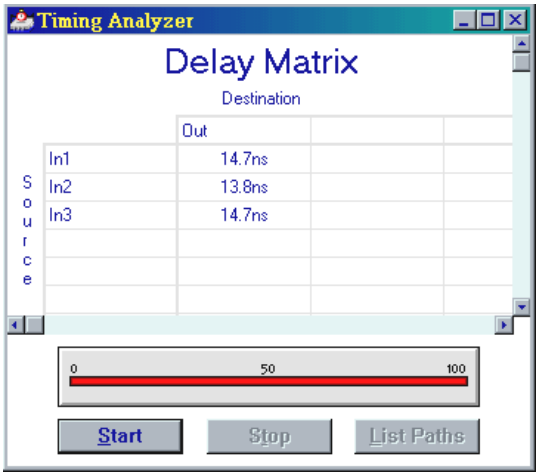


Рис. 9.51

Як бачимо, часова затримка складає близько 15 нс. Система враховує цю затримку і відображає її у часовій діаграмі (рис. 9.50). У верхній частині редактора розташовані індикатори. Вони дають змогу проектанту орієнтуватися у часових параметрах схеми. Вікно *Timing Analyser* відображає часові затримки, які мають місце при проходженні сигналів по кожному з входів. Слід звернути увагу на той факт, що затримки по кожному з входів можуть бути різними.

На рис. 9.51 приведені величини часових затримок по входах 1, 2, 3, якщо вибрані виводи мікросхеми відповідно 1, 2, 3. Як ми бачимо, по входу in2 величина затримки менша, ніж по інших входах (13,8 нс порівняно з 14,7 нс).

На практиці така невідповідність може привести до появи завад в вихідному сигналі (рис. 9.50). Ця схема працюватиме при частотах до 65 МГц.

Повернемося до розгляду функцій компілятора. Компіляцію можна запустити з будь-якого редактора MAX+plus II. Компілятор автоматично обробляє всі файли поточного проекту. Процес компіляції можна спостерігати у вікні компілятора (рис. 9.32). При цьому можна побачити наступне:

- спустошується і перевертається пусковий годинник, що вказує на активність компілятора;
- по черзі з'являються прямокутники модулів компілятора, коли компілятор завершує кожен етап обробки;
- під прямокутником модуля компілятора з'являється піктограма вихідного файлу, що створений даним модулем. Для відкриття відповідного файлу слід двічі натиснути лівою кнопкою миші на піктограмі;
- відсоток завершення компіляції поступово збільшується (до 100%);
- під час розбивки і монтажу кнопка компілятора *Stop* (Зупинити) перетворюється в кнопку *Stop*/*Show Status* (Зупинити/Показати стан). Ви можете натиснути її для відкриття діалогового вікна, у якому показується поточний стан розбивки і монтажу проекту;
- при виявленні в процесі компіляції будь-яких помилок автоматично відкривається вікно оброблювача повідомлень, у якому відображається список повідомлень про помилку та інформаційних повідомлень, а також відразу дається довідка по виправленню помилки. Натиснувши двічі лівою кнопкою миші на текстовому повідомленні, автоматично буде показане місце, де є помилка. Помилка виділиться червоним кольором.
- Компілятор може працювати у фоновому режимі. Ви можете зменшити до мінімуму вікно компілятора, поки він обробляє проект, і продовжити роботу над іншими файлами проекту. Індикатор під піктограмою зменшеного вікна компілятора дозволяє вам спостерігати за просуванням процесу компіляції, в той час як Ви можете зосередити свою увагу на іншій задачі.

Компілятор системи MAX+plus II обробляє проект, використовуючи наступні модулі та утиліти:

- *Compiler Netlist Extractor* (екстрактор списку ланцюгів), що включає вбудовані програми читання форматів EDIF, VHDL, Verilog і XNF;
- *Database Builder* (будівник бази даних);
- *Logic Synthesizer* (логічний синтезатор);
- *Partitioner* (роздільник);
- *Fitter* (трасувальник);
- *Functional SNF Extractor* (екстрактор для функціонального тестування);
- *Timing SNF Extractor* (екстрактор для тестування часових параметрів);
- *Linked SNF Extractor* (екстрактор для тестування компонування);
- *EDIF Netlist Writer* (програма запису вихідного файлу у формат EDIF);
- *Verilog Netlist Writer* (програма запису вихідного файлу у формат Verilog);
- *VHDL Netlist Writer VHDL* (програма запису вихідного файлу в VHDL);
- *Assembler* (модуль асемблера);
- *Design Doctor Utility* (утиліта діагностики проекту).

Модуль *Compiler Netlist Extractor* (екстрактор форматів) перетворює кожен файл проекту в один чи декілька двійкових файлів з розширенням **.cnf** (*compiler netlist file*). Оскільки компілятор підставляє значення всіх параметрів, що використовуються у параметризованих функціях, вміст файлу **.cnf** може змінюватися при послідовній компіляції, якщо значення параметрів змінюються. Даний модуль створює також файл ієрархічних взаємозв'язків (**.hif**) (*hierarchy interconnect file*), у якому документуються ієрархічні зв'язки між файлами проекту, а також міститься інформація, необхідна для показу ієрархічного дерева проекту у вікні *Hierarchy Display* (рис. 9.52).

Крім того, даний модуль створює файл бази даних вузлів (**.ndb**) (*node database*), у якому містяться імена вузлів проекту для бази даних призначення ресурсів.

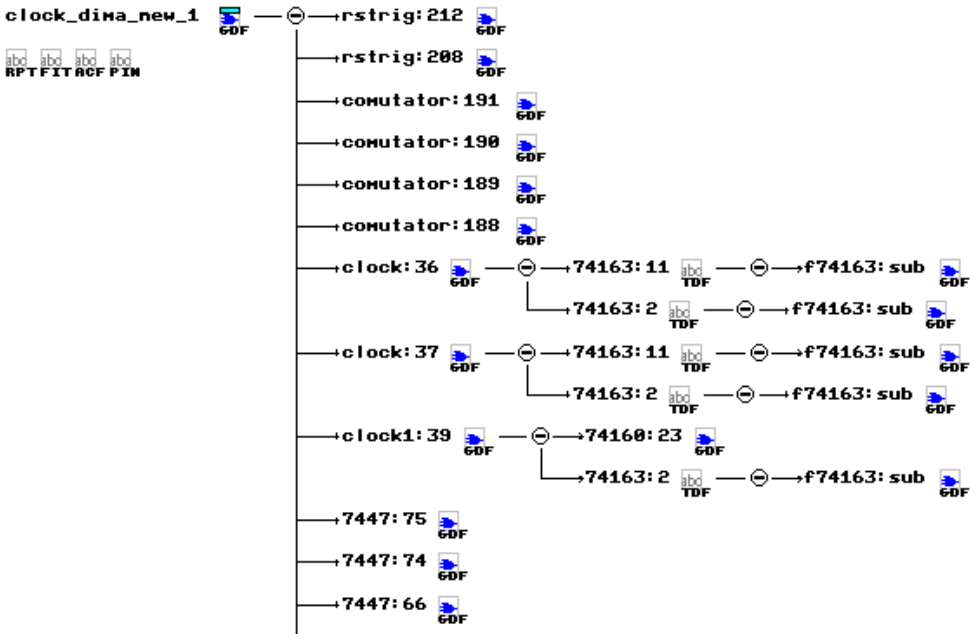


Рис. 9.52

Вбудовані програми читання форматів EDIF, VHDL, Verilog і XNF автоматично транслюють інформацію проекту з файлів відповідних форматів **.edf**, **.vhd**, **.v**, **.xnf**, у формат, сумісний із системою MAX+plus II. Програма читання формату EDIF обробляє вхідні файли EDIF за допомогою бібліотечних файлів **.lmf**, (*library mapping file*) які установлюють відповідність між логічними функціями, розробленими в інших САПР, і функціями системи MAX+plus II. Програма читання формату XNF може створювати файл для експорту текстового дизайну **.tdx** (*text design export file*), що містить інформацію мовою AHDL.

Модуль *Database Builder* (будівник бази даних) використовує файл ієрархічних зв’язків .hif для компонування створених компілятором файлів .cnf, у яких міститься опис проекту. На підставі даних про ієрархічну структуру проекту даний модуль копіює кожен файл .cnf в одну базу даних без ієрархічної структури. Таким чином, ця база даних зберігає електричний зв’язок проекту.

При створенні бази даних модуль досліджує логічну повноту і налагодженість проекту, а також перевіряє наявність синтаксичних помилок (наприклад, вузол без місця призначення або джерела).

На цій стадії компіляції виявляється більшість помилок, що можуть бути відразу легко виправлені. Кожен модуль компілятора послідовно обробляє й оновлює цю базу даних.

Перший раз, коли компілятор обробляє проект, усі файли проекту компілюються. Ви можете використовувати можливість “швидкої повторної компіляції” (*Smart recompile*) для створення розширеної бази даних проекту, що допомагає прискорити наступні компіляції. Ця база даних дозволяє вам змінити призначення ресурсів фізичного пристрою, такі як призначення виводів і логічних елементів, а також повторно компілювати проект без повторної побудови бази даних і повторного синтезу логіки проекту. Використовуючи можливість “повної повторної компіляції” (*Total recompile*), можливо зробити вибір між повторною компіляцією тільки тих файлів, що редагувалися після останньої компіляції, і повною повторною компіляцією всього проекту.

Модуль логічного синтезатора (*Logic Synthesizer*) застосовує ряд алгоритмів, що зменшують використання ресурсів і забирають дубльовану логіку, забезпечуючи тим самим максимально ефективне використання структури логічного елемента для архітектури сім’ї пристроїв. Даний модуль компілятора застосовує також способи логічного синтезу для вимог користувача по часових параметрах. Крім того, логічний синтезатор шукає логіку для несполучених вузлів. Якщо він знаходить неприєднаний вузол, він забирає примітиви, що відносяться до цього вузла.

Для керування логічним синтезом маються три описаних стилі і велике число опцій.

На рис. 9.53 приведено вигляд меню *Processing* компілятора.

Після вибору команди *Report File Setting* з’явиться вікно, показане на рис. 9.54.

Треба поставити мітки навпроти тих пунктів, інформацію про які необхідно відобразити у файлі-рапорті. Можна окремо відображати номери

виводів та тип мікросхеми, ієрархію файлів, з'єднання між осередками, рівняння, або всю цю інформацію разом.

Рис. 9.53

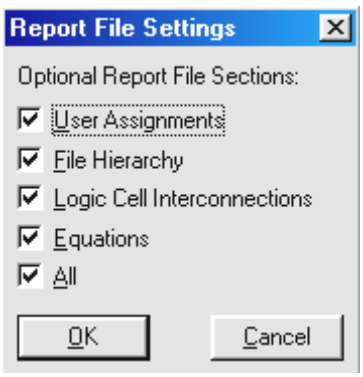


Рис. 9.54

В меню *Options* будь-якого з редакторів є команда *Preferences*. Вибравши її побачимо діалогове вікно, що показане на рис. 9.55. В ньому налаштовуються параметри системи.

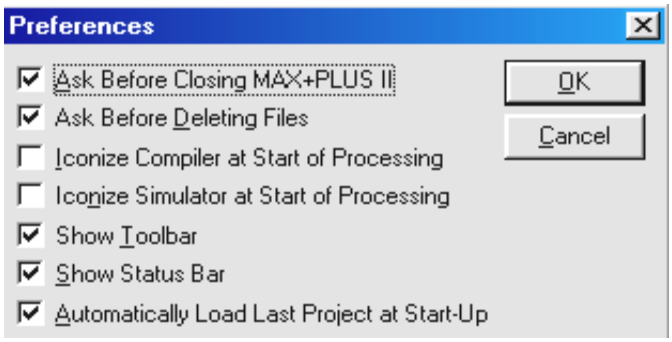


Рис. 9.55

- *Ask Before Closing MAX+plus II* – запит на підтвердження виходу з системи;
- *Ask Before Deleting Files* – підтвердження знищення файлів;
- *Iconize Compiler at Start of Processing* – згортає вікно компілятора при його роботі;
- *Iconize Simulator at Start Processing* – згортає вікно симулятора при його роботі;

- *Show Toolbar* – показує панель інструментів;
- *Show Status Bar* – показує строку стану;
- *Automatically Load Last Project at Start-Up* – автоматично завантажувати останній редагований проект при запуску системи.

З власного досвіду попереджаємо, що третій та четвертий пункти не слід вибирати, оскільки проектант не матиме змоги слідкувати за процесом компіляції.

При розробці великих систем зручніше буде працювати з сіткою. Для того, щоб вона з'явилася, треба з меню *Options* вибрати пункт *Show Guidelines* (рис. 9.56).

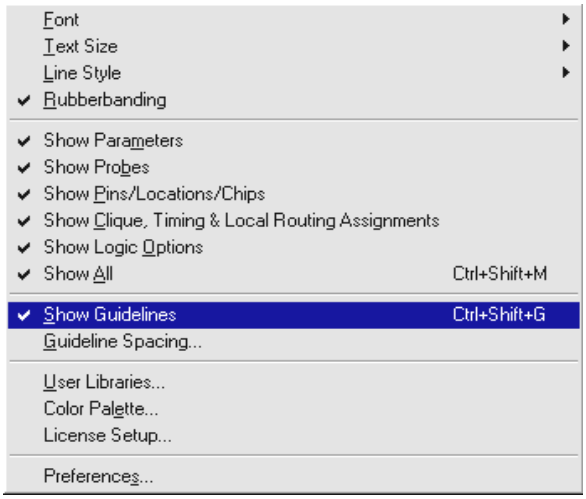


Рис. 9.56

Відстань між лініями встановлюється командою *Guideline Spacing* і вимірюється у відносних одиницях. Діалогове вікно зображене на рис. 9.57.

Користувач може за своїм смаком налаштувати кольори системи. Вибравши команду *Color Palette*, з'явиться вікно, показане на рис. 9.58.

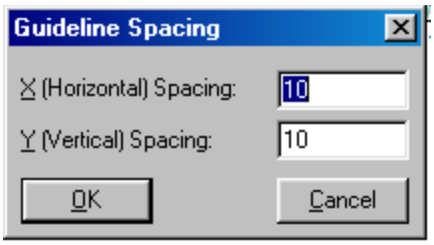


Рис. 9.57

Із меню з лівої частини вікна вибирається елемент, колір якого треба змінити. Потім з гами кольорів вибирається потрібний і натискається кнопка *Preview*. Для того, щоб повернутися до стандартних параметрів, слід натиснути *Reset*

Якщо проект не вміщується при монтажі в один пристрій, модуль *Partitioner* (роздільник) розділяє базу даних, оновлену логічним синтезатором, на ПЛІС того самого сім'ї, намагаючись при цьому розділити проект на мінімально можливе число пристроїв. Розбивка проекту відбувається по границях логічних елементів, а число виводів, які використовуються для з'єднання між пристроями, мінімізується.

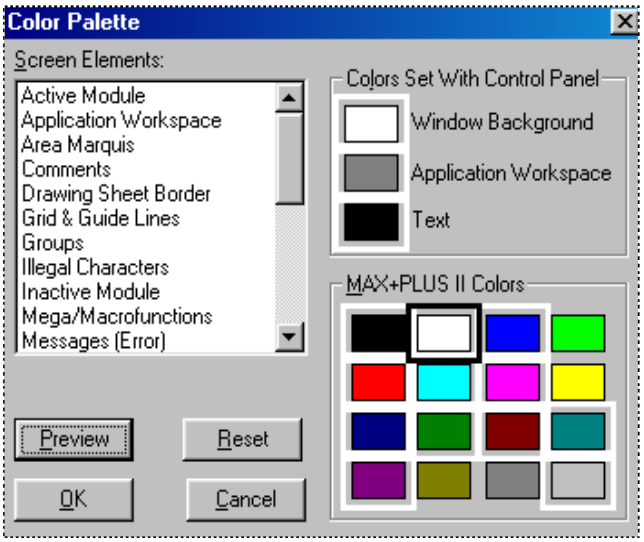


Рис. 9.58

Розбивка може бути проведена цілком автоматично, під частковим керуванням з боку проектанта або цілком під керуванням користувача. Призначення пристроїв і установки для автоматичного вибору пристроїв дозволяють застосувати той рівень керування, що найбільше підходить для конкретного проекту.

Коли працюють модулі *Partitioner* і *Fitter*, Ви можете призупинити компіляцію. Компілятор відобразить інформацію про поточний стан процесів розбивки і трасування кристалу, у тому числі порівняння необхідних і наявних ресурсів. Це потрібно для того, щоб прийняти рішення, чи продовжувати компіляцію, чи вносити кардинальні зміни в проект.

Використовуючи базу даних, оновлену модулем розбивки, модуль трасування (*Fitter*) приводить у відповідність вимоги проекту з ресурсами одного чи декількох пристроїв. Він призначає кожній логічній функції розташування реалізуючого її логічного елемента і вибирає відповідні шляхи взаємних з'єднань і призначення виводів. Даний модуль намагається оптимізувати призначення ресурсів, тобто виводів, логічних елементів, елементів вводу/виводу, комірок пам'яті, чипів, пристроїв, місцевого трасування, часових параметрів. Модуль має параметри, що дозволяють

визначити способи трасування – наприклад, автоматичне введення обмеження на коефіцієнт об’єднання по входу. Якщо трасування не може бути виконане, модуль видає повідомлення і пропонує Вам вибір проігнорувати деякі чи усі призначення або припинити компіляцію.

Незалежно від того, чи завершене повне трасування проекту, даний модуль генерує файл звіту (**.rpt**) (*report file*), у якому документується інформація про розбивку проекту, іменах вхідних і вихідних контактів, часових параметрах проекту і невикористаних ресурсів для кожного пристрою в проекті. Ви можете включити у файл звіту розділи, що показують призначення користувача, файлову ієрархію, взаємні з'єднання логічних елементів і рівняння, реалізовані в логічні елементах.

Отже, підіб’ємо підсумки. Склад програмного забезпечення системи MAX+plus II є повним комплектом, що забезпечує створення цифрових систем на базі ПЛІС фірми Altera із програмованою логікою, у тому числі сім’ї пристроїв Classic, MAX 5000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000 і FLEX 10K. Інформація про інші, підтримувані сім’ї пристроїв фірми Altera приведена у файлі **readme.txt** у системі MAX+plus II.

Система MAX+plus II пропонує повний спектр можливостей логічного дизайну: різноманітні засоби опису схеми для створення проектів з ієрархічною структурою, потужний логічний синтез, компіляцію з заданими часовими параметрами, поділ на частини, функціональне і часове тестування (симуляцію), тестування декількох зв'язаних пристроїв, аналіз часових параметрів системи, автоматичну локалізацію помилок, а також програмування і верифікацію пристроїв. У системі MAX+plus II можна як читати, так і записувати файли мовою AHDL і файли трасування у форматі EDIF, файли на мовах опису апаратури Verilog HDL і VHDL, а також схемні файли OrCAD. Крім того, система MAX+plus II читає файли трасування, створених за допомогою ПЗ Xilinx, і записує файли затримок у форматі SDF для зручності взаємодії з пакетами, що працюють з іншими промисловими стандартами.

В лабораторії мікропроцесорної техніки та мікропроцесорних систем керування, збору і обробки інформації кафедри теоретичної електротехніки та електронних систем Національного університету кораблебудування ім. адмірала Макарова (м. Миколаїв) розроблений комплекс засобів для навчання проектуванню цифрових пристроїв на ПЛІС фірми *ALTERA* з використанням системи MAX+plus II. Комплекс включає:

- методичні матеріали у вигляді опису лабораторного практикуму;
- програмне забезпечення у вигляді пакету MAX+plus II фірми *ALTERA*, який працює на персональних комп’ютерах у середовищі Windows;
- апаратне забезпечення у вигляді плати лабораторного стенду і пристрою типу *ByteBlaster* (рис. 9.59).

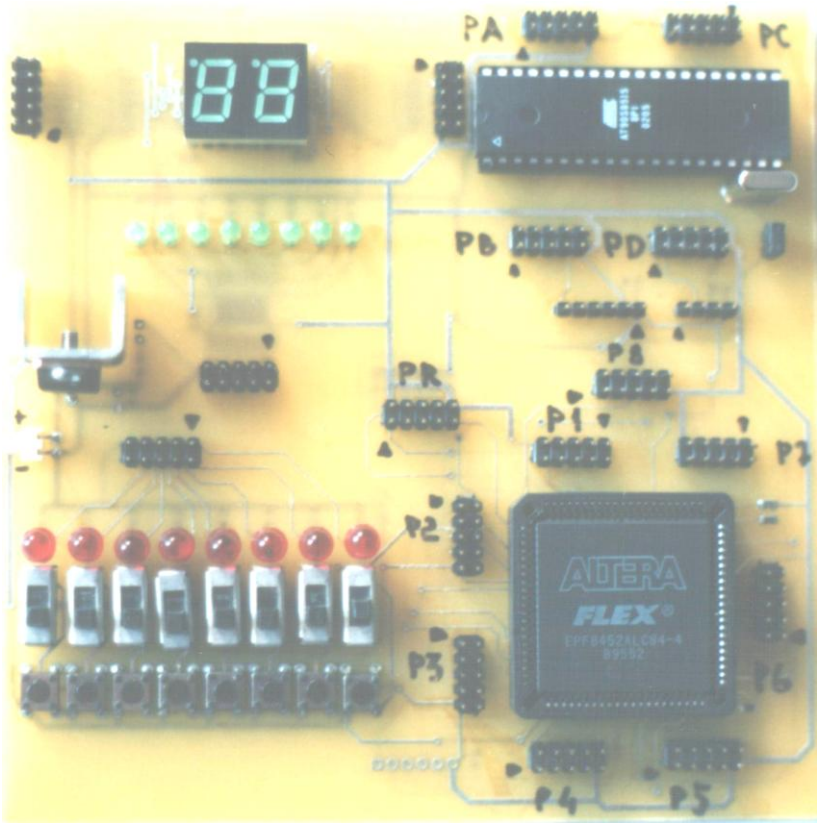


Рис. 9.59

Комплекс може бути встановлений на окремий комп’ютер для індивідуального навчання, а також на локальну мережу комп'ютерів навчального класу.

Лабораторний стенд реалізований на базі ПЛІС EPF8452ALC84-4 широко розповсюдженого сім'ї FLEX8000 фірми *ALTERA*. Ця ПЛІС містить близько 4000 еквівалентних вентилів, має 68 ліній вводу/виводу. У реальних проектах може бути використано до 2500 вентилів і понад 400 тригерів, що цілком достатньо як для виконання навчальних завдань практикуму, так і для практичної роботи фахівців.

Крім ПЛІС, стенд містить пристрої введення і відображення даних, генератор тактових імпульсів частотою 4 МГц. Додаткові роз'єми дозволяють підключати до макету різні модулі розширення.

Конфігурація ПЛІС здійснюється за допомогою персонального комп'ютера через кабель типу *ByteBlaster*, що підключається до стенду. При виконанні практикуму використовується метод послідовного завантаження конфігурації від комп'ютера, на якому встановлена система проектування MAX+plus II. Технологія внутрисистемного програмування ПЛІС дозволяє реалізувати на базі даного стенду широку номенклатуру цифрових пристроїв різної складності, проектування та випробовування яких здійснюється в процесі виконання лабораторних робіт. На стенді розташований мікроконтролер AT90S8515. Таким чином є можливість розробки та реалізації доволі складних цифрових систем з використанням різної сучасної елементної бази та програмних засобів.

9.4.4. Використання стандартних мікросхем в MAX+plus II

З попереднього матеріалу ми знаємо, що для додавання елемента у схему треба двічі натиснути ліву кнопку миші на робочому полі графічного редактора (*Graphic Editor*). При цьому з'явиться вікно, що показане на рис. 9.30. У полі *Symbol Name* треба ввести ім'я потрібного символу, або вибрати з меню *Symbol Files*. Разом з системою в каталог /*maxplus*2/*max*2*lib* встановлюються бібліотеки розроблених фірмою *ALTERA* примітивів. Яка саме з бібліотек буде

використовуватися, користувач встановлює шляхом вибору з меню *Directories* відповідної папки. Мається три бібліотеки: *примітивів*, *мегафункцій* та *макрофункцій*. У бібліотеці примітивів мається широкий набір базових логічних елементів, починаючи від інверторів та повторювачів і закінчуючи тригерами різних видів. Досвідчені користувачі можуть вибирати елементи, вводячи їх назви і тим самим витрачати менше часу на проектування системи. Найчастіше використовуються елементи *input* (вхід), *output* (вихід), *vcc* (живлення) *gnd* (земля) та інші. Більш складні типи знаходяться в бібліотеках макро- та мегафункцій. Вони мають виводи для підключення джерела живлення та землі. Базові логічні елементи таких виводів не мають. При проектуванні цифрових систем вважається, що до них підведене живлення.

Як приклад, на рис. 9.60. приводяться умовні зображення декількох типів мультиплексорів, що є в бібліотеці мегафункцій MAX+plus II.

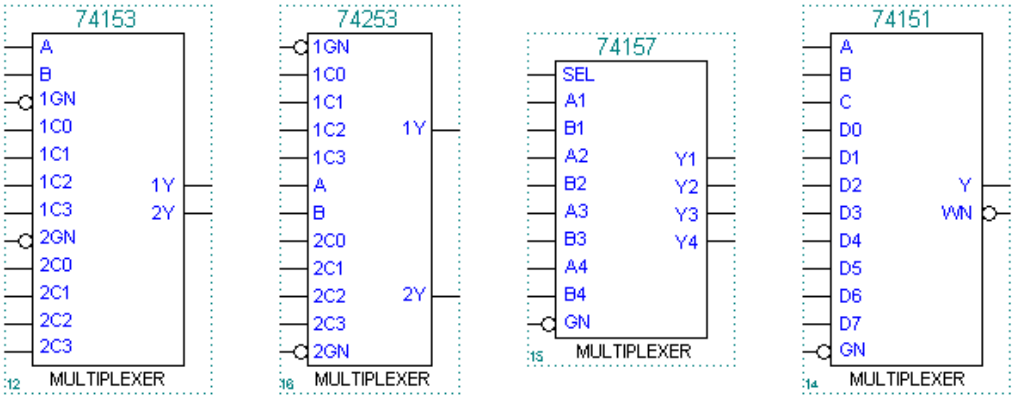


Рис. 9.60

Використання цих мікросхем нічим не відрізняється від прикладів, розглянутих вище. Якщо необхідно отримати допоміжну інформацію про роботу мікросхеми, що планується до використання, то подвійне натискання лівої кнопки миші дає можливість отримати таблицю станів. Приклад такої таблиці для мультиплексора 74153 приводиться у табл. 9.5.

Таблиця 9.5

Inputs							Outputs
Select*		Data				Enable	
B	A	C0	C1	C2	C3	GN	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

На рис. 9.61 приводиться приклад схеми з використанням мультиплексора 74153 для реалізації логічної функції:

$$y = x_0\ x_1\ \overline{x_3}\ x_4 + \overline{x_0}\ \overline{x_3}\ x_4 + \overline{x_0}\ \overline{x_2}\ x_3 + x_2\overline{x_3} + \overline{x_1}\ x_2\,.$$

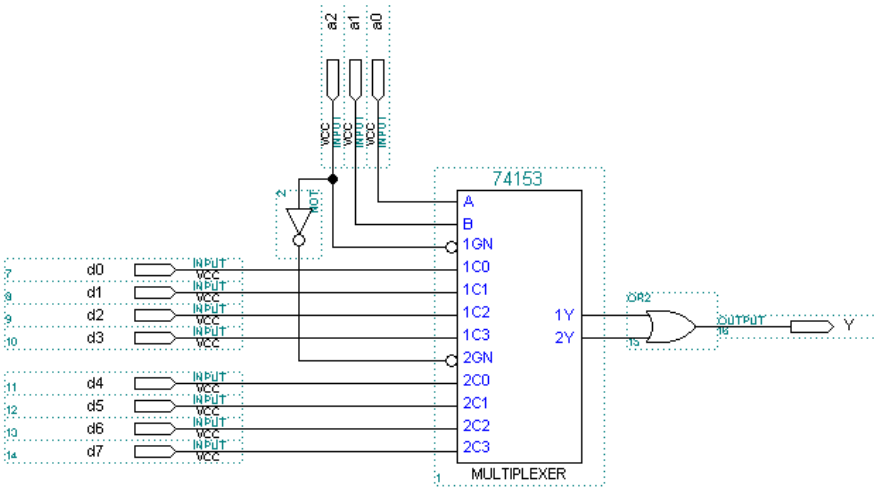


Рис. 9.61

Бібліотеки MAX+plus II містять широку номенклатуру дешифраторів (дешифраторів-демультиплексорів) з тих мікросхем, що серійно виготовляються рядом фірм.

У бібліотеці системи містяться елементи різного призначення. Іноді важко швидко знайти потрібний елемент, особливо якщо не відома його назва а є лише функціональне призначення. В такому випадку слід з меню *Help* вибрати команду *Old-Style Macrofunctions*. Зі списку *Macrofunction Categories* вибрати потрібну категорію. З'явиться перелік елементів: номенклатура та призначення.

Суматори в MAX+plus II можуть створюватись на основі базових елементів (наприклад, напівсуматора, повного однорозрядного суматора), так і на основі моделей серійних мікросхем. На рис. 9.62 приводиться приклад побудови послідовного суматора. Для побудови багато розрядних арифметичних пристроїв можна використовувати шинну архітектуру. Для цього необхідно створити шину. Для створення шини слід з контекстного меню вибрати пункт *Line Style >> Solid* Приклад побудови пристрою для виконання операцій додавання та віднімання приводиться на рис. 9.63.

Тип операції що виконується задається рівнем сигналу на вході *P*.

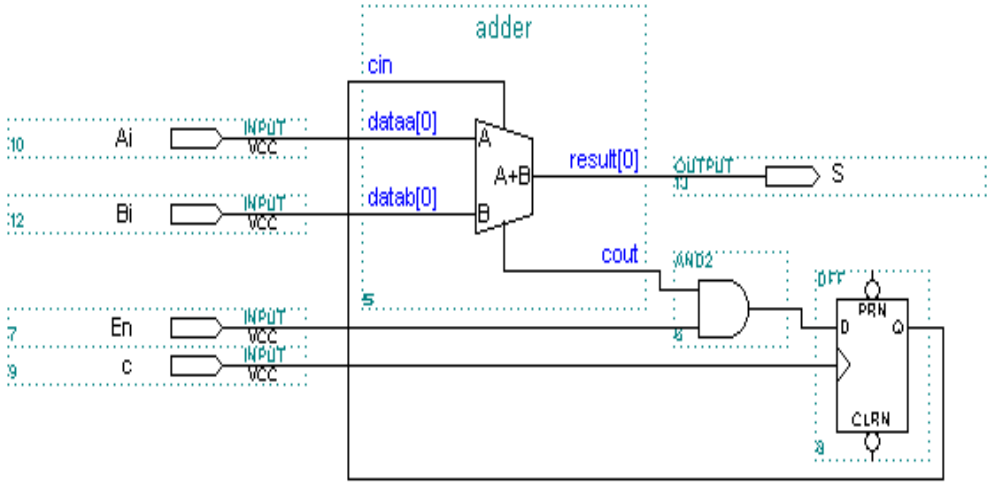


Рис. 9.62

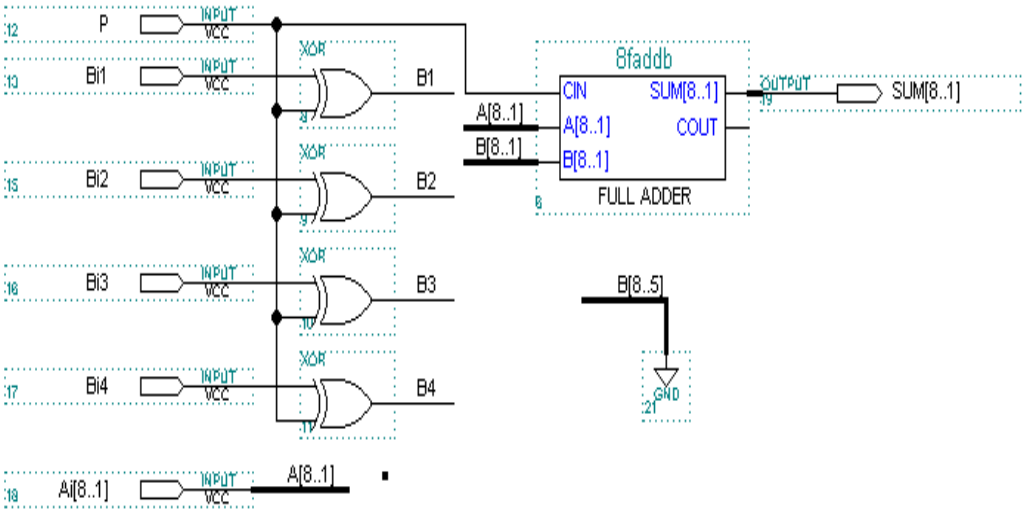


Рис. 9.63

Використовуючи *Timing Analizer*, який запускається шляхом вибору однойменної команди з головного меню системи можна проаналізувати часові затримки, що виникають при виконанні арифметичних операцій. Запустивши його через команду *Start*, отримаємо значення затримок, що приведені у табл. 9.6.

Таблиця 9.6

	SUM2	SUM3	SUM4	SUM5	SUM6	SUM7	SUM8
A1	13.7ns	16.6ns	19.8ns	23.3ns	23.6ns	25.5ns	25.2ns
A2	8.4ns	11.3ns	14.5ns	18.0ns	18.3ns	20.2ns	19.9ns
A3		8.4ns	11.6ns	15.1ns	15.4ns	17.3ns	17.0ns
A4			9.0ns	12.5ns	12.8ns	14.7ns	14.4ns
A5				10.9ns	11.0ns	13.4ns	13.1ns
A6					11.0ns	13.4ns	13.1ns
A7						8.7ns	8.4ns
A8							8.6ns
Bi1	13.7ns	16.6ns	19.8ns	23.3ns	23.6ns	25.5ns	25.2ns
Bi2	8.4ns	11.3ns	14.5ns	18.0ns	18.3ns	20.2ns	19.9ns
Bi3		8.7ns	11.9ns	15.4ns	15.7ns	17.6ns	17.3ns
Bi4			9.0ns	12.5ns	12.8ns	14.7ns	14.4ns
P	8.1ns/11.0ns	8.1ns/13.9ns	8.4ns/17.1ns	11.9ns/20.6ns	12.2ns/20.9ns	14.1ns/22.8ns	13.8ns/22.5ns

У бібліотеці MAX+plus II є декілька типів компараторів (7485, 74518, 74518B, 74684, 74686, 74688), арифметично-логічні пристрої (74181, 74381, 74382), а також перемножувачі (7497, 74261, 74284, 74285).

Схеми тригерів у середовищі MAX+plus II можна використовувати стандартні, або розробляти свої на основі базових логічних елементів, а потім створюючи відповідний символ.

Для створення символу слід з меню MAX+plus II вибрати пункт *Symbol Editor*. Відчиниться вікно редактора символів. Панель інструментів має ті ж самі кнопки, що і в графічному редакторі. Їх призначення детально описане в попередньому розділі.

Вигляд символу показаний на рис. 9.64.

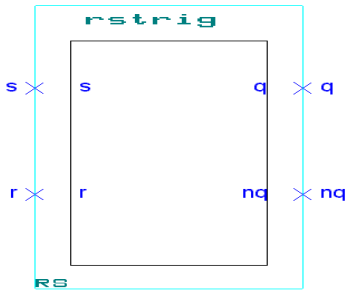


Рис. 9.64

Навівши курсор миші на вертикальну блакитну лінію (зліва), двічі натиснути ліву кнопку. З’явиться вікно, показане на рис. 9.65.

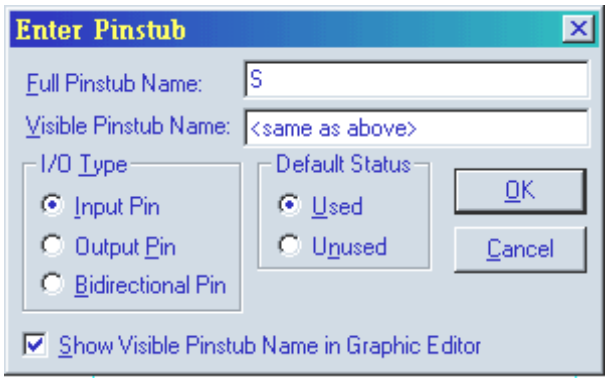


Рис. 9.65

У полі *Full Pinstub Name* слід ввести назву виводу на схемі у графічному редакторі. Треба слідкувати за тим, щоб назви виводів співпадали з тими, що є у графічному редакторі. Необхідно встановити тип виводу: вхідний, вихідний чи двонаправлений. Після цього натиснути кнопку ОК.

На рис. 9.66 показано вигляд символу *RS*-тригера після того, як всі виводи призначені.

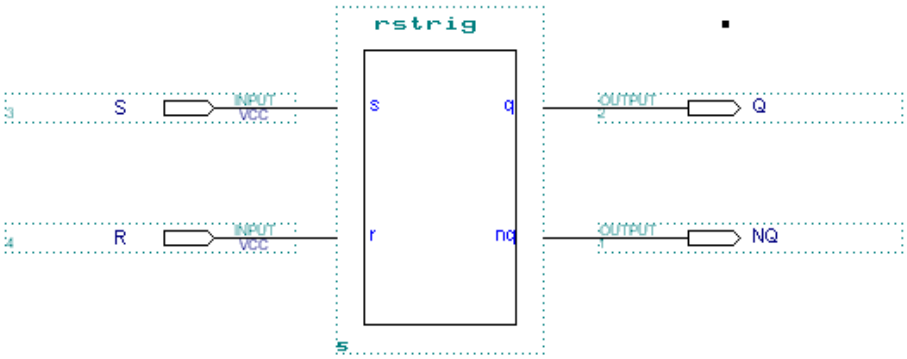


Рис. 9.66

Замість *SYM_NAME* слід ввести назву символу, яка відображала б його призначення. Це дозволить швидко знаходити символ і запобігти плутанини.

Таким же шляхом можна створювати інші функціональні блоки цифрових пристроїв – наприклад, спеціалізовані регістри, лічильники, модулі пам'яті і т. п.

Стандартні модулі ЦІС, що використовуються в MAX+plus II приводяться у відповідних додатках.

Для отримання таблиці станів елементів, що використовуються, необхідно натиснути на піктограмі *HELP* панелі інструментів, а потім натиснути один раз лівою кнопкою миші не елементі про який потрібна інформація.

При роботі з лічильниками та регістрами користуватись приведеними вище часовими діаграмами не зовсім зручно, оскільки використовується велика кількість виводів. Тому в таких випадках для багато розрядних цифрових пристроїв зручніше використовувати діаграми шин. Часова діаграма буде мати вигляд, приведений на рис. 9.67.

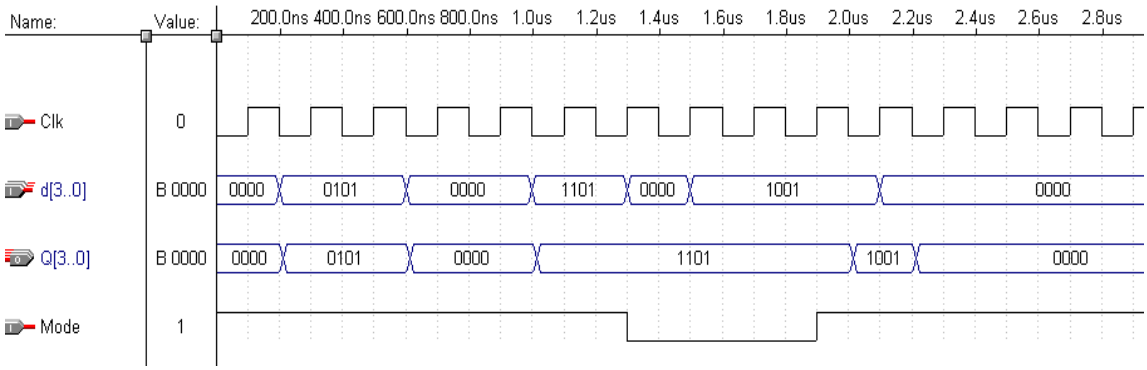


Рис. 9.67

Для створення шинної системи часових діаграм у сигнальному редакторі (*Waveform editor*) у полі *Node Name* діалогового вікна *Insert Node* вводиться назва виводів та у квадратних дужках номери. В схемі, створеній у графічному редакторі (*Graphic Editor*) виводи мають назви *D*0, *D*1, *D*2, *D*3 – вхідні; *Q*0, *Q*1, *Q*2, *Q*3 – вихідні. Для того, щоб зробити шину, необхідно у полі *Node Name* ввести назву *D*[3..0] або *Q*[3..0]. В лівій частині робочого поля редактора

з’явиться зображення виводів. Їх буде декілька, розташованих один за одним на відміну від одного виводу. Регістр букв не має значення в назвах виводів. Великі та маленькі букви трактуються компілятором як однакові. При роботі з шинами даних в сигнальному редакторі на панелі інструментів стають активними ще дві кнопки: *Overwrites a single selected node or group with a specified count sequence* та *Overwrites a single selected node or group with a different logic level*. За допомогою цих інструментів можна задати зміну сигналу на виводі або шині у відповідній послідовності та задати визначений код на шині у потрібному проміжку часу. Двічі натиснувши ліву кнопку миші на зображенні виводу, побачимо діалогове вікно, що зображене на рис. 9.68.

За допомогою перемикачів у групі *Radix* можна вказати в якому коді буде відображатися код:

- *BIN* – двійковому;
- *DEC* – десятковому;
- *OCT* – вісімковому;
- *HEX* – шістнадцятковому.

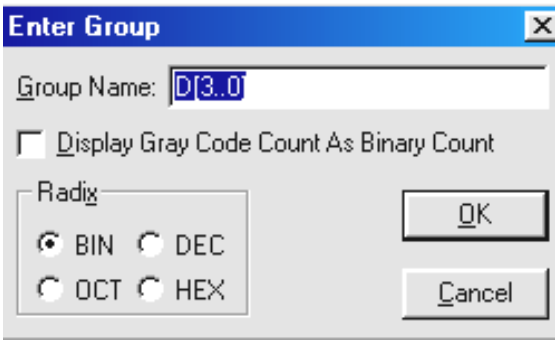


Рис. 9.68

Встановивши позначку *Display Gray Code Count As Binary Count*, в якості двійкового коду відображатиметься код Грея.

Розглянемо діалогове вікно, що з’явиться при натисканні на кнопку *Overwrites a single selected node or group with a specified count sequence*, яка має зображення букви С.

На рис. 9.69 зображено вигляд вікна *Overwrite Count Value*.

У полі *Starting Value* вводиться початковий код на виводі. Перемикання позначок у групі *Count Type* визначає тип коду:

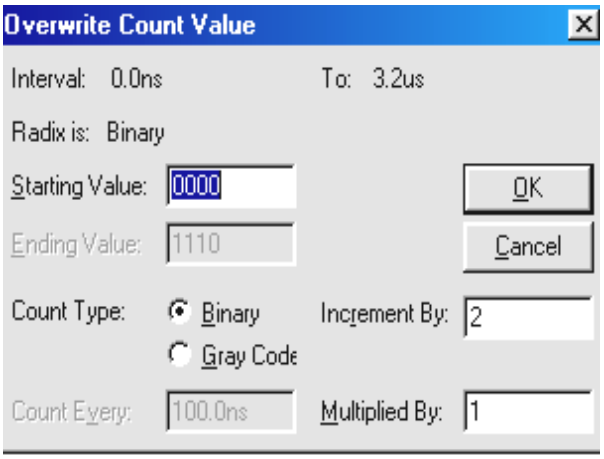


Рис. 9.69

двійковий чи код Грея. У полях *Increment By* та *Multiplied By* вводяться значення доданка чи множника. Саме значеннями в цих полях визначається алгоритм послідовності зміни сигналу.

9.4.5. Використання нестандартних елементів в MAX+plus II

У системі MAX+plus II є можливість створювати власні елементи. Розглянемо створення мультиплексорів за допомогою *MegaWizard Plug-In Manager*. Для початку роботи з ним потрібно з меню *File* вибрати відповідну команду. У першому діалоговому вікні буде запропоновано створити власний символ (мегафункцію), або символ на базі існуючого. Після натиснення кнопки *Next* з'явиться наступне діалогове вікно. в якому потрібно вибрати тип створюваного елемента. Мультиплексори (*LPM_MUX*) розташовані у підкаталозі *gates*. Користувачу пропонується вибрати мову опису апаратури, на якій буде описаний створюваний елемент. Не слід забувати вказувати місце розташування нового елементу та його назву. Після цього натиснути кнопку *Next*.

На рис. 9.70 показане діалогове вікно *MegaWizard Plug-In Manager* [*page* 2a].

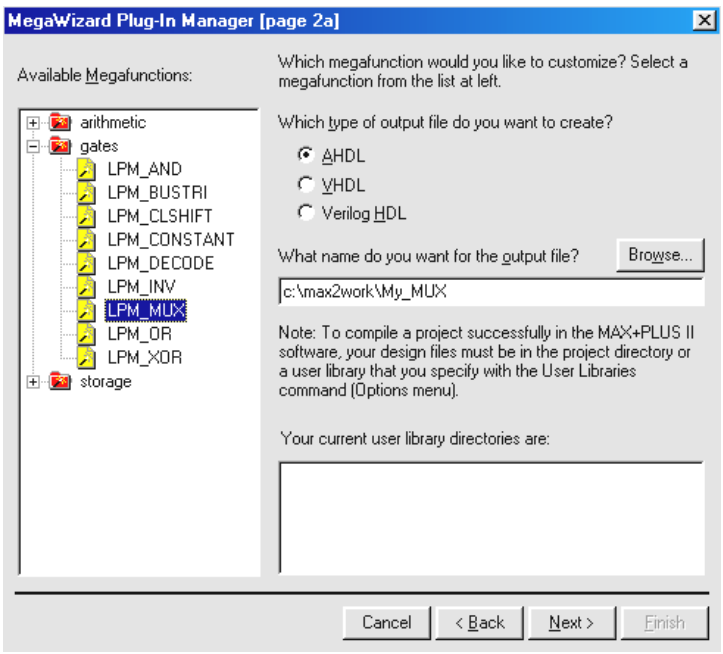


Рис. 9.70

У наступному діалоговому вікні (рис. 9.71) користувач має можливість безпосередньо виконати всі необхідні налаштовування нового елемента.

How many 'data' inputs do you want ? – потрібно вказати кількість входів даних.

How wide should the 'data' input....? – вказується ширина кожного входу (тобто розрядність шини даних). а також ширина вихідної шини.

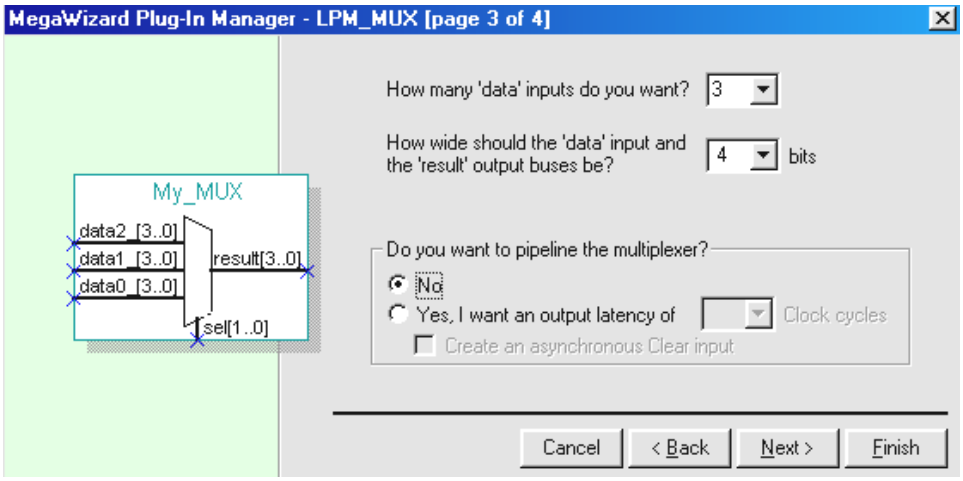


Рис. 9.71

Є можливість тактувати мультиплексор. Вхід тактування можна добавити, встановивши перемикач у положення *Yes, I want an output latency of*. Навпроти встановлюється кількість тактів генератора, після яких вхідні дані будуть передані на вихід в залежності від стану адресних входів. Натискання кнопки *Next* призведе до появлення наступного діалогового вікна, в якому міститься інформація про файли, які будуть створені, та місце їх розташування. Натискання кнопки *Finish* завершує роботу *MegaWizard Plug-In Manager*.

Як бачимо, створення нових елементів не є проблемою. Користувач може задати кількість входів даних до 256, а розрядність кожного з них також може бути до 256, тобто створювати мультиплексори шин даних. Отже реально проблема нарощення розрядності мультиплексорів не виникає.

Розглянемо тепер особливості створення дешифраторів із нестандартними характеристиками за допомогою *MegaWizard Plug-In Manager*. На початку процедура створення нового елемента однакова для всіх елементів.

Дешифратори знаходяться в підкаталозі *gates*, як і мультиплексори.

У діалоговому вікні *MegaWizard Plug-In Manager – LPM_DECODE* [*page* 3 of 6], яке показане на рис. 9.72, користувач повинен вказати кількість входів у полі *How wide should be 'data' input bus*. Найбільша кількість входів дорівнює 8. Встановлення перемикачу *Create an Enable Input* додає до елементу вхід дозволу.

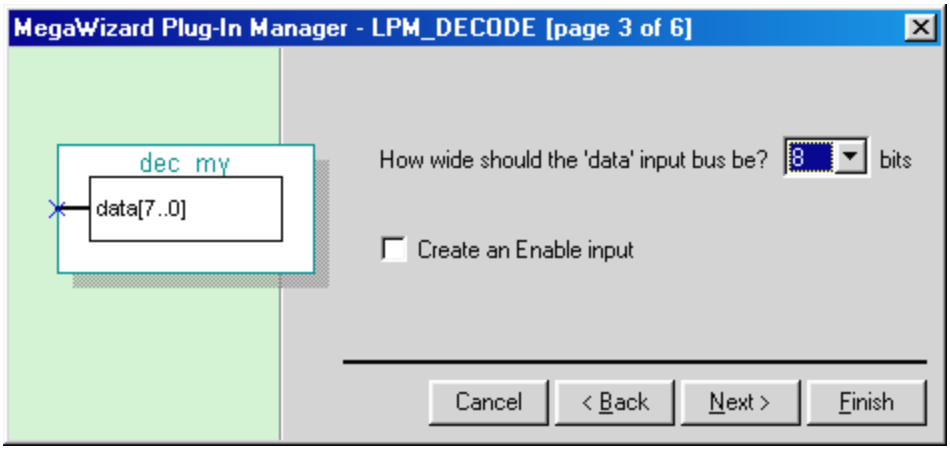


Рис. 9.72

У діалоговому вікні, що зображене на рис. 9.73, користувач вказує кількість виходів дешифратора. Максимальна кількість виходів дорівнює 256. У групі *Radix* встановлюється система числення – десяткова (*decimal*) або шістнадцяткова (*hex*).

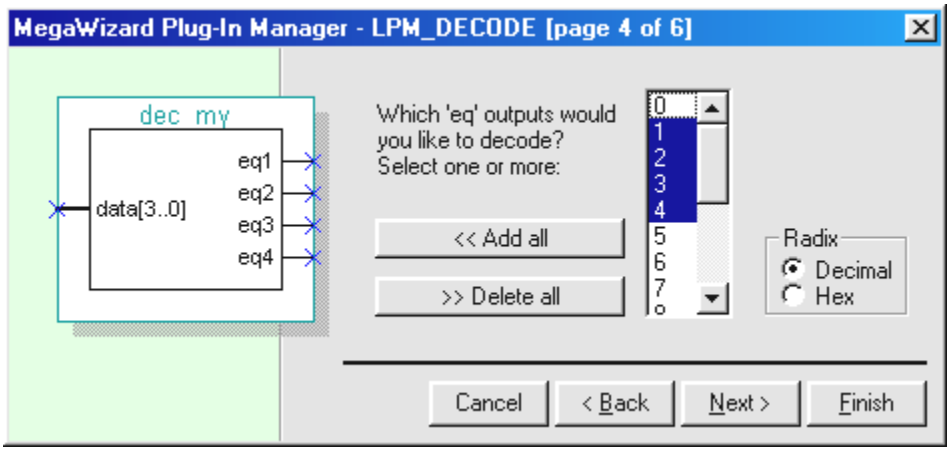


Рис. 9.73

Вже на цьому етапі можна завершити створення елемента дешифратора.

Можна створювати елементи перемножувачів, використовуючи *MegaWizard Plug-In Manager*. На рис.9.74 приведено схему використання

перемножувача двох напівбайтових слів, розробленого *MegaWizard Plug-In Manager*.

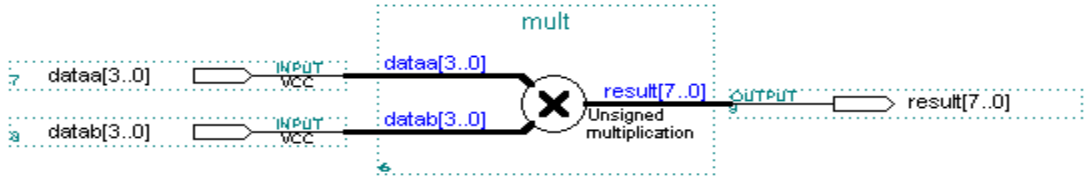


Рис. 9.74

Часова діаграма (рис. 9.75) демонструє роботу схеми. Як бачимо, при зміні вхідного коду, на виході виникають перешкоди, які обумовлені часовими затримками.

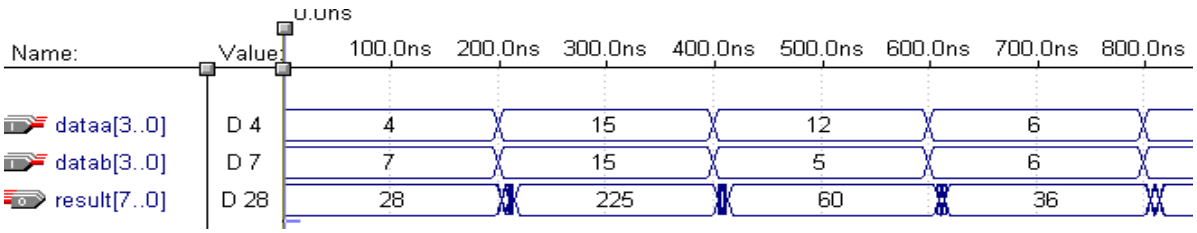


Рис. 9.75

Розглянемо тепер послідовність створення елементів пам’яті з використанням *Mega-Wizard Plug-In Manager*.

Елементи пам’яті містяться у бібліотеці \maxplus2\max2lib\mega_lpm. Розробник може використати ОЗП з окремими шинами адреси для запису та читання інформації, з загальною шиною адреси окремими вхідною та вихідною шинами даних і загальною адресною шиною та двонаправленою шиною даних, а також ПЗП. При використанні елементів пам’яті з бібліотеки системи розробник може використовувати коди з довільною кількістю розрядів, адже їх кількість не регламентується. На елементах пам’яті є виводи дозволу запису і дозволу читання, сигнали тактування, вхід дозволу тактування, адресні шини та шини даних. Звичайно, при розробці цифрової апаратури, може знадобитися елемент пам’яті, в якому будуть непотрібними деякі з виводів. Тому в системі є модуль *Mega-Wizard Plug-In Manager*, який дозволяє створити майже будь-який елемент цифрової електроніки, описаний на одній з мов: AHDL, VHDL та Verilog HDL. Для запуску *Mega-Wizard Plug-In Manager* потрібно з меню *File* вибрати команду з такою ж назвою. Перше діалогове вікно пропонує вибрати

наступну дію: створити нову користувальницьку мегафункцію або виправити вже існуючу мегафункцію. При натисканні кнопки *Next* з'явиться діалогове вікно, що зображене на рис. 9.76 (Page 2).

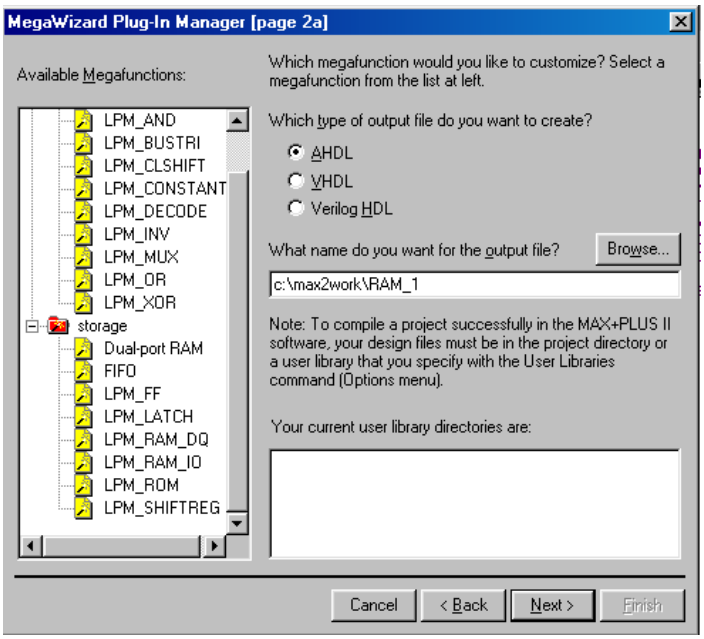


Рис. 9.76

У лівій частині діалогового вікна пропонується вибрати мегафункцію, яка буде створюватися. Елементи пам'яті можна знайти у розділі *storage*.

Розробнику доступні *Dual-port RAM* – пам'ять з роздільними шиною даних для запису та читання інформації та окремими шинами адресу, *LPM_RAM_DQ* пам'ять з загальною адресною шиною, *LPM_RAM_IO* – пам'ять з двонаправленою шиною даних та загальною шиною адресу, *LPM_ROM* – постійний запам'ятовуючий пристрій. У правій частині діалогового вікна пропонується вибір мови, на якій буде описано розроблюваний елемент. Після завершення створення елемента системою буде автоматично згенерований код, який описує функції мікросхеми. Трохи нижче у полі необхідно вказати місце розташування на жорсткому диску комп'ютера вихідного файлу, а також вказати його назву. Саме з такою назвою буде створений відповідний елемент, тому бажано давати не випадкові назви, а такі, що мають зміст про призначення елемента або виконувані ним функції. Місце розташування файлу можна вказати, набравши його на клавіатурі, або натиснувши кнопку *Browse* вибрати потрібну директорію. Після натискання

кнопки *Next*, яка розташована в нижній частині діалогового вікна, з'явиться наступне діалогове вікно, вміст якого буде визначатися типом створюваного елемента (мегафункції).

Розглянемо третій етап створення елемента пам'яті на прикладі *LPM_RAM_DQ*. Для елементів ОЗП (RAM) вміст цього вікна (рис. 9.77, *Page* 3) дещо відрізняється, але більшість параметрів схожі.

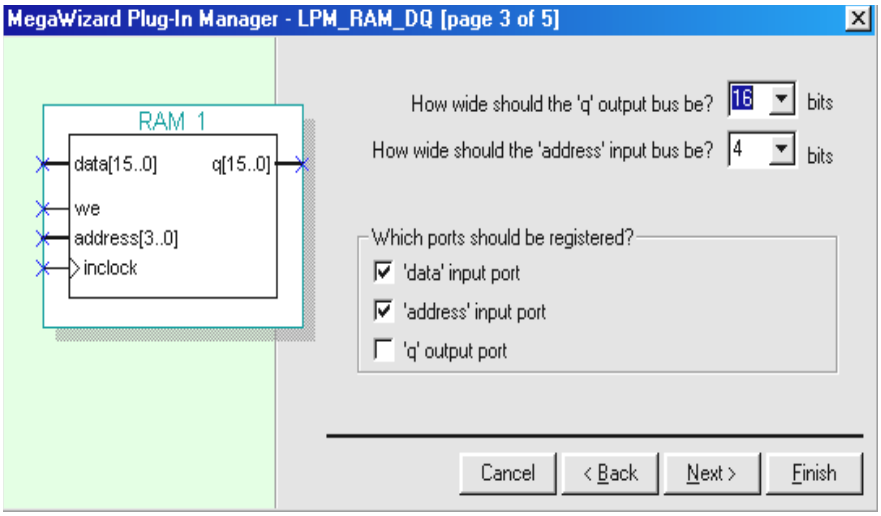


Рис. 9.77

На третьому етапі користувач має можливість визначити розрядність шини даних та розрядність адресної шини. Ці значення вказуються у полях *How wide should the 'q' output bus be* та *How wide the ' adress' input bus be* відповідно. У розділі *Which ports should be registered*, який розташований у нижній частині діалогового вікна, вказується, які додаткові виводи повинен мати розроблюваний елемент пам'яті. Встановлення позначок *'data' input port* та *'address' input port* визначають наявність в елементі входу для сигналів тактування. Встановивши їх в лівій частині діалогового вікна, на зображенні елемента, можна побачити цей вивід. Позначка *'q' output port* додає в елемент вивід для сигналів тактування вихідного коду. Максимальна довжина слів складає 256 бітів. Максимальна розрядність шини адреса складає 16 біт. В елементі пам'яті також наявний вхід дозволу на запис.

На рис. 9.78, *Page* 4 зображене наступне діалогове вікно.

На цьому етапі вказується, яка саме інформація зберігатиметься у пам'яті після програмування мікросхеми. Її можна залишити пустою або вказати, яка

саме інформація у ній зберігатиметься. Як вже згадувалося, інформація про вміст пам'яті міститься у файлі з розширенням **.mif** або **.hex.**

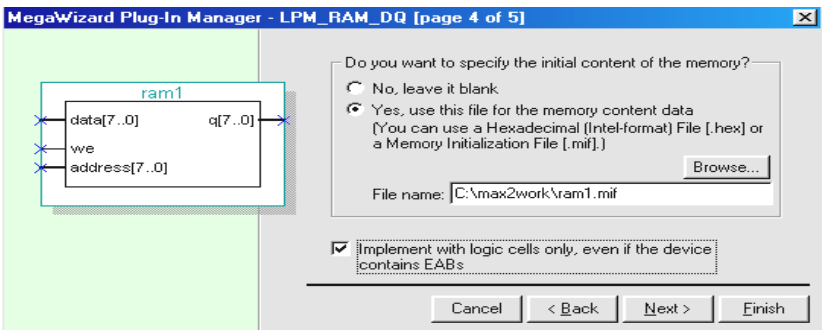


Рис. 9.78

Натискання кнопки *Next* дозволяє перейти до останнього етапу створення елементу. На цьому етапі вже не потрібно задавати параметри. Майстер створення елементів повідомляє про створення потрібних файлів та вказує їх знаходження на диску. Тепер створений елемент можна використовувати в будь-якій схемі. Двічі натиснувши ліву кнопку миші на робочому полі графічного редактора, у діалоговому вікні, яке з'явиться, потрібно відшукати створений елемент і добавити його в схему.

Розглянемо етапи створення елемента постійної пам'яті. У розділі *storage* необхідно вибрати *LPM_ROM*. Не слід забувати вказувати назву створюваного символу, інакше буде неможливо перейти до наступного етапу. Процес створення цього елемента схожий з процесом створення елемента ОЗП. У наступному діалоговому вікні вказується розрядність шини даних та адресної шини, наявність входів для сигналу тактування адреса та вихідних даних. При необхідності їх можна убрати. У третьому діалоговому вікні вже необхідно вказати файл, в якому описуватиметься вміст кожної комірки пам'яті – адже інформація у ПЗП заноситься один раз і назавжди на етапі виготовлення або самим користувачем. Процес створення елементу ПЗП можна порівняти з процесом виготовлення самої мікросхеми.

Синтаксис файлу з розширенням **.mif** приведений нижче:

```
DEPTH = 32;  % Кількість потрібних комірок пам'яті %
WIDTH = 14;  % Розрядність даних %
```

```
ADDRESS_RADIX = HEX; % Формат адреси: BIN, DEC, HEX або OCT %
DATA_RADIX = HEX;    % Формат представлення даних %
CONTENT
  BEGIN
[0..F] : 3FFF;       % Діапазон – кожна адреса від 0 до F = 3FFF %
    6 : F;           % Одинична адреса – адреса 6 = F %
    8 : F E 5;       % початок діапазону з визначеної адреси %
  END ;              % Addr[8] = F, Addr[9] = E, Addr[A] = 5 %
```

Такий файл можна створити за допомогою будь-якого текстового редактора або скористатися *Text Editor* системи MAX+plus II.

Приклад 9.1. Реалізувати наступну логічну функцію:

$$y = x_3\ x_2\ x_1\ x_0 + \overline{x_3}\ x_2\ x_1\ x_0 + x_3\ x_2\ \overline{x_1}\ x_0 + \overline{x_3}\ x_2\ \overline{x_1}\ x_0.$$

Розв'язання. Функція містить чотири змінні. Отже, нам знадобиться елемент ПЗП, який матиме чотири адресних входи та двохрозрядну шину даних (прямий та інверсний вихід).

Створимо такий елемент за допомогою *Mega-Wizard Plug-In Manager*. Карта прошивки ПЗП приведена у табл. 9.7.

Карта прошивки ПЗП для реалізації логічної функції

Таблиця 9.7

Адреса	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	2	2	2	2	1	2	2	1	2	1	2	2	2	2	1	1

Файл ініціалізації пам'яті матиме наступний вигляд:

```
DEPTH = 16;
WIDTH = 2;
ADDRESS_RADIX = BIN;
DATA_RADIX = BIN;
```

```
CONTENT
    BEGIN
  0000:   10; 0001:   10; 0010:   10;
  0011:   10; 0100:   01; 0101:   10;
  0110:   10; 0111:   01; 1000:   10;
  1001:   01; 1010:   10; 1011:   10;
  1100:   10; 1101:   10; 1110:   01;
  1111:   01;
   END ;
```

На наш погляд, для зручності у даному випадку краще було представити карту прошивки у двійковій формі, хоча від цього робота схеми та виконувані нею функції не змінилися.

На рис. 9.79 зображена схема реалізації логічної функції за допомогою ПЗП.

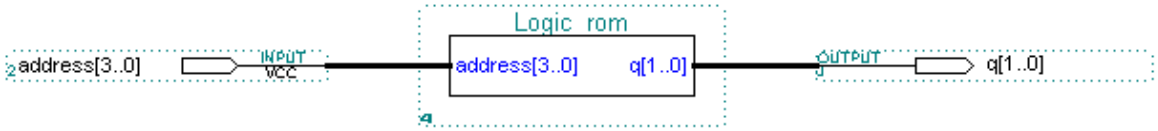


Рис. 9.79

У системі MAX+plus II є можливість створювати елементи пам’яті з різним об’ємом (від декількох байт до декількох мегабайт), з різною кількістю розрядів (1, 4, 8, 16 і т. д.), з різними методами керування. У кожному конкретному випадку потрібно підбирати оптимальну пам’ять, що в найбільшій мірі відповідає вимогам розв’язуваної задачі.

При створенні елементів ОЗП у системі MAX+plus II можна вказати потрібну кількість розрядів адресної шини (максимум – 256) та шини даних (максимум – 16). Для збільшення кількості адресних розрядів використовуються ті ж методи, що і у випадку ПЗП.

На рис. 9.80 зображена схема використання елемента пам’яті з загальною шиною адреса та розділеними шинами даних.

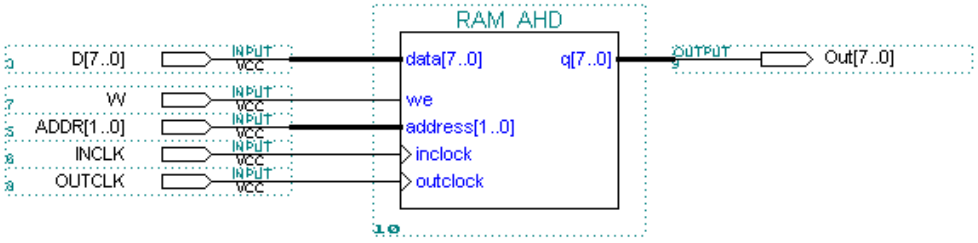


Рис. 9.80

Подібно до описаного вище, можна створювати і пристрої пам'яті типу *FIFO* та типу *LIFO*. Необхідно пам'ятати лише, що для пам'яті *FIFO* потрібно зберігання двох кодів адреси (адреса для запису та адреса для зчитування), а для пам'яті *LIFO* достатньо лише одного коду адреси.

КОНТРОЛЬНІ ПИТАННЯ

1. Чим обумовлена необхідність використання ПЛІС?
2. Поясніть узагальнену структуру програмованих логічних матриць та призначення кожного з блоків.
3. Дайте характеристику основним параметрам ПЛМ.
4. Поясніть, як на основі ПЛМ можуть бути реалізовані пристрої комбінаційної схемотехніки.
5. Чим відрізняється ПЛМ від ПМЛ?
6. Дайте характеристику БМК. Поясніть їх призначення.
7. Дайте коротку характеристику типам сучасних ПЛІС (*CPLD*, *FPGA*, *FLEX*).
8. Які типи ключів використовуються в сучасних ПЛІС? Дайте їм коротку характеристику.
9. Перелічіть основні типи блоків *FPGA* та поясніть та поясніть їх призначення.
10. Що є суттєвою відмінністю ПЛІС останнього покоління порівняно з попередніми?

11. Дайте визначення терміну *“інтерфейс JTAG”*.

12. Дайте визначення терміну *“периферійне сканування”*.

13. Які критерії використовуються розробником сучасної електронної системи при виборі ПЛІС?

14. Перелічіть основні елементи функціональної схеми ПЛІС.

15. Перелічіть типи глобальних кіл ПЛІС.

16. Якими параметрами характеризуються ПЛІС з точки зору їх ємності?

17. Які типи затримок ПЛІС у розповсюдженні сигналів необхідно знати розробнику електронних систем на їх базі?

18. Дайте загальну характеристику САПР MAX+plus II.

19. Приведіть послідовність створення нового проекту в САПР MAX+plus II.

20. Поясніть призначення прикладних програм, назви яких відображені у вікні, приведеному на рис. 9.11.

21. Дайте визначення терміну *“файл проекту”*. Чим він відрізняється від інших файлів, створюваних у САПР MAX+plus II?

22. Які типи файлів проекту Вам відомі?

23. Чим відрізняються між собою бібліотеки елементів? Приведіть їх перелік.

24. Які можливості має сигнальний редактор?

25. Які типи нестандартних комбінаційних пристроїв можуть бути реалізовані на базі ПЛІС за допомогою САПР MAX+plus II?

26. Які типи нестандартних пристроїв пам’яті можуть бути реалізовані на базі ПЛІС за допомогою САПР MAX+plus II?

Розділ 10

ІМПУЛЬСНІ ПРИСТРОЇ НА БАЗІ ЦИФРОВИХ ІНТЕГРАЛЬНИХ СХЕМ

10.1. Пристрої формування імпульсів

Під терміном *"формування імпульсів"* мається на увазі перетворення аналогових або цифрових сигналів з метою отримання їх часових характеристик відповідно до поставлених вимог.

Оскільки основними часовими параметрами імпульсів, що описані в **Розділі 2**, є тривалості фронту та зрізу, тривалості імпульсу та паузи, то в цілому пристрої формування імпульсів розв'язують такі задачі:

- формування меандру з синусоїдального або іншого типу аналогового сигналу;
- формування імпульсів з короткими фронтами та зрізами (спадами);
- скорочення та розширення тривалості імпульсів;
- організація часової затримки імпульсів;
- формування короткочасних імпульсів з заданою тривалістю фронту (спаду) імпульсу.

Затримки цифрових сигналів необхідні перш за все для часового узгодження в розповсюдженні сигналів різними шляхами в цифровому пристрої з метою боротьби з критичними змаганнями, які порушують працездатність пристроїв (наприклад, автоматів з пам'яттю).

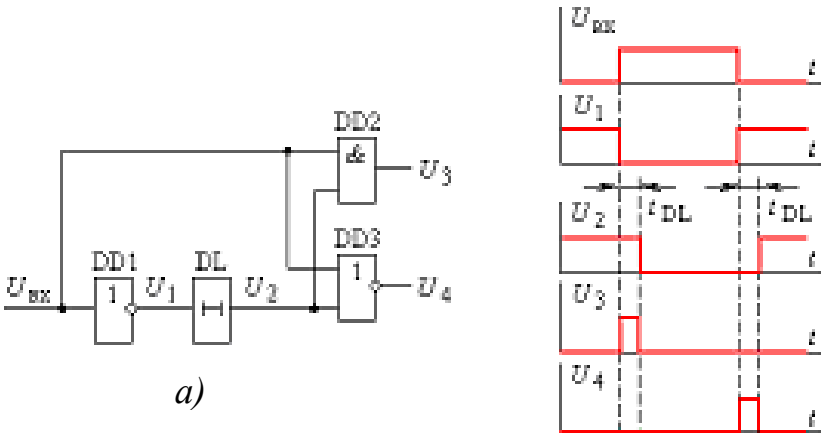
Часові перетворення імпульсів можуть виконуватись як з використанням лише логічних елементів, так і з використанням інтегруючих та диференціюючих *RC*-ланок.

10.1.1. Пристрої часових перетворень на основі логічних елементів

У цьому випадку часові перетворення забезпечуються за рахунок внутрішніх затримок логічних елементів. Оскільки ці затримки для сучасної елементної бази визначаються одиницями-десятками наносекунд, то такі

пристрої забезпечують формування досить коротких імпульсів, які можуть бути зафіксовані далеко не кожним осцилографом.

На рис. 10.1, *а* приведена узагальнена схема пристрою, за допомогою якої можна пояснити можливості формування імпульсів та організації їх затримок. Часові діаграми, що пояснюють роботу пристрою, приведені на рис. 10.1, *б*.



а) *б)*

Рис. 10.1

Вхідний сигнал $U_{вх}$ інвертується ЛЕ DD1 і подається на вхід пристрою затримки DL, в якості якого може використовуватись будь-яка парна кількість типових логічних елементів, кожен з яких має час затримки $t_З$, тобто:

$$t_{DL} = 2\,k\,t_з,$$

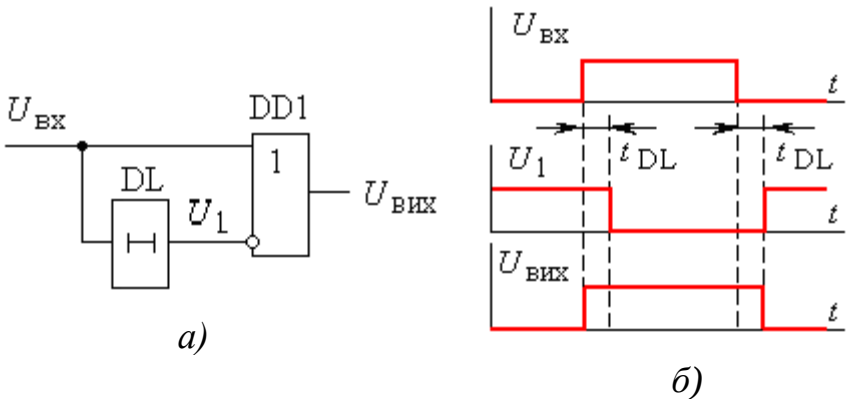
де k – кількість пар логічних елементів. Часовою затримкою інвертора DD1 можно знехтувати при $t_{3DD1} << t_{DL}$, або додати її до затримки пристрою DL, якщо чисельно вони сумірні.

Оскільки елемент **І** забезпечує формування вихідного імпульсу при співпаданні високих рівнів вхідних сигналів, то він фіксує інтервал часу затримки за фронтом вхідного імпульсу.

Елемент **АБО-НІ** фіксує співпадання низьких рівнів вхідних сигналів, тобто інтервал часу затримки за спадом вхідного сигналу. В обох випадках отримання вихідного імпульсу базується на співпаданні вхідного сигналу і сигналу з виходу елементу затримки.

Слід пам’ятати, що величина часу затримки для кожної з серій мікросхем має свої значення, які приводяться у довідковій літературі, і мають значний розкид, а тому точність формування затримки досить низька. Для мікросхем КМОН, які можуть працювати в широкому діапазоні напруг живлення, величина $t_З$ змінюється залежно від напруги живлення, причому чим вона менша , тим більша величина затримки. Така властивість КМОН ІС іноді використовується для зміни величини $t_З$ або зменшення кількості логічних елементів, що використовуються в лініях затримки.

Подібним шляхом можна збільшувати тривалість вихідних імпульсів, використовуючи схему рис. 10.2,а без інвертора, поклавши функцію інвертування на елемент затримки, використавши, наприклад, в якості елементу DL непарну кількість послідовно з’єднаних ЛЕ; реалізуючих функцію НІ.



а) *б)*

Рис. 10.2

Враховуючи той факт, що розкиди фронтів, спадів та інтервалів часу затримок досить значні, очікувати високу точность формування імпульсів не слід.

10.1.2. Використання зовнішніх RC-ланок

Зовнішні *RC*-ланки можуть мати постійні часу, величини яких на декілька порядків перевищують часові затримки логічних елементів. Інтегруючі ланки, встановлені між логічними елементами, можуть формувати часові затримки

значних величин, суттєво змінювати тривалості імпульсів, а також створювати принципово нові пристрої. Диференціюючі ланки, встановлені між ЛЕ, дають можливість створювати короткі імпульси.

Використання інтегруючих RC-ланок. Прикладом використання інтегруючої ланки є пристрій, схема якого приведена на рис. 10.3.

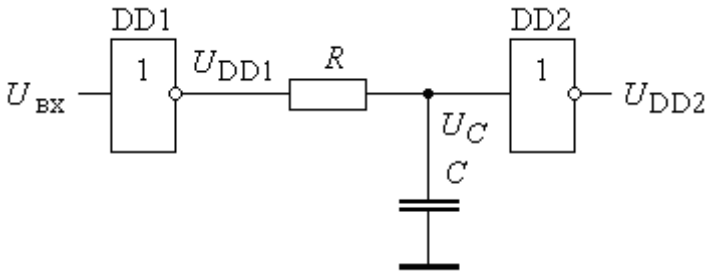


Рис. 10.3

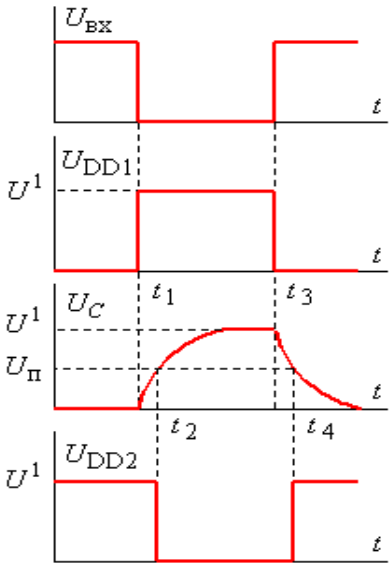


Рис. 10.4

Робота пристрою основана на використанні часової зміни напруги на конденсаторі при його заряді або розряді і фіксації конкретних значень напруги U_C за допомогою порогових пристроїв. Тому ЛЕ DD2 використовується як пороговий пристрій, напруга перемикання якого $U_П$ і є тією пороговою величиною, яка змінює його стан.

Якщо виходити з умови ідеальності ЛЕ DD1 і DD2, тобто вихідний опір DD1 при $U_{DD1} = U^1$ дорівнює нулю, вхідний опір DD2 сягає нескінченності, а внутрішня затримка на перемикання обох елементів $t_3 = 0$, то робота пристрою може бути зрозумілою з часових діаграм, приведених на рис. 10.4.

При високому рівні вхідного сигналу $U_{вх}$ на виході DD1 маємо напругу низького рівня, і конденсатор C розряджений. При зміні вхідного сигналу з одиничного до нульового рівня в момент часу t_1 на його виході виникне напруга високого рівня U^1 і починається процес заряджання конденсатора з постійною часу $\tau_{RC} = R\,C$.

Коли напруга на конденсаторі U_C досягне рівня порогової напруги $U_П$ (момент t_2), ЛЕ DD2 змінить свій стан, і на його виході рівень напруги зміниться на нульовий.

До моменту часу t_3 конденсатор C практично повністю зарядиться, а з моменту t_3 зміни стану виходу DD1 почне розряджатися з тією ж постійною часу, поки не досягне порогового рівня.

Величина затримки $t_З = t_2 - t_1 = t_4 - t_3$ при вказаних вище умовах розраховується за відомою формулою:

$$t_З = R\,C \ln \frac{U^1}{U_п}.$$

При використанні КМОН ІС умови роботи реального пристрою майже повністю співпадають з описаними, якщо внутрішні опори каналів транзисторів DD1 у відкритому стані набагато менші величини опору резистора R_1. Оскільки напруга нульового рівня на виході КМОН ЛЕ досить незначна $U^0 \approx 0$, $U_П = 0{,}5\,U^1$, то для практичних розрахунків справедливою буде формула:

$$t_З = R\,C \ln 2 \approx 0.7\,RC.$$

При використанні ТТЛ ІС реальні умови роботи пристрою суттєво змінюються, оскільки технічні характеристики мікросхем цих серій далекі від ідеальних. Перш за все, через вхід мікросхеми DD2 при $U_C < U_П$ протікатиме струм $I^0_{вх}$, який задається базовим резистором $R_Б$ багатоемітерного транзистора (рис. 10.5). Цей струм додатково заряджатиме конденсатор C.

Оскільки вхід ЛЕ DD2 (рис. 10.3) перебуватиме під дією напруги конденсатора, що змінюється, то величина його вхідного струму також змінюватиметься.

З моменту часу, коли $U_C > U_П$, емітерний перехід багатоемітерного транзистора ЛЕ DD2 зміститься в зворотному напрямку, а конденсатор продовжуватиме заряджатись за рахунок струму, який протікатиме через резистор R і внутрішній $R_В$ ЛЕ DD1. При зміні стану виходу ЛЕ DD1 на нульовий, конденсатор C розряджатиметься з постійною часу $R\,C$ в

інтервалі t_3 - t_4. Як тільки напруга U_C зменшиться до порогового рівня, між обома логічними елементами утвориться електричне коло струму від джерела напруги живлення +Е через базовий резистор багатоемітерного транзистора $R_Б$, перехід база – емітер, резистор R та відкритий нижній транзистор ЛЕ DD1. Наявність такого кола призведе до того, що напруга на конденсаторі зменшиться не до нуля, а триматиметься на рівні:

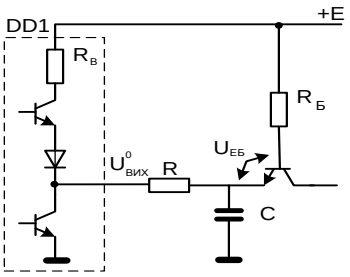


Рис.10.5

$$U_C = U^0_{\text{вих}} + I^0_{\text{вх}} \cdot R.$$

Схема буде працездатною, якщо виконуватиметься умова: $U_C < U_{\text{П}}$, з якої знаходиться обмеження на вибір резистора R:

$$R < \frac{(U_{\text{П}} - U^0_{\text{вих}}) \cdot R_{\text{Б}}}{E - U_{\text{ЕБ}} - U_{\text{П}}}.$$

З метою забезпечення стійкої роботи пристрою і через необхідність врахування розкидів порогових рівнів величину опору резистора R вибирають приблизно на порядок меншою, ніж дає розрахунок за формулою. Ємність конденсатора C не повинна перевищувати 2000 пФ.

У практичній схемотехніці інтегруюча ланка знаходить досить широке використання. На рис. 10.6 приводиться схема пристрою, призначеного для скорочення тривалості імпульсу. Його роботу пояснюють часові діаграми, приведені на рис. 10.7.

При відсутності вхідного імпульсу ($U_{\text{вх}} = 0$) на виході ЛЕ DD1 маємо високий рівень напруги, і конденсатор знаходиться в зарядженому стані.

На входах DD2 маємо різні рівні сигналів, внаслідок чого на його виході забезпечується високий рівень напруги ($U_{\text{вих}} = U^1$). Після появи вхідного імпульсу в момент t_1 протягом часового інтервалу t_1 - t_2 на входах DD2 діятимуть високі рівні сигналів, що забезпечить $U_{\text{вих}} = U^0 \approx 0$. Як тільки в момент часу t_2 виконається умова: $U_C \le U_{\text{П}}$, на входах DD2 знову будуть сигнали різного рівня, а на виході матимемо високий рівень ($U_{\text{вих}} = U^1$).

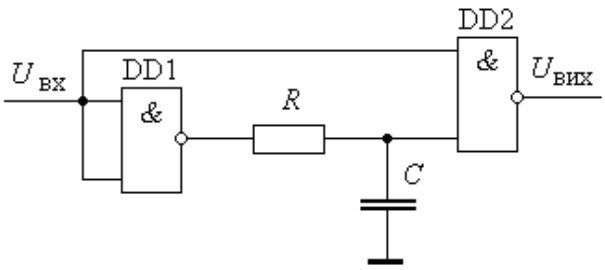


Рис. 10.6

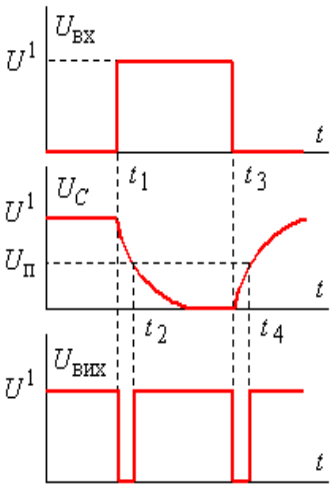


Рис. 10.7

Приклад 10.1. Побудувати часові діаграми і пояснити роботу пристрою, схема якого приведена на рис. 10.8.

Розв'язання. Зобразимо вхідний імпульс у вигляді сигналу U^1 в інтервалі $t_1 \ldots t_2$. Тоді на виході DD1 матимемо інвертований сигнал у тому ж часовому інтервалі (затримкою перемикання DD1 нехтуємо). Будуємо часову діаграму зміни напруги U_C на конденсаторі C. З моменту t_1 конденсатор заряджається від джерела вхідного сигналу, а з моменту t_2 він розряджається до нуля через джерело вхідного сигналу, переходячи кожного разу через пороговий рівень $U_П$ (рис. 10.9).

Логічна одиниця на виході DD2 буде в будь-який інтервал часу, коли його вхідні сигнали різні, а логічний нуль – лише в тому випадку, коли обидва сигнали мають високий рівень. Таким чином в інтервалі часу $t_2 \ldots t_3$ на його виході буде низький рівень сигналу, який забезпечує інвертовану затримку за спадом імпульсу.

З приведеного прикладу витікає важлива особливість використання *RC*-ланок для створення часових затримок. Вона полягає в тому, що розряд конденсатора відбувається через джерело вхідного сигналу, величина внутрішнього опору якого буде суттєво впливати на час розряду і, відповідно, на величину затримки.

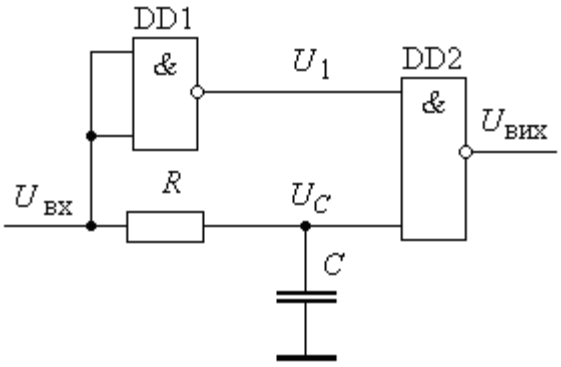


Рис. 10.8

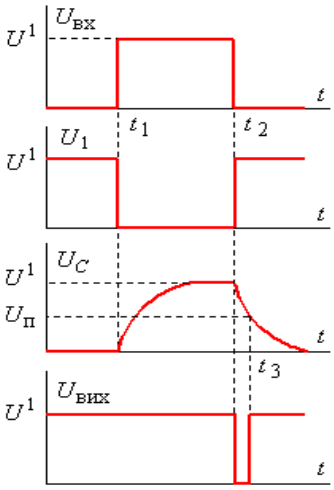


Рис. 10.9

Приклад 10.2. Вважаючи в попередньому прикладі, що використовуються КМОН ІС, обчислити параметри *R* і *C* для забезпечення часової затримки $t_3 = 10^{-3}$ с.

Розв'язання. Для КМОН ІС справедлива формула (10.2), з якої знаходимо:

$$RC = \frac{1}{0.7} \cdot 10^{-3} = 1.43 \cdot 10^{-3} \text{ с}.$$

Задавши величину опору *R* = 10 кОм, знаходимо:

$$C = 1.43 \cdot 10^{-3} \cdot \frac{1}{10^4} = 1.43 \cdot 10^{-7} \approx 0.15 \text{ мкФ}.$$

Приклад 10.3. Запропонувати схему пристрою розширювача імпульсів. Пристрій повинен розширювати тривалість імпульсів з 2 мкс до 3 мкс за спадом.

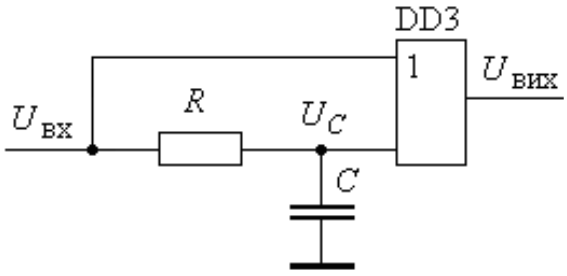


Рис. 10.10

Розв'язання. Оскільки пристрій повинен розширювати імпульси за спадом, то на виході необхідно мати ЛЕ, що реалізує функцію **АБО**, який би об'єднував вхідний імпульс та імпульс з інтегруючої ланки. Схема пристрою приводиться на рис. 10.10.

Розрахунок параметрів інтегруючої ланки виконується аналогічно попередньому прикладу.

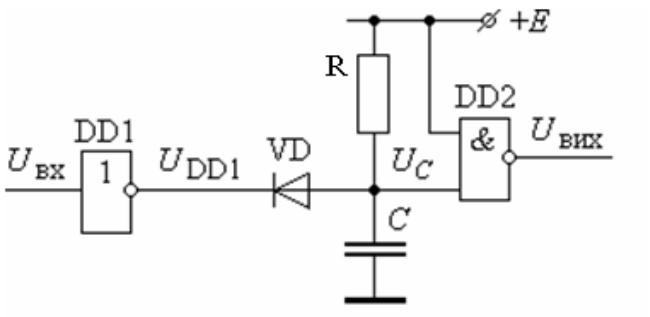
При визначенні параметрів *RC*-ланок здебільшого виходять з умови експоненціального характеру процесу заряду-розряду конденсатора.

Тривалість будь-якого інтервалу t такого процесу може бути визначена з використанням формули:

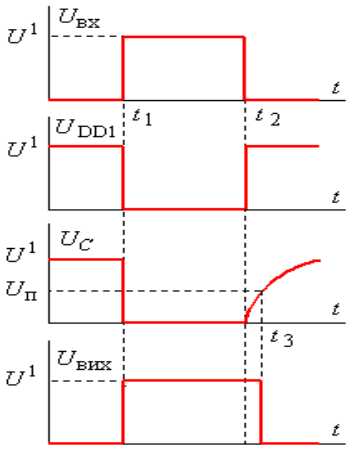
$$t_i = R\,C \ln \frac{U(\infty) - U(0)}{U(\infty) - U(t_i)},$$

де $U(\infty) - U(0)$ – розмах експоненти; $U(\infty) - U(t_i)$ – частина експоненти, що залишається поза інтервалом часу t_i.

Інший варіант розширювача імпульсу за його спадом реалізується з використанням елемента **2І** (рис.10.11,а), а часові діаграми − на рис. 10.11, б.



а)



б)

Рис. 10.11

При відсутності вхідного імпульсу вихідний сигнал U_{DD1} ЛЕ DD1 має високий рівень. Діод VD- закритий, а конденсатор С заряджений, і напруга U_C на ньому відповідає напрузі високого рівня U^1. Внаслідок цього на вході DD2 діють два сигнали високого рівня, а на його виході маємо U^0. При появі вхідного імпульсу напруга на виході DD1 приймає значення, близьке до нуля, і конденсатор C через діод VD швидко розряджається. Напруга U_C стає близькою до нуля, і стан DD2 змінюється на протилежний. В інтервалі часу $t_1 \ldots t_2$ дії вхідного імпульсу вихід ЛЕ DD1 шунтує конденсатор С, тому стан ЛЕ DD2- не змінюється.

У момент t_2 рівень вхідного сигналу змінюється до нуля, а вихід ЛЕ DD1- навпаки $U_{DD1} = U^1$. Внаслідок цього починає заряджатись конденсатор C з постійною часу $R\,C$.

У момент t_3 напруга на ньому досягає порогової величини $U_C = U_П$, і стан ЛЕ DD2 змінюється на протилежний.

Величина подовження тривалості імпульсу обчислюється за формулою:

$$t_3 - t_2 = R\,C \ln\left(\frac{E - U_{VD}}{E - U_п}\right), \qquad (10.4)$$

де U_{VD} – падіння напруги на діоді у відкритому стані.

У розглянутому розширювачі використовуються переважно КМОН ІС.

Більш точна фіксація порогових рівнів забезпечується при використанні в якості елементів DD1 і DD2 тригерів Шмідта.

Подальше підвищення точності роботи може бути досягнуто, використанням в якості DD2 регенеративних порогових схем – наприклад, *RS*-тригерів (рис. 10.12). У цій схемі в якості порового елемента використовується *RS*-тригер на основі тригерів Шмідта (мікросхема КР1533ТЛ3).

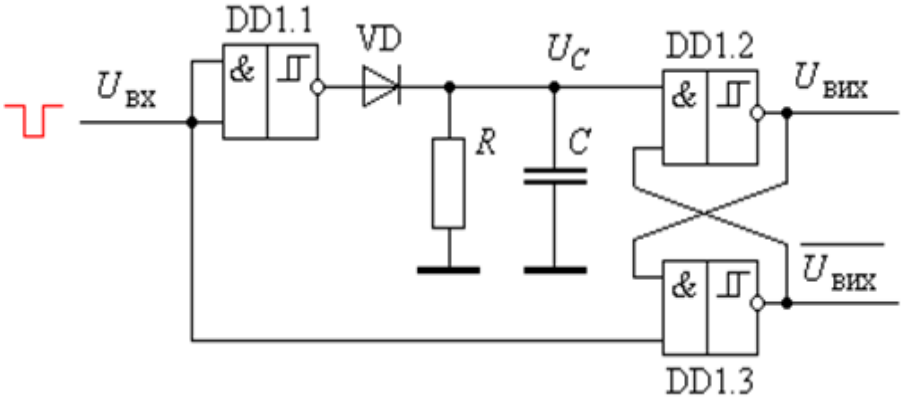


Рис. 10.12

Часові діаграми, що пояснюють роботу пристрою, зображені на рис. 10.13. При наявності на вході напруги високого рівня *($U_{вх}$ = U^1)* на виході DD1.1 маємо низький рівень напруги, і конденсатор *C* розряджений, оскільки він запаралелений резистором *R*. Відомо (див. **Розділ 4**), що при такій комбінації сигналів на вході *RS*-тригера на його виходах маємо: $U_{вих} = U^1$; $\overline{U_{вих}} = U^0$.

Поява вхідного сигналу низького рівня в момент t_1 (рис. 10.13), який одночасно подається на входи DD1.1 і DD1.3, призводить до зміни стану *RS*-тригера: напруга на виходах стає: $U_{\text{вих}} = U^0$, а $\overline{U_{\text{вих}}} = U^1$. Конденсатор C зарядиться завдяки високому рівню напруги на виході DD1.1.

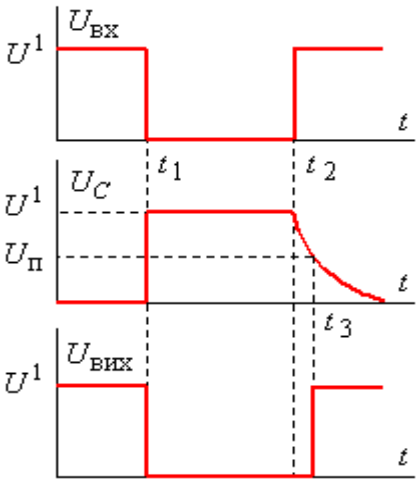


Рис. 10.13

У момент часу t_2 вхідний імпульс прийме значення U^1, але напруга на конденсаторі в інтервалі часу $t_2 \ldots t_3$ підтримуватиметься на рівні вище порогового ($U_C > U_{\text{п}}$), тому на обох входах DD1.2 буде високий рівень сигналу і стан *RS*-тригера не змінюватиметься. В момент часу t_3, коли стане $U_C < U_{\text{п}}$, тригер змінить свій стан і на його прямому виході установиться знову високий рівень напруги. Величина часової затримки $t_3 - t_2$ обчислюється аналогічно попередній схемі.

Використання диференціюючих RC-ланок. Головна особливість диференціюючої ланки полягає в тому, що при дії однополярного позитивного імпульсу на її виході створюються два різнополярні імпульси, причому негативний може пошкодити вхідні кола наступної мікросхеми. Тому при використанні мікросхем, в яких відсутні вхідні антидзвонові діоди, паралельно резисторові необхідно передбачити включення в зворотному напрямку діода.

При використанні КМОН ІС робота пристроїв з диференціючими ланками мало чим відрізняється від вище описаних схем з інтегруючими ланками.

На рис. 10.14 приводиться схема пристрою, призначеного для скорочення тривалості імпульсу, а на рис. 10.15 – часові діаграми, пояснюючі його роботу. Приймемо мікросхеми DD1 і DD2 як ключові повторювачі, в якості який можуть використовуватись ЛЕ **2І**, **2АБО**.

При появі високого рівня напруги U^1 на виході DD1 і розрядженому конденсаторі C через RC ланцюг починає протікати струм, який в момент t_1 обмежується лише опором резистора R. Вся напруга при цьому прикладається

до резистора. З часом конденсатор заряджається, і струм, що протікає через елементи, зменшується за експоненціальним законом.

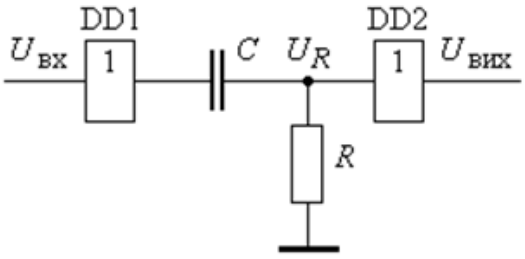


Рис. 10.14

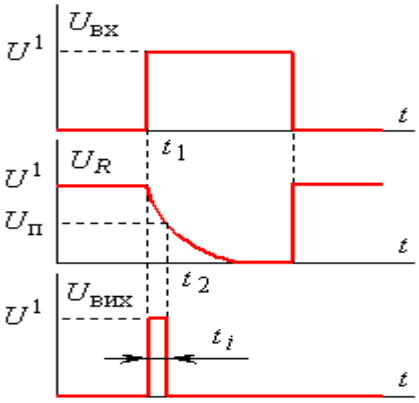


Рис. 10.15

В інтервалі часу t_1 - t_2 напруга U_R перевищує пороговий рівень $U_П$, тому вихідний стан ЛЕ DD2 відповідає рівню U^1. В момент t_2 нерівність $U_R > U_П$ порушується, і сигнал на виході приймає рівень U^0.

Для пристроїв, в яких використовуються ЛЕ КМОН:

$$t_i = R\,C \ln\left(\frac{U^1}{U^1 - U_П}\right) \approx 0,7\,R\,C\,.$$

При використанні ТТЛ ІС разом з диференціюючими ланками існують деякі особливості. Вони полягають в тому, що при високому рівні сигналу на виході DD1 величина струму через елементи *RC* ланки обмежується не лише опором R, а й опором R_3 вихідного каскаду ТТЛ ІС, що розміщений у колекторному колі верхнього транзистора (див. **Розділ 2** рис.2.15, рис.2.16). У такому випадку максимальна величина падіння напруги на резисторі R обмежується значенням:

$$U_{R\max} = E\frac{R}{R + R_3} \qquad (10.5)$$

де R_3 – колекторний опір виходу ЛЕ DD1. Виходячи з (10.5), бажано, щоб виконувалася нерівність $R >> R_3$.

При низькому рівні напруги на виході DD1 через резистор R протікає вхідний струм мікросхеми DD2 і створює падіння напруги $U_R = I_{вх}\,R$.

При цьому можливі два режими роботи.

При виконанні умови $I_{вх}R = U_R < U_П$, на виході DD2 маємо також низький рівень напруги U^0. В цьому випадку при появі вхідного імпульсу інтервал затримки:

$$t_i = (R + R_3) \cdot C \cdot \ln\left[\frac{(E - U_{KE} - U_{VD})R}{U_П(R + R_3)}\right] \approx (R + R_3) \cdot C \cdot \ln\frac{3.9R}{1.3(R + R_3)}$$

Якщо ж величина опору резистора R вибрана такою що, падіння напруги на ньому $U_R > U_{\text{п}}$, то на виході незмінно утримуватиметься високий рівень $U_{\text{вих}} = U^1$. Тому величина опору резистора R має верхню межу, яка залежить від величини вхідного струму ТТЛ ІС. Як приклад, для ІС серії КР1533 при наявності на вході сигналу U^0 величина вхідного струму $I_{\text{вх}}^0 = 0.2$ мА, а порогова напруга високого рівня −2 В. Тому $R << \frac{U_{\text{п}}}{I_{\text{вх}}^0} = \frac{2}{0.2 \cdot 10^{-3}} = 10$ кОм.

Як висновок, в результаті аналізу, можемо стверджувати:

- скорочення тривалості імпульсу високого рівня досягається використанням диференціюючої ланки з невеликим значенням опору резистора;
- величина опору резистора диференціюючої ланки залежить від типу використовуваних ТТЛ мікросхем і їх порогу перемикання;
- тривалість вихідного імпульсу характеризується невисокою точністю, оскільки напруга порогового рівня ЛЕ співпадає з пологою областю напруги на конденсаторі при його заряді.

Розглянемо ще один приклад роботи формуючого пристрою з інвертуючими ЛЕ (рис. 10.16, *а*), часові діаграми роботи на рис. 10.16, *б*.

Особливість роботи *RC*-ланки полягає в тому, що при нульовому рівні вхідного імпульсу напруга на виході DD1 (точка **А**) має високий рівень.

В той же час, при значній величині опору R вхідний струм ЛЕ DD2 створюватиме на ньому падіння напруги:

$$U_R > U_{\text{п}}, \qquad (10.6)$$

внаслідок чого в точці **B** також підтримуватиметься високий рівень напруги і конденсатор перебуватиме в розрядженому стані.

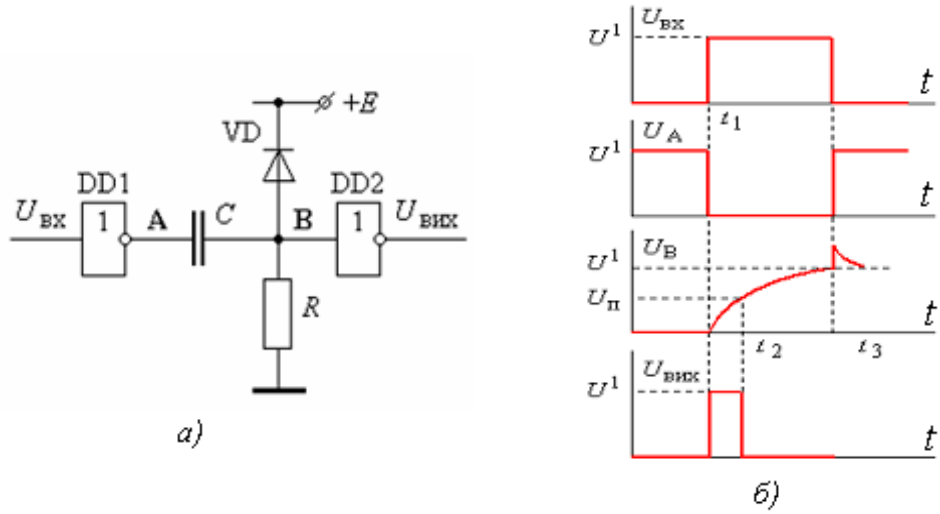


Рис. 10.16

При появі в момент t_1 вхідного імпульсу напруга в точці **A** стає близькою до нуля, і, при розрядженому конденсаторі, напруга в точці **B** короткочасно також матиме низький рівень. Внаслідок цього на виході DD2 виникає імпульс високого рівня.

З моменту t_1 конденсатор C починає заряджатись через вхід DD2 і при досягнення порогової напруги (момент t_2) ЛЕ DD2 змінює свій стан на протилежний.

В результаті маємо наступні особливості роботи пристрою:

- скорочення тривалості імпульсу досягається при збільшенні порівняно опору резистора диференціюючої ланки;
- точність отримання заданої тривалості вихідного імпульсу досить низька із-за причин, аналогічних попередній схемі;
- підвищення точності пристроїв, призначених для скорочення тривалості вхідних імпульсів, може бути досягнуто шляхом використання в якості порогових елементів тригерів Шмідта та інших регенеративних схем зі стабільнимпороговим рівнем.

10.1.3. Пристрої перетворення форми імпульсів

Пристрої перетворення форми імпульсів призначені для формування прямокутних імпульсів з крутим фронтом та зрізом з аналогових сигналів та імпульсів іншої форми.

Щоб зрозуміти необхідність таких пристроїв, розглянемо задачу, яка досить часто зустрічається на практиці – формування меандру з синусоїдальної напруги. Допустимо, що необхідно розробити пристрій, який би з високою точністю перетворював період синусоїди промислової мережі в меандр. Цифрові пристрої живляться від джерела напруги +5 В.

Точність перетворювача визначається точністю відтворення моментів часу переходу синусоїдою через нуль. Тому в таких випадках необхідно мати високу крутизну вхідної напруги, яка забезпечується лише допустимим її амплітудним значенням, а також пристрої, що забезпечать високий коефіцієнт підсилення вхідного сигналу при малих амплітудах.

Зрозуміло, що високу напругу на вхід схеми подавати недопустимо з наступних причин:

- знижується надійність пристрою;
- при використанні ТТЛ ІС виникає режим насичення транзисторів.

У той же час, обмеженою є величина коефіцієнта підсилення ЛЕ, а також наявність порогової напруги, яка призводить до затримки перемикання відносно нульового рівня вхідної напруги.

У таких випадках використовують аналогові компаратори, які мають високу чутливість і швидкодію. При використанні ЛЕ спочатку встановлюються обмежувачі напруги (наприклад, на основі стабілітронів), після яких використовують два послідовно з’єднаних ЛЕ. Таким шляхом забезпечується висока крутизна фронтів вхідного сигналу і значний коефіцієнт підсилення. При наявності завад у вхідному сигналі рекомендується використовувати тригери Шмідта, які є в багатьох серіях ІС.

У цифровій схемотехніці широке застосування знаходить пристрій, схема якого приведена на рис. 10.17.

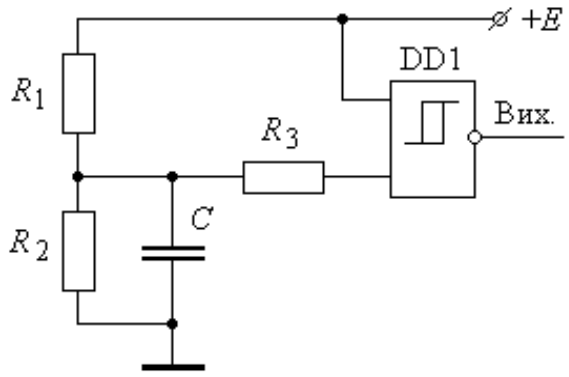


Рис. 10.17

На виході пристрою з’являється прямокутний імпульс при вмиканні напруги живлення. Цей імпульс призначається для того, щоб встановити в вихідний (початковий) стан різноманітні вузли, що містять тригери і можуть мати невизначений стан при вмиканні. Робота пристрою визначається інтегруючою ланкою на вході гістерезисного компаратора. При вмиканні живлення розряджений конденсатор шунтує резистор R_2, і на виході DD1 матимемо високий рівень напруги, який зберігатиметься певний інтервал часу, протягом якого конденсатор зарядиться до порогового рівня. Як тільки напруга на конденсаторі підніметься вище порогової напруги компаратора, на його виході встановиться низький рівень сигналу.

10.2. Одновібратори

Одновібратором називається імпульсний пристрій, призначений для генерації

одиночних прямокутних імпульсів заданої тривалості при дії на нього зовнішніх сигналів. У літературі він має також назву *моновібратор*, *очікуючий мультивібратор*.

Характерною особливістю одновібраторів є наявність позитивного зворотного зв’язку регенеративного типу, що забезпечує високу швидкість перемикання. Цим і досягається ряд позитивних характеристик одновібраторів порівняно з пристроями, приведеними в попередньому параграфі. До них слід віднести:

- високу крутизну вихідних імпульсів;

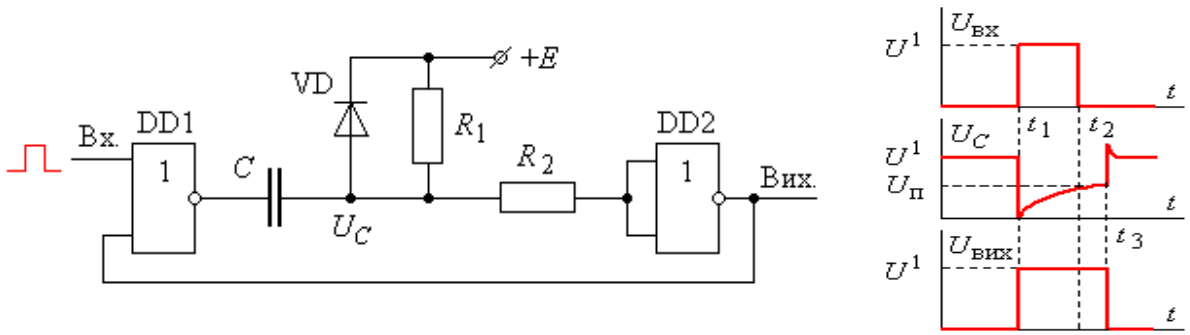
- високу часову стабільність вихідних імпульсів;
- високу завадостійкість пристроїв.

Розглянемо основні принципи побудови і використання одновібраторів – як виготовлених на основі логічних елементів, так і на основі спеціалізованих схем.

10.2.1. Одновібратори на основі логічних елементів

Найпростішими одовібраторами є пристрої, виготовлені на основі логічних елементів. Ці пристрої мають не досить високі технічні характеристики, але значно простіші в порівнянні зі спеціалізованими. Одна з досить розповсюджених схем одновібратора приведена на рис. 10.18, *а*, робота пояснюється часовими діаграмами, зображеними на рис. 10.18, *б*.

При відсутності вхідного імпульсу пристрій знаходиться у *режимі очікування*, на вході ЛЕ DD2 діє сигнал високого рівня від джерела живлення.



а) *б)*

Рис. 10.18

Відповідно, вихід DD2 матиме низький рівень сигналу, який подається на один з входів ЛЕ DD1. Тому в режимі очікування на виході DD1 зберігається сигнал високого рівня. Обкладки конденсатора *C* знаходяться під однаковими потенціалами, і він розряджений.

При подачі вхідного імпульсу високого рівня вихід ЛЕ DD1 змінить свій початковий стан на нульовий. Оскільки конденсатор *C* розряджений, вхід ЛЕ DD2 прийме стан низького рівня. По гілці зворотного зв’язку інвертований

сигнал подається на вхід DD1 і підтримуватиме на його виході низький рівень. Звідси витікає, що тривалість вхідного імпульсу повинна бути достатньою для того, щоб встановити елементи в інший, протилежний, якщо виходити з рівнів сигналів на виходах ЛЕ, стан. Якщо, наприклад, час перемикання ЛЕ складає 10 нс, то мінімальна тривалість вхідного імпульсу повинна складати не менш ніж $2 \cdot 10^{-5} c = 20$ нс.

Описаний другий стан називається *станом збудження*. Тривалість знаходження пристрою в цьому стані визначається інтервалом часу, протягом якого напруга на конденсаторі U_{C} буде меншою порогового рівня ЛЕ DD2. Тому такий стан є стійким лише тимчасово. Поява на виході ЛЕ DD1 низького рівня сигналу приводить до появи електричного кола (+*E*, R_1, *C*, ЛЕ DD1, загальна шина), по якому протікає струм заряду конденсатора, напруга U_{C} якого зростатиме. В момент t_3, коли $U_C > U_{\text{п}}$, ЛЕ DD2 змінює свій стан, на його виході напруга зменшується до нуля, через гілку зворотного зв'язку вона переводить DD1 в стан, характерний для *режиму спокою* пристрою. Ліва обкладка конденсатора через вихід ЛЕ DD1 приєднується до джерела живлення, і накопичений на ньому заряд розсіюється на внутрішньому опорі DD1 і VD.

Тривалість перебування одновібратора в збудженому стані визначається часом заряду конденсатора до порогового рівня і задає інтервал дії імпульсу високого рівня на виході ЛЕ DD2:

$$\tau_i = t_3 - t_1 = R_1 C \ln\left(\frac{E}{E - U_{\text{п}}}\right), \qquad (10.6)$$

Оскільки для мікросхем КМОН приймається $U_{\text{п}} = \frac{E}{2}$, то формула (10.6) приймає відомий вигляд: $\tau_i = 0.7 R_1 C$.

Враховуючи властивість оберненості бінарної логіки (на основі теореми де Моргана), аналогічний одновібратор може бути побудований на елементах **2I-HI**.

Схема такого одновібратора приводиться на рис. 10.19 з тривалістю генерованого імпульсу $\tau_i = R_1 C \ln \frac{E}{U_{\text{п}}}$.

Читачам пропонується самостійно проаналізувати роботу одновібратора і побудувати часові діаграми.

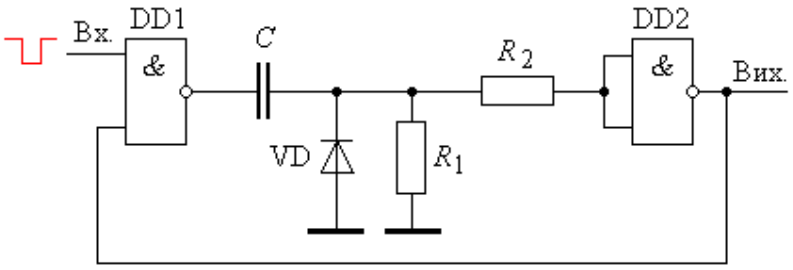


Рис. 10.19

При спробі виготовлення одновібраторів слід враховувати деякі суто практичні деталі. В обох схемах конденсатор розряджається через діод VD і вихід ЛЕ DD1. Але якщо в першій схемі в колі розряду знаходиться опір верхнього вихідного транзистора ЛЕ DD1, що обмежує величину струму розряду, то в другій − розряд відбувається через нижній низькоомний транзистор ЛЕ DD1. Тому в другій схемі слід обмежувати величину ємності конденсатора.

У мікросхемах КМОН на входах встановлюються діоди, але їхні допустимі струми незначні. Тому для обмеження величини розрядних струмів використовується резистор R_2. Діод VD фактично шунтує внутрішні діоди мікросхем, тому при його наявності опір R_2 необов'язковий.

Оскільки величина затримки суттєво залежить від порогового рівня напруги, який коливається для різних мікросхем у значних межах, то в розглянутих одновібраторах досягти високу точність і стабільність імпульсів проблематично. Покращення цих параметрів може бути досягнуто вже відомим використанням в якості DD2 тригерів Шмідта або схем з застосуванням двох часозадаючих *RC*-кіл.

При використанні ТТЛ ІС застосовують ті ж схеми, що приведені на рис. 10.18 і рис. 10.19. Оскільки вхідні опори ТТЛ ІС незначні, це приводить до необхідності зниження величини опору резистора R_1. При тих же параметрах імпульсів це призводить до необхідності збільшення ємності конденсаторів і пов'язаного з цим зменшення стабільності схем.

Оскільки на процес заряду в схемах з ТТЛ ІС впливає вхідний струм, то його необхідно врахувати у розрахункових формулах (10.5) і (10.6) шляхом заміни R на еквівалентний опір $R_{\text{Е}} = R \parallel R_{\text{Б}}$.

У спеціальній літературі приводиться ряд схем одновібраторів, які забезпечують достатньо високу стабільність роботи [Зельд].

10.2.2. Одновібратори на спеціалізованих мікросхемах

У ТТЛ ІС різних серій існують спеціалізовані мікросхеми одновібраторів. У серіях ТТЛШ (КР1533 і К555) широко використовується одновібратор АГ3, умовне позначення якого приведено на рис. 10.20. Подібний одновібратор виготовляється і в серіях КМОН ІС (561, 564) (одновібратор АГ1).

Таблиця станів (табл. 10.1) визначає режими роботи мікросхеми.

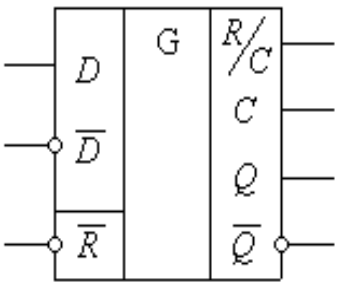


Рис. 10.20

Таблиця 10.1

Входи			Виходи		Режим роботи
R	D	$\overline{D}$	Q	$\overline{Q}$	
L	x	x	L	H	Стійкий стан
x	H	x	L	H	
x	x	L	L	H	
H	L	⎍	⎍	⎍	Запуск
H	⎍	H	⎍	⎍	
⎍	L	H	⎍	⎍	
⎍	x	x	L	H	Скидання

Призначення виводів вказане в табл. 10.2.

Таблиця 10.2

Вивід	Призначення виводу
D, $\overline{D}$	Прямий та інверсний інформаційні входи
R	Вхід установки в стан логічного нуля
Q, $\overline{Q}$	Прямий та інверсний інформаційні виходи
R/C	Вивід для приєднання зовнішніх резистора і конденсатора
C	Вивід для приєднання зовнішнього конденсатора

Типове включення *RC*-ланки полягає в тому, що конденсатор приєднується між входами R/C та C, а резистор між входом R/C та джерелом живлення.

Тривалість вихідного імпульсу одновібратора визначається формулою [Зельд.]: $\tau_i = 0.28\,R\,C\left(1+\frac{0.7}{R}\right)$, яка при C > 1000 пФ спрощується: $\tau_i = 0.45\,R\,C$, де C задається в пікофарадах; R – в кОм, а τ_i – в нс.

Часові діаграми, що поясняють роботу мікросхеми, приведені на рис. 10.21.

Запуск одновібратора забезпечується або по прямому входу *D* активним високим рівнем вхідного сигналу, або по інверсному $\overline{D}$ активним низьким рівнем.

Якщо одновібратор запущений вхідним сигналом *D*, його вихідний імпульс *Q* можна продовжити (рис. 10.21), якщо до закінчення інтервалу τ_i подати повторно сигнал високого рівня на вхід *D*, або сигнал низького рівня на вхід $\overline{D}$. Вихідний імпульс можливо також обірвати, якщо подати на вхід $\overline{R}$ сигнал низького рівня (рис. 10.21, графіки часової зміни *R* і *Q*).

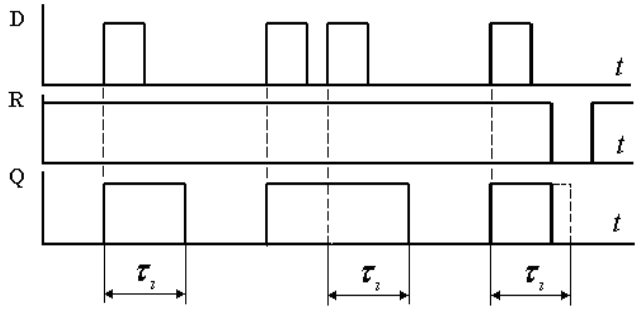


Рис. 10.21

Використання обох одновібраторів, з'єднаних по кільцевій схемі (в одному корпусі КР1533АГ3 (SN74LS123) міститься два одновібратори), дає можливість створювати високостабільні автогенераторні пристрої.

Відмінність схемотехніки одновібраторів, виготовлених за КМОН-технологією (561АГ1), полягає лише в тому, що вивід *C* мікросхеми заземлюється, що дає можливість забезпечити більшу завадостійкість використовуваних пристроїв.

Одновібратори АГ1 та АГ3 дають можливість створювати тривалості імпульсів в інтервалі від сотень наносекунд до декількох хвилин, що забезпечує їм досить широке використання.

Наявність двох керуючих входів відкриває допоміжні можливості використання одновібраторів. Наприклад, існують способи з'єднання декількох мікросхем в вигляді кільцевих або послідовних схем та інше.

10.2.3. Одновібратори на основі таймера КР1006ВИ1

Серед мікросхем, що використовуються в пристроях імпульсної схемотехніки, особливе місце посідає інтегральний таймер типу КР1006ВИ1 (зарубіжні аналоги NE555, SA555, KA555, LM555). Його функціональна схема приведена на рис. 10.22.

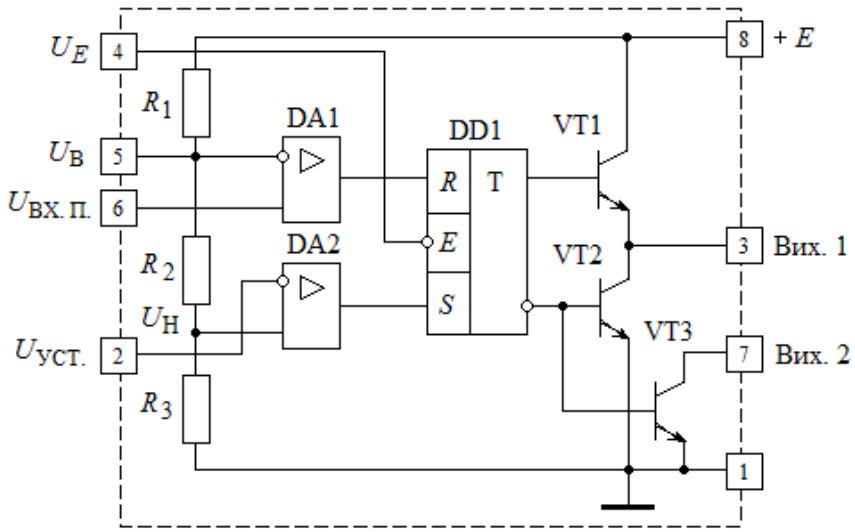


Рис. 10.22

Таймер містить у собі два компаратори DA1 і DA2, *RS*-тригер асинхронного типу, транзисторний ключ з відкритим колектором VT3, двотактний транзисторний підсилювач на VT1 і VT2, а також прецизійні резистори $R_1 \ldots R_3$, призначені для рівномірного розподілення напруги між ними, при цьому $U_{\text{Н}} = \frac{E}{3}$; $U_{\text{В}} = \frac{2E}{3}$. Резисторний дільник напруги виготовлений таким чином, що за допомогою зовнішніх резисторів, які приєднуються паралельно встановленим, можна змінювати пропорції між напругами $U_{\text{Н}}$ та $U_{\text{В}}$.

Вихідні каскади забезпечують досить високу універсальність таймера. Навантаження, що приєднується одним з виводів до **Вих. 1**, іншим може бути приєднаним як до загальної шини, так і до напруги живлення. Вихідний струм при обох способах приєднання навантаження може досягати 100 мА. Вихідний опір 10 Ом.

Вивід **Вих. 2** використовується як допоміжний, а також як вивід для приєднання зовнішніх елементів зворотного зв'язку. Вихідний струм до 100 мА.

Компаратори в таймері призначені для порівняння вхідної напруги $U_{\text{ВХ. П.}}$, що подається на вивід **6**, з еталонною. Установочний вхідний сигнал низького рівня $U_{\text{УСТ.}}$ подається на вхід **2**. Компаратор при цьому формує сигнал високого рівня, який встановлює *RS*-тригер в одиничний стан на прямому виході, транзистор VT2 відкривається, і на виході **Вих. 1** маємо сигнал високого рівня. Якщо напруга $U_{\text{ВХ. П.}}$ на вході **6** перевищить пороговий рівень $2E/3$, то компаратор DA1 своїм вихідним сигналом високого рівня встановить *RS*-тригер у початковий стан, а на виході **Вих. 1** буде сигнал низького рівня. Вхідні опори таймера досить високі, а тому приєднані до входів кола практично не навантажуються. Вхідний струм компараторів DA1, DA2 не перевищує 0,5мкА.

Зовнішній вхід E (вивід 4) тригера забезпечує доступ до *RS*-тригера безпосередньо. При низькому рівні напруги на вході E ($U_E \le 0{,}4$ В) тригер

встановлюється в початковий стан незалежно від рівнів сигналів на входах **2** і **5.** Сигнал високого рівня $(U_E > 0{,}4\ \text{В})$ на роботу схеми не впливає.

Якщо на виходи тригера одночасно поступають сигнали установки в різні стани, то тригер встановлюється в відповідності до *пріоритетів сигналів.* Найвищий пріоритет має сигнал дозволу U_E, що подається на вхід **4**. Його дія була описана вище. Другим за ієрархією є сигнал $U_{УСТ.}$, що подається на вхід **2.** При наявності сигналу дозволу, тобто при $U_E = 1$, за умови $U_{УСТ.} < U_H$ прямий вихід тригера, як вже відмічалось, прийме одиничне значення незалежно від рівня напруги $U_{ВХ.\,П.}$ на вході **6**. Найнижчий пріоритет має сигнал $U_{ВХ.\,П.}$. При виконанні умов: $U_E = 1$; $U_{УСТ.} > U_H$ і при $U_{ВХ.\,П.} > U_B$ тригер встановлюється в нуль.

При створенні пристроїв на основі таймерів часові параметри задаються зовнішніми *RC*-колами, а швидкий розряд конденсатора може бути забезпечений за допомогою транзистора VT3. У той же час, оскільки пороговим елементом є компаратор DA1 (вхід **6**), то звідси витікає типова схема використання таймера. Як варіант побудови одновібратора, вона приводиться на рис. 10.23, *а*.

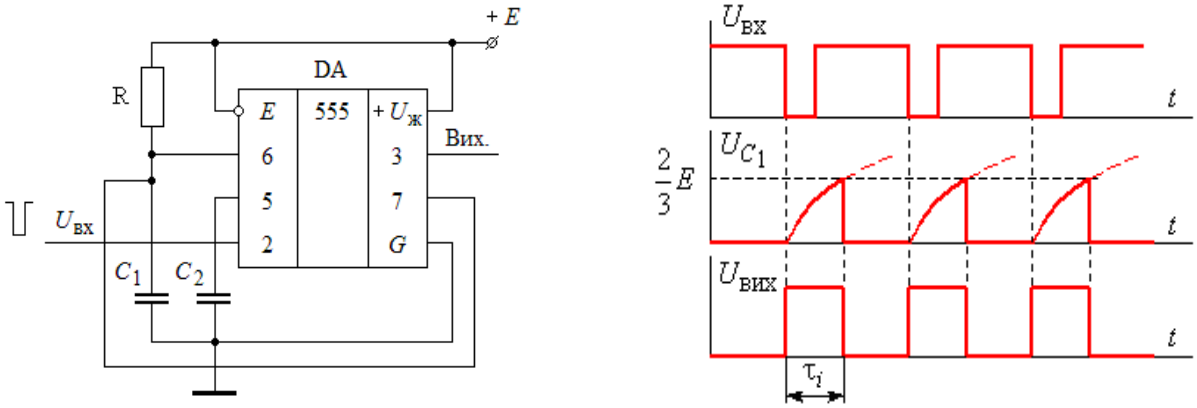
При високому рівні сигналу на вході **2** тригер одновібратора знаходиться у початковому стані (сигнал високого рівня на інверсному виході), транзистор VT3 відкритий і шунтує конденсатор C_1. Пристрій знаходиться в стійкому стані.

Поява короткочасного імпульсу низького рівня на вході **2** змінить стан тригера, і транзистор VT3 закриється. З цього моменту через резистор R від джерела живлення $+E$ почне заряджатись конденсатор C_1. У момент часу, коли напруга на ньому і, відповідно, на вході **6** досягне рівня $\frac{2}{3}E$, тригер змінить свій стан, і конденсатор миттєво розрядиться через транзистор VT3. Часові діаграми, які ілюструють описаний процес, приведені на рис. 10.23, *б*.

Тривалість вихідного імпульсу [Зельд.]: $\tau_i = R\,C_1 \ln 3 \approx 1{,}1\,R\,C_1$.

З формули видно, що величина τ_i не залежить від напруги живлення, що обумовлено тим, що вона однаково впливає як на порогову напругу на вході **6**, так і на напругу на конденсаторі.

Мінімальна тривалість τ_i обумовлена швидкодією елементів таймера і складає приблизно 10 мкс. Максимальна величина τ_i обмежується активною складовою опору конденсатора C_1 та вхідним струмом мікросхеми, який обмежує величину опору R зверху. Обумовлено це тим, що зарядний струм повинен значно перевищувати величину вхідного струму.



а) *б)*

Рис. 10.23

Обмеження опору R знизу накладає величина допустимого колекторного струму транзистора *VT3* (рис.10.23).

Тривалість запускаючого імпульсу завжди повинна бути меншою, ніж τ_i вихідного. Якщо за час дії τ_i поступає декілька запускаючих імпульсів, то вони не можуть змінити стан одновібратора.

10.2.4. Одновібратори на основі тригерів

Виходячи з принципу роботи одновібраторів, можна стверджувати, що для їх побудови необхідно мати тригери, які мають окремі входи для установки в одиничний та нульовий стани. Іншою особливістю є те, що для побудови

одновібраторів слід використовувати лише тригери КМОН-технологій, які мають стабільний пороговий рівень напруги перемикання та низькі вхідні струми.

Виходячи зі сказаного, розглянемо схему одновібратора, що приведена на рис. 10.24 з використанням тригера ТР2 серії 561.

У початковому стані на прямому виході тригера маємо сигнал низького рівня, конденсатор C розряджений і шунтується низькоомним виходом мікросхеми.

При подачі на вхід S імпульсу, що відповідає логічній **“1”**, тригер DD змінює свій стан на протилежний. Як наслідок, створюється коло для заряду конденсатора – джерело живлення $+E$, внутрішній (верхній) резистор $R_{\text{В}}$ (рис 10.5), резистор R, конденсатор C, загальна шина.

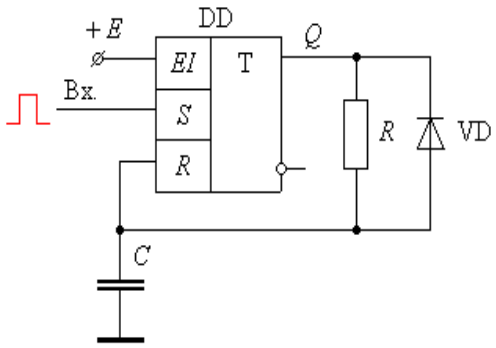


Рис. 10.24

Коли напруга на конденсаторі перевищує величину $\approx 0{,}5\,E$, тригер встановлюється в початковий стан, а конденсатор швидко розряджається через діод VD та низькоомний вихід тригера Q. Після цього пристрій готовий до наступного запуску.

Тривалість вихідного імпульсу з достатньою для практики точністю може бути визначена за формулою:

$$\tau_i = 0{,}7\,(R + R_{\text{В}})\,C\,.$$

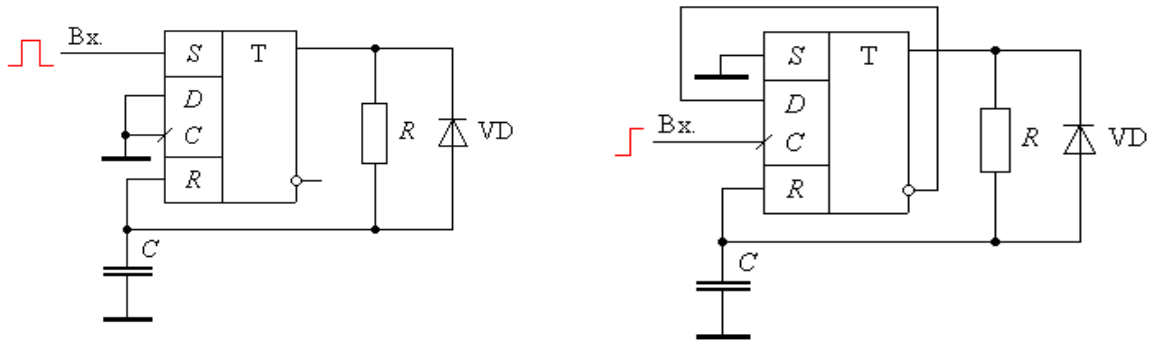
Оскільки при переході виходу Q з **“0”** в **“1”** конденсатор розряджений, то максимальне значення вихідного струму обмежується лише опором резистора R. Тому, виходячи з величини максимально допустимого значення струму виходу тригера, нижня межа опору резистора R не може бути меншою, ніж 20 кОм.

Подібні обмеження повинні мати місце і при розряді конденсатора, на практиці це забезпечується установкою обмежуючого резистора послідовно з VD.

Приклад 10.4. Розробити схеми одновібратора на основі *D*-тригера КР1554ТМ2 (74AC74).

Розв'язання. *D*-тригер ТМ2, як відомо, є динамічним тригером з керуванням за фронтом синхросигналу і асинхронними *R* та *S* прямими входами. Тому є можливість побудови декількох варіантів схем одновібраторів, які відрізнятимуться лише вхідними колами.

На рис. 10.25, *а – б* приведені два варіанти таких схем. У першій з них запуск одновібратора забезпечується подібно до схеми з асинхронним *RS*-тригером. У другій використовується режим *T*-тригера.



а) *б)*

Рис. 10.25

10.3. Генератори прямокутних імпульсів

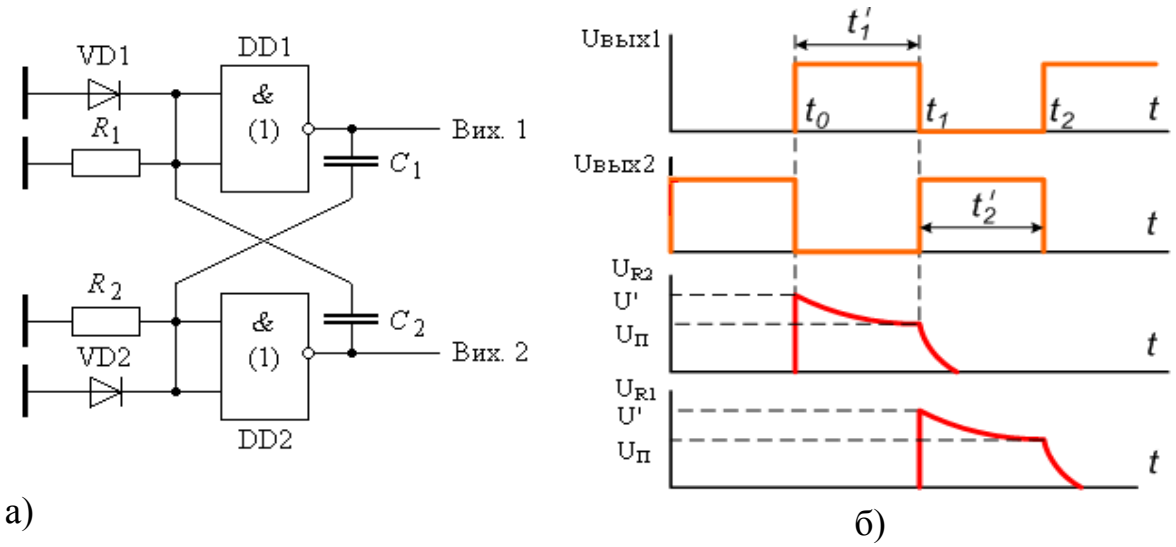
Генератори прямокутних імпульсів або *мультивібратори* – це пристрої, призначені для генерації прямокутних імпульсів заданої частоти.

Основним показником якості роботи генераторів імпульсів є стабільність генерації коливань, яка оцінюється коефіцієнтом відносної нестабільності частоти: $K_f = \frac{|f - f_0|}{f_0} = \frac{|\Delta f|}{f_0}$, де f, f_0 – відповідно поточне і початкове значення частоти.

До дестабілізуючих факторів відносяться коливання напруги живлення, часова зміна параметрів використовуваних елементів (наприклад, старіння), вплив температури та ін.

Для зниження відносної нестабільності K_f використовується широка гама засобів, які надають можливість забезпечити допустиму нестабільність для кожного конкретного випадку. Генератори, в яких не передбачаються засоби стабілізації частоти, мають K_f у межах $(1\ldots3)\cdot10^{(-3\ldots-4)}$. Кварцові генератори без прийняття спеціальних івстабілізації частоти мають $K_f \approx 10^{-6}$.

Розглянемо роботу мультивібратора на прикладі одного з найпростіших генераторів, схема якого приведена на рис. 10.26.



а) б)

Рис. 10.26

Припустимо, що на виході ЛЕ DD1 присутній високий потенціал. Тоді на його вході повинен бути нульовий сигнал, а це означає, що через резистор R_1 або зовсім не протікає струм, або його величина настільки незначна, що падіння напруги на R_1 від його протікання менше порогового рівня DD1. При використанні КМОН ІС вхідний струм мікросхеми можна вважати нульовим. Це означає, що у колі – **Вих. 2**, C_2, R_1, загальна шина – струм не протікає.

Відсутність струму в приведеному колі можлива лише при низькому потенціалі виходу **Вих. 2**, що забезпечується тим, що падіння напруги на R_2

відповідає рівню логічної **“1”**. При цьому у колі – **Вих. 1**, C_1, R_2, загальна шина – протікає струм заряду конденсатора відповідно до рівняння:

$$u_C(t) = E\left[1 - \exp\left(-\frac{t}{R_2 C_1}\right)\right], \qquad (10.7)$$

де E – напруга живлення.

У довільний момент часу напруга на R_2:

$$u_{R_2}(t) = E - u_C(t) = E\exp\left(-\frac{t}{R_2 C_1}\right) \qquad (10.8)$$

і зменшується при зарядці конденсатора.

У момент часу t_1 напруга U_{R2} зменшується до порогового рівня $U_П$ логічного елемента DD2. В результаті стан цього логічного елемента зміниться, і на його виході з’явиться сигнал високого рівня. Тепер в колі – **Вих. 2**, C_2, R_1, загальна шина – з’явиться струм заряду конденсатора C_2, тривалість інтервалу t'_2 якого також визначається формулою (10.8). Як результат, період коливань $T = t'_1 + t'_2$, де t'_1 і t'_2 визначаються за формулою (10.8) при $U_{R_2} = U_п$:

$$t_{1',2'} = R_2 C_1 \ln\frac{U_п}{E}.$$

При $R_1 = R_2$, $C_1 = C_2$, а також враховуючи, що для КМОН ІС величина $U_п \approx E/2$, знаходимо: $t_{1',2'} = 0{,}7\,RC$.

З проведеного аналізу зробимо деякі висновки.

Робота мультивібратора характеризується двома тимчасово стійкими станами. Для розглянутої схеми це стан, коли виходи **Вих. 1** та **Вих. 2** мають рівні сигналів **“1”**, **“0”** або **“0”**, **“1”**. Тривалість обох станів однакова і в сумі визначає період генерації коливань. Зміна станів відбувається протягом дуже короткого інтервалу часу, практично миттєво (при розгляді процесу перемикання конденсатори C_1 і C_2 можна не враховувати, і тоді процес перемикання подібний до тригерного). Такий процес перемикання, який характеризується тимчасовим переходом мікросхем в активний режим,

наявністю позитивного зворотного зв'язку між мікросхемами і високим контурним коефіцієнтом підсиленням кожної з них, називається *регенеративним*. Він закінчується тим, що кожна з мікросхем переходить у режим насичення (або запирання) з низьким коефіцієнтом підсилення.

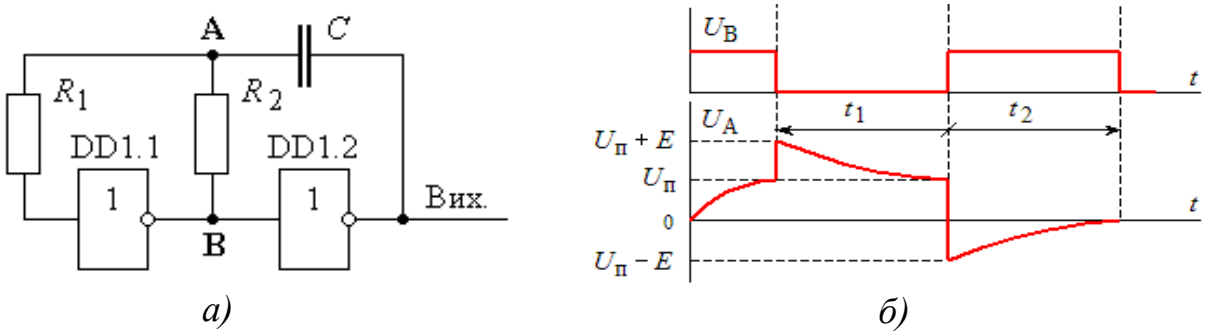
Частота генерації задається часозадаючими елементами, в якості яких використанні два *RC*-кола.

У мультивібраторах може використовуватись одне *RC*-коло, *LC*-контур або кварцовий резонатор. Зміною постійних часу *RC*-кола чи параметрів *LC*-контуру можна забезпечити керовану зміну частоти генерації. При використанні компонентів, керованих напругою або струмом, можна будувати генератори, керовані напругою або струмом. Такі генератори здебільшого мають досить низьку стабільність фіксованої частоти, а також обмежений діапазон регулювання.

10.3.1. Мультивібратор на основі логічних елементів

Оскільки в мультивібраторах логічні елементи використовуються як інвертори, то реально не має значення, які з них беруться для побудови генераторів. Найбільш широке використання знаходять пристрої з однією часозадаючою ланкою, оскільки це спрощує схемотехніку, забезпечує більшу стабільність, полегшує розв'язання задач зміни робочої частоти.

Розглянемо особливість роботи мультивібратора на прикладі схеми, приведеної на рис. 10.27.



а) *б)*

Рис. 10.27

При вмиканні напруги живлення конденсатор C розряджений. На виході може встановитись високий $U_{\text{вих}} = U^1 = E$ або низький $U_{\text{вих}} = U^0 = 0$ потенціал. На входах і виходах ЛЕ DD1.1 і DD1.2 рівні сигналів відповідатимуть логічним зв'язкам.

Припустимо, що при вмиканні напруги живлення $U_{\text{вих}} = U^0 = 0$. Тоді потенціал точки **В** $U_{\text{В}} \approx E$, а $U_{\text{А}} \approx 0$. В результаті маємо коло, по якому протікатиме струм зарядки конденсатора: від джерела живлення через вихід ЛЕ DD1.1, резистор R_2, конденсатор C, вихід DD1.2 на загальну шину джерела. Конденсатор заряджатиметься з (+) полярністю в точці **А**. Оскільки вхідний струм КМОН ІС відсутній, то потенціал лівої обкладки конденсатора прикладається до входу DD1.1. Коли напруга на конденсаторі досягне порогового рівня $U_{\text{П}}$ мікросхеми DD1.1, її стан зміниться на протилежний, потенціал точки **В** прийме значення $U_{\text{В}} = 0$, а на виході встановиться високий рівень напруги. Як наслідок, потенціали виходу і конденсатора додаються, і потенціал точки **А** складатиме: $U_{\text{А}} = E + U_{\text{п}} \approx 1{,}5\,E$.

Завдяки резистору R_2 потенціал на вході DD1.1 буде обмеженим величиною: $E + U_{\text{VD}} \approx E + 0{,}7$ В, де $U_{\text{VD}} \approx 0{,}7$ В – падіння напруги на діоді вхідної захисної ланки інвертора.

Зміна станів ЛЕ приводить до перезарядки конденсатора, тепер уже з позитивним потенціалом на правій обкладці. Потенціал точки **А** при цьому визначатиметься різницею потенціалів конденсатора і виходу DD1.2, тобто

$$U_{\text{А}} = E - U_{\text{п}} \approx 0{,}5\,E.$$

На рис. 10.27, *б* приведені часові діаграми напруги $U_{\text{а}}$ (коливання напруги на конденсаторі), а також напруг $U_{\text{вих}}$ і $U_{\text{б}}$.

Опір резистора R_1 повинен перевищувати опір резистора R_2 і вибирається в діапазоні $2\,R_2 \le R_1 \le 10\,R_2$. При виконанні цих умов інтервали часу t_1 і t_2 визначаються за допомогою формул [Зельд.]:

$$t_1 = - R_2 C \ln\left[\frac{U_{\text{П}}}{E + U_{\text{П}}}\right];$$

$$t_2 = - R_2 C \ln\left[\frac{E - U_{\text{П}}}{2E - U_{\text{П}}}\right];$$

а частота коливань $f = \frac{1}{T}$, $T = t_1 + t_2$.

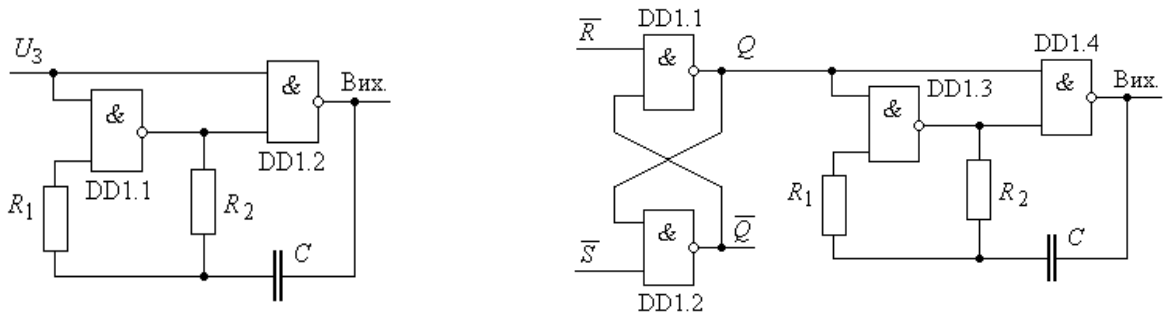
У приведених формулах при стабільній пороговій напрузі $U_{\text{п}} \approx 0{,}5\,E$ період T не буде залежати від коливань напруги живлення і може визначатись за спрощеною формулою: $T = 2{,}2\,R_2\,C$.

При побудові мультивібратора, як вже відмічалося, можуть бути використані будь-які логічні елементи, що забезпечують інвертування вхідного сигналу (ЛЕ **І-НІ**, **АБО-НІ**, **ВИКЛ. АБО** і т. п.).

Допоміжні входи використовуваних ЛЕ можуть виконувати функції керуючих, запускаючих (старт-стопних) або мати інші призначення. Як приклад, на рис. 10.28, *а – б* приведені дві схеми, в яких забезпечується керування процесом генерації.

У схемі мультивібратора, яка приведена на рис. 10.28, *а*, наявність запускаючого сигналу $U_3 = 1$ забезпечує безперервну роботу пристрою. Для зупинки процесу генерації необхідно подати сигнал $U_3 = 0$.

У схемі, що приводиться на рис. 10.28, *б*, запуск мультивібратора здійснюється шляхом подачі імпульсу низького рівня на вхід $\overline{S}$. Для зупинки роботи мультивібратора, подається імпульс низького рівня на вхід $\overline{R}$.



а) ***б)***

Рис. 10.28

У великій кількості практичних випадків виникає необхідність зміни коефіцієнту заповнення імпульсів $\gamma = t_i/T = t_1/(t_1 + t_2)$. Для розглянутої схеми мультивібратора зміна коефіцієнту заповнення може бути досягнута шляхом зміни постійних часу в інтервалах t_1 і t_2. Це може бути забезпечено розділенням шляхів заряду та розряду конденсатора, наприклад, за допомогою діодних кіл.

На рис. 10.29 приведена схема мультивібратора, в якій регулювання інтервалів часу t_1 і t_2 забезпечується зміною величин опорів $R_2 + R'_3$ та $R_4 + R''_3$.

Зміна положення повзунка змінного резистора R_3 відносно його середини забезпечує збільшення (зменшення) опору зарядного (розрядного) резистора і тим самим збільшує одну постійну часу і зменшує іншу.

Приклад 10.5. Розробити принципову схему мультивібратора з регулюванням частоти за допомогою керуючої напруги.

Розв'язання. Для побудови такого мультивібратора необхідно використовувати опір, керований напругою. В якості такого можна використати польовий транзистор, опір каналу якого на деяких ділянках вихідних характеристик має лінійну залежність від напруги на затворі U_3. Для обмеження опору каналу паралельно транзисторному регулятору встановлюється допоміжний резистор R_3.

Схема мультивібратора з використанням польового транзистора в якості резистора часозадаючої ланки приведена на рис. 10.30.

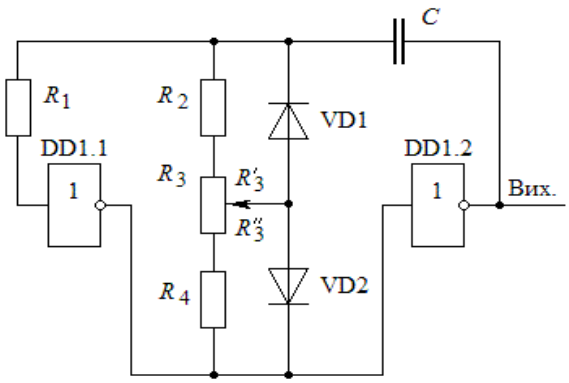


Рис. 10.29

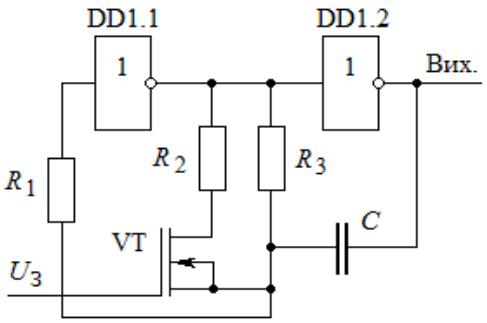


Рис. 10.30

Можливості вдосконалення схемотехніки мультивібраторів досить широкі і різноманітні, тому в літературі можна знайти досить велику кількість схем, які в цілому за принципом роботи мало чим відрізняються.

Мультивібратори ТТЛ будуються за тим же принципом, але суттєвою різницею в них є те, що порогові рівні елементів мають значний розкид, величини вхідних струмів елементів різні для різних рівнів сигналів. Тому стабільність таких генераторів з часозадаючими *RC*-ланками значно нижча, ніж з використанням КМОН ІС.

10.3.2. Мультивібратори на основі тригерів

Звернемось до схеми мультивібратора з двома часозадаючими ланками (рис. 10.26). Оскільки ЛЕ **2І-НІ** в ній виконують лише функції інверторів, то їх можна замінити на ЛЕ **2АБО-НІ**.

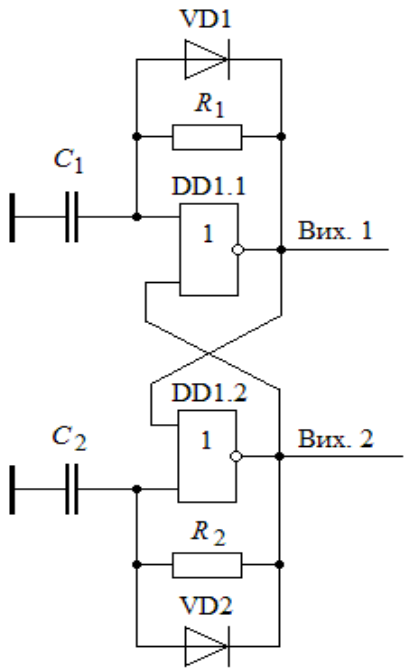


Рис. 10.31

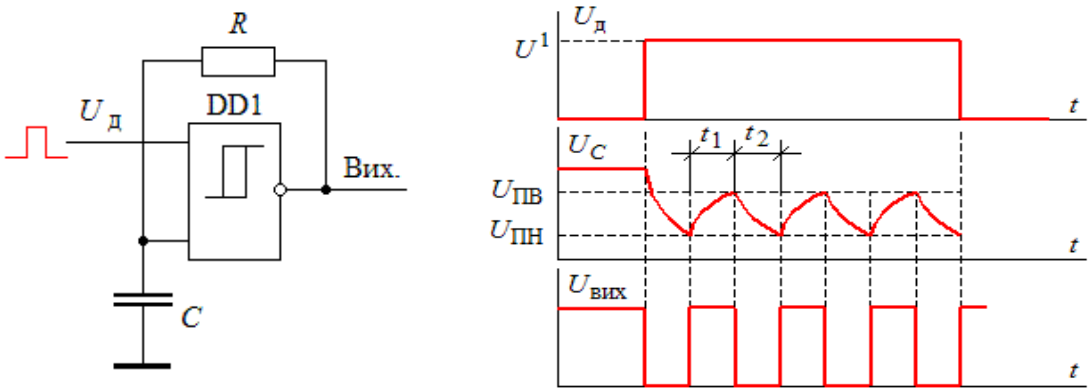
Оскільки для кожного часового інтервалу використовується окрема часозадаюча ланка, не принципово де будуть розміщенні опори з шунтуючими діодами і конденсатори. Виходячи з цього, можемо побудувати схему мультивібратора, що відповідає рис. 10.31.

Зі схеми видно, що мультивібратор складається з *RS*-тригера на базі ЛЕ **2АБО-НІ** та часозадаючих ланок. Це дає підстави стверджувати, що мультивібратори, подібно до одновібраторів, можуть бути побудовані на основі будь-якого тригера з двома незалежними входами і рівнозначними станами або на основі несиметричного тригера з двома порогами спрацьовування та однією часозадаючою ланкою (на основі тригера Шмідта).

Частота генерації коливань мультивібратора, схема якого приведена на рис. 10.31, визначається за формулою: $f = \dfrac{1}{0{,}7\,(R_1 C_1 + R_2 C_2)}$.

Тригери Шмідта КМОН-технологій (К561ТЛ1, 564ТЛ1, КР1561ТЛ1) мають логічний елемент **2І** на входах та інверсний вихід. Це не тільки суттєво спрощує схемотехніку мультивібраторів, а й надає можливість забезпечувати керування процесом генерації. На рис. 10.32, *а* приведена схема мультивібратора на основі тригера Шмідта з інверсним виходом, а на рис. 10.32, *б* – часові діаграми, що пояснюють його роботу.

При побудові мультивібраторів на мікросхемах КМОН слід враховувати, що різниця між верхнім і нижнім пороговими рівнями залежить від напруги живлення. При напрузі живлення E = 5 В різниця між ними складає 0,6 В; при E = 10 В - збільшується до 2 В.



а) *б)*

Рис. 10.32

При низькому рівні дозволяючої напруги $U_д = U^0$ на вході DD1, вихідна напруга $U_{вих} = U^1$, а конденсатор C заряджений до $U_C = U^1$. При $U_д = U^1$ вихідна напруга зменшується до величини $U_{вих} = U^0 = 0{,}8\ldots1$ В (у залежності від напруги живлення), і конденсатор починає розряджатись. Як тільки напруга на конденсаторі U_C зменшиться до нижнього порогового рівня ($U_C = U_{ПН}$), стан виходу тригера зміниться на протилежний ($U_{вих} = U^1$) і почнеться процес зарядки конденсатора C. При $U_C = U_{ПВ}$ тригер знову змінить свій стан, і цикл повторюватиметься.

Інтервали часу зарядки та розрядки конденсатора t_1 і t_2 будуть однаковими, оскільки для обох інтервалів використовується одна часозадаюча ланка, тому:

$$T = t_1 + t_2 = 2\,R\,C \ln \frac{U_{\text{ПВ}}}{U_{\text{ПН}}}.$$

Оскільки вихідні струми $\left| I^{1}_{\text{вих}\,m} \right| = \left| I^{0}_{\text{вих}\,m} \right|$ однакові, то величина опору R обмежується значенням $R_{min} = E / I^{1}_{\text{вих}\,m}$ і залежить від напруги живлення. Верхня межа опору R обмежується величиною вхідного струму $I_{\text{вх}}$, який повинен як мінімум на порядок бути меншим мінімального зарядного струму конденсатора. Ємність конденсатора повинна не менш ніж на порядок перевищувати паразитні ємності мікросхем і монтажу.

Регулювання частоти і скважності імпульсів може забезпечуватись тими ж способами, що і в мультивібраторах на основі логічних елементів.

При використанні тригерів Шмідта ТТЛ схема мультивібратора залишається незмінною. Особливість її роботи полягає лише в тому, що зарядка конденсатора відбувається не лише через резистор R, а й завдяки наявності вхідного струму. Тому при виборі резистора R цю особливість слід враховувати, а також зважати на той факт, що тривалості імпульсу та паузи можуть бути різними.

10.3.3. Мультивібратори на основі мікросхем-одновібраторів

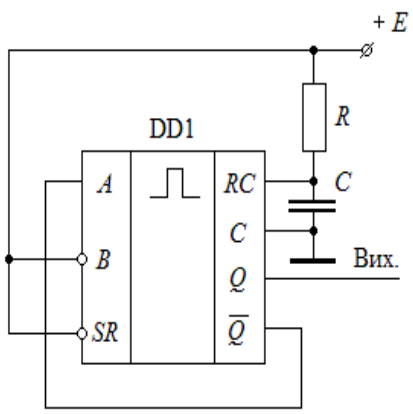


Рис. 10.33

Спеціалізовані мікросхеми одновібраторів (АГ1, АГ3) мають високі експлуатаційні характеристики, які надають можливість будувати високостабільні мультивібратори. Можливість побудови мультивібратора на основі одновібратора витікає з таблиці станів мікросхем (табл. 10.1, режим “запуск”). Оскільки, на відміну від одновібратора, запуск генератора

повинен забезпечуватись не від зовнішнього джерела, а автоматично, то схемотехніка варіантів мультивібраторів визначатиметься способами автоматичного запуску.

З табл. 10.1 витікає один зі способів автоматичного запуску – на основі зворотного зв'язку. В режимі "запуск" на входи $\overline{SR}$ і $\overline{B}$ необхідно подати високий рівень сигналу, а на вхід A — запускаючий імпульс з інверсного виходу $\overline{Q}$. Звідси і витікає проста схема мультивібратора, що зображена на рис. 10.33. Особливість приведеної схеми генератора полягає в тому, що тривалість імпульсу задається RC-ланкою, а повторний запуск забезпечується з малою внутрішньою затримкою після завершення імпульсу. Тому незалежне регулювання періоду і коефіцієнту заповнення імпульсів у такій схемі неможливе. Забезпечити таку можливість дає використання двох одновібраторів, перший з яких працює в автогенераторному режимі, а другий – у режимі очікування.

Використання двох і більшої кількості одновібраторів та наявність у них двох керуючих входів відкриває можливості побудови багатофазних схем мультивібраторів, з'єднуючи їх послідовно у кільце. Такі генератори виробляють симетричні послідовності імпульсів – прямі та інверсні, забезпечують регулювання коефіцієнту заповнення та частоти в широкому діапазоні, генерують зміщені на заданий інтервал часу послідовності імпульсів, забезпечуючи при цьому високу стабільність вихідних параметрів. Приклад схеми такого багатофазного мультивібратора з використанням ІС ТТЛ АГ3, приведений на рис. 10.34.

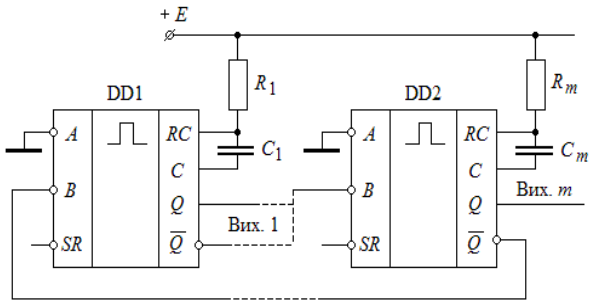
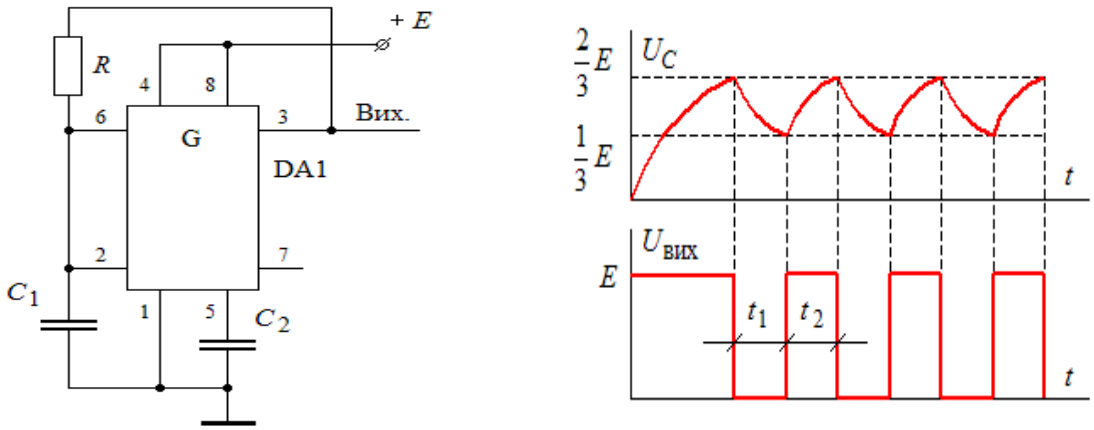


Рис. 10.34

10.3.4. Мультивібратори на основі таймера КР1006ВИ1



а) б)

Рис. 10.35

Для використання таймеру в якості генератора прямокутних імпульсів з однією часозадаючою ланкою необхідно, як витікає з попередньої теми, перетворити його в тригер Шмідта. Найпростіший спосіб перетворення – це з'єднання входів 2 і 6 обох компараторів (див. рис. 10.22). При цьому нижній компаратор визначатиме нижній пороговий рівень напруги UПН , а верхній – відповідно, UПВ . Мультивібратор матиме одну часозадаючу ланку, параметри якої можуть легко змінюватись. Приклад схеми мультивібратора приведений на рис. 10.35.

Оскільки в момент подачі напруги живлення конденсатор C1 розряджений, то на входах 2 і 6 матимемо напругу низького рівня. На виході 3 встановиться напруга високого рівня, і конденсатор C1 почне заряджатися через резистор R. При досягненні на ньому напруги UC = 2E / 3 спрацює компаратор DA1 (рис. 10.22) і переключить тригер DD1 у стан, при якому на виході 3 встановиться низький рівень (Uвих = 0).

Конденсатор C_1 почне розряджатись з тією ж часовою сталою. Коли напруга на ньому знизиться до U_C = E / 3, компаратор DA2 сформує сигнал на зворотнє перемикання тригера, і на виході (3) знову встановиться високий рівень напруги.

Період коливань визначається з умови: $U^{1}_{\text{в и х}} = E$; $U^{0}_{\text{вих}} = 0$. Вважаючи інтервали часу однаковими $t_1 = t_2$, період коливань може бути визначений за формулою: $T = t_1 + t_2 \approx 0{,}7\, R\, C_1 + 0{,}7\, R\, C_1 = 1{,}4\, R\, C_1$.

Реально у розглянутій схемі $U^{1}_{\text{вих}} < E$ на величину падіння напруги на відкритому транзисторі ($U_{\text{КЕ}} \approx 0{,}6 \ldots 0{,}9$ В). Тому інтервал часу заряду t_2 буде дещо більшим, ніж інтервал розряду t_1. Для усунення цього недоліку використовують описаний у **Розділі 2** спосіб "підтягування" напруги $U^{1}_{\text{вих}}$ до E. Це досягається шляхом підключення між джерелом живлення E і виходом **3** допоміжного резистора, який зменшує і стабілізує величину падіння напруги на транзисторі.

Розглянута схема, як і попередні, має недолік, суттю якого є залежність частоти коливань від величини навантаження, оскільки зменшення опору навантаження призводить до зниження вихідної напруги. Забезпечити ж незалежність частоти і навантаження можливо в тих пристроях, в яких навантаження не впливає на роботу часозадаючих кіл.

Прикладом такого мультивібратора є схема, приведена на рис. 10.36.

Струм зарядки конденсатора протікає від джерела живлення E через резистори R_1 та R_2 при закритому транзисторі VT3 (рис. 10.22). Напруга на виході **3** має рівень U^{1}. У момент часу, коли напруга на конденсаторі C_1 стає $U_C = 2E / 3$, внутрішній тригер змінює свій стан, і відкривається транзистор VT1 (рис.10.22). Конденсатор починає розряджатись через резистор R_2 і транзистор VT1. При зменшенні напруги до $U_C = E / 3$ відбувається чергове перемикання тригера, транзистор VT1 закривається, і

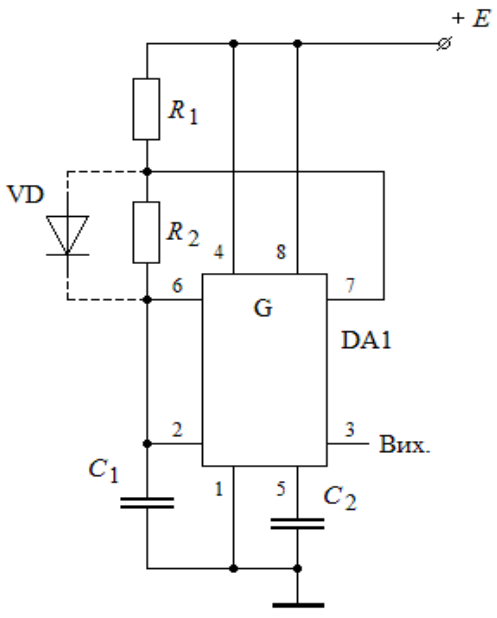


Рис. 10.36

починається новий цикл зарядки конденсатора. Таким чином, час зарядки приблизно визначається як: $t_З \approx 0{,}7\,(R_1 + R_2)\,C_1$, а час розрядки – як: $t_Р \approx 0{,}7\,R_2\,C_1$.

Для забезпечення однакової тривалості імпульсу та паузи можна зашунтувати резистор R_2 діодом (див. рис. 10.36). З використанням таймера КР1006ВИ1 розроблена велика кількість схем генераторів різного призначення. Багато з них описані в спеціальній літературі.

10.3.5. Кварцові генератори

Кварцові генератори знаходять досить широке використання в цифровій схемотехніці, оскільки забезпечують генерацію високостабільних імпульсних коливань на різних частотах.

Відомо[Зельдин], що кварц як електричний прилад має схему заміщення, що приведена на рис. 10.37.

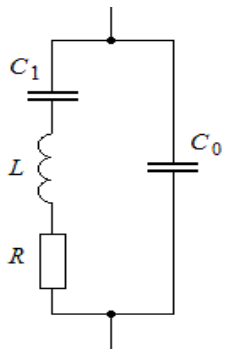


Рис. 10.37

Але особливість кварцу проявляється не в самій схемі заміщення, а в її параметрах. Так, наприклад, для кварцу з основною резонансною частотою 2 МГц маємо: R = 100 Ом; L = 0,52 Гн; C_1 = 0,012 пФ; C_0 = 4 пФ; добротність $Q = 54 \cdot 10^3$. Тобто кварц як коливальний контур може забезпечити недосяжну для дискретних компонентів величину добротності.

Кварцовий резонатор має дві резонансні частоти, при яких повний опір має активний характер. Перша – частота послідовного резонансу:

$$f_1 = \frac{1}{2\pi\sqrt{LC_1}};$$

друга – частота паралельного резонансу:

$$f_2 = \frac{1}{2\pi\sqrt{L\,\dfrac{C_0\,C_1}{C_0 + C_1}}}.$$

Оскільки $C_0 >> C_1$, то $\frac{C_0 C_1}{C_0 + C_1} \approx C_1$ і різниця між частотами f_1 та f_2 досить мала. В діапазоні між вказаними двома частотами опір кварцу носить індуктивний характер і використовується безпосередньо для побудови кварцових генераторів як на основі послідовного, так і паралельного резонансів.

При використанні послідовного резонансу кварц застосовується або як безпосередньо резонансний контур, або як індуктивний опір у діапазоні частот f_1 та f_2. Узагальнені схеми таких генераторів приведені на рис. 10.38.

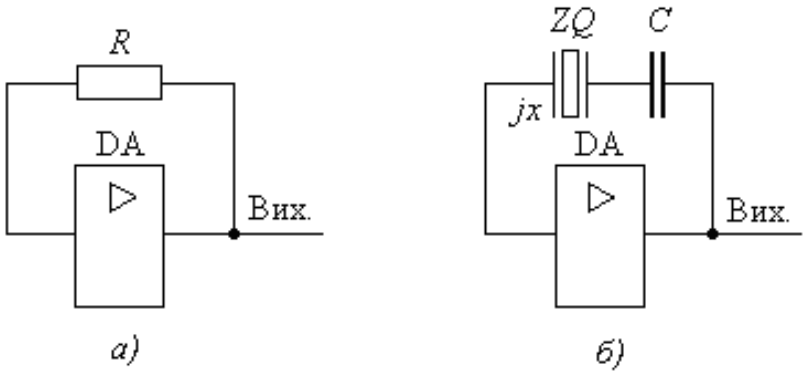


Рис. 10.38

У схемі рис. 10.38, *а* кварц на резонансній частоті має нульовий фазовий зсув, тому коло зворотного зв'язку не вносить фазових спотворень. Підсилювач, який в подібних схемах може складатися з двох інверторів, також не повинен вносити фазових спотворень.

У схемі, приведеній на рис. 10.38, *б*, кварц діє як індуктивність, а конденсатор *C* компенсує фазовий зсув, що вноситься індуктивним опором *jx* кварцу. Така схема дає можливість підвищувати частоту генерації коливань порівняно з власною резонансною частотою кварцу. Змінюючи величину ємності конденсатора, можливо у незначних межах забезпечувати регулювання частоти генерації.

Узагальнені схеми генераторів з використанням паралельного резонансу приведені на рис. 10.39.

В обох варіантах схем генератор містить інвертуючий підсилювач, який забезпечує зсув фази генерованого сигналу на 180°, і фазообертаючий контур, що також забезпечує зсув фази на 180°. В обох випадках кварц використовується як індуктивний елемент з опором jx, створюючи з конденсаторами C_1 та C_2 (рис. 10.39, *а*) або з конденсатором C та індуктивністю L (рис. 10.39, *б*) коливальний контур з резонансною частотою $f_\text{р}$.

Розглянуті схеми генераторів іноді називають **генераторами з позитивним реактивним опором**.

На відміну від генераторів синусоїдальних коливань, які виготовляються з використанням високочастотних транзисторів та широкого набору засобів стабілізації частоти, імпульсні генератори мають значно більшу нестабільність частоти ($10^{-4}…10^{-5}$ проти $10^{-7}…10^{-8}$).

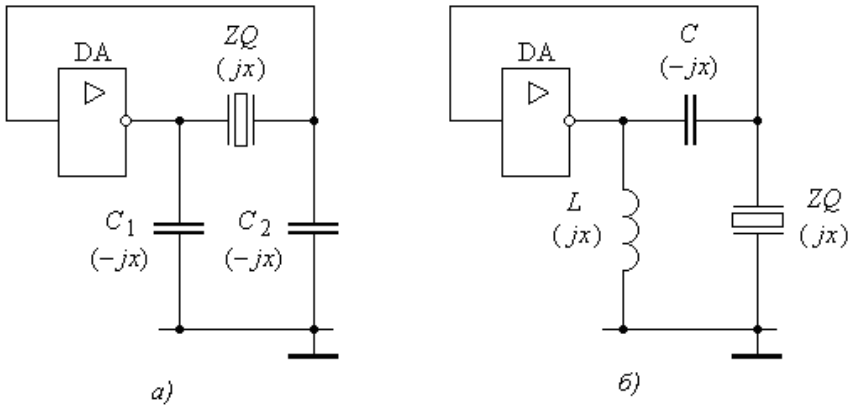


Рис. 10.39

При використанні двох ЛЕ один з них переводиться в активний режим для забезпечення необхідного коефіцієнту підсилення, тобто умов збудження. Безпосередньо генератор відокремлений від навантаження буферним каскадом, який підвищує стабільність коливань і одночасно формує фронти вихідних імпульсів.

Розглянемо декілька прикладів практичних схем генераторів.

Одним з найпростіших є генератор, схема якого приведена на рис. 10.40.

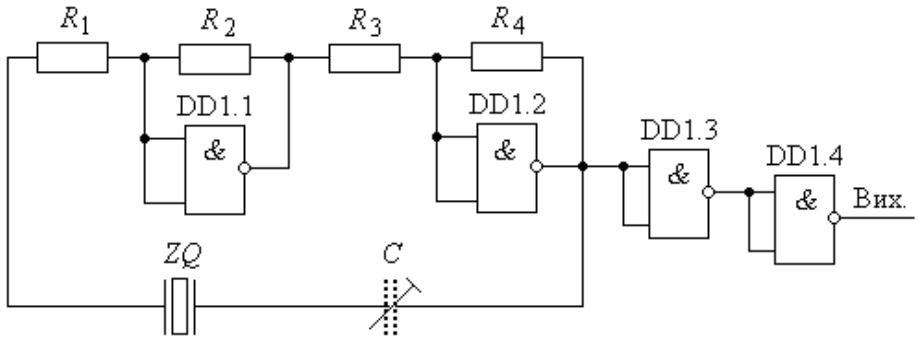


Рис. 10.40

Пристрій працює на власній частоті послідовного резонансу кварцу. ЛЕ DD1.1 і DD1.2 працюють в активному режимі, який забезпечується резисторами $R_1 \ldots R_4$. ЛЕ DD1.3 і DD1.4 виконують функції буферного підсилювача. Більш широко приведена схема використовується з допоміжним конденсатором C, який змінює режим роботи кварцу з резонансного на режим позитивного реактивного опору. Відношення опорів резисторів R_1/R_2 та R_4/R_3 повинно перевищувати **1**. Фазовий зсув підсилювача 0° або 360° забезпечується в широкому частотному діапазоні.

Кварцовий генератор з використанням паралельного резонансу ілюструється схемою, що приведена на рис. 10.41.

Завдяки резистору R_2 ЛЕ DD1.1 працює в активному режимі, а коефіцієнт підсилення забезпечується відношенням опорів резисторів R_2/R_1 і повинен бути більшим **1.**

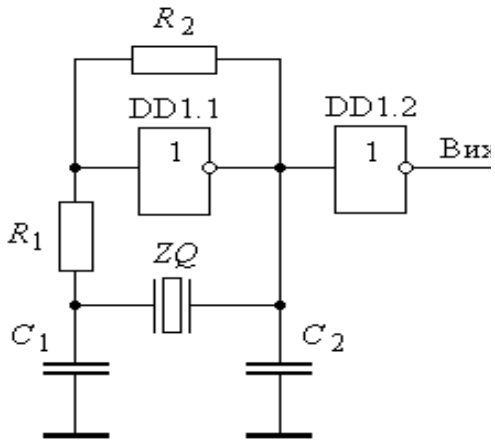


Рис. 10.41

Паралельний контур утворюється кварцем ZQ, який працює в режимі позитивного реактивного опору, і компенсуючими конденсаторами C_1 та C_2. Підсилювач на ЛЕ DD1 забезпечує зсув фази на 180°. Ланка зворотного зв'язку ZQ, C_1, C_2 також забезпечує зсув фази на 180°. Таким чином забезпечуються всі

умови для генерації коливань. ЛЕ DD1.2 виконує функцію буферного каскаду. Виготовлений на мікросхемах ТТЛ або КМОН-технологій, генератор може працювати в частотному діапазоні, в якому безпосередньо мікросхеми не вносять значного фазового зміщення. Нестабільність частоти для генераторів на КМОН ІС може досягати величини $5 \cdot 10^{-5}$, а на ТТЛ ІС – 10^{-4}.

Кварцові генератори можуть будуватися за схемами звичайних мультивібраторів. На рис. 10.42 приведена схема кварцового генератора, яка практично повністю співпадає зі схемою мультивібратора (рис. 10.27).

У цій схемі кварц працює в режимі послідовного резонансу, а ЛЕ DD1.3 є буферним елементом і може також виконувати функцію керованого ключа.

Враховуючи низький вхідний опір ЛЕ на біполярних транзисторах, такі генератори можуть працювати на частотах, які обмежуються лише частотними властивостями безпосередньо логічних елементів. Кварци для таких генераторів повинні мати досить високу добротність.

Приклад 10.6. На рис. 10.43 приведена схема кварцового генератора. Пояснити принцип його роботи. Пояснити особливості вибору опорів ризисторів і ємності конденсатора. Обґрунтувати існування умов збудження коливань.

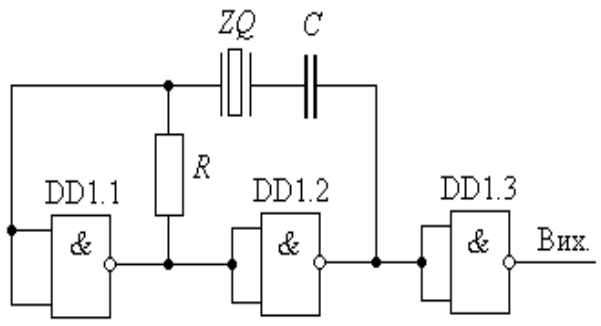


Рис. 10.42

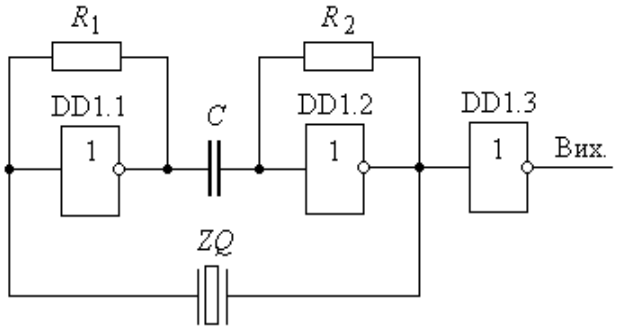


Рис. 10.43

Пояснення. Логічні елементи DD1.1 і DD1.2 охоплені від’ємним зворотним зв’язком за допомогою резисторів R_1 та R_2, тому вони працюють в лінійному режимі з коефіцієнтом підсилення більшим **1**. Якщо на частоті генерації конденсатор C разом з вхідним опором DD1.2 не впливатиме на величину фазового зсуву, то інвертуючі підсилювачі

на DD1.1 і DD1.2 забезпечать фазовий зсув в 360^0 і коефіцієнт підсилення більший **1**, тобто умови збудження будуть виконаними.

Конденсатор C разом із вхідним опором може створювати диференціюючу ланку. Якщо вибрати його ємність досить великою, щоб вона виконувала функцію розділюючої, то вносити помилку в фазовий зсув генерованого сигналу конденсатор не буде. Для цього необхідно виконання умови: $\left(R_{\text{вх}}C\right)^{-1} << f_{\text{г}}$, де $R_{\text{вх}}$ – вхідний опір DD1.2; $f_{\text{г}}$ – частота генерації коливань. Опори резисторів R_1 та R_2 повинні мати однакові значення і забезпечувати величину вхідного струму для зміщення інвертора в активний режим. Для ТТЛ ІС цей опір може мати величину від 400 Ом до 2-3 кОм. Кварц ZQ в схемі працює в режимі послідовного резонансу.

Кварцові генератори, побудовані на основі таймерів, мають свої переваги порівняно з описаними. Вони не мають допоміжних буферів, стійко працюють при зміні напруги живлення та зовнішніх факторів в широких межах.

На рис. 10.44 зображена схема такого кварцового генератора.

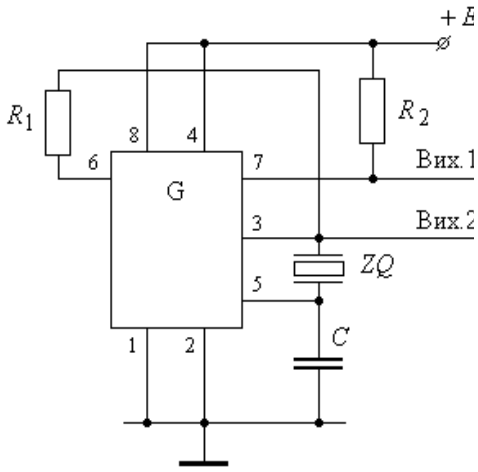


Рис. 10.44

Збудження коливань забезпечується колом від входу **6**, через резистор R_1, кварц ZQ, вхід **5** до конденсатора C.

Недоліком генераторів на базі таймерів є досить вузький частотний діапазон, що не сягає 1 МГц.

10.4. Універсальні генераторні мікросхеми

10.4.1. Мікросхема К1108ПП1

Мікросхема К1108ПП1 має досить широку область використання, незважаючи на те, що за функціональним призначенням вона може виконувати переважно дві функції:

- забезпечувати пропорційне перетворення однополярної вхідної напруги в частоту прямокутних імпульсів (перетворювач $U \rightarrow f$);

- перетворювати частоту вхідного сигналу в пропорційну їй напругу позитивної полярності (перетворювач $f \to U$).

Завдяки своїм функціональним властивостям мікросхема К1108ПП1 має велику кількість аналогів. Лише компанія Analog Device виготовляє декілька типів подібних мікросхем. Це AD537, AD650, AD651, AD654, які відрізняються між собою показниками точності та частотним діапазоном вихідних коливань.

На рис. 10.45 приведена спрощена схема мікросхеми, яка дає можливість пояснити спосіб перетворення напруги в частоту [Зельд.].

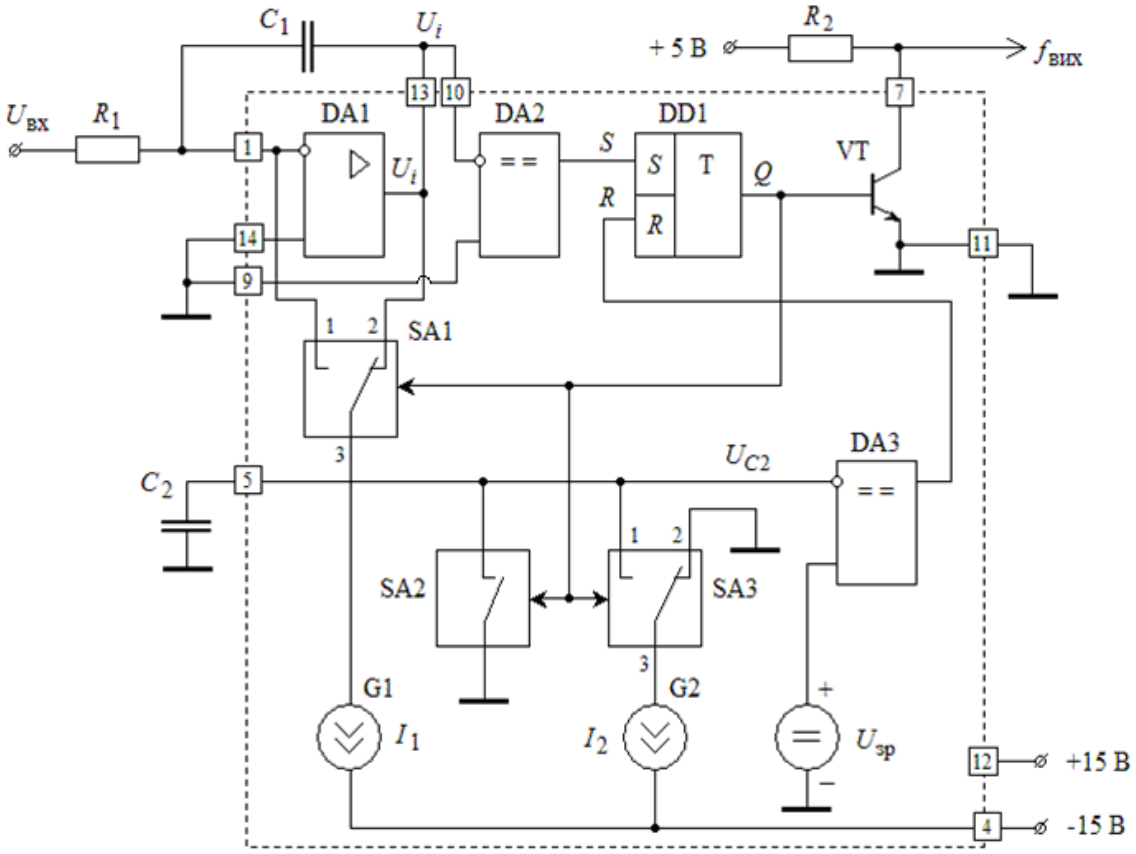


Рис. 10.45

Основою перетворювача є інтегратор, на основі операційного підсилювача DA1 та зовнішніх елементів R_1 і C_1. При постійній напрузі $U_{вх}$ на виході інтегратора (вивід **13** мікросхеми) матимемо напругу, що змінюється за лінійним законом зі швидкістю, яка задається постійною часу $R_1 C_1$, тобто:

$$U_i = -\frac{U_{вх}}{R_1 C_1} t .$$

На інтервалі часу до моменту t_1 (рис. 10.46) електронний ключ SA1 замикає контакти **2-3**, і струм I_1 від джерела струму $G1$ протікає через конденсатор C_1, зменшуючи напругу на виході інтегратора DA1. Ключ SA2 знаходиться в замкненому стані і шунтує конденсатор C_2, що приєднаний до виводу **5**. Ключ SA3 замикає контакти **2-3**, забезпечуючи замикання струму I_2 від джерела $G2$ через загальну шину.

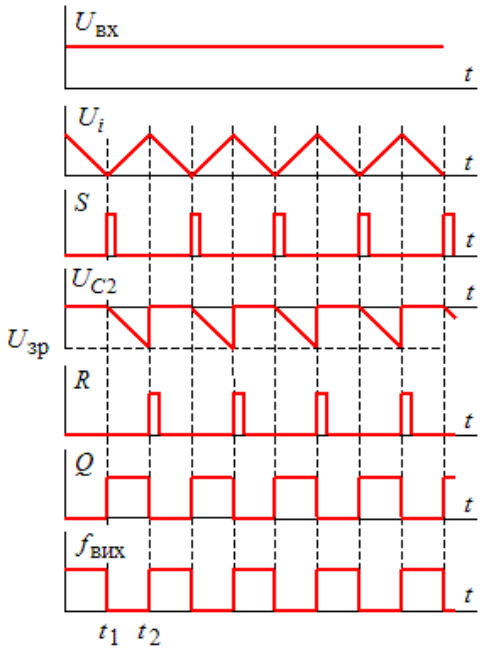


Рис. 10.46

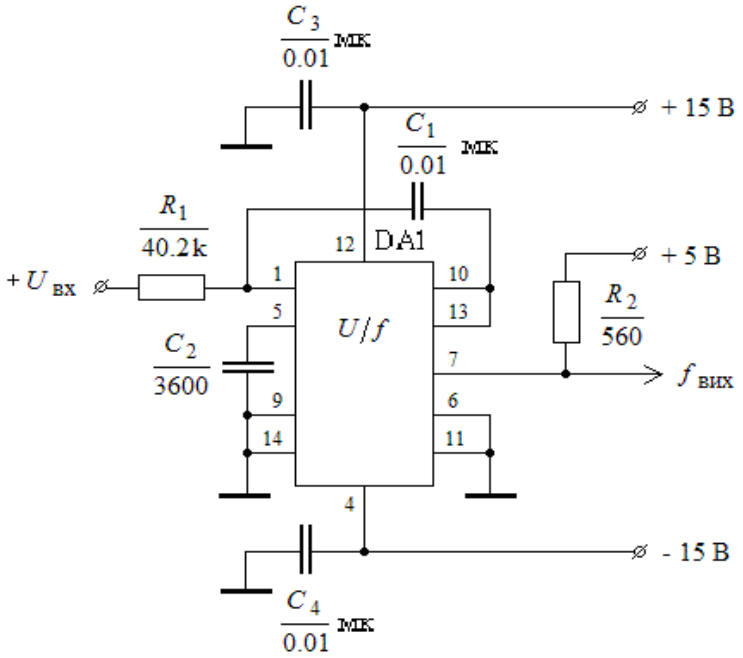


Рис. 10.47

У момент t_1 напруга U_i досягне нульового рівня, що зафіксується зміною стану компаратора DA2. Короткий імпульс S, що з'явиться на його виході, встановить тригер DD1 в одиничний стан по виходу Q, внаслідок чого стани ключів SA1…SA3 зміняться на протилежні. Ключ SA1 замкне контакти **1-3**, внаслідок чого струм $I_1 > (U_{вх}/R_1)$ від джерела $G1$ почне заряджати конденсатор C_1. Компаратор DA2 на своєму виході матиме нульовий рівень сигналу, але стан тригера DD1 при цьому не зміниться. Ключ SA2 розімкнеться і забезпечить можливість зарядки конденсатора C_2. Ключ SA3 замикає контакти **1-3** і забезпечує цим протікання зарядного струму I_2 від джерела $G2$ через конденсатор C_2. Останній почне заряджатись і зменшувати потенціал U_{C2} на інверсному вході компаратора DA3.

У момент часу t_2 при $U_{C2} = U_{ЗР}$ компаратор DA3 на своєму виході сформує короткий позитивний імпульс, який по входу *R* встановлює *RS*-тригер DD1 у початковий стан, приводячи цим в початковий стан і ключі SA1…SA3. На виході **7** мікросхеми матимемо імпульси, інвертовані по відношенню до виходу *Q* тригера DD1.

З проведеного аналізу витікає, що період імпульсної послідовності, яка генерується мікросхемою, визначається вхідною напругою $U_{вх}$ і постійною часу інтегратора, а пауза між імпульсами (інтервал t_1-t_2) − ємністю конденсатора C_2.

Частота вихідних імпульсів визначається за формулою: $f_{вих} = U_{вх} / (k\, R_1\, C_1)$, де коефіцієнт *k* має розмірність напруги, а його величина лежить в діапазоні 7...8 В [Зельд.].

На рис. 10.47 приведена принципова схема перетворювача $U \rightarrow f$ при позитивній вхідній напрузі. Параметри пасивних компонентів, що приведені на схемі, відповідають крутизні перетворення 1 кГц/В при зміні вхідної напруги в діапазоні 0...10 В.

При використанні від'ємної напруги $U_{вх}$ вона подається на прямий вхід DA1 (вивід **14** мікросхеми), а резистор R_1 приєднується до загальної шини. Діапазон вхідної напруги при цьому 0 ... -10В. Досвід роботи з мікросхемою, а також довідкові дані [Якуб.] свідчать, що при високій стабільності і точності пасивних компонентів перетворювач має високу лінійність крутизни перетворення, стабільність параметрів, низькі абсолютну та відносну похибки в діапазоні частот перетворення до 500 кГц. Високі технічні показники дозволяють використовувати мікросхему в різноманітних вимірювальних перетворювачах.

Розглянемо тепер особливість використання мікросхеми в якості перетворювача $f \rightarrow U$.

Імпульси вхідного сигналу частоти $f_{вх}$ подаються на вхід **10** компаратора DA2 (рис. 10.45). Після установки тригера DD1 у стан, при якому $Q = 1$,

починається заряд конденсатора C_2 від джерела струму $G2$ і одночасно відбувається перетворення струму I_1 від джерела $G1$ у напругу на виході DA1. Для забезпечення такого перетворення резистор R_1 приєднується паралельно конденсатору C_1, створюючи цим самим активний фільтр. Протягом всього інтервалу часу зарядки конденсатора C_2 до напруги $U_{зр}$ напруга на виході DA1 (вивід **13**) матиме величину $U_{13} = I_1 R_1$. Після спрацьовування компаратора DA3 напруга U_{13} буде поступово зменшуватись за рахунок розряду конденсатора C_1 на резистор R_1. Оскільки інтервал часу t_1–t_2 заряду конденсатора C_2 є величиною постійною, то з ростом частоти він зростатиме, і зростатиме середнє значення напруги U_{13} на виході перетворювача.

Вихідний транзистор VT мікросхеми не має внутрішнього навантаження. Це дає можливість вибирати і встановлювати зовнішній резистор R_2, а також напругу живлення, виходячи з умов узгодження із зовнішнім навантаженням.

У реальних схемах між вихідним транзистором VT і тригером DD1 встановлений буферний каскад, керуючий вхід якого приєднаний до виводу **6.** Сигнал високого рівня, що подається на вивід **6**, дає можливість блокувати роботу транзистора. Це суттєво розширює функціональні можливості мікросхеми, оскільки надає змогу з'єднувати їх паралельно по виходу.

Приклад 10.7. Розробити схему перетворювача $f \rightarrow U$ для інтервалу частот 0…10 кГц.

Розв'язання. За базову береться схема перетворювача, приведена на рис. 10.47. Відповідно до опису роботи перетворювача, сигнал вхідної частоти повинен подаватись на вхід **10**, а виводи **1** і **10** з'єднуються між собою через паралельно з'єднані R_1 і C_1, параметри яких залишаються незмінними. Вивід **13** використовується як вихід перетворювача.

Мікросхеми, які забезпечують перетворення $U \rightarrow f$ та $f \rightarrow U$, окрім високої лінійності перетворень, мають досить високі технічні характеристики по входу і виходу. Вхідний операційний підсилювач має високу стабільність коефіцієнту підсилення, незмінний в широкому діапазоні температур і зовнішніх напруг вхідний опір, що дозволяє безпосередньо приєднувати різноманітні низькорівневі датчики. Величина температурного зміщення вхідного

потенціалу жорстко регламентована, а у деяких приладах має лінійну залежність 1 мВ/°С (AD537) в широкому діапазоні температур. Вихідний транзисторний ключ забезпечує комутацію струмів у декілька десятків міліампер (для різних мікросхем у межах 10…50 мА), має високу робочу напругу, що дає можливість приєднувати до нього широкий ряд виконавчих пристроїв, світлодіоди, десятки входів ТТЛ та ін.

Прямокутна форма вихідної напруги дає можливість знизити внутрішні втрати енергії та підвищити стабільність функціонування, а також зменшити проблеми , які можливі при використанні приладу для більшості прикладних задач.

Усе це забезпечує найрізноманітніше використання мікросхеми. Перш за все, вона широко застосовується за безпосереднім призначенням. Перетворення $U \rightarrow f$ дає можливість створювати керовані генератори частоти, програмовані генератори частоти (перетворювачі код-частота). Таке використання дає можливість будувати на базі мікросхеми перетворювачі струму, температури, опору в частоту вихідних імпульсів. Це, в свою чергу, відкриває можливість на основі потенціометричних схем будувати перетворювачі лінійних і кутових переміщень у частоту вихідних імпульсів.

Надзвичайно широке використання мікросхеми знаходять в телеметрії, в задачах передачі інформації, в тому числі по оптоволоконних каналах зв’язку. В таких випадках розв’язуються проблеми невідповідності потенціалів датчика та приймача, проблеми завад та ін. Широке використання мікросхеми знаходять як генератори модульованих коливань з частотною та (рідше) широтною модуляцією. В радіотехніці та пристроях радіоавтоматики мікросхеми використовуються в генераторах дискретних сіток частот на основі фазового автопідстроювання частоти.

Мікросхеми-перетворювачі $U \rightarrow f$ широко використовуються в пристроях аналого-цифрового перетворення з досить високими технічними характеристиками.

Перетворювачі $f \to U$, окрім виконання зворотних перетворень в описаних вище пристроях, знаходять також використання як програмовані джерела напруги, перетворювачі кутових чи лінійних переміщень у напругу та інших задачах подібного характеру.

10.4.2. Генератор з системою ФАПЧ К564ГГ1

Принцип фазової автоматичної підстройки частоти (ФАПЧ) керованих генераторів синусоїдальних або імпульсних сигналів спочатку знайшов широке використання в різних областях радіотехніки та радіолокації, а потім – у пристроях промислової автоматики, енергетичної електроніки, в вимірювальних приладах та інших напрямках електронної техніки.

Основою будь-якої системи ФАПЧ є керований напругою генератор, аналогічний перетворювачеві $U \to f$ К1108ПП1 і *фазовий детектор* – пристрій, що перетворює різницю фаз між сигналами двох рівних (кратних) частот у напругу постійного струму, середнє значення якої пропорційне різниці фаз, тобто $U_{\text{ФД}} = k\,\varphi_{\text{р}}$. В якості фазових детекторів серед розглянутих вище пристроїв можуть використовуватись *RS*- та *JK*-тригери, ЛЕ **ВИКЛ. АБО** та ін.

Функціональна схема, що пояснює роботу системи ФАПЧ, приведена на рис. 10.48.

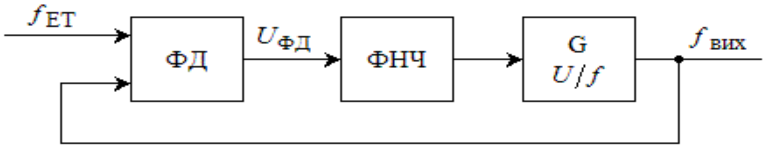


Рис. 10.48

Частоти $f_{\text{ЕТ}}$ і $f_{\text{вих}}$ однакові й відрізняються лише фазами. Якщо сигнали обох частот представляють собою меандр, а в якості фазового детектора використовується, наприклад, ЛЕ **ВИКЛ. АБО**, то, враховуючи лінійну залежність між $U_{\text{ФД}}$ і $\varphi_{\text{р}}$, при $\varphi_{\text{р}} = 90^{\circ}$ величина $U_{\text{ФД}} = 0{,}5\,E_{\text{ж}}$. Частота генератора, що відповідає цій напрузі, називається *центральною частотою* φ_0.

Особливість системи ФАПЧ полягає в тому, що вона може працювати лише в тому випадку, якщо з самого початку $f_{\text{ЕТ}} \approx f_{\text{вих}}$. При досить незначній різниці між частотами сигнал $U_{\text{ФД}}$ матиме вигляд широтно-модульованого сигналу. Фільтр низьких частот ФНЧ виділятиме модуляційну складову, що повинна забезпечувати керування генератором. Якщо таке керування має місце, то вважається, що система ФАПЧ "підхопила" зміни вхідного сигналу. Такий процес називається *"захопленням" вхідного сигналу*. Різниця між частотами, при яких має місце явище "захоплення", досить незначна, а відповідний частотний діапазон в околиці f_0 називається діапазоном захоплення частоти $2\,\Delta f_0$. Якщо система ФАПЧ працює, то зміна частоти $f_{\text{ЕТ}}$ в деяких межах буде нею відслідковуватись. Діапазон частот $f_{\text{ЕТ}}$, в якому забезпечуватиметься робота ФАПЧ, називається *діапазоном автопідстройки частоти* $2\,\Delta f_{\text{ЕТ}}$. Реально $\Delta f_{\text{ЕТ}} >> \Delta f_0$.

Система ФАПЧ може забезпечувати підстройку як близьких частот, так і кратних, тобто в n разів більших $f_{\text{ЕТ}}$. Для забезпечення стійкої роботи такої ФАПЧ у контурі зворотного зв'язку (рис. 10.48) встановлюється пристрій ділення частоти $f_{\text{вих}}$ в n разів.

Важливу роль у структурі ФАПЧ відіграє ФНЧ. Здебільшого це пасивний або активний інтегратор, призначений для зниження пульсацій вихідної напруги ФД. Збільшення постійної часу фільтра підвищує завадостійкість, стабільність роботи системи, але в той же час погіршує динамічні властивості, оскільки знижує швидкість відслідковування частоти $f_{\text{ЕТ}}$.

На рис. 10.49 приведена функціональна схема генератора К564ГГ1 (CD4046) із зовнішніми елементами, що призначені для забезпечення якісного його функціонування.

Генератор містить в собі безпосередньо керований генератор *G*, два фазових детектори ФД1 і ФД2 та два підсилювачі П1 і П2. Перший підсилювач одночасно здійснює формування прямокутних сигналів для забезпечення

якісної роботи ФД1. Підсилювач П2 працює як повторювач. Він має високий вхідний і низький вихідний опори. Завдяки цьому фільтр $R_3\,C_2$ працює в режимі холостого ходу без впливу зовнішнього навантаження $R_н$.

Частота генератора G одночасно визначається зовнішніми елементами R_1, R_2 і C_1 та зовнішньою керуючою напругою U_y, що подається на вхід **9** мікросхеми. Вхідний опір генератора має такий же порядок ($\approx 10^{12}$ Ом), що і вхідний опір підсилювача П2. Це має суттєве значення для розрахунку та підбору параметрів елементів фільтра $R_3\,C_2$. Частота генерації коливань має лінійну залежність від напруги керування U_y. При $R_2 = 0$ ця залежність визначається рівнянням: $f_{вих} = K_{1f} \cdot U_y^*$, де K_{1f} – крутизна перетворення[кГц/В], залежить від параметрів R_1 і C_1. $U_y^* = U_y / E_{ж}$.

При $R_2 \neq 0$ $f_{вих} = K_{2f} \cdot U_y^* + f_{min}$, де f_{min} – частота генерації коливань при $U_y^* = 0$.

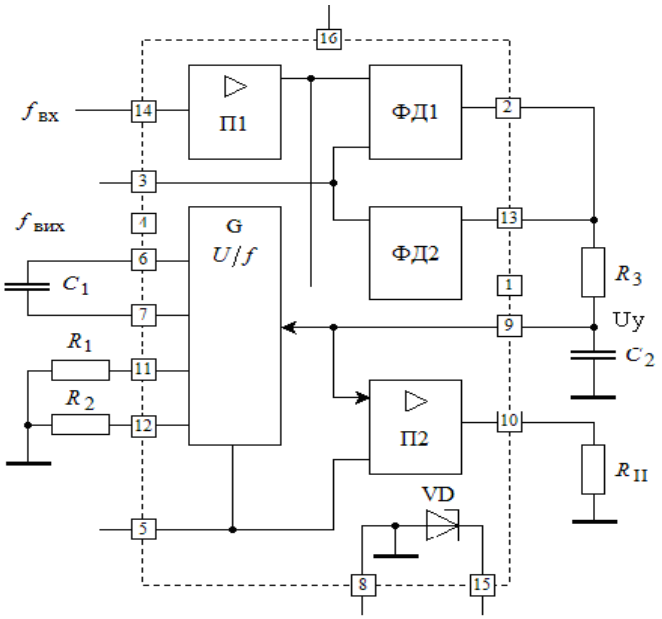


Рис. 10.49

У довідковій літературі [Зельд.] приводяться детальні діаграми, які дозволяють підібрати параметри R_1, R_2 і C_1 для вибраного діапазону частот генерації.

Генерація коливань може бути зупинена. Для цього використовується вивід **5**, високий рівень сигналу на якому зупиняє роботу генератора і підсилювача П2.

Як вже було сказано вище, мікросхема має два фазових детектори ФД1 і ФД2. Перший з них виконаний на ЛЕ **ВИКЛ. АБО**, який забезпечує максимальний діапазон автопідгонки при вхідних сигналах, що представляють собою симетричні меандри. При різниці між фазами вхідних сигналів в 90° вихідна частота $f_{вих} = f_0$. Фазовий детектор ФД2 має порівняно складну схемотехніку, і його особливістю є те, що при його використанні коефіцієнт заповнення вхідних імпульсів не впливає на діапазон автопідгонки. При співпадінні частот і фаз вхідних сигналів вихід **13** перейде у *Z*-стан, а на виході **1** з'явиться сигнал високого рівня, який повідомляє про синхронність та синфазність двох сигналів – вхідного й сигналу керованого генератора. Ще однією особливістю детектора ФД2 є те, що при його використанні діапазони частот автопідгонки та захоплення однакові й не залежать від параметрів фільтра.

На рис. 10.50 приведена схема генератора прямокутних імпульсів на мікросхемі К564ГГ1, що забезпечує регулювання частоти генерації за допомогою резистора R_1.

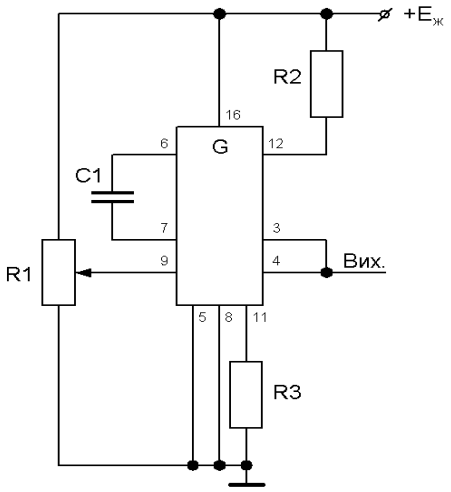


Рис. 10.50

Мікросхема забезпечує стабільну генерацію коливань у діапазоні частот від 0 Гц до 0,8 МГц (при напрузі джерела живлення 15 В) при використанні високоякісних компонентів. Діапазон вибору резисторів R_1, R_2 знаходиться у межах від 10 кОм до 1 МОм, а конденсатора C_1 – від 100 пФ до 1 мкФ. Точні значення параметрів пасивних компонентів розраховуються за допомогою спеціальних номограм, після чого уточнюються при налагодженні.

Мікросхема К564ГГ1 може використовуватись для генерації синхронної сітки частот у складних цифрових та мікропроцесорних системах з різними частотами синхросигналів, у системах точного керованого електропривода та ряді інших пристроїв цифрової електроніки та автоматики.

КОНТРОЛЬНІ ПИТАННЯ

1. Які задачі розв'язують пристрої формування імпульсів?
2. Які величини часових затримок можуть бути забезпечені безпосередньо логічними елементами?
3. Поясніть, які функції виконують інтегруючі *RC*-ланки у пристроях формування імпульсів.
4. Поясніть, в чому полягає принципова різниця в роботі пристроїв формування імпульсів на основі КМОН і ТТЛ ІС.
5. Чим обумовлені обмеження, що накладаються на діапазон вибору параметрів елементів *RC*-ланки при використанні в пристроях формування імпульсів ТТЛ ІС?
6. Які засоби використовуються для підвищення точності часових затримок в пристроях формування імпульсів?
7. У чому полягають особливості використання диференціюючих *RC*-ланок з ТТЛ ІС?
8. Дайте визначення терміну *"одновібратор"*. Поясніть призначення цього пристрою та його переваги перед пристроями формування імпульсів.
9. Які практичні рекомендації слід враховувати при виготовленні одновібраторів на основі логічних елементів?
10. Дайте загальну характеристику таймеру КР1006ВИ1.
11. Поясніть вплив напруги живлення на величину часової затримки таймера.

12. Які обмеження накладаються на величини зовнішніх часозадаючих елементів?

13. Які вимоги пред'являються до тригерів при їх використанні в схемах одновібраторів?

14. Що є основним показником якості генераторів імпульсів?

15. Приведіть визначення терміну *"мультивібратор"*.

16. Дайте пояснення регенеративному принципу збудження коливань. Поясніть характерні відмінності між роботою генераторів прямокутних імпульсів і генераторів синусоїдального сигналу.

17. Які вимоги пред'являються до тригерів при їх використанні в схемах мультивібраторів?

18. Поясніть, який принцип закладений у роботу мультивібраторів на основі тригерів Шмідта.

19. Які обмеження накладаються на величину опору резистора в мультивібраторі на основі тригера Шмідта? З чим вони пов'язані?

20. Які обмеження накладаються на ємність конденсатора в мультивібраторі?

21. Поясніть, у чому полягає особливість мультивібраторів на основі мікросхем одновібраторів АГ1, АГ3?

22. Поясніть, як можна перетворити таймер КР1006ВИ1 у тригер Шмідта?

23. Чи зміниться режим роботи мультивібратора, схема якого приведена на рис. 10.35, якщо резистор R і конденсатор C_1 поміняти місцями? Відповідь обґрунтуйте.

24. У схемах мультивібраторів, що розглянуті у § 10.3, навантаження приєднується до тих же виводів, що і часозадаючі ланки. Як величина навантаження впливатиме на частоту генерованих коливань? Відповідь обґрунтуйте.

25. У чому полягає головна особливість кварцу як елемента електричної схеми?

26. Як використовується кварц у генераторах коливань?

27. У чому полягає принципова різниця між генераторами, що будуються на основі послідовного і паралельного резонансів?

28. Які функції виконує мікросхема КР1108ПП1?

29. Поясніть, у чому полягає принцип перетворення рівня напруги в частоту імпульсів генератора?

30. Дайте пояснення ідеї зворотного перетворення частоти в напругу.

31. Приведіть приклади використання мікросхеми КР1108ПП1 як перетворювача $U \to f$ та $f \to U$.

32. Поясніть принцип роботи системи ФАПЧ.

33. Приведіть приклади використання мікросхеми К564ГГ1.

ВПРАВИ І ЗАВДАННЯ

1. Побудувати часові діаграми і пояснити роботу пристроїв формування короткочасних імпульсів, що приведені на рис. 10.51.

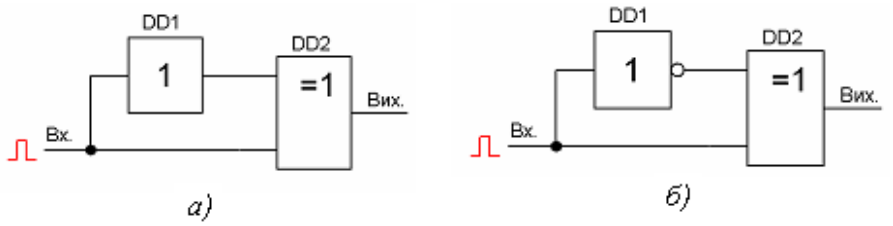


Рис. 10.51

2. Цифровий пристрій містить послідовно з'єднані 4 інвертори з внутрішньою затримкою кожного з них 10 нс. Побудувати часову діаграму вихідного імпульсу, якщо на вхід пристрою подається позитивний імпульс тривалістю 100 нс.

3. Умова попередньої задачі, але кількість послідовно з'єднаних інверторів дорівнює 5.

4. Схема, що приведена на рис. 10.3, модифікована шляхом приєднання паралельно резистору R_1 ланки з послідовно з'єднаних резистора R_2 і діода VD,

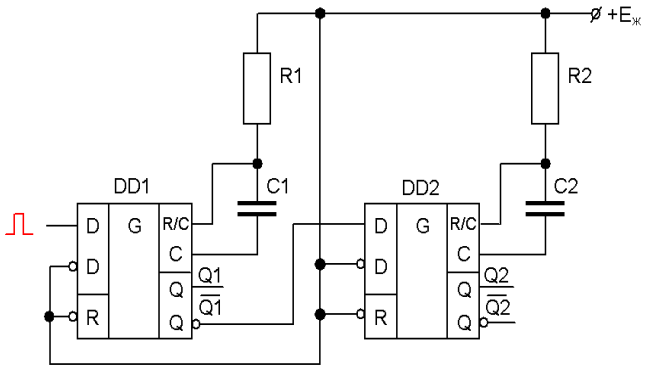


Рис. 10.53

приєднаного анодом до входу ЛЕ DD2. Пояснити зміни, що відбудуться у часових діаграмах, приведених на рис. 10.4.

5. Одновібратор виготовлений на основі ЛЕ, що мають максимальну величину затримки на перемикання 5 нс. Яка повинна бути мінімальна тривалість вхідного імпульсу? Чому?

6. Одновібратор, виготовлений на базі мікросхеми 561АГ1, має величину часової затримки τ_i = 100 мкс. Яка форма вихідних імпульсів буде при частоті запускаючих імпульсів, що дорівнює 20 кГц? Яка максимально допустима частота запускаючих імпульсів для даного одновібратора? Яка повинна бути частота вхідних імпульсів, при якій коефіцієнт заповнення $\tau_i / T = 0{,}5$?

7. Розробити схему пристрою для ділення частоти вхідних імпульсів на 4, використовуючи одновібратор 561АГ1.

8. Розробити схему одновібратора з використанням *JK*-тригера 561ТВ1.

9. Обґрунтувати можливість і розробити схему мультивібратора на основі ЛЕ **ВИКЛ. АБО**.

10. Чи можливо у схемі мультивібратора, яка приведеній на рис. 10.28, поміняти місцями зарядний резистор і конденсатор? Виконати аналіз роботи пристрою і пояснити, які зміни необхідно внести до схеми для забезпечення умов збудження коливань.

11. На рис. 10.53 приведена схема послідовного з'єднання двох одновібраторів К561АГ1. Тривалість імпульсу вхідної послідовності частотою 10 кГц дорівнює 1 мкс. Величина часової затримки кожного одновібратора $\tau_i = 10$ мкс. Побудувати часові діаграми напруг на виходах Q_1, $\overline{Q_1}$, Q_2, $\overline{Q_2}$.

12. На рис. 10.54 приведена схема мультивібратора на КМОН ІС. Пояснити принцип роботи і побудувати часові діаграми на виходах ЛЕ.

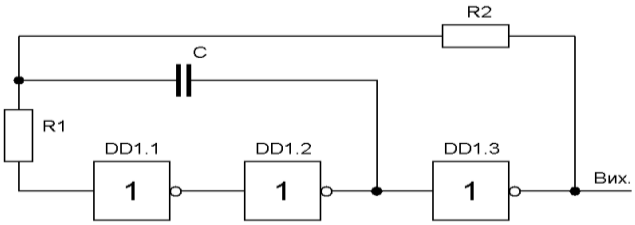


Рис. 10.54

13. На рис. 10.56 приведена практична схема генератора прямокутних імпульсів, яка може знаходити широке використання у лабораторній практиці [Зельд.], оскільки забезпечує високу стабільність коливань і широкий діапазон їх регулювання (за допомогою резистора R3 їх частоту можливо змінювати приблизно у 200 разів).

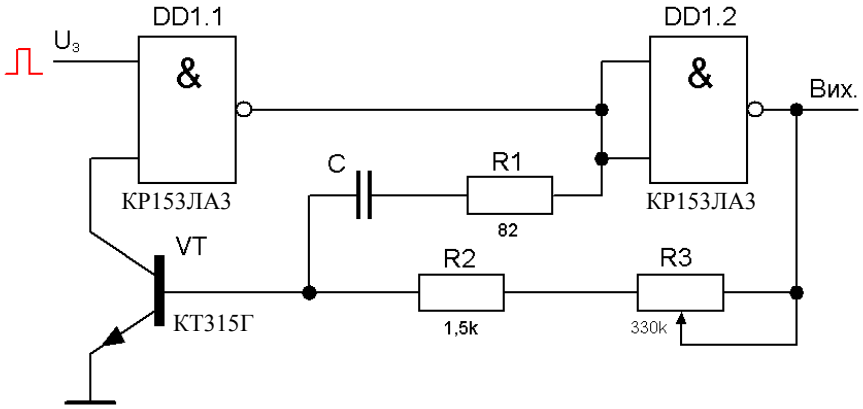


Рис. 10.56

Допоміжна зміна ємності конденсатора *C* від 20 пФ до 10 мкФ дозволяє змінювати частоту генерації від одиниць мегаГерц до долей Герц. Проаналізувати роботу генератора. Обґрунтувати умови вибору резисторів. Пояснити, в якому режимі працює транзистор VT, якщо $U_з$ – запускаюча напруга. Встановити залежність між параметрами часозадаючих елементів і частотою генерованих коливань.

14. На основі мультивібратора, схема якого приведена на рис. 10.32, *а*, і двонаправлених ключів К561КТ3 розробити схему генератора з чотирьохрозрядним цифровим способом задання частоти генерації.

15. На основі мультивібратора, побудованого з використанням тригера Шмідта, розробити схему генератора зі шпаруватістю вихідних імпульсів $S = 4$.

16. Використовуючи умови та результати розв'язання двох попередніх задач, розробити схему генератора з чотирьохрозрядним цифровим способом задання шпаруватості вихідних імпульсів.

17. Схему, отриману у попередній задачі, доробити так, щоб керуючий двійковий код задавати у послідовному форматі.

18. Використовуючи мікросхеми одновібраторів АГ1, АГ3, розробити схему генератора з незалежним регулюванням частоти та шпаруватості вихідних імпульсів. Привести обґрунтування роботи генератора і забезпечення ним заданих параметрів.

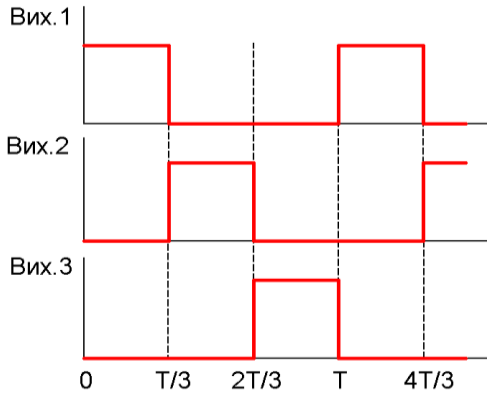


Рис. 10.57

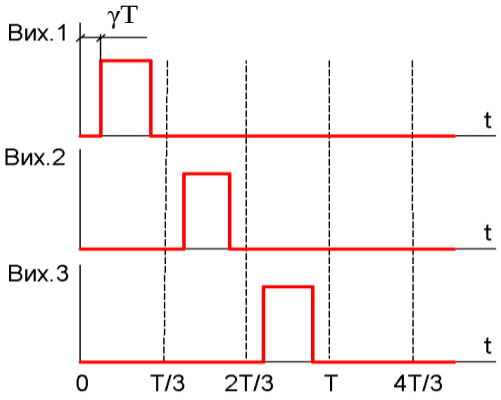


Рис. 10.58

19. Використовуючи мікросхеми одновібраторів АГ1, АГ3, розробити схему трифазного генератора, який повинен забезпечувати часові співвідношення між вихідними сигналами у відповідності до рис. 10.57.

20. Обґрунтувати можливість або неможливість побудови трифазного генератора, який повинен генерувати послідовності імпульсів у відповідності до часових діаграм, що приведені на рис. 10.58 ($0 \le \gamma \le 0{,}5$).

21. Побудувати часові діаграми на виході одновібратора, на вхід якого подаються короткі імпульси, період слідування котрих у два рази менший від постійної часу затримки.

22. Розробити схему мультивібратора на основі таймера КР1006ВИ1, в якому може забезпечуватись широтно-імпульсна модуляція вихідних імпульсів.

Розв'язання. Зрозуміло, що схема генератора повинна відповідати рис. 10.36. Широтна модуляція коливань може бути реалізована шляхом зміни рівня зразкової напруги, яка діє на виводі **5** мікросхеми. Тому джерело модулюючого сигналу необхідно ввімкнути між загальною шиною і нижньою обкладинкою конденсатора C_2. При зміні зразкової напруги, яка дорівнює $U_з = 2/(3E_ж)$, для забезпечення спрацювання компаратора DA1 напруга на другому його виході – виводі **6** – повинна змінюватись відповідно. Оскільки напруга на виводі **6** визначається часом зарядки і розрядки часозадаючого конденсатора C_1, тривалість інтервалів часу $t_з$ і $t_р$ змінюватиметься пропорційно модулюючій напрузі.

23. На рис. 10.59 приведена схема генератора з використанням інверторів ТТЛ. Виконати аналіз роботи пристрою. Обґрунтувати виконання умов збудження коливань, режим роботи кварцу, призначення використаних елементів.

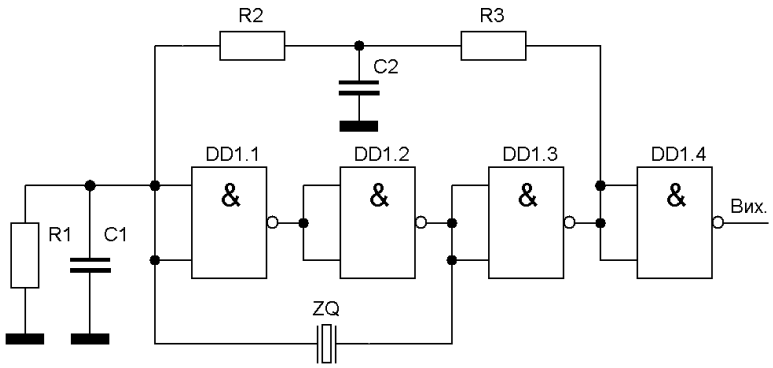


Рис. 10.59

24. На рис. 10.60 приведена схема кварцового генератора з допоміжними інтегруючими *RC*-ланками. Пояснити роботу пристрою. Обґрунтувати виконання умов збудження коливань, режим роботи кварцу. Пояснити, чи можлива робота генератора без кварцу. Якщо так, то яку функцію у схемі виконує кварц?

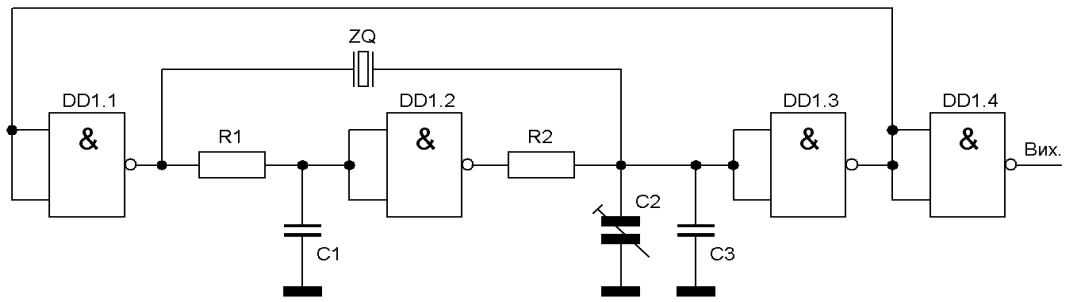


Рис. 10.60

Пояснення. Інтегруючі ланки $R_1 C_1$ та $R_2 (C_2 + C_3)$ забезпечують зсув фази на 180º. Такий же фазовий зсув дає інвертор на DD1.2. Таким шляхом від виходу DD1.1 до входу DD1.3 фазовий зсув може досягати 360º і визначається постійними часу *RC*-ланок. Два інвертори-підсилювачі DD1.1 і DD1.3, які входять у коло, охоплене зворотним зв'язком, забезпечують також фазовий зсув 360º. Тобто умова збудження коливань забезпечується і без використання кварцу. Але, якщо генерація коливань почнеться на частотах, близьких до частоти послідовного резонансу кварцу $(0{,}9 \ldots 0{,}95)\, f_{\text{п}}$, то, внаслідок високої добротності останнього, коливання "підтягнуться" до частоти коливань кварцу.

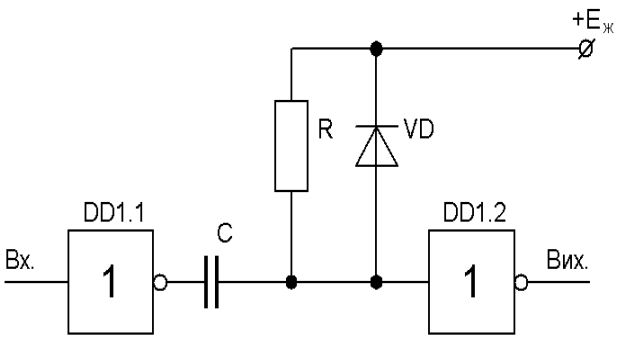


Рис. 10.61

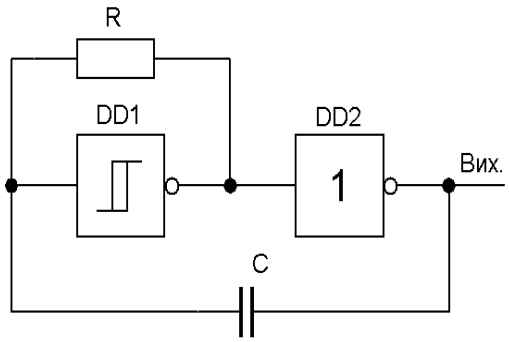


Рис. 10.62

25. На рис. 10.61 приведена схема пристрою, призначеного для скорочення тривалості вхідного імпульсу. Пояснити принцип роботи. Побудувати часові діаграми роботи пристрою. Пояснити призначення діоду VD. У яких випадках потрібне його використання? Порівняти температурну стабільність при використанні ТТЛ і КМОН ІС.

26. На рис. 10.62 приведена схема генератора прямокутних імпульсів з використанням тригера Шмідта. Пояснити принцип роботи. Порівняти особливості її роботи при використанні ТТЛ і КМОН ІС.

27. Проаналізувати роботу пристрою, схема якого приведена на рис. 10.9, при використанні ТТЛ і КМОН ІС. Пояснити, в якому випадку температурна стабільність пристрою буде вищою. Чому?

28. Привести порівняльний температурний аналіз пристрою, схема якого приведена на рис. 10.12, *а*, при використанні ТТЛ і КМОН ІС.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АЛМ ─ арифметично-логічний пристрій

АМ – амплітудна модуляція

АІМ – амплітудно-імпульсна модуляція

БіКМОН – поєднання технологій КМОН з біполярними транзисторними ключами на виході

БМК – базовий матричний кристал

БТ – багатоемітерний транзистор

ВІС – велика інтегральна схема

ЕЗЛ – емітерно-зв’язана логіка

ЕП – елемент пам’яті

ЗП – запам’ятовуючий пристрій

ДТЛ – діодно-транзисторна логіка

ДНФ – диз’юнктивна нормальна форма запису логічних функцій

ДДНФ ─ досконала диз’юнктивна нормальна форма запису логічних функцій

ДКНФ ─ досконала кон’юнктивна нормальна форма запису логічних функцій

ІМС – інтегральна мікросхема

ІС – інтегральна схема

ІКМ – імпульсно-кодова модуляція

ІІЛ – інтегральна інжекційна логіка

КНФ – кон’юнктивна нормальна форма запису логічних функцій

КМОН (CMOS) – комплементарні МОН-структури

ЛЕ – логічний елемент

ЛІЗМОН – МОН структури з лавинною інжекцією заряду

МОН – структура на базі з’єднання метал-окисел-напівпровідник

МС – мікросхема

НВІС – надвелика інтегральна схема

ПЗЗ – прилад з зарядовим зв’язком

ПЗП – постійний запам’ятовуючий пристрій

ПКЧ – перетворювач код-частота

ПЛІС – програмована логічна інтегральна схема

ПЛМ – програмована логічна матриця

ПМД – послідовність максимальної довжини

ПМЛ – програмована матрична логіка

ПТП – початкова таблиця переходів скінченного автомата

ОЗП – оперативний запам’ятовуючий пристроій

РПЗП – репрограмований постійний запам’ятовуючий пристрій

РТЛ – резистивно-транзисторна логіка

ТТЛ (TTL) – транзисторно-транзисторна логіка

ТТЛШ – транзисторно-транзисторна логіка з діодами Шоткі

ФАПЧ – фазова автопідстройка частоти

ФД – фазовий детектор

ФМ – фазова модуляція

ФІМ – фазо-імпульсна модуляція

ФНЧ – фільтр нижніх частот

ЦА – цифровий автомат

ЦІС – цифрова інтегральна схема

ЧМ – частотна модуляція

ЧІМ – частотно-імпульсна модуляція

ШІМ – широтно-імпульсна модуляція

CPLD – Complex Programmable Logic Device

EPROM (РПЗП) – Electrically Programmable ROM

PROM (РПЗП) – Programmable Read Only Memory

FPGA – Field Programmable Gate Array

Flex – Flexible Logic Element Matrix

GA (БМК) – Gate Array

ISP – In-System Programming

JFET – Junction Field Effect Tranzistor

JTAG – Joint Test Action Group

MESFET – Metal Semicondactor Field Effect Tranzistor

PAL (ПМЛ) – Programmable Array Logic

PLA (ПЛМ) – Programmable Logic Array

SOC – System on Chip

SPI – Serial Peripherial Interface

ЛІТЕРАТУРА

1. Алексенко А. Г., Шагурин И. И. Микросхемотехника. – М.: Радио и связь, 1990.- 496 с.
2. Барнс Д. Электронное конструирование: методы борьбы с помехами. – М: Мир, 1990.- 238 с.
3. Баскаков С. И. Радиотехнические цепи и сигналы. – М.: Высшая школа, 1983.- 536 с.
4. Бойко В. І., Багрій В. В. Цифрова схемотехніка. – К: ІЗМН, 2001.- 228 с.
5. Большие интегральные схемы запоминающих устройств. Справочник. Под ред. Гордонова А. Ю., Дьякова Ю. Н. – М.: Радио и связь, 1990.-288 с.
6. Борисенко О. А. Цифрові автомати. – Суми: Видавництво СумДУ, 2001.- 168 с.
7. Вениаминов В. Н., Лебедев О. Н., Мирошниченко А. И. Микросхемы и их применение. – М.: Радио и связь, 1989.- 240 с.
8. Власов А. И., Сулимов Ю. И. Электронные промышленные устройства. – М.: Высшая школа, 1988.- 304 с.
9. Голдсуорт Б. Проектирование цифровых логических устройств. – М.: Машиностроение, 1985.- 287 с.
10. Гольденберг Л.М. Импульсные устройства. –М.: Радио и связь, 1981.-224 с.
11. Гулый В. Д., Артеменко М. Б. Методические указания по изучению дисциплины Электронные промышленные устройства. – К.: КПИ, 1986.- 32 с.
12. Гурвич И. С. Защита ЭВМ от внешних помех. – М.: Энергоатомиздат, 1984.- 224 с.
13. Гутников В. С. Интегральная электроника в измерительных устройствах. – Л.: Энергоатомиздат, 1988.- 304 с.
14. Завадский В. А. Компьютерная электроника. – К.: ТОО ВЕК, 1996.- 360 с.

15. Зельдин Е.А. Импульсные устройства на микросхемах. – М.: Радио и связь, 1991.- 160 с.
16. Зельдин Е. А. Цифровые интегральные микросхемы в информационно-измерительной аппаратуре. – Л.: Энергоатомиздат, 1986.- 280 с.
17. Калабеков Б. А. Микропроцессоры и их применение в системах передачи и обработки сигналов. – М.: Радио и связь, 1988.- 368 с.
18. Лебедев О. Н. Микросхемы памяти и их применение. – М.: Радио и связь, 1990.- 160 с.
19. Логические ИС. Справочник в 2-х томах. Бином, 1993.
20. Мейдза Ф. Интегральные схемы. Технология и применение. – М.: Мир, 1981.- 280 с.
21. Новиков Ю. В. Основы цифровой схемотехники. – М.: Мир, 2001.-380 с.
22. Новожилов О.П. Основы цифровой техники / учебное пособие. –М.: ИП РадиоСофт, 2004.- 528 с.
23. Опадчий Ю. Ф., Глудкин О. П., Гуров А. И. Аналоговая и цифовая электроника. – М.: Горячая линия – Телеком, 1999.- 768 с.
24. Петросян О. А., Козырь И. Я., Колядов Л. А., Щетинин Ю. И. Схемотехника БИС постоянных запоминающих устройств. – М.: Радио и связь, 1987.- 304 с.
25. Полупроводниковые БИС запоминающих устройств. Под ред. Гордонова А. Ю., Дьякова Ю. Н. – М.: Радио и связь, 1987.- 360 с.
26. Потемкин И. С. Функциональные узлы цифровой автоматики. – М.: Энергоатомиздат, 1988.- 320 с.
27. Применение интегральных микросхем в электронной вычислительной технике. Справочник. Под. ред. Б.Н. Файзулаева, Б.В. Тарабрина. –М.: Радио и связь; 1987.- 384 с.
28. Пухальский Г. И., Новосельцева Т. Я. Проектирование дискретных устройств на интегральных микросхемах. – М.: Радио и связь, 1990.- 304 с.

29. Пухальский Г. И., Новосельцева Т. Я. Цифровые устройства. – СПб.: Политехника, 1996.- 880 с.
30. Скаржепа В. А., Луценко А. Н. Электроника и микросхемотехника. Кн.1. – К.: Вища школа, 1989.- 430 с.
31. Смирнов В. С. Електронні імпульсні пристрої. – Київ: НТУ КПІ, 1998.- 140 с.
32. Справочник по цифровой схемотехнике / В.И. Зубчук, В.П. Сигорский, А.Н. Шкуро. – К.: Техника, 1990. – 448 с.
33. Точчи Р., Уидмер Н. Цифоровые системы. Теория и практика, 8-е издание.: Пер. с англ. – М.:издательский дом «Вильямс», 2004. - 1024 с.
34. Трачик В. Дискретные устройства автоматики. – М.: Энергия, 1978.-456 с.
35. Угрюмов Е. Цифровая схемотехника. – СПб.: БХВ, 2000.- 528 с.
36. Цифровые и аналоговые интегральные микросхемы. Справочник. Под ред. Якубовского. – М.: Радио и связь, 1989. - 496 с.
37. Цифрова схемотехніка: Підручник у двох томах, том 2. Жуйков В.Я., Бойко В.І., Зорі А.А. та ін. –К.: Аверс, 2002.- 408 с.
38. Цифровые интегральные микросхемы. Справочник. – Минск: Беларусь, 1991.- 494 с.
39. Шило В. Л. Популярные цифровые микросхемы. – М.: Радио и связь, 1987.- 352 с.
40. Энциклопедия кибернетики: в 2-х т. – К.: УСЭ, 1974, т.1.