МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ ім. О.С. ПОПОВА

Кафедра комп'ютерно-інтегрованих технологічних процесів і виробництв

Антонов О.С.

ЦИФРОВІ ПРИСТРОЇ

Конспект лекцій

дисципліни «Цифрові пристрої»
для студентів 3 курсу ННІ Радіо, телебачення, електроніки

Одеса 2015

УДК 682.335.5(075) План НМВ 2014 р.

Рецензент – Загребнюк В.І., к.т.н., доцент ОНАЗ ім. О.С. Попова

Антонов О.С. Цифрові пристрої: конспект лекцій [з дисципліни «Цифрові пристрої» для студентів 3 курсу, ННІ Радіо, телебачення, електроніки] / Антонов О.С. – Одеса: ОНАЗ ім. О.С. Попова, 2015. – 140 с.

В конспекті лекцій «Цифрові пристрої», розглядаються принципи побудови та функціонування цифрових та мікропроцесорних систем. На прикладах мікропроцесорів та мікроконтролерів фірм Motorola показані принципи проектування мікропроцесорних систем та їх програмування, у тому числі призначених для цифрового оброблення сигналів.

СХВАЛЕНО
на засіданні кафедри КІТП і В
та рекомендовано до друку.
Протокол № 3 від 06. 10. 2014 р.

ЗАТВЕРДЖЕНО
методичною радою
академії зв’язку.
Протокол № 10/14 від 04.07.2014 р.

ЗМІСТ

ВСТУП

Подальший розвиток і удосконалення систем зв’язку та систем автоматичного керування значною мірою обумовлені широким впровадженням засобів цифрової техніки. Впровадження цифрової техніки обумовлено відомими перевагами цифрових пристроїв порівняно з аналоговими. Вони є більш надійними, мають високу стабільність параметрів, а також забезпечують високу точність оброблення сигналів.

Сьогодні вже неможливо собі уявити телекомунікаційне обладнання без сучасних мікропроцесорів і мікроконтролерів. Широкий спектр функцій, які реалізують системи комутації, шлюзи, маршрутизатори, інтегровані платформи, сервери, робочі станції, системи автоматичного керування, вимагає від процесорів і мікроконтролерів високої продуктивності та багатофункційності. Десятки років можна було спостерігати процес взаємного стимулювання розвитку процесорів, з одного боку, та побудованого на їх основі телекомунікаційного обладнання, з іншого.

Конспект лекцій призначено для самостійної роботи студентів напряму 6.050901 – Радіотехніка, радіоелектронні апарати та зв'язок з дисципліни “Цифрові пристрої”. Дисципліна має 288 годин і складається з двох змістових модулів:

1. Цифрові пристрої.
2. Введення в мікропроцесорну техніку.

При вивченням дисципліни студенти отримують такі знання та уміння: подавати та трактувати вхідні та вихідні числові дані для подальшого цифрового оброблення. Співвідносити логічні змінні та функції з цифровими сигналами, що їх реалізують. Синтезувати цифрові пристрої, використовуючи типові цифрові блоки, вузли та елементи. Ставити та розв’язувати задачі, пов’язані з вибором засобів обчислювальної техніки, мікропроцесорів і мікроконтролерів за їх технічними, експлуатаційними та економічними характеристиками. Створювати та налагоджувати програмне забезпечення для мікропроцесорів.

І З дисципліни “Основи схемотехніки”:

1 Інтегральна схемотехніка, технологія МОП та КМОП, великі (ВІС) та надвеликі (НВІС) інтегральні схеми, конструктивна реалізація.

ІІ З дисципліни “Інформатика”:

1 Призначення обчислювальних систем, задачі, які можуть розв’язувати обчислювальні системи.

2 Алгоритмізація задач, які розв’язуються обчислювальними системами, складання структурних схем алгоритмів розв’язуваних задач.

3 Мови високого рівня Delphi, C^{++}.

4 Мати навики складання та налагодження програм, які мають розгалуження та цикли, написаних мовами високого рівня.

ІІІ З дисципліни “Дискретна математика”:

1 Позиційні системи числення – двійкова, десяткова, шістнадцяткова, двійкова-десяткова – перехід від однієї системи числення до іншої.

2 Алгебра логіки – поняття логічної змінної та логічної функції, закони алгебри логіки, мінімізація логічних функцій методом Квайна та методами координатних діаграм.

І ЦИФРОВІ ПРИСТРОЇ

Сьогодні оброблення значної кількості інформації виконується у цифровій формі. Цифрова форма подання інформації передбачає використання цифрових сигналів.

Цифровий сигнал – це дискретний електричний сигнал, який у визначений спосіб оброблений і поданий у вигляді цифрового коду. Звичайно цифровий сигнал є пов'язаним з відповідним йому аналоговим сигналом, хоча це не є обов'язковим. У цифрових системах оброблення інформації найбільш поширеною є двійкова форма представлення цифрового сигналу. При такій формі сигнал подано двома значеннями 0 і 1, які задаються певними значеннями рівнів електричного сигналу (напруги або струму). Для визначеності будемо вважати, якщо це окремо не обумовлено, що 1 кодується додатним значенням електричного сигналу, 0 – відсутністю сигналу. Часові діаграми такого сигналу показано на рис. 1.1.

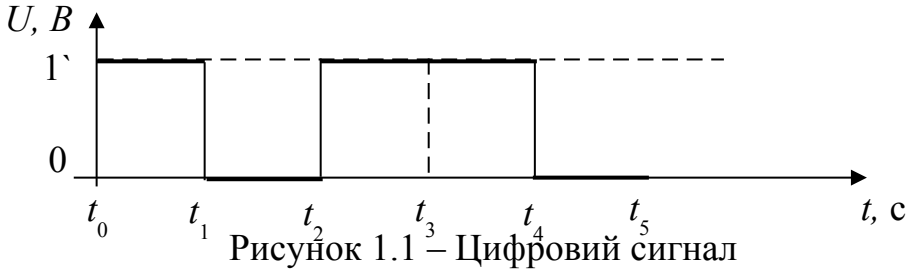


Рисунок 1.1 – Цифровий сигнал

Для оброблення цифрових сигналів використовуються цифрові пристрої.

Цифровий пристрій – це пристрій, який призначено для приймання, оброблення і видання цифрової інформації. Цифрові пристрої складаються з елементів.

Елементи цифрових пристроїв – це найменші функційні частини, на які можна розподілити цифровий пристрій при його логічному проектуванні й технічній реалізації. Прийнято вважати, що елементами цифрових пристроїв є елементи, які виконують певні логічні функції, тобто логічні елементи.

1.1 Двійкова арифметика. Форми подання чисел з фіксованою та плаваючою точками. Подання алфавітно-цифрової інформації

Двійкові числа для оброблення розміщуються у комірках пам'яті та регістрах мікропроцесора таким чином, що для кожного біта (двійкового розряду) призначено окремий елемент пам'яті. Сукупність елементів, які утворюють комірку пам'яті, визначають довжину двійкового числа, називаються *розрядною сіткою*. Довжина розрядної сітки завжди обмежена і визначається конструктивними особливостями пристрою. Тому значення чисел, що оброблюються, завжди обмежені. Ще більше обмежень постає при поданні дробових чисел.

Для подання дробових чисел у цифровій техніці використовуються дві форми:

- форма з фіксованою точкою;
- форма з плаваючою точкою.

При використанні форми з фіксованою точкою розрядна сітка конструктивно розподіляється на три частини: для подання знака, цілої і дробової частини числа. Використання такої форми пов'язано з труднощами при підготовці даних для оброблення, для забезпечення необхідного діапазону поданих даних. Також при обробленні даних важко забезпечити необхідну точність оброблення (розрахунків).

Значно ефективніше використання форми з плаваючою точкою. Таке подання даних дозволяє розширити діапазон чисел і забезпечити необхідну точність подання результатів.

У цій формі дробове число подається у вигляді мантиси M і порядку E *(Exponent)*

$$D = \pm M \cdot B^{\pm E},$$

де B – основа системи числення, у нашому випадку $B = 2$ – двійкова система числення.

Для однакового подання різних чисел мантису завжди нормалізують у межах

$$0{,}1 \le M < 1,$$

що виключає наявність нульового розряду системи числення після коми. Якщо у результаті обчислення, результат перестає бути нормалізованим, то перед його збереженням у пам'яті його необхідно нормалізувати, виконавши необхідний зсув мантиси і змінити значення порядку.

У формі з плаваючою точкою кількість розрядів порядку визначає діапазон оброблюваних чисел, а кількість розрядів мантиси – точність їх подання.

Для подання алфавітно-цифрової інформації, що включає цифри, літери різних алфавітів, розділові знаки, математичні та інші символи, використовуються різні коди для оброблення такої інформації. Так у персональних комп'ютерах використовується код системи ASCII.

Система ASCII (American Standard Code for Information Interchange – Американській „стандартний код для обміну інформацією") є угодою за стандартом кодування символів, яка схвалена Асоціацією стандартів США. Наявність такого коду спрощує обмін інформацією між різними пристроями комп'ютера. Фірма IBM запропонувала для використання у персональних комп'ютерах розширену версію системи ASCII, яка відрізняється від стандартної тим, що всі вісім бітів символу використовуються для кодування інформації. Це дало змогу збільшити кількість символів на 128 і включити до їх складу символи національних алфавітів.

Стандартні коди ASCII можна розподілити на три групи:

- коди керування передаванням даних;
- коди керування форматом;
- коди друкованих символів.

Коди керування передаванням даних подають інформацію, яка призначена для керування процесами обміну даними.

Коди керування форматом зображення використовуються для керування принтером або екраном монітора.

1.2 Кодування двійкових чисел і виконання математичних операцій над ними

Існують два види кодів подання інформації – послідовний і паралельний, які обумовлюють порядок надходження даних для оброблення. Це пов'язано з конструктивними особливостями ліній передавання інформації, які бувають *двопровідними* (телефонна лінія, радіорелейна лінія тощо) і *багатопровідними* – шинами. У двопровідній лінії інформація передається побітно (у кожний момент часу на лінії є лише один біт), і код, що використовується, називається *послідовним*. Оброблення послідовного коду відбувається за декілька тактів (визначається розрядною сіткою), в міру надходження окремих бітів. У багатопроводовій шині у кожний момент часу інформація подана кількома розрядами числа, які можуть оброблятися одночасно, за один такт. Такий код називається *паралельним*.

Цілі беззнакові двійкові числа можуть бути подані натуральним кодом числа.

Натуральний код числа передбачає подання даних, як цілих беззнакових двійкових чисел. Діапазон подання чисел у натуральному коді визначається довжиною розрядної сітки. При цьому максимальне число, що може бути подане у натуральному коді розрядної сітки становить 2^n–1, де n – кількість розрядів.

Необхідність виконання алгебраїчного додавання передбачає зображення числа разом зі своїм знаком, тому для подання таких чисел використовуються прямий, обернений і додатковий коди. Спосіб побудови цих кодів передбачає забезпечення таких вимог:

- запис алгебраїчного знака числа;
- подання від'ємних чисел за допомогою допоміжних додатних чисел, які відрізняються від відображення вихідних додатних чисел таким чином, щоб їх області зображення не збігалися;
- повна ідентичність алгоритмів виконання операцій над числами з однаковими і різними знаками, для забезпечення однотипності апаратного забезпечення для виконання цих операцій.

Прямий код (ПК) передбачає подання від'ємного числа таким чином, щоб у старшому розряді числа було показано його знак у вигляді 1 або 0, а в інших розрядах – його модуль. Старший розряд при цьому називають *знаковим*. Зображення додатного числа співпадає з його зображенням у натуральному коді, тому що знак + кодується у вигляді 0. Під час запису зручно знаковий розряд відокремлювати точкою після нього. Наприклад, $-98_D = 1.1100010_{ПК}$.

Діапазон подання від'ємних чисел у прямому коді сягає від 0 до $2^{n-1} - 1$, а для додатних чисел – 2^{n-1}.

До недоліків використання прямого коду можна віднести:

− не забезпечується ідентичність алгоритмів виконання операцій над числами з однаковими і різними знаками, що призводить до збільшення кількості операцій для отримання результату;

− нуль може мати два значення: додатне число (у вигляді байта) – 0.0000000 і від'ємне число – 1.000000, що також потребує виконання додаткових операцій при обчисленнях.

Для усунення цих недоліків і виконання операції віднімання (алгебраїчного додавання) в обчислювальній техніці використовуються *обернений* і *доповнювальний* коди. Додатні числа у цих кодах подаються аналогічно прямому, а від'ємні обчислюються за певними алгоритмами.

Обернений код (ОК) від'ємного двійкового числа A, для певної розрядної сітки, обчислюється за виразом

$$A_{\text{об}} = N - |A|,$$

де N – значення найбільшого беззнакового числа, що можливо розташувати у певній розрядній сітці – значення N для десяткових дробів становить

$$N = 2 - 2^{-(n-1)},$$

а для цілих чисел

$$N = 2^n - 1.$$

У цих виразах n – кількість бітів розрядної сітки для подання числа. Діапазон подання чисел в оберненому коді, який можна подати у розрядній сітці, такий самий, як і у прямому коді.

За визначенням обернений код від'ємного числа є доповненням його модуля до найбільшого беззнакового числа, яке можна розмістити у розрядній сітці. Таким чином, взаємне перетворення прямого й оберненого кодів від'ємного числа виконується як операція порозрядної інверсії всіх розрядів числа крім знакового, в якому необхідно записати 1.

Наприклад,

$$-98_D = 1.1100010_{\text{ПК}} \Rightarrow \quad 1.0011101_{\text{ОК}}.$$

До переваг оберненого коду можливо віднести простий взаємозв'язок прямого й оберненого кодів, у результаті чого взаємне перетворення цих кодів є порозрядною операцією, що спрощує і прискорює її виконання.

Недоліками цього коду є необхідність урахування перенесення зі старшого розряду, яке виникає при виконанні додавання, і наявність двох значень для нуля: додатне – 0.0000000 та від'ємне – 1.1111111.

Доповнювальний код (ДК) від'ємного двійкового числа A для певної розрядної сітки обчислюється за виразом

$$A_{\text{об}} = K - |A|,$$

де K – значення ваги розряду, який знаходиться за старшим розрядом використовуваної розрядної сітки – значення K для десяткових дробів становить

$$K = 2,$$

а для цілих чисел

$$K = 2^n.$$

Діапазон подання чисел у доповнювальному коді для від'ємних чисел становить $0\ldots2^{n-1}$, де n – кількість розрядів подання числа.

Доповнювальний код від'ємного числа легко отримати, додаючи одиницю молодшого розряду до оберненного коду цього числа.

Для попереднього прикладу

$$-98_D = 1.1100010_{\text{ПК}} \Rightarrow \quad 1.0011101_{\text{ОК}}$$

$$\begin{array}{rr} & 1.0011101 \\ + & \underline{\qquad\qquad 1} \\ & 1.0011110 \end{array}$$

$$1.0011101_{\text{ОК}} \Rightarrow \ 1.0011110_{\text{ДК}}$$

Для визначення прямого коду числа, яке подано у доповнювальному коді, віднімається одиниця із молодшого розряду, в результаті чого отримаємо обернений код, а далі виконуємо інверсію всіх розрядів числа.

Використання доповнювального коду ліквідує недоліки оберненного коду: перенесення із старшого розряду втрачається і не враховується у подальших обчисленнях; нуль у доповнювальному коді – тільки додатне число.

Виконання операції додавання у кодах дещо відрізняється від звичайного:

– розряди знаковий і розряди числа є рівноправними і перенесення у знаковий розряд необхідно враховувати та додавати до знакових розрядів;

– при виконанні операції додавання в ОК перенесення із знакового розряду необхідно додавати до молодшого розряду числа;

– при виконанні операції додавання в ДК перенесення із знакового розряду ігнорується і відкидається.

Розглянемо операцію додавання цілих двійкових чисел зі знаком у кодах. Для подання у кодах необхідно користуватися 8-розрядною сіткою (з урахуванням знака) у вигляді байта.

1 Для чисел з різними знаками

$$27_D - 30_D = 27_D + (-30_D) = 11011_B - \mathit{11110}_B$$

представимо двійкові числа у кодах

ОК	ДК
$+\,11011_B \Rightarrow$ 0.0011011	$+\,11011_{\text{В}} \Rightarrow$ 0.0011011
$-\,11110_B \Rightarrow$ 1.1100001	$-\,11110_B \Rightarrow$ 1.1100010

Виконаємо операцію додавання у кодах

ОК	ДК
0.0011011	0.0011011
$+$ 1.1100001	$+$ 1.1100010
1.1111100	1.1111101

Результати отримано у відповідних кодах. Перетворимо їх на ПК і подамо у десятковій системі числення

$$1.1111100_{\text{ОК}} \Rightarrow \ 1.0000011_{\text{ПК}} \Rightarrow \ -3_D.$$

Результат при роботі з оберненим кодом правильний.

Для перевірки результату, поданого у ДК, його треба спочатку перевести в ОК, для чого від молодшого розряду необхідно відняти одиницю.

$$1.1111101_{ДК} - 1 = 1.1111100_{ОК} \Rightarrow \ 1.0000011_{ПК} \Rightarrow \ -3_D.$$

Результат також правильний.

Мікропроцесори виконують операцію додавання у доповнювальному коді, тому на етапі підготовки даних їх слід подати у вигляді ДК. Для двох інших чисел з різними знаками

$$-27_D + 30_D = -1011_B + 11110_B$$

подамо двійкові числа у кодах

ОК	ДК
$-11011_B \Rightarrow$ 1.1100100	$-11011_B \Rightarrow$ 1.1100101
$+11110_B \Rightarrow$ 0.0011110	$+11110_B \Rightarrow$ 0.0011110

Виконаємо операцію додавання у кодах

ОК	ДК
1.1100100	1.1100101
+ 0.0011110	+ 0.0011110
1 0.0000010	1 0.0000011
+ 1	
0.0000011	

При виконанні операції в ОК перенесення із знакового розряду було враховано в результаті, а при виконанні операції в ДК – відкинуто. Результат в обох випадках отримано у вигляді додатного числа $11_B = 3_D$, що є правильним.

1.3 Логічні основи ЦТ. Аксіоми Бульової алгебри, логічні елементи. Логічне проектування цифрових схем

Крім обчислювальних функцій у цифровій техніці можуть виконуватися операції, які пов'язані з визначенням і керуванням станами певних об'єктів. Для виконання таких дій використовується, так звана, алгебра логіки або Булева алгебра.

Для формального опису алгебра логіки використовує спеціальні змінні й функції, які мають назву логічних (бульових). Логічна змінна − це змінна, яка може мати лише два значення *L0* (логічний нуль) і *L1* (логічна одиниця). Логічні змінні можуть поєднуватися за допомогою логічних функцій. Функція:

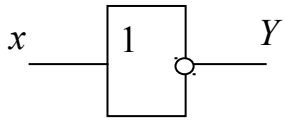
$$Y = f(x_0, x_1, x_2 \ldots x_n) \qquad \text{при } y, x_i \in 1,0$$

є *n*-розрядною логічною функцією, область значень якої визначена двійковою системою числення зі значеннями 0 і 1. Логічні функції найчастіше визначаються у вигляді аналітичної формули, яка є записом логічної функції, або таблиці істинності, яка є найбільш наочним виглядом відповідної логічної функції.

Операція НІ (NOT) − інверсія. Логічна функція *Y* є інверсією якщо її значення є протилежним значенню змінної *x*. Аналітична форма запису цієї функції має вигляд:

$$Y = \overline{x}.$$

Таблиця істинності й умовне позначення цього елемента показано на рис. 1.2.



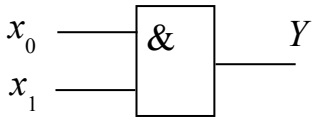
x	Y
0	1
1	0

Рисунок 1.2 − Умовне позначення логічного елемента НІ та його таблиця істинності

Операція ТА (AND) − логічне множення (кон'юнкція). Логічна функція Y дорівнює логічній одиниці, якщо всі змінні x дорівнюють логічній одиниці. Аналітична форма запису цієї функції має вигляд:

$$Y = x_1 \wedge x_0 .$$

Умовне позначення елемента та його таблиця істинності наведена на рис. 1.3.



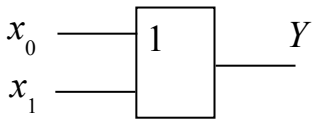
x_1	x_0	Y
0	0	0
0	1	0
1	0	0
1	1	1

Рисунок 1.3 − Умовне позначення логічного елемента ТА та його таблиця істинності

Операція АБО (OR) − логічне додавання (диз'юнкція). Логічна функція Y дорівнює логічній одиниці, якщо хоча б одна зі змінних x дорівнює логічній одиниці. Аналітична форма запису цієї функції має вигляд:

$$Y = x_1 \vee x_0 .$$

Умовне позначення елементу й його таблиця істинності наведена на рис. 1.4.



x_1	x_0	Y
0	0	0
0	1	1
1	0	1
1	1	1

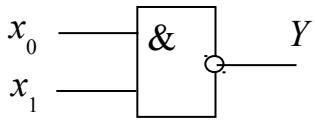
Рисунок 1.4 − Умовне позначення логічного елемента АБО

У цифровій техніці також використовуються логічні функції, які є простими комбінаціями основних. До таких функцій відносяться наступні:

Операція ТА-НІ − логічне множення з інверсією. Ця логічна функція виконується як інверсія результату логічного множення, у спеціальній літературі також має назву «штрих Шеффера». Аналітична форма запису цієї функції має вигляд:

$$Y = \overline{x_1 \wedge x_0}.$$

Умовне позначення елемента та його таблиця істинності наведена на рис. 1.5.



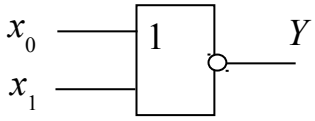
x_1	x_0	Y
0	0	1
0	1	1
1	0	1
1	1	0

Рисунок 1.5 − Умовне позначення логічного елемента ТА-НІ та його таблиця істинності

Операція АБО-НІ − логічне додавання з інверсією. Ця логічна функція виконується як інверсія результату логічного додавання, у спеціальній літературі також має назву «стрілка Пірса». Аналітична форма запису цієї функції має вигляд:

$$Y = \overline{x_1 \vee x_0}.$$

Умовне позначення елемента та його таблиця істинності наведена на рис. 1.6.



x_1	x_0	Y
0	0	1
0	1	0
1	0	0
1	1	0

Рисунок 1.6 − Умовне позначення логічного елемента АБО-НІ та його таблиця істинності

Операція ТА-АБО-НІ. Елемент, який виконує таку операцію також використовується при побудові цифрових схем. Функція реалізовується при послідовному виконанні вказаних логічних операцій над чотирма вхідними логічними змінними. Аналітичний вираз цієї функції має такий вигляд:

$$Y = \overline{(x_3 \wedge x_2) \vee (x_1 \wedge x_0)}.$$

Умовне позначення елементу показано на рис. 1.7.

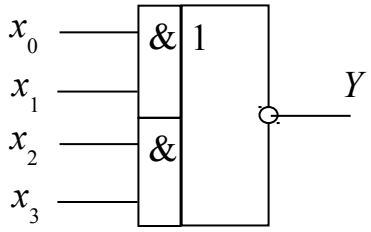
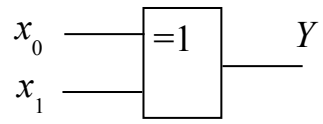


Рисунок 1.7 − Умовне позначення логічного елемента ТА-АБО-НІ

Операція ВИЛУЧАЛЬНЕ АБО (XOR). Ця операція виконується як додавання за модулем 2, відповідно до таблиці істинності, яка показана на рис. 1.8 разом з умовним позначенням цього елемента.



x_1	x_0	Y
0	0	0
0	1	1
1	0	1
1	1	0

Рисунок 1.8 − Умовне позначення логічного елемента виключальне АБО та його таблиця істинності

Побудова схем цифрових пристроїв зазвичай виконується у певних базисах**.** *Базис* – це функційно повна система логічних елементів, тобто мінімальний набір (сукупність) логічних елементів за допомогою яких можливо побудувати логічну схему будь-якої складності. Базис, який складається з основних логічних функцій НІ, ТА, АБО має назву основний логічний базис. Крім цього, кожний з логічних елементів ТА-НІ і АБО-НІ також складають логічні базиси, які відповідно мають назви ТА-НІ і АБО-НІ. Логічний елемент ТА-АБО-НІ також складає відповідний логічний базис. Ці базиси мають назву універсальних.

Логічне проектування схем цифрових пристроїв охоплює низку проблем, які постають на початковому етапі розробки цифрового пристрою.

Звичайно, логічне проектування виконується у такій послідовності:

1) відповідно до опису алгоритму роботи пристрою, який подано в будь-який спосіб, складається таблиця істинності;

2) відповідно до таблиці істинності складається логічний вираз, який описує алгоритм роботи пристрою;

3) виконується аналіз логічного виразу з метою побудови різних варіантів математичного виразу і знаходження кращого з них відповідно до певних критеріїв;

4) побудова функційної (логічної) схеми пристрою із заздалегідь вибраного набору елементів.

1.4 Цифрові автомати (ЦА). Асинхронні і синхронні ЦА. Комбінаційні і послідовнісні цифрові пристрої

Перетворення інформації в обчислювальній техніці відбувається за допомогою електронних цифрових пристроїв (ЦП), які будуються з логічних елементів і забезпечують виконання арифметичних і логічних операцій над сукупністю цифрових вхідних сигналів X_i, перетворюючи їх у сукупність вихідних сигналів Y_j. Розподіляють два класи таких пристроїв: *комбінаційні схеми* (КС) і *послідовнісні схеми* – цифрові автомати (ЦА).

У *комбінаційних схемах* результат перетворення – сукупність цифрових вихідних сигналів у будь-який момент часу t_n залежить лише від сукупності (комбінації) цифрових сигналів X_i, які надходять на її входи у даний момент часу і не залежать від стану схеми, який передував надходженню сигналів X_i. У таких схемах відсутні елементи пам'яті, тому сигнали, які діють у такій схемі, не зберігається. Такі ЦП ще називають цифровими автоматами без пам'яті (примітивними автоматами). До КС відносяться досить прості елементи, вузли та блоки мікропроцесорних систем: шифратори, дешифратори, мультиплексори, демультиплексори, комбінаційні суматори, перетворювачі кодів, схеми контролю тощо.

До *послідовнісних схем* відносяться ЦП, в яких результат перетворення Y_j залежить не лише від комбінації цифрових сигналів X_i, які надходять на її входи у даний момент часу t_n, а й від послідовності попередніх цифрових станів входів і виходів схеми – внутрішніх станів Z_f. Для запам'ятовування попередніх станів така схема повинна мати елементи пам'яті. Подібні схеми називають *цифровими автоматами* (ЦА)**.** Можна стверджувати, що кількість внутрішніх станів і кількість елементарних запам'ятовувальних пристроїв в цифрових автоматах зв'язані між собою такою залежністю $Z = 2^k$, де k – кількість запам'ятовувальних елементів.

Загальна теорія ЦА поділяється на *абстрактну* і *структурну*. Абстрактна теорія досліджує поведінку автомата відносно зовнішнього середовища, на рівні алгоритмів його роботи, не вирішуючи задач його побудови. Важливий висновок, який отримано у цих дослідженнях, полягає в тому, що в ЦА можливо реалізувати нескінченні регулярні події, на відміну від КС, де є можливість реалізувати лише скінченні події. Регулярними подіями вважаються такі події, які у скінченному алфавіті $X = \{x_1, x_2, \ldots, x_n\}$ можливо отримати з елементарних слів алфавіту X при здійсненні над ними операцій диз'юнкції, перемноження та ітерації. Нерегулярні події реалізувати у ЦА неможливо, тому що це вимагає нескінченного об'єму пам'яті. На практиці звичайно розглядаються ЦА, які мають обмежений об'єм пам'яті і називаються *скінченними ЦА***.**

Структурна теорія досліджує проблеми побудови ЦА, кодування вхідних сигналів і реакції схеми на них. Використовуючи апарат бульової алгебри, структурна теорія надає важливі рекомендації щодо розробки схем пристроїв обчислювальної техніки.

Абстрактний ЦА можна повністю описати за допомогою сукупності п'яти об'єктів:

- множини вхідних сигналів (вхідний алфавіт) – $X = \{x_0, x_1 \ldots x_i\}$;
- множини вихідних сигналів (вихідний алфавіт) – $Y = \{y_0, y_1 \ldots y_j\}$;
- множини стійких станів ЦА (алфавіт станів ЦА) – $Z = \{z_0, z_1 \ldots z_f\}$;
- функції переходів – φ, яка визначає стан ЦА Z у залежності від вхідного сигналу і попереднього стану;
- функції виходів – λ, яка визначає залежність вихідного сигналу від вхідного і попереднього станів.

У залежності від способу формування функції виходів абстрактні ЦА поділяються на *цифрові автомати Мілі* і *цифрові автомати Мура*.

Якщо закон функціонування ЦА можна описати системою наступних рівнянь:

$$\begin{cases} y(t+1) = \lambda\{z(t), x(t)\} & \text{– вихідний сигнал ЦА після перемикання;} \\ z(t+1) = \varphi\{z(t), x(t)\} & \text{– стан ЦА після перемикання,} \end{cases}$$

де $t = 0, 1, 2, \ldots$, то такий ЦА називається автоматом Мілі. Отже вихідний сигнал після переключення залежить від стану $z(t)$, в якому ЦП знаходився у момент надходження сигналу і вхідного сигналу $x(t)$.

В обчислювальній техніці широко використовуються пристрої, які можуть бути описані як ЦА Мура, який описується іншою системою рівнянь:

$$\begin{cases} y(t+1) = \lambda\{x(t)\} & \text{– вихідний сигнал ЦА після перемикання;} \\ z(t+1) = \varphi\{z(t), x(t)\} & \text{– стан ЦА після перемикання,} \end{cases}$$

де $t = 0, 1, 2, \ldots$.

Звідси видно, що вихідний сигнал такого ЦА не залежить від стану автомата, а повністю визначений вхідним сигналом. Відповідні структурні схеми автоматів Мура і Мілі показано на рис. 1.9,а і 1.9,б.

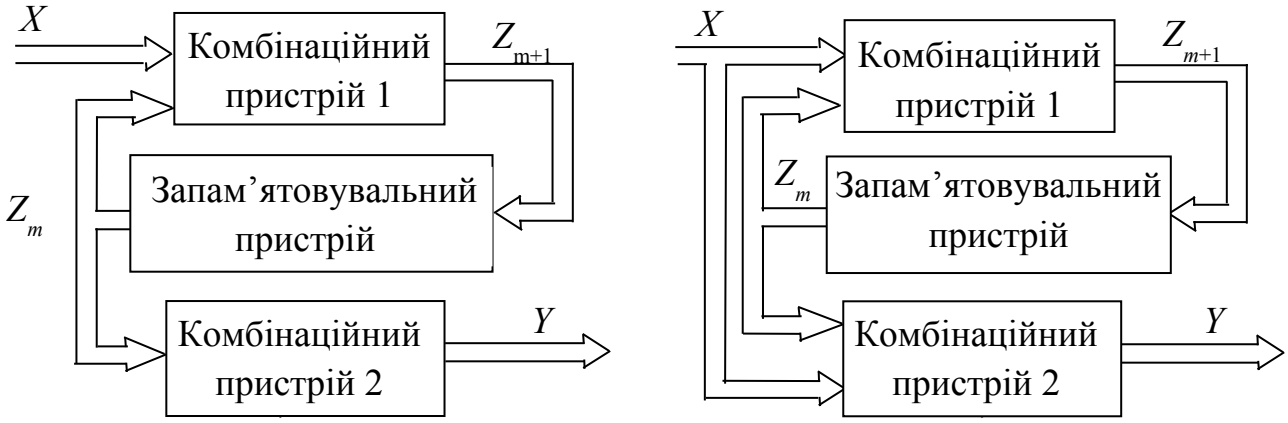


Рисунок 1.9 – Структурні схеми автоматів Мура і Мілі

Абстрактний ЦА, у якого виділено спеціальний початковий стан z_0 називається *ініціальним*. Звичайно, усі вихідні сигнали у такому початковому стані дорівнюють нулю. Виділення такого стану обумовлено вимогами практичного використання пристроїв, в яких він є обов'язковим для правильного функціонування всієї системи. Таким чином, описати роботу ініціальних ЦА можна так: якщо на вхід автомата Мілі або Мура, який знаходиться у початковому стані, подавати послідовність символів вхідного алфавіту $x(0)$, $x(1)$, $x(2)$, ... $x(i)$, то на виході отримаємо послідовність символів вихідного алфавіту. Тому на рівні абстрактної теорії функціонування ЦА описується як перетворення вхідного алфавіту у вихідний.

Якщо у ЦА функція переходів і функція виходів визначені для всіх пар X_i і Z_f, то він називається *повністю визначеним*. ЦА, для якого функція переходів

або функція виходів, або обидві разом не визначені для всіх пар X_i і Z_f, називається *частково визначеним*.

Реальні ЦА завжди скінченні, тому що множина вхідних і вихідних сигналів, а також множина внутрішніх станів обмежені.

Функціонування ЦА відбувається у певні часові інтервали – такти. Залежно від способу визначення такту ЦА поділяються на *синхронні* і *асинхронні*.

Синхронні ЦА змінюють свій внутрішній стан у строго визначені моменти часу, які задаються за допомогою спеціального пристрою – генератора тактових імпульсів (ГТІ), який формує послідовність імпульсів спеціального сигналу синхронізації. При цьому тривалість тактового інтервалу визначена і залишається постійною протягом усього часу роботи схеми.

Асинхронні ЦА змінюють свій стан у довільні моменти часу, які визначаються моментом надходження (зміни) вхідних сигналів. При цьому, тривалість тактового інтервалу точно не визначена і дорівнює часу між сусідніми змінами вхідних сигналів. Зміна вхідних сигналів може відбуватися лише після завершення поточного такту.

Структурна схема ЦА у загальному виді показана на рис. 1.10 (ЦА Мура).

До цієї схеми входять: КС1 – комбінаційна схема 1, яка формує сигнали запису інформації (вхідний алфавіт) у запам'ятовувальний пристрій – ЗП, який може бути побудований з тригерів або інших пристроїв, які здатні запам'ятовувати інформацію. Цей пристрій фіксує внутрішній стан (алфавіт станів) ЦА і зберігає його протягом такту. КС2 – комбінаційна схема 2, яка формує вихідні сигнали (вихідний алфавіт) ЦА.

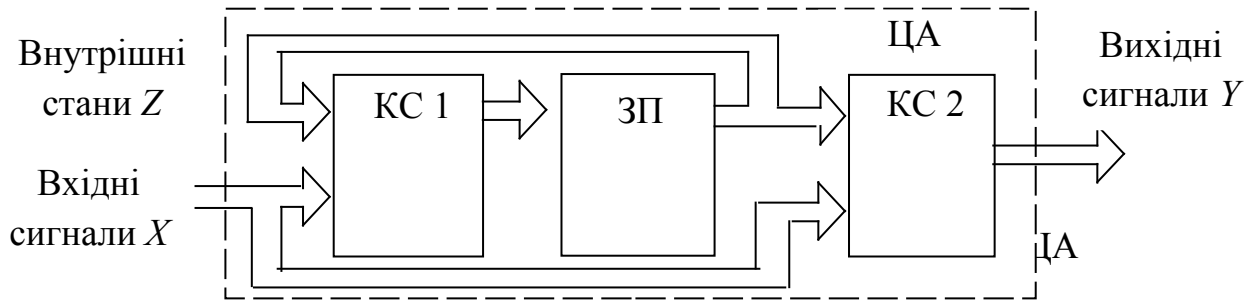


Рисунок 1.10 – Структурна схема ЦА у загальному виді

Слід зазначити, що як вхідні сигнали також використовуються всі або будь-яка частина сигналів внутрішніх станів ЦА, які по ланцюгах зворотного зв'язку надходять на входи КС1 з виходів ЗП. На входи КС2 можуть надходити вхідні сигнали або деякі з них. Наявність таких зв'язків необхідна для спрощення схем КС1 і КС2.

Описування алгоритму роботи абстрактного ЦА можна задати трьома способами: *матричним*, *графічним* і *аналітичним*.

При використанні матричного способу алгоритм роботи ЦА подається у вигляді таблиці переходів і таблиці виходів або у вигляді матриці з'єднань.

Таблиця переходів визначає функцію переходів ЦА – φ, а таблиця виходів – функцію виходів λ. Таблиця переходів для деякого конкретного ЦА наведена

у табл. 1.1. Вона будується таким чином, що стовпці цієї таблиці визначають символи станів цього автомата (вважаємо, що кількість бітів інформаційних сигналів відповідних кожному стану дорівнює 4), і вхідного алфавіту, символи якого перемикають ЦА у наступні стани (вважаємо, що в описуваному ЦА вхідний алфавіт складається з чотирирозрядних двійкових чисел). Рядки таблиці задають символи алфавіту станів у порядку їх появи. Вважаємо, що цей ЦА є ініціальним і після проходження циклу перемикань ЦА повертається у початковий стан Z_0. Тому що ЦА описується лише чотирма станами, то він є частково визначеним (усього в цього ЦА може бути $Z = 2^4 = 16$ стійких станів).

Таблиця 1.1 – Таблиця переходів ЦА

Стани ЦА	Побітне значення алфавіту станів ЦА				Побітне значення алфавіту вхідних сигналів			
	z_3	z_2	z_1	z_0	x_3	x_2	x_1	x_0
Z_0	0	0	0	0	0	1	0	1
Z_1	0	1	0	1	0	1	1	1
Z_2	0	1	1	1	1	0	0	1
Z_3	1	0	0	1	1	0	1	1
Z_4	1	0	1	1	–	–	–	–

Таблиця виходів ЦА визначає вихідні сигнали у кожному зі стійких станів. Таблиця виходів ЦА, який описано в табл. 1.1, подана в табл. 1.2.

Таблиця 1.2 – Таблиця виходів ЦА

Стани ЦА	Побітне значення алфавіту станів ЦА				Вихідний алфавіт	Побітне значення алфавіту вихідних сигналів								
	z_3	z_2	z_1	z_0		y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
Z_0	0	0	0	0	Y_0	0	0	0	0	0	0	0	0	0
Z_1	0	1	0	1	Y_1	0	0	1	1	0	0	0	1	1
Z_2	0	1	1	1	Y_2	0	0	1	1	0	1	0	1	1
Z_3	1	0	0	1	Y_3	0	1	0	0	1	0	1	1	0
Z_4	1	0	1	1	Y_4	1	1	1	0	1	0	1	1	1

При графічному описуванні функціонування робота ЦА описується орієнтованим графом, вершини якого відповідають стійким станам, а переходи зі стана в стан показують дугами між відповідними вершинами. Кожній дузі відповідає деяка літера вхідного алфавіту, яка викликає перемикання, і відповідна літера вихідного алфавіту. Вид спрямованого графа, що відповідає алгоритму робо[illegible] подано на

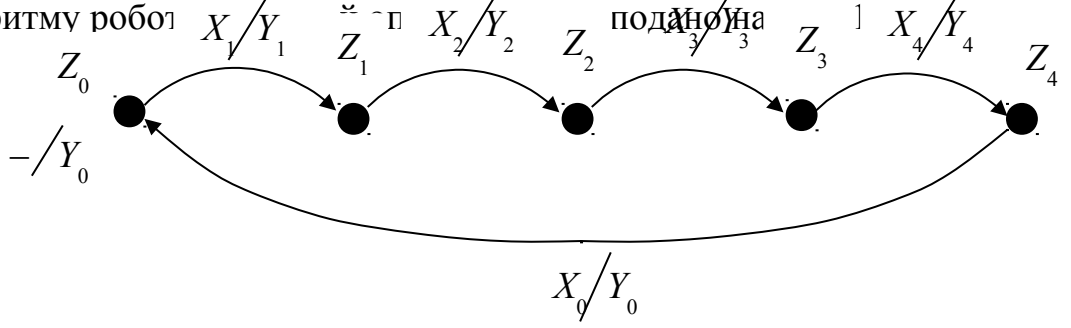


Рисунок 1.11 – Спрямований граф роботи цифрового автомата

При аналітичному описуванні роботи ЦА алгоритм його роботи задається у вигляді логічних функцій, що відповідають залежностям між різними характеристиками ЦА, які були описані вище.

При розробленні принципової схеми ЦА вигляд і склад запам'ятовувального пристрою, як правило визначено, тому задача розроблення ЦА зводиться до задачі проектування цифрових пристроїв – комбінаційних схем КС 1 і КС 2, які представляють собою логічні схеми.

ЦА широко використовуються в обчислювальній техніці та в галузі зв'язку як перетворювачі інформації і для генерування сигналів керування певними об'єктами. При цьому в залежності від інформації, що надійшла, буде виконуватись один з багатьох можливих алгоритмів. Стосовно до обчислювальної техніки такий алгоритм характеризує роботу і мікропроцесорної системи, і мікропроцесора, і їхніх вузлів.

Розв'язання задачі синтезу передбачає побудову структурної і електронної схем пристрою відповідно до опису алгоритму його роботи, для чого необхідно зробити вибір певних логічних елементів і передбачити їхнє з'єднання таким чином, щоб забезпечити виконання правил його функціонування, які подано в аналітичній формі. При розв'язанні цієї задачі необхідно зробити мінімізацію синтезованої схеми і виконати оцінку параметрів принципової схеми комбінаційного цифрового пристрою (КЦП).

Правила функціонування логічних схем (ЛС) можуть задаватися у різний спосіб:

- словесне описування;
- за допомогою таблиць істинності;
- за допомогою часових діаграм;
- аналітичний – за допомогою логічних (бульових) виразів;
- у вигляді координатних діаграм.

Усі ці способи описують один і той самий процес, тому по одному з них легко уявити функціонування ЛС і перейти від одного виду описування до іншого.

Слід зазначити, що для спрощення розв'язання задачі синтезу загальна ЛС поділяється на декілька схем, кожна з яких має один вихід (із сукупності вихідних сигналів Y_j) і стільки входів, скільки входить у сукупність вхідних сигналів X_i, необхідних для формування Y_j. Для більшого спрощення дозволяється подальше роздрібнення схеми таким чином, щоб зменшити кількість вхідних сигналів X. Після закінчення синтезу схем усі ці дрібні схеми

об’єднують в одну загальну схему КЦП. Розв’язання задачі синтезу виконується за декілька етапів.

На першому етапі відбувається описування функціонування ЛС, як правило, у вигляді таблиці істинності, яка описує кожен із стійких станів ЦП, як співвідношення вхідних і вихідного сигналів, і кожен рядок таблиці відповідає певному часовому інтервалу роботи. Описування виконується переважно для схем, які мають один вихід Y, а кількість входів рекомендується вибирати не більше ніж п’ять. Таблиця істинності повинна описувати всі можливі стани роботи ЛС, тому кількість рядків такої таблиці дорівнює кількості можливих станів. Якщо ЛС є частково визначеною, то значення вихідної функції позначається як ~.

На другому етапі відбувається подання описування роботи ЛС в аналітичному виді – у вигляді досконалої диз’юнктивної нормальної форми (ДДНФ) або у вигляді досконалої кон’юнктивної нормальної форми (ДКНФ). Вибір форми подання можна обумовити співвідношенням кількості нулів та одиниць у вихідному сигналі (логічній функції) і їх розміщенням за номерами рядків таблиці істинності або технічним завданням на проведення синтезу схеми.

На третьому етапі відбувається мінімізація схеми КЦП. Найбільш просто і наочно рекомендується проводити мінімізацію за допомогою координатних діаграм (карти Карно або діаграми Вейча). Різниця між цими способами полягає лише у поданні системи координат на діаграмі. На рис. 1.12, а показано вид карти Карно, а на рис. 1.12, б – вид діаграми Вейча.

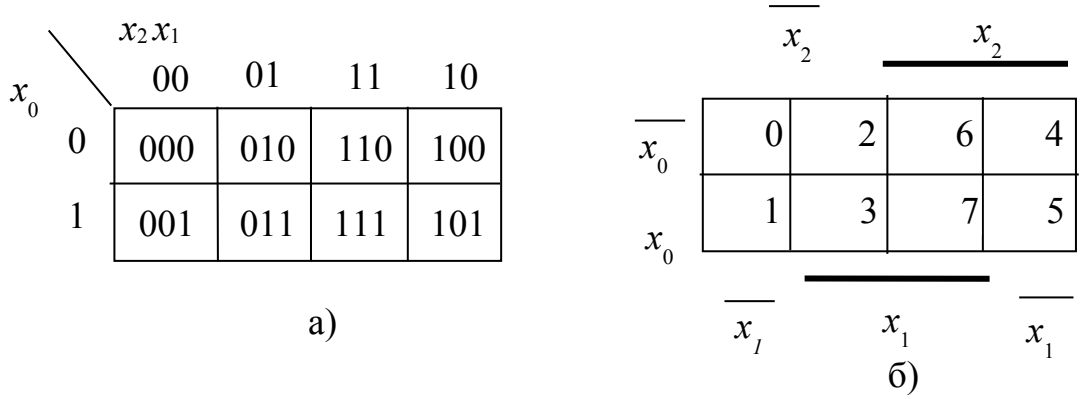


Рисунок 1.12 – Приклади координатних діаграм

Кількість можливих комбінацій значень вхідних сигналів (логічних змінних) X, а також кількість можливих стійких станів (логічних функцій) дорівнює $K = 2^N$, де N – кількість вхідних сигналів.

У результаті мінімізації отримуємо аналітичну функцію, яка складається з мінімальної кількості логічних функцій, за допомогою яких вона реалізується у вигляді мінімальної диз’юнктивної нормальної форми (МДНФ) або мінімальної кон’юнктивної нормальної форми (МКНФ).

Критеріями оптимальності для цих способів є розробка схеми ЛС з мінімальною кількістю логічних елементів і мінімальною кількістю з’єднань

між ними, внаслідок чого схема буде споживати мінімум енергії і вартість її буде менше, ніж у інших можливих.

На четвертому етапі за отриманими МДНФ або МКНФ можливо побудувати принципову схему КЦП у базисі ТА-АБО-НІ. Якщо принципову схему необхідно мати в іншому базисі, то виконуємо необхідне перетворення і будуємо схему у відповідному базисі ТА-НІ (АБО-НІ).

1.5 Аналіз і синтез ЦА. Синтез комбінаційного ЦА: мінімізація логічної функції та її реалізація у вигляді логічної схеми.

Як приклад зробимо синтез схеми комбінаційного цифрового пристрою для керування одним з елементів семисегментного індикатора, який відображує десяткові цифри 0...9. Структурна схема пристрою показана на рис. 1.13.

Рисунок 1.13 – Структурна схема КЦП

1. Таблиця істинності, що завдає алгоритм роботи цього цифрового пристрою наведена в табл. 1.3.

У цій таблиці у рядках стовпчика *Y* одиничні значення відповідають відлікам часу у які елемент індикатора є активним (горить), а значення, що дорівнюють 0, відповідають не активному стану елемента. Рядки, у яких стан елементу є не визначенним, позначені символом ~ .

2. За даними цієї таблиці запишемо вираз у вигляді досконалої диз'юнктивної нормальної форми (ДДНФ) або у вигляді досконалої кон'юнктивної нормальної форми (ДКНФ). Досконалість логічних функцій передбачає включення у вираз, який отримується, усіх логічних функцій, які використовуються для формування логічної функції. Будь-який з цих двох способів однозначно описує алгоритм роботи ЛС.

Таблиця 1.3 – Таблиця істинності цифрового пристрою

№ набору вхідних змінних	x_3	x_2	x_1	x_0	Y
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
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	~

Для подання у вигляді ДДНФ вибираємо рядки таблиці, на яких значення вихідної функції Y дорівнює $L1$ (логічній одиниці) і записуємо у вигляді кон'юнкції (логічного добутку) набори вхідних змінних кожного рядка, які об'єднуємо у вигляді диз'юнкції (логічної суми). Математичний вираз, що описує функцію на одному рядку, називається *термом*. Вхідні змінні повинні входити у кон'юнкцію таким чином, щоб значення логічного добутку дорівнювало $L1$, для чого вхідні змінні, які мають значення $L0$ (логічного нуля) необхідно інвертувати. Так, для нульового набору (рядок таблиці з номером 0) кон'юнкція вхідних змінних матиме вигляд:

$$Y_0 = \bar{x}_3 \wedge \bar{x}_2 \wedge \bar{x}_1 \wedge \bar{x}_0 \ .$$

Рядки таблиці, на яких логічна функція не визначена, не беруться до уваги. Тоді логічна функція у вигляді ДДНФ буде записана як

$$Y_{\text{ДДНФ}} = \left(\overline{x_3} \wedge \overline{x_2} \wedge \overline{x_1} \wedge \overline{x_0}\right) \vee \left(\overline{x_3} \wedge \overline{x_2} \wedge x_1 \wedge \overline{x_0}\right) \vee \left(\overline{x_3} \wedge x_2 \wedge x_1 \wedge \overline{x_0}\right) \vee$$
$$\vee \left(x_3 \wedge \overline{x_2} \wedge \overline{x_1} \wedge x_0\right).$$

Для подання логічної функції у вигляді ДКНФ, вибираємо рядки, на яких значення вихідної функції Y дорівнює логічному нулю $L0$, і записуємо логічну функцію у вигляді кон'юнкції (логічного добутку) диз'юнкцій вхідних логічних змінних кожного рядка. Для опису вихідної логічної функції Y у вигляді $L1$ логічні зміні кожного рядка, які входять у вираз ДКНФ, необхідно проінвертувати. ДКНФ логічної функції має вигляд:

$$Y_{\text{ДКНФ}} = \left(x_3 \vee x_2 \vee x_1 \vee \overline{x_0}\right) \wedge \left(x_3 \vee x_2 \vee \overline{x_1} \vee \overline{x_0}\right) \wedge \left(x_3 \vee \overline{x_2} \vee x_1 \vee x_0\right) \wedge$$
$$\wedge \left(x_3 \vee \overline{x_2} \vee \overline{x_1} \vee \overline{x_0}\right) \wedge \left(\overline{x_3} \vee x_2 \vee x_1 \vee \overline{x_0}\right).$$

3. Мінімізацію досконалих логічних функцій виконуємо за допомогою діаграми Вейча, яка для логічної функції з чотирма вхідними змінними матиме вигляд квадрата 4×4 клітинки. Діаграма Вейча для логічної функції, яка описана табл. 1.3, наведена на рис. 1.14.

Для мінімізації логічної функції за одиницями або за нулями на діаграмі Вейча необхідно визначити мінімальну кількість найбільших груп (контурів), що складаються з одиниць або нулів, у вигляді квадратів або прямокутників (об'єднувати можна лише сусідні клітинки). Сусідніми вважаємо клітинки, що мають спільні сторони, при цьому клітинки, розміщені по горизонтальних та вертикальних сторонах квадрата, є сусідніми. Взагалі, ознакою сусідства можливо вважати відміну двійкового коду номера клітинки лише в одному розряді. Таким чином, аналізуючи номери клітинок у верхньому і нижньому рядках, а також у правому і лівому, можливо зробити висновок, що відповідні клітинки у цих рядках також є сусідніми, і діаграму Вейча можна згорнути у циліндр по горизонталі і вертикалі. Кількість клітинок у групі повинна дорівнювати цілому степеню двійки, тобто 1, 2, 4, 8, 16. На рис. 1.14 показано контури, які об'єднують одиниці і нулі поданої логічної функції. Контури об'єднань можуть будь-яку кількість разів перетинатися, але не можуть повністю суміщатися. Якщо функція має невизначені стани (символ – ~), то функції цієї клітинки можливо надати значення 0 або 1 таким чином, щоб отримати контур, до якого буде входити більша кількість клітинок з однаковими значеннями 0 або 1.

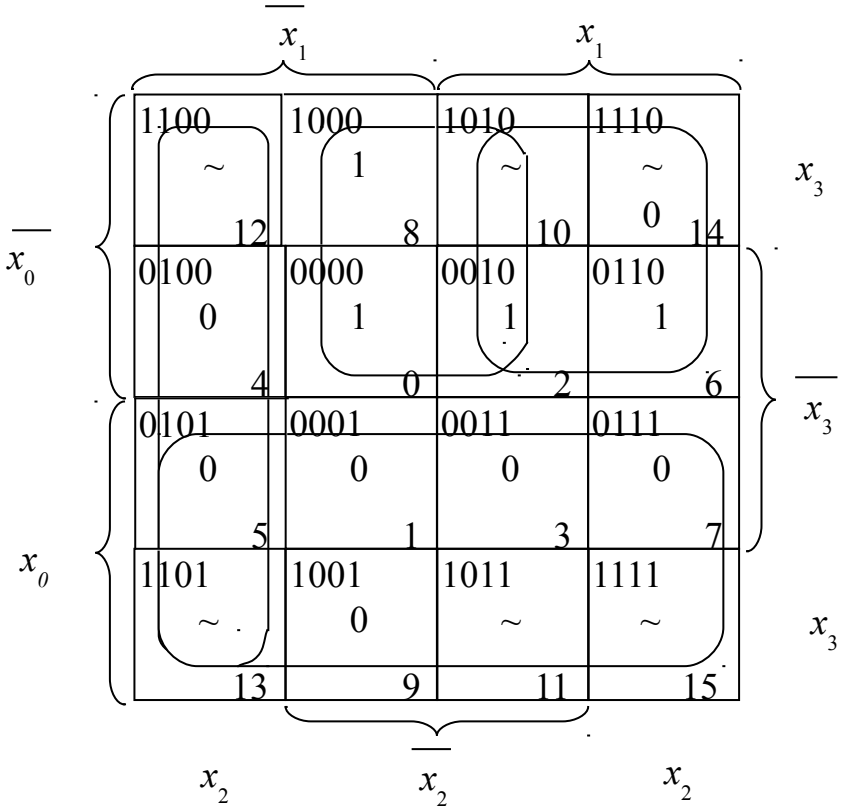


Рисунок 1.14 – Діаграма Вейча з контурами для одиниць і нулів

Таким чином, об'єднавши 1 і невизначені стани, отримали два контури, до яких входять клітинки з номерами 0, 2, 8, 10 і 2, 6, 10, 14. Мінімальною формою логічної функції для кожного з контурів будуть координати контурів на діаграмі з відсутніми координатами, на яких функція Y змінює своє значення. Так, координата x_1 у контурі з номерами клітинок 0, 2, 8, 10 змінює своє значення; у стовпчику з клітинками 0, 8 вона має значення 0, а у стовпчику з номерами клітинок 2, 10 – 1. Таким чином, координата x_1 буде відсутня при записуванні мінімальної ДНФ (МДНФ).

Запишемо мінімальні логічні функції, які отримуємо після мінімізації

$$Y_{\text{МДНФ}} = \left(\overline{x_2} \wedge \overline{x_0}\right) \vee \left(x_1 \wedge \overline{x_0}\right).$$

$$Y_{\text{МКНФ}} = \left(x_2 \vee \overline{x_1}\right) \wedge x_0.$$

Оцінимо $K_{\text{зв}}$ для мінімізованих функцій:

для МДНФ $K_{\text{зв}}$ = 2НІ + 2ТА + 1АБО = 5,

для МКНФ $K_{\text{зв}}$ = 1НІ + 1АБО + 1ТА = 3.

4. По одному з отриманих у попередньому пункті виразах МДНФ або МКНФ можемо побудувати схему логічного пристрою у базисі ТА-АБО-НІ. Критерієм вибору є менше значення $K_{\text{зв}}$ або завдання на розробку пристрою.

Логічні схеми будують, як правило, на елементах одного типу в базисах ТА-НІ, АБО-НІ. Це робиться для того, щоб запобігти розкиду параметрів часових затримок у схемі.

Для перетворення виразу до необхідного базису використовується правило подвійної інверсії $\overline{\overline{x}} = x$ та закон де Моргана у такому вигляді:

$$\overline{x_1 \vee x_0} = \overline{x_1} \wedge \overline{x_0}$$

$$\overline{x_1 \wedge x_0} = \overline{x_1} \vee \overline{x_0}.$$

Простіше перетворювати вираз МДНФ до базису ТА-НІ, а вираз МКНФ до базису АБО-НІ, взагалі можливі будь-які перетворення, але при перетворенні МДНФ до базису АБО-НІ і навпаки МКНФ до базису ТА-НІ, схеми будуть більші на один логічний елемент.

Виконаємо перетворення МДНФ у базис ТА-НІ

$$Y_{\text{МДНФ}} = \left(\overline{x_2} \wedge \overline{x_0}\right) \vee \left(x_1 \wedge \overline{x_0}\right) = \overline{\overline{\left(\overline{x_2} \wedge \overline{x_1}\right) \vee \left(x_1 \wedge \overline{x_0}\right)}} = \overline{\overline{\left(\overline{x_2} \vee \overline{x_0}\right)} \vee \overline{\left(x_1 \vee \overline{x_0}\right)}},$$

і перетворення МКНФ у базис АБО-НІ

$$Y_{\text{МКНФ}} = \left(x_2 \vee \overline{x_1}\right) \wedge x_0 = \overline{\overline{\left(x_2 \vee \overline{x_1}\right) \wedge x_0}} = \overline{\overline{\left(x_2 \vee \overline{x_1}\right)} \vee \overline{x_0}}.$$

За цими виразами будується схема дешифратора. На рис. 1.15 схема подана у базисі ТА-НІ, а на рис. 1.16 – у базисі АБО-НІ.

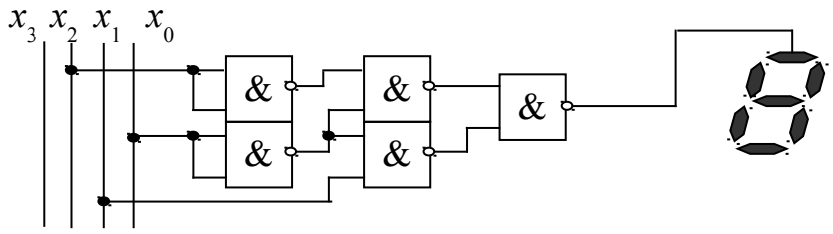


Рисунок 1.15 – Схема дешифратора у базисі ТА-НІ

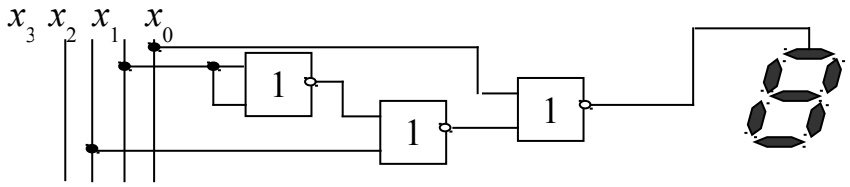


Рисунок 1.16 – Схема дешифратора у базисі АБО-НІ

1.6 Комбінаційні цифрові пристрої: перетворювачі кодів, дешифратори, шифратори, мультиплексори, демультиплексори, суматори

Комбінаційні цифрові пристрої (КЦП) – це клас цифрових пристроїв, які не мають внутрішніх запам'ятовуючих елементів і тому стан вихідних сигналів у кожний момент часу однозначно визначений набором (комбінацією) вхідних сигналів, які надходять у цей момент. До цього класу відносяться наступні типові цифрові пристрої: перетворювачі довільних кодів, шифратори, дешифратори, мультиплексори, демультиплексори, комбінаційні напівсуматори і суматори, цифрові компаратори, арифметико-логічні пристрої, схеми прискоренного перенесення тощо.

Перетворювачі довільних кодів. Ці пристрої призначені для перетворення будь-якого довільного коду, який побудовано за недостатньо відомими правилами, у певний заданий. В цьому випадку, єдиною формою синтезу такого пристрою, є побудова таблиці істинності. Після чого синтез такого пристрю відбувається так, як було розглянуто вище.

Прикладами такого перетворювача можуть слугувати: перетворювач двійкового коду в двійково-десятковий та навпаки, перетворювач двійкового коду в код Грея тощо. Умовне графічне позначення перетворювача довільних кодів має вигляд, який показано на рис. 1.17.

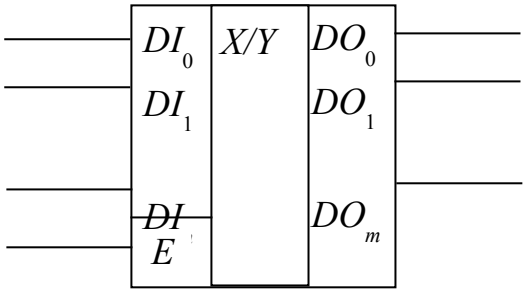


Рисунок 1.17 – Умовне графічне позначення перетворювача довільних кодів

Виводи перетворювача кодів, який показано на рис. 1.17 мають назву: DI_0…. DI_n (*Date Input*) – входи даних, DO_0…. DO_m (*Date Output*) – виходи, *E* (*Enable*) – вхід дозволу роботи мікросхеми.

Шифратори. Шифратор або кодер (*encoder*) – це цифровий пристрій призначений для перетворення унітарного (одиночного) коду в позиційний (двійковий) код. Шифратор має *m* входів, кількість яких відповідає кількості символів унітарного коду й *n* виходів, необхідних для представлення найбільшого вхідного символу.

Якщо вихідний код є двійковим, то такий шифратор має назву двійкового. В такому шифраторі кількість входів і виходів визначені наступним співвідношенням:

$$m = 2^n.$$

Шифратор може використовуватися в пристроях уводу інформації, наприклад, для відображення номера кнопки, яку було натиснуто, положення багатопозиційного перемикача і таке інше.

Умовне позначення шифратора показано на рис. 1.18

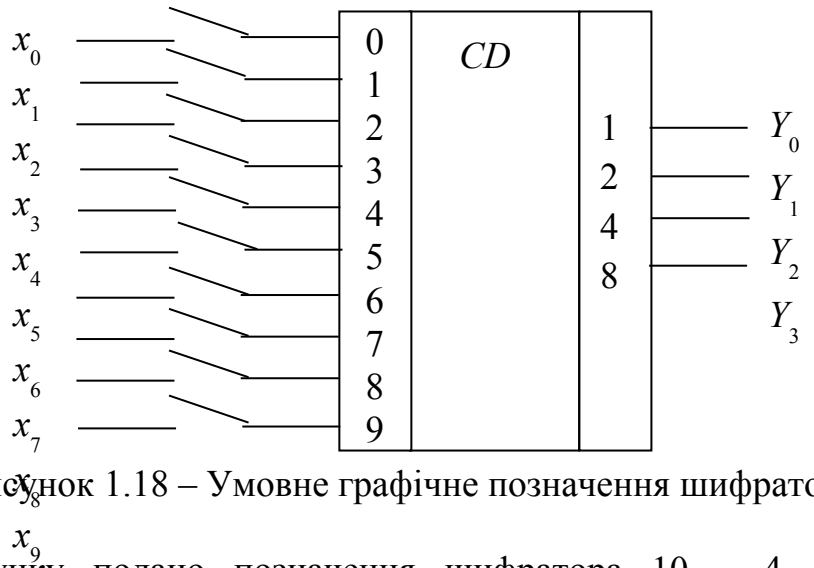


Рисунок 1.18 – Умовне графічне позначення шифратора

На рисунку подано позначення шифратора 10 – 4, який виконує перетворення активного сигналу з одного входу в двійковий сигнал, який відображує номер цього входу.

У реальних схемах заборонено подавати сигнали одночасно на декілька входів, тому в реальних схемах частіше використовуються пріоритетні шифратори, в яких при надходженні декількох сигналів вихідній сигнал відповідає лише молодшому або найстаршому з вхідних.

Шифратори, в тому числі й пріоритетні випускаються як інтегральні схеми середнього ступеня інтеграції, які можуть виконуватися за різними технологіями (ТТЛ, ТТЛШ, КМОН тощо).

Дешифратори. Дешифратор або декодер (*decoder*) це пристрій, який перетворює вхідний сигнал у вигляді багаторозрядного (*n* – розрядів) позиційного коду (найчастіше двійкового) в унітарний (*m* – розрядів). Кількість розрядів вхідних і вихідних кодів визначається співвідношенням

$$m = 2^n$$

Дешифратор, кількість вхідних і вихідних сигналів якого точно відповідає цьому співвідношенню, має назву *повного дешифратора*. Також використовуються неповні дешифратори, в яких кількість вихідних сигналів m може бути меншою ніж 2^n.

У цифровий та обчислювальній техніці дешифратори використовуються для декодування певного коду в набір сигналів керування для різних виконавчих пристроїв, а також для дешифрування адрес пристроїв, які входять до складу системи.

Також дешифратори можуть мати один або декілька сигналів дозволу роботи, які позначаються E (*enable*). Якщо таких сигналів декілька, то між собою вони поєднуються логічною функцією ТАК. Дешифратори з такими входами мають назву – декодер-демультиплексор. Умовне графічне позначення дешифраторів і декодерів-демультиплексорів подано на рис. 1.19.

Пристрої, які показано на рис. 1.16 і 1.17 використовуються для відображення стану процесу за допомогою семисегментного індикатора у вигляді цифр від 0 до 9.

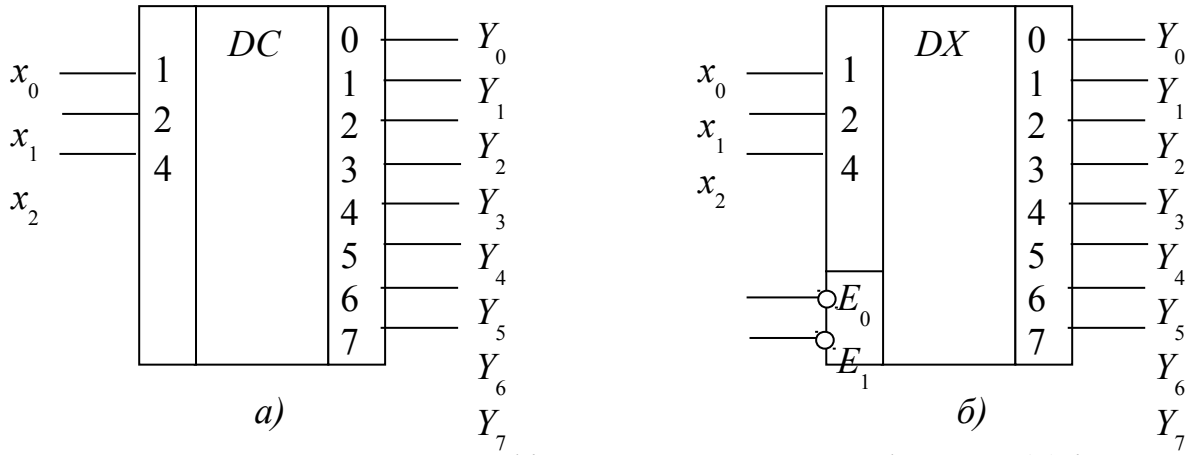


Рисунок 1.19 – Умовне графічне позначення дешифратора (*а*) і декодера-демультиплексора (б)

Дешифратори і декодери-демультиплексори випускаються як інтегральні схеми середнього ступеня інтеграції, які можуть виконуватися за різними технологіями (ТТЛ, ТТЛШ, КМОН тощо), вони відрізняються наявністю сигналів керування, кількістю розрядів вхідних й вихідних сигналів тощо.

Мультиплексори. Мультиплексор (multiplexor) – функційний пристрій, якій здійснює комутацію сигналу (даних) одного з декількох входів до одного виходу. При цьому, номер входу, який підключено до виходу визначається двійковим кодом сигналу, який подано на адресні входи мультиплексора.

Кількість входів даних і кількість адресних входів мультиплексора пов’язані співвідношенням

$$N_D = 2^{N_A},$$

де N_D – кількість входів даних;

N_A – кількість адресних входів.

Керування вибором адреси здійснюється за допомогою дешифратора, який є складовою частиною мультиплексора.

При циклічному, послідовному змінюванні адресних сигналів процес мультиплексування можна розглядати як процес об'єднання (часового ущільнення) сигналів для передавання їх по одному каналу зв'язку.

Для керування роботою мультиплексорів, які входять до складу складних пристроїв, також можуть використовуватись сигнали дозволу роботи *E*.

Умовне графічне позначення мультиплексора показано на рис. 1.20.

На рис. 1.20 показано мультиплексор, який виконує перетворення 4 – 1 (чотири сигнали даних в один вихід). Активним сигналом входу *E* є сигнал логічного 0.

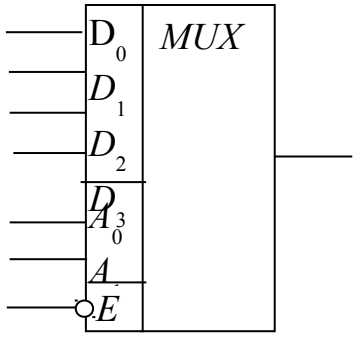


Рисунок 1.20 – Умовне графічне позначення мультиплексора

Демультиплексори. Демультиплексор виконує операцію зворотну до алгоритму роботи мультиплексора. Він має один інформаційний вхід та декілька виходів і виконує комутацію (розподілення) сигналу з цього входу на один з виходів. Номер виходу в кожну мить визначається двійковим кодом, який надходить на адресні входи.

Умовне графічне позначення демультиплексора, який виконує операцію зворотну до алгоритму роботи мультиплексора, який показано на рис. 2.3 наведено на рис. 1.21.

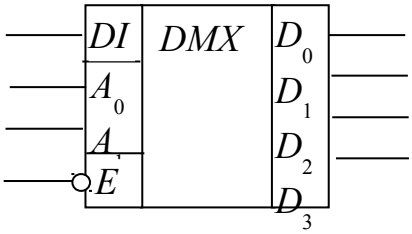


Рисунок 1.21 – Умовне графічне позначення демультиплексора

Суматори. Суматор це пристрій, якій виконує операцію арифметичного додавання чисел, які подані у двійковому коді.

Відповідно до правил виконання арифметичних операцій в позиційних системах числення, додавання багаторозрядних чисел виконується порозрядно, починаючи з молодшого розряду. Для реалізації такого алгоритму використовуються напівсуматори та суматори. Напівсуматор використовується

для виконання операції додавання без урахування вхідного перенесення, а суматор виконує додавання з урахуванням вхідного перенесення.

Для побудови багаторозрядного комбінаційного суматора напівсуматори об'єднуються таким чином, щоб розряди чисел, які додаються, були включені паралельно, а сигнали перенесення формувалися послідовно. Умовне графічне позначення однорозрядного комбінаційного суматора показано на рис. 1.22, *а*, а 4-розрядний суматор показано на рис. 1.22, б.

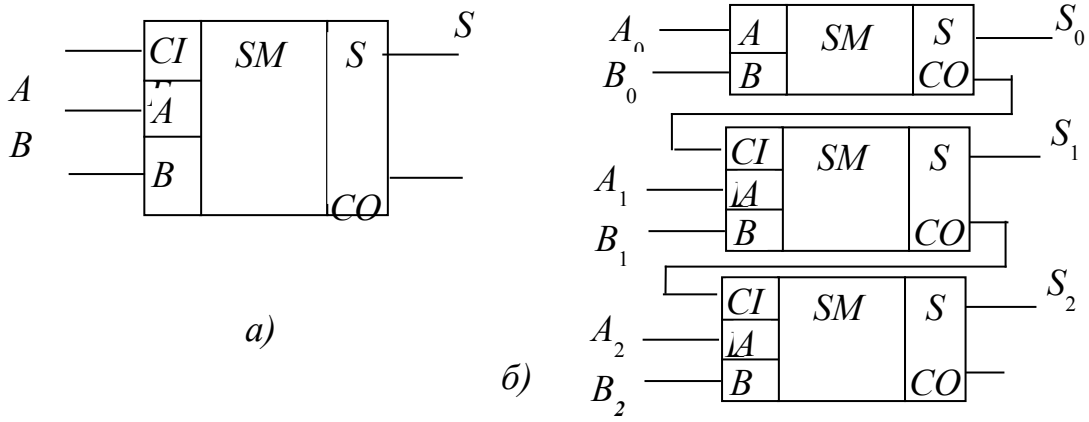


Рисунок 1.22 – Умовне графічне позначення суматора

На рис. 1.22 прийняті такі позначення:

CI – (*carry input*) вхідне перенесення;

CO – (*carry output*) вихідне перенесення;

S – (*sum*) результат додавання (порозрядний);

A_n , B_n – значення відповідних розрядів двійкових чисел.

Комбінаційний суматор можна також використовувати для виконання операції віднімання чисел зі знаками, для чого операнди, що надходять на схему, повинні подаватися у доповнювальному коді.

Для збільшення швидкодії можна використовувати суматор з паралельним перенесенням. У такому суматорі додавання виконується як порозрядна операція, і вхідне перенесення для кожного старшого розряду формується незалежно від формування перенесення у попередньому молодшому розряді. Для всіх розрядів сигнали перенесення також формуються паралельно. Формування сигналів перенесення відбувається в результаті формування й оброблення додаткових двох сигналів: розповсюдження перенесення (*CRP – Carry Propagation*) і генерування перенесення (*Carry Generation*).

1.7 Цифрові компаратори. Арифметико-логічні пристрої

Цифровим компаратором називають функційний вузол обчислювальної техніки, який порівнює два багаторозрядних числа *A* і *B* між собою, і результат порівняння подає у вигляді сигналів співвідношення між ними $A = B$, $A < B$, $A > B$.

Для порівняння сигналів використовується суматор, який виконує операцію віднімання (алгебраїчного додавання) вхідних чисел і за цими результатами формуються сигнали, що показують співвідношення між ними. За результатами виконання операції $A - B$ можна зробити такі висновки:

– якщо результат дорівнює 0, то це є ознакою того, що значення A і B співпадають;

– якщо результат не дорівнює 0, а вихідне переповнення дорівнює 1, то $A < B$;

– якщо вихідне переповнення дорівнює 0, то це є ознакою, що $A > B$.

Сигнали, що показують співвідношення між вхідними числами, формуються за допомогою логічних схем. Функційна схема цифрового компаратора для чотирирозрядних чисел наведена на рис. 1.23, *а*, а його умовне графічне позначення на рис. 1.23, *б.*

Для нарощування розрядності вхідних даних цифрові компаратори дозволяється каскадувати, для чого набір вхідних сигналів компаратора доповнюють вихідними сигналами попереднього каскаду $A = B$, $A < B$, $A > B$ (рис. 1.23, *б*). Формування кінцевого результату порівняння відбувається з урахуванням цих сигналів.

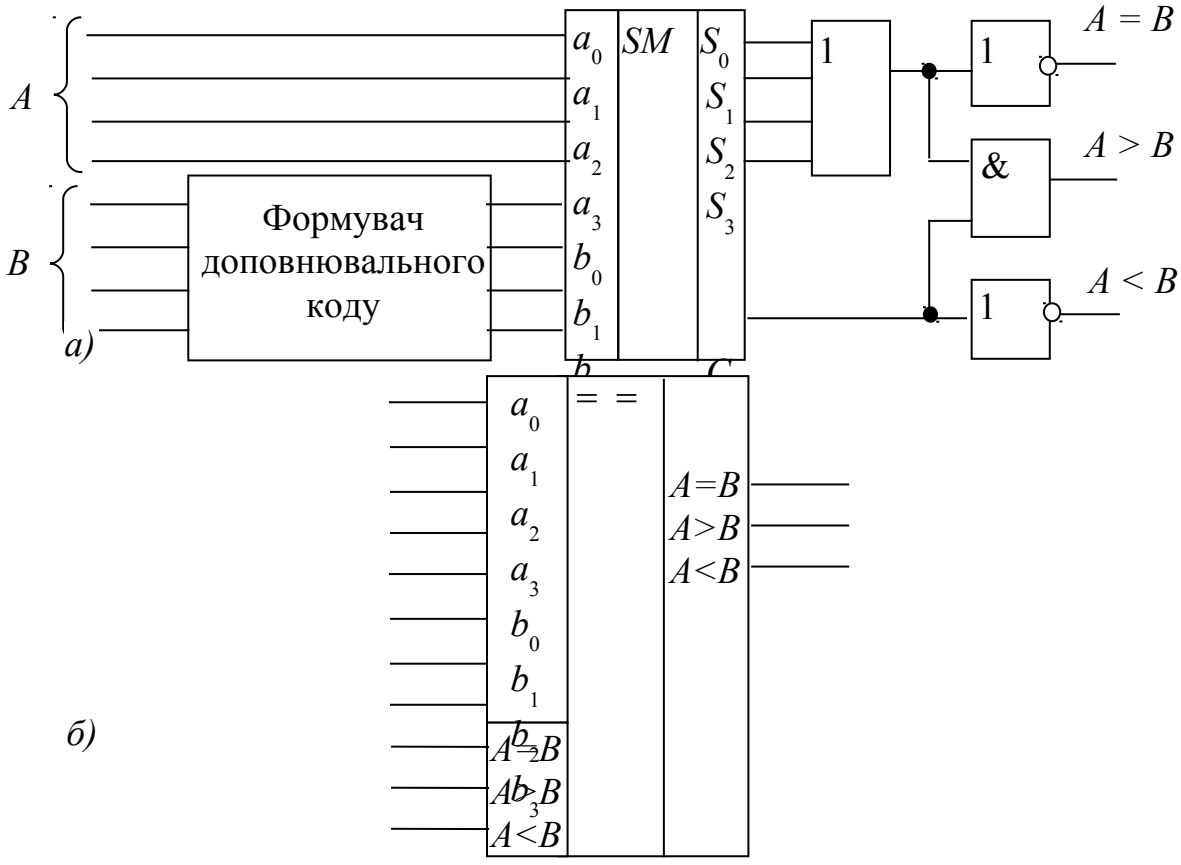


Рисунок 1.23 – Функційна схема цифрового компаратора (*а*) та його умовне графічне позначення (*б*)

Арифметико-логічні пристрої (АЛП) широко використовуються для побудови арифметичних вузлів, зокрема, АЛП є обов'язковою складовою

частиною будь-якого процесора. АЛП використовується для виконання арифметичних і логічних операцій з даними, які до нього надходять і являють собою числа або будь-який інший вид інформації.

Операції, які виконуються в АЛП, можна розділити на такі групи:

- арифметичні операції над двійковими числами з фіксованою точкою;
- арифметичні операції над двійковим числами з плаваючою точкою;
- операції десяткової арифметики;
- операції індексної арифметики (для модифікації адрес команд);
- логічні операції з операндами, які є логічними змінними;
- спеціальні арифметичні операції;
- операції над алфавітно-цифровими полями.

Арифметичні операції в АЛП виконуються за допомогою багаторозрядного суматора, а логічні – відповідними логічними схемами. В залежності від характеру використання цих пристроїв АЛП поділяються на **блочні** і **багатофункційні**. В блочних АЛП для виконання різних типів операцій використовуються окремі блоки. Це дає змогу підвищити швидкодію, але збільшує апаратні витрати. В багатофункційних АЛП для виконання операцій різних типів використовуються одні й ті самі пристрої, які комутуються по-різному, в залежності від виконуваної операції. Вибір операції і необхідних блоків здійснюється за допомогою сигналів керування, якими кодується операція, яку необхідно виконати. Крім результату АЛП формує набір спеціальних сигналів – ознаки результату (прапорці). Умовне позначення АЛП на схемах показано на рис. 1.24.

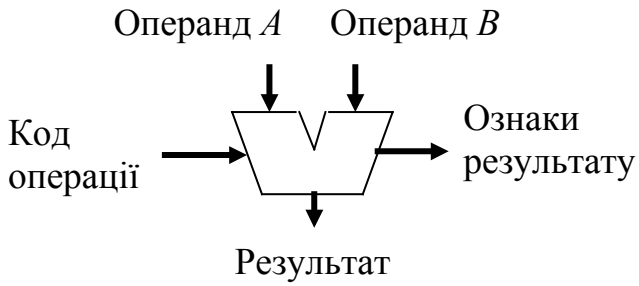


Рисунок 1.24 – Умовне графічне позначення АЛП

Тому що, двійковий суматор, який використовується в АЛП для виконання арифметичних операцій, є комбінаційним пристроєм, то для одночасного подання двох операндів на його входи необхідно мати запам’ятовувальні пристрої для тимчасового зберігання операндів і результату.

До АЛП операнди можуть надходити з двох регістрів загального призначення (РЗП) або із регістра і комірки пам’яті, а результат надходить до регістру або до комірки пам’яті, які визначені як приймач результату. Така система має назву *двохадресної*. Тому МПС повинна подати адреси обох операндів. Схему організації двохадресної системи показано на рис. 1.25, а.

В інших МПС один з операндів до початку виконання команди обов’язково зберігається в акумуляторі, а після закінчення результат записується на місці операнда, який безповоротно втрачається. Така система

має назву *одноадресної*, тому що необхідно адресувати лише один операнд. Схему організації одноадресної системи показано на рис. 1.25, б.

До важливої характерної ознаки АЛП відноситься формування ним ознак результату (прапорців) – тобто деяких властивостей результату, які можуть вплинути на подальше виконання програми. Ознаки результату формуються після отримання результату і записуються у спеціальний регістр – регістр ознак результату (або інші назви цього регістру – регістр прапорців, регістр стану), де вони зберігаються до формування результату наступної операції. Слід зазначити, що ознаки результату формують лише команди арифметичних і логічних операцій. Регістр ознак, у залежності від типу мікропроцесора, може мати різну кількість розрядів, які відповідають властивостям результату і зберігають інформацію про стан деяких апаратних або програмних компонентів процесора. Формат регістру ознак для відповідної мікросхеми може бути таким, як показано на рис. 1.26.

Кожна з ознак результату (прапорець) зберігається в одному розряді регістру і кодує наявність відповідної ознаки значенням 0 або 1. Розглянемо кожну з ознак результату з короткою характеристикою:

– *C (carry)* – ознака зберігає значення перенесення до наступного старшого розряду за межами розрядної сітки або значення позики з цього розряду;

– *S (sign)* – розряд регістру зберігає копію знакового розряду результату операції. Значення цієї ознаки, що дорівнює 1 відповідає від'ємному результату;

– *Z (zero)* – ознака нульового результату. Для результату, що дорівнює 0 – ознака набуває значення 1;

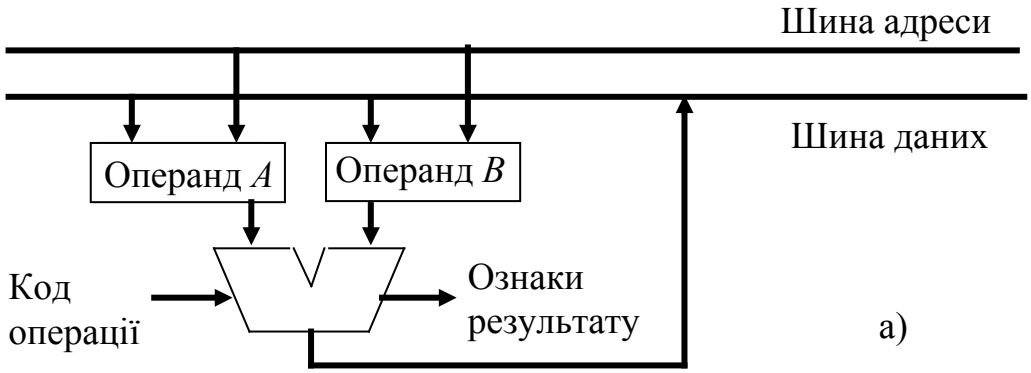
– *AC (auxiliary carry)* – ознака зберігає значення перенесення зі старшого розряду молодшої тетради у молодший розряд старшої тетради байта при додаванні двійково-десяткових чисел або – значення позики зі старшої тетради до молодшої при відніманні;

– *OVR (overflow)* – ознака, яка сигналізує про втрату старшого біта результату додавання або віднімання чисел зі знаком, що обумовлено наявністю перенесення у знаковий розряд. При додаванні ознака набуває значення 1, якщо є перенесення у старший (знаковий) біт результату, але немає перенесення з нього і навпаки якщо є перенесення у біт *C*, але немає перенесення у старший (знаковий) біт. При відніманні ознака дорівнює 1, якщо є позика зі старшого (знакового) біта результату, але немає позики з нього і навпаки якщо є позики з біта *C*, але немає позики зі старшого (знакового) біта. Значення ознаки дорівнює результату виконання логічної операції ВИКЛЮЧНЕ АБО над значеннями бітів *C* і перенесенням до старшого (знакового) біта результату

$$C \oplus C_{D7},$$

де C_{D7} – перенесення до старшого (знакового) біта результату.

– *P (parity)* – значення цієї ознаки дорівнює 0, якщо результат операції складається з непарної кількості одиниць.



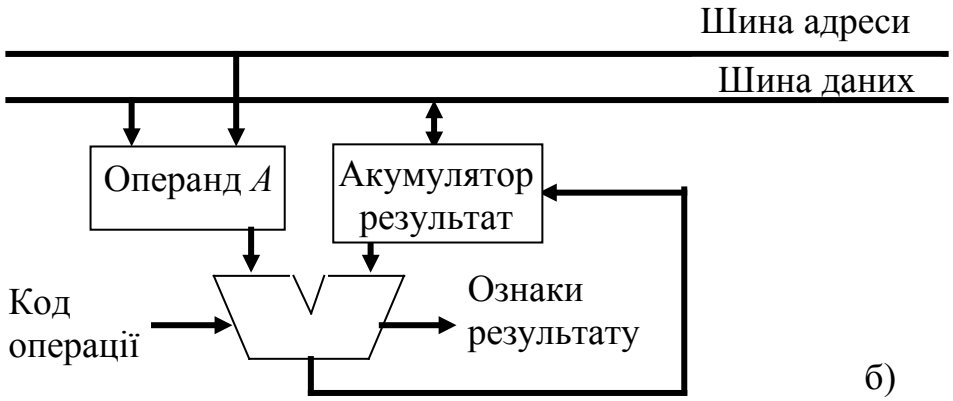


Рисунок 1.25 – Схема організації двохадресного (а) і одноадресного (б) АЛП

C	*S*	*Z*	*AC*	*OVR*	*P*
Ознака перенесення (позики)	Ознака знаку	Ознака нуля	Ознака допоміжного перенесення	Ознака переповнення	Ознака додатності

Рисунок 1.26 – Формат регістру і назви ознак результату

Формування ознак покажемо на прикладі виконання арифметичної і логічної операцій для восьмирозрядних чисел:

– арифметична операція над двійковими числами

$$\begin{array}{r} 11101111 \\ + \quad 01111000 \\ \hline 101100111 \\ \quad\; C \quad AC \end{array}$$

Якщо вважати, що операція виконується над беззнаковими числами, то це $239_D + 120_D = 359_D$ і результат з урахуванням перенесення правильний, якщо вважати що числа зі знаками, то це $(-17_D) + 120_D = +\ 103_D$. Враховуючи, що числа зі знаком подані в доповнювальному коді, то результат також

правильний. Ознаки результату не залежать від того над якими числами виконувалась операція. Вони набувають таких значень:

$C = 1$ – перенесення у дев'ятий розряд має місце;

$S = 0$ – восьмий розряд результату, що кодує знак, дорівнює 0, результат додатний;

$Z = 0$ – результат не дорівнює нулю;

$AC = 1$ – відбулося перенесення з молодшої тетради до старшої. Слід сказати, що АЛП виставляє ознаки, формально, за їх наявності, не враховуючи конкретні обставини виконання операції;

$OVR = 1$ – відбулося переповнення розрядної сітки (біт C) і не відбулося перенесення до старшого (знакового) розряду.

$$C \oplus D7 = 1 \oplus 0 = 1.$$

Якщо це числа зі знаком і передбачається їх подальше оброблення, то необхідно збільшити довжину розрядної сітки;

$P = 0$ – результат (без урахування перенесення) складається з непарної кількості (5) одиниць.

– логічна операція кон'юнкції над восьмирозрядними логічними змінними

$$\begin{array}{r} 11101111 \\ \wedge\ 01111000 \\ \hline 01101000 \end{array}$$

При виконанні логічних операцій перенесення не відбувається, тому ознаки C, AC і OVR не формуються, а зберігають свої значення, яких набули при попередній операції. Ознака S формується формально копіюванням старшого розряду результату. Після виконання логічної операції ознаки набувають таких значень:

$C = X$ – зберігає попереднє значення;

$S = 0$ – восьмий розряд результату, що кодує знак, дорівнює 0;

$Z = 0$ – результат не дорівнює нулю;

$AC = X$ – зберігає попереднє значення;

$OVR = X$ – зберігає попереднє значення;

$P = 0$ – результат складається з непарної кількості (3) одиниць.

Умовне графічне позначення мікросхеми чотирирозрядного АЛП наведено на рис. 1.27.

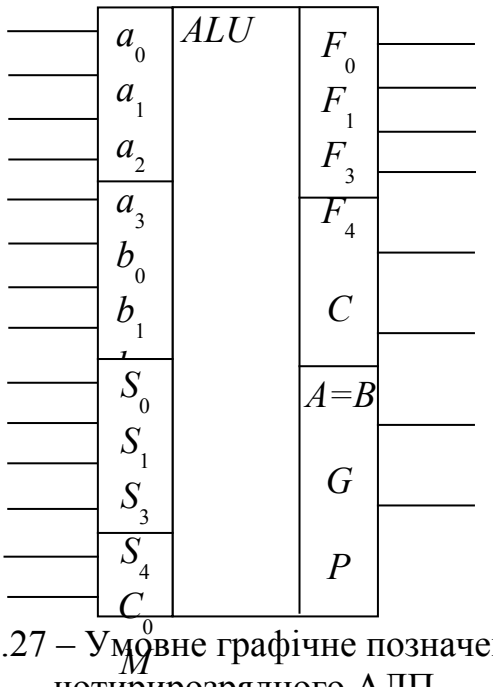


Рисунок 1.27 – Умовне графічне позначення мікросхеми чотирирозрядного АЛП

Виводи цієї мікросхеми мають таке функційне призначення:

- $a_0 – a_3$ – входи операнда A;
- $b_0 – b_3$ – входи операнда B;
- $S_0 – S_3$ – входи для подання коду операції, що буде виконуватись;
- C_0 – вхідне перенесення;
- M – код режиму роботи $M = 1$ – АЛП може виконувати 16 логічних операцій, $M = 0$ – 16 арифметичних операцій;
- $F_0 – F_3$ – результат виконання операції;
- C – вихідне перенесення;
- $A = B$ – ознака рівності вхідних операндів;
- G – вихід сигналу генерування перенесення;
- P – вихід сигналу розповсюдження перенесення.

Останні два сигнали використовуються для нарощування розрядності операндів і збільшення швидкодії схеми, яка буде отримана.

1.8. Послідовнісні цифрові пристрої. Тригери: *RS*- *D*- *T*- *JK*-тригер

Скінченні ЦА з невеликим обсягом внутрішньої пам’яті являють собою елементи послідовнісних цифрових пристроїв. До таких пристроїв відносяться **тригери**. За способом запису інформації тригери можна розподіляти на асинхронні і синхронні.

Асинхронні тригери. Тригер – це цифровий пристрій, в якому організовано перехрестні зворотні зв’язки і який може нескінченно довго знаходитися в одному з двох стійких станів, які мають назви одиничний і нульовий. Перемикання з одного стану в інший відбувається стрибком при

надходженні вхідного сигналу й відповідно до стану в якому триггер знаходився до перемикання.

Перехід тригера в одиничний стан (*set*) відбувається за наявності активного сигнала на вході, який позначається літерою *S*, а перехід тригера у нульовий стан (*reset*) – за наявності сигналу на вході *R*.

До асинхронних тригерів відносяться так звані *RS*-тригери.

Умовне графічне позначення *RS*-тригера показано на рис. 1.28. Видно, що такий тригер має два входи і два виходи: прямий Q та інверсний $\overline{Q}$.

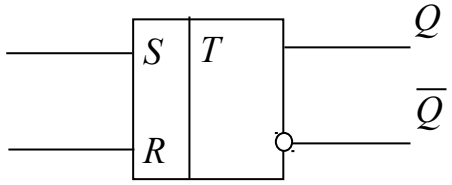


Рисунок 1.28 – Умовне графічне позначення RS-тригера

Алгоритм роботи будь-якого послідовнісного пристрою можливо описати за допомогою таблиці перемикань, яка наведена у табл. 1.4. В цій таблиці визначено стани тригера і названо режими роботи, які відповідають цим станам. Вважаємо, що активними сигналами, які призводять до перемикання тригера є сигнали логічної 1.

Таблиця 1.4 – Таблиця перемикань асинхронного *RS*-тригера

№	S_n	R_n	Q_n	Q_{n+1}	$\overline{Q_{n+1}}$	Режим
0	0	0	0	0	1	Зберігання інформації
1	0	0	1	1	0	
2	1	0	0	1	0	Встановлення 1
3	1	0	1	1	0	
4	0	1	0	0	1	Встановлення 0
5	0	1	1	0	1	
6	1	1	0	0	0	Заборонений стан
7	1	1	1	0	0	

У таблиці прийняті такі позначення: S_n , R_n , Q_n – сигнали, які діють на входах і на на прямому виході тригера перед спрацюванням, Q_{n+1} , $\overline{Q_{n+1}}$ – сигнали на прямому й інверсному виходах тригера після перемикання.

У режимі зберігання інформації на входи тригера не надходять активні сигнали і стан тригера не змінюється, інформація зберігається.

У режимі встановлення 1, активний сигнал надходить на вхід запису 1 (вхід *S*), а на вхід *R* надходить неактивний сигнал 0 – в тригер записується 1.

У режимі встановлення 0, активний сигнал надходить на вхід запису 0 (вхід *R*), а на вхід *S* надходить неактивний сигнал 0, – в тригер записується 0.

При поданні активного сигналу на обидва входи *S* і *R* тригер буде знаходитися у забороненому стані, який характеризується тим, що будуть порушені закони роботи тригера, і на обох виходах формуються однакові

сигнали, що дорівнюють 0. При переході із забороненого режиму в режим зберігання інформації у тригері буде зберігатися випадкова інформація.

За допомогою таблиці перемикань можна синтезувати схему тригера. Синтез будемо виконувати за допомогою діаграми Вейча, як було показано вище. В якості вхідних змінних будемо використовувати сигнали *S, R* і сигнал поточного стану Q_n , а в якості вихідного значення – сигнал $\overline{Q_{n+1}}$. Діаграму Вейча будуємо для двох виходів Q_{n+1} і $\overline{Q_{n+1}}$. Діаграми наведені на рис. 1.29. На рис. 1.29, а наведено діаграму Вейча для виходу $\overline{Q_{n+1}}$, вона реалізує схему по одиницям і на рис. 1.29, б для виходу Q_{n+1}, яка реалізує схему по нулях.

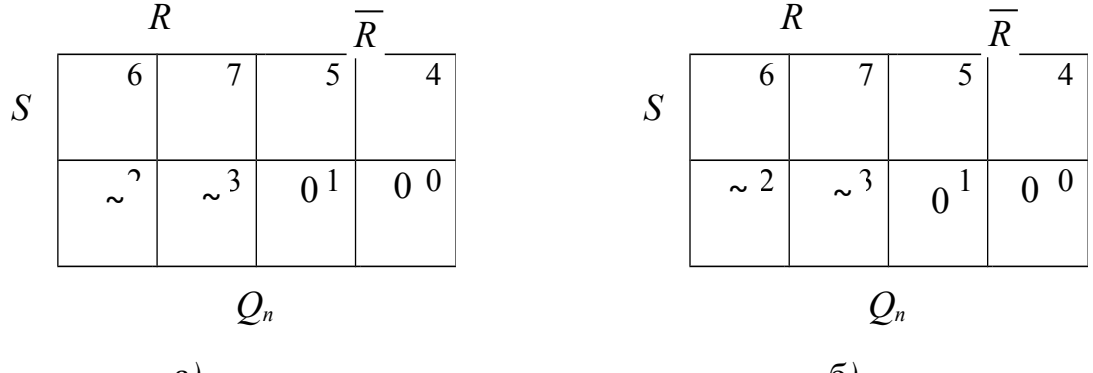


Рисунок 1.28 – Діаграми Вейча для *RS*-тригера

Клітинки, які відповідають забороненому стану роботи можуть вважатися невизначеними і використовуватися для мінімізації в якості будь-якого символу.

Відповідно до рис. 1.28, а отримуємо логічний вираз для формування сигналу Q_{n+1}, який має вигляд:

$$Q_{n+1} = \overline{R \vee (\overline{S \vee Q_n})} , \qquad (1.1)$$

і відповідно рис. 1.29, б отримуємо логічний вираз для формування сигналу $\overline{Q_{n+1}}$, який має вигляд:

$$\overline{Q_{n+1}} = \overline{S \vee (\overline{R \vee Q_n})} . \qquad (1.2)$$

Відповідно до цих виразів схема *RS*-тригера буде мати вгляд, як показано на рис. 1.30. Такий тригер має назву асинхронний *RS*-тригер з прямими входами, тому що активними рівнями сигналів, відповідно до виразів будуть сигнали логічної 1.

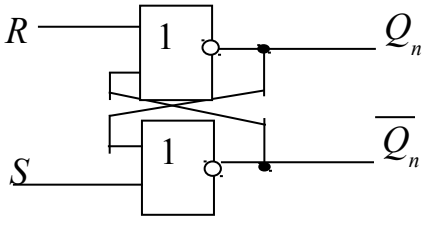


Рисунок 1.30 – Функційна схема *RS*-тригера з прямими входами

Роботу цього тригера можливо описати за допомогою часових діаграм, як показано на рис. 1.31.

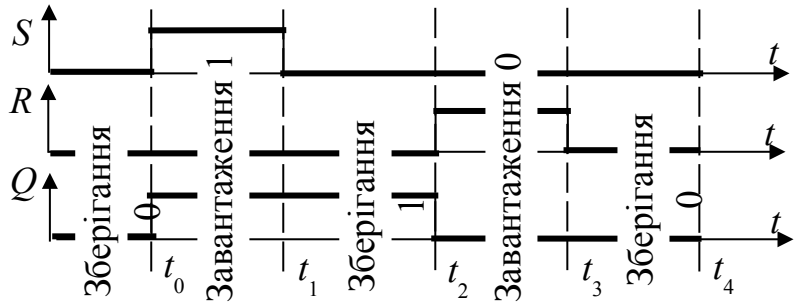


Рисунок 1.31 – Часові діаграми, які описують роботу асинхронного *RS*-тригера з прямими входами

Вважаємо, що до початку аналізу роботи, (час до моменту t_0) тригер знаходиться у стані зберігання 0. У момент t_0 на вхід *S* надходить сигнал логічної 1, а на вході *R* зберігається неактивний стан логічного 0. Така комбінація сигналів відповідає режиму запису 1, тому на виході *Q* сигнал встановлюється у стан логічної 1. Вихід *Q* не показано, тому що на ньому буде встановлено стан інверсний до виходу *Q*. В наступні моменти (t_1 …t_4) стани тригера будуть змінюватися відповідно до написів на рисунку.

Аналогічно можливо побудувати асинхронний *RS*-тригер на елементах ТА-НІ. Схема такого тригера наведена на рис. 1.32.

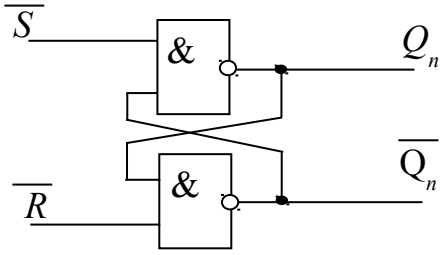


Рисунок 1.32 – Функційна схема *RS*-тригера з інверсними входами

Ця схема має назву тригер з інверсними входами, тому що активними сигналами для керування є сигнали логічного 0.

Асинхронні тригери миттєво спрацьовують при надходженні вхідного активного сигналу. Однак у багатьох випадках необхідно синхонізувати перемикання всіх пристроїв схеми, тому що відсутність одночасно спрацьовування може призвести до непередбачених наслідків.

Синхронні тригери. Синхронні тригери відрізняються від асинхронних наявністю окремого додаткового входу для приймання спеціального сигналу – сигналу синхронізації (сигналу тактової частоти) (*clock*) , який визначає момент

спрацьовування тригера. Тобто, в такому тригері є дві групи входів. Одна – для приймання інформаційних сигналів, а друга – визначає моменти часу в які проводиться цей запис. Ці моменти є загальними для всіх вузлів, які входять до складу всієї схеми (пристрою).

Використання сигналу синхронізації обумовлено можливістю «перегонів» (*races*) між сигналами, які проходять по паралельних колах схеми, як показано на рис. 1.33.

На рис. 1.33 показано, що вхідний сигнал розгалужується на дві частини схеми. У кожній з цих частин власне значення часу затримки сигналу Δt_1 і Δt_2 , тому сигнали R і S надійдуть на входи тригера з розбіжностями в часі. Ці розбіжності можуть призвести до невірного спрацьовування тригера.

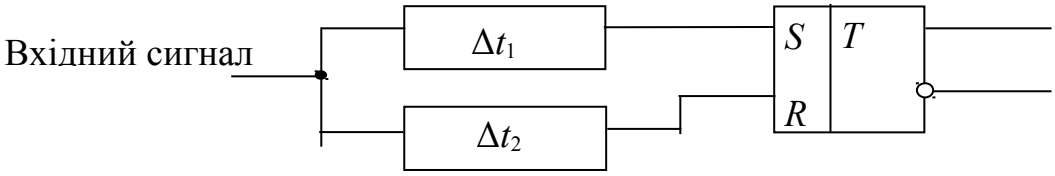


Рисунок 1.33 – Схема, яка пояснює механізм «перегонів»

Для організації входу для приймання сигналу синхронізації схема асинхронного тригера доповнюється комбінаційною схемою, яка забезпечує спрацьовування тригера лише за умови наявності сигналу синхронізації.

Синхронний RS-тригер. Схема найпростішого синхронного *RS*-тригера показана на рис. 1.33, а і його умовне позначення на рис. 1.34, б.

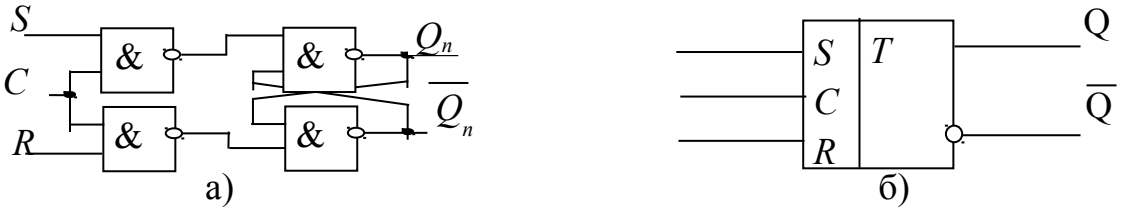


Рисунок 1.34 – Схема й умовне позначення синхронного *RS*-тригера

Запис інформації у цей тригер відбувається за наявності активного сигналу на вході синхронізації C (сигнал логічної 1), а при неактивному сигналі – тригер знаходиться у стані зберігання інформації й не реагує на зміни сигналів на інформаційних входах. Перемикання цього тригера наведені у табл. 1.5.

У синхронних тригерів є можливість організувати ще одну групу входів, які є асинхронними і мають назву *входи попереднього запису*. Вони мають назви R і S, відповідно до інформації, яку кожен записує.

Характерною особливістю такого тригера є те, що поки діє сигнал синхронізації він є «прозорим», тобто сигнали з інформаційних входів проходять на вихід схеми, що зменшує завадостійкість цього тригера.

Таблиця 1.5 – Таблиця перемикань синхронного *RS*-тригера

№	C	S_n	R_n	Q_n	Q_{n+1}	$\overline{Q_{n+1}}$	Режим
0	0	0	0	0	0	1	Зберігання 0
1	1	0	0	0	0	1	
2	0	0	0	1	1	0	Зберігання 1
3	1	0	0	1	1	0	
4	1	1	0	0	1	0	Встановлення 1
5	1	1	0	1	1	0	
6	1	0	1	0	0	1	Встановлення 0
7	1	0	1	1	0	1	
8	1	1	1	0	1	1	Заборонений стан
9	1	1	1	1	1	1	

***D*-тригер**. Для запобігання переходу тригера у заборнений стан можна об'єднати входи R і S через інвертор, як показано на рис. 1.35, *а*, або використовуючи можливості елемента ТА-НІ, як показано на рис. 1.35, б. В цьому випадку інформаційний сигнал буде один. Такий тригер має назву *D*-тригер (*delay* – затримка), а об'єднаний вхід – назву *D*.

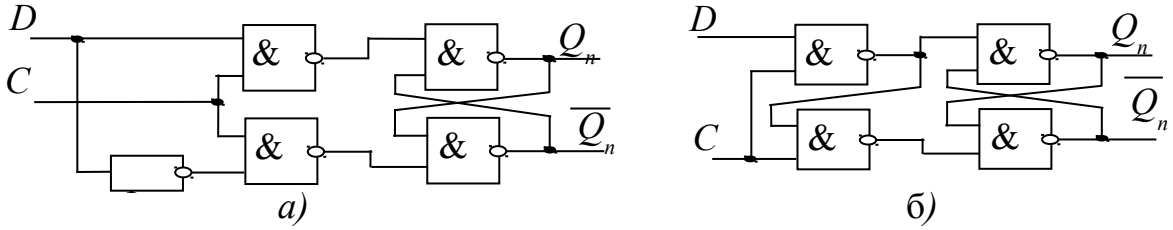


Рисунок 1.35 – Схеми побудови *D*-тригера

D-тригер буде записувати інформацію зі входу D за наявності сигналу логічної 1 на вході синхронізації, тобто *D*-тригер, який побудовано за такою схемою також є «прозорим».

Часові діаграми сигналів на виводах *D*-тригера і його графічне позначення показано на рис. 1.36.

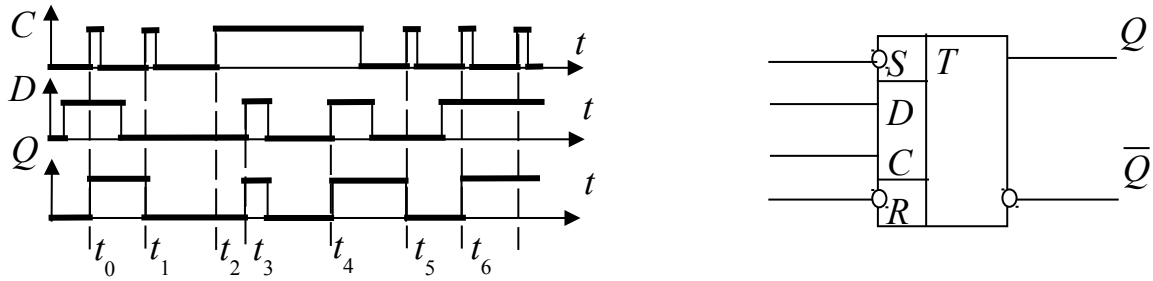


Рисунок 1.36 – Часові діаграми роботи та його графічне позначення *D*-тригера

На рис. 1.36, *а* видно, що у момент часу t_0 відбувається запис 1 у тригер, а у момент t_1 – запис 0, з моменту t_2 показано як тригер працює під час «прозорості». Так, у момент t_3 на вхід D надходить імпульс, який проходить на

вихід тригера, а у момент t_4 надходить імпульс, який продовжується після закінчення синхроімпульсу, тому значення 1, яке було записано у тригер зберігається до моменту t_5. Взагалі моменти t_3 і t_4 можна вважати дією завад.

Двоступеневі тригери (*MS*-тригери). Ці тригери розроблено для підвищення завадостійкості тригерів розглянутих вище. Вони вміщують два синхронних *RS*-тригери, які включено послідовно. Перший з цих тригерів має назву ведучого або *M*-тригера (*master* – хазяїн), а другий – ведений або *S*-тригер (*slave* – раб). Схема такого тригера наведена на рис. 1.37.

M-тригер і *S*-тригер спрацьовують у різні моменти часу. Тому коли на вхід синхронізації схеми, надходить сигнал з рівнем 1, то він дозволяє запис у *M*-тригер, але забороняє спрацьовування *S*-тригера.

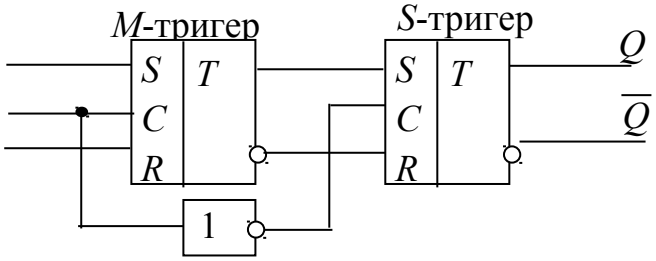


Рисунок 1.37 – Схема двоступеневого синхронного *RS*-тригера

Таким чином, незважаючи на те що кожний ступінь *MS*-тригера «прозорий», в будь-який момент часу один з них заблокована, і інформація з інформаційних входів не проходить на вихід. Тригер змінює свій стан лише по зрізу сигналу синхронізації (у моменти часу t_1, t_2 і t_4), як показано на рис. 1.38. У моменти t_0 і t_3 тригер знаходиться у стані збереження інформації.

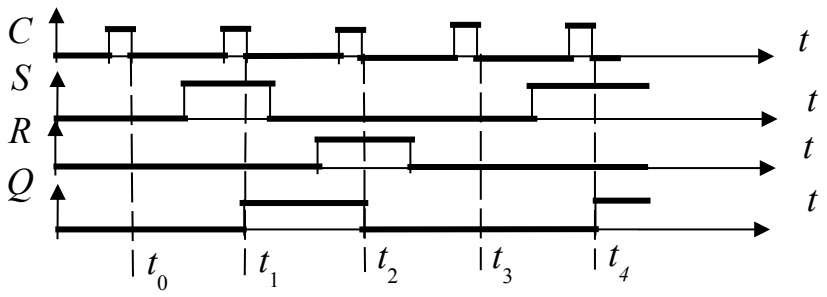


Рисунок 1.38 – Часові діаграми роботи двоступеневого синхронного RS-тригера

***JK*-тригер.** Для усунення заборонених станів схема ускладнюється, і вона набуває вигляду, який показано на рис. 1.39. Такий тригер має назву *JK*-тригер. Він є непрозорим і в ньому немає заборонених станів. Активний сигнал на вході *J* і зріз імпульсу синхронізації призводить до запису 1 у тригер, а активний сигнал на вході *K* і зріз імпульсу синхронізації записує у тригер 0.

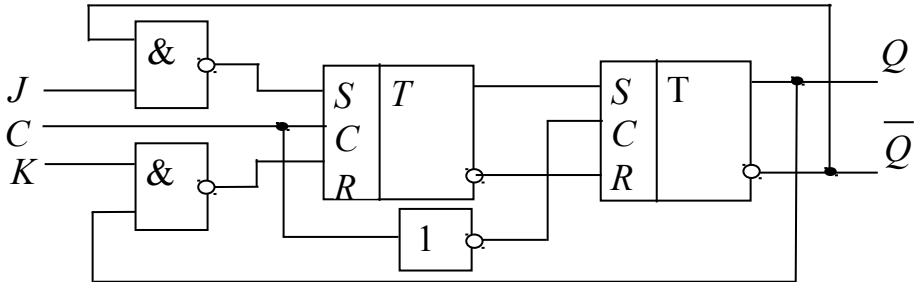


Рисунок 1.39 – Схема *JK*-тригера

При одночасній появі активних сигналів на обох входах, зріз імпульсу синхронізації змінює стан тригера на інверсний до поточного. Такий режим має назву режим **лічби імпульсів**. Перемикання цього тригера наведені у табл. 1.6.

Таблиця 1.6 – Таблиця перемикань *JK*-тригера

J	K	C	Q_n	Q_{n+1}	Режим
0	0	↓	0	0	Збереження інформації
0	0	↓	1	1	Збереження інформації
1	0	↓	1	1	Запису 1
1	0	↓	0	1	Запису 1
0	1	↓	1	0	Запису 0
0	1	↓	0	0	Запису 0
1	1	↓	0	1	Лічби імпульсів
1	1	↓	1	0	Лічби імпульсів

У цій таблиці символом ↓ показано, що перемикання тригера буде відбуватися за наявності зрізу синхроімпульсу на вході С.

Умовне позначення *JK*-тригера показано на рис. 1.40.

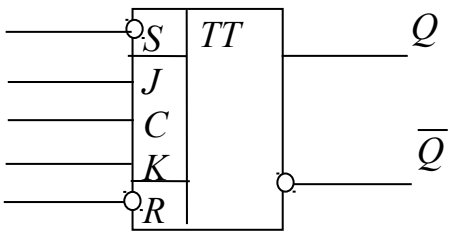
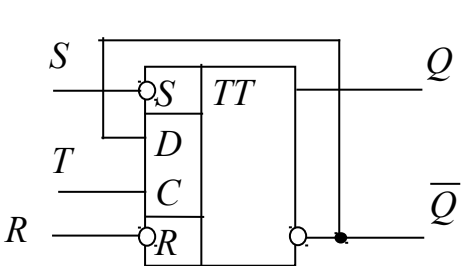
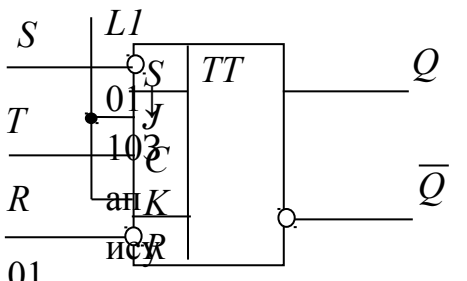


Рисунок 1.40 – Умовне графічне позначення *JK*-тригера

***T*-тригери.** Для побудови послідовних лічильників імпульсів використовуються тригери з одним асинхронним входом, які змінюють свій стан при надходженні на вхід кожного імпульсу. Такі тригери мають назву *T*-тригери. Такі тригери можуть бути побудовані з *D*-тригерів або *JK*-тригерів. Схеми показано на рис. 1.41.





01
↓1
03
ап
ис
у
0

↓1
03
ап
ис
у
0

а) б)

Рисунок 1.41 – Схеми перетворення для *D*- і *JK*-тригерів

Так, на рис. 1.42 показано часові діаграми вхідного і вихідних для *D*-тригера (Q_D) і *JK*-тригера (Q_{JK}), які працюють у режимі лічби імпульсів.

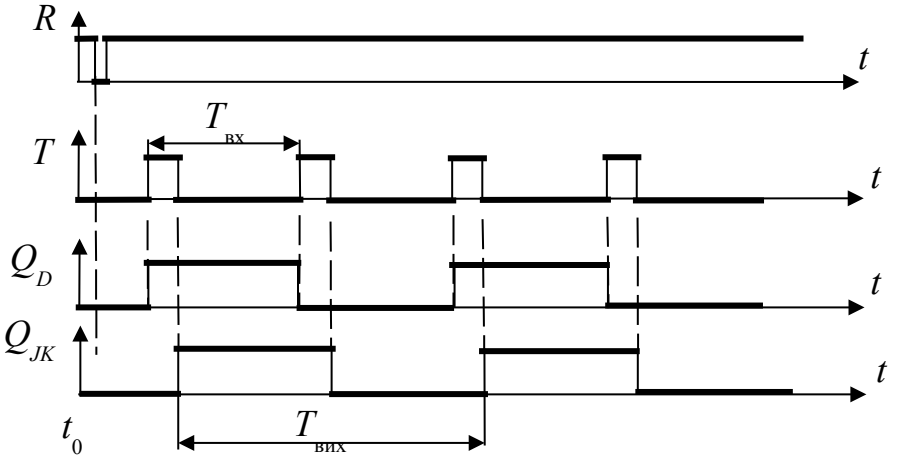


Рисунок 1.42 – Часові діаграми

1.9 Регістри. Паралельний регістр. Послідовний регістр. Універсальний регістр

Регістр – це послідовнісний ЦП (цифровий автомат), який призначено для тимчасового зберігання інформації, яка подана у вигляді двійкових слів і виконання деяких операцій над ними. Кожен розряд двійкового слова зберігається в окремій комірці, яка являє собою тригер, тип якого визначається вибором схеми керування процесами запису/зчитування інформації в регістрі. Регістри можуть бути синхронним або асинхронними.

Регістри використовуються для:

- оперативного зберігання інформації;
- затримки інформації на певний час;
- перетворення послідовного коду подання інформації в паралельний та навпаки;
- зсуву коду, що зберігається, на один або декілька розрядів вліво або вправо.

Регістри також можуть використовуватися для побудови лічильників імпульсів та генераторів послідовностей імпульсів.

Головною класифікаційною ознакою регістрів є спосіб запису інформації до нього. Відповідно до цієї ознаки регістри можнарозподіліти на три групи:

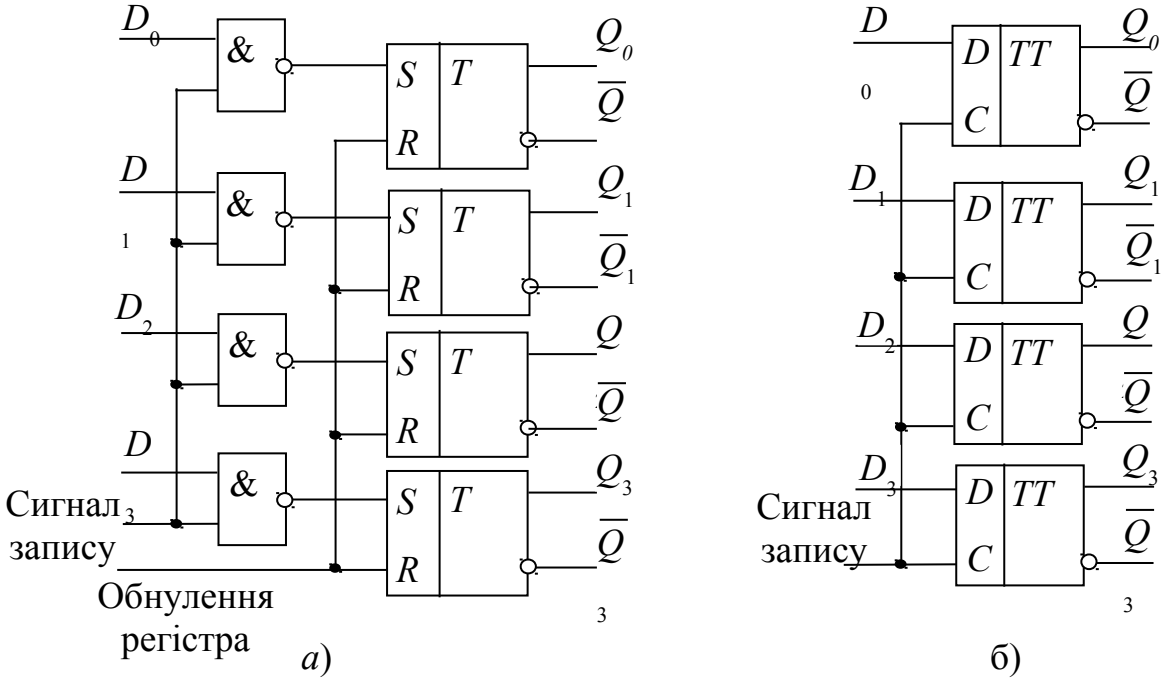
- паралельні (регістри зберігання);

– послідовні (регістри зсуву);
– послідовно-паралельні (універсальні регістри).

У паралельні регістри запис коду відбувається одночасно в усі розряди регістру. Паралельні регістри можуть бути побудовані з будь-яких типів тригерів.

У послідовні регістри запис інформації відбувається у результаті зсуву двійкової інформації вправо або вліво. Для побудови послідовних регістрів можна використовувати лише двоступеневі синхронні тригери.

Паралельний регістр – це пристрій, який призначено для введення, зберігання і виведення інфомації в паралельному коді. На рис. 1.43, *а* показано схему паралельного регістра, який побудований з асинхронних *RS*-тригерів, а на рис.1.43, б – паралельний регістр з синхронних *D*-тригерів. Кожен з цих регістрів призначений для роботи з чотирирозрядними даними й складається з 4-х тригерів (розрядів).



а) б)

Рисунок 1.43 – Схеми асинхронного (*а*) і синхронного (*b*) паралельних регістрів

Запис інформації в асинхронний регістр (рис.1.43, *а*) відбувається за два такти. У першому такті в усі розряди регістру записуються нулі (регістр обнулюється), а в другому відбувається запис одиниць в усі розряди, на які надійшли одиниці. Запис відбувається за появи сигналу на вході «Сигнал запису». Цей сигнал повинен бути сформованим тільки після встановлення інформації на входах даних.

Запис інформації у синхронний регістр, який зображено на рис. 1.43, б, відбувається за один такт сигналу «Сигнал запису». При цьому, у кожен розряд регістру буде записано значення інформаційного сигналу, який надходить по шині $D_0 \ldots D_3$.

Послідовний регістр. У такому регістрі запис інформації, яка надходить у послідовному коді, відбувається побітно. При цьому, інформація

просувається, відповідно до надходження сигналів синхронізації (запису інформації). У відповідності до того, в якому напряму відбувається зсув інформації, розрізнюють регістри зі зсувом вліво, якщо інформація переміщується від молодшого розряду до старшого (зсув старшим розрядом уперед), як показано на рис. 1.44. Зсув відбувається порозрядно у відповідності до надходження тактового сигналу.

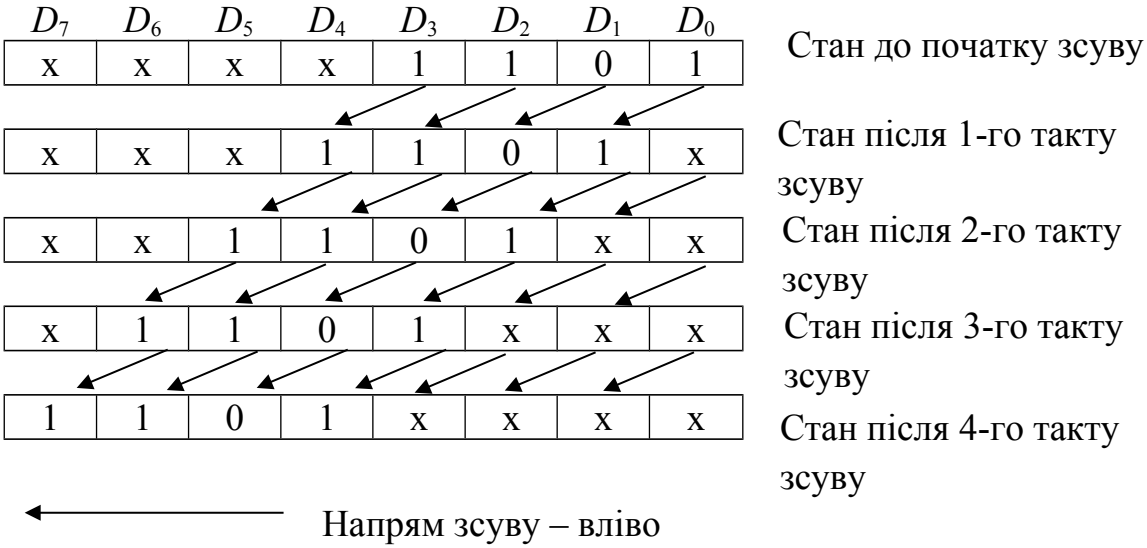


Рисунок 1.44 – Процес зсуву інформації вліво

Інформація, яка висувається з біта D_7, якщо не прийняти спеціальних заходів, безповорото втрачається. У біт D_0 може уводитися будь-який символ (0 або 1), що на рисунку показано символами x. З рисунку видно, що повне пересування інформації відбувається за стількі тактів, скількі бітів вміщує сама інформація (число).

Послідовні регістри будуються, звичайно, з тригерів таким чином, щоб інформаційні виходи і входи були з'єднані послідовно, а сигнал синхронізації (запису) надходив на всі розряди одночасно (паралельно). Схема чотирирозрядного послідовного регістру показана на рис. 1.45.

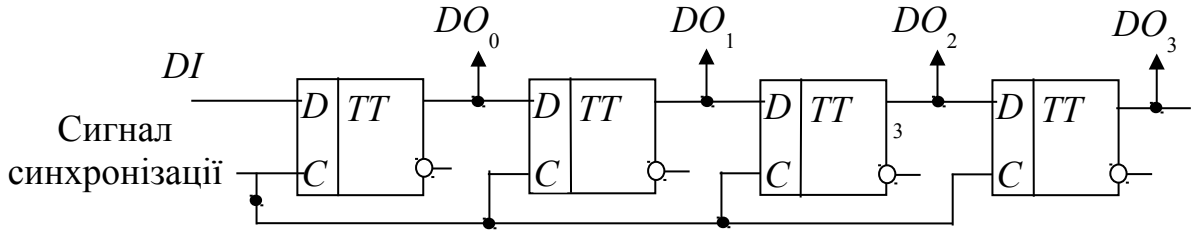


Рисунок 1.45 – Схема чотирьохрозрядного послідовного регістру

Напрям зсуву на схемі послідовного регістру визначається тим, яку нумерацію будуть мати виходи (*DO* – date output) окремих розрядів. На рис. 1.45 показано послідовний регістр зі зсувом вліво. Часові діаграми, які показують функціонування цього регістра, при надходженні на його

інформаційний вхід одного одиничного біту (число 1000) наведено на рис. 1.46. Вважаємо, що до початку запису в усіх розрядах регістру було записано 0.

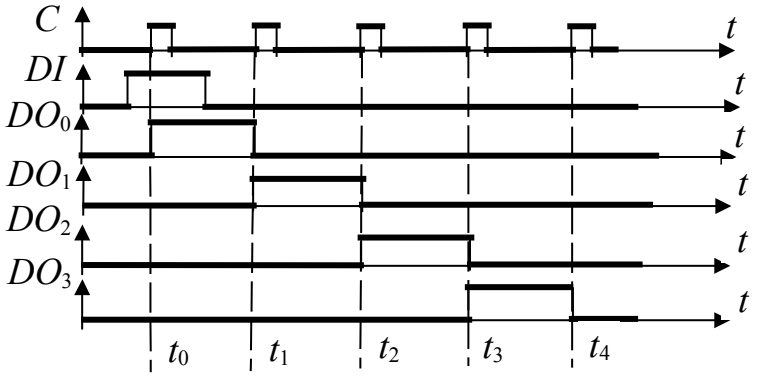


Рисунок 1.46 – Часові діаграми роботи послідовного регістру з зсувом вліво

На рис. 1.46 бачимо, що у момент t_0 відбувається запис одиничного значення у перший розряд регістру, а потім з приходом кожного наступного синхроімпульсу це значення зсувається до кожного наступного розряду. З рисунку видно, що інформацію на виходах усіх розрядів, після закінчення зсуву, можна зчитати у паралельному вигляді.

Таким же чином будується схема регістру зі зсувом вправо. Вхідний сигнал у такий регістр буде подаватися починаючи з молодшого розряду.

Універсальні регістри. Якщо доповнити послідовний регістр комбінаційними схемами, які забезпечуть можливість запису інформації в паралельному вигляді, то отримаємо регістр, що має назву послідовно-паралельного.

У цифровій техніці також використовуються регістри в яких можна змінювати напрям зсуву у процесі роботи. Такі регістри мають назву реверсивних. У них з'єднання розрядів між собою виконується за допомогою логічних схем, які здійснюють комутацію відповідно до вхідних сигналів керування. Умовне графічне позначення такого регістру показано на рис. 1.47.

На рис. 1.46 виводи мають наступні позначення:

SL – сигнал керування зсувом вліво, повинен мати активний рівень впродовж усього циклу зсуву інформації;

DL – вхід даних при зсуві вліво;

SR – сигнал керування зсувом вправо, повинен мати активний рівень впродовж всього циклу зсуву інформації;

DR – вхід даних при зсуві вправо;

C – вхід синхронізації;

DO_0 …DO_3 – виходи даних.

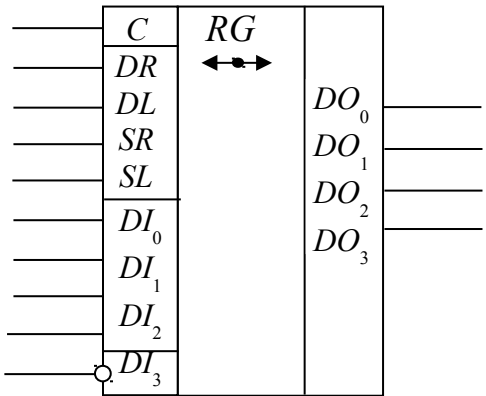


Рисунок 1.47 – Умовне графічне позначення універсального чотирирозрядного регістру

Промисловість випускає мікросхеми регістрів з різною кількістю розрядів і різними можливостями.

1.10 Лічильники імпульсів: підсумовуючий двійковий лічильник, віднімаючий двійковий лічильник, реверсивний двійковий лічильник

Лічильник імпульсів (ЛІ) – це цифровий пристрій, який здійснює лічбу і зберігання коду кількості підрахованих імпульсів. ЛІ – це цифровий автомат Мура, в якому його наступний стан визначається попереднім станом і логічною змінною на вході.

Кількість можливих станів ЛІ має назву – *коефіцієнт (модуль) лічби* і позначається літерою *К*. Цей коефіцієнт визначає кількість стійких станів у лічильнику і визначається

$$K = 2^m,$$

де m – кількість розрядів (тригерів), які входять до лічильника.

Класифікувати ЛІ можна за наступними характеристиками:

− **коефіцієнт (модуль) лічби.** Відповідно до цієї характеристики ЛІ діляться на: двійкові, двійково-десяткові (декадні), з довільним коефіцієнтом лічби.

Для двійкового ЛІ коефіцієнт (модуль) лічби дорівнює $K = 2^m$ і, відповідно, кількість стійких станів, а також тригерів у складі лічильника, завжди дорівнює цілому степеню числа 2. У цьому випадку число М, яке записано у ЛІ можна визначити, як

$$M = Q_m\, 2^m + Q_{m-1}\, 2^{m-1} + \ldots + Q_1\, 2^1 + Q_0\, 2^0 ,$$

де Q_m – номер виходу лічильника;

2^0 – вага першого (молодшого) розряду лічильника;

$2^1 \ldots 2^m$ – відповідно ваги наступних розрядів ЛІ.

Якщо необхідно побудувати ЛІ, у якого $K \neq 2^m$, то для цього можна уводити додаткові логічні зв’язки – зворотні та прямі, за допомогою яких двійковий лічильник перетворюється у недвійковий, у цілому, з довільним коефіцієнтом лічби. Найбільшого поширення серед таких ЛІ набули двійково-десяткові (декадні), у яких $K = 10$. Десяткова лічба виконується у двійково-десятковому коді (двійковому – за кодом лічби і десятковому – за кількістю станів);

− **напрям лічби.** Відповідно до цієї характеристики ЛІ діляться на: підсумовуючі, віднімаючі, реверсивні.

Ця характеристика визначає напрям зміни числа, яке записане у ЛІ, під час його роботи.

Так, у підсумовуючому ЛІ зміна відбувається, як додавання одиниці до поточного (інкрементування) значення стану.

У віднімаючих ЛІ, навпаки, зміна стану виконується як зменшення поточного значення на одиницю (декрементування).

У реверсивних ЛІ напрям зміни лічби може змінюватися безпосередньо під час роботи, відповідно до сигналів керування;

− **спосіб організації внутрішніх зв'язків.** Відповідно до цієї характеристики ЛІ діляться на: лічильники з послідовним перенесенням, з паралельним перенесенням, з комбінованим перенесенням, кільцеві. Ця характеристика визначає особливості організації перенесення результатів лічби між розрядами ЛІ.

При послідовному перенесенні сигнал перенесення надходять з попереднього розряду на вхід наступного. При побудові таких ЛІ використовуються *T*-тригери. Головним недоліком цих ЛІ є простота їх побудови. До недоліків можна віднести порівняно низьку швидкодію.

ЛІ з паралельним перенесенням будуються із синхронних тригерів. Імпульси, які надходять на вхід ЛІ, надходять одночасно на входи всіх розрядів і кожен з тригерів, які входять до схеми, для наступних, є джерелом сигналу. Спрацьовування усіх тригерів відбувається одночасно, відповідно до надходження сигналів синхронізації, кількість яких і рахує цей лічильник. За рахунок цього, швидкодія такого ЛІ є максимальною і визначається затримкою сигналу в одному тригері та затримкою у колах формування сигналів перенесення. Також ці ЛІ є більш завадостійкими.

У ЛІ з комбінованим перенесенням тригери об'єднуються в групи таким чином, що окремі групи будуються як лічильники з паралельним перенесенням, а між собою з'єднуються послідовно.

До основних експлуатаційних характеристик ЛІ відносяться:

− розподілювальна здатність. Це мінімальний час між двома сусідніми імпульсами, які надійно фіксуються ЛІ;

− максимальна швидкодія. Відповідає кількості імпульсів, які фіксуються ЛІ в одиницю часу, є зворотньою до розподілювальної здатності;

− інформаційна ємність. Максимальне число, яке може бути записано у ЛІ. Кількісно цей параметр дорівнює коефіцієнту (модуль) лічби.

Підсумовуючий двійковий лічильник. Послідовний підсумовуючий двійковий лічильник будується з певної кількості *T*-тригерів, які з'єднані послідовно. Використання *T*-тригерів визначає, що такий ЛІ буде асинхронним, незважаючи на те, що він є побудованим з синхронних тригерів.

Кількість тригерів визначається з виразу $K = 2^m$. Лічба в такому лічильнику починається з 0 і закінчується числом $m - 1$. Схема такого ЛІ, який має $K = 2^m = 16$, і відповідно складається з 4-х розрядів показана на рис. 1.48.

Більш доцільним, для використання у такому ЛІ, вважається *T*-тригер, який перетворено з *JK*-тригера. Перед початком лічби всі тригери ЛІ встановлюється в нульовий стан за допомогою сигналу скидання. При

надходженні вхідного сигналу перший тригер буде змінювати свій стан при надходженні кожного імпульсу і передавати цей сигнал у наступний. Перемикання відбувається при появі зрізу вхідного сигналу.

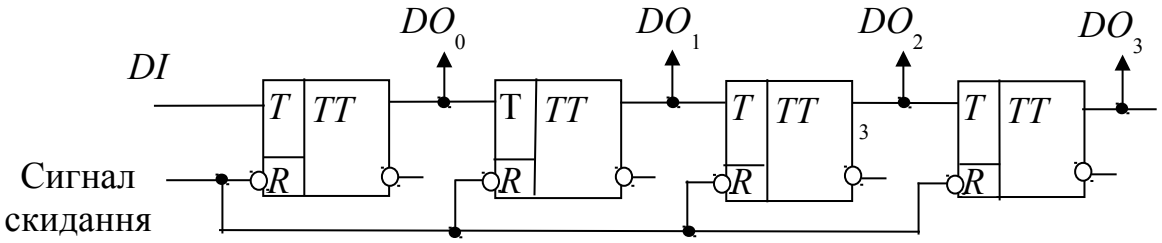


Рисунок 1.48 – Послідовний підсумовуючий двійковий лічильник з $K = 16$

Процес лічби у послідовному двійковому ЛІ показано за допомогою часових діаграм на рис. 1.49.

Тому що *T*-тригери ЛІ є перетвореними *JK*-тригерами, то вони спрацьовують під час зрізу вхідних сигналів, що видно на рис. 1.48. Після закінчення циклу, процес повторюється, тому імпульс, який надходить за 15 можна вважати 0. Відповідно до значення коефіцієнта лічби видно, що період сигналу на виході останнього розряду дорівнює 16 періодам вхідного сигналу *DI*.

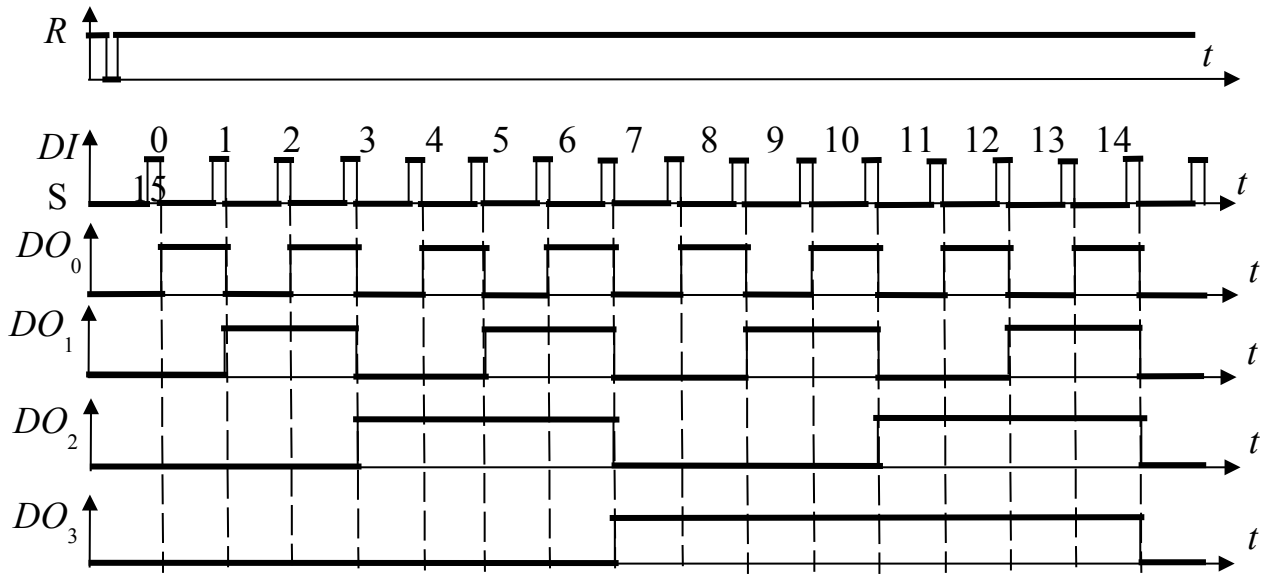


Рисунок 1.49 – Часові діаграми

Головним недоліком такого ЛІ є значна затримка розповсюдження сигналу у схемі, яка збільшується при нарощуванні кількості розрядів (коефіцієнта лічби). Це явище показано на рис. 1.50.

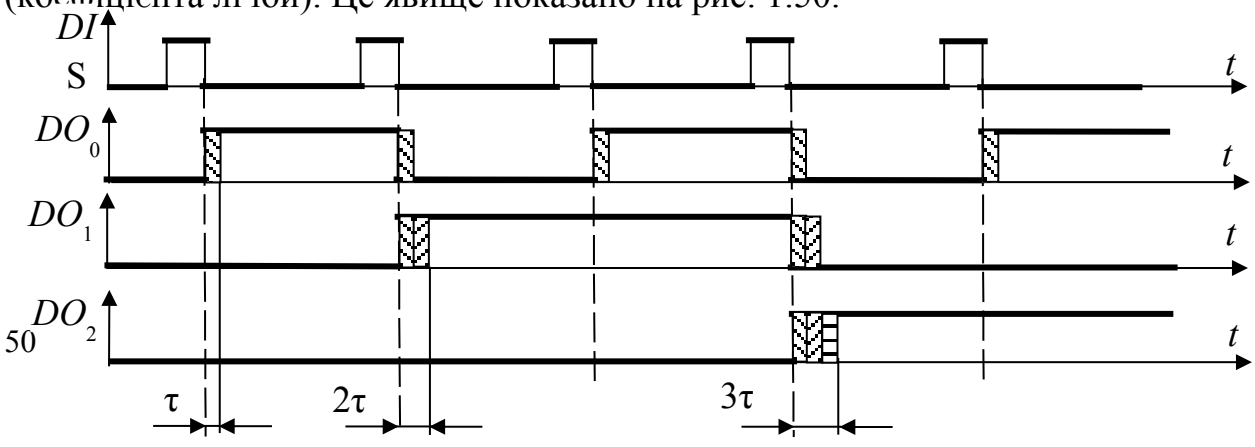


Рисунок 1.50 – Збільшення затримки розповсюдження сигналу у послідовному ЛІ

На рис. 1.50 показано як буде відбуватися затримка сигналу при перемиканні тригерів. При цьому вважаємо, що максимальна затримка сигналу в одному тригері дорівнює τ, і в кожному наступному вона збільшується на саме це значення. Тому вважається, що загальна затримка в такому ЛІ дорівнює

$$T_{\text{затр}} = n \times \tau ,$$

де n – кількість тригерів у ЛІ;

τ – затримка в одному тригері.

Для прискорення роботи необхідно, щоб перемикання окремих тригерів відбувалося одночасно, при надходженні вхідного сингалу. Такі ЛІ можуть будуватися як асинхронні, так і синхронні. Прикладом асинхронної схеми може бути ЛІ з наскрізним перенесенням.

ЛІ з наскрізним перенесенням можна побудувати також з використанням синхронних тригерів. Приклад побудови такого синхронного чотирирозрядного ЛІ з наскрізним перенесенням показано на рис. 1.51.

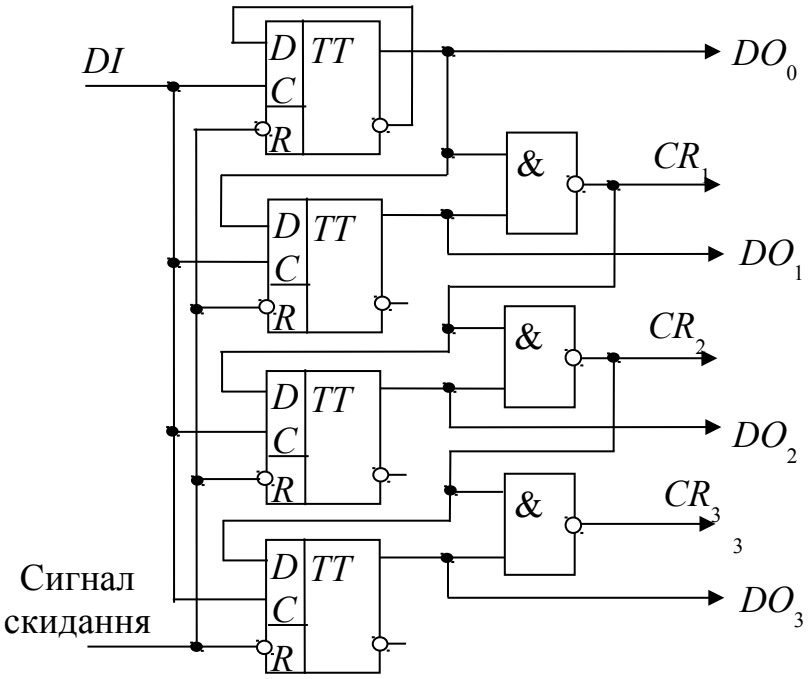


Рисунок 1.51 – Схема синхронного ЛІ з наскрізним перенесенням

Схему побудовано з використанням синхронних D-тригерів. Вхідний сигнал надходить одночасно на входи синхронізації C всіх тригерів, а на входи D надходять сигнали перенесення інформації CR, які сформовані в елементах ТА і дозволяють спрацьовування відповідних розрядів. Сигнал перенесення інформації з першого розряду CR_0 не сформовано тому, що тригер у цьому розряді працює у режимі лічби імпульсів. Часові діаграми для опису роботи цієї

схеми показано на рис. 1.52. Відповідно до часових діаграм можна стверджувати, що затримка сигалу в такій схемі дорівнює

$$T_{\text{затр}} = \tau_{\text{тр}} + (n - 2) \times \tau_{\text{ЛЕ}},$$

де n – кількість тригерів у ЛІ;

$\tau_{\text{тр}}$ – затримка в одному тригері;

$\tau_{\text{ЛЕ}}$ – затримка в одному логічному елементі ТА.

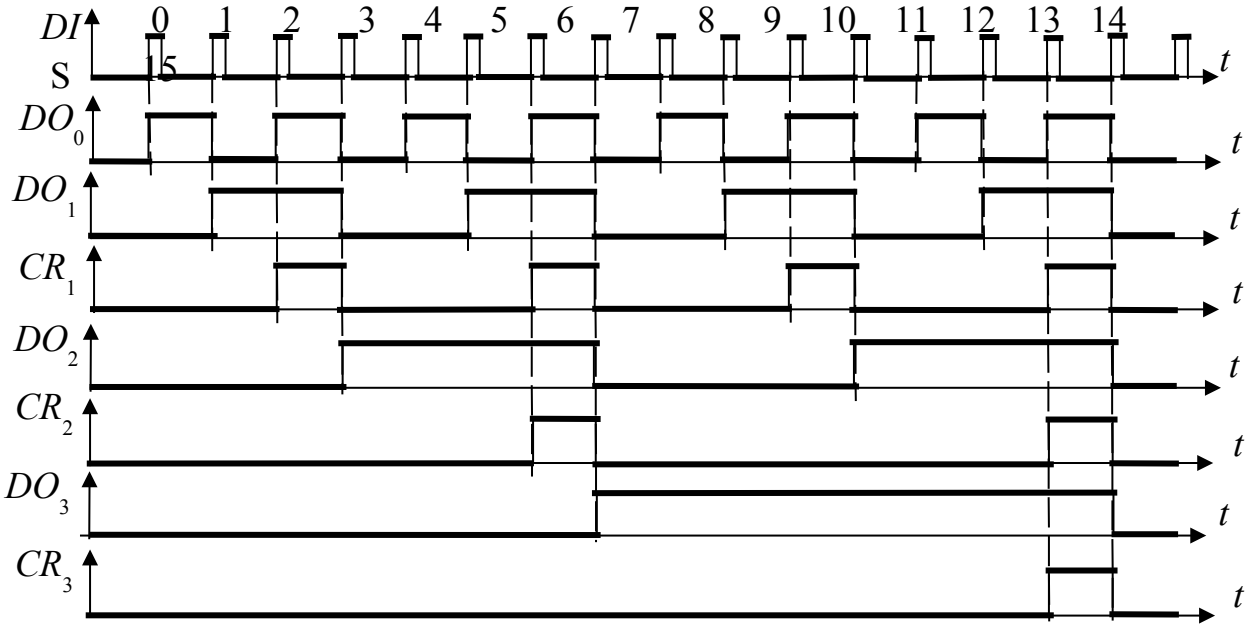


Рисунок 1.52 – Часові діаграми роботи синхронного ЛІ з наскрізним перенесенням

Для подальшого зменшення часу затримки використовуються ЛІ з паралельним (одночасним) перенесенням. У таких схемах створюються кола паралельного перенесення на елементах ТА. Схема такого ЛІ наведена на рис. 1.53.

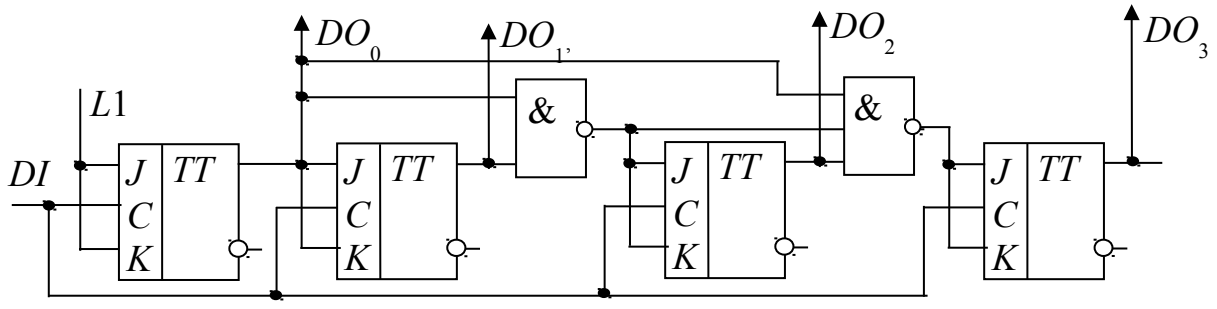


Рисунок 1.53 – Схема ЛІ з паралельним перенесенням

Використовуючи *JK*-тригери у яких декілька інформаційних входів *J* і *K*, можна побудувати таку схему без додаткових логічних елементів.

Віднімаючий двійковий лічильник. Віднімаючі двійкові лічильники будуються аналогічно підсумовуючим ЛІ з послідовним, паралельним і наскрізним перенесенням. Відміни полягають у тому, що пренесення з попереднього каскаду в наступний, повинно бути організовано таким чином,

щоб викликати переключення у протилежний стан. Крім того, перед початком лічби всі розряди ЛІ необхідно встановити в одиничний стан, в усі розряди записати 1, тоді з надходженням кожного наступного вхідного імпульсу буде відбуватися декремент відповідних розрядів.

Якщо виконувати побудову ЛІ з послідовним перенесенням, який побудовано на базі *JK*-тригерів, то схема буде мати вигляд, як показано на рис. 1.54.

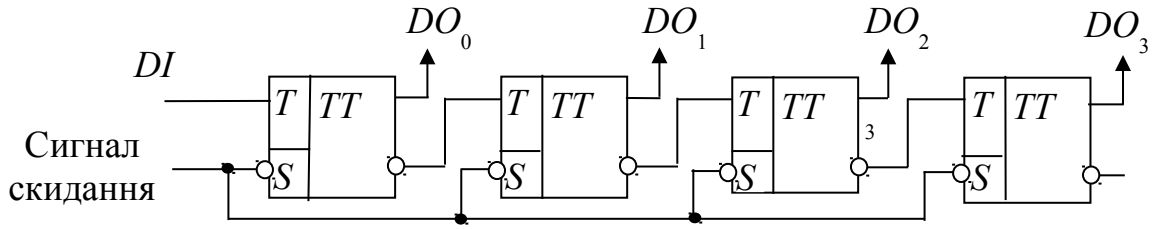


Рисунок 1.54 – Віднімаючий ЛІ з послідовним перенесенням

Реверсивний двійковий лічильник. Реверсивний двійковий лічильник відрізняється лише тим, що може змінювати напрям лічби відповідно до сигналу керування. В усіх інших умовах і вимогах до організації таких ЛІ вони співпадають з розглянутими вище.

Для зміни напряму лічби необхідно мультиплексування сигналів міжрозрядних перенесень. Можливий варіант організації такої схеми для ЛІ з послідовним перенесенням показано на рис. 1.55.

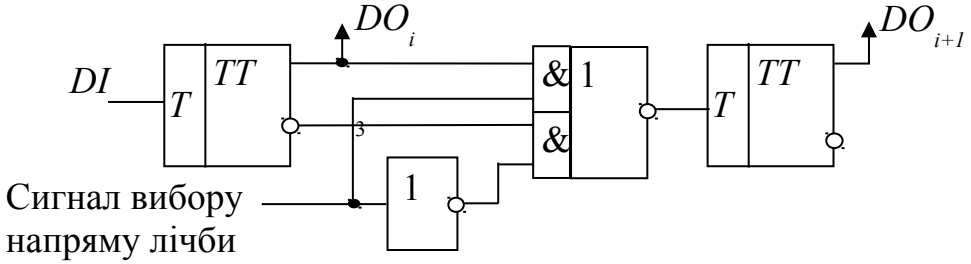


Рисунок 1.55 – Організація формування сигналів перенесення у реверсивному ЛІ з послідовним перенесенням

1.11 Лічильники з довільним коефіцієнтом лічби. Десятковий лічильник. Використання лічильників для поділу частоти

На практиці є необхідність використання ЛІ, які мають коефіцієнт лічби, що описується виразом $K \neq 2^m$ і не є кратним ступенем числа 2. Найбільш поширеним є спосіб виключення зайвих станів у двійковому лічильнику. Виключення зайвих станів може бути зреалізовано у різний спосіб. Вибір варіанту може базуватися на оцінці апаратних витрат для реалізації, наявності натурального порядку лічби тощо.

Найбільш розповсюдженими є два способи виключення зайвих станів:

− виключення перших станів з можливих і продовження лічби до повного заповнення ЛІ. Такі ЛІ мають назву «ЛІ з примусовим встановленням початкового стану»;

− виключення останніх станів; встановлення ЛІ в початковий (нульовий стан) після закінчення лічби відповідно до значення потрібного коефіцієнта лічби. Вони мають назву «ЛІ із встановленням кінцевого стану».

Припустимо, що необхідно побудувати ЛІ, який має коефіцієнт лічби, що дорівнює 10 (декадний лічильник). Вибираємо кількість тригерів, які повинні бути в складі цього ЛІ, відповідно до виразу $2^{m-1} \le K \le 2^m$. Для нашого випадку цей вираз буде мати вигляд:

$$2^3 \le 10 \le 2^4 .$$

З цього виразу видно, що схема повинна будуватися з 4-х тригерів, яка має коефіцієнт лічби, дорівнює 16. Можна визначити кількість зайвих станів:

$$16 - 10 = 6.$$

Таким чином, у схемі буде 6 зайвих станів, які необхідно виключити за допомогою зворотних зв'язків, які будуть примусово встановлювати окремі розряди ЛІ у відповідний стан.

Вважаємо, що будемо використовувати *T*-тригери. Для розробки ЛІ з примусовим встановленням початкового стану подамо число кількості зайвих станів у двійковій системі числення і за тим значенням визначимо адреси подавання сигналів зворотного зв'язку.

$$6_D = 0110_B .$$

Таким чином, видно, що до початку лічби ЛІ необхідно записати число 0110_B, для чого необхідно подати сигнал на входи R_0, S_1, S_2 і R_3. Встановлення цієї інформації також повинно відбуватися після закінчення повного циклу лічби (завантаження в ЛІ числа 15_D). Схема такого ЛІ наведена на рис. 1.56.

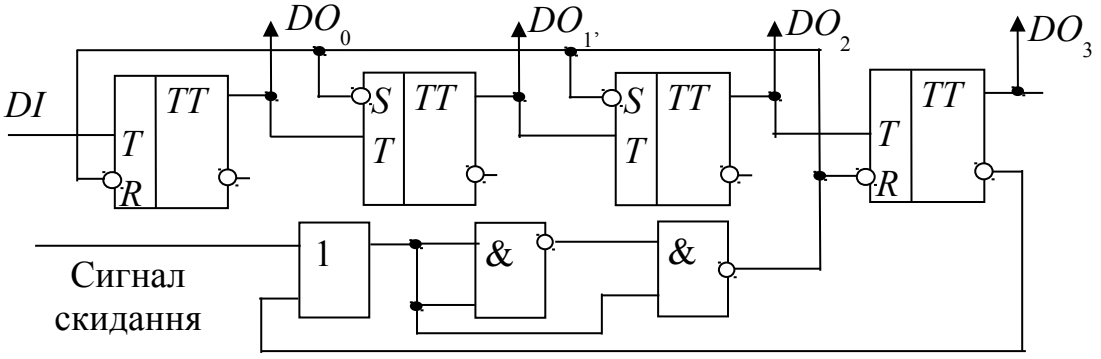


Рисунок 1.56 – Схема декадного ЛІ з примусовим встановленням початкового стану

Перемикання цієї схеми можна показати у вигляді табл. 1.7.

Таблиця 1.7 – Таблиця перемикань декадного ЛІ

№ імпульсу	DO_3	DO_2	DO_1	DO_0
Скидання	0	1	1	0
1	0	1	1	1
2	1	0	0	0
3	1	0	0	1
4	1	0	1	0

5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1
10	0	1	1	0

Робота цього ЛІ також може бути описана за допомогою часових діаграм, які показано на рис. 1.57.

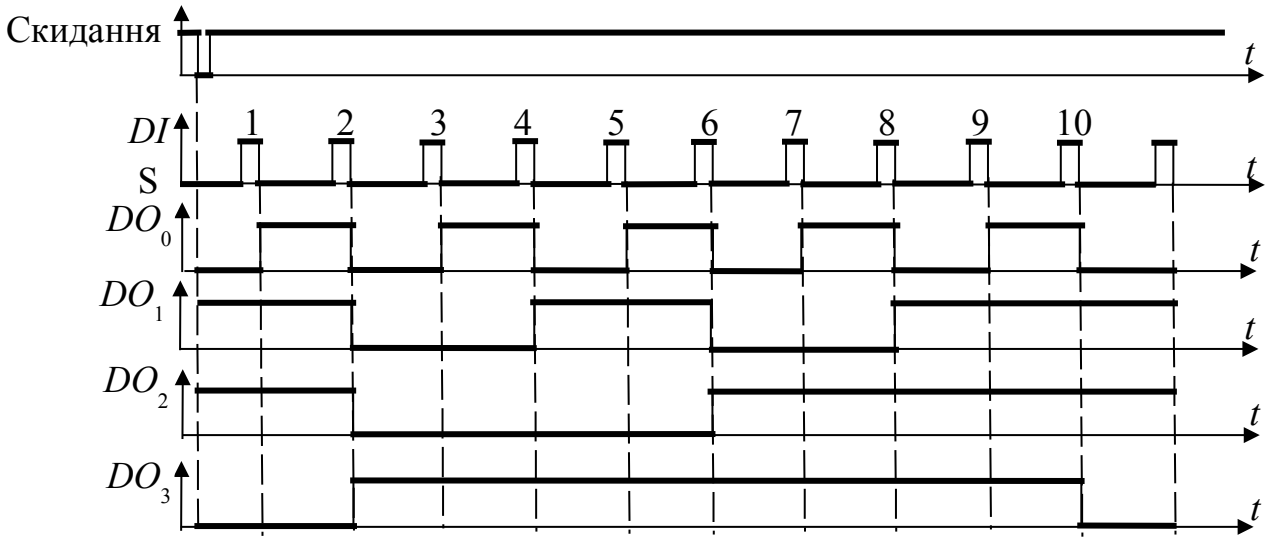


Рисунок 1. 57 – Часові діаграми роботи декадного ЛІ

ЛІ із встановленням кінцевого стану будується за тими самими передумовами, які розглянуті вище.

Розглянуті вище ЛІ відносяться до розряду лічильників із позиційним кодуванням. Крім того, для побудови ЛІ з $K \neq 2^m$ часто використовуються, так звані, кільцеві лічильники імпульсів. Ці лічильники будуються на основі зсувного регістру, в якому сигнал з виходу регістру подається на його ж вхід. Вони мають назву – *кільцеві лічильники імпульсів*.

Якщо в один із розрядів увести логічну одиницю, при тому що інші розряди встановлені в 0, то ця одиниця при надходженні кожного тактового імпульсу буде пересуватися від розряду до розряду у циклі, довжина якого буде дорівнювати кількості тригерів. При цьому, місцезнаходження цієї одиниці у лічильнику однозначно визначає кількість імпульсів, які надійшли до схеми. Це надає можливість використовувати такий лічильник без дешифраторів, що відповідно зменшує апаратні витрати і збільшує швидкодію системи. Недоліком таких ЛІ є те, що для таких схем коефіцієнт лічби $K = m$, тобто дорівнює кількості тригерів. Схема такого лічильника наведена на рис. 1.58.

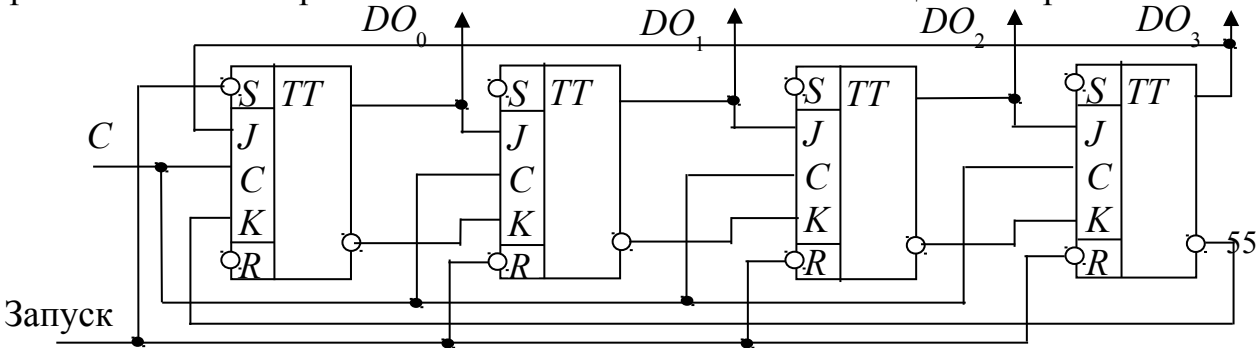


Рисунок 1.58 – Схема кільцевого лічильника імпульсів

Перед початком роботи зі входу «Запуск» у ЛІ записується 1, яка з приходом імпульсів синхронізації просувається по розрядах лічильника, і в кожний момент часу тільки на одному з виходів є одиничний сигнал, як показано на часових діаграмах, на рис. 1.59.

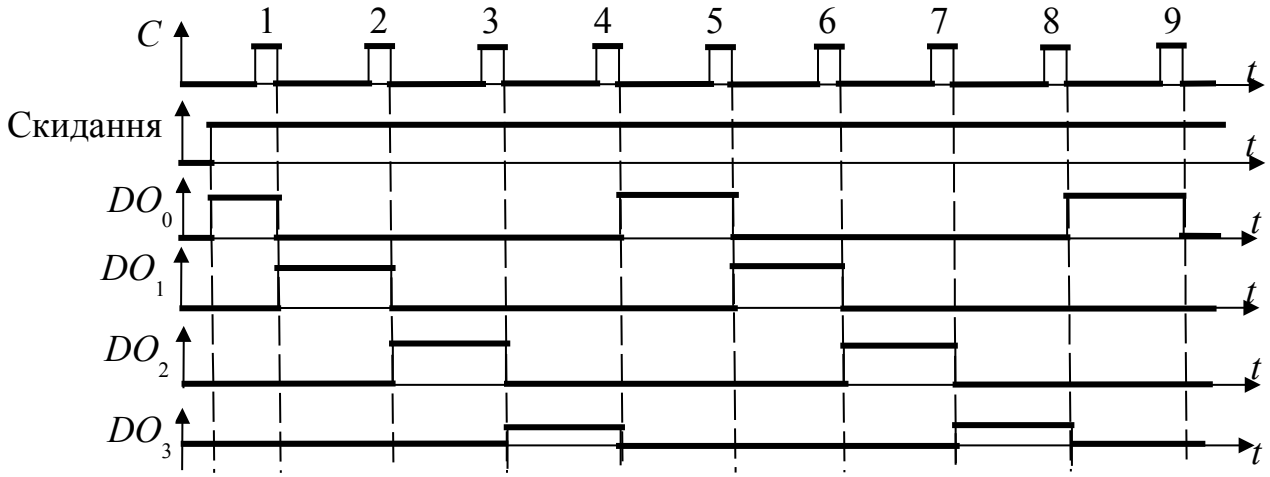
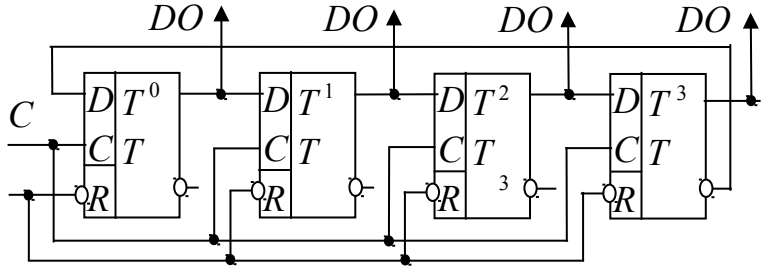


Рисунок 1.59 – Часові діаграми кільцевого ЛІ

Коефіцієнт лічби такого ЛІ можна збільшити до значення $K = 2m$, якщо один із зворотних зв'язків зробити перехресним, тобто вихід одного з тригерів з'єднати з інверсним входом попереднього. Такі пристрої мають назву – лічильник Джонсона. Схема такого лічильника і його таблиця перемикань показана на рис. 1.60.

Особливістю такого лічильника є те, що при надходженні вхідних імпульсів кожного разу перемикається лише один тригер, що зменшує ризик появи помилкових станів. Це також призводить до того, що у сусідніх рядках таблиці перемикань відмінності є лише в одному розряді вихідного сигналу, що спрощує організацію дешифратора стану ЛІ.



№ імпульсу	DO_3	DO_2	DO_1	DO_0
Скидання	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	0
7	1	0	0	0
8	0	0	0	0

Рисунок 1.60 – Схема лічильника Джонсона і його таблиця перемикань

Часові діаграми, які описують роботу схеми лічильника Джонсона наведені на рис. 1.61.

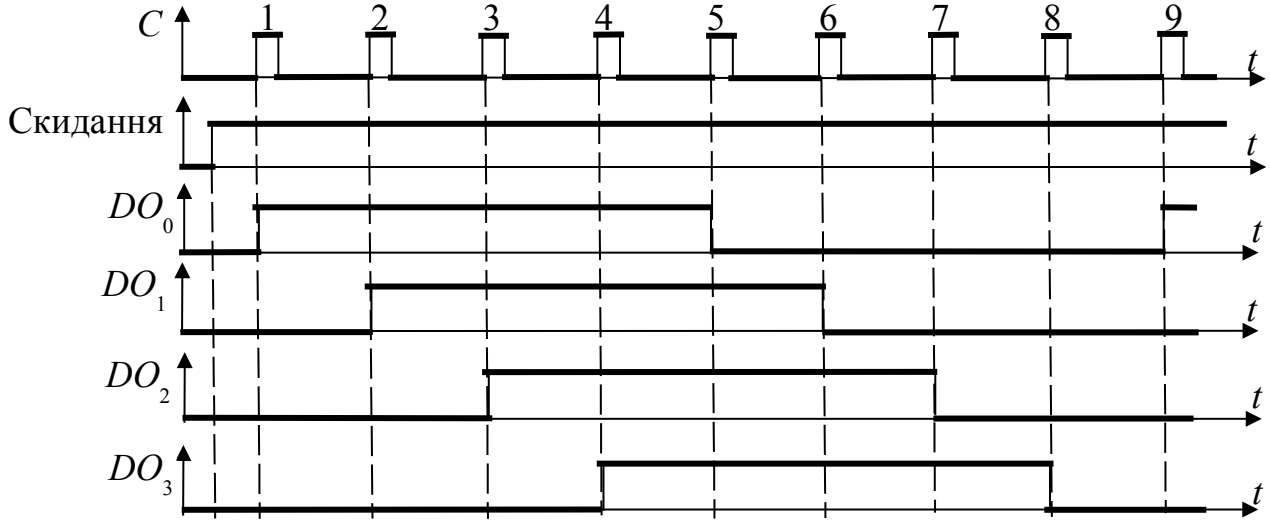


Рисунок 1.61 – Часові діаграми роботи лічильника Джонсона

1.12 Запам'ятовувальні пристрої (ЗП). Класифікація ЗП. Характеристики ЗП

Запам'ятовувальні пристрої (ЗП) призначені для зберігання двійкової інформації. Основними операціями у пам'яті є запис, зберігання і вибирання (читання) інформації. Сукупність операції запису і вибирання називається зверненням до пам'яті.

ЗП будуються з двопозиційних *елементів пам'яті* (ЕП)**,** кожен з яких зберігає один біт інформації. Сукупність декількох елементів пам'яті створюють *комірку пам'яті*, яка призначена для зберігання багаторозрядної двійкової інформації, і звернення до елементів якої відбувається одночасно. Звернення до ЗП відбувається за адресним принципом, який передбачає наявність у кожній комірці пам'яті відповідного номера, що називається адресою і який необхідно явно або неявно вказувати при зверненні. Крім адресних, використовують асоціативні ЗП, звернення до комірок яких відбувається за результатами аналізу деяких розрядів інформації, яка зберігається.

У залежності від природи фізичного середовища, у якому зберігається інформація, ЗП розділяються на напівпровідникові, пристрої із зарядовим зв'язком (ПЗЗ), магнітні, оптичні тощо. Далі будемо розглядати напівпровідникові ЗП.

Для порівняння запам’ятовувальних пристроїв між собою розроблена система параметрів, що характеризує їх властивості. До таких параметрів відносяться:

− **інформаційна ємність** – це кількість одиниць інформації, що одночасно може зберігатися у ЗП, і визначається у двійкових одиницях бітах або байтах. У разі використання одиниці вимірювання – байт, ємність пам’яті подають через число $K = 1024$ байт = 1 кбайт;

− **розрядність даних** – визначається вмістом (кількістю розрядів) комірки пам’яті;

− **організація пам’яті** – це параметр, який поєднує два попередніх у вигляді виразу

$$N \times n,$$

де N – кількість комірок пам’яті, які входять у ЗП;

n – розрядність даних.

Добуток, який буде отримано, при виконанні множення, буде дорівнювати інформаційній ємності у тій одиниці вимірювання, яка визначена розрядністю даних;

– час вибирання – інтервал часу з моменту надходження запиту на передавання даних до моменту їх появи на виході ЗП;

– тривалість циклу звернення (цикл пам’яті) – мінімальний інтервал часу між двома сусідніми зверненнями до ЗП, кожен з яких буде нормально виконуватись;

– напруга живлення ЗП – визначає вибір необхідного пристрою живлення;

– потужність енергоспоживання – електрична потужність, яку споживає ЗП при роботі. Для деяких типів ЗП розрізняється потужність енергоспоживання у режимі звернення (запис, читання) і у режимі зберігання. Потужність енергоспоживання у режимі звернення набагато більша за потужність у режимі зберігання;

– питома вартість – визначає співвідношення вартості йінформаційної ємності ЗП.

Вимоги до об’єму і швидкодії пам’яті є суперечними. Досягнення високої швидкодії є досить важкою технічною задачею. Для раціональної організації пам’яті передбачено класифікацію ЗП по швидкодії, яка окреслює певну ієрархію пристроїв пам’яті. Ієрархія ЗП обчислювальних систем показана на рис. 1.62. Вершину [illegible] ЗП певної МПС, які входять до складу ц[illegible]ивний ЗП (НОЗП) і внутрішня кеш-пам[illegible]

Надоператив[illegible]гістрів загального призначення ЦП, кі[illegible]роцесора, кількість розрядів у кожному з я[illegible] моделлю процесора. Швидкість роботи з ними є максимальною.

Рисунок 1.62 – Ієрархія пам’яті МПС

Внутрішня кеш-пам’ять – це різновид програмно недоступної оперативної пам’яті ємністю 1 … 16 кбайт, яка вбудована у ЦП. У залежності від типу процесора може бути одна кеш-пам’ять, спільна для даних і команд або дві окремих.

Зовнішня кеш-пам’ять – це також оперативна пам’ять, яка встановлюється на системній платі і призначена для прискорення процедури звернення до інших типів пам’яті, що входять до складу МПС. Ємність цієї пам’яті становить 64 кбайт – 1 Мбайт і постійно зростає у кожній наступній моделі комп’ютера. Швидкість роботи цієї пам’яті визначається швидкодією системної шини.

Оперативний запам’ятовувальний пристрій (ОЗП) – основний тип пам’яті МПС, який значною мірою визначає властивості МПС. Складається з двох видів пам’яті – *постійного запам’ятовувального пристрою* і, власне, *оперативного запам’ятовувального пристрою*. ОЗП будується за адресним принципом і має довільний доступ до кожної з комірок пам’яті.

Постійний запам’ятовувальний пристрій (ПЗП) – будується з мікросхем постійних (ПЗП) або перепрограмованих запам’ятовувальних пристроїв (ППЗП). Ці мікросхеми не допускають оперативної зміни інформації, що зберігається в них під час виконання програми і не втрачають її при вимиканні живлення. Інформація до них записується під час виготовлення або при першому програмуванні мікросхеми і надалі змінитися не може. Тому, під час роботи МПС інформація з них лише зчитується. У ПЗП зберігаються таблиці кодів команд, константи, стандартні підпрограми, наприклад, *BIOS* тощо.

Оперативний запам’ятовувальний пристрій (ОЗП) – пристрій пам’яті призначений для запису, зберігання і зчитування будь-якої інформації під час виконання програми. В залежності від типу використовуваних мікросхем, ОЗП буває двох видів: статичний і динамічний. Статичний ОЗП будується з мікросхем елементом пам’яті яких є транзисторний тригер. Такий елемент пам’яті може зберігати інформацію дуже довго, доки є живлення, незалежно від

кількості звернень для зчитування цієї інформації. ОЗП динамічного типу у якості елемента пам’яті використовує конденсатор, який є паразитною ємністю деяких схем, що побудовані з транзисторів зі структурою метал-діелектрик-напівпровідник (МДН). У результаті саморозряду такої ємності, інформація, що записана може пропадати, тому вони потребують періодичного її поновлення (регенерування).

До **пристроїв зовнішньої пам’яті** відносяться всі нагромаджувачі зі змінними і незмінними носіями: на твердих і гнучких магнітних дисках, лазерних компакт-дисках (CD-ROM) тощо. Обмін інформацією з такими пристроями відбувається з малою швидкістю, що пояснюється наявністю у їх складі електромеханічних вузлів. У цілому, ємність пристроїв зовнішньої пам’яті не обмежена.

Наявність того або іншого типу пам’яті визначається функційним призначенням МПС і умовами її роботи.

Окрім адресної пам’яті, до складу МПС можуть входити пристрої з іншим механізмом звернення до окремих комірок. До них відносяться асоціативна та стекова пам’ять.

Асоціативна пам’ять відрізняється тим, що звернення до комірок відбувається не за адресою, а за асоціативними ознаками самої інформації, що визначаються у результаті порівняння її з необхідними. До критеріїв визначення необхідної комірки можна віднести: рівність вмісту комірки наперед визначеному числу або будь-які інші.

Стекова пам’ять (стек) – це пам’ять з послідовним доступом, звернення до комірок якої відбувається за безадресним принципом, за алгоритмом *LIFO (Last Input First Output)* – останній увійшов – перший вийшов. У МПС стекова пам’ять широко використовується під час виклику підпрограм, оброблення переривань та при обробленні вкладених структур даних.

Стекову пам’ять реалізують за апаратним або апаратно-програмним способами.

Постійні запам’ятовувальні пристрої (ПЗП) можуть бути зреалізовані при застосуванні різних фізичних принципів і елементів. Взагалі, вони відрізняються кількістю можливих змін інформації, способом запису інформації і способом її зміни. Напівпровідникові ПЗП є енергонезалежними пристроями з довільним вибором інформації.

За кількістю можливих змін інформації ПЗП розрізняються на програмовані одноразово та багаторазово. До одноразово програмованих ПЗП інформацію записують один раз при виготовленні мікросхеми або перед першим включенням. У подальшому цю інформацію змінити неможливо. До таких мікросхем ПЗП відносяться: програмовані маскою ПЗП (ПЗПМ або *ROM– Read Only Memory*), інформація до яких записується шляхом створення на кристалі певного (замовного) фотошаблону і мікросхеми програмовані одноразово (ППЗП – програмовані ПЗП або *PROM*), у яких користувач має змогу перепалити плавкі перемички на кристалі для занесення певної інформації.

ПЗП характеризуються найбільшою простотою організації та керування. ЕП напівпровідникових ПЗП – це звичайно один елемент (біполярний або МОН транзистор). Простота і малі розміри запам'ятовувальних елементів веде до збільшення щільності зберігання інформації, що, в свою чергу, веде до збільшення інформаційної ємності і зменшення питомої вартості зберігання інформації.

До групи багаторазово програмованих ПЗП відносять мікросхеми, у яких інформацію, що зберігається, можливо стерти за допомогою ультрафіолетового випромінювання, яке надходить до кристалу через спеціальне віконце у корпусі (*EPROM*) і мікросхеми з електричним стиранням (*EEPROM*). До ПЗП з електричним стиранням також відносять **флеш-пам'ять.**

Програмовані маскою ПЗП. Основним елементом **програмованих маскою ПЗП** (ПЗПМ) є матриця нагромаджувача, яка складається з масиву ЕП, кожен з яких розміщено на перехресті рядка і стовпчика. Елементом пам'яті ПЗПМ є резистивна або напівпровідникова (діодна, транзисторна) перемичка між рядком і стовпчиком. Відповідно до розробленої маски, у процесі виготовлення мікросхеми перемички формують у тих точках матриці, де має бути записана логічна 1 і не формують, де має бути логічний 0.

Для керування читанням інформації з матриці нагромаджувача використовують дешифратори рядків і стовпців, на які інформація надходить з формувача адреси, який підключено до адресної шини МПС. Дешифратор рядків формує сигнал дозволу читання певного рядка, а дешифратор стовпців формує сигнал читання вмісту певних ЕП рядка. Сигнал з виходу дешифратора стовпців надходить на входи селектора, який з'єднує виходи вибраних ЕП з входами підсилювачів-формувачів вихідного сигналу. Спрощена структурна схема типового ПЗПМ показана на рис. 1.63. На цьому рисунку резистори, що з'єднують лінію вибраного рядка з лініями, яки підключені до селектора відповідають сигналам логічної 1, що записані у відповідні комірки. Нагромаджувач на цьому рисунку – це квадратна матриця $(N-1) \times (N-1)$ ЕП, з вмісту рядка якої можна сформувати вихідні дані потрібного розміру.

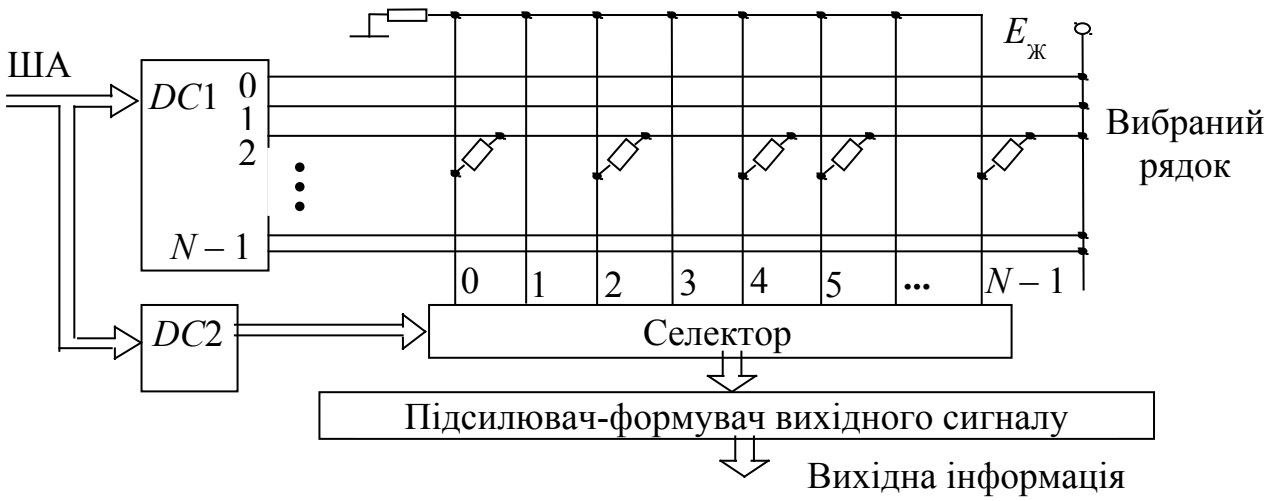


Рисунок 1.63 – Спрощена структурна схема ПЗПМ

Виходи усіх мікросхем ПЗПМ мають ТТЛ-рівні, і вони побудовані, переважно, за схемою з трьома стійкими станами.

Одноразово програмовані ПЗП (ППЗП або *PROM – Programmed Read Only Memory*). Мікросхеми цього типу за принципами побудови і принципами функціонування аналогічні ПЗПМ, але допускають програмування їх безпосередньо користувачем. Побудова мікросхеми ППЗУ, в цілому, аналогічно поданому на рис. 1.63. Відміни полягають у наявності пристроїв формування струму програмування, який подається на вихідні виводи мікросхеми. Операція програмування відбувається шляхом знищення (перепалювання) плавких перемичок на поверхні кристалу в місцях перетину рядків і стовпців матриці нагромаджувача, де потрібно записати логічний 0 або 1, тому ці мікросхеми програмуються лише один раз.

Програмування проводиться за допомогою спеціальних пристроїв – програматорів, які є досить простими приладами. Програмування проводиться подачею імпульсів електричного струму з амплітудою 30...50 мА, за відповідними адресами.

ППЗП використовуються для зберігання налагоджених програм для керування МПС різного призначення тощо.

Більшість мікросхем ППЗП побудована за ТТЛШ-технологією, але є невелика частина мікросхем, які побудовані за іншими технологіями – ЕСЛ, КМДН тощо.

ПЗП, багаторазово програмовані з ультрафіолетовим стиранням (репрограмовані ПЗП – РПЗП-УФ або *EPROM – Erasable Programmed Read Only Memory*). Ці мікросхеми дозволяють багаторазове їх перепрограмування самим користувачем. Ця властивість забезпечується використанням *n*-МОН транзисторів із застосуванням механізму лавинної інжекції заряду (ЛІЗМОН) з подвійним затвором. Ці транзистори відрізняються від звичайних МДН транзисторів наявністю двошарового підзатворного діелектрика, який дістав назву «плавучого затвору» (ПЗ). Шар діелектрика, що прилягає до каналу виготовлено з окису кремнію, товщиною менше ніж 5 нм. Другий шар зроблено з нітриду кремнію, товщиною близько 0,1 мкм. Електричний опір цього шару значно вище ніж опір шару окису.

Принцип роботи такого транзистора, який утворює ЕП, пов’язаний з нагромаджуванням заряду між шарами діелектриків і впливом цього заряду на значення порогової напруги транзистора.

У режимі програмування на керувальний затвор, джерело і стік подають імпульс напруги 21…25 В позитивної полярності. У зворотно зміщених *p-n* переходах виникає процес лавинного розмноження носіїв заряду й інжекції частини електронів у ПЗ. У результаті чого, на ПЗ нагромаджується негативний заряд, який зміщує передатну характеристику транзистора в область високої граничної напруги (праворуч), що відповідає запису 0 (рис. 1.64).

Для стирання інформації, перед перепрограмуванням, мікросхему розміщують під ультрафіолетове випромінювання дугової ртутної лампи. Під впливом цього випромінювання посилюється тепловий рух носіїв електричного заряду і електрони, що формували негативний заряд на ПЗ розсмоктуються у підкладку. Це зміщує передатну характеристику в область низької граничної напруги (ліворуч), що відповідає запису 1 (рис. 1.64).

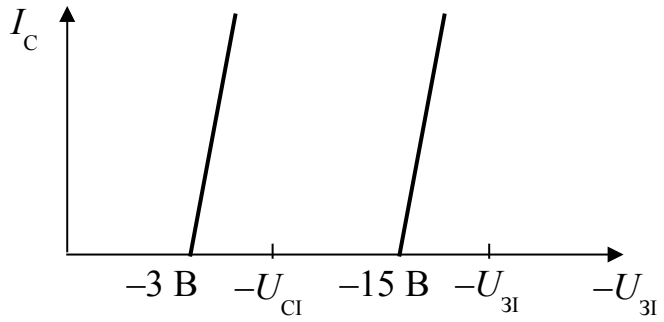


Рисунок 1.64 – Передатна характеристика *n*-МОН транзистора

Ці мікросхеми мають досить хороші властивості: порівняно високу швидкодію, велику кількість варіантів щодо організації пам'яті, невисоку вартість.

До недоліків цих мікросхем можна віднести: малу кількість циклів перепрограмування (від 10 до 100), що пояснюється швидким старінням діелектрика під впливом ультрафіолетового випромінювання, потребою у спеціальному обладнанні для стирання інформації, значний час стирання, досить високу чутливість до висвітлення та можливість випадкового стирання інформації.

ПЗП, багаторазово програмовані з електричним стиранням (репрограмовані ПЗП – РПЗП-ЕС або *EEPROM – Electrical Erasable Programmed Read Only Memory*). Ці мікросхеми за принципами побудови аналогічні РПЗП-УФ, але стирання інформації не потребує використання ультрафіолетового випромінювання. ЕП такої мікросхеми – це МОН-транзистор з індукованим каналом *p*-типу або *n*-типу, що має двошаровий діелектрик під затвором. Верхній шар формують з нітриду кремнію, нижній з – оксиду кремнію. Якщо до затвору відносно підкладки прикласти імпульс напруги позитивної полярності з амплітудою 30…40 В, то під дією сильного електричного поля електрони проходять через шар оксиду кремнію і нагромаджують заряд між шарами діелектриків. Цей заряд знижує граничну напругу і зміщує передатну характеристику транзистора ліворуч, що відповідає запису в ЕП логічної 1 (рис. 1.64). Для запису логічного 0, необхідно знищити заряд між шарами діелектриків, що нагромаджується при подачі на затвор імпульсу негативної полярності з амплітудою 30…40В. При цьому заряд електронів витісняється в підкладку. Відсутність заряду у шарі діелектрика зміщує передатну характеристику в область високих граничних напруг.

Режими стирання і програмування забезпечуються напругами однієї полярності: негативної для *p*-МНОН структур і позитивної для *n*-МНОН структур.

Принципи побудови цих мікросхем аналогічні описаним вище. Крім вузлів, що забезпечують роботу мікросхеми в якості ПЗП, до їх складу входять пристрої, що забезпечують її роботу у режимах стирання і програмування – комутатори режимів і формувачі імпульсів необхідної амплітуди і тривалості з напруги програмування U_{PR} , що подається на відповідний вхід мікросхеми.

До переваг мікросхем РПЗП-ЕС можна віднести: значну кількість циклів перепрограмування і можливість її перепрограмування безпосередньо у складі певного пристрою, що розширює функційні можливості таких мікросхем.

Флеш-пам'ять (*Flash*-пам'ять). Мікросхеми пам'яті такого типу були розроблені фірмою *Intel* у 1988 році.

У якості ЕП флеш-пам'яті використовується МОН-транзистор з ПЗ, який виготовлено за спеціальною технологією, яка називається *ETOX* (*EPROM Thin Oxide*) і запатентована фірмою *Intel*. В цілому, структура МОН-транзистора з ПЗ подібна описаним вище. Відмінністю, яка забезпечується технологією *ETOX* є зменшення товщини шару оксиду кремнію більш ніж втричі, що дозволило зменшити напругу програмування до 12 В і зменшити напругу стирання за рахунок тунельного ефекту, також до 12 В. Ці заходи дозволяють виконувати перепрограмування флеш-пам'яті безпосередньо у складі МПС і забезпечують можливість збільшення кількості циклів запису інформації.

Для забезпечення правильної організації роботи флеш-пам'яті фірмою *Intel* розроблена низка заходів, що дозволяють уникнути виходу її з ладу під час програмування. До них можна віднести:

− застосування спеціальних алгоритмів запису і стирання з контролем стану і завершенням процесу за результатами контролю;

− попереднє програмування в режимі стирання, коли перед стиранням усі ЕП матриці встановлюються в стан 0;

− включення до складу мікросхеми регістру, який зберігає ідентифікатори фірми виготовлювача і типу мікросхеми, що дозволяє захистити елемент від помилок вибору алгоритму;

− вбудування в мікросхему кіл, що реалізують алгоритм стирання і запису. Це спрощує зовнішнє керування і захищає від помилок під час перезапису.

Існує три групи мікросхем флеш-пам'яті:

− мікросхеми першого покоління, які виготовлені у вигляді єдиного масиву (блока), інформація в якому стирається цілком (*BULK-ERASE*);

− мікросхеми, масив пам'яті яких поділено на блоки різного розміру, що мають різні рівні захисту від випадкового звернення до них (*BOOT-BLOCK*);

− мікросхеми третього покоління, які мають найбільший розмір масиву, що розділено на блоки однакового розміру з незалежним стиранням (*FLASH-FILE*).

Мікросхеми різних груп мають відмінності у їх використанні. Так мікросхеми *BULK-ERASE* можуть використовуватись замість традиційних

мікросхем *EPROM*, з можливістю перепрограмування безпосередньо у складі обладнання під керівництвом процесора самої системи. Мікросхеми *BOOT-BLOCK* застосовуються для зберігання *BIOS* у персональних комп'ютерах, що дає змогу оновлення системи безпосередньо із зовнішніх носіїв інформації. Мікросхеми *FLASH-FILE* використовуються для зберігання даних значного обсягу в *Flash*-картах, які є альтернативою жорстким магнітним дискам. Очікується, що *Flash*-картки зможуть замінити жорсткі магнітні диски, особливо у системах, що працюють в умовах сильних механічних впливів.

Швидкодія флеш-пам'яті у 125…250 разів перевищує цей параметр для жорсткого диску, але поступається йому щодо інформаційної ємності, яка не перевищує 40 Мбайт.

Напруга живлення мікросхем флеш-пам'яті становить – 5 В, а стирання і програмування -12 В. Споживаний струм істотно залежить від режиму роботи мікросхеми. Так в режимі очікування (*Standby*) споживаний струм значно менший за струм, який споживається у режимі стирання і запису, переважно у колі джерела 12 В.

Більшість мікросхем флеш-пам'яті працюють з даними у вигляді послідовного коду з використанням шини I^2C (*Inter Integrated Circuit Bus*). Ця шина складається з двох двоспрямованих ліній: *SCL* (*Serial Clock*) і *SDA* (*Serial Date*), до яких можна підключати до128 пристроїв. Один з пристроїв є ведучим (*master*), інші – веденими (*slave*). Ведучий пристрій генерує імпульси синхронізації *SCL* і керує всією роботою шини. Ведені працюють під керуванням ведучого, обслуговуючи його запити.

Оперативні запам'ятовувальні пристрої

ОЗП статичного типу (*Static Random Access Memory – SRAM*) призначений для оперативного запису, зберігання і зчитування інформації під час виконання МПС будь-яких програм. У ОЗП статичного типу інформація зберігається у тому місці (комірці пам'яті або запам'ятовувальному елементі), де вона була записана і не стирається під час її зчитування.

Структура ВІС ОЗП схожа на структуру мікросхем ПЗП з тією різницею, що в якості ЕП використовується транзисторний тригер. Елементною базою для побудови тригерів можуть бути як біполярні, так і МОН-транзистори. Тому що, для функціонування тригера потрібне живлення, тоді пам'ять такого типу є енергетично залежною (*volatile memory*). При відмиканні живлення інформація, що зберігалася втрачається. Запис інформації у тригер відбувається шляхом встановлення в один з двох його можливих станів. Для зміни стану необхідно подати на входи тригера необхідні сигнали запису.

Типова структура ОЗП статичного типу включає: матрицю нагромаджувача і схеми запису/зчитування інформації, схеми дешифрування адреси ЕП або комірки пам'яті, схеми керування режимом тощо, які інтегровані на одному кристалі. В залежності від побудови нагромаджувача розрізняють ОЗП з однорозрядною та багаторозрядною організацією пам'яті. Основна відміна у структурних схемах полягає у тому, що нагромаджувач ОЗП з багаторозрядною організацією пам'яті складається з кількох шарів однакових

матриць, і одну комірку пам’яті складають елементи з однаковими адресами у всіх матрицях.

Організація пам’яті однорозрядного ОЗП становить $2^m \times 1$ біт, де m – кількість розрядів шини адреси, що можуть бути підключені до цієї ВІС, а для багаторозрядного ОЗП становить $2^m \times n$ біт, де n – кількість розрядів шини даних. Багаторозрядні *SRAM*, переважно, мають байтову організацію ($2^m \times 8$).

Спрощену структурну схему багаторозрядного ОЗП з організацією $2^m \times 4$ показано на рис. 1.65.

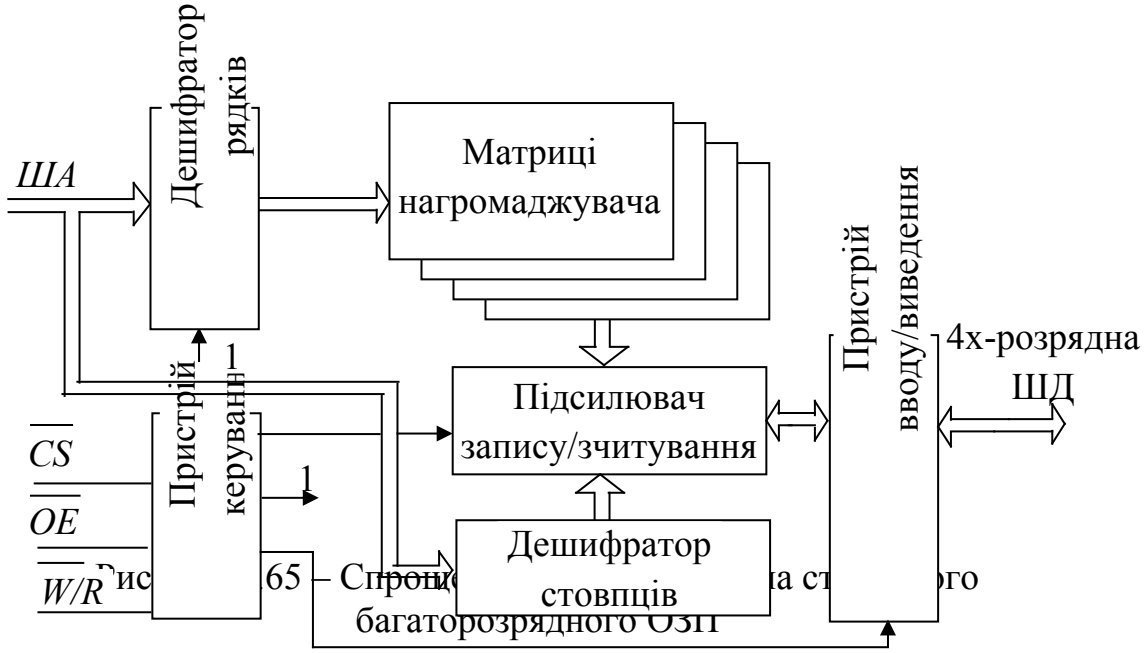


Рис. 1.65 – Спрощена структурна схема статичного багаторозрядного ОЗП

На цьому рисунку звернення проводиться одночасно до чотирьох матриць нагромаджувача по одному ЕП в кожній. Розряди шини адреси розподіляються на дешифратор рядків і стовпчиків для вибору відповідних рядків і стовпчиків одночасно у 4-х матрицях.

На пристрій керування надходять такі сигнали:

− $\overline{CS}$ (*Chip Select*) – сигнал вибору мікросхеми. Сигнал логічного 0 на цьому виводі дозволяє роботу вибраної мікросхеми. Відсутність цього сигналу переводить мікросхему у неактивний стан;

− $\overline{OE}$ (*Output Enable*) – сигнал дозволу виходу. Сигнал логічного 0 (активний рівень для цього входу) дозволяє роботу виходу. Сигнал логічної 1 визначає перехід мікросхеми у *z*-стан;

− $\overline{W/R}$ ($\overline{Write}$ / *Read*) – $\overline{\text{запис}}$ / зчитування. Цей сигнал керує режимом роботи мікросхеми, забезпечуючи виконання необхідних функцій мікросхеми.

Для виконання операцій запису/зчитування необхідна одночасна наявність рівнів логічного 0 на виводах *CS* і *OE*. Відсутність будь-якого з них переведе мікросхему у режим зберігання інформації.

Запис інформації, яка надходить з чотирирозрядної шини даних, виконується сигналом логічного 0 на вході *W/R*, при активних рівнях сигналів *CS* і *OE*. Запис проводиться у комірку пам'яті, адреса якої встановлена на шині адреси.

Для зчитування вмісту комірки пам'яті необхідно подати активні рівні сигналів *CS* і *OE*, на шині адреси встановити адресу необхідної комірки, на вхід *W/R* подати сигнал з рівнем логічної 1. Зчитування відбувається на чотирирозрядну шину даних.

Будь-які інші комбінації сигналів на входах керування переводять ВІС у режим зберігання інформації.

ОЗП динамічного типу (*Dynamic Random Access Memory – DRAM*) також призначений для оперативного запису, зберігання і зчитування інформації під час виконання МПС будь-яких програм. Модулі ОЗП сучасних МПС, як правило, будуються на базі мікросхем такого типу.

В якості ЕП ВІС *DRAM* використовується ємність *p-n*-переходу МДН-транзистора, стан заряду якої відповідає інформації, що зберігається у цій комірці. Вважають, що заряджений конденсатор зберігає інформацію логічної 1, а розряджений – логічного 0. Для тривалого зберігання інформації виконується порядкова регенерація (*refresh*) всього вмісту *DRAM* з інтервалом 2 або 4 мс. Поновлення інформації відбувається також під час запису і зчитування інформації, а також під час спеціального циклу регенерації. Порівняно з ВІС *SRAM* мікросхеми ОЗП динамічного типу мають більшу інформаційну ємність. Останнім часом випускаються ВІС з організацією пам'яті 1М × 1, 4М × 1, 16М × 1, 64М × 1. До недоліків мікросхем *DRAM* можливо віднести лише меншу швидкодію.

Для забезпечення збільшення інформаційної ємності, мікросхеми *DRAM* повинні мати адресну шину з більшою кількістю розрядів, що викликає певні труднощі, тому всі ВІС цього типу мають мультиплексовану адресну шину. Звернення до ЕП відбувається за два етапи формування її адреси – окремо для рядка і окремо для стовпчика, що забезпечується наявністю двох спеціальних входів: *CAS* (*Column Address Strobe*) – строб адреси стовпця і *RAS* (*Row Address Strobe*) – строб адреси рядка.

Після закінчення процесу запису стан внутрішніх кіл ВІС необхідно відновити, подавши на вхід *RAS* високий рівень сигналу. Тривалість дії цього сигналу дорівнює інтервалу між сусідніми сигналами *RAS*.

Зчитування інформації проводиться на вихідну лінію *DO* (*Date Output*).

Процес регенерації автоматично виконується для всіх ЕП рядка до якого відбувається звернення для запису або зчитування.

Цикл регенерації складається з послідовного перебору адрес всіх рядків і звернення до них. Формування адрес відбувається за допомогою зовнішнього лічильника циклів звернень. Звернення до матриці можна організувати у кожному з можливих режимів функціонування: запису, зчитування, зчитування/модифікації/запису, а також у спеціальному режимі регенерації – сигналом *RAS* (за наявності сигналу *CAS,* що має неактивний рівень). Такий вид регенерації називається прихованою регенерацією (*hidden refresh*), «прозорою» (*transparent refresh*) або захопленням циклу (*cycle stealing*).

Зараз стандартними для більшості мікропроцесорних систем є модулі модифікації *DIMM DDR* (*Dual Inline Memory Modules Double Data Rate* – модулі пам’яті з дворядним розміщуванням виводів і подвійним стандартом даних), які мають ємність від 64М до 1Гбайта. Існує декілька варіантів реалізації оперативної пам'яті стандарту DIMM DDR, відмінних пропускною спроможністю, яка визначається кількістю біт за секунду, що приймаються і передаються оперативною пам’яттю в процесі її функціонування. Нині випускаються модулі пам’яті DIMM DDR стандартів PC1600, PC2100, PC2700 і PC3200 (пропускна здатність 1600, 2100, 2700 і 3200 Мбайт/с відповідно).

Ще більш сучасними є модулі пам’яті *RIMM* корпорації *RAMBUS*, які мають більшу пропускну здатність ніж модулі *DIMM*. Пропускна здатність модуля *RIMM* на частоті 400/800 МГц становить 1,6/3,2 Гбайта/с.

Умовне позначення мікросхем пам’яті складається з трьох полів: основного – де знаходиться інформація про функційне призначення мікросхеми й двох доповнювальних, в яких розміщується інформація про функційне призначення кожного виводу й тип сигналів на входах даних. Розміри кожного з полів встановлюються відповідно до ДСТУ на зображення інтегральних мікросхем.

У залежності від функційного призначення, в основному полі, можливі такі позначення:

− *ROM* – загальне позначення постійного запам’ятовувального пристрою, без уточнення до якого типу відноситься. Якщо необхідно точно вказати використовуваний тип пристрою, то загальне позначення може доповнюватись;

− *PROM* – постійний запам’ятовувальний пристрій програмований маскою;

− *EPROM* – репрограмований ПЗП з електричним записом і ультрафіолетовим стиранням;

− *EEPROM* – репрограмований ПЗП з електричним записом і електричним стиранням;

− *RAM* – умовне позначення оперативного запам'ятовувального пристрою статичного типу;

− *RAMD* – умовне позначення оперативного запам'ятовувального пристрою динамічного типу;

− *CAM* – умовне позначення асоціативного запам'ятовувального пристрою.

У лівому доповнювальному полі розміщується інформація про функційне призначення входів, а у правому – виходів мікросхеми. Для різних типів ЗП виводи мають різне функційне призначення, але в цілому, є певний набір, з якого можливо вибирати необхідні:

A_0 –A_N – адресні входи, номер розряду адреси показано у вигляді індексу;

DI_0 – DI_M – входи даних. Для багаторозрядних ОЗП номер розряду вхідного сигналу подається у вигляді індексу;

DO_0 – DO_M – виходи даних. Для багаторозрядних ЗП номер розряду вхідного сигналу подається у вигляді індексу;

DIO_0 – DIO_M – входи/виходи даних. Виводи, функції яких об'єднано і вибір необхідної визначається сигналами керування;

C (*CLK*) – вхід синхронізації. Призначено для приймання сигналу синхронізації від генератора тактових імпульсів;

CAS – строб адреси стовпця;

RAS – строб адреси рядка;

EO – сигнал дозволу виходу;

ER (*Enable Read*) – сигнал дозволу стирання (обнулення вмісту ОЗП);

RD (*Read*) – сигнал дозволу зчитування;

WR (*Write*) – сигнал дозволу запису;

W/R – сигнал керування процесом запису/читання;

REF (*refresh*) – сигнал зовнішньої регенерації;

CS – сигнал вибору мікросхеми;

XACK – вхід сигналу кінця циклу запису/читання. Вказує на закінчення циклу взаємодії динамічної пам'яті з центральним процесором. Формується контролером динамічної пам'яті;

SACK – вхід сигналу початку циклу запису/зчитування. Формується контролером динамічної пам'яті;

U_{PK} – вхід напруги програмування;

U_{CC} – вивід для подання напруги живлення;

GND – загальна лінія (цифрова «земля»).

Виводи U_{PK}, U_{CC} і *GND* на принципових схемах дозволяється не показувати.

У залежності від режиму роботи й можливостей формувати різні сигнали, виходи мікросхеми можуть перебувати у трьох станах – стан формування логічного 0, стан формування логічної 1 і у високоімпедансному стані (*z*-стані). Високоімпедансний стан відповідає стану виходу, який відключено від лінії. Крім того, в залежності від схемотехнічних особливостей вихідних каскадів мікросхем, виходи можуть бути: по-перше – з відкритим колектором (відкритим стоком) – це вихідний каскад на біполярному або МОН-транзисторі

у якого немає резистора у колі колектора або у колі стоку; по-друге – з відкритим емітером – це каскад на біполярному транзисторі у якого відсутній резистор у колі емітера. Для забезпечення правильної роботи таких схем необхідно передбачити підключення резистора необхідної величини. Наявність у мікросхеми таких особливостей позначають спеціальними знаками:

–  вихід з трьома станами;

–  вихід з відкритим колектором;

–  вихід з відкритим емітером.

Приклади зображення мікросхем ПЗП різних типів подано на рис. 1.66.

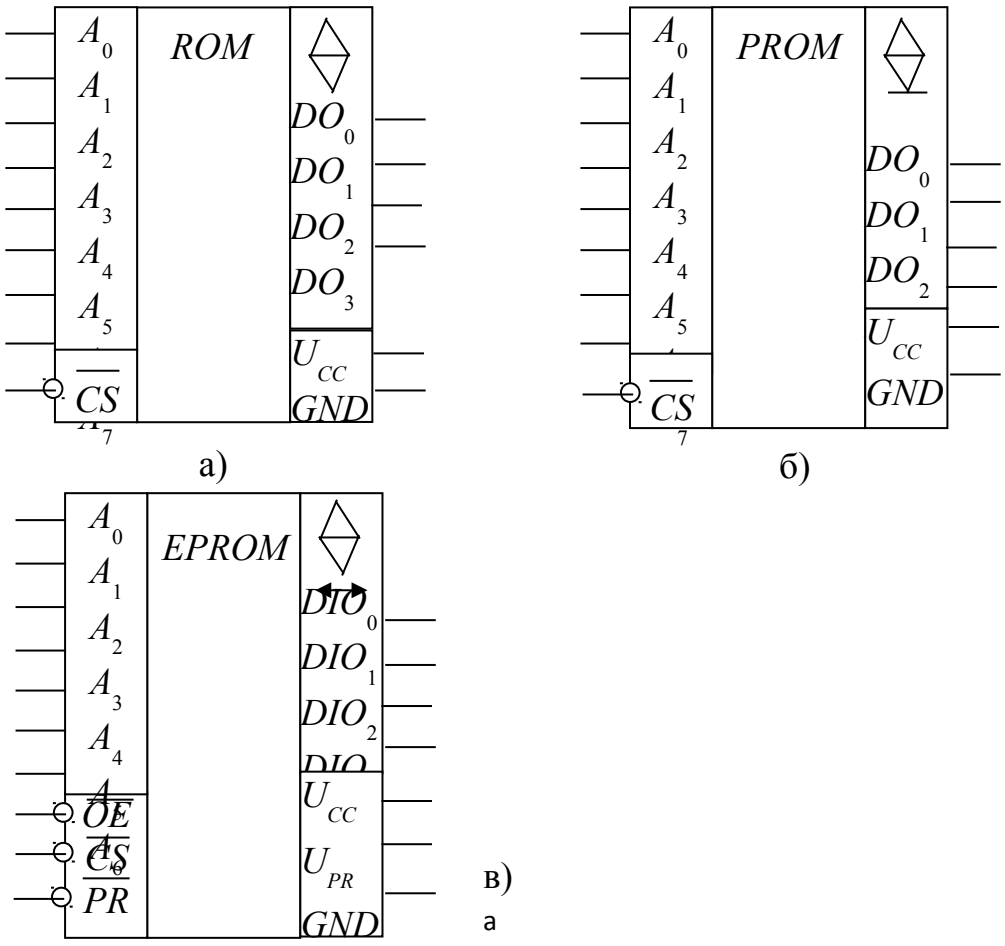


Рисунок 1.66 – Умовні графічні позначення мікросхем ПЗП

На рис. 1.66, а зображена мікросхема ПЗП програмована матрицею, ємністю 256 байт, з виходами, що можуть набувати трьох станів. Сигнал вибору мікросхеми *CS* має активний рівень, що дорівнює логічному 0. Тому для зчитування вмісту комірки необхідно подати код адреси і сигнал логічного 0 на вхід *CS*); на рис. 1.65, б показана мікросхема РПЗП програмована однократно з організацією 256 × 4. Вихідні каскади якої побудовано за схемою з відкритим

колектором. Керування зчитуванням відбувається як у попередньому випадку. Сигнал логічної 1 на вході *CS* переводить мікросхему у режим збереження інформації; на рис. 1.66, в показана мікросхема РПЗП з електричним стиранням. Виводи мікросхеми двоспрямовані.

Умовні графічні позначення ВІС ОЗП будуть відрізнятися від наведених наявністю відповідних сигналів керування.

1.13 Принципи побудови ЗП із завданою організацією

Задача побудови модуля (блока) ЗП в залежності від задання і початкових умов може розв'язуватися у різний спосіб. Умови задачі також можуть мати свої відмінності в залежності від конкретних умов функціонування цього блока.

Технічне завдання на розробку модуля ЗП повинно вміщувати:

− технічні характеристики МПС, для якої буде розроблюватися модуль: потрібна інформаційна ємність, розрядність і типи сигналів шини адреси та шини даних, розподіл адресного простору, наявність сигналів керування і їх рівні, довжина ліній проходження сигналів, інформаційна організація модуля пам'яті, наявність блоків живлення – рівні напруг та величини електричних струмів, які вони забезпечують тощо;

− часові характеристики сигналів керування: тривалість імпульсів керування, часові затримки між ними, співвідношення між перепадами цих сигналів;

− часові характеристики сигналів, які формуються блоком, що розроблюється, їх співвідношення з внутрішніми і зовнішніми сигналами в МПС ;

− електричні характеристики сигналів, на виході блока ЗП: рівні напруги, навантажувальна спроможність виходів блока.

Головна задача, що при цьому розв'язується – забезпечення необхідної інформаційної ємності і забезпечення розрядності сигналів даних. Розв'язання цієї задачі, в залежності від наявності мікросхем пам'яті може бути двох видів:

– в номенклатурі мікросхем пам'яті існує ВІС, яка відповідає завданню на інформаційну організацію модуля і забезпечує відповідні часові характеристики. В цьому випадку мікросхема встановлюється у ЦП і виконується узгодження рівнів сигналів керування або їх формування в разі необхідності. На цьому побудова модуля пам'яті вважається закінченою;

– в номенклатурі мікросхем пам'яті не існує ВІС, яка відповідає заданню на організацію модуля. В цьому випадку необхідно з наявних типів ВІС побудувати схему, що буде відповідати завданню. Ця задача має два варіанти: по-перше – у наявності є мікросхеми, що мають необхідну інформаційну ємність, але мають меншу розрядність даних, по-друге у наявності можуть бути ВІС, які мають меншу інформаційну ємність, ніж задано, але забезпечують розрядність шини даних.

Побудова модулів пам'яті ПЗП і ОЗП відбувається аналогічно.

Розглянемо приклад побудови ПЗП для використання у ЦП, який має 24-розрядну шину адреси, 8-розрядну шину даних, і на шині керування якого формуються сигнали:

– $\overline{OE}$ – з активним рівнем логічного 0, окремо для блока ПЗП;

– $\overline{W}/R$ – сигнал керування процесом запису/зчитування, активний рівень логічного 0 має сигнал *W*;

Інформаційна організація модуля, що розроблюється становить 115к × 8, в модулі пам'яті необхідно використовувати мікросхему РПЗП-УФ типу *AM*27*C*512, що має організацію 64к × 8. Початкова адреса комірки пам'яті для модуля – 000000_B.

По-перше, визначимо кількість мікросхем для забезпечення необхідної організації

$$N = \frac{115к \times 8}{64к \times 8} = 1{,}7968 \approx 2 .$$

Якщо у результаті отримано дробове число, то його необхідно заокруглювати **обов'язково** до більшого цілого числа. По-друге, визначимо останню адресу комірки пам'яті модуля. Дві ВІС модуля повинні працювати по-черзі, обробляючи кожен по 64к інформації. У сумі обидві ВІС можуть обробити 128к інформації, що буде відповідати $2^{17} = 131072_D$ адресам. Якщо подати це число у шістнадцятьковій системі числення, то отримаємо число $1FFFF_H$, яке й буде останньою адресою модуля.

ВІС, яку необхідно використовувати має такі виводи: шістнадцять адресних входів – $A_0 – A_{15}$; вісім виходів даних – $DO_0 – DO_7$; входи керування – $\overline{OE}$, $\overline{CS}$ і $\overline{CE}$. Таким чином, необхідно з'єднати 2 ВІС з шинами МПС і між собою так, щоб забезпечити чергування їх роботи в залежності від встановлення адреси.

Схема блоку приведена на рис. 1.67.

Таблиця істинності мікросхеми *AM*27*C*512 приведена у табл. 1.8.

Таблиця 1.8 – Таблиця істинності мікросхеми *AM*27*C*512

Назва сигналу, значення сигналу					Режим роботи
CE	*OE*	*CS*	A_0–A_{15}	*DO*	
1	X	X	X	z	Неактивна
0	1	X	X	z	Неактивна
0	0	1	X	z	Неактивна
0	1	0	X	z	Зберігання
0	0	0	*A*	*D*	Зчитування

Робота кожної з мікросхем відбувається відповідно до цієї таблиці за сигналами керування, що надходять від схеми ЦП. Робота модуля відбувається в діапазоні адрес, який був визначений раніше. Якщо ЦП формує адресу більшу ніж $01FFFF_H$, то на виході 7-входового елемента АБО (елемент *DD*1) формується сигнал логічної 1, який з'явиться на виході 2-входового елементу АБО (елемент *DD*2) і переведе обидві мікросхеми ВІС до неактивного стану.

Вибір однієї з двох мікросхем ПЗП відбувається сигналом A_{16} (лінія 17 адресної шини). Якщо на цій лінії є сигнал логічного 0 адреси в діапазоні $000000_H – 00FFFF_H$), то дозволяється робота мікросхеми *DD*4 на вхід *CE* якої надходить сигнал логічного 0. Якщо діапазон адрес, який виставлено на шину адреси становить $010000_H – 01FFFF_H$, то дозволяється зчитування з мікросхеми

*DD*5 на вхід *CE* якої надходить сигнал логічного 0 з виходу інвертора (елемент *DD3*). Зчитування інформації відбувається сигналом $\overline{W}/R$, що надходить на входи *OE* обох мікросхем.

Розглянемо приклад побудови ПЗП для використання у ЦП, який має 24-розрядну шину адреси, 8-розрядну шину даних, і на шині керування якого формуються сигнали:

– $\overline{OE}$ – з активним рівнем логічного 0, окремо для блока ПЗП;

– $\overline{W}/R$ – сигнал керування процесом запису/зчитування, активний рівень логічного 0 має сигнал *W*.

Інформаційна організація модуля, що розроблюється становить 64к × 16, але розрядність шини даних становить 16 розрядів (слово). В модулі пам'яті необхідно використовувати мікросхему ОПЗП статичного типу *AM*21*C*512, що має організацію 64к × 8. Початкова адреса комірки пам'яті для модуля – 000000_B.

Визначимо кількість мікросхем для забезпечення необхідної організації

$$N = \frac{115к \times 16}{64к \times 8} = 1{,}7968 \approx 2$$

Визначимо останню адресу комірки пам'яті модуля. Тому що, інформаційна ємність модуля складає 64к, то остання адреса буде $00FFFF_H$.

ВІС, яку необхідно використовувати має такі виводи: шістнадцять адресних входів – A_0 – A_{15}; вісім входів/виходів даних – DIO_0 – DIO_7; входи керування – $\overline{OE}$, $\overline{CS}$, $\overline{CE}$ і $\overline{W}/R$.

Для збільшення кількості розрядів шини даних слід з'єднати мікросхеми таким чином, щоб забезпечити їх одночасну роботу з однаковими адресами.

Схема блока наведена на рис. 1.68.

Робота мікросхем ОЗП відбувається згідно з таблицею істинності табл. 1.9 відповідно сигналів керування, що надходять від МПС. Обидві мікросхеми працюють одночасно (паралельно сигналів адрес і сигналів керування).

Таблиця 1.9 – Таблиця істинності мікросхеми ОЗП типу *AM*21*C*512

Назва сигналу, значення сигналу						Режим роботи
CE	*OE*	*CS*	*W/R*	A_0–A_{15}	DIO_0–DIO_8	
1	1	1	X	X	z	Неактивний стан
1	1	0	X	X	z	Неактивний стан
0	1	0	X	X	z	Зберігання
0	0	0	1	*A*	*DI*	Запис вхідних даних
0	0	0	0	*A*	*DO*	Зчитування даних

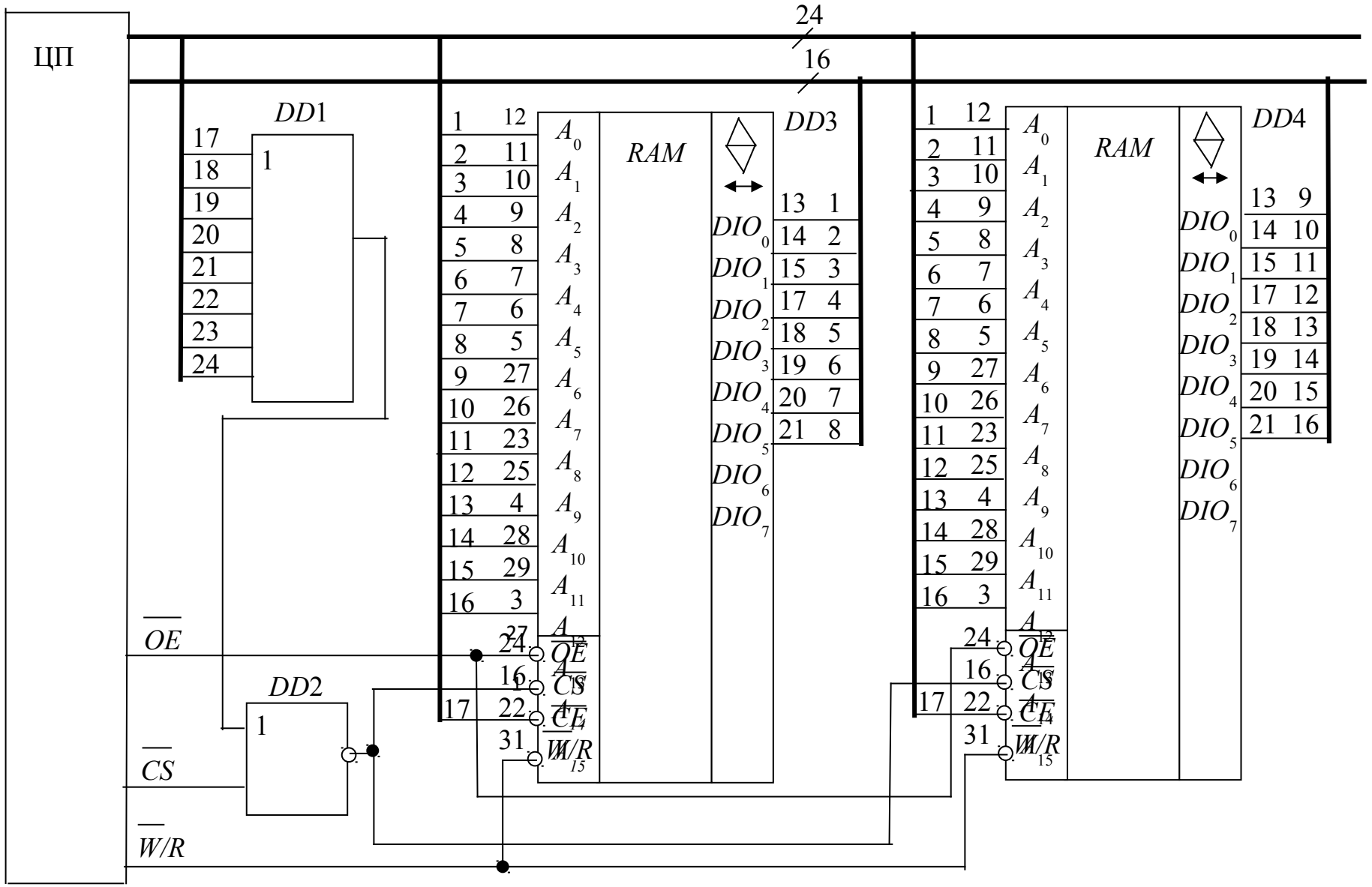


Рисунок 1.67 – Модуль ПЗП з організацією 128к × 8

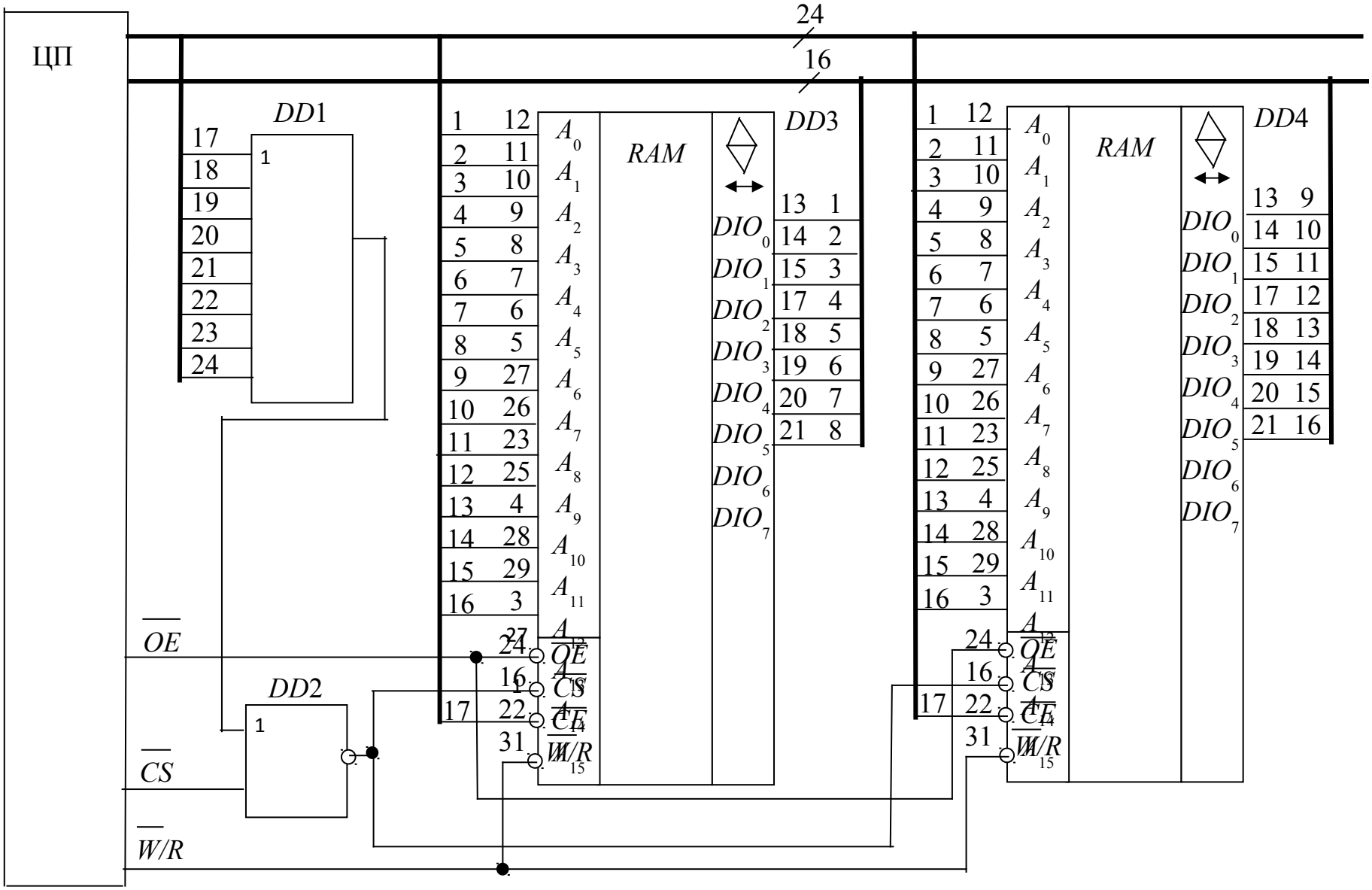


Рисунок 1.68 – Модуль ОЗП з організацією 64к × 16

75

1.14 Програмовані логічні інтегральні схеми

У цифровій техніці у багатьох випадках, наприклад, у телекомунікаціях, при виконанні цифрового оброблення сигналів можна доповнювати, а іноді навіть замінювати мікропроцесорні засоби на програмовані логічні інтегральні схеми (ПЛІС).

На них можна будувати спеціалізовані цифрові пристрої, керувальні засоби – контролери, і застосовувати у тих областях, де апаратним розв'язанням задач можна надати перевагу порівняно з програмними рішеннями, які завжди є послідовними. Застосування ПЛІС збільшує продуктивність системи іноді у десятки разів. У той самий час зберігається така сама гнучкість реалізації алгоритмів, як і при програмному способі. Задавати алгоритм дії проектованого цифрового пристрою для реалізації на ПЛІС можна у вигляді часових діаграм, текстового опису, схем на логічних елементах, у вигляді логічних функцій. Для отримання оптимального алгоритму використовується процедура мінімізації, як було показано вище. ПЛІС – це надвелика інтегральна схема, яка вміщує на кришталі універсальні налаштовувані користувачем функціональні перетворювачі та програмовані зв'язки між ними, що дозволяє скомпонувати в одному корпусі електронну схему, яка є еквівалентною до стандартної, що складається від десятків до кількох сотень стандартних логічних мікросхем. ПЛІС можуть реалізовувати блоки пам'яті, блоки цифрового оброблення сигналів, вбудовані процесорні ядра з периферією, швидкісні канали введення-виведення тощо.

При створенні систем на основі ПЛІС всі етапи розробки виконуються з використанням систем автоматичного проектування (САПР). Кожна компанія-виробник ПЛІС пропонує власну САПР, яка забезпечує повний цикл розробки схеми для кожного типу ПЛІС.

Найбільшу групу ПЛІС складають спеціалізовані по застосуванню апаратні засоби – *ASIC* (*application specific integrated curcuit*). До них відносяться інтегральні схеми, які за допомогою фізичних змін і/або спеціального програмного забезпечення можна привести до необхідного виду, який відповідає вимогам до розроблюваної схеми. Взагалі ця група представлена програмованими логічними інтегральними схемами – *PLD* (*programmable logic devices*).

У залежності від способу програмування мікросхеми *PLD* розподіляються на наступні групи:

- ВІС з плавкими перемичками (*fuse link*), які перепалюються;
- ВІС без плавких перемичок (*anti fuse*), в яких потрібні з'єднання формуються електричними засобами;
- однобітові *RAM*-комірки, тригери;
- *EPROM*-комірки з довготривалим зберіганням заряду, інформація в цих комірках може знищуватися під впливом ультрафіолетового випромінювання;
- *EPROM*-комірки з довготривалим зберіганням заряду, інформація в цих комірках може знищуватися електричним стиранням.

У залежності від структури ВІС *PLD*, в яких використовуються програмовані ТА- й АБО- матриці, розрізняють наступні типи *PLD*:

− *PLA* (*programmable logic array*) – програмовані ТА- і АБО- матриці;

− *PAL* (*programmable array logic*) – програмована ТА-матриця, фіксована АБО-матриця;

− *GAL* (*gated array logic*) – теж саме, що і *PLA,* але з додатковими програмованими вихідними схемами;

− *EEPROM* – програмована АБО-матриця, фіксована ТА-матриця;

− *FPGA* (*field programmable gate array*), *LCA* (*logic cell array*) – електрично програмована енергонезалежна логічна матриця;

− *CPLD* (*complex programmable logic device*), *EPLD* (*erasable programmable logic device*) – електрично програмована енергонезалежна логічна матриця, інформація може знищуватися ультрафіолетовим або електричним стиранням.

Схема ТА-матриці, яка використовується у ВІС *PLD* показана на рис. 1.69. На рис. 1.69, *а* показана повна схема ТА-матриці, а на рис. 1.69,б – спрощена.

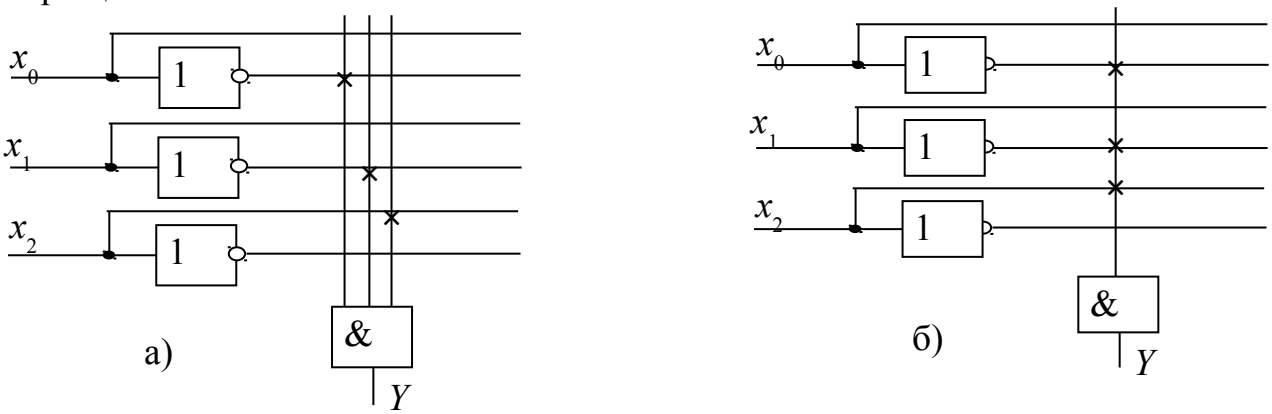


Рисунок 1.69 – Умовне графічне позначення ТА-матриці

На цьому рисунку передбачено, що символом × позначено з'єднання, яке вже встановлено, інші з'єднання можуть бути запрограмованими, в разі необхідності. Схема, яка показана на рис. 1.68 реалізує логічний вираз

$$Y = x_0 \wedge x_1 \wedge x_2 \ .$$

Така ТА-матриця разом з АБО-матрицею дає представлення про загальну схему ВІС *PLA*, яка показана на рис. 1.70.

На цьому рисунку кожен з трьох вихідних елементів АБО, на яких формуються вихідні сигнали $F_0 \ldots F_2$, має по шість входів.

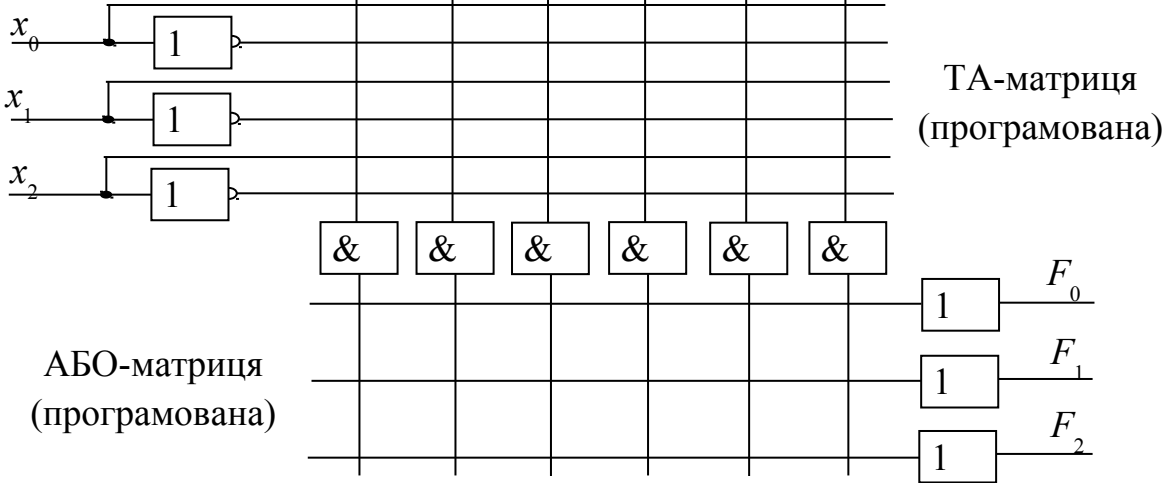


Рисунок 1.70 – Загальна схема ВІС *PLA*

Ця схема може доповнюватися елементами, які реалізують інверсію вихідних сигналів, можуть організовуватися виходи з «трьома стійкими станами», вихідні сигнали можуть тимчасово запам’ятовуватися тощо. У поєднанні з пристроями, які забезпечують формування вхідних і вихідних сигналів, а також певні можливості їх оброблення ці схеми становлять блоки з яких можна створювати інші типи ПЛІС, які розглядалися вище.

ІІ ВВЕДЕННЯ В МІКРОПРОЦЕСОРНУ ТЕХНІКУ

2.1 Обчислювальні та мікропроцесорні системи. Архітектура мікропроцесорів (МП). Програмні моделі універсальних мікропроцесорів фірми “Моторола”

Обчислювальна техніка **(ОТ)** (computer science, computing machinery) – це систематизована сукупність наукових дисциплін і галузей техніки, що досліджує обчислювальні машини, принципи їх побудови і використання; займається розробленням і тестуванням апаратних засобів для оброблення і зберігання інформації; архітектури обчислювальних систем; різні аспекти програмування, в тому числі питання розроблення і створення будь-якого програмного забезпечення; інформаційні структури; мови програмування тощо.

Електронна обчислювальна машина (ЕОМ)**,** *комп’ютер* – комплекс технічних засобів, призначених для автоматизованого оброблення інформації в процесі розв’язання обчислювальних та інформаційних задач.

Під *обчислювальним пристроєм* звичайно розуміють будь-який пристрій оброблення цифрової інформації: електронна обчислювальна машина (ЕОМ), мікропроцесор, персональний комп’ютер (ПК), мікропроцесорна система тощо.

Обчислювальна система (ОС) – це сукупність програм і технічних засобів, призначених для оброблення інформації.

Архітектура обчислювальної системи – це загальна логічна організація обчислювальної системи, яка визначає процес оброблення даних у ній та поєднує методи кодування даних, склад, призначення, принципи взаємодії технічних засобів і програмного забезпечення.

Процесор – це функційний пристрій, що забезпечує конкретне застосування сукупності команд.

Мікропроцесор (МП) – це обробляючий та керуючий цифровий пристрій, виконаний за технологією великих інтегральних схем (ВІС), який під програмним керуванням здатний виконувати оброблення інформації, а саме: арифметичні та логічні операції, введення-виведення та зберігання інформації, а також приймати рішення.

За типом архітектури розрізняють МП з фоннеймановською архітектурою і МП з гарвардською архітектурою.

За типом побудови мови програмування розрізняють *CISC*-процесори (*Complete Instruction Set Computing*) з повним набором команд та *RISC*-процесори (*Reduced Instruction Set Computing*) – зі зменшеним набором команд.

Сучасні мікропроцесори мають ознаки як *CISC*, так і *RISC*-архітектури.

Мікропроцесорна система (МПС) – це багатофункційна програмно-керована система оброблення інформації, яка складається з підсистеми центрального процесора, підсистеми пам’яті та підсистеми введення-виведення, об’єднаних інформаційними каналами.

Комп’ютер сам по собі є також мікропроцесорною системою.

У багатокомп'ютерних системах інформаційно комп'ютери взаємодіють один з одним по мережі, і кожен з них може працювати під керуванням своєї операційної системи, що знижує динамічні характеристики та надійність ОС.

Багатопроцесорні ОС працюють під керуванням однієї операційної системи, мають більш високу швидкодію та надійність.

Універсальні мікропроцесори призначені для застосування в обчислювальних системах та керувальних персональних комп'ютерах, робочих станціях, серверах, у суперкомп'ютерах.

Процесори цифрового оброблення сигналів (сигнальні процесори) – розраховані на оброблення у реальному часі цифрових потоків.

Сигнальні процесори апаратно підтримують фільтрацію та згортання сигналів, обчислення кореляційної функції двох сигналів, пряме та обернене Фур'є-перетворення сигналу тощо.

Медійні та мультимедійні мікропроцесори забезпечують апаратну підтримку оброблення аудіосигналів, графічної інформації, відеозображень тощо.

Мікроконтролер є інтегрована на одному кришталі ВІС мікропроцесорна система, яка складається з одного або кількох мікропроцесорів, підсистеми пам'яті та набору периферійних пристроїв.

Мікропроцесорний комплект – це сукупність інтегральних мікросхем різного ступеня інтеграції для побудови мікропроцесорних систем.

Контролером називається пристрій, часто вбудований, який призначений для керування технологічним пристроєм або процесом. Контролер будується на основі одного або кількох процесорів або мікроконтролера.

Трансп'ютери – це мікрокомп'ютери з власною внутрішньою пам'яттю та каналами (лінками) для підключення до інших трансп'ютерів з метою створення багатопроцесорних систем та паралельних обчислювальних систем.

Архітектура мікропроцесорів (МП)

Під архітектурою мікропроцесорів розуміють структурну схему самого МП, програмну (регістрову) модель МП, організацію пам'яті, яку забезпечує МП у складі мікропроцесорної системи, спосіб організації введення-виведення та мову Асемблера, яка керує цим процесором.

З цієї точки зору існують два основні типи архітектури – *фоннеймановська* та *гарвардська*.

Фоннеймановська архітектура показана на рис. 2.1.

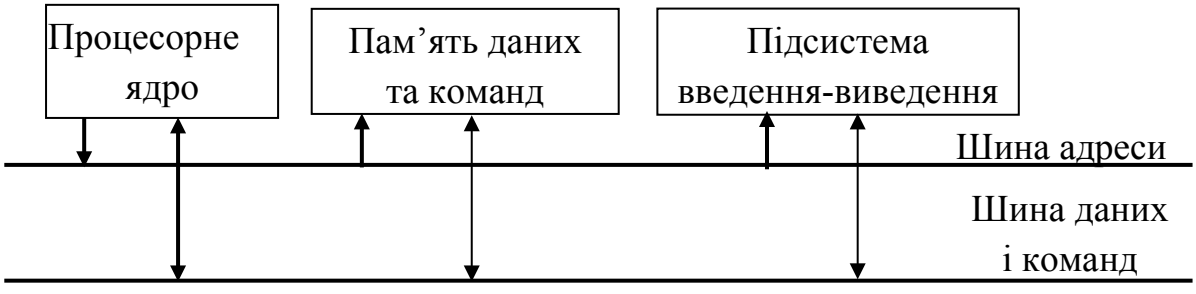


Рисунок 2.1 – Фоннеймановська архітектура

До загальних архітектурних властивостей та принципів побудови фоннеймановської архітектури можна віднести такі особливості:

1 Принцип програми, що зберігається, – це означає, що код програми та її дані знаходяться в єдиному адресному просторі в оперативній пам'яті, доступ до якої здійснюється по одній шині даних і команд.

2 Принцип мікропрограмування – полягає в тому, що певна комбінація мікрокоманд (зсуву, пересилання інформації, логічних операцій) може створювати набір команд МП.

3 Лінійний простір пам'яті, яку адресує МП, – сукупність комірок пам'яті з послідовним адресуванням.

4 Послідовне виконання команд програми – на послідовних ділянках програми МП вибирає з пам'яті команди строго послідовно. Розгалуження програм виконується за використанням спеціальних команд умовного та безумовного переходів і при зверненні до підпрограм.

5 Дані та команди розміщуються в одному просторі пам'яті у вигляді послідовності нулів та одиниць; процесор не бачить принципової різниці між даними та командами і намагається трактувати вміст деяких послідовних комірок пам'яті як коди машинної команди, а якщо це не так, то програма завершується аварійно. Тому важливо у програмі чітко розподіляти простір даних та команд.

6 Мікропроцесору все одно, яке логічне навантаження несуть дані, які він їх обробляє.

Для прискорення оброблення потоків цифрових сигналів у спеціалізованих процесорах цифрового оброблення сигналів використовують так звану гарвардську архітектуру, відповідно до якої процесорне ядро взаємодіє з двома банками пам'яті, як показано на рис. 2.2.

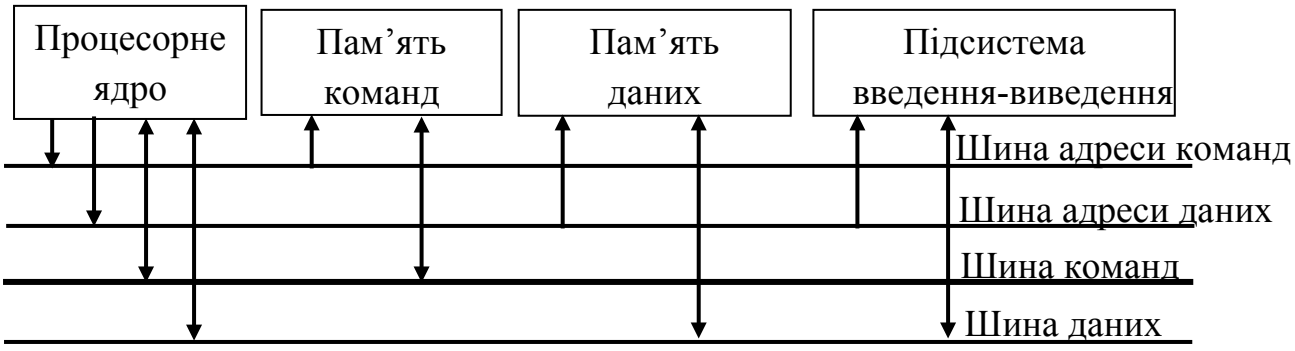


Рисунок 2.2 – Гарвардська архітектура

Звернення до кожного з банків пам'яті виконується за допомогою двох незалежних шин адреси та даних. Гарвардська архітектура дозволяє процесорному ядру за один цикл команди паралельно звертатись до пам'яті даних і пам'яті програм.

Програмні моделі універсальних мікропроцесорів фірми "Моторола"

МП сімейства MC680X0 фірми Motorola належать до універсальних МП з класичною CISC архітектурою і застосовуються у персональних комп'ютерах, серверах, цифрових системах комутації, маршрутизаторах, мобільному зв'язку,

контрольно-вимірювальній апаратурі тощо. Фірма Motorola випускає також широкий спектр напівпровідникових пристроїв від транзисторів до надвеликих інтегральних мікросхем. Вироби фірми Motorola відрізняються високими технічними характеристиками, якістю та надійністю, а вихід бездефектної продукції становить 99,9997 %. Все це зумовило широке розповсюдження виробів фірми Motorola на світовому ринку.

Базовою моделлю МП MC680X0 є МП MC68000.

Мікропроцесори цього сімейства використовуються в персональних комп'ютерах Macintosh, а також в розподілених системах керування складними об'єктами, обмін даними з якими відбувається стандартною шиною *VME*.

Базова модель – **мікропроцесор MC68000** – має 16-розрядну зовнішню шину даних та 24-розрядну шину адреси, що дозволяє адресувати простір пам'яті 16 Мбайт.

Максимальна тактова частота для цього процесора – 16,7 МГц. МП MC68000, який продукується за n-МОП-технологією, споживає близько 1,6 Вт, а мікропроцесор MC68HC000, який створено за КМОП-технологією, – 260 мВт. Тактову частоту процесора MC68EC000 доведено до 20 МГц, за споживання потужності 380 мВт.

Залежно від класу розв'язуваних завдань, програмні й апаратні ресурси МПС, поділяються в такий спосіб, щоб забезпечити передусім керування роботою власне мікропроцесорної системи, а також для розв'язання прикладних завдань користувача. Для розв'язання цих завдань реалізовано два режими МП:

– режим супервізора;
– режим користувача.

У режимі супервізора дозволено виконання усіх команд і забезпечено доступ до усіх регістрів МП. У режимі користувача заборонено виконування деяких команд і обмежено доступ до певних регістрів.

До складу регістрової моделі МП MC68000, наведеної на рис. 2.3, входять два набори 32-розрядних регістрів, кожний з яких вміщує по вісім однакових регістрів, програмний лічильник і регістр стану.

Вісім регістрів даних *D*7...*D*0 призначено для зберігання даних.

Вісім регістрів адреси *A*7...*A*0 призначено для зберігання адрес комірок пам'яті, визначених програмістом. Регістр *A*7 (*A'*7) використовується як вказівник стека. Залежно від режиму функціонування МП, він слугує вказівником стека користувача *USP* або вказівником стека супервізора – *SSP*.

Програмний лічильник PC призначено для зберігання адрес команд.

Регістр стану SR складається з системного байта та байта користувача. В режимі користувача доступ є лише до байта користувача – регістра ознак CCR, який використовується окремо. Формат регістру стану подано на рис. 2.4.

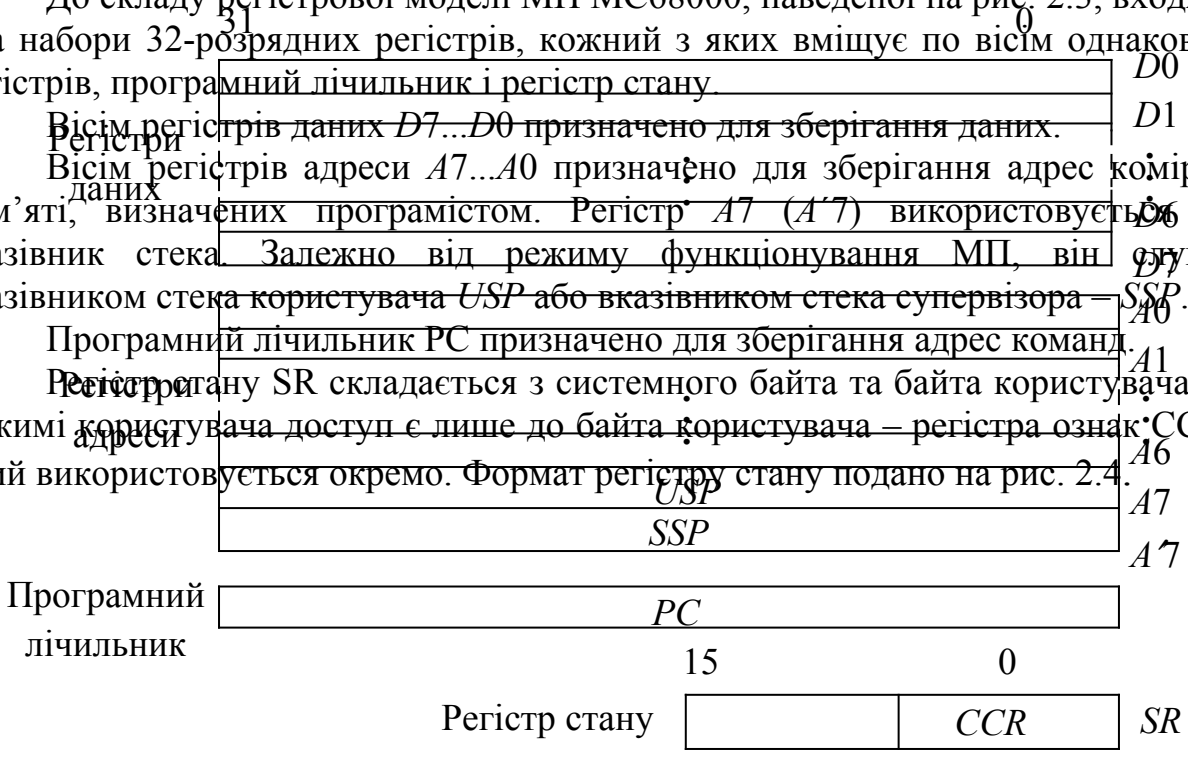


Рисунок 2.3 – Регістрова модель мікропроцесора MC68000

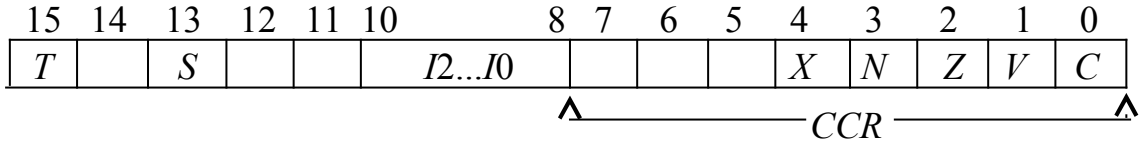


Рисунок 2.4 – Формат регістра стану *SR*

Регістр ознак (прапорців) *CCR* складається з восьми тригерів, п’ять з яких призначено для зберігання певних ознак, якими можна схарактеризувати результат виконання операції АЛП.

Окремі біти регістра ознак мають таке призначення:

− *C* (ознака перенесення) – перенесення зі старшого розряду при виконанні арифметичних операцій та зберігання вмісту висуваного розряду при виконанні операцій зсувів. Набирає значення *C* = 1 за виникнення перенесення зі старшого розряду результату виконуваної операції;

− *V* (ознака переповнення) – набуває значення *V* = 1 в разі переповнення розрядної сітки при обробленні операндів зі знаком;

− *Z* (ознака нульового результату) – за здобуття результату, який дорівнює 0, ознака набуває значення 1;

− *N* (ознака знаку) – зберігає копію знакового розряду результату операції. Значення ознаки N = 1 відповідає від’ємному результатові;

− *X* (ознака розширення) – при виконанні переважної більшості операцій копіює значення біта *C*. У певних випадках значення цього біта формується залежно від виконуваної операції.

Біти системного регістру *SR* визначають режим функціонування МП і мають призначення:

− *T* (ознака трасування) – за допомогою цієї ознаки здійснюється перехід до покрокового виконання програми за $T = 1$;

− *S* (ознака супервізора) – визначає режим роботи МП. Значення ознаки $S = 1$ відповідає функціонуванню МП в режимі супервізора, а $S = 0$ – в режимі користувача;

− *I*2...*I*0 – поле маски переривань, визначає мінімальний рівень обслуговуваних запитів переривань.

Інші біти регістру *SR* не використовуються або їх зарезервовано для використання в наступних моделях.

Мікропроцесор MC68000 обробляє дані – операнди, які можуть бути бітами, байтами, словами та довгими словами, а також дані, які подано у вигляді двійково-десяткових чисел. Операнди можуть розміщуватись у регістрах мікропроцесора (даних або адреси), а також у комірках пам’яті. При цьому, тому що шина даних МП має місткість 16 біт, а пам’ять має байтову організацію, то для роботи зі словами або довгими словами слід задавати адресу старшого байта операнда, яка має бути парною або кратною до 4. Це дозволяє звертатися спочатку до старшого байта слова, а потім до молодшого, який розміщується у комірці пам’яті, адреса якої є більша на 1 за адресу старшого байта. Таке послідовне розміщення операндів у пам’яті відповідає звичайному розміщенню розрядів числа при написанні чисел. Такий порядок адресування байтів називається big-endian. Адреса *N* завжди має бути парною.

Підключення ВІС мікропроцесора до шин МПС здійснюється за допомогою виводів, функційне призначення яких та їхні умовні назви подано на рис. 2.5 (виводи живлення на схемі не позначено). Напрям проходження сигналів зазначено стрілками. Підключення МП до шини адреси МПС здійснюється за допомогою 24-розрядної шини адреси (*A*23...*A*0), на яку виставляється адреса пристрою, до якого здійснюється звернення. Команди та дані надходять на 16-розрядну шину даних (*D*15...*D*0), якою здійснюється обмін словами. Передавання довгих слів здійснюється за два цикли шини (спочатку старша частина довгого слова, потім молодша).

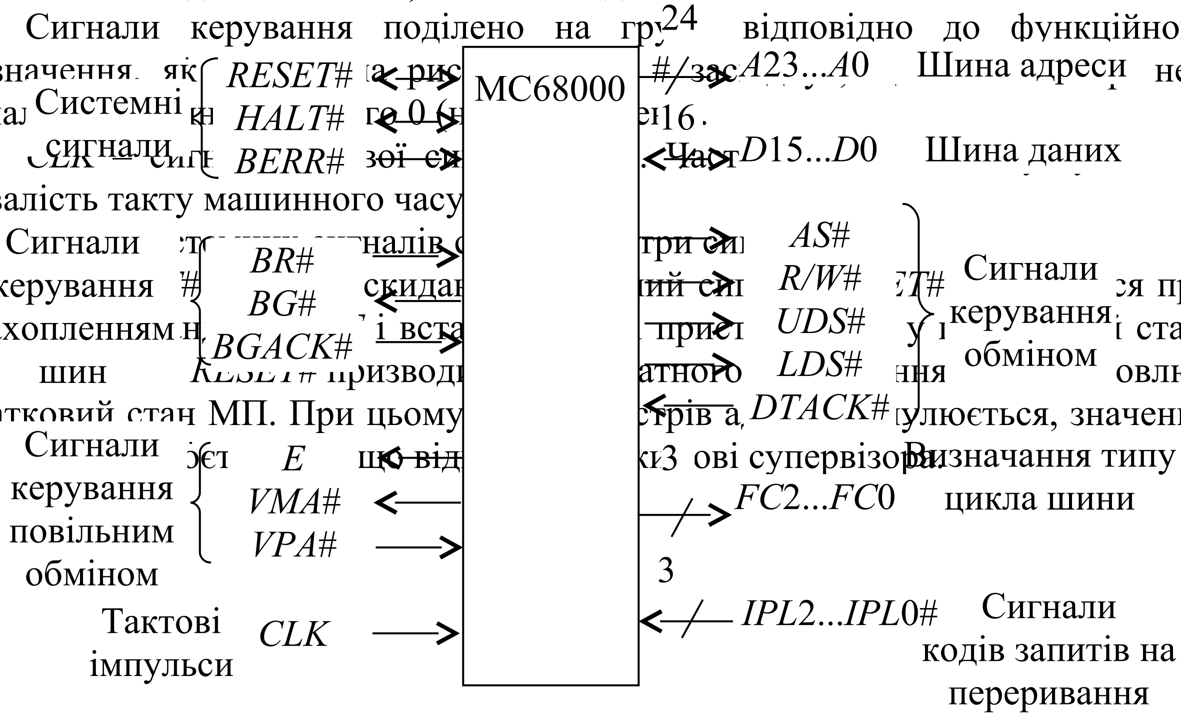


Рисунок 2.5 – Призначення виводів ВІС МС68000

– *HALT#* – сигнал зупину. Вхідний сигнал *HALT#* припиняє виконання поточної програми, переводить шину адреси та шину даних до відключеного стану, а виходи керування до неактивного стану. Вихідний сигнал *HALT#* формується за подвійної помилки звернення до шини. Вихід зі стану зупину відбувається подаванням вхідного сигналу *RESET#*.

– *BERR#* – вхідний сигнал помилки звернення до шини адреси. Цей сигнал формується схемою контролера шини, якщо на шину адреси подано адресу пристрою, що не існує, або адресу, яка не входить до адресного простору МПС. Цей сигнал також формується за тривалої відсутності сигналу готовності *DTACK#* і за інших порушень обміну.

Групу сигналів керування обміном подано п’ятьма сигналами:

– *AS#* – адресний строб є сигнал, формування якого зумовлює момент завершення формування адреси на шині; активний стан цього сигналу зберігається до завершення циклу обміну.

– *R/W#* – читання/запис. Значення сигналу зумовлює напрям передавання даних. Сигнал R/W#, який дорівнює 1, зумовлює читання (введення) даних в МП; значення сигналу, що дорівнює 0, –виведення (запис) даних з МП.

– *UDS#* – передавання старшого байта даних.

– *LDS#* – передавання молодшого байта.

Сигнали *UDS#*, *LDS#* зумовлюють розмір і порядок передавання даних шиною. Значення *LDS#* = 0 водночас з *UDS#* = 1 відповідає передаванню молодшого байта, *LDS#* = 0 та *UDS#* = 0 – передаванню слова.

– *DTACK#* – вхідний сигнал підтвердження готовності до обміну, який надходить від пам’яті або зовнішніх пристроїв.

Групу сигналів керування захопленням шин становлять сигнали, які визначають порядок спільного використання шини даних кількома пристроями системи. Ці сигнали використовуються для забезпечення режиму

безпосереднього доступу до пам’яті, а також для організації спільної роботи кількох МП у багатопроцесорних системах. До цієї групи входять сигнали:

– *BR#* – вхідний сигнал будь-якого зовнішнього пристрою на захоплення шини. Отримавши такий сигнал, якщо переривання дозволено, МП завершує виконання поточного циклу обміну, припиняє виконування команди, переводить власні виводи *A*23...*A*0 та *D*15...*D*0 до відключеного стану, а виходи керування до неактивного стану.

– *BG#* – вихідний сигнал дозволу захоплення шин. Формується після відключення МП від шини.

– *BGACK#* – вхідний сигнал підтвердження захоплення шини зовнішнім пристроєм. Формується зовнішнім пристроєм після отримання сигналу *BG#* і переходу до режиму керування шиною. Формування сигналу *BR#* після цього припиняється.

Сигнали керування повільним обміном дозволяють підключати до МП зовнішні пристрої, які мають відносно невелике значення тактової частоти. До цих сигналів належать:

– *E* – тактовий сигнал для зовнішнього пристрою.

– *VPA#* – сигнал готовності до повільного обміну. Цей сигнал формує зовнішній пристрій, адресу якого встановлено на шині адреси, і він є готовий до роботи.

– *VMA#* – сигнал підтвердження повільного обміну. Цей сигнал формує МП у відповідь на отримання ним сигналу VPA#. Після його встановлення розпочинається обмін даними у повільному режимі.

Мікропроцесор MC68020 є представником другого покоління цього сімейства. Характерними ознаками якого, на відміну від MC68000, є 32-розрядні шини адреси і даних, введення внутрішньої кеш-пам’яті.

Регістрова модель супервізора MC68020 показана на рис. 2.6.

Регістрова модель супервізора MC68020 вміщує додатково два 32-розрядні регістри *ISP* (*A*7') та *MSP* (*A*7''). Регістр *ISP* виконує функції вказівника стека супервізора при обробленні переривань та виключень, а регістр *MSP* використовується операційною системою при роботі у багатозадачному режимі. Регістр *VBR* вміщує базову адресу таблиці векторів переривань.

До трьох молодших розрядів регістрів *SFC* та *DFC* заноситься код адресного простору, який надходить на зовнішні виводи *FC*2…*FC*0 при пересиланні даних між регістрами процесора *D*7…*D*0 або *A*7…*A*0 та зовнішньою пам’яттю у привілейованих командах.

Регістри *CACR* та *CAAR* керують кеш-пам’яттю.

Формат регістру стану *SR* показано на рис. 2.7.

Призначення окремих бітів цього регістру таке саме як і для MC68000, крім таких:

− *T*1, *T*0 – визначають режим трасування;

− *M* – визначає вибір вказівника стека у режимі супервізора. Якщо *M* = 0, то використовується регістр *ISP*, а якщо *M* = 1, то використовується регістр *MSP*.

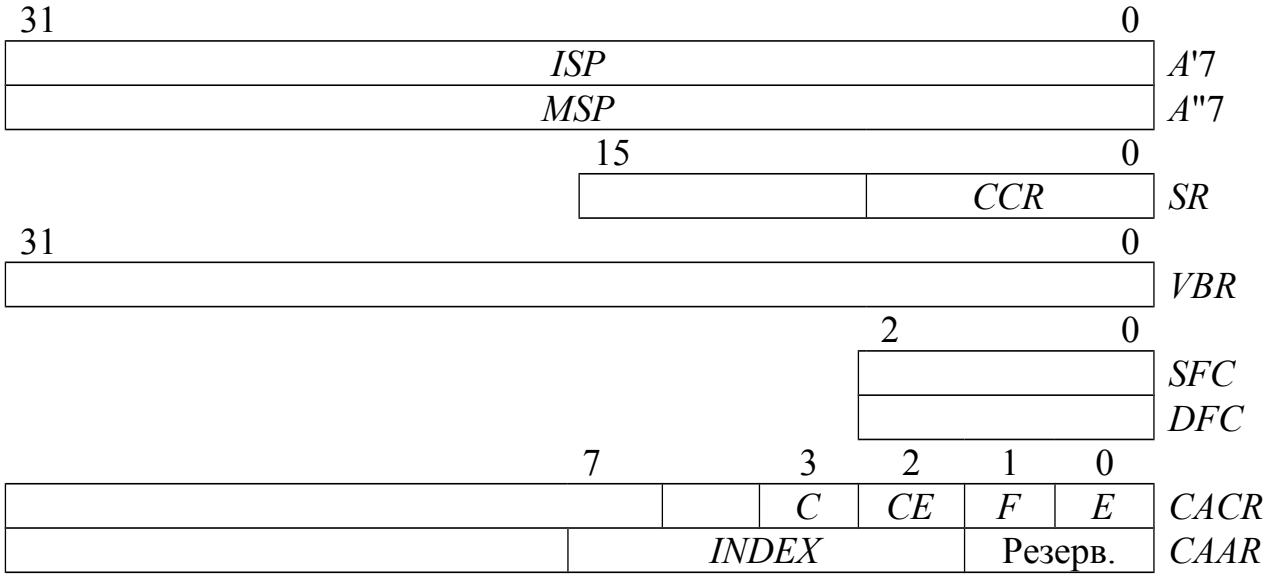


Рисунок 2.6 – Регістрова модель супервізора MC68020

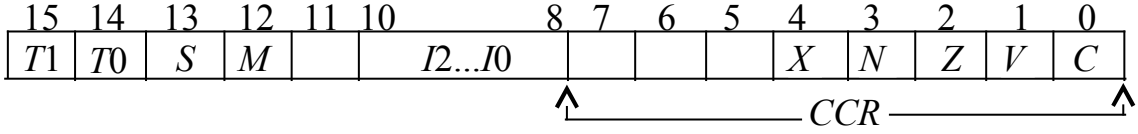


Рисунок 2.7 – Формат регістра стану SR

Функційне призначення виводів MC68020 та їхні умовні назви подано на рис. 2.8

Тому що цей МП є подальшим розвитком MC68000, то частина сигналів зберігає назви і функційне призначення.

Деякі сигнали в цьому МП відсутні, але є й такі що введено саме в ньому. До них відносяться наступні:

– *DSACK*1,0 – вхідні сигнали підтвердження готовності зовнішніх пристроїв до обміну;

– *DS#* – стробімпульс даних, який у циклі читання дорівнює одиниці зі стробом адреси, у циклі з значення *DS#* = 0 через один такт після адресного стробімпуль відсутнє, що МП видає шину даних;

Сигнал керування – вихідний си цикл обміну, який набирає значення *ECS#* = 0 упродовж першого та вого циклу;

– *OCS#* – вихідний с циклу обміну даними, який має значення *OCS#* = 0 упродовж у початкового циклу передавання даних; кеша

– *RMC#* – вихідний си тання-модифікація-запис.

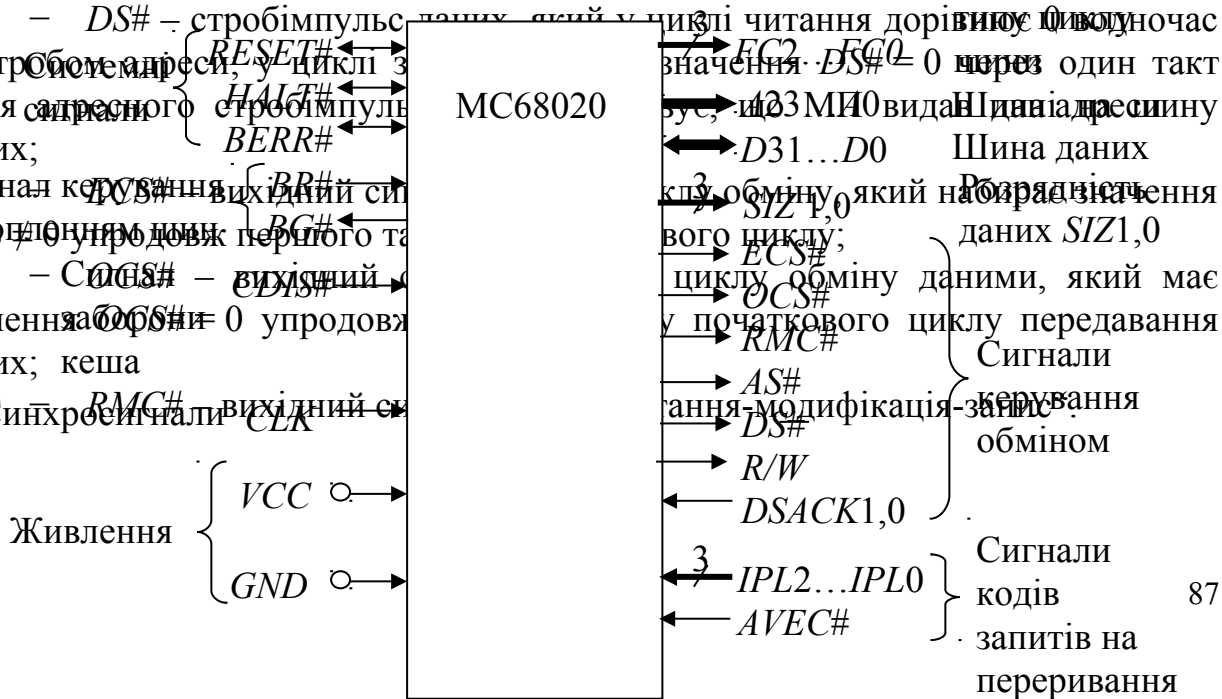


Рисунок 2.8 – Виводи MC68020 та їх функціональне призначення

Сигнал заборони роботи кеша *CDIS#* 0 звичайно використовується у режимі налагодження МПС.

Сигнал *AVEC#*, який дорівнює логічному нулю, дозволяє автовекторні переривання.

Значення вихідних сигналів *SIZ*1,0 вказують на розрядність даних.

На вхід *CLK* подаються зовнішні синхросигнали з частотою 16 МГц. Напруга живлення процесора становить 5 В, споживана потужність не перевищує 2 Вт.

2.2 Організація підсистеми уведення-виведення

Підсистема уведення-виведення інформації є компонентом типової архітектури МПС, яка надає можливість взаємодії із зовнішнім середовищем. Для МПС часто використовують термін – **інтерфейс**, як сукупність апаратного і програмного забезпечення для обміну інформацією між МП, пам’яттю і периферійним обладнанням.

Складність задач, розв’язуваних інтерфейсом, часто недостатня потужність буферних схем, які входять до складу самої ВІС МП, призводять до розподілення засобів інтерфейсу між різними пристроями:

1 Пристроями керування пам’яттю та уведення-виведення.

2 Безпосередньо інтерфейсним пристроєм, який є проміжним ланцюгом між МП і пам’яттю та пристроями уведення-виведення (системний інтерфейс).

3 Спеціалізованими пристроями керування (контролерами) уведенням-виведенням, призначеними для реалізації алгоритмів керування, специфічних для різних пристроїв (клавіатури, магнітних дисків, моніторів тощо).

4 Контролерами для спряження за допомогою шин з периферійними пристроями (давачами, виконавчими механізмами, комутаторами, АЦП, ЦАП).

Організація обміну між МП та пам’яттю або введенням-виведенням у простих випадках можлива на основі засобів, які є у самому МП. Ті функції, які не підтримуються апаратно мікропроцесором, реалізуються програмно.

Керувальні сигнали МП повинні забезпечувати функціональну та часову взаємодію між МП, пам’яттю та уведенням-виведенням у процесі обміну інформацією в основних режимах:

– для організації обміну з пам’яттю, крім адресних сигналів та сигналів даних повинні видаватись сигнали запису, читання, іноді сприйматися сигнал готовності ЗП;

– для програмного обміну даними, крім адресної інформації та самих даних, необхідні код вибирання пристрою введення-виведення, сигнали керування записом та читанням, повинні сприйматися сигнали готовності уведення-виведення тощо;

– для організації прямого доступу до пам’яті (ПДП) повинні бути підключені шина адреси для передавання адреси пам’яті у контролер ПДП та керувальні сигнали: код вибирання пристрою введення-виведення, запити від цього пристрою, дозвіл на обмін даними;

– у процесі роботи МП виробляє сигнали стану (зупину, чекання роботи у ПДП тощо) та синхронізуючі сигнали для інших пристроїв;

– для обміну даними у режимі переривань необхідно видавати з мікропроцесора керувальні сигнали дозволу переривань, підтвердження переривань та отримувати при готовності пристрою введення-виведення сигнали запиту на переривання.

Незважаючи на широке застосування системних інтерфейсів, можна виділити два способи звернення до пристроїв введення-виведення:

– із застосуванням спеціальних команд введення-виведення;

– за аналогією зі зверненням до комірок пам’яті за адресою.

У першому випадку адреса пристрою передається по тій самій шині адреси, що й адреси комірок пам’яті, і трактується як його номер тільки при формуванні спеціальних керувальних сигналів Введення або Виведення з пристроїв уведення-виведення, які ініціюються відповідними командами МП.

Для організації програмно-керованого обміну даними з пристроями введення-виведення (ПВВ-ПВИВ) на першому рівні (МП – контролер ПВВ-ПВИВ) достатньо простого набору сигналів (рис. 2.10).

На рис. 2.10, а показано використання керувальних сигналів при виконанні команд введення та виведення. Виконання операції введення починається з того, що МП виставляє на ША адресу ПВВ і сигналом Введення вказує на тип виконуваної операції. За сигналом Введення контролер ПВВ, який адресується, зчитує байт або слово даних з ПП, виставляє на лініях шини даних значення розрядів зчитаного слова та сигналом Готовність сповіщає про це МП. Прийнявши дане через контролер по ШД процесор знімає сигнали з ША та Введення. При виконанні операції виведення МП виставляє на ША адресу пристрою виведення, на ШД значення розрядів байта або слова, що виводиться, та сигналом Готовність сповіщає процесор, що дані прийняті і можна зняти інформацію з ША та сигнал Виведення.

При реалізації обміну за аналогією зі зверненням до пам’яті (рис. 2.10, б) використовуються тільки керувальні сигнали Запис у пам’ять та Читання з пам’яті, які використовуються при обміні МП з пам’яттю. Для адрес ПВВ та

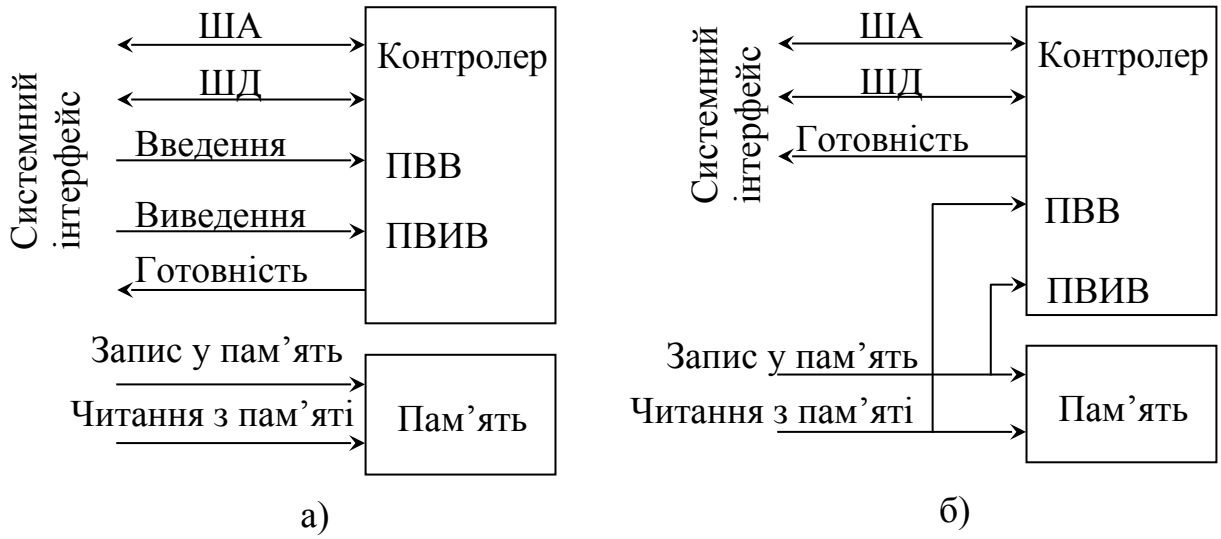


Рисунок 2.10 – Порядок використання керувальних пристроями ПВВ-ПВИВ

ПВИВ відводиться визначена частина адресного простору, де можна адресувати необхідну кількість пристроїв.

До складу мікропроцесорного комплекту мікросхем сімейства MC680x0 входять спеціальні мікросхеми за допомогою яких можна організувати паралельний та послідовний інтерфейси.

Асинхронний послідовний інтерфейс *DUART* – MC68681 фірми *Motorola*

ВІС фірми Motorola MC68681 (*DUART – DUAL ASYNCHRONOUS RESEIVER/TRANSMITTER*) – подвійний асинхронний приймач-передавач, призначений для організації двох незалежних послідовних асинхронних дуплексних каналів обміну (*A* та *B*) із зовнішніми пристроями.

DUART забезпечує такі можливості як:

– організація двох незалежних асинхронних дуплексних послідовних каналів введення-виведення;

– робота кожного з каналів на 18-ти фіксованих швидкостях обміну, які можуть програмуватися незалежно для кожного з каналів;

– робота з даними, довжина яких може становити від 5 до 8 бітів даних;

– можуть вибиратися такі режими роботи каналів:

нормальний (повний дуплекс);

автоматичного ехо-сигналу;

місцева кільцева перевірка;

віддалена кільцева перевірка;

– робота в режимі селекторного виклику;

– багатофункційний 16-розрядний лічильник/таймер може працювати у двох режимах:

формування часових інтервалів;

генерування імпульсів;

– 6-розрядний паралельний порт введення може слугувати за багатофункціональний пристрій введення інформації або використовувати чотири входи для вимірювання часових характеристик сигналів;

– визначення помилок роботи (зупин лінії, фальшстарт, надходження переривання тощо);

– робота від внутрішнього генератора з кварцовим стабілізуванням частоти або від зовнішнього генератора імпульсів;

– ВІС сумісна з ТТЛ-логікою.

Структурну схему *DUART* наведено на рис. 2.11. До структурної схеми входять:

– системний інтерфейс – схема, яка забезпечує керування роботою вузлів *DUART*, а також керування обміном інформацією з ЦП. На схему надходять сигнали:

R/W# – читання /запис;

CS# – вибір мікросхеми, при надходженні цього сигналу ВІС переводиться до активного стану;

RESET# – сигнал скидання;

*RS*4-*RS*1 (*Register Select*) – відповідні розряди шини адреси, які визначають адресу внутрішнього вузла регістра *DUART*;

DTACK# – сигнал підтвердження готовності ВІС до обміну, підтверджує наявність інформації на шині даних;

– буфер шини даних – схема, яка забезпечує обмін даними між шиною даних МПС і внутрішньою шиною *DUART*;

– схема керування перериваннями. На її входи надходить сигнал *IACK#* – сигнал підтвердження переривання – і формується сигнал *IRQ#* – номер запиту на переривання. До схеми входять такі вузли:

IMR (*Interrupt Mask Register*) – регістр маски переривання;

ISR (*Interrupt Status Register*) – регістр стану переривання;

ACR (*Auxiliary Control Register*) – допоміжний регістр керування;

IVR (*Interrupt Vector Register*) – регістр вектора переривань;

– схема керування синхронізацією. Схема визначає швидкість обміну інформацією у послідовних каналах. До її складу входять:

CSRA (*Clock Select Register channel A*) – регістр визначення швидкості обміну інформацією в каналі *A*;

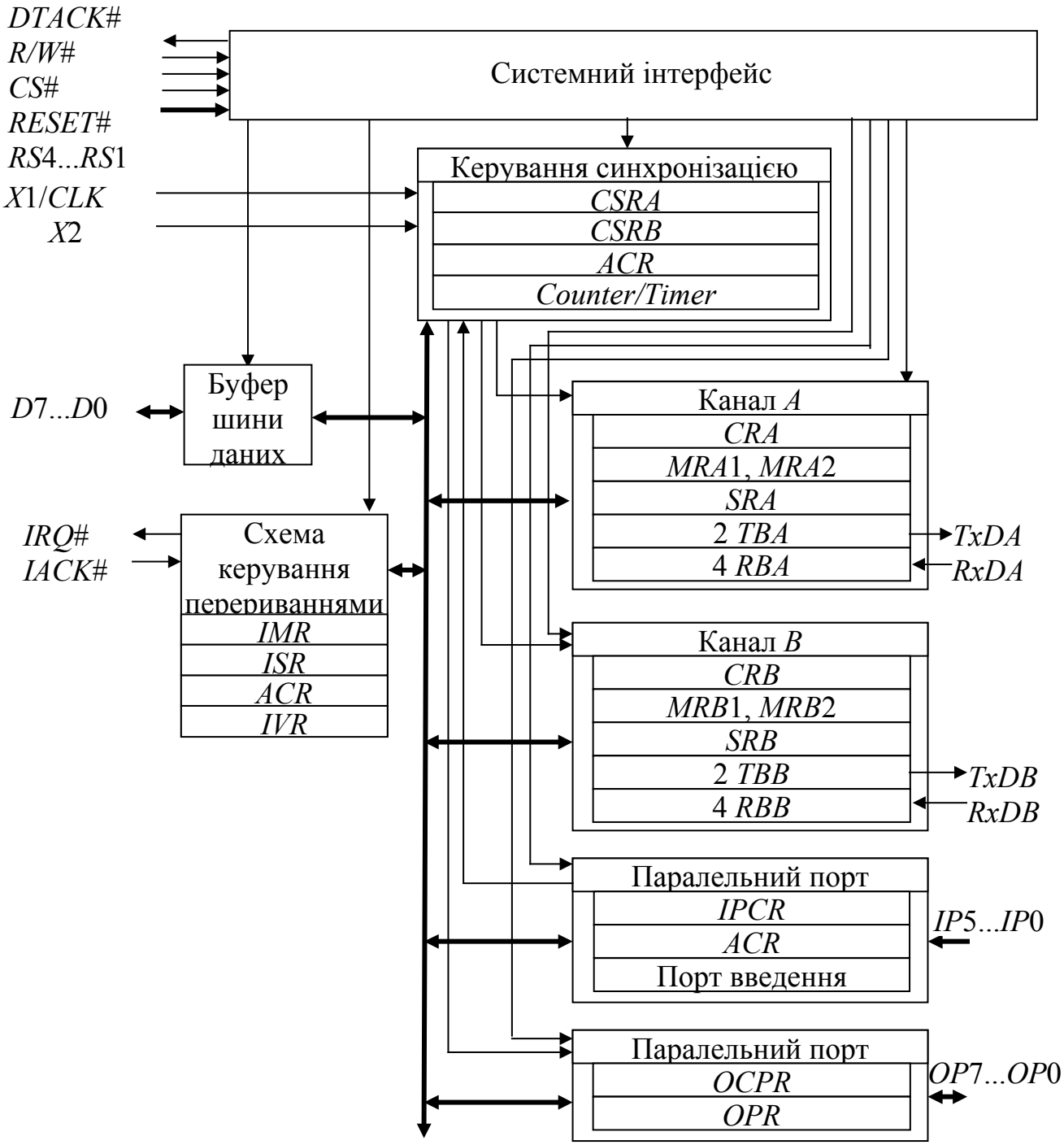


Рисунок 2.11 – Структурна схема *DUART*

CSRB (*Clock Select Register channel B*) – регістр визначення швидкості обміну інформацією в каналі B;

ACR (*Auxiliary Control Register*) – допоміжний регістр контролю;

Counter/Timer – 16-розрядний лічильник/таймер, складається з двох 8-розрядних регістрів, кожен з яких програмується окремо;

– два аналогічних асинхронних послідовних канали – *A* і *B*, які побудовано з однакових вузлів:

CRA (*B*) (*Command Register channel A* (*B*)) – командний регістр, який керує виконанням команд;

*MRA*1, *MRA*2 (*Mode Register* 1, 2) – регістри, вміст яких визначає режим роботи каналів введення-виведення даних;

SRA (*B*) (*Status Register Channel A* (*B*)) – регістри стану, зберігають інформацію про результати обміну;

TBA (*B*) (*Transmit Buffer channel A* (*B*)) – буферні регістри передавача відповідного каналу;

RBA (*B*) (*Receive Buffer channel A* (*B*)) – буферні регістри приймача відповідного каналу;

– 6-розрядний паралельний порт введення. До його складу входять:

IPCR (*Input Port Change Register*) – регістр стану порту введення;

ACR (*Auxiliary Control Register*) – допоміжний регістр контролю;

порт введення (6-розрядний регістр);

– 8-розрядний паралельний порт уведення-виведення. Багатоцільовий універсальний порт. До його складу входять:

OPCR (*Output Port Configuration Register*) – регістр стану.

OPR (*Output Port Register*) – 8-розрядний паралельний порт.

Кожен з двох каналів забезпечує роботу на 18-ти фіксованих швидкостях, які визначаються внутрішньнім генератором або сигналом від зовнішнього генератора, сигнал з якого надходить на вивід X1/CLK.

Програмування ВІС DUART здійснюється за допомогою запису сигналів керування (байтів) до відповідних регістрів.

Умовне графічне позначення ВІС і схема її підключення до шин МПС показано на рис. 2.12.

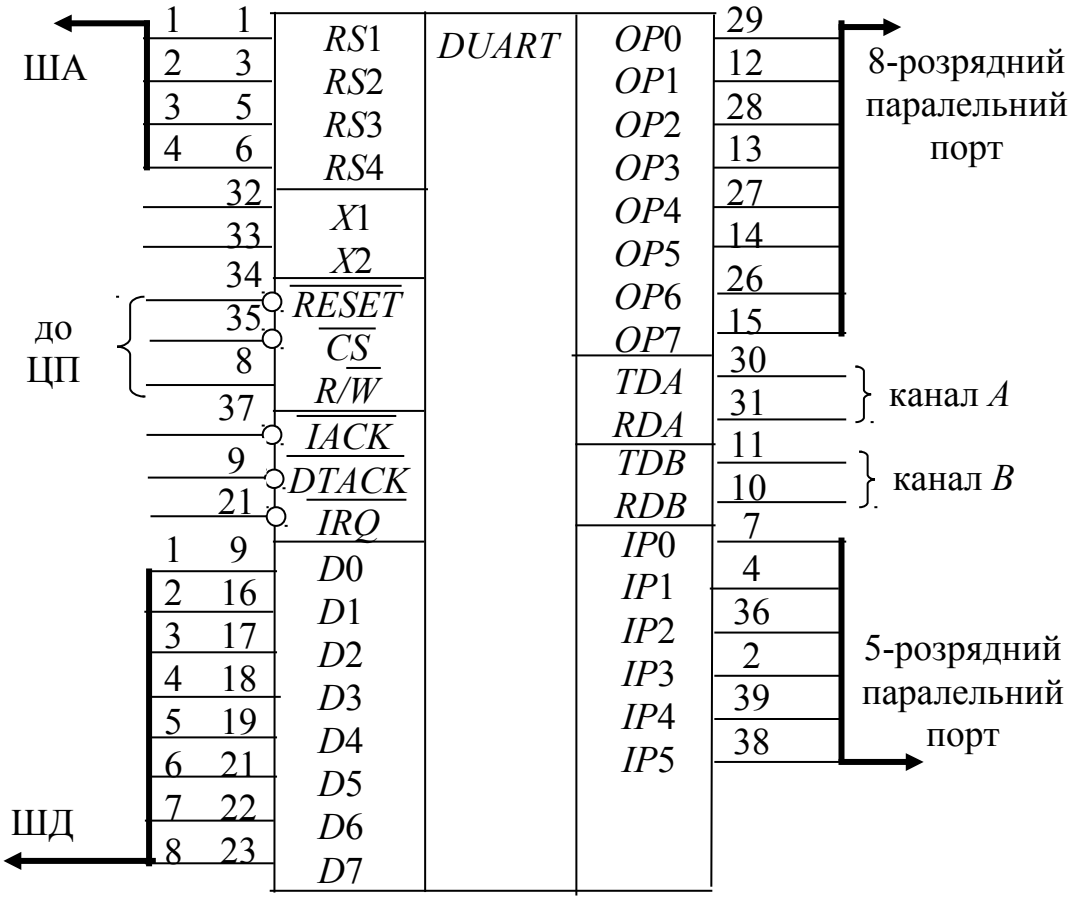


Рисунок 2.12 – Схема підключення ВІС *DUART*

Периферійний інтерфейс/таймер (*PIT*) MC68230 фірми *Motorola*

ВІС *PI/T* є паралельний інтерфейс/таймер, призначений для обміну 8-розрядними даними між мікропроцесором і зовнішніми пристроями (давачами, пристроями керування тощо). Структурну схему *PI/T* наведено на рис. 2.13.

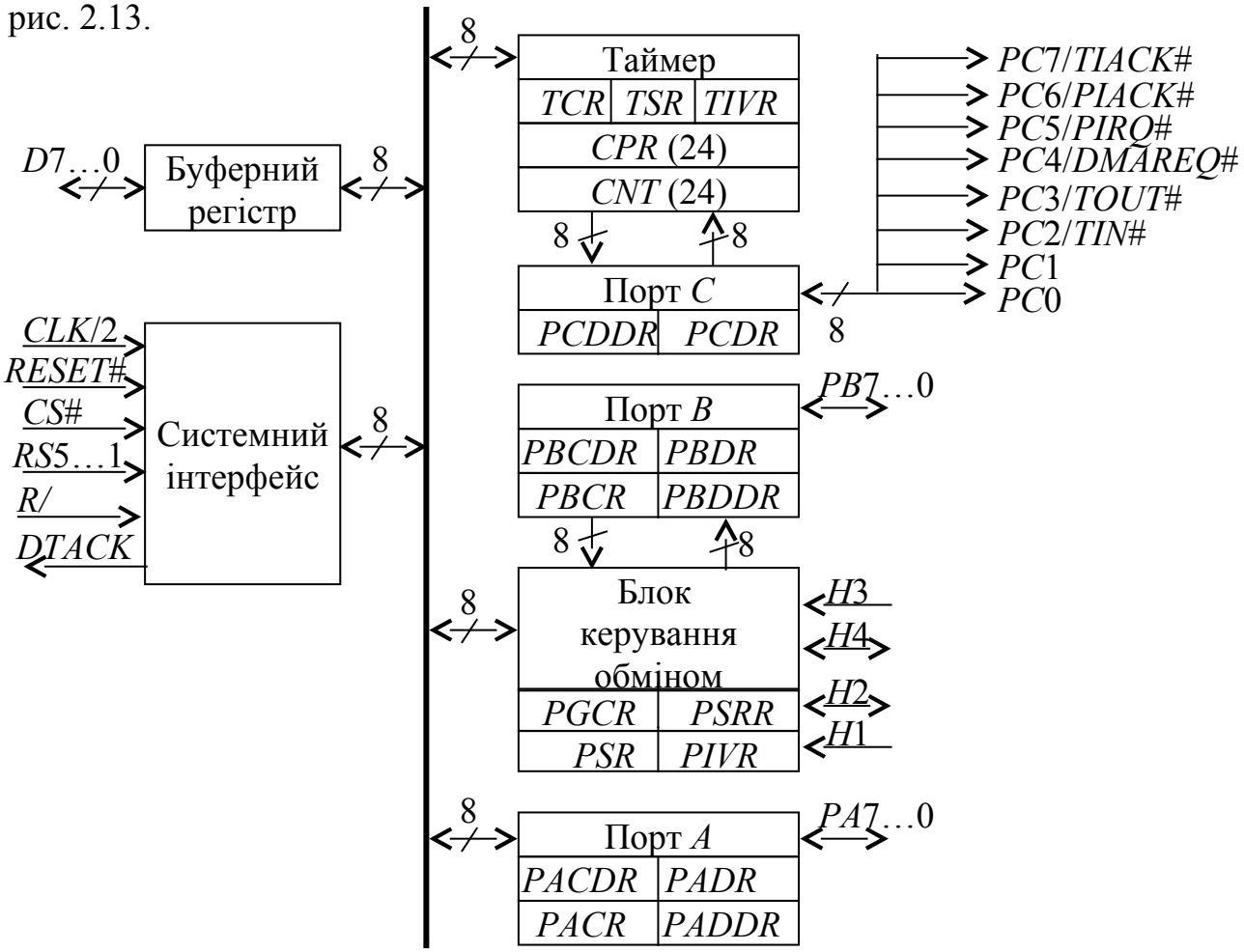


Рисунок 2.13 – Структурна схема *PI/T*

PI/T складається з блоків, які забезпечують зв'язок з мікропроцесором (буферний регістр, системний інтерфейс), і блоків, які обслуговують зовнішні пристрої (8-розрядні порти *A*, *B*, *C*, 24-розрядний таймер, блок керування обміном). Виводи *PC*7...*PC*2 можуть програмуватись для передавання сигналів таймера *TIN#*, *TOUT#*, переривання *PIRQ#*, *PIACK#*, *TIACK*, запиту прямого доступу *DMAREQ#*. Зв'язок *PI/T* з МП здійснюється за допомогою виводів *D*7…*D*0 у циклі читання або запису. МП видає на вхід системного інтерфейсу сигнал $R/\overline{W}$, а *PI/T* видає сигнал підтвердження готовності *DTACK#*.

Дані зчитуються або записуються до одного з регістрів таймера, порту *A*, *B*, *C* або блока керування обміном. Вибір регістра визначається кодом адреси, який надходить на входи адреси *RS*5…*RS*1. Всі регістри, крім лічильника *CNT* та регістру попереднього встановлення лічильника *CPR*, які мають 24 розряди, є 8-розрядні й адресуються як байт.

При зверненні до *PI/T* на входи *RS*5...*RS*1 надходить адреса регістру, яку формує МП, і сигнал *CS#* від адресного дешифратора. На вхід *CLK* надходять

синхросигнали *CLK/*2, а на вхід *RESET#* – загальний для всієї системи сигнал скидання.

PI/T є програмована ВІС, у керувальні регістри якої для реалізації різних режимів роботи треба записати керувальні коди, тобто зініціалізувати її.

Порти *A* та *B* забезпечують паралельний обмін даними між МП та зовнішніми пристроями. Для виведення даних МП записує їх до регістру даних відповідного порту: *PADR* або *PBDR*, а їхнє введення здійснюється шляхом їхнього зчитування з регістрів даних. Кожний вивід портів *A* та *B* може програмуватись окремо на введення або виведення встановленням у 1 (виведення) або у 0 (введення) бітів відповідного регістру – *PADDR* або *PBDDR*.

Блок керування обміном здійснює загальне керування портами *A* та *B*, а також керування функціями виводів *H*3...*H*1.

Порти *A*, *B* можуть програмуватись для реалізації чотирьох різних режимів обміну.

Порт *C* може використовуватись для обміну 8-розрядними даними, які записуються або зчитуються мікропроцесором через регістр *PCDR*. Виводи *PC*2, *PC*3, *PC*7 може бути запрограмовано для обслуговування таймера, виводи *PC*5, *PC*6 – для запиту на переривання та приймання підтвердження переривання від мікропроцесора. Вивід *PC*4 може використовуватись для формування запиту на прямий доступ до пам’яті.

Таймер, який входить до складу PI/T, реалізовано на базі 24-розрядного лічильника *CNT*, який працює на віднімання. Початковий стан лічильника встановлюється за ініціалізації регістру попереднього встановлення *CPR*. Запуск таймера здійснюється через запис до регістру керування таймером *TCR* керувального коду, який визначає режим його функціонування. Сигнали на *CNT* можуть надходити як від генератора синхроімпульсів *CLK*, так і від зовнішніх пристроїв. Таймер може працювати у режимі лічби імпульсів або у режимі ділення частоти вхідних імпульсів на 32.

Поточний стан таймера контролюється мікропроцесором через зчитування та аналіз вмісту регістру *TSR*.

Умовне графічне позначення ВІС *PI/T* показано на рис. 2.14.

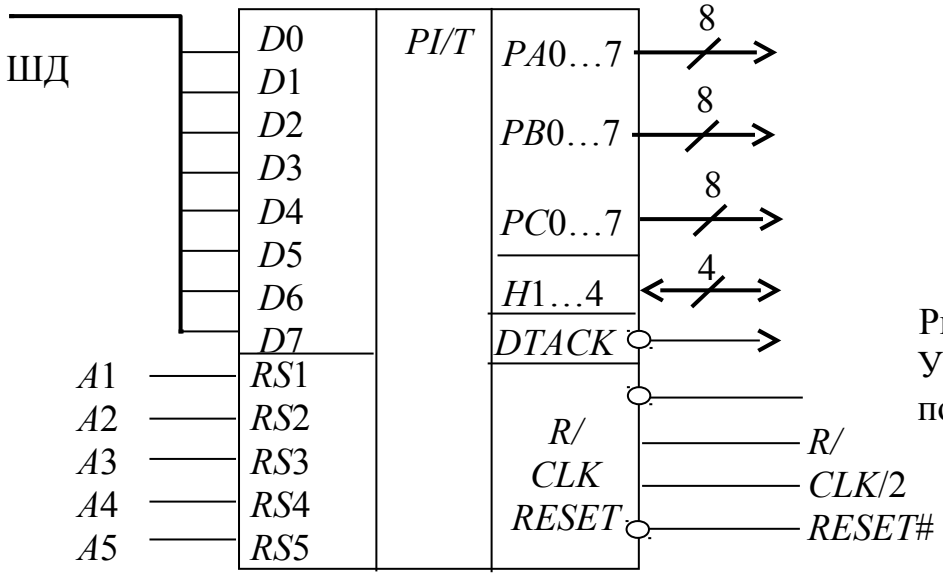


Рисунок 2.14 – Умовне графічне позначення ВІС *PI/T*

2.3 Організація підсистеми пам’яті

Підсистема пам’яті складається з блока ОЗП і блока ПЗП, які будуються за однаковими принципами. Різниця між ними полягає лише у необхідності забезпечення і формування специфічних для кожного з них сигналів керування.

Побудова підсистеми пам’яті здійснюється відповідно до положень розділу 1.13, які доповнюються тим, що ПЗП і ОЗП МПС MC68000 повинні працювати з даними, які можуть бути байтами, словами та довгими словами. Відповідно до цього, одночасно можливе звернення до однієї, двох або чотирьох комірок пам’яті. Шина даних в МПС MC68000 16-розрядна, і робота з довгими словами виконується за два цикли шини, тому при роботі з байтами і словами відбувається звернення до комірок пам’яті з однією адресою, а молодше і старше слова довгих слів розміщуються у двох сусідніх парах комірок. Для реалізації такого принципу побудови пам’яті необхідно будувати пам’ять з чотирьох блоків, кожен з яких призначено для роботи з байтами даних, поєднуючи їх відповідно до довжини операндів. Організацію такої пам’яті подано на рис. 2.15. Блоки ПЗП та ОЗП будуються в однаковий спосіб.

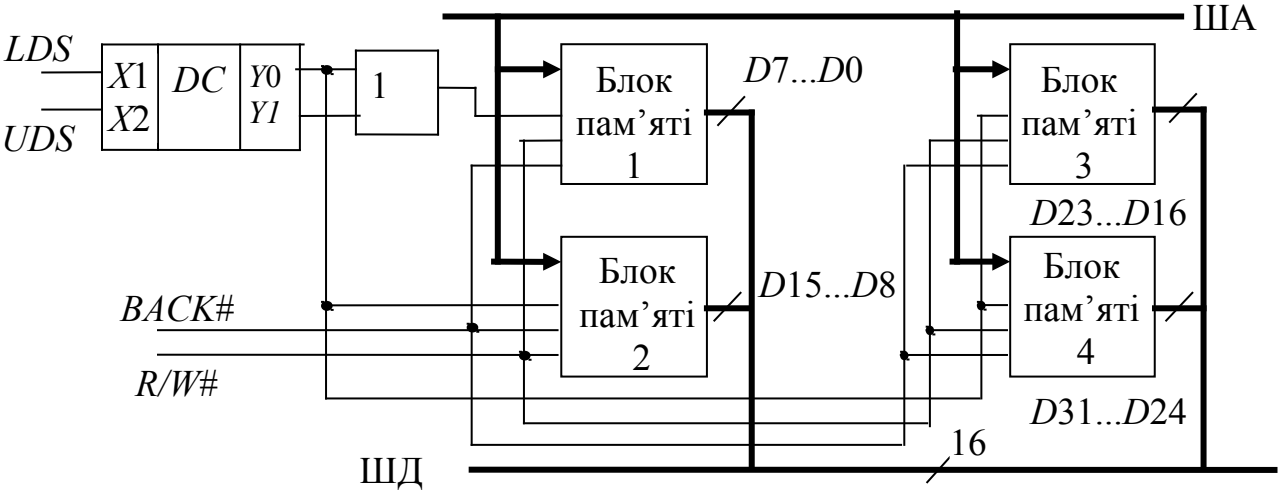


Рисунок 2.15 – Організація чотириблокової пам’яті

У цій пам’яті всі блоки пам’яті до шини адреси під’єднуються паралельно (до одних і тих самих розрядів), що забезпечить звернення до комірок з однаковими номерами. Сигнали вибору блока (BACK) і читання/запис (*R/W*) також надходять одночасно. Вибір відповідного блока здійснюється за допомогою дешифратора і схеми АБО. Якщо на входи дешифратора надходить код 00 (робота зі словами), то активний сигнал формується лише на виході *Y*0 і надходить на відповідні входи всіх блоків (на вхід блока 1 він проходить через логічну схему АБО). Отже, одночасно всі 16 виводів двох блоків пам’яті будуть з’єднані з шиною даних. Якщо на входи дешифратора надходить код 01 (робота з байтом), то сигнал *Y*1 дозволить роботу лише блокові 1; інші блоки в цей момент будуть перебувати в режимі зберігання інформації (будуть неактивними). Сукупність мікросхем пам’яті в усіх чотирьох блоках, які обслуговують однакові адреси, можна назвати шаром пам’яті. Отже, якщо

кожен з блоків пам’яті складається з кількох мікросхем пам’яті, то можна говорити про використовування багатошарової пам’яті.

Побудова кожного з блоків пам’яті здійснюється відповідно до положень розділу 1.13. Припустимо, що треба побудувати ОЗП з організацією 115к×8 з мікросхем АМ21С512, які було розглянуто в розділі 1.13. ОЗП має працювати з байтами, словами та подвійними словами. Кількість мікросхем, потрібних для побудови визначатиметься за виразом

$$N = 4\frac{115к \times 8}{64к \times 8} = 7{,}1872 \approx 8,$$

де 4 – кількість блоків пам’яті; 115к×8 – організація ОЗП кожного з блоків; 64к×8 – організація мікросхеми АМ21С512.

Якщо у результаті здобуто дробове число, то його слід заокруглювати **обов’язково** до більшого цілого числа кратного 4.

Отже, блок ОЗП вміщуватиме чотири блоки пам’яті, кожен з яких складатиметься з двох мікросхем. Схему цієї пам’яті наведено на рис. 2.16.

Керування схемою здійснюється сигналами, які формуються ВІС FPGA. Сумарний об’єм пам’яті кожного блока 128к. Блок 1 може працювати з байтами, разом з блоком 2 – зі словами, і всі чотири блоки – з довгими словами. Молодша частина довгого слова оброблюється блоками 1 та 2, старша – блоками 3 та 4. Керування шарами пам’яті в кожному з блоків здійснюється за сигналом адреси А16 у такий спосіб, що молодші комірки пам’яті з адресами $00000...$0FFFF оброблюються верхньою (за схемою) мікросхемою, а старші з адресами $10000...$1FFFF – нижньою. Для керування використовується вхід ОЕ, на який подається або сам сигнал *A*16, або його інверсія. Отже, якщо на вхід верхньої мікросхеми надходить сигнал безпосередньо з шини адреси, то в діапазоні адрес $10000...$1FFFF робота цього блока буде забороненою.

2.4 Організація переривань у мікропроцесорній системі (МПС). Типи переривань

Переривання і виключення – це спеціальні процедури, виконувані МП, якщо при роботі МПС виникають ситуації, які потребують тимчасового припинення головної програми й обслуговування інших програм (оброблення переривань) для відповідної реакції системи на обставини, які призвели до цього.

МП може обслуговувати до семи запитів на переривання від зовнішніх пристроїв, які формують сигнали IRQ1...IRQ7, відповідно до свого пріоритету. Ці сигнали оброблюються за допомогою пріоритетного шифратора і надходять на входи IPL#.

– **Виключення.** Виключення (Exception) – особливі ситуації при роботі МПС, які є реакцією системи на непередбачувані, переважно аварійні, ситуації, обслуговування яких передбачає переривання головної програми і перехід до підпрограми оброблення цього виключення. Виключення бувають апаратними або програмними. Апаратні виключення зумовлено будь-якими порушеннями

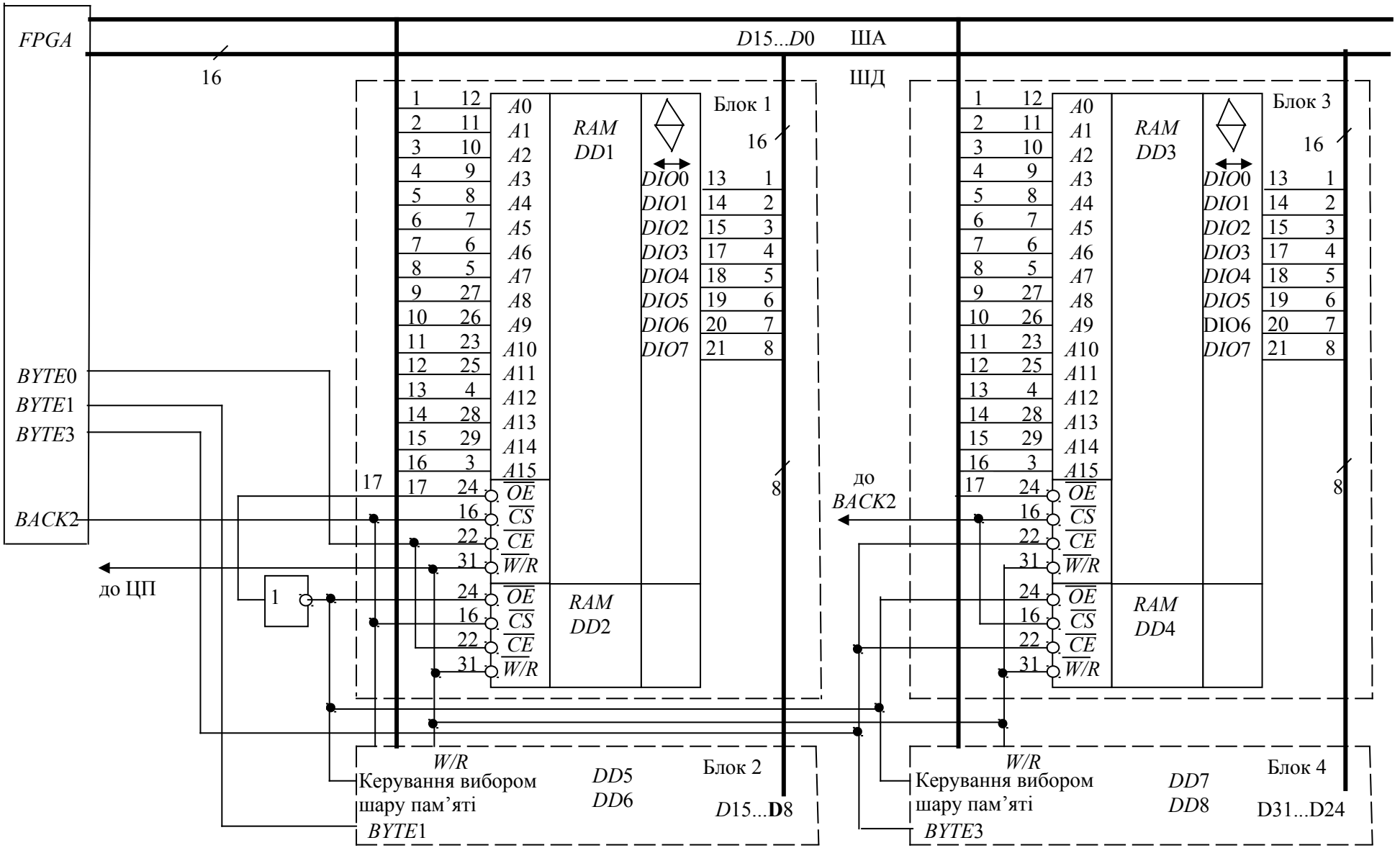


Рисунок 2.16 – Принципова схема блока пам'яті

функціонування пристроїв системи, як периферійних, так і внутрішніх, або аварійними (відключення живлення, відмикання з'єднувальних ліній тощо).

Програмні виключення виникають за неправильного функціонування програми: формування неіснуючої адреси, формування помилкового коду команди, ділення на 0, покрокового виконання програми тощо. При обслуговуванні виключень МП автоматично переходить до режиму супервізора, що надає підпрограмі обслуговування виключення доступу до всіх ресурсів системи.

Апаратні виключення зумовлено такими причинами:

– надходження зовнішнього сигналу RESET# ;

– формування зовнішнього сигналу помилки звернення до шини BERR#;

– надходження від периферійних пристроїв запитів на переривання;

– відсутність сигналів підтвердження готовності до обміну DTACK# або сигналу готовності до повільного обміну VPA#;

– надходження запиту на переривання від периферійного пристрою, який попередньо не ініційовано;

– відмикання живлення та інші аварійні ситуації.

Програмні виключення виникають за таких причин:

– надходження команд, які спричинюють виключення RESET, TRAP, TRAPV, CHK;

– надходження привілейованих команд у режимі користувача;

– ділення на 0 (команди DIVS, DIVU);

– надходження команди, яка має помилковий код операції;

встановлення покрокового режиму роботи (встановлення біта $T = 1$).

Переривання (Interruption) – особливі ситуації, які виникають при функціонуванні МПС для обслуговування певних пристроїв, робота яких відбувається за участі центрального процесора і не передбачена програмою, яка виконується.

Для обслуговування переривань МП припиняє виконання головної програми, зберігає у стеку адресу останньої команди, яку він виконав (адреса повернення), завантажує початкову адресу підпрограми обслуговування пер розпочинає її виконувати. Після завершення підпр МП пон програмному лічильнику адресу пов ення і продовжує вик овної програми. Обслуговування переривання розпочинається за запи переривання, як пристроєм, що обслуговування.

пристрої, які жу ити переривання ілено відпові жливості їхнього фу і кожному з них ачено власний р тет обслуговування. М уговувати до 1 ит ів на переривання від різних п їв.

схеми фор іння сиг о на рис. 2.17.

Рисунок 2.17 – Схема формування сигналів переривання

При опрацьовуванні векторного переривання зовнішній пристрій, який отримав сигнал *IACK#*, встановлює на молодші розряди шини даних *D*7...*D*0 свій номер виключення N_e і підтверджує передання номера, формуючи сигнал *DTACK#*. МП приймає значення номера і відповідно до нього формує адресу вектора виключення A_V = $100...$3FF.

Якщо при підтвердженні переривання не буде сформовано сигнали *VPA#* або *DTACK#*, то МП буде виконувати виключення з номером 24 – «помилкове переривання».

2.5 Поняття про мови програмування низького рівня. Мова програмування Асемблер мікропроцесорів фірми “Моторола”

Кожна мікропроцесорна система на МП певної моделі має свою власну мову програмування – мову машинних команд (машинну мову) і функціонувати може безпосередньо лише під керуванням програми, яка написана цією мовою. Машинна мова є сукупністю двійкових кодів усіх операцій, які може виконувати певний процесор. Машинна мова є незручною для широкого використання, тому що потребує досить великих витрат часу для написання і налагодження таких програм. Більш значного поширення набули мови програмування, які не співпадають з машинною мовою і є зручнішими за неї при використанні. Усі мови програмування можна поділити на дві групи: мови високого і низького рівнів (машинно-орієнтовані мови), що відбиває ступінь близькості цих мов до машинної.

Мова Асемблер – це машинно-орієнтована мова, яка дозволяє використовувати усі структурні особливості мікропроцесорної системи, що

пов’язані з апаратними можливостями, набором машинних команд, складом периферійного обладнання тощо. Мова Асемблер – це символічне подання машинної команди у вигляді її мнемонічного зображення. Програмування на Асемблері вимагає знання способів подання й оброблення даних на рівні машинних команд, що забезпечується знанням різних систем числення та архітектури МПС. Мова Асемблер поєднує у собі переваги машинної мови і деякі особливості мов високого рівня. Асемблер є найбільш ефективною мовою програмування при розв’язанні задач розробки системного програмного забезпечення, розробки програм для обміну даними з нестандартним периферійним обладнанням (драйвери), програмуванням систем реального часу для керування технологічними процесами й обладнанням.

Усі команди мови Асемблер можливо поділити за групами, відповідно до їх функціонального призначення: пересилання даних, арифметичні операції, логічні операції і зсуви, передачі керування, оброблення рядків даних і команди керування станом центрального процесора.

Текст програми мовою Асемблер складається з операторів (речень) чотирьох типів:

- команди, які є символічними аналогами машинних команд МПС;
- макрокоманди – конструкції, які при трансляції програми замінюються на послідовність відповідних команд;
- директиви – інструкції транслятору з виконання певних дій;
- коментарі – пояснення з виконання команди або фрагмента програми, використовуються для документування і кращого розуміння змісту програми; коментар ігнорується асемблером, тому він може бути написаний будь-якою мовою, і відокремлюється символом ; від іншої частини речення.

Речення повинно розміщуватися у межах одного рядка, переносити речення на інший рядок не дозволяється. Речення Асемблера формуються із найпростіших конструкцій мови – **лексем**, синтаксично нероздільних послідовностей допустимих символів алфавіту Асемблера, які мають значення для транслятора. Як символи алфавіту Асемблера використовуються:

- усі літери латинського алфавіту, при цьому великі й малі літери є еквівалентні;
- цифри від 0 до 9;
- спеціальні і розподілювальні символи, до яких відносяться: ?, @, $, _, &, [ ], ( ), < >, { }, +, –, /, *, %, !, “ “, \, =, #,^ і деякі недруковані символи.

До лексем відносяться: ідентифікатори, рядки символів, цілі числа двійкової, шістнадцяткової або десяткової систем числення.

Ідентифікатор – це послідовність символів, яка визначена програмістом і використовується для іменування об’єктів програми (міток, змінних, назв операцій тощо). Ідентифікатор може складатися з одного або кількох символів, але починатися може лише літерою. Крапка може бути лише першим символом ідентифікатора. Інші спеціальні символи використовувати в ідентифікаторах не дозволяється. Довжина ідентифікатора становить до 255 символів, але транслятор сприймає лише 32 перших символи, інші ігноруються.

Ідентифікатори поділяються на **службові слова** й **імена**. До службових слів відносяться ідентифікатори, значення яких однозначно визначено (назви регістрів, команд і таке інше). Імена – це символічні назви, які користувач призначає певним об'єктам програми (змінним, коміркам пам'яті тощо).

2.6 Формати даних і команд мови програмування Асемблер

Мова Асемблер МП сімейства M680X0 є спільна для МП MC68000 як базової моделі і подальших моделей і використовує формат двохадресної типової команди.

У загальному вигляді типова двохадресна команда мови Асемблер МП MC68000 має вигляд

COP.x <*src*>,<*dst*>,

де *COP* – мнемокод відповідної команди, замість *x* ставиться символ, який визначає розрядність операндів: *B* – байт, *W* – слово (16 розрядів), *L* – довге слово (32 розряди); якщо символ розрядності є відсутній, то за замовчуванням операнд є слово. Операнди умовно позначаються як <*src*> – джерело, <*dst*> – приймач, який слугує за власне приймач результату операції. При записуванні конкретних команд замість <*src*> та <*dst*> зазначаються символічні адреси операндів, відповідно до способу їхнього адресування. За безпосереднього адресування замість <*src*> зазначається число, перед яким ставиться префікс #.

Операнди, адреси та зміщення у командах можуть подаватись у системах числення, які зазначаються символами:

& – десяткова система;
% – двійкова система;
@ – вісімкова система;
$ – шістнадцяткова система,

які розміщують перед даними.

За відсутності символу, який визначає систему числення, дане сприймається як десяткове число.

2.7 Способи адресування операндів

Мікропроцесори MC68000 та старші моделі використовують такі способи адресування операндів:

- регістрове (операнд у регістрі даних або адреси), наприклад
  MOVE.B D1,D0 ; пересилання вмісту регістру *D*1 у регістр *D*0
  MOVEA.L A2,A3 ;
- непряме регістрове (операнд у комірці пам'яті, яка є адресована вмістом регістру адреси), наприклад,
  MOVE (A0),D1 ; пересилання вмісту комірки пам'яті, адреса якої знаходиться у регістрі *A*0 у регістр *D*1
- непряме регістрове з постінкрементом (операнд у комірці пам'яті, адресу якої розміщено в адресному регістрі і після виконання команди

нарощується на 1, 2 або 4 залежно від довжини зазначеного операнда), наприклад,

```
MOVE D3,(A1)+        ; Вміст A1 після виконання команди
                     ; нарощується на 2
```

− непряме регістрове з предекрементом (операнд у комірці пам'яті, адресу якої розміщено в адресному регістрі і перед виконанням команди зменшується на 1, 2 або 4 залежно від довжини операнда), наприклад,

```
MOVE.L -(A1),D2      ; Вміст A1 перед виконанням команди
                     ; зменшується на 4
```

− пряме (операнд у комірці пам'яті, адреса якої задається числом зі знаком, прямо зазначеним у команді), наприклад

```
MOVE.B $1234, D2     ;
```

− відносне (операнд у комірці пам'яті, адреса якої є сума поточного вмісту програмного лічильника *PC* та заданого у команді зміщення); поточний вміст *PC*, який використовується для обчислення відносної адреси, дорівнює адресі першого слова виконуваної команди плюс 2, наприклад

```
JMP *+$10            ; Для МП MC68000
JMP (*+$10,PC)       ; Для старших моделей
```

− безпосереднє (значення операнда задано безпосередньо у команді), наприклад

```
MOVEQ #$40,D3        ; Швидке завантаження
MOVEQ #64,D3         ; безпосередньо заданого операнда у D3
```

2.8 Команди пересилань. Лінійні програми

Основною командою цієї групи є команда MOVE – переслати, яка має низку варіантів, за допомогою яких можна прискорити виконання програми. Операції пересилань можуть використовувати всі способи адресування. Перелік команд наведено у табл. 2.1. У таблиці використовуються наступні умовні позначення:

− <EA> – вміст будь-якого запам'ятовувального пристрою МПС, який задано за допомогою його ефективної адреси, яка подана відповідно до способу адресування, який використовується. В якості <EA> також може використовуватися мітка, яка буде відповідати адресі певної команди;
− A_n – регістр адреси, до якого є звернення у команді;
− D_n – регістр даних, до якого є звернення у команді;
− R_x,R_y - будь-які регістри;
− #Im – безпосередній операнд (число), який завантажується у команді.

Таблиця 2.1 – Перелік команд пересилань

Синтаксис Асемблера	Розрядність операндів	Виконувана операція
1	2	3
MOVE <EA>,<EA>	B,W,L	Пересилання даних з операнда-джерело до операнда-приймача
MOVEA <EA>,A_n	W,L	Команда завантаження регістрів адреси

MOVEQ #Im,D_n	L	Завантаження у регістр даних безпосереднього числа
LEA <EA>,A_n	32	Визначення ефективної адреси, відповідно до способу адресування, який використовується і завантаження його до регістру A_n
PEA <EA>	32	Визначення ефективної адреси, відповідно до способу адресування, який використовується і завантаження його до стека
EXG R_x,R_y	16	Обмін даними між двома будь-якими регістрами даних або адреси

Слід відзначити, що команда пересилання MOVE виставляє прапорець $N = 1$ у разі негативного операнда, який пересилається, і скидає всі інші прапорці. Якщо пересилається операнд, який дорівнює 0, встановлюється прапорець $Z = 1$, а інші скидаються. Прапорець X є індиферентний відносно значення операнда, який пересилається.

Команди арифметичних операцій

Основними командами у цій групі є команди додавання, віднімання, множення і ділення. Основні команди наведено у табл. 2.2.

Таблиця 2.2 – Команди арифметичних операцій

Синтаксис Асемблера	Розряд-ність операндів	Виконувана операція	Опис команди
1	2	3	4
Команди додавання			
ADD D_n ,<EA> ADD <EA>, D_n	B,W,L	<EA> + D_n → <EA>	Додавання вмісту операнда-джерело до вмісту операнда-приймача. Результат записується в операнда-приймач

Закінчення табл. 2.2

ADDX D_n ,D_m	B,W,L	$D_m + D_n + X \rightarrow D_m$	Додавання операндів з урахуванням значення ознаки розширення
ADDX –(A_n), –(A_m)	B,W,L	<Оп2> + <Оп1> + X → <Оп1>	Додавання операндів з урахуванням значення ознаки розширення – *X*. Операнди задано непрямим регістровим адресуванням
Команди віднімання			
SUB D_n ,<EA>	B,W,L	<EA> − D_n → <EA>	Віднімання операнда-джерело від вмісту регістру даних. Результат записується у регістр даних
NEG <EA>	B,W,L	0 − <EA> → <EA>	Команда, яка змінює знак операнда і перетворює його у доповнювальний код
NEGX <EA>	B,W,L	0 − <EA> − X → <EA	Команда, яка змінює знак операнда і перетворює його у доповнювальний код з урахуванням значення ознаки розширення
Команди множення			
MULS <EA>, D_n	16	D_n * <EA> → D_n	Множення зі знаком слова з *D*2 на слово, яке є заданим дозволеним способом адресування
MULU <EA>, D_n	16	D_n * <EA> → D_n	Множення без знаку слова з *D*2 на слово, яке є заданим дозволеним способом адресування
Команди ділення			
DIVS <EA>, D_n	16	D_n / <EA> → D_n	Ділення зі знаком слова з *D*2 на слово, яке є заданим дозволеним способом адресування
DIVU <EA>, D_n	16	D_n / <EA> → D_n	Ділення без знаку слова з *D*2 на слово, яке є заданим дозволеним способом адресування
CLR <EA>	B,W,L		Обнулення значення операнда

Команди арифметичних операцій формують значення всіх прапорців ознак результатів виконаної операції.

До групи команд арифметичних операцій також відносяться команди виконання операцій над двійково-десятковими числами.

Команди логічних операцій

Основними командами у цій групі є команди логічного додавання (АБО), логічного множення (ТА), виключного АБО, інверсії (НІ). Основні команди логічних операцій наведено у табл. 2.3.

Таблиця 2.3 – Команди логічних операцій

Синтаксис Асемблера	Розряд-ність операндів	Виконувана операція	Опис команди
1	2	3	4
Команди логічного множення			
AND D_n,<EA>	B,W,L	<EA> $\wedge$ $D_n \rightarrow$ <EA>	Логічне множення (ТА) операнда-джерело з вмістом регістру даних. Результат записується у регістр даних
AND <EA>, D_n	B,W,L	$D_n \wedge$ <EA> $\rightarrow$ <EA>	
Команди логічного додавання			
OR D_n,<EA>	B,W,L	<EA> $\vee D_n \rightarrow$ <EA>	Логічне додавання (АБО) операнда-джерело до вмісту регістру даних. Результат записується у регістр даних
OR <EA>, D_n	W,L	$A_n \vee$ <EA> $\rightarrow A_n$	
Команди виконання операції виключне АБО			
EOR D_n,<EA>	B,W,L	<EA> $\forall$ $D_n \rightarrow$ <EA>	Виконання операції виключне АБО з операндом-джерело і вмістом регістру даних. Результат записується у регістр даних
EOR <EA>,D_n	W,L	$A_n \forall$ <EA> $\rightarrow A_n$	

Команди операцій зсувів

Команди зсувів дозволяють зсунути вміст регістру або комірки пам’яті на певну кількість розрядів праворуч або ліворуч.

Команди зсувів поділяються на арифметичні і логічні. Для логічних зсувів характерно, що біт, який виходить за межі регістру втрачається, а на місто біта, який зсунувся, у регістр записується 0. При виконанні арифметичних

зсувів праворуч знаковий біт не втрачається, а зберігається у сусідньому, тим самим зберігаючи знак числа. Кількість розрядів, на яку виконується зсув, записується або безпосередньо у команді, як безпосереднє число (1...8), або у регістрі даних – операнда-джерело. Операнд, який записано у пам'яті, можна зсувати тільки на 1 біт, і його розмір має бути не більше за слово.

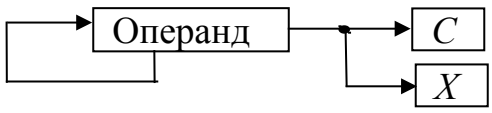
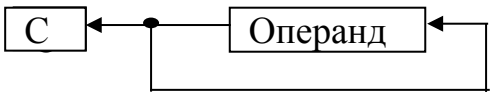
Команди зсувів і схеми за якими вони виконуються наведено у табл. 2.4.

Команди операцій над окремими бітами

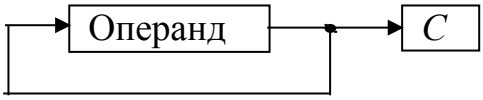
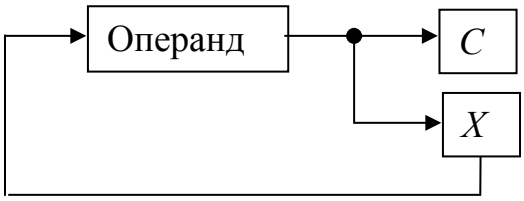
Команди операцій над окремими бітами виконуються з окремими бітами операнда, який зазначено в команді. Якщо операнд розміщено у регістрі даних, він трактується як довге слово, якщо у комірці пам'яті, – як байт. Номер тестованого біта задається вмістом регістру *Dn* або безпосереднім операндом *Nb*. Значення $Z = 1$ встановлюється, якщо тестований біт $b_n = 0$, і $Z = 0$, якщо $b_n = 1$. Деякі операції також встановлюють відповідне значення біта, який тестується. Команди операцій над окремими бітами наведено в табл. 2.5.

У сучасних додатках рідко використовується програмне забезпечення, повністю написане мовою Асемблера. Частіше мова Асемблера використовується разом з мовою високого рівня, наприклад, *C/C*$^{++}$. Частина програми, написана мовою Асемблера, звичайно призначена для керування периферійними пристроями, оскільки розв'язання таких задач мовами високого рівня є більш складним і менш ефективним. Крім того, програми мовою Асемблер виконуються значно швидше, ніж написані будь-якою мовою високого рівня.

Таблиця 2.4 – Команди зсувів

Мнемоніка та формат команди	Опис команди	Опис виконання команди
1	2	3
ASL #Im,D_n ASL D_m,D_n	Арифметичний зсув ліворуч	
LSL #Im,D_n LSL D_m,D_n	Логічний зсув ліворуч	
ASR #Im,D_n ASR D_m,D_n	Арифметичний зсув праворуч	
LSR #Im,D_n LSR D_m,D_n	Логічний зсув праворуч	
ROL #Im,D_n ROL D_m,D_n	Циклічний зсув ліворуч	

Закінчення табл. 2.4

1	2	3
ROR #Im,D_n ROR D_m,D_n	Циклічний зсув праворуч	
ROXL #Im,D_n ROXL D_m,D_n	Циклічний зсув ліворуч через прапорець *X*	
ROXR #Im,D_n ROXR D_m,D_n	Циклічний зсув праворуч через прапорець *X*	

Основним принципом отримання ефективних програм є ефективний розподіл та використання ресурсів процесора. Оптимізація програми реалізується за рахунок мінімізації пересилань, використання вказівників на дані замість самих даних при роботі зі складними структурами, розміщення структури даних в одній локальній області пам'яті, розподіл регістрів для локальної ділянки програми тощо.

Таблиця 2.5 – Команди операцій над окремими бітами

Синтаксис Асемблера	Розрядність	Операції	Адресування
BTST Dn,<EA> BTST #Nb, <EA>	*B, L*	$\overline{b}_n \to Z$	*Dn* – регістрове, *Nb* – безпосереднє, операнда – усі види
BSET Dn, <EA> BSET #Nb, <EA>	*B, L*	$\overline{b}_n \to Z$, $1 \to b_n$	*Dn* – регістрове, *Nb* – безпосереднє, операнда – регістрове, непряме регістрове, відносне, відносне з індексуванням
BCLR Dn, <EA> BCLR #Nb, <EA>	*B, L*	$\overline{b}_n \to Z$, $0 \to b_n$	*Dn* – регістрове, *Nb* – безпосереднє, операнда – регістрове, непряме регістрове, відносне, відносне з індексуванням
BCHG Dn, <EA> BCHG #Nb, <EA>	*B, L*	$\overline{b}_n \to Z$, $\overline{b}_n \to b_n$	*Dn* – регістрове, *Nb* – безпосереднє, операнда – регістрове, непряме регістрове, відносне, відносне з індексуванням

2.9 Команди умовних та безумовних переходів. Побудова розгалужених і циклічних програм

До операцій керування (передавання керування) відносяться команди за допомогою яких здійснюється зміни порядку виконання команд програми. Це команди умовних і безумовних переходів, звернення до підпрограм і повернення з підпрограми до основної програми, а також команди організації програмних циклів. Основні види команд передавання управління наведено в табл. 2.6.

Таблиця 2.6 – Основні види команд передавання керування

Синтаксис Асемблера	Операція	Адресування	Опис операції
1	2	3	4
JMP <EA>	<EA> → PC	Непряме регістрове (усі види), пряме (коротке та довге), відносне, відносне з індексуванням	Безумовний перехід до команди, яку адресовано <EA>
JSR <EA>	SP – 4 → SP, PC → (SP), <EA> → PC	Непряме регістрове (усі види), пряме (коротке та довге), відносне, відносне з індексуванням	Команда звернення до підпрограми, розміщення якої адресовано за допомогою <EA>
RTS	(SP) → SP, SP + 4 → SP		Повернення з підпрограми до основної підпрограми
Bcc <EA>	Якщо (cc) виконується, то <EA> → PC	Непряме регістрове (усі види), пряме (коротке та довге), відносне, відносне з індексуванням	Умовний перехід. Якщо умова виконується, то програма переходить на команду яку адресовано <EA>
DBcc Dn <EA>	Якщо (cc) не виконується, то Dn–1→Dn; якщо Dn ≠ -1, то <EA> → PC	Непряме регістрове (усі види), пряме (коротке та довге), відносне, відносне з індексуванням	Команда організації програмних циклів. Програма повертається у цикл, доки значення *Dn* не зрівняється з -1

Як умови розгалуження у команді Bcc використовуються 14 різних значень ознак *N, Z, C, V* та їхніх комбінацій. Види умов, які використовуються у команді Bcc наведено у табл. 2.7. Ці символи використовуються в команді замість символів cc (наприклад BNE).

Умови *T* та *F* використовуються у складі **команд DBF** – безумовне виконання заданої кількості циклів та **DBT** – безумовний вихід з циклу.

Для організації циклів використовується **команда DB_{cc}**. Зазначений у команді регістр *Dn* є лічильником циклів у цьому циклі.

Таблиця 2.7 – Види умов, які використовуються у командах

Символи мови	Умова, що перевіряється	Значення ознак
NE	Не дорівнює (ненульовий результат)	$Z = 0$
EQ	Дорівнює (нульовий результат)	$Z = 1$
HI	Вище	$C + N = 0$
LS	Нижче або дорівнює	$C + N = 1$
HS (або CC)	Вище або дорівнює (перенесення немає)	$C = 0$
LO (або CS)	Нижче (перенесення є)	$C = 1$
GE	Більше або дорівнює	$N \oplus V = 0$
LT	Менше	$N \oplus V = 1$
PL	Результат додатний	$N = 0$
MI	Результат від'ємний	$N = 1$
GT	Більше	$Z + (N \oplus V) = 0$
LE	Менше або дорівнює	$Z + (N \oplus V) = 1$
VC	Переповнення немає	$V = 0$
VS	Переповнення є	$V = 1$
T	Розгалуження є	1
F	Розгалуження немає	0

При виконанні команди DB_{cc} спочатку перевіряється виконання умови, заданої у команді. Якщо умова виконується, то МП вибирає наступну команду програми (умовний вихід з циклу). Якщо ж умова не виконується, то вміст регістру Dn декрементується. Якщо при цьому вміст регістру *Dn* становить –1, то також обирається наступна команда (цикл завершується). Якщо ж вміст *Dn* не дорівнює –1, то виконується перехід до команди з адресою (PC + Ds), яка є початком циклу. Команда DB_{cc} припускає використовування будь-яких умов, зазначених у табл. 2.7. Команда організації циклу DBF зреалізовує безумовне виконання заданої кількості циклів, а команда DBT – безумовний вихід з циклу.

У багатьох випадках розгалуження програм виконується залежно від результату порівняння двох операндів за допомогою команди порівняння знакових та беззнакових чисел: CMP, CMPA, CMPI, CMPM, TST та TAS. Ці команди використовуються для неруйнівного порівняння операндів. При цьому різниця не формується, а лише формуються ознаки цього результату. Команди порівняння наведено у табл. 2.8.

2.10 Типові мікроконтролери (МК) фірми “Моторола”: HC05, HC08, HC11. Структура. Вбудовані засоби МК

Фірма Motorola є провідним виробником мікроконтролерів (МК) у світі. Вона випускає 8-розрядні мікроконтролери, 16-, 32-розрядні модульні комунікаційні мікроконтролери та 32-розрядні *RISC* мікроконтролери *PowerPC*.

8-розрядні МК є найбільш поширеними представниками мікропроцесорної техніки. Їхня вартість не є високою, а досить широкий спектр функціональних можливостей дозволяє використовувати ці контролери у різноманітних пристроях та приладах. Усі 8-розрядні МК входять до складу трьох сімейств – 68HC05, 68HC08 та 68HC11.

Таблиця 2.8 – Команди порівняння операндів

Синтаксис Асемблера	Розряд-ність	Операції	Адресування
CMP <EA>,Dn	*B, W, L*	Dn – <EA>	<*scr*> – усі, *Dn* – регістрове
CMPA <EA>,An	*W, L*	An – <EA>	<*scr*> – усі, *An* – регістрове
CMPI #Im,<EA>	*B, W, L*	<EA> – Im	Im – безпосереднє, <EA> – регістрове, непряме регістрове з усіма модифікаціями або пряме, коротке чи довге
CMPM (Ay)+,(Ax)+	*B, W, L*	<*dst*> – <*scr*>	Обидва операнди подаються з постінкрементуванням
TST <EA>	*B, W, L*	<EA> – 0	Регістрове або непряме регістрове з усіма модифікаціями

МК M68HC05 є найбільш прості й, відповідно, такі, що мають обмежені функціональні можливості, але є досить дешевими. До складу цього сімейства входять різні типи мікроконтролерів, які відрізнюються об’ємом та типом пам’яті, номенклатурою периферійних пристроїв, що їх розміщено на кристалі, а також іншими експлуатаційними характеристиками.

МК сімейства 68HC08 мають більшу продуктивність, більший обсяг внутрішньої пам’яті та розширені функціональні можливості.

На кристалі МК сімейства 68HC11 зреалізовано велику номенклатуру периферійних пристроїв, вони також забезпечують можливість розширення об’єму пам’яті за рахунок підключення зовнішніх запам’ятовувальних пристроїв.

Загальна кількість різних моделей 8-розрядних МК, які сьогодні продукуються, становить близько 100.

Сімейство M68HC05/705

Усі моделі найбільш поширеного сімейства 8-розрядних МК M68HC05/705 мають однакове процесорне ядро CPU05 і відрізняються за наявністю, об’ємом та типом пам’яті, яка входить до їхнього складу, певними

характеристиками, а також номенклатурою периферійних пристроїв, розміщених на кристалі. Загальна кількість 8-розрядних контролерів, рекомендованих сьогодні до використовування, становить близько 40 моделей.

Мікроконтролери M68HC05 та M68HC705 відрізняються лише наявністю у складі останнього репрограмованого ПЗП з електричним стиранням – РПЗП-ЕС (EEPROM).

Загальну структурну схему МК сімейства M68HC05/705 наведено на рис. 2.18.

До складу всіх 8-розрядних контролерів входять: процесор CPU05, блок програмування, блок контролю функціонування, генератор тактової частоти і блок внутрішнього запам'ятовувального пристрою. Склад паралельних портів і периферійних пристроїв відрізняється у МК різних типів. Блок внутрішнього запам'ятовувального пристрою складається з оперативного запам'ятовувального пристрою даних, об'ємом від 32 до 920 байт (для різних моделей), постійного запам'ятовувального пристрою програм, об'ємом до 32 кбайт та службового постійного запам'ятовувального пристрою, об'ємом 240 байт. До складу деяких моделей входять репрограмовані ПЗП з електричним стиранням – РПЗП-ЕС (*EEPROM*), об'ємом до 920 байт.

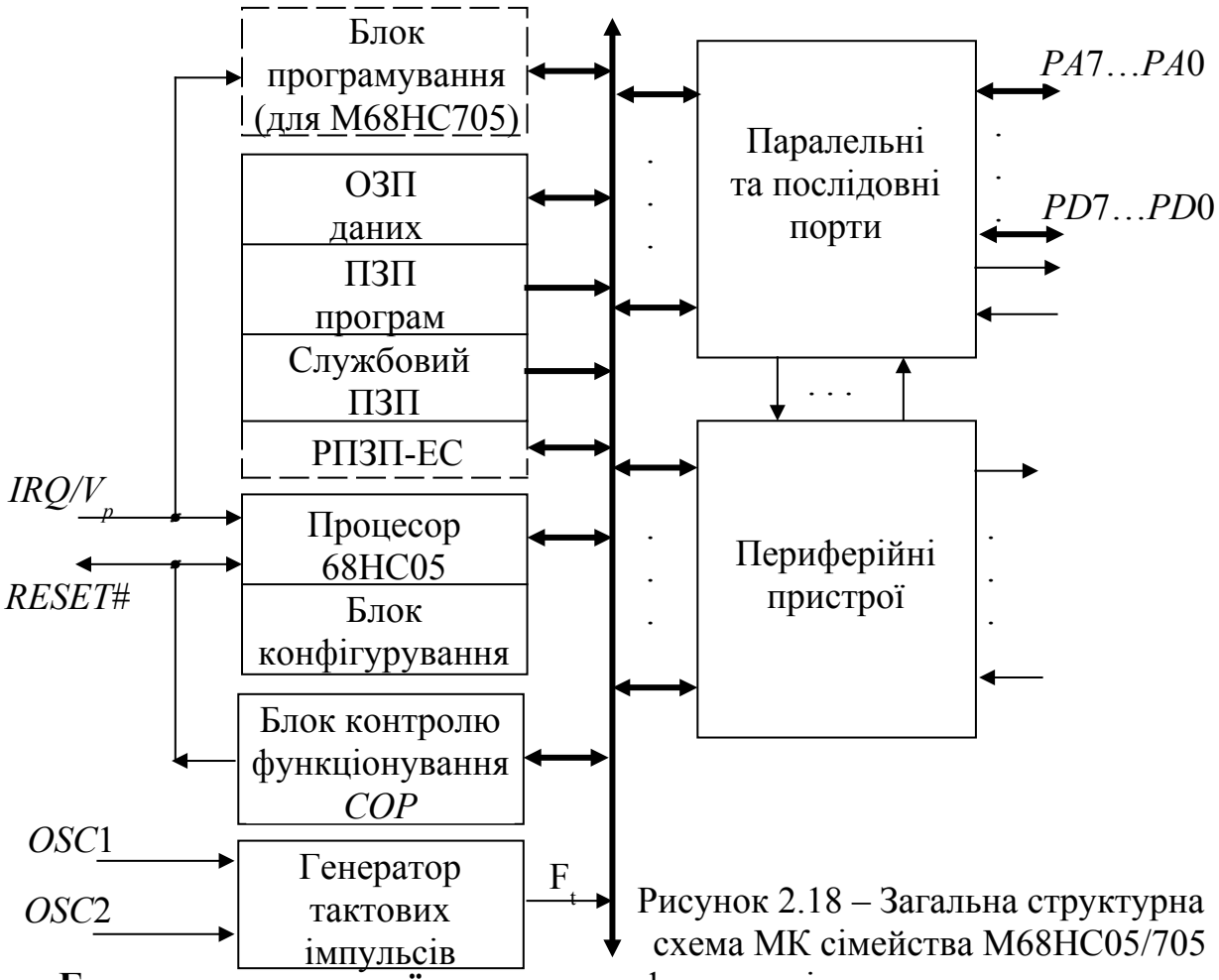


Рисунок 2.18 – Загальна структурна схема МК сімейства M68HC05/705

Генератор тактової частоти формує імпульси, частота яких визначається кварцовим або керамічним резонатором, *RC*-ланцюгом або зовнішнім генератором, які підключаються до виводів *OSC*1, *OSC*2. Частота імпульсів тактової частоти МК для більшості моделей становить F_t = 2,1 МГц за

напруги живлення $V_ж$ = 5 В і F_t = 1,0 МГц за $V_ж$ = 3,3 В. Деякі моделі мають підвищену тактову частоту F_t = 3,0 або 4,0 МГц за $V_ж$ = 5 В.

Блок програмування використовується для завантаження інформації до РПЗП-ЕС. Завантаження здійснюється через асинхронний послідовний порт відповідно до програми завантаження, яка зберігається у службовому ПЗП. При цьому здійснюється контроль правильності запису кожного байта.

Блок конфігурування використовується для визначення режиму роботи пристроїв, які входять до складу МК.

Блок контролю функціонування забезпечує контроль за виконанням програми за допомогою вартового таймера *WDT* (*Watch-Dog Timer*), а також контроль частоти імпульсів тактової синхронізації.

Блок паралельних і послідовних портів, у різних моделях МК вміщує від двох до чотирьох 8-розрядних паралельних портів, усі виводи яких можуть використовуватися для двоспрямованого обміну. В деяких моделях окремі виводи може бути призначено лише для виведення або введення, а також для приймання сигналів запитів переривань, входів аналого-цифрового перетворювача, входу таймера і для організації виводів синхронних та асинхронних портів.

Для послідовного обміну використовуються асинхронні порти *SCI* (*Serial Communication Interface*) або *SCI+* і синхронні порти *SPI* (*Serial Peripheral Interface*) або *SIOP* (*Simple Synchronous Serial Input-Output Port*) або *SSPI* (*Simple Serial Peripheral Interface*). Деякі серії мають у своєму складі спеціалізовані послідовні порти, призначені для організації мікроконтролерних мереж. Такі МК широко використовуються у складі пристроїв автоматики, вимірювальної апаратури тощо.

До складу **блока периферійних пристроїв** можуть входити таймери, широтно-імпульсний модулятор, спеціальні вихідні схеми для підключення пристроїв індикації тощо.

МК сімейства M68HC05/705 можуть працювати у двох режимах зі зниженим енергоспоживанням: очікування і зупину.

У режимі очікування (*Wait mode*) зупиняється робота процесора, але продовжується робота послідовних портів, таймерів, інших периферійних пристроїв та блока контролю функціонування. Перехід до цього режиму виконується при надходженні команди WAIT. Повернення до робочого режиму здійснюється при надходженні зовнішнього запиту переривання, сигналів переривання від таймера, сигналу запуску від блока контролю функціонування і зовнішнього сигналу *RESET#*. Повернення відбувається за один період тактової частоти. Залежно від джерела надходження сигналу повернення до робочого режиму до програмного лічильника PC буде завантажено або адресу відповідного вектора переривання, або вектор початкового завантаження. В цьому режимі споживана потужність зменшується в кілька разів.

Перехід до режиму зупину відбувається при надходженні команди STOP. У цьому режимі припиняється робота генератора тактових імпульсів, процесора, послідовних портів, таймерів, інших периферійних пристроїв та блока контролю функціонування. Вихід з цього режиму відбувається при надходженні зовнішнього запиту на переривання (на вхід *IRQ#* або на вивід МК, який запрограмовано як вхід сигналів переривань) або зовнішнього

сигналу *RESET#*. До програмного лічильника PC буде завантажена адреса зовнішнього переривання або вектор початкового завантаження. Повернення до робочого режиму відбувається за час, що дорівнює $4064T_t$.

При переході до режиму зниженого енергоспоживання у регістрі ознак CCR встановлюється значення ознаки $I = 0$, щоб забезпечити дозвіл переривання і повернення до робочого режиму.

У деяких моделях МК сімейства передбачено режим неповного зупину (*Halt mode*), який відрізнюється від режиму очікування лише тим, що генератор тактових імпульсів відключається від процесора, що зменшує споживану потужність. Для повернення до робочого режиму в такому разі потрібен більший час, щоб забезпечити встановлення заданих параметрів тактових імпульсів.

Регістрову модель процесора CPU05 подано на рис. 2.19.

Процесор виконує оброблення 8-розрядних операндів і зреалізовує 65 команд. До складу регістрової моделі входять:

– 8-розрядний акумулятор, використовуваний для зберігання одного з операндів та результату виконання арифметичних або логічних операцій;

– 8-розрядний індексний регістр *X*, використовуваний для формування адреси при індексному адресуванні. Цей регістр також є джерелом одного з операндів при виконанні операції множення, а також може використовуватися задля тимчасового зберігання операндів та результатів;

– 8-розрядний регістр прапорців *CCR*, який зберігає значення п’яти ознак результату, як показано на рис. 2.19:

− *C* (*Carry/Borrow Flag*)– ознака перенесення (набирає значення $C = 1$, якщо при виконуванні операції відбувається перенесення зі старшого розряду результату);

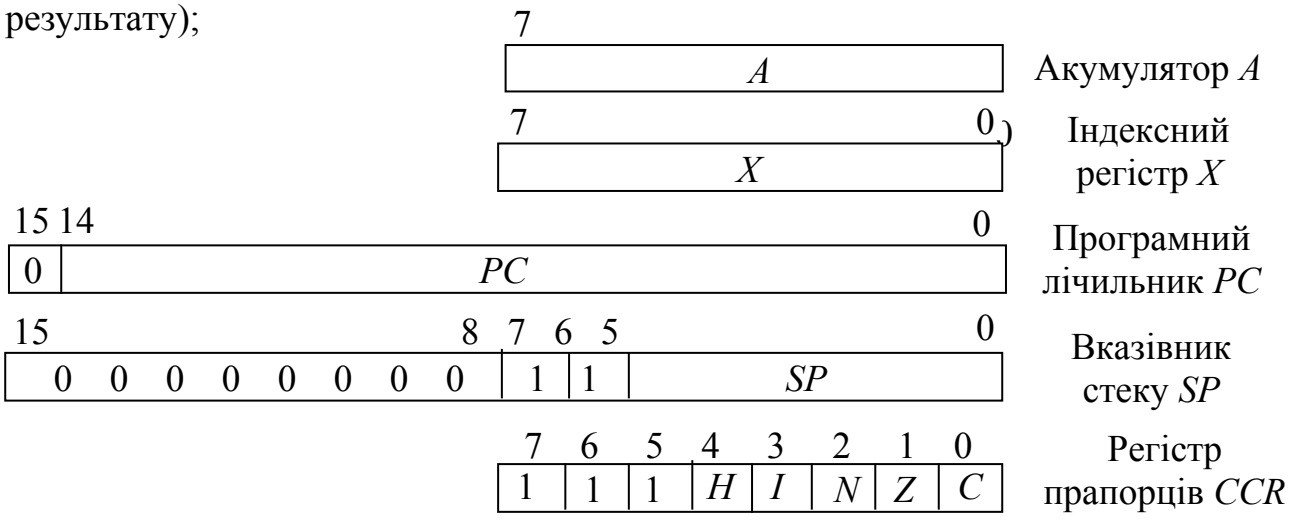


Рисунок 2.19 – Регістрова модель процесора CPU05

– *Z* (*Zero Flag*) – ознака нульового результату ($Z = 1$, якщо результат операції дорівнює нулю);

– *N* (*Negative Flag*) – ознака знаку результату ($N = 0$, якщо результат додатний, та $N = 1$ за від’ємного результату);

– *I* (*Interrupt Mask*) – ознака дозволу переривання ($I = 0$ – обробляння переривань дозволено, $I = 0$ – заборонено);

– *H* (*Half-Carry Flag*) – допоміжне (міжтетрадне) перенесення. Використовується при виконуванні операцій над двійково-десятковими числами.

– 16-розрядний програмний лічильник *PC* вміщує адресу поточної виконуваної команди. Максимальний об'єм, що може адресуватися, становить 32 кбайти (розряди 0...14). При включенні МК (виконання команди *RESET*) до програмного лічильника автоматично завантажується адреса першої команди програми (вектор початкового завантаження) з двох останніх комірок пам'яті ($FFFE...$FFFF);

– 16-розрядний вказівник стека *SP*, який використовується для роботи зі стеком. Стек в МК сімейства M68HC05/705 має обсяг 64 байти і розміщується наприкінці молодших 256 байтів ОЗП даних, тому для адресування використовується лише 6 молодших розрядів регістру *SP*, куди при запуску МК автоматично завантажується $00FF.

Блок конфігурування, призначений для визначення певних характеристик і режимів роботи МК і складається з регістрів конфігурування. Ці регістри в МК підсімейства 68HC05 є комірками пам'яті ПЗП, програмованими маскою на етапі виготовлення мікросхеми, тому їхній вміст при експлуатації не змінюється. В МК підсімейства 68HC705 ці регістри є комірками одноразово програмованих ПЗП, які програмуються при першому включенні мікросхеми і за подальшої роботи не змінюються.

Блок контролю функціонування складається з блоків контролю функціонування *COP* (*Computer Operating Property*), до складу яких входить вартовий таймер, а також монітор тактових імпульсів (у деяких моделях).

Вартовий таймер блока контролю функціонування (*WDT*) МК сімейства M68HC05/705 контролює правильність виконання програми. Правильність виконування перевіряється за результатами запису певних значень до відповідних регістрів. Якщо такий запис не виконується за певний час T_w, то блок контролю формує сигнал *RESET#* – і відбувається перезавантаження контролера. Період контролю визначається при конфігуруванні МК.

Блок паралельних і послідовних портів МК MC68HC705J1A вміщує лише два паралельних порти *A* та *B* з можливих п'яти, передбачених у складі блока паралельних та послідовних портів сімейства M68HC05/705.

Вісім виводів порту *A* МК MC68HC705J1A може бути запрограмовано на введення/виведення даних у паралельному форматі або чотири з них можуть використовуватися в якості входів для сигналів переривань. Порт *B* має лише шість виводів для організації обміну у паралельному форматі.

Робота портів організовується за допомогою регістрів даних *PORTA* та *PORTB*, які мають адреси відповідно $00 та $01, а також двох регістрів напрямку пересилання *DDRA* та *DDRB*. Регістри напрямку пересилання визначають режим роботи відповідних виводів, які після початкового завантаження запрограмовані на приймання даних (в регістрах *DDRx* записано 0 у всіх розрядах).

Асинхронний послідовний порт (*SCI*), який входить до складу МК типу MC68HC705C8A, забезпечує стандартний асинхронний формат приймання-

передавання даних у вигляді кадру, який вміщує один стартовий і один стоповий біти, вісім інформаційних бітів (*D*0...*D*7) і за потреби може доповнюватися одним додатковим контрольним бітом (*D*8*). Формат кадру подано на рис. 2.20.

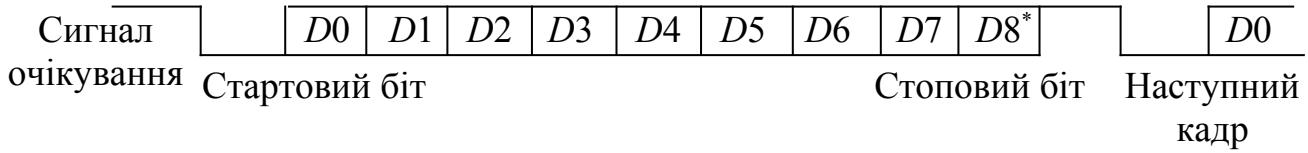


Рисунок 2.20 – Формат кадру даних порту *SCI*

Синхронний послідовний порт (*SPI*), який входить до складу порту *D* МК MC68HC705C8A, складається з регістру даних *SPDR* (адреса $0C), регістру керування SPCR (адреса $0A) і регістру стану *SPSR* (адреса $0B).

Обмін здійснюється 8-розрядними даними, які зчитуються з регістру даних SPDR або записуються до нього.

Підключення пристроїв для здійснення синхронного обміну виконується відповідно до схеми, яка наведена на рис. 2.21.

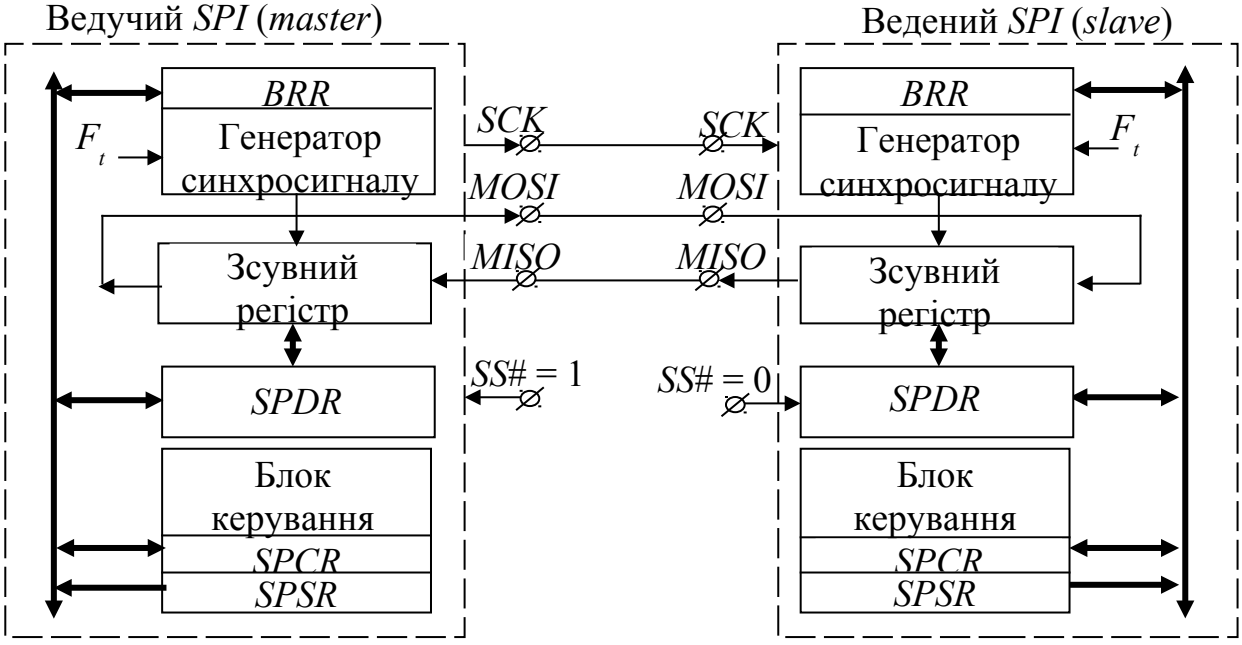


Рисунок 2.21 – Організація синхронного обміну даними через порт *SPI*

Обмін проводиться по лініях *MOSI* або *MISO*, і в кожному такті здійснюється запис одного біта до молодшого розряду зсувного регістру і вивід одного біта зі старшого. Обмін проводиться байтами, які зчитуються або записуються до регістру *SPDR*.

Сімейство M68HC08/908

Сімейство M68HC08/908 є подальшим розвиненням МК M68HC05/705. Вони зберігають їхні архітектурні особливості, однак дозволяють підвищувати техніко-економічні характеристики пристроїв, до складу яких вони входять. До

складу МК підсімейства M68HC908 входить Flash-пам’ять, що дозволяє широко використовувати його у пристроях, які випускаються малими серіями.

Програмна сумісність з МК M68HC05/705 дозволяє використовувати програмне забезпечення, розроблене для них для програмного керування новими пристроями.

До складу МК сімейства M68HC08/908 входять: процесорне ядро CPU08; внутрішня пам’ять програм – ПЗП, програмований маскою, об’ємом до 32 кбайт або *Flash*-пам’ять, об’єм якої становить 60 кбайт; ОЗП даних, який має ємність від 128 байт до 2 кбайт. У деяких моделях є РПЗП-ЕС об’ємом 512 байт або 1 кбайт.

Залежно від функціонального призначення МК, до їхнього складу може входити від 5-ти до 8-ми паралельних портів, послідовні порти *SCI* та *SPI*. До складу МК деяких серій входять спеціалізовані послідовні порти, призначені для організації мікроконтролерних мереж.

До складу периферійних пристроїв усіх МК сімейства M68HC08/908 входять 16-розрядні таймери. Більшість моделей вміщує 8- або 10-розрядні АЦП.

Характерною особливістю побудови МК цього сімейства є побудова МК зі стандартним набором модулів (модульний принцип). Поєднання окремих модулів на одному кристалі надає змогу формувати МК різноманітного функціонального призначення.

Загальну структурну схему МК цього сімейства наведено на рис. 2.22. До складу цієї схеми входять стандартні модулі, кожен з яких має власне функціональне призначення.

Модуль формування тактових сигналів *CGM*08 (*Clock Generator Module*) призначено для формування послідовностей імпульсів, потрібних для роботи процесора та периферійних модулів.

Модуль *CG*M08 вміщує два генератори імпульсів: *CG*, який формує послідовність імпульсів, частота яких F_q визначається зовнішнім кварцовим резонатором, і генератор *PG* – частота сигналу якого $F_p = N \cdot F_q$ визначається роботою схеми фазового автоналаштування частоти.

Регістр *PCTL* – регістр керування модулем *CG*M08.

Регістр *PBWC* – регістр керування формуванням частоти.

Регістр *PPG* – регістр задавання коефіцієнтів, які визначають часові характеристики сигналів

Модуль системної інтеграції *SIM*08 виконує початкове завантаження МК при підмиканні живлення та перезавантаження при надходженні зовнішнього сигналу скидання *RST#* або сигналу від модуля контролю функціонування *COP*08, або при формуванні помилкового коду команди або зверненні до неіснуючої адреси.

Регістр *SRSR* (адреса $FE01) – регістр стану модуля системної інтеграції *SIM*08. Його вміст дозволяє встановити причину запуску МК.

Регістр *SBSR* (адреса $FE00) – регістр коригування адреси повернення з підпрограми обслуговування переривань.

Регістр *SBFCR* (адреса $FE03) – регістр дозволу зміни стану периферійних модулів.

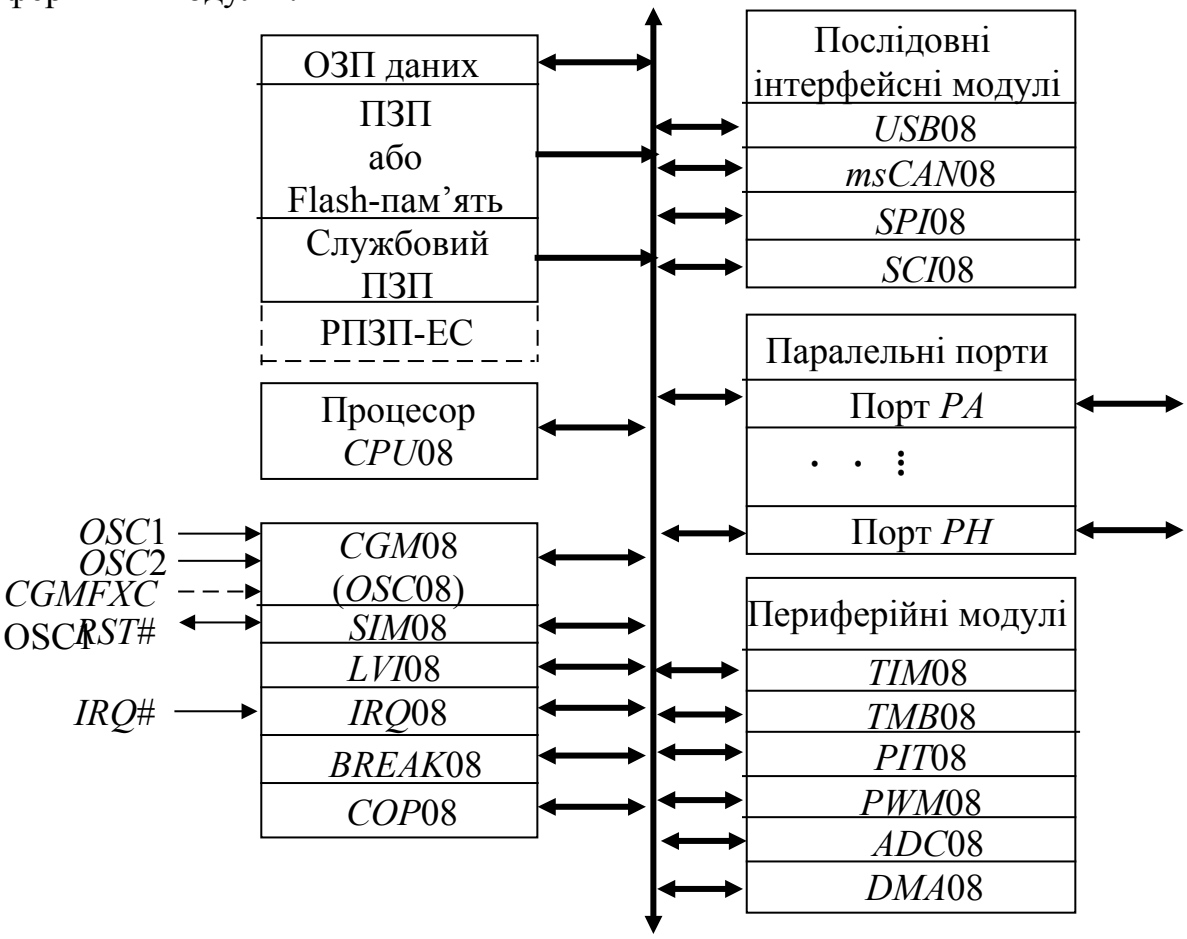


Рисунок 2.22 – Загальна структурна схема МК сімейства M68HC08/908

Модуль керування зовнішнім перериванням *IRQ*08 обслуговує зовнішні запити переривання, які надходять на вхід *IRQ#*, відповідно до вибраного режиму.

Модуль переривання у контрольній точці *BREAK*08 використовується при налагодженні програмного забезпечення для реалізації режиму зупину у контрольній точці.

Модуль контролю напруги живлення *LVI*08 використовується для контролю напруги живлення. Якщо при роботі МК напруга живлення знижується понад задане значення, то модуль контролю напруги живлення *LVI*08 переводить МК до початкового стану, який зберігається до встановлення нормального рівня напруги.

Модуль контролю функціонування *COP*08 контролює виконування програми за допомогою вартового таймера.

До складу послідовних інтерфейсних модулів входять:

Модуль інтерфейса з послідовною шиною *USB*08 забезпечує обмін даних по шині *USB*, яка використовується в обчислювальній техніці.

Модуль послідовного інтерфейсу *msCAN*08 забезпечує обмін даними в стандартному та розширеному форматах відповідно до специфікацій *CAN* 2.0 *A* та *CAN* 2.0 *B*.

Модуль синхронного периферійного інтерфейса *SPI*08 забезпечує синхронний обмін даними зі швидкістю до 4 Мбіт/с. Цей модуль використовується для організації зв'язку між МК та іншими пристроями на незначній відстані.

Модуль асинхронного інтерфейсу зв'язку *SCI*08 зреалізовує стандартний асинхронний протокол обміну 8- або 9-бітними даними з одним стартовим та одним стоповим бітами зі швидкістю до 130 кбіт/с.

Блок паралельних портів вміщує від 2-х до 8-ми паралельних 8-розрядних портів (*PA*, *PB*, ..., *PG*, *PH*).

До блока периферійних модулів для різних серій можуть входити:

Таймерний модуль *TIM*08 призначений для формування часових інтервалів, потрібних для керування МК та іншими пристроями, які входять до складу системи, в цілому. Він побудований на базі 16-розрядного лічильника.

Модуль базового таймера *TBM*08 забезпечує періодичне формування сигналів запиту переривання при переповненні базового лічильника. Він вміщує 15-розрядний лічильник, на вхід якого надходять сигнали кварцового резонатора. В режимі очікування цей таймер продовжує функціонувати і забезпечує перехід МК до робочого режиму.

Модуль таймера періодичних переривань *PIT*08 використовується для періодичного формування сигналів запиту переривання.

Модуль широтно-імпульсного модулятора *PWM*08 вміщує 12-розрядний 6-канальний ШІМ модулятор.

Модуль аналого-цифрового перетворення *ADC*08 використовується для аналого-цифрового перетворення вхідних аналогових сигналів. У більшості моделей зреалізовано 8-розрядне перетворення аналогового сигналу, а в деяких моделях – 10-розрядне. Кількість аналогових входів у різних моделях може становити від 4 до 15.

Модуль прямого доступу до пам'яті *DMA*08 забезпечує роботу зі швидкодіючими зовнішніми пристроями.

Процесорний модуль *CPU*08 є модифікованим варіантом процесора CPU05. Він має розширений набір команд та способів адресування і програмно цілковито є сумісний з *CPU*05. Регістрову модель процесора CPU08 подано на рис. 2.23.

До регістрової моделі входять:

– 8-розрядний акумулятор *A*;

– 16-розрядний індексний регістр *H:C*. Для забезпечення програмної сумісності з *CPU*05 індексний регістр складається з двох частин, роздільне використовування яких дозволяє забезпечувати функціонування програм МК сімейства M68HC05/705;

– 16-розрядний програмний лічильник *PC*;

– 16-розрядний вказівник стека *SP*;

– 8-розрядний регістр прапорців *CCR*. Вміщує такі ознаки, які формуються у процесорі *CPU*05. До них додано ознаку *V* – ознаку переповнювання при обробленні чисел зі знаком. Ознака встановлюється *V* = 1,

якщо результат має кількість розрядів більшу, ніж можна розмістити їх в розрядній сітці МК.

– Обсяг адресного простору, який може адресувати МК сімейства M68HC08/908, становить 64 кбайти. Деякі МК сімейства мають менший обсяг адресного простору, тому в їхньому адресному просторі є області адрес, які не відповідають певним пристроям. При зверненні до таких адрес реалізовується переривання, яке відповідає наявності помилки у програмі – зверненні до неіснуючої комірки. Внаслідок цього відбувається перезавантаження МК, до програмного лічильника PC автоматично завантажується адреса першої команди оброблення переривання з двох останніх комірок адресного простору ($FFFE – $FFFF).

Рисунок 2.23 – Регістрова модель процесора CPU08

При перезавантаженні МК у вказівник стека автоматично записується адреса вершини стека ($00FF). При роботі користувач має змогу замінити цю адресу вершини стека на іншу в межах адресного простору ОЗП.

Сімейство MC68HC11/711.Найбільш досконалим сімейством 8-розрядних МК є MC68HC11/711. До складу структурної схеми МК входять модулі, які за функціональним призначенням є аналогічні до описаних вище. Різні моделі сімейства мають однакове процесорне ядро і відрізнюються обсягом і типом внутрішнього запам’ятовувального пристрою, складом периферійного обладнання й деякими іншими характеристиками. Особливістю цього сімейства є можливість підключення зовнішньої пам’яті об’ємом від 64 кбайт до 4 Мбайт.

МК сімейства MC68HC11/711 у своєму складі мають внутрішню пам’ять програм – ПЗП – у складі MC68HC11 або РПЗП – у MC68HC711, об’ємом до 32 кбайт; ОЗП даних об’ємом від 192 до 1024 байт. Деякі моделі мають внутрішній РПЗП-ЕС об’ємом до 540 байт.

При використовуванні МК цього сімейства можна організовувати роботу в двох режимах – автономному і розширеному. При роботі в автономному режимі використовується лише внутрішня пам’ять МК, а в розширеному дозволяється підключення зовнішньої пам’яті, робота з якою відбувається за допомогою мультиплексованої або окремої зовнішньої шини адреси/даних.

Усі моделі мають у своєму складі 16-розрядний таймер, який може мати 3-4 входи сигналів захоплення частоти (IC) і 4-5 входів сигналів збігу частот.

Цей таймер також використовується для формування сигналів періодичних переривань. Крім таймера, до складу МК входить 8-розрядний лічильник зовнішніх подій. Деякі МК вміщують 8-розрядні широтно-імпульсні модулятори.

МК сімейства вміщують від 4-х до 10-и паралельних 8-розрядних паралельних портів, асинхронний і синхронний послідовні порти *SCI* (*SCI*+) і *SPI*.

До складу більшості моделей входить 8-розрядний АЦП з 8 аналоговими входами.

Процесорне ядро сімейства складається з процесора 68HC11, регістрову модель якого зображено на рис. 2.24.

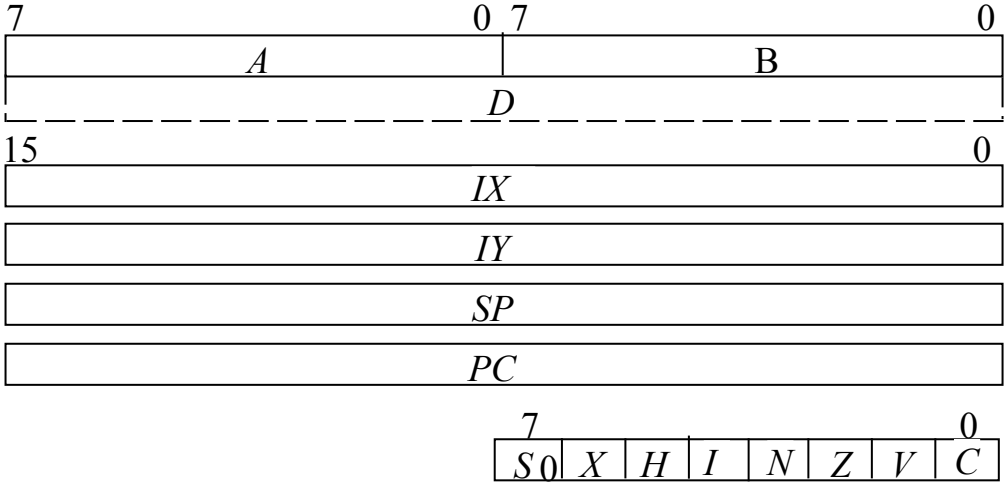


Рисунок 2.24 – Регістрова модель процесора 68HC11

До складу регістрової моделі процесора 68HC11 входять:

– два 8-розрядні акумулятори *A* і *B*, які при виконанні деяких команд об’єднуються у 16-розрядний регістр *D*;

– два 16-розрядні індексні регістри *IX* і *IY*, які використовуються для формування адреси при індексному адресуванні операндів;

– 16-розрядний вказівник стека *SP*;

– 16-розрядний програмний лічильник *PC*;

– 8-розрядний регістр прапорців *CCR*, який вміщує ознаки *H, I, N, Z, V* і *C*, призначення й використовування яких є аналогічні до 8-розрядних МК. До складу цього регістру до *CPU*12 додатково введено ознаку *X*, встановлення якої *X* = 1 забороняє обслуговування зовнішнього запиту переривання по входу *XIRQ*#, і ознаку *S*, яка при встановленні *S* = 1 забороняє переключення МК до режиму зупину при надходженні команди STOP.

Процесор виконує оброблення 8- і 16-розрядних операндів і реалізовує 108 команд. При роботі з операндами він використовує такі способи адресування як сімейство 68HC05/705; більшість команд, виконуваних МК, є аналогічні до команд сімейства 68HC05/705. Відмінності зумовлено наявністю двох акумуляторів. Додатково до системи команд введено команди ділення і розширено можливості команд бітових операцій.

2.11 Система команд мови Асемблер 8-розрядних МК

Мікроконтролери цього сімейства можуть реалізовувати набір з 65-ти базоваих команд, які виконують оброблення 8-розрядних операндів. Операнди можуть розміщуватися в регістрах *A, X* та пам'яті. Команди мають розмір від одного до трьох байтів: у першому байті розміщується код операції, а другий та третій забезпечують адресування операндів, тобто можуть бути команди, які не вміщують операндів. МК реалізовує такі способи адресування:

– регістрове – операнд розміщується в регістрі *A* або *X*;

– непряме регістрове (індексне) – за адресу операнда слугує вміст регістру *X*;

– індексне зі зміщенням – адреса операнда є сумою вмісту регістру *X* та 8- або 16-розрядного зміщення (число без знаку), що задано в другому та третьому байтах команди;

– пряме – адреса операнда визначається значеннями другого та третього байтів команди;

– безпосереднє – 8-розрядний операнд (число) вказується у другому байті команди;

– відносне – використовується лише в командах розгалуження. Адреса наступної команди визначається як сума поточного вмісту програмного лічильника *PC* і 8-розрядного зміщення (число зі знаком), яке розміщено у другому байті команди.

Усі команди, що входять до системи команд сімейства M68HC05/705, можна поділити на групи відповідно до операцій, які вони виконують.

До **команд пересилань** належать команди, які виконують завантаження операндів з пам'яті до регістрів *A, X* або завантажування комірок пам'яті з цих регістрів, пересилання операндів між регістрами *A* та *X*, а також обнулення вмісту регістрів *A, X* і комірок пам'яті, що адресуються короткими способами адресування. Перелік команд пересилань і способів подання операндів подано у табл. 2.9.

До **команд арифметичних операцій** належать команди, які виконують відповідні арифметичні операції над 8-розрядними операндами, один з яких розміщується в акумуляторі *A*, а другий міститься у комірці пам'яті, адреса якої визначається способом адресування в команді. Результат виконання операції неодмінно записується до акумулятора. Операції додавання та віднімання при виконанні можуть використовувати значення біта перенесення *C*. До таких операцій також належать команди інкрементування та декрементування, котрі, відповідно, зреалізовують додавання (віднімання) 1 до (від) вмісту акумулятора, індексного регістру або комірки пам'яті.

Таблиця 2.9 – Перелік основних команд пересилань

Назва	Мнемокод	Операція	Спосіб адресування
Команди завантаження регістра *A*	LDA #opr	#opr → *A*	Безпосереднє
	LDA $addr	(*M*) → *A*	Пряме коротке
	LDA ,X	(*M*) → *A*	Непряме регістрове

Команди завантаження регістра *X*	LDX #opr	#opr → *X*	Безпосереднє
	LDX $addr	(*M*) → *X*	Пряме коротке
	LDX ,X	(*M*) → *X*	Непряме регістрове
Команди завантаження комірки пам'яті з регістра *A*	STA $addr	(*A*) → *M*	Пряме коротке
	STA ,X	(*A*) → *M*	Непряме регістрове
	STA $addr,X	(*A*) → *M*	Індексне зі зміщенням коротке
	STA $addr,X	(*A*) → *M*	Індексне зі зміщенням довге
Команди завантаження комірки пам'яті з регістра *X*	STX $addr	(*X*) → *M*	Пряме коротке
	STX $addr	(*X*) → *M*	Пряме довге
	STX ,X	(*X*) → *M*	Непряме регістрове
	STX $addr,X	(*X*) → M	Індексне зі зміщенням коротке
Пересилання вмісту регістра A в регістр *X*	TAX	(*A*) → *X*	Регістрове
Пересилання вмісту регістра X в регістр *A*	TXA	(*X*) → *A*	Регістрове
Обнулення вмісту регістра *A*	CLRA	$00 → *A*	Регістрове
Обнулення вмісту регістра *X*	XLRX	$00 → *X*	Регістрове
Обнулення вмісту комірки пам'яті	CLR $addr	$00 → *M*	Пряме коротке
	CLR, X	$00 → *M*	Непряме регістрове
	CLR $addr,X	$00 → *M*	Індексне зі зміщенням коротке

МК також зреалізовує операцію беззнакового множення 8-розрядних даних, які розміщуються в акумуляторі *A* та індексному регістрі *X*; результат множення записується до пари регістрів *X:A*, отже, старший байт неодмінно записується до регістру *X*.

До команд арифметичних операцій також належать команди порівняння і тестування операндів. Існують дві групи таких команд: порівняння вмісту акумулятора *A* і вмісту комірки пам'яті та вмісту регістру *X* і комірки пам'яті. Команди обох груп виконуються як віднімання від вмісту відповідного регістру вмісту комірки пам'яті, при цьому прапорці змінюються відповідно до результату, а результат не формується.

Тестування виконується для операндів, які можуть розміщуватися в пам'яті, акумуляторі та індексному регістрі. Їхнє виконання полягає у відніманні числа $00 від операнда. При цьому формуються ознаки, що відповідають самому операндові.

Команди зміни знаку (формування доповнювального коду) також належать до цієї групи. В них можуть використовуватися лише короткі способи адресування операндів.

Перелік команд арифметичних операцій подано у табл. 2.10.

Таблиця 2.10 – Арифметичні команди

Назва	Мнемокод	Операція	Спосіб адресування
Додавання з перенесенням	ADC opr	$(A)+(M)+C\rightarrow A$	Використовуються всі способи адресування і подання операндів
Додавання	ADD opr	$(A)+(M)\rightarrow A$	
Віднімання з позикою	SBC opr	$(A)-(M)-C\rightarrow A$	
Віднімання	SUB opr	$(A)-(M)\rightarrow A$	
Інкрементування комірки пам'яті	INC $addr	$(M)+1\rightarrow M$	Пряме коротке
	INC ,X		Непряме регістрове
Інкрементування регістру *A*	INCA	$(A)+1\rightarrow A$	Регістрове
Інкрементування регістру *X*	INCX	$(X)+1\rightarrow X$	Регістрове
Декрементування комірки пам'яті	DEC $addr	$(M)-1\rightarrow M$	Пряме коротке
	DEC ,X		Непряме регістрове
Декрементування регістру *A*	DECA	$(A)-1\rightarrow A$	Регістрове
Декрементування регістру *X*	DECX	$(X)-1\rightarrow X$	Регістрове
Беззнакове множення	MUL	$(X)\times(A)\rightarrow X{:}A$	
Порівняння операндів	CMP opr	$(A)-(M)$	Використовуються всі способи адресування і подання операндів
	CPX opr	$(X)-(M)$	
Тестування операндів	TST $addr	(M)–$00	Пряме коротке
	TST ,X		Непряме регістрове
	TST $addr,X		Індексне зі зміщенням коротке
	TSTA	$(A)-$00	Регістрове
	TSTX	$(X)-$00	Регістрове

До **логічних операцій,** які виконує МК M68HC05/705, належать команди логічного множення (кон'юнкції), логічного додавання (диз'юнкції), виключного АБО та логічного інвертування, які виконуються з 8-розрядними операндами. Команда бітового тестування, яка також належить до цієї групи, виконує логічне множення операндів, але не формує результату. Перелік команд логічних операцій подано у табл. 2.11.

До групи **команд зсувів,** які реалізовує МК M68HC05/705, належать команди арифметичних, логічних і циклічних зсувів, які виконуються над вмістом регістрів *A, X* та комірок пам'яті, що адресуються короткими способами адресування.

Команди бітових операцій, які входять до системи команд МК M68HC05/705, встановлюють необхідне значення (0 або 1) відповідного біта b_n у 8-розрядному операнді, адреса якого вміщується у другому байті команди (використовується лише пряме адресування). Номер біта *n* задається в команді.

До цієї групи команд також належать команди встановлення необхідного значення прапорців *C* та *I* в регістрі ознак *CCR*.

Таблиця 2.11 – Команди логічних операцій

Назва	Мнемокод	Операція	Спосіб адресування
Логічне множення	AND opr	$A \wedge (M) \rightarrow A$	Використовуються всі способи адресування і подання операндів
Логічне додавання	OR opr	$A \vee (M) \rightarrow A$	
Виключне АБО	EOR opr	$A \veebar (M) \rightarrow A$	
Логічна інверсія операнда	COM $addr	$FF–(M) \rightarrow M$	Пряме коротке
	COM ,X		Непряме регістрове
	COM $addr,X		Індексне зі зміщенням коротке
	COMA	$FF–(A) \rightarrow A$	Регістрове
	COMX	$FF–(X) \rightarrow X$	Регістрове

До групи **команд керування програмою** входять команди, які наведено у табл. 2.12.

Таблиця 2.12 – Команди керування програмою

Назва	Мнемокод	Операція
1	2	3
Безумовний перехід	JMP $addr	$(EA)_{корот.} \rightarrow PC$
	JMP ,X	$(X) \rightarrow PC$
	JMP $addr,X	$(X) + \$addr_{корот.} \rightarrow PC$
Умовний перехід	B_{CC} $rel8	$(PC)+2+\$rel8 \rightarrow PC$, якщо умова виконується

Закінчення табл. 2.12

1	2	3
Безумовне розгалуження	BRA $rel8	(*PC*)+2+$rel8 → *PC*
Відсутність розгалуження	BRN $rel8	(*PC*)+2 → *PC*
Розгалуження при встановленні відповідного біта n	BRSET n,$addr,rel8	(*PC*)+2+rel8 → *PC*, якщо $b_n = 1$
	BRCLR n,$addr,rel8	(*PC*)+2+rel8 → *PC*, якщо $b_n = 0$
Перехід до підпрограми	JSR $addr JSR ,X JSR $addr,X	(*PC*)+*n* → *PC* (*PCl*)→*SP*; *SP* – 1→*SP* (*PCh*)→*SP*; *SP* –1→*SP* (*EA*)→*PC*
Розгалуження до підпрограми	BSR $rel8	(*PC*)+2 → *PC* (*PCl*)→*SP*; *SP* – 1→*SP* (*PCh*)→*SP*; *SP* –1→*SP* (*PC*)+2+rel8→*PC*
Вихід з підпрограми	RTS	*SP*+1→*SP*; (*SP*)→*PCh* *SP*+1→*SP*; (*SP*)→*PCl*

Команда JMP завантажує до програмного лічильника адресу переходу, відповідно до способу адресування, що використовується.

У командах умовного переходу і розгалужень використовується лише відносний спосіб адресування. Адреса переходу формується як сума поточного вмісту програмного лічильника, збільшеного на 2 (для звернення до наступної команди в програмі, що виконується), і 8-розрядного зміщення ($rel8) відносно початкової адреси розміщення програми, яке є числом зі знаком. Умови, що перевіряються, наведено в табл. 2.13.

Таблиця 2.13 – Мнемокоди й умови виконування команд умовних переходів

Мнемокод умови CC	Умова, що перевіряється	Логічний вираз
NE	Не дорівнює (ненульовий результат)	$Z = 0$
EQ	Дорівнює (нульовий результат)	$Z = 1$
HI	Вище	$(Z + C) = 0$
LS	Нижче або дорівнює	$(Z + C) = 1$
HS	Вище або дорівнює (немає перенесення)	$C = 0$
LO	Нижче (є перенесення)	$C = 1$
PL	Додатний результат	$N = 0$
MI	Від’ємний результат	$N = 1$
HCC	Немає міжтетрадного перенесення	$H = 0$
HCS	Є міжтетрадне перенесення	$H = 1$
MC	Переривання дозволено	$I = 0$
MS	Переривання заборонено	$I = 1$
IH	Відсутність запиту переривання	*IRQ*# = 1
IL	Надходження запиту переривання	*IRQ*# = 0

Команда безумовного розгалуження BRA є аналогом команди безумовного переходу з використанням відносного адресування, а команда відсутності розгалуження BRN є аналогічна до команди NOP. Вона пропускає у програмі два байти, після чого триває виконання програми. Команду BRN зручно використовувати при налагодженні програм замість команди B_{CC} для виконання програми без розгалуження.

Команди розгалуження при встановленні відповідного біта *n* BRSET і BRCLR перевіряють значення цього біта в операнді, визначеному за допомогою 8-розрядного прямого адресування. Зміщення в цих командах ($rel8) визначається відносно початкової адреси розміщення програми.

Команди переходу й умовного переходу до підпрограм JSR та BSR зберігають у стеку адресу команди повернення до головної програми і завантажують до програмного лічильника PC початкову адресу підпрограм, яка визначається за допомогою прямого, індексного або індексного зі зміщенням адресування. При виконанні команди JSR адреса, що завантажується до стека, має бути більшою за вміст регістру PC на 1, 2 або 3, залежно від способу адресування. При використанні індексного адресування це 1, при індексному адресуванні з 8-розрядним зміщенням або короткому прямому – 2 і при адресуванні з 16-розрядним зміщенням і довгому прямому – 3. В команді BSR використовується відносне адресування. При поверненні до головної програми вміст регістру PC відновлюється зі стека.

До **команд керування процесором** належать команди, які подано у табл. 2.14.

Таблиця 2.14 – Команди керування процесором

Назва	Мнемокод	Операція
Встановлення початкового значення вказівника стека	RSP	$00FF→*SP*
Відсутність операцій	NOP	(*PC*)+1 → *PC*
Перехід до режиму очікування	WAIT	Зупинення процесора 0→ I
Перехід до режиму зупину	STOP	Зупинення генератора тактових імпульсів 0→ I

Команда встановлення початкового значення вказівника стека завантажує адресу $00FF до регістра SP.

Команди WAIT та STOP переводять МК до режиму енергозберігання з можливістю повернення до нормального режиму, для чого при їхньому виконуванні встановлюються значення ознаки дозволу переривань I = 0.

2.12 Використання МК для програмної реалізації пристроїв цифрової техніки

Виводи паралельних портів МК MC68HC705J1A можна використовувати для обміну будь-якими сигналами. При використанні окремих розрядів можна формувати на них сигнали у послідовному вигляді, тобто, МК можна використовувати в якості генератора послідовності імпульсів потрібної частоти і щільності. Блок-схему алгоритму функціонування МК MC68HC705J1A в якості генератора імпульсів подано на рис. 2.25.

Визначення напрямку обміну зумовлюється встановленням відповідного біта в регістрі напрямку пересилання *DDRA* (адреса $0004) або *DDRB* (адреса $0005). Вважаємо, що це будуть всі 8 виводів порту *A*.

Виведення здійснюється з регістра даних порту *PORTA*, до якого треба завантажити значення сигналу з акумулятора *A*. Виведення даних триватиме упродовж часу, поки сигнали на виходах порту не змінять свого значення. Тому треба сформувати часовий інтервал, який визначатиме час формування імпульсу або паузи між імпульсами. Цей часовий інтервал формується за допомогою підпрограми формування тривалості; вважаємо, що послідовність розпочинається з паузи (рівень логічного 0). Звичайно формування часових інтервалів може відбуватися за допомогою таймера або у програмний спосіб. При розв'язанні нашого завдання ця підпрограма може бути побудована за алгоритмом, який подано на рис. 2.26. На цьому рисунку не зазначено блоки входу та виходу, тому що вважається, що робота цієї підпрограми здійснюється згідно з алгоритмом рис. 2.25.

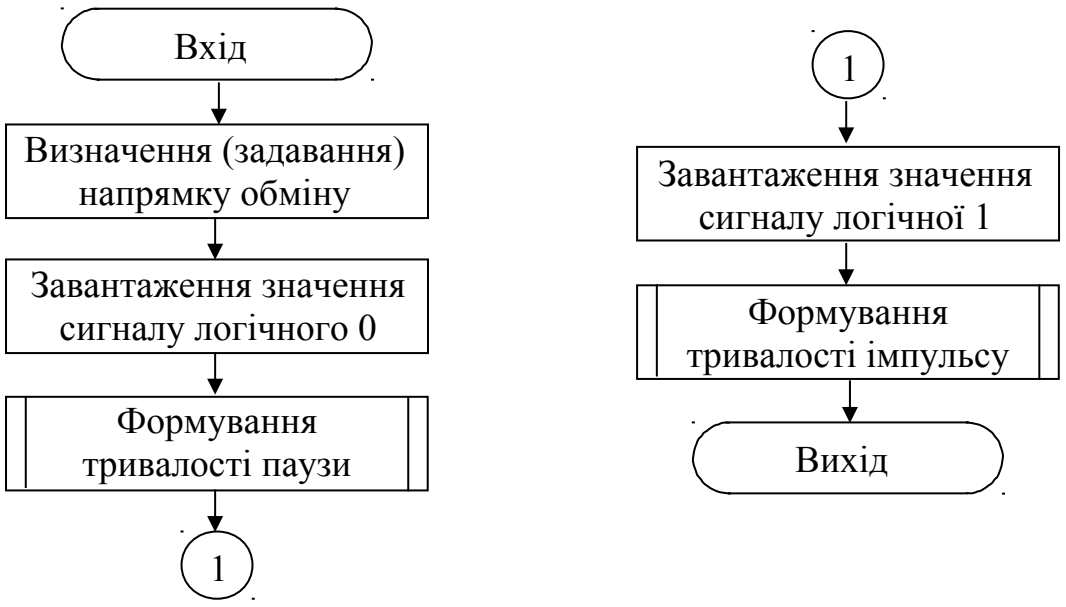


Рисунок 2.25 – Блок-схема алгоритму роботи MC68HC705J1A в якості генератора імпульсів

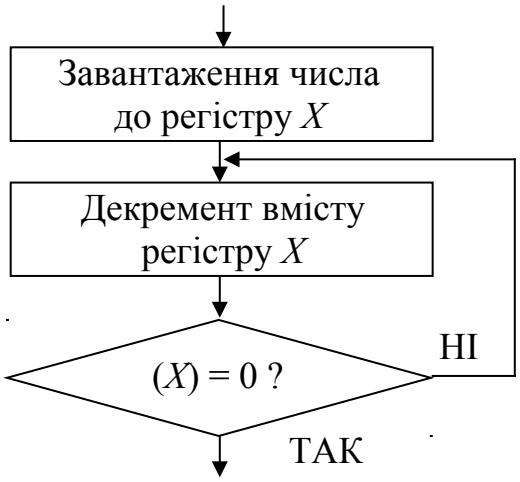


Рисунок 2.26 – Блок-схема алгоритму підпрограми часової затримки

Число, завантажуване до регістра, визначає час затримки підпрограми.

Після виконання цієї підпрограми слід завантажити до регістру даних нове значення сигналу і сформувати часову затримку, що дорівнювала б часові виведення цього значення.

За алгоритмом рис. 2.25, з урахуванням алгоритму рис. 2.26, розроблюємо програму генератора імпульсної послідовності мовою Асемблер.

```
        LDA  #FF        ; Визначення напрямку обміну на виведення
        STA  $0004      ; даних каналом A
M3:     LDA  #00        ; Завантаження рівня сигналу, який
                        ; виводитиметься каналом A
        STA  $0000      ; Виведення каналом A сигналу логічного 0
        JSR  M1         ; Безумовний перехід до підпрограми затримки,
                        ; розміщеної за міткою M1
        LDA  #FF        ; Встановлення рівня логічної 1 і виведення цього
        STA  $0000      ; сигналу каналом A
        JSR  M2         ; Безумовний перехід до підпрограми,
                        ; розміщеної за міткою M3
        BRA  M3         ; Безумовний перехід до формування нового
                        ; періоду сигналу
M1:     LDX #8          ; Завантаження до індексного регістру X кількості
                        ; циклів виконання підпрограми формування
                        ; сигналу логічного 0
M4:     DECX            ; Підпрограма формування часового інтервалу
                        ; паузи між імпульсами. Декремент вмісту
                        ; регістру X
        BNE  M4         ; Умовний перехід до команди DECX
        RTS             ; Вихід з підпрограми формування часового
                        ; інтервалу паузи між імпульсами. Повернення
                        ; здійснюється до команди LDA #FF, що є
                        ; наступною за командою звернення до цієї
                        ; підпрограми
```

```
M2:  LDX #4          ; Завантаження до індексного регістру X кількості
                     ; циклів виконування підпрограми формування
                     ; сигналу логічної 1
M5:  DECX            ; Підпрограма формування одного імпульсу
                     ; з послідовності
                     ; Декремент вмісту регістру X
     BNE  M5         ; Умовний перехід до команди DECX
     RTS             ; Вихід з підпрограми формування часового
                     ; інтервалу паузи між імпульсами. Повернення
                     ; здійснюється до команди BRA
```

Ця програма формуватиме імпульсну послідовність, частота слідування імпульсів в якій визначатиметься тактовою частотою МК.

Використовуючи таймер, який входить до складу МК, можна також формувати сигнали з потрібною часовою затримкою і використовувати ці сигнали для керування потрібними об'єктами. Для формування потрібних часових інтервалів можна використовувати сигнали переривання від таймера або від периферійних пристроїв.

2.13 *RISR*-процесори фірми "Моторола"

Назва *RISC* (*Reduced Instruction Set Computing*) означає "комп'ютер зі зменшеним набором команд". Система команд *RISC*-процесорів має фіксований формат команд, що значно спрощує керувальний пристрій та зменшує об'єм мікропрограмної пам'яті і призводить до зменшення розміру кристала *RISC*-процесорів, зниження їхньої вартості та підвищення тактової частоти. Система команд є вільна від складних і рідко використовуваних команд та способів адресування. Введення фіксованого набору команд забезпечує високу ефективність роботи конвеєра, зменшує кількість тактів простоювання та очікування процесорів, що підвищує їхню ефективність.

Переваги та ефективність *RISC*-процесорів зумовили їхнє виробництво усіма провідними фірмами, які випускають мікропроцесори: *Motorola, Intel, Hewlett-Packard, IBM, Sun Microsystems, MIPS*.

За найоптимальніші *RISC*-процесори та *RISC*-контролери вважаються розробки консорціуму фірм *Motorola, IBM, Apple Computers – MPC*6*XX PowerPC*, а також розробки фірми *Motorola – MFC*5*XXX ColdFire*.

До основних особливостей *RISC*-процесорів можна віднести:

– розширений об'єм регістрової пам'яті: від 32-х до кількох сотень регістрів загального призначення;

– використання в командах оброблення даних лише регістрового адресування; команди звернення до пам'яті використовуються лише при завантаженні та зберіганні вмісту регістрів, а також у командах керування програмою;

– відмову від апаратної підтримки складних способів адресування – з постінкрементом, предекрементом та деяких непрямих;

– фіксований формат команд, звичайно чотири байти, замість змінного формату (від одного до 12-ти байтів), що є характерним для CISC-мікропроцесорів;

– відсутність команд, які не відповідають прийнятому форматові.

За базову модель МП *PowerPC* можна вважати *MPC*604. У цій моделі, як і у сімействі *MC*68*XXX*, зберігається принцип виокремлення певних ресурсів для розв'язування завдань супервізора та користувача. Це означає, що архітектура процесора вміщує окремі регістри, які входять і до моделі користувача і до моделі супервізора, і має привілейовані команди, які виконуються лише в режимі супервізора.

До регістрової моделі користувача (рис. 2.27), яка є спільною для всього сімейства *PowerPC*, входять:

– тридцять два 32-розрядні регістри загального призначення *GPR*31...*GPR*0 для зберігання цілочисельних операндів;

– тридцять два 64-розрядних регістри *FPR*31…*FPR*0 для зберігання операндів з плаваючою точкою;

– 32-розрядні регістри прапорців (умов) *CR* та станів *FPSCP* (рис. 2.28, а) при обробленні даних з плаваючою точкою;

– 32-розрядні регістри *XER*, *LR*, *CTR*, які використовуються при обробленні виключень.

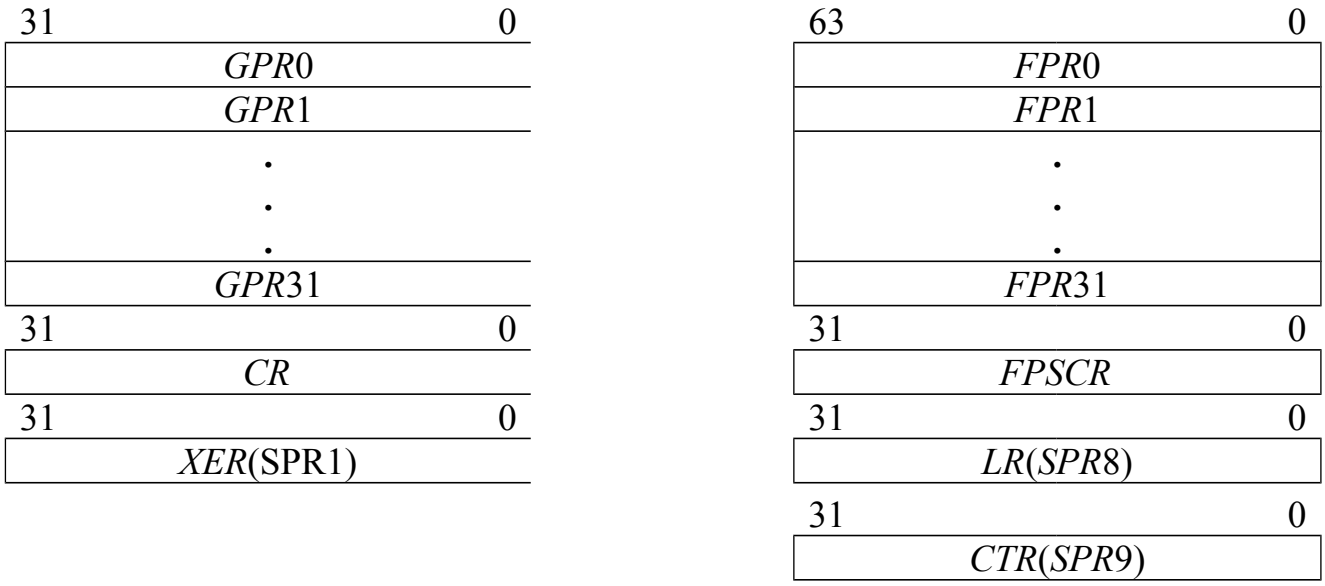


Рисунок 2.27 – Регістрова модель користувача
для мікропроцесорів *PowerPC*

Слід звернути увагу на те, що регістрова модель не вміщує ні програмного лічильника, ні вказівника стека.

На рис. 2.27 у дужках подано також номери певних поіменованих регістрів, які використовуються у певних командах звернення до цих регістрів як до службових.

Регістрова модель супервізора, крім регістрової моделі користувача, вміщує кілька груп службових регістрів, основним з яких є регістр керування

MSR, який визначає режим функціонування процесора. Режими функціонування процесора можуть бути такими:

– зі зниженим енергоспоживанням;
– з встановленням порядку *big-endian* або *little-endian* оброблення байтів при виключеннях;
– з обслуговуванням зовнішніх запитів переривань;
– режим користувача або режим супервізора;
– режим виконання операцій над числами з плаваючою точкою;
– режим трасування;
– режим задавання базової адреси векторів переривань;
– режим дозволу трансляції адрес команд та даних за допомогою пристроїв керування пам'яттю *IMMU*, *DMMU*;
– режим з контролем продуктивності процесора тощо.

Процесор вміщує таймер базового часу, який перемикається тактовою частотою.

На рис. 2.28 подано структуру МП MPC604. Процесор має суперскалярну структуру, яка складається з шести виконавчих пристроїв, які працюють паралельно:

– блок оброблення розгалужень *BPU*;
– два блоки для виконання одноциклових цілочисельних операцій *SIU*1, *SIU*2;
– один пристрій для виконання багатоциклових цілочислових операцій *MIU*;
– пристрій оброблення даних з плаваючою точкою *FPU*;
– блок вибирання операндів з пам'яті *LSU*.

Суперскалярна структура забезпечує одночасне виконання чотирьох команд.

Через суперскалярну структуру процесора можливе звернення виконавчих пристроїв одночасно до одних і тих самих регістрів. З метою зниження помилок за спроби запису до регістру введено буферні регістри, які затримують запис доти, поки інший пристрій не зчитає старі дані. Вони слугують для проміжного зберігання операндів, які дублюють регістри *GPR*, *FPR*, використовувані при виконанні поточних операцій.

Після завершення цих операцій здійснюється перезапис результатів у *GPR* та *FPR*, так званий зворотний запис.

Конвеєр виконання команд має шість основних каскадів: вибирання команди (1), дешифрування (2), розподілення команд між виконавчими пристроями (3), виконання команд (4), запису результатів до буфера (5) та зворотного запису результатів до регістрів *SPR* чи *FPR*. Пристрій керування одночасно вибирає з кеша і декодує чотири команди, які розподіляються по відповідних виконавчих пристроях.

Рисунок 2.28 – Структура мікропроцесора MPC604

133

Більшість команд виконуються за один такт. Ці команди зреалізовуються за допомогою *SIU*1 та *SIU*2, які здійснюють додавання, віднімання, порівняння, зсуви, логічні операції. Цілочисельне множення здійснюється за 3 або 4 такти, ділення – за 20 тактів. Ці операції виконує пристрій *MIU*.

Інтерфейс мікропроцесора із зовнішніми пристроями забезпечується контролером шини *BIU*. Для кожного байта адреси та даних передаються і приймаються біти парності, за допомогою яких здійснюється контроль помилок звернення до шини. Шина адреси має 32 розряди, а шина даних – 64 розряди, що підвищує продуктивність мікропроцесора. Припустиме є підключення зовнішнього кеша 2-го рівня.

Мікропроцесор вміщує спеціалізований послідовний порт для виконування тестування за стандартом *JTAG* (*IEEE* 1149.1).

Для живлення мікропроцесора використовується напруга $V_{ж} = 3{,}3$ В. Зниження енергоспоживання (400 мВт) забезпечується при зупині мікропроцесора, коли припиняється виконання команд, але продовжують працювати базовий таймер, система декрементування та блок обслуговування запитів на переривання.

2.14 Загальні принципи цифрового оброблення сигналів (ЦОС). Архітектура та принципи побудови цифрових процесорів (DSP)

До основних напрямків ЦОС належать цифрова фільтрація та спектральний аналіз. Цифрова фільтрація реалізовується засобами пропускання цифрового сигналу через цифрові фільтри зі скінченною та нескінченною імпульсними характеристиками – СІХ- та НІХ-фільтри. Обидва типи фільтрів є лінійні системи з постійними параметрами, у яких вхідна x_n та вихідна y_n послідовності сигналів пов’язані відношеннями типу згортки. Відгук системи на поодинокий імпульс (імпульсна характеристика) позначимо через h_k, тоді залежність поміж відліками вихідного y_n та вхідного x_n сигналів можна описати виразом

$$y_n = \sum_{k=0}^{n} h_k x_{n-k}\,,\ n = 0,\ 1,\ 2,\ ..., \qquad (2.1)$$

де x_n, y_n – відліки вхідного та вихідного сигналів; x_{n-k} – вхідний відлік, затриманий на k інтервалів дискретизації.

У СІХ-фільтрі відлік вихідного сигналу визначається лише значеннями вхідного сигналу, а у НІХ-фільтрі – значеннями вхідного та вихідного сигналів. Вираз, який описує НІХ-фільтр, має вигляд

$$y_n = \sum_{k=0}^{N-1} b_k x_{n-k} - \sum_{k=1}^{M-1} a_k y_{n-k}\,, \qquad (2.2)$$

де N, M є цілі константи; a_k, b_k – постійні коефіцієнти, які описують конкретний фільтр; x_n, y_n – відліки вхідного та вихідного сигналів.

СІХ-фільтр описується виразом

$$y_n = \sum_{k=0}^{N-1} b_k x_{n-k} \tag{2.3}$$

або

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} + \ldots + b_n x_{n-N+1}. \tag{2.4}$$

Ефективна система цифрової фільтрації потребує ефективної реалізації дискретної згортки, яку можна розкласти на операції множення, накопичуючого підсумовування та операції затримки.

В області спектрального аналізу використовується пряме та обернене дискретне перетворення Фур'є (ДПФ), швидке перетворення Фур'є (ШПФ) тощо. Спектральний аналіз базується на відомих методах подання заданої функції за допомогою інших, які називаються базовими і властивості яких є відомі.

Потрібні для цифрового оброблення сигналів операції підтримуються апаратно у процесорах цифрового оброблення сигналів. Узагальнену архітектуру процесорів DSP подано на рис. 2.29.

Процесори DSP563XX використовуються в мобільному зв'язку, цифрових системах комутації, пристроях оброблення мовного сигналу, тонального набору, цифрових телевізійних- та радіосистемах, диктофонах, локальних мережах, портативній техніці, відеотелефонах, цифрових фільтрах, аналізаторах спектра, криптографії, системах керування, навігаційному обладнанні, супутниковому зв'язку, розпізнаванні образів тощо. DSP563XX побудовані за гарвардською архітектурою і мають середню швидкодію 80 MIPS.

Периферія вміщує 8-бітний паралельний хост-інтерфейс, 32-бітний універсальний хост-інтерфейс, два розширені синхронні послідовні інтерфейси – *ESSI*1 та *ESSI*0, послідовний комунікаційний інтерфейс SCI, модуль таймера. На кристалі є вбудовані співпроцесори:

– фільтр-співпроцесор *FCCP* (*Filter Co-Processor*), який реалізовує алгоритми фільтрації;

– Вітербі-співпроцесор *VCCP* (*Viterbi Co-Processor*), який реалізовує алгоритм для відновлення з максимальною вірогідністю сигналу зі спотвореннями, наприклад *GSM*;

– співпроцесор циклічного коду *CCOP* (*Cyclic-code Co-Processor*), який зреалізовує кодування та декодування даних, генерування коду парності та контролю.

Підсистема пам'яті – ОЗП програм, кеш-інструкцій, ОЗП даних *X*, ОЗП даних *Y* – може бути сконфігурована в чотири способи і вміщує також ПЗП програм з організацією 6144x24, ПЗП даних *Y* – 3072x24, ПЗП, яке завантажує програми із зовнішньої пам'яті – 192x24.

Узагальнену архітектуру процесорів сімейства DSP563XX подано на рис. 2.29.

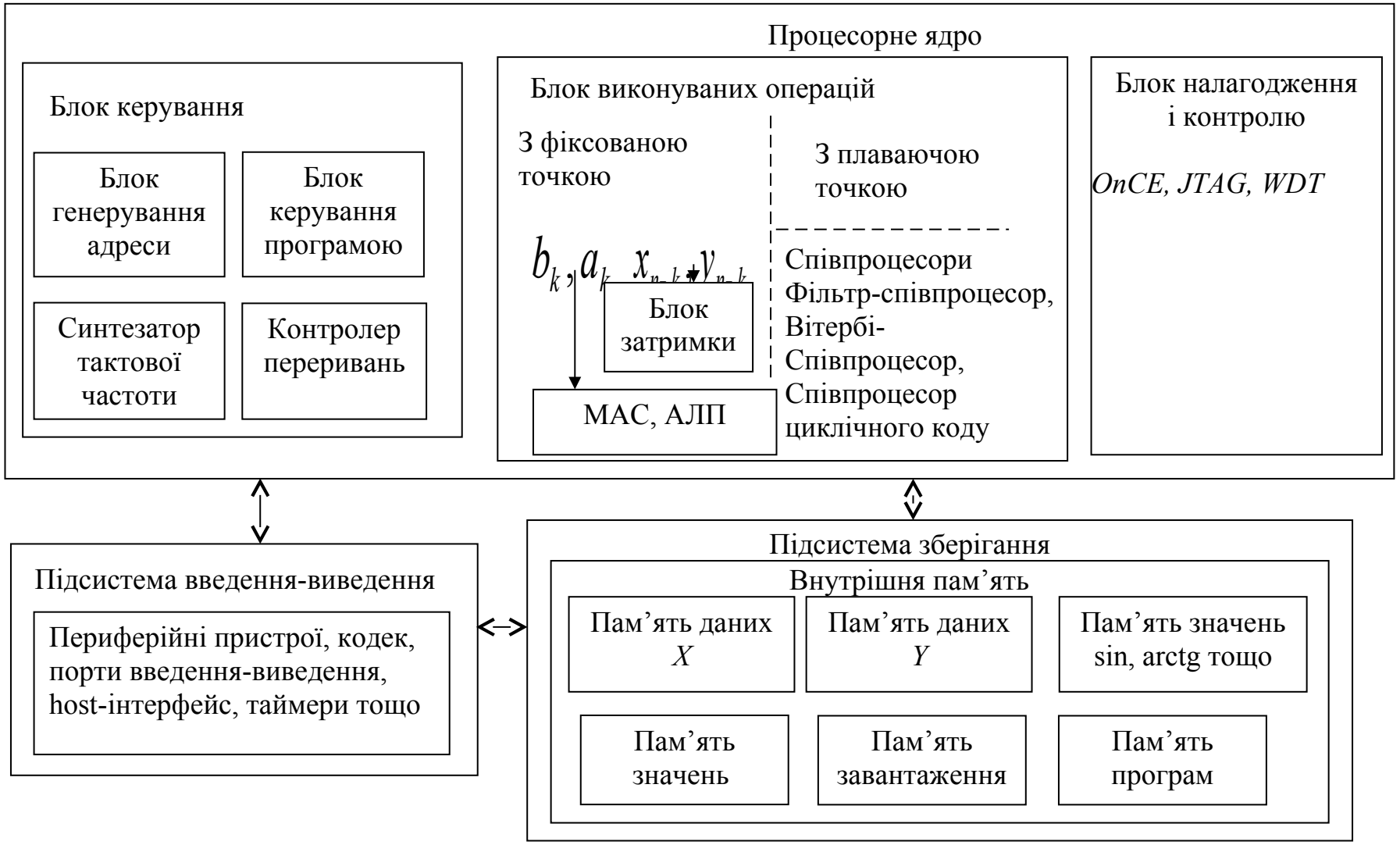


Рисунок 2.29 – Узагальнена архітектура DSP

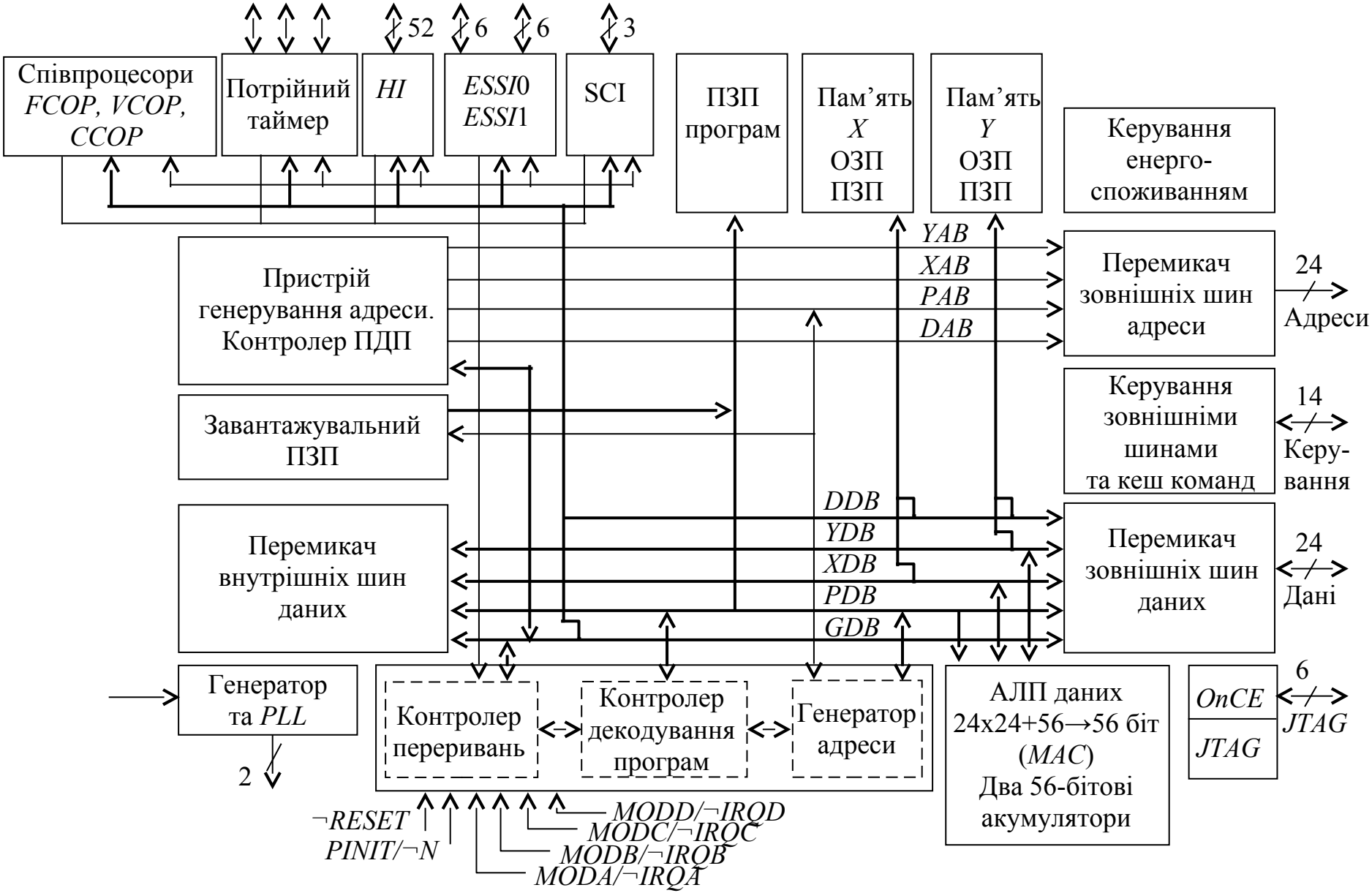


Рисунок 2.30 – Узагальнена архітектура процесорів сімейства DSP563XX

137

Сімейство DSP 563XX (рис. 2.30) вміщує нове ядро, базоване на новітніх технологіях, які зумовили низьку вартість, низьке енергоспоживання, високу ефективність мікропроцесорів. За рахунок включення кеша команд об'ємом 1кх24 процесори допускають підключення повільної зовнішньої пам'яті при зберіганні такої самої ефективності. Продуктивність процесора лінійно залежить від частоти генератора. На кристалі BIC розміщено синтезатор частоти, порт налагодження *JTAG*, потрійний таймер, *host*-інтерфейс (*HI*) для побудови багатопроцесорних систем, розширені синхронні послідовні інтерфейси *ESSI*0 та *ESSI*1, контролер кеша, послідовний комунікаційний інтерфейс *SCI*, пристрій керування потужністю, шестиканальний контролер ПДП (*DMA Data Bus*) контролер переривань для оброблення переривань у режимах *MODA*, *MODB*, *MODC*, *MODD*.

До архітектури процесорів сімейства долучено додаткову шину даних *DDB* (*DMA Data Bus*), що дозволяє за допомогою контролера ПДП передавати блоки інформації, не уповільнюючи роботу процесора.

Сімейство DSP563XX підтримує промислові стандарти щодо комп'ютерної техніки, мікропроцесорів, *DSP* та контролерів ПДП.

32-розрядна шина хост-інтерфейсу (*HI*) зреалізовує три класи інтерфейсів: шини *PCI*, універсальну шину, порт введення-виведення загального призначення.

Сімейство DSP563XX має продуктивність 66/80/100 *MIPS* на частотах відповідно 66/80/100 Мгц.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1 Угрюмов Е.П. Цифровая схемотехника Угрюмов Е.П. – СПб.: БХВ – Петербург, 2004, 528 с.: ил.

2 Китаев Ю. В. Основы цифровой техники: учеб. пособ. Китаев Ю. В. – СПб: СПбГУ ИТМО, 2007 – 87 с.

3 Шагурин И. И. Микропроцессоры и микроконтроллеры фирмы Motorola: справ. пособ. Шагурин И. И. – М.: Радио и связь, 1998. – 560 с.: ил.

4 Хіхловська І.В., Антонов О.С. Обчислювальна техніка та мікропроцесори : Підручник. – [2-ге вид.] Хіхловська І.В., Антонов О.С. – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 440 с.: іл..

5 Шагурин И. И. Современные микроконтроллеры и микропроцессоры Motorola: справ. пособ. Шагурин И. И. – М.: Горячая линия – Телеком, 2004. – 952 с.: ил.

6 Куприянов М. С. Цифровая обработка сигналов: процессоры, алгоритмы, средства проектирования. – [2-е изд., перераб. и доп.] М. С. Куприянов М. С., Б. Д. Матюшин – С.Пб.: Политехника, 1999. – 592 с.: ил.

7 Солонина А. И. Цифровые процессоры обработки сигналов фирмы Motorola Солонина А. И., Уласович Д. А., Яковлев Л. А. – С.Пб.: БХВ-Петербург, 2000. – 512 с.: ил.

Навчальне видання

Антонов Олександр Сергійович

ЦИФРОВІ ПРИСТРОЇ

Конспект лекцій

дисципліни «Цифрові пристрої» ННІ Радіо, телебачення, електроніки
для студентів 3 курсу

Редактор В.В. Терземан
Комп. верстання Ж.А. Гардиман

Підписано до друку 12.05.2015
Формат 60/88/16 Зам. № 04/15
Тираж 25 прим. Обсяг 8,3 ум. друк. арк.
Віддруковано на видавничому устаткуванні фірми RISO
у друкарні редакційно-видавничого центру ОНАЗ ім. О.С. Попова
ОНАЗ, 2015