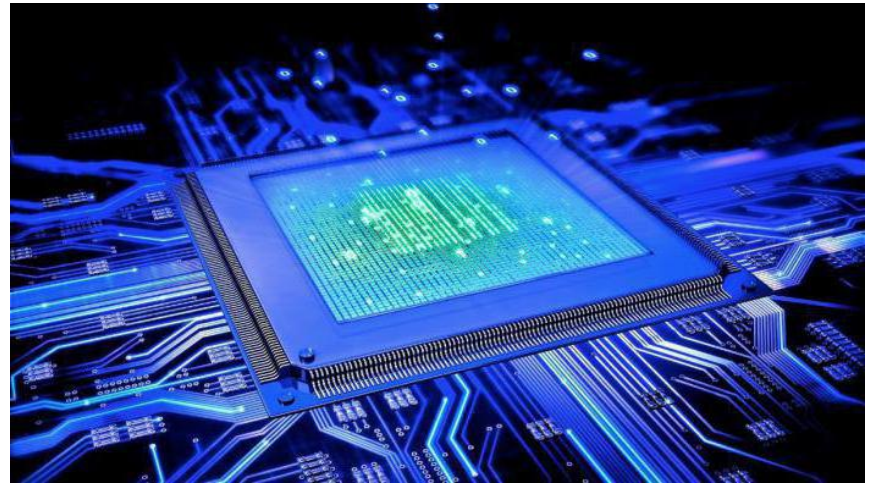
К. В. Огородник, Б. П. Книш

МІКРОПРОЦЕСОРНА ТЕХНІКА



Міністерство освіти і науки України
Вінницький національний технічний університет

К. В. Огородник, Б. П. Книш

МІКРОПРОЦЕСОРНА ТЕХНІКА

Навчальний посібник

Вінниця
ВНТУ
2018

УДК 004.31-022.513(075.8)
О-39

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 8 від 29.03.2018 р.)

Рецензенти:
В. М. Кичак, доктор технічних наук, професор
В. С. Осадчук, доктор технічних наук, професор
А. Я. Кулик, доктор технічних наук, професор

Огородник, К. В.

О-39 Мікропроцесорна техніка : навчальний посібник / К. В. Огородник, Б. П. Книш. – Вінниця : ВНТУ, 2018. – 106 с.

У навчальному посібнику наведено загальні принципи побудови мікропроцесорів, мікроконтролерів та мікропроцесорних систем, розглянуто особливості архітектури і функціональні можливості процесорів, способи організації й побудови модулів пам’яті та пристроїв введення/виведення.

Метою навчального посібника є ознайомлення читача із сучасними принципами мікропроцесорної техніки.

Посібник розроблено відповідно до навчальної програми з дисципліни «Мікропроцесорна техніка» та розраховано для студентів спеціальностей 153 – «Мікро- та наносистемна техніка» та 171 – «Електроніка».

УДК 004.31-022.513(075.8)

ЗМІСТ

ВСТУП

Важливе місце у схемотехніці електронних систем посідають системи керування з мікропроцесорами та мікроконтролерами, які дозволяють реалізувати складні закони керування електронними пристроями. Знання схемотехніки аналогових та цифрових систем створює базу для вивчення принципів побудови мікропроцесорних систем керування. Перевага мікропроцесорних систем керування – їх гнучкість: систему, розроблену для виконання конкретного завдання керування, легко застосувати для вирішення інших завдань зміною програмного забезпечення.

Сучасні мікропроцесорні системи містять усі складові електронних обчислювальних машин – мікропроцесор, пам'ять даних, пам'ять програм, інтерфейсні схеми – та ефективно використовуються в системах керування промислового та побутового обладнання.

Розширення функцій мікропроцесорних систем потребує вдосконалення знань спеціалістів різних профілів у цьому напрямі. Тому вивчення основ мікропроцесорної техніки є неодмінною складовою підготовки спеціалістів вищих навчальних закладів. Незважаючи на велику різноманітність типів мікропроцесорів та функцій, що вони виконують, логіка побудови систем і створення програмного забезпечення залишається незмінною. Вивчення загальних принципів побудови, особливостей архітектури, використання різних типів пам'яті дає теоретичну базу для розробки та використання мікропроцесорних систем різних типів [1].

Тому метою навчального посібника є ознайомлення читача з сучасними принципами мікропроцесорної техніки.

В основу посібника покладено курс лекцій з дисципліни «Мікропроцесорна техніка».

Посібник складається з восьми розділів, що містять загальні відомості про мікропроцесорну техніку, системи числення і кодування інформації, архітектуру мікропроцесорної системи, фізічну та логічну організацію пристроїв пам'яті, пристрої введення/виведення, сигнальні процесори, програмовані логічні інтегральні схеми.

Посібник адресований широкому колу читачів, що займаються розробкою електронних та мікропроцесорних систем та розрахований на студентів спеціальностей 153 – «Мікро- та наносистемна техніка» та 171 – «Електроніка» й може бути корисним студентам інших спеціальностей.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО МІКРОПРОЦЕСОРНУ ТЕХНІКУ

1.1 Основні поняття

Мікропроцесорна техніка (МПТ) – це технічні і програмні засоби, які використовуються для побудови різних мікропроцесорних систем і пристроїв.

Мікропроцесор (МП) – програмно-керований пристрій, призначений для обробки цифрової інформації та управління процесом цієї обробки, виконаний у вигляді однієї (або декількох) інтегральної схеми з високим рівнем інтеграції електронних елементів [2].

Мікропроцесорна система (МПС) складається з однієї або декількох мікросхем, які містять мікропроцесор, модулі пам'яті та модулі введення/виведення. Спрощена структурна схема МПС має вигляд, показаний на рис. 1.1.

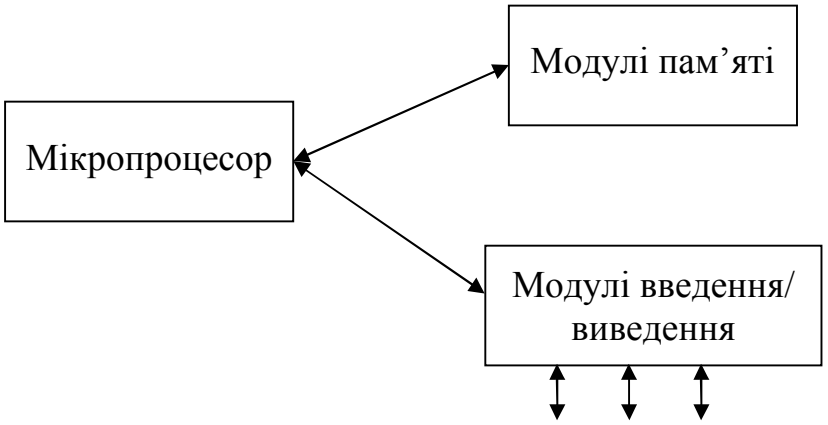


Рисунок 1.1 – Спрощена структурна схема МПС

У складі МПС на МП покладається завдання виконання всіх програмних дій, необхідних відповідно до алгоритму роботи. У блоці пам'яті зберігаються команди програми процесора, а також значення констант і змінних величин, які використовуються в обчисленнях. Блок введення/виведення виконує функцію сполучення МПС з об'єктом управління.

Зменшення вартості, споживаної потужності і габаритних розмірів, підвищення надійності і продуктивності мікропроцесорів сприяли значному розширенню сфери їх використання. Разом з традиційними обчислювальними системами вони все частіше почали використовуватися в задачах управління.

При цьому перед мікропроцесором ставилися завдання програмного управління різними периферійними об'єктами в реальному часі. Широке використання мікропроцесорної техніки саме для завдань управління привело до появи на ринку спеціалізованих мікропроцесорних пристроїв, орі-

єнтованих на подібний тип використання. Особливістю цих мікросхем є те, що окрім власне процесора, на цьому ж кристалі розташована і система введення/виведення, що дозволяє знизити функціональну складність і габаритні розміри мікропроцесорної системи управління.

Мікроконтролер (МК) – обчислювальний керівний пристрій, який призначений для виконання функцій логічного контролю і управління периферійними пристроями, виконаний у вигляді однієї великої інтегральної схеми, містить мікропроцесорне ядро, що поєднує в собі пам'ять і набір вбудованих пристроїв введення/виведення.

Мікропроцесорний пристрій є функціонально і конструктивно закінченим виробом, що складається з декількох мікросхем, до складу яких входить МП. Він призначений для виконання певного набору функцій: отримання, обробка, передача, перетворення інформації і управління.

МП нині переважно використовуються для виробництва персональних комп'ютерів, а МК є основою створення різних вбудовуваних систем, телекомунікаційного, портативного устаткування тощо.

Сучасні системи автоматизацій технологічних процесів будуються на спеціалізованих мікропроцесорний пристроях, які називаються програмованими логічними контролерами (PLC) та потужнішими програмованими контролерами для автоматизації (PAC), що виконують не лише роль безпосереднього управління процесами за рахунок вбудованих периферійних пристроїв, але є цифровими регуляторами системи захисту і діагностики, системи зв'язку з мережею вищого рівня.

Окрім МП та МК мікропроцесорна техніка охоплює сигнальні процесори (СП) для перетворення та цифрової обробки аналогових сигналів та програмовані логічні інтегральні схеми (ПЛІС) для програмного конфігурування та реалізації схем цифрової логіки [1, 2].

1.2 Аналогові і цифрові методи обробки інформації

Завданням будь-якої системи управління є обробка інформації про поточний режим роботи керованого об'єкта і продукування на основі цього сигналів управління з метою наближення поточного режиму роботи об'єкта до заданого. В електронних системах існують два основні способи обробки інформації: аналоговий і цифровий.

При аналоговому способі обробки інформації кожній змінній величині в системі ставитися у відповідність один з плавно-змінних параметрів певної ділянки електричного кола (струм, напруга, частота, фаза). Функціональні залежності між різними змінними в системі реалізуються шляхом побудови відповідних електричних кіл.

Принциповою особливістю аналогового способу обробки інформації є можливість плавної (у відомих межах) зміни величин електричних сигналів, відповідних змінним системи. Всі перетворення здійснюються практично миттєво.

При цифровому способі обробки інформації кожній змінній величині в системі ставитися у відповідність її цифровий код. Функціональні залежності в системі реалізуються шляхом безпосереднього розв'язання рівнянь системи тими або іншими чисельними методами за заздалегідь закладеною програмою. Пристрій, що реалізовує це рішення, називається процесором.

Відмінною особливістю цифрових систем управління є дискретизація сигналу за рівнем, величина якого визначається розрядністю обчислень (рис. 1.2). Так, у випадку 8-розрядної системи, весь діапазон зміни значення сигналу ділиться на 256 ділянок і цифровий код, відповідний цьому сигналу, може набувати лише одного з 256 значень. Це, вочевидь, накладає обмеження на точність цифрової системи управління. Внаслідок цього довгий час в прецизійних системах продовжували (і у ряді випадків продовжують) використовувати аналогові методи обробки інформації. Проведемо порівняльний аналіз. Нехай в аналоговій системі деякий сигнал, в амплітуді якого закладена інформація, може змінюватися в межах від 0 до 10 В. Рівень шуму при цьому не перевищує 1 мВ. Для достовірної передачі інформації, що виключає вплив шумів, мінімальний приріст сигналу має становити, як мінімум, 1 мВ. Таким чином, за допомогою подібного сигналу можна передати 10000 одиниць інформації.

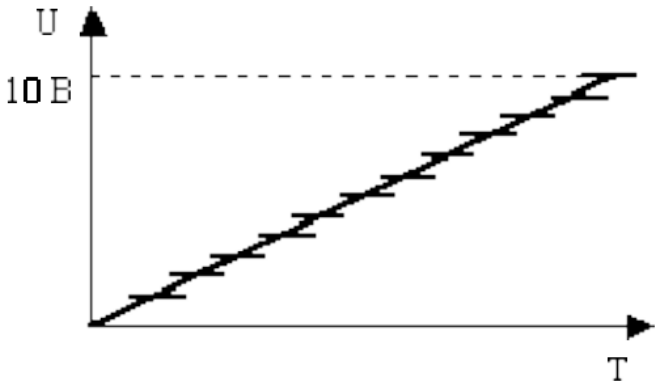


Рисунок 1.2 – Дискретизація аналогового сигналу за рівнем

Для передачі такої ж кількості інформації у цифровому коді необхідно мати розрядність, як мінімум, 14 двійкових розрядів. Отже, цифрові системи з меншою розрядністю поступатимуться за точністю описаній аналоговій системі. Проте, за наявності розрядності, більшої за 14 бітів цифрова система може не лише не поступатися, але і перевершувати за точністю аналогову, оскільки її параметри не змінюються з часом під впливом таких зовнішніх чинників, як температура, вологість і тому подібне, що значною мірою властиво практично всім аналоговим системам.

Іншою відмінною особливістю МПС є послідовне в часі виконання команд процесором. Всі команди, кожна з яких має скінченний час виконання, виконуються послідовно, одна за одною. Отже, від моменту початку виконання алгоритму до кінцевого результату проходить деякий інтервал

часу. Це негативно позначається на швидкодії цифрових систем і, зокрема, на області стійкості і смузі пропускання цифрових систем управління. Проте процес підвищення швидкодії мікропроцесорних пристроїв йде неухильно вперед і нині існують мікропроцесори, час виконання команди в яких близько 1 нс. За допомогою сучасних мікропроцесорів можливо створювати системи управління із смугою пропускання в десятки і навіть сотні кілогерц. Зі свого боку, аналогові системи, незважаючи на практично миттєве протікання сигналів, також мають кінцеву швидкодію через неідеальності компонентів і наявності паразитних реактивних зв'язків в системі.

Завдяки всьому вищенаведеному на сьогодні відбувається повномасштабне впровадження МПС практично у всі сфери людської діяльності, причому навіть там, де ще вчора панували аналогові методи обробки інформації [2].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Що являє собою мікропроцесор?
2. Які елементі входять до складу МПС, в чому їх призначення?
3. Наведіть спрощену структурну схему МПС.
4. На яких програмованих пристроях будуються сучасні системи автоматизацій технологічних процесів?
5. В чому різниця між аналоговими і цифровими сигналами?

2 СИСТЕМИ ЧИСЛЕННЯ І КОДУВАННЯ ІНФОРМАЦІЇ

2.1 Одиниці вимірювання інформації

Кодуванням інформації прийнято називати її перетворення на основі певних стандартних алгоритмів. Результат такого перетворення, який являє собою задання інформації в іншій формі, називається кодом [3, 4].

Будь-яка інформація в цифрових системах – числа, символи, графіка, команди програми – зберігається у двійковій формі, тобто у вигляді послідовностей нулів та одиниць. Причина цього полягає в особливостях фізичної реалізації, при якій елементи цифрового комп'ютера можуть перебувати в одному з двох стійких станів: напруга висока – напруга низька або струм є –струму немає тощо.

Біт – один двійковий розряд, достатній для того, щоб закодувати одне з двох можливих значень – 1 або 0, істина або хибність. Це мінімальна одиниця інформації про об'єкт , який може знаходитися в одному із двох станів [3].

Якщо об'єкт має більше двох станів, то для кодування потрібно виділити відповідну кількість бітів, яка залежить від методу кодування.

Приклад для 4-х станів:

Весна	Літо	Осінь	Зима	
0001	0010	0100	1000	унітарний код
00	01	10	11	позиційний код

В унітарному коді кількість бітів дорівнює кількісті можливих станів, при цьому тільки один розряд встановлений в одиницю. Такий метод кодування дуже простий, але й водночас дуже вимогливий до обсягу пам'яті. Тому у сучасних МПС застосовується позиційний код із всіма можливими комбінаціями 0 та 1.

Набір з n двійкових розрядів дає кількість комбінацій $N=2^n$. Типовим набором двійкових розрядів є байт, що дорівнює 8 бітів. Кількість комбінацій $N=2^8=256$, діапазон значень натуральних чисел 0..255.

Типові одиниці інформації наведено у табл. 2.1

Таблиця 2.1 – Одиниці інформації

Одиниця	Назва	Розрядів, n	Комбінацій, N
Bit	Біт	1	2^1
Byte	Байт	8	$2^8 = 256$
Word	Слово	16	$2^{16} = 65536$
Dword	Подвійне слово	32	$2^{32} \sim 4$ млрд.
Qword	Четвірне слово	64	2^{64}

Похідними одиницями кількості інформації є кіло (1 к=2^{10}=1024), мега (1 М=2^{20}), гіга (1 Г=2^{30}). Необхідно звернути увагу на таке важливе питання. Як зазначалося, вся інформація в цифровому комп'ютері зберігається у вигляді послідовностей бітів і байтів. Як має інтерпретуватися, наприклад, послідовність 01000001? Інакше кажучи, звідки процесор може знати, що означає ця послідовність – двійковий запис числа 65 чи ASCII-код літери «А», чи положення точки на екрані ? Тут потрібно розуміти, що не задається ніяких апріорних правил для того, щоб визначити, що означають послідовності бітів. Правила інтерпретації двійкових кодів повністю залежать від конкретної програми [5].

2.2 Поняття про позиційні системи числення

Під системою числення розуміють сукупність правил зображення чисел цифровими знаками. Розрізняють позиційні й непозиційні системи числення.

В непозиційних системах числення вага кожного знака не залежить від його положення відносно інших знаків у числі, кількість знаків не обмежена. У римській системі числення: I – 1, V – 5, X – 10 тощо.

В одиничній системі числення число сім подається сімома одиничками: 7 = «11111111».

Недоліками непозиційних систем числення є:

- громіздкість зображення чисел;
- труднощі у виконанні операцій.

Система числення називається позиційною, якщо при записуванні числа одна і та ж цифра має різне значення, яке визначається місцем (позицією), на якому вона знаходиться.

В позиційній системі числення для записування числа використовується обмежена кількість знаків – цифр, яка визначає назву системи числення і називається її основою. Для позиційних систем числення характерним є те, що значення кожної цифри залежить від її положення у числі.

В десятковій системі числення для записування числа використовується десять цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 і основою є число 10. Число у десятковій системі числення можна подати у вигляді полінома (у вигляді степенів десяти):

$$353{,}3_{10} = 3\cdot10^2 + 5\cdot10^1 + 3\cdot10^0 + 3\cdot10^{-1}.$$

Формально позиційну систему числення можна визначити так. Нехай зафіксовано ціле число b, яке є основою системи числення. Будь-яке ціле число M задається у позиційній системі числення за основою b у вигляді послідовності розрядів $a_n a_{n-1} \ldots a_0$, причому всі a і цілі, і $0 \le a_i \le b_{-1}$. Значення числа обчислюється за формулою:

$$M = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 b^0.$$

Крайній правий розряд зі значенням a_0 називається молодшим розрядом, а крайній лівий зі значенням a_n – старшим розрядом. Число M за основою b позначається як M_b. Приклад:

$$709710_{10} = 7 \cdot 10^0 + 9 \cdot 10^1 + 0 \cdot 10^2 + \ldots + 7 \cdot 10^3.$$

Для комп'ютера більш типовою є двійкова система (за основою 2).

$$100110_2 = 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 0 \cdot 2^3 + 1 \cdot 2^5.$$

2.3 Двійкова система числення

Будь-яке число у двійковій системі числення записується у вигляді певної послідовності нулів та одиниць. Наприклад: 101011.

Наведемо в загальних рисах правила переведення чисел з двійкової системи в десяткову і навпаки . Алгоритм переведення чисел з двійкової системи в десяткову безпосередньо спирається на визначення позиційної системи числення. Всі розряди домножуються на відповідні ступені двійки (крайній справа – на 1, наступний – на 2 тощо), після чого отримані добутки додаються за правилами десяткової системи.

$$101011_2 = 1 + 2 \cdot 1 + 4 \cdot 0 + 8 \cdot 1 + 16 \cdot 0 + 32 \cdot 1 = 43_{10}.$$

Переведення цілого числа з десяткової системи числення до двійкової здійснюється шляхом його послідовного цілочисельного ділення на 2, поки в частці не буде отримано 0:

43 / 2 = 21, остача 1;
21 / 2 = 10, остача 1;
10 / 2 = 5, остача 0;
5 / 2 = 2, остача 1;
2 / 2 = 1, остача 0;
1 / 2 = 0, остача 1.

Остачі від ділень, якщо їх прочитати знизу вверх, утворюють число в двійковій системі (перша остача записується в молодший, тобто крайній справа, розряд, друга – в наступний за ним тощо) [4, 5].

2.3.1 Логічні операції над бітами та багаторозрядними числами

Операція OR («або» – диз'юнкція) дає результат «істина», якщо хоча б один з операндів дорівнює «істині» *b*1 OR *b*2.

Операція XOR (виключне «або») дає результат «істина», якщо тільки один з операндів дорівнює «істині», тобто операнди не дорівнюють один одному *b*1 XOR *b*2.

Операція AND («і» – кон'юнкція) дає результат «істина», коли обидва операнди дорівнюють одиниці *b*1 AND *b*2.

Операція NOT («не» – інверсія) – результат є протилежним значенню *b*1 NOT *b*1.

Найпоширеніші логічні операції істинності над бітами *b*1 і *b*2 зведені до табл. 2.2.

Таблиця 2.2 – Логічні операції істиності

*b*1	*b*2	OR	XOR	AND	NOT *b*1
0	0	0	0	0	1
0	1	1	1	0	1
1	0	1	1	0	0
1	1	1	0	1	0

При виконанні аналогічних операцій над багаторозрядними числами, операції виконуються порозрядно.

$$0110 \text{ and } 1101 = 0100 \ (6_{10} \text{ and } 13_{10} = 4_{10}).$$
$$\text{NOT } 10_{10} = \text{NOT } 1010_2 = 0101_2 = 5_{10}.$$

2.3.2 Операції зсуву бітів у багаторозрядному числі

Команди зсувів дозволяють побітово зсувати код операнда вправо (у бік молодших розрядів) чи вліво (у бік старших розрядів). Тип зсуву (логічний, арифметичний чи циклічний) визначає, яке буде нове значення старшого біта (при зсуві вправо) чи молодшого біта (при зсуві вліво), а також визначає, чи буде десь збережене колишнє значення старшого біта (при зсуві вліво) чи молодшого біта (при зсуві вправо). Наприклад, при логічному зсуві вправо в старшому розряді коду операнда встановлюється нуль, а молодший розряд записується як біт (прапорець) перенесення в регістр стану процесора. А при арифметичному зсуві вправо значення старшого розряду зберігається незмінним (нулем чи одиницею), молодший розряд також записується як прапорець перенесення.

Циклічні зсуви дозволяють зсувати біти коду операнда по колу (по годинниковій стрілці при зсуві вправо чи проти годинникової стрілки при зсуві вліво). При цьому в кільце зсуву може входити або не входити прапорець перенесення. У біт прапорця перенесення (якщо він використовується) записується значення старшого біта при циклічному зсуві вліво і молодшого біта при циклічному зсуві вправо. Відповідно значення біта прапорця перенесення буде переписуватися в молодший розряд при циклічному зсуві вліво й у старший розряд при циклічному зсуві вправо.

Для прикладу на рис. 2.1 показано дії, які виконуються командами зсуву вправо.

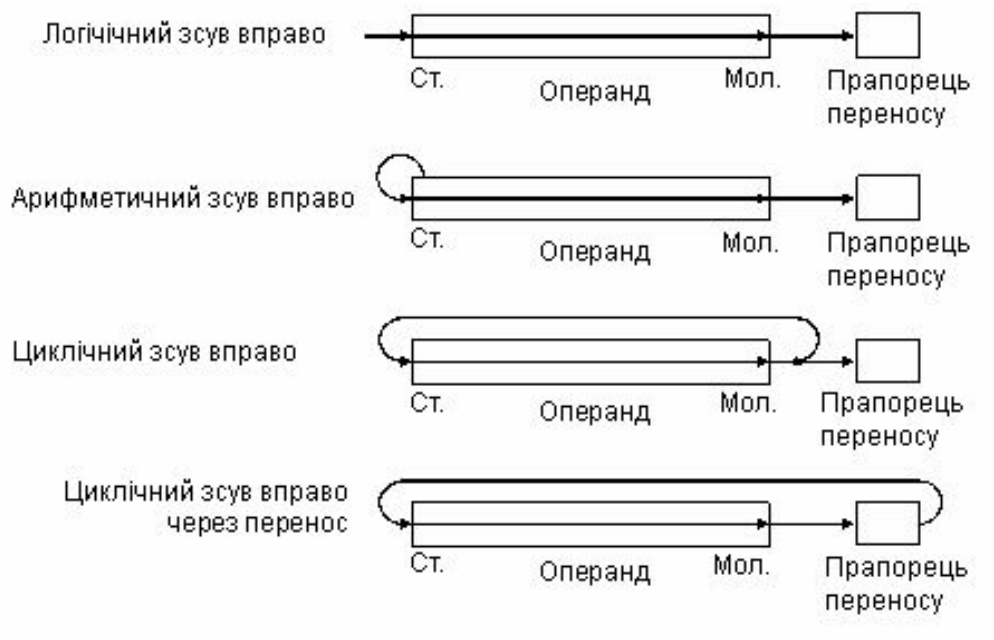


Рисунок 2.1 – Команди зсувів вправо

2.3.3 Двійкова арифметика

Арифметічні операції над двійковими числами (додавання, віднімання, множення, ділення) здійснюються аналогічно до звичних десяткових операцій.

Спочатку наведемо в таблиці 2.3 додавання, віднімання і множення найпростіших двійкових чисел.

Таблиця 2.3 – Додавання, віднімання і множення найпростіших двійкових чисел

Додавання	Віднімання	Множення
0+0=0	0-0=0	0·0=0
1+0=1	1-0=1	1·0=0
0+1=1	1-1=0	0·1=0
1+1=10	0-1=1	1·1=1

Для самоперевірки виконаємо додавання десяткових еквівалентів. Сума 5+3 дорівнює 8, що в двійковій системі числення записується як 1000. Отже, обчислення виконані правильно.

Додавання двох багаторозрядних двійкових чисел проводиться порозрядно з урахуванням одиниць, що переносяться з попередніх розрядів:

```
  1  1 1      – одиниці, що переносяться
   10110
+ 10111
  101101     (38 + 39 = 77).
```

В цьому прикладі додавання двох чисел виникло перенесення одиниць з 1-го, 2-го та 4-го розрядів у наступні старші розряди.

Віднімання багаторозрядних двійкових чисел аналогічно додаванню, починається з молодших розрядів. Якщо взяти одиницю в старшому розряді, створюються дві одиниці в молодшому розряді.

$$\begin{array}{r} 1010 \\ -\ \underline{0110} \\ 0100 \end{array} \quad (10 - 6 = 4).$$

Якщо при відніманні двох чисел, поданих в двійковій системі, зменшувана цифра менша тієї, що віднімається, то потрібно взяти одиницю в старшому розряді.

Множення є багаторазовим і зсувом вліво, і додаванням проміжних сум.

$$\begin{array}{r} 10011 \\ +\ \underline{\ \ 101} \\ 10011 \\ 00000 \\ \underline{10011\ \ } \\ 1011111 \end{array} \quad (19 \cdot 5 = 95).$$

Аналізуючи приклади множення в двійковій системі числення, необхідно звернути увагу на одну важливу особливість виконання цієї операції в даній системі, яка плягає в тому, що операція множення замінюється послідовністю зсувів і додавання.

Процес ділення складається з операцій множення і віднімання, які повторюються.

$$\begin{array}{rl} 101010 & \underline{\ 111} \\ \underline{-\ 111} & \ \ 110 \\ 0111 & \\ -\ \underline{111} & \\ 0000 & \\ 0000 & \end{array}$$

На практиці віднімання замінюють додаванням до числа, що має протилежний знак.

Таким чином, найважливіша перевага двійкової арифметики полягає в тому, що вона дозволяє всі арифметичні дії звести до одної – додавання, а це значно спрощує будову процесора [2, 5].

2.3.4 Подання від'ємних чисел

При здійсненні обчислень, звичайно, доводиться мати справу не тільки з цілими невід'ємними числами, але також з від'ємними і дробовими.

У програмуванні виділяють два типи чисел: беззнакові та знакові. Всі беззнакові числа вважаються невід'ємними, і всі їх розряди використовуються для задання абсолютної величини числа. Так, за допомогою одного байт а можна закодувати цілі беззнакові числа від 0 до 255.

Для подання ж від'ємних чисел потрібно виділити один біт для знака. Як правило, це старший біт. Якщо один біт в числі виділяється під його знак, таке число називається знаковим. Як правило, 0 у старшому (крайньому зліва) біті відповідає додатним числам, а 1 – від'ємним.

Подання від'ємних чисел залежить від кількості байтів, що відводиться на число. Для визначеності будемо розглядати однобайтові знакові числа.

Виділяють три основних способи подання від'ємних чисел:

- прямий код, який утворюється з коду відповідного додатного числа шляхом встановлення знакового біта в 1;
- обернений код, який утворюється шляхом заміни значення кожного біта на протилежне;
- додатковий код, який утворюється шляхом додавання 1 до молодшого біта оберненого коду.

Приклад. Розглянемо число -3. Двійковим еквівалентом відповідного додатного числа 3 є 00000011.

Прямий код цього числа визначається шляхом встановлення знакового біта в 1. Всі інші біти залишаються без змін. В результаті отримаємо 10000011.

Обернений код знаходиться заміною кожного біта на протилежний (1 на 0; 0 на 1). Результатом буде 11111100.

Додатковий код визначається шляхом додавання 1 до оберненого коду; в результаті – 11111101.

Додатковий код є найпоширенішим в МПС, саме для нього виконується правило – сума однакових за модулем додатного та від'ємного чисел дає нулі в усіх розрядах. Додамо значення двійкових кодів чисел +3 і -3:

```
  00000011
+ 11111101
1 00000000
```

Результатом буде байт із значенням 0 в усіх 8-ми розрядах. При цьому відбулось перенесення одиниці в 9-й розряд, тобто за межі байта. Цю одиницю мікропроцесори зберігають у службовому біті – «прапорці перенесення».

У табл. 2.4 подано відповідність між десятковими числами зі знаком та їх додатковими двійковими та шістнадцятковими еквівалентами у межах

байту. Не складно помітити, що від'ємні числа (-128..-1) зайняли місце беззнакових чисел (128..255), відповідно.

Таблиця 2.4 – Відповідність між десятковими числами зі знаком та їх додатковими двійковими та шістнадцятковими еквівалентами

DEC	BIN	HEX
127	01111111	7F
126	01111110	7E
125	01111101	7D
…		
3	0000011	03
2	0000010	02
1	00000001	01
0	0000000	00
- 1	11111111	FF
- 2	11111110	FE
- 3	11111101	FD

Продовження таблиці 2.4

DEC	BIN	HEX
- 4	11111100	FC
…		
-126	10000010	82
-127	10000001	81
-128	10000000	80

Для апаратної реалізації віднімання чисел використовують додавання з додатковими кодами.

Додавання і віднімання чисел в оберненому і додатковому кодах виконується з використанням звичайного правила арифметичного додавання багаторозрядних чисел.

Загальною для цих кодів особливістю є те, що при порозрядному додаванні чисел розряди, що відображають знаки чисел, розглядаються як рівноправні розряди двійкового числа, які додаються один з одним і з одиницею перенесення з попереднього розряду числа за звичайними правилами арифметики.

Відмінності ж оберненого і додаткового кодів пов'язані з тим, що робиться з одиницею перенесення із старшого розряду.

При додаванні чисел в додатковому коді одиниця перенесення із старшого розряду ігнорується (втрачається), а в оберненому коді цю одиницю треба додати до молодшого розряду результату [5].

2.3.5 Подання дробових чисел

Для дійсних чисел виділяють два основних формати подання:

- з фіксованою крапкою: положення десяткової крапки фіксується програмним шляхом; тоді все, що знаходиться зліва від крапки, вважається цілою частиною, а все, що справа – дробовою;

- з рухомою крапкою: цей формат застосовується для операцій з дуже великими або дуже маленькими числами; грунтується на поданні у вигляді $a \cdot 10b$; a називається мантисою, а b – порядком. Мантиса і порядок зберігаються окремо.

Для переведення правильного дробу з десяткової системи числення у будь-яку іншу потрібно помножити заданий дріб на основу нової системи числення. Отримана ціла частина добутку буде першою цифрою після коми дробу в новій системі числення. Далі за чергою домножаються дробові частини добутків на основу нової системи. Отримані цілі частини добутків будуть цифрами дробу у новій системі числення. Цей процес продовжують до тих пір, поки не буде знайдено число із заданою точністю.

Для переведення змішаного числа з десяткової системи числення в іншу необхідну окремо перевести цілу й дробову частини за вказаними правилами, а потім об'єднати результати у змішане число.

У прикладних задачах досить часто доводиться оперувати дуже великими або дуже маленькими дійсними числами, наприклад такими, як маса Сонця, що становить $2 \cdot 10^{30}$ кг, або маса електрона, яка становить $9 \cdot 10^{-28}$ г. Записати в пам'ять подібні числа, враховуючи всі значущі цифри, і виконати над ними арифметичні операції, використовуючи арифметику з фіксованою крапкою, неможливо. У цьому разі для запису чисел використовується формат із рухомою крапкою, коли кожне число розбивається на дві групи цифр. Перша група цифр називається мантисою, друга – порядком. Число записується у вигляді добутку.

$$Y = \pm M \, x \, S^{\pm p},$$

де Y – значення дійсного числа;

M – мантиса числа (дріб зі знаком);

S – основа системи числення;

p – порядок числа (ціле число зі знаком).

Мантиса і порядок зображуються в системі числення з основою S. Знак числа збігається зі знаком мантиси.

Порядок p є додатним або від'ємним цілим числом і визначає положення крапки в числі Y. Таким чином, у мантисі зберігаються значущі цифри числа, а порядок визначає його величину.

Дійсні числа записуються в 4, 6, 8 або 10 байтах. Для того, щоб в комірці оперативної пам'яті не зберігати знак порядку, замість нього використовується характеристика. Вона утворюється додаванням до порядку пев-

ного цілого числа. Це число вибирається так, щоб характеристика завжди була додатною. Характеристику іноді називають зсуненим порядком. При цьому формат зображення дійсних чисел є таким, як показано на рис. 2.2.

Зсунений порядок						Модуль мантиси			
Знак М	p_r	p_{r-1}	…	p_1	p_0	M_{-1}	M_{-2}	…	M_{-n}

Рисунок 2.2 – Формат чисел із плаваючою крапкою зі зсуненим порядком

Для збільшення кількості значущих цифр у зображенні дійсного числа і запобігання переповнення при виконанні арифметичних операцій мантису нормалізують. Нормалізація означає, що значення мантиси має знаходитися в діапазоні від *S*-1 до 1, де *S* – основа системи числення.

У двійковій системі числення *S*=2 і мантиса набуває значення від 2^{-1} до 1. Це означає, що мантиса будь-якого числа, зображеного у форматі з плаваючою крапкою, має починатися з одиниці у двійковій системі числення. Наведений метод нормалізації є класичним, при якому результат нормалізації зображається у вигляді правильного дробу, тобто з одиницею після крапки і нулем у цілій частині числа.

Є різні алгоритми нормалізації мантиси. В IBM-сумісних комп'ютерах старший біт нормалізованої мантиси розташований зліва від крапки. В оперативній пам'яті цей біт не зберігається, тобто він є прихованим, а його вага дорівнює одиниці, тому мантиса належить інтервалу 1 <= M < 2. Нормалізована мантиса додатних і від'ємних чисел зображається у прямому коді.

Діапазон чисел з плаваючою крапкою залежить від кількості розрядів, виділених для зображення порядку і мантиси, а також від основи системи числення. Потрібно зазначити, що розташування та довжини полів у зображеннях дійсних чисел визначаються типом комп'ютера та мовою програмування [2, 5].

2.3.6 Кодування символів

Для виведення інформації на пристрої відображення, наприклад дисплей або принтер, а також для введення або передачі даних використовуються буквено-цифрові коди.

Букви, цифри, математичні символи, розділові знаки, символи для рисування ліній, керівні символи (відміна дії, перенесення каретки принтера), і деякі інші кодуються однобайтовими числами.

Існує декілька різновидів таких таблиць кодів, наприклад: ASCII («аскі» код), КОІ-7, КОІ -8. Загальне число символів в цих кодах дорівнює 256. Нині розроблено і використовується 16-бітовий Unicode з 65536 різними символами.

Фрагмент відповідності ASCII кодів і символів наведено в табл. 2.5.

Таблиця 2.5 – Відповідність ASCII кодів і символів

Код	Символ
20	пропуск
21	!
22	"
23	#
24	$
25	%
26	&
27	'
28	(
29	)
2A	*
2B	+
2C	,
2D	-
2E	.
2F	/
30	0

2.4 Вісімкова та шістнадцяткова системи числення

Двійкове подання чисел є занадто громіздким. Так, для запису десяткового числа 43 потрібно аж 6 двійкових розрядів. Тому при програмуванні використовують позиційну шістнадцяткову й позиційну вісімкову систему числення. Вони використовуються для скороченого запису двійкових кодів.

У вісімковій системі числення як цифри використовують символи: 0, 1, 2, 3, 4, 5, 6, 7.

Приклад:

$$7517_8 = 7\cdot8^3 + 5\cdot8^2 + 6\cdot8^1 + 7\cdot8^0 = 3959_{10}.$$

Оскільки $8 = 2^3$, то переведення чисел з двійкової системи до вісімкової спрощується: одній вісімковій цифрі відповідає три двійкових розряди, причому ця відповідність є взаємно однозначною.

$$011\ 000\ 100\ 010_2 = 3042_2.$$

В шістнадцятковій системі використовують 16 символів – десять арабських цифр і п’ять букв латинського алфавіту, що утворюють послідовність: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Десяткові еквіваленти цифр: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Приклад:

$$1F3B_{16} = 1\cdot16^3 + 15\cdot16^2 + 3\cdot16^1 + 11\cdot16^0 = 7995_{10}.$$

Оскільки $16 = 2^4$, то переведення чисел з двійкової системи до шістнадцяткової спрощується: одній шістнадцятковій цифрі відповідає чотири двійкових розряди, причому ця відповідність є взаємно однозначною.

$$0101\ 0100\ 1111\ 1100_2 = 54FC_{16}.$$

Простіше за все зіставити запис одних і тих же чисел в цих системах числення можна з використанням табл. 2.6.

Таблиця 2.6 – Система числення

10-DEC	2-BIN	8-OCT	16-HEX
0	0	0	0
1	1	1	1
2	01	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11

Безпосереднє переведення цілого числа з десяткової системи числення у будь-яку іншу здійснюється шляхом послідовного ділення числа на основу нової системи числення. Ділення виконується до тих пір, поки остання частка не стане меншою дільника. Отримані остачі від ділення, взяті у зворотному порядку, будуть значеннями розрядів числа в новій системі числення. Остання частка дає старшу цифру числа. Для прикладу переведемо число 687 у шістнадцятковий код:

687 / 16 = 42, остача 15 (F);
42 / 16 = 2, остача 10 (A);
2 / 16 = 0, остача 2 (2).

Перевіримо: $2AF_{16} = 2\cdot16^2 + 10\cdot16^1 + 15\cdot16^0 = 512 + 160 + 15 = 687_{10}$.

2.5 Двійково-десяткова система числення

Ця система отримала значне поширення в сучасних комп'ютерах, зважаючи на легкість переведення в десяткову систему і назад. Вона викорис-

товується там, де основна увага приділяється не простоті технічної побудови машини, а зручності роботи користувача. У цій системі числення всі десяткові цифри окремо кодуються чотирма двійковими цифрами і у такому вигляді записуються послідовно одна за одною. У двійково-десятковій системі числення основою є число 10, але кожна десяткова цифра (0, 1, ..., 9) кодується двійковими цифрами.

Приклад: десяткове число 1726 в двійково-десятковій системі виглядає так: 0001 0111 0010 0110. Відзначимо, що що для переводу можна користуватися табл. 2.1, де двійковий код збігається для десяткових або шістнадцяткових цифр від 0 до 9.

Двійково-десяткова система неекономічна з точки зору реалізації технічної будови машини (приблизно на 20% збільшується необхідне устаткування), але дуже зручна при подготовці завдань і при програмуванні [2 – 5].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які системи числення Ви знаєте?
2. Що таке основа системи числення?
3. У чому відмінності позиційної системи числення і непозиційної?
4. Розкажіть про двійкову систему числення.
5. Які форми подання чисел застосовуються?
6. Які логічні операції Ви знаєте?
7. Наведіть таблицю істинності для логічних операцій.
8. Як виконуються логічні операції над багаторозрядними числами?
9. Які види зсувів бітів застосовуються в МПС?
10. Наведіть приклади арифметичного та циклічного зсуву бітів.
11. Як здійснюється додавання двох багаторозрядних двійкових чисел?
12. Наведіть приклад віднімання двійкових чисел з неохідністю «позичання» одиниці.
13. Поясніть принципи множення та ділення двійкових чисел.
14. Як кодуються від'ємні числа?
15. В яких формах кодуються дробові числа?
16. Переведення дійсних чисел у двійковий код з фіксованою крапкою та обернене перетворення.
17. Як виконуються арифметичні дії в двійковій системі?
18. Які переваги і недоліки десяткової, двійкової, вісімкової і шістнадцяткової систем числення?
19. Наведіть правила переведення з десяткової системи числення в двійкову систему і навпаки.
20. Наведіть правила переведення з двійкової системи числення у вісімкову систему і навпаки.
21. Наведіть правила переведення з двійкової системи числення в шістнадцяткову систему і навпаки.

3 АРХІТЕКТУРА МІКРОПРОЦЕСОРНОЇ СИСТЕМИ

3.1 Апаратний та програмний способи реалізації обчислень

Мікропроцесорна система може розглядатися як окремий випадок електронної системи, що призначена для обробки вхідних сигналів і видачі вихідних сигналів (рис. 3.1).

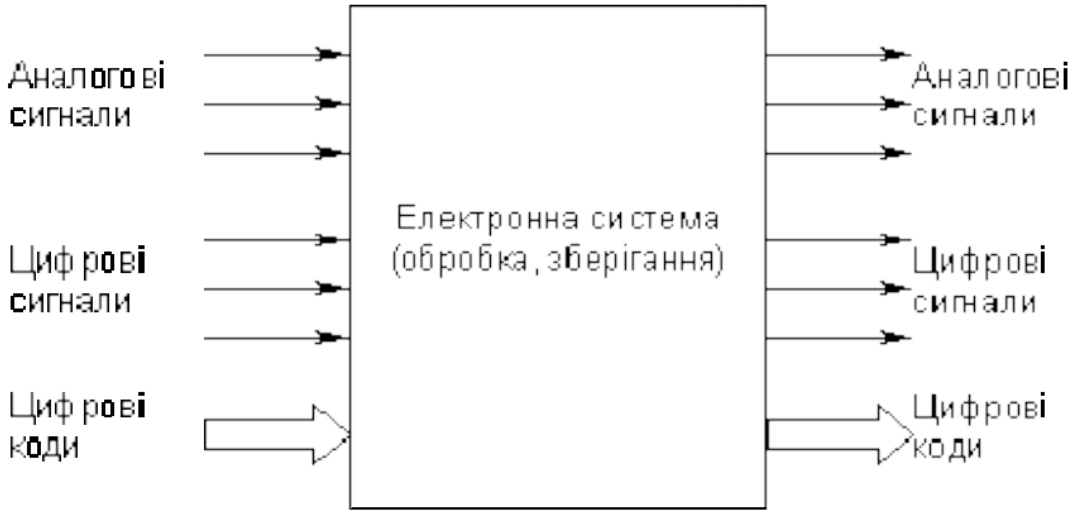


Рисунок 3.1 – Електронна система

Як вхідні і вихідні сигнали при цьому можуть використовуватися аналогові сигнали, одиночні цифрові сигнали, цифрові коди, послідовності цифрових кодів. Усередині системи може відбуватися збереження, накопичення сигналів (чи інформації). Якщо система цифрова (а мікропроцесорні системи відносяться до розряду цифрових), то вхідні аналогові сигнали перетворяться на послідовності кодів вибірок за допомогою аналого-цифрового перетворювача (АЦП), а вихідні аналогові сигнали формуються з послідовності кодів вибірок за допомогою цифро-аналогового перетворювача (ЦАП). Обробка і збереження інформації відбувається в цифровому вигляді.

Характерною рисою традиційної цифрової системи є те, що алгоритми обробки і збереження інформації в ній жорстко пов’язані зі схемотехнікою системи. Тобто зміна цих алгоритмів можлива тільки шляхом зміни структури системи, заміни електронних вузлів, які входять до системи, і/або зв’язків між ними. Наприклад, якщо потрібно реалізувати додаткову операцію додавання, то необхідно додати в структуру системи додатковий суматор. Або якщо потрібна додаткова функція збереження коду протягом одного такту, то потрібно додати в структуру ще один регістр. Природно,

що це практично неможливо зробити в процесі експлуатації, обов'язково потрібний новий виробничий цикл проектування, виготовлення, налагодження всієї системи. Саме тому традиційна цифрова система часто називається системою на «жорсткій логіці».

Будь-яка система на «жорсткій логіці» обов'язково є спеціалізованою системою, яка налаштована винятково на одну задачу чи, рідше, на декілька близьких, заздалегідь відомих задач. Це має свої безперечні переваги.

По-перше, спеціалізована система (на відміну від універсальної) ніколи не має апаратурної надлишковості, тобто кожен її елемент обов'язково працює в повну силу (звичайно, якщо ця система грамотно спроектована).

По-друге, саме спеціалізована система може забезпечити максимально високу швидкодію, тому що швидкість виконання алгоритмів обробки інформації визначається в ній тільки швидкодією окремих логічних елементів і обраною схемою шляхів проходження інформації. А саме логічні елементи завжди мають максимальну на цей момент швидкодію.

Але в той самий час великим недоліком цифрової системи на «жорсткій логіці» є те, що для кожної нової задачі її потрібно проектувати і виготовляти заново. Це процес тривалий, дорогий і потребує високої кваліфікації виконавців. А якщо розв'язувана задача раптом змінюється, то вся апаратура має бути повністю замінена. На сьогодні це є досить марнотратним [1, 2].

Шлях подолання цього недоліку досить очевидний: потрібно побудувати таку систему, що могла б легко адаптуватися під будь-яку задачу, перебудовуватися з одного алгоритму роботи на іншій без зміни апаратури. І задавати той чи інший алгоритм можна було б шляхом введення в систему деякої додаткової керівної інформації, програми роботи системи (рис. 3.2).

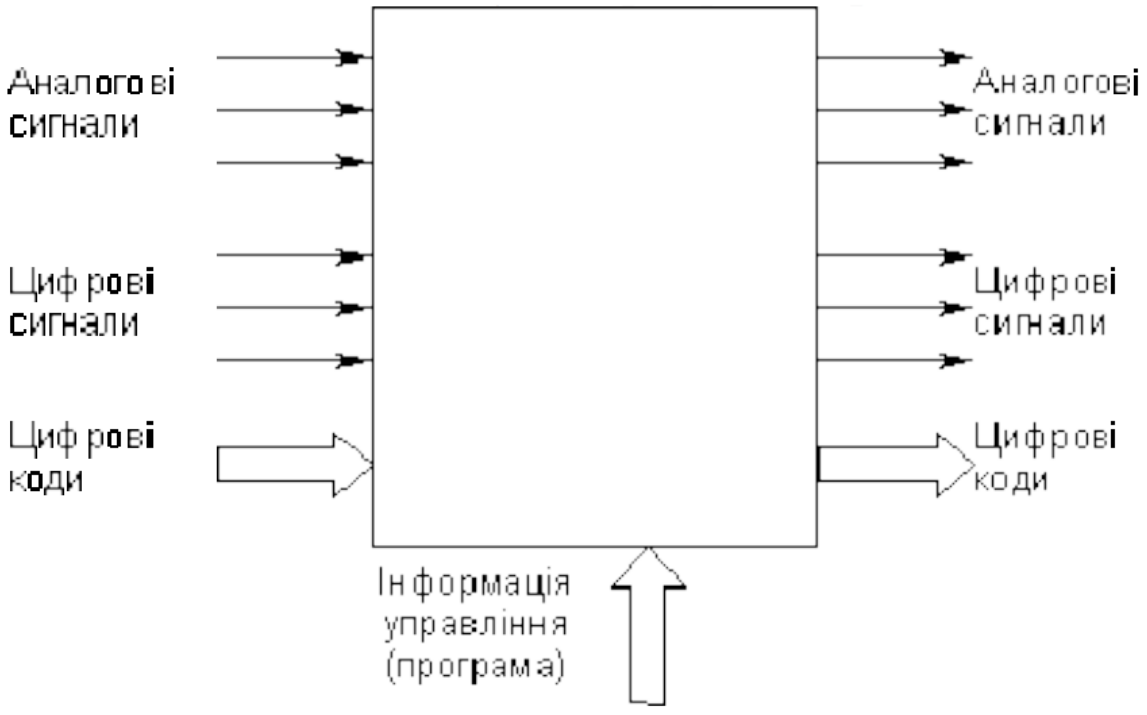


Рисунок 3.2 – Програмована (універсальна) електронна система

Після цього система стає універсальною або програмованою, не жорсткою, а гнучкою. Саме це і забезпечує мікропроцесорна система.

Але будь-яка універсальність обов'язково призводить до надмірності.

Вирішення максимально важкої задачі потребує набагато більше засобів, ніж вирішення максимально простої задачі. Тому складність універсальної системи має бути такою, щоб забезпечувати вирішення найважчої задачі, а при вирішенні простої задачі система буде працювати далеко не на повну силу, буде використовувати не усі свої ресурси. І чим простішою є задача, тим більша надмірність, і тим менш виправданою стає універсальність. Надмірність веде до збільшення вартості системи, зниження її надійності, збільшення споживаної потужності тощо.

Крім того, універсальність, як правило, викликає істотне зниження швидкодії. Оптимізувати універсальну систему так, щоб кожна нова задача вирішувалась максимально швидко, просто неможливо. Загальне правило таке: чим більшою є універсальність, гнучкість, тим меншою є швидкодія. Більше того, для універсальних систем не існує таких задач (нехай навіть і найпростіших), які б вони вирішували з максимально можливою швидкодією.

Таким чином, системи на «жорсткій логіці» доцільні там, де вирішувавна задача не змінюється тривалий час, де потрібно забезпечити найвищу швидкодію, де алгоритми обробки інформації гранично прості. А універсальні, програмовані системи доцільні там, де часто змінюються поставлені задачі, де висока швидкодія не надто важлива, де алгоритми обробки інформації складні. Тобто будь-яка система має бути на своєму місці.

Однак за останні десятиліття швидкодія універсальних (мікропроцесорних) систем значно виросла (на кілька порядків). До того ж великий обсяг випуску мікросхем для цих систем викликав різке зниження їх вартості. Як результат – галузі застосування систем на «жорсткій логіці» різко звузилися. Більше того, високими темпами розвиваються зараз програмовані системи, які призначені для вирішення однієї задачі або декількох близьких задач. Вони вдало поєднують у собі переваги як систем на «жорсткій логіці», так і програмованих систем, забезпечуючи поєднання досить високої швидкодії і необхідної гнучкості [2].

3.2 Склад і функції мікропроцесора

Ядром будь-якої мікропроцесорної системи є мікропроцесор або просто процесор, тобто вузол, який робить всю обробку інформації усередині МПС. Інші вузли виконують усього лише допоміжні функції: збереження інформації (у тому числі і керівної інформації, тобто програми), зв'язку з зовнішніми пристроями, зв'язку з користувачем тощо. Процесор замінює практично всю «жорстку логіку», що знадобилася б у випадку традиційної цифрової системи. Він виконує арифметичні функції (додавання, множення тощо), логічні функції (зсуву, порівняння,

маскування кодів тощо), тимчасове збереження кодів (у внутрішніх регістрах), пересилання кодів між вузлами мікропроцесорної системи і багато чого іншого. Кількість таких елементарних операцій, що виконуються процесором, може досягати декількох сотень.

Але при цьому потрібно враховувати, що усі свої операції процесор виконує послідовно, тобто одну за одною, по черзі. Звичайно, існують процесори з паралельним виконанням деяких операцій, зустрічаються також мікропроцесорні системи, у яких кілька процесорів працюють над однією задачею паралельно, але це рідкісні винятки. З одного боку, послідовне виконання операцій – безсумнівна перевага, тому що дозволяє за допомогою тільки одного процесора виконувати будь-які найскладніші алгоритми обробки інформації. Але, з іншого боку, послідовне виконання операцій призводить до того, що час виконання алгоритму залежить від його складності. Прості алгоритми виконуються швидше складних. Тобто мікропроцесорна система здатна зробити все, але працює вона не надто швидко, адже всі інформаційні потоки доводиться пропускати через один-єдиний вузол – мікропроцесор (рис. 3.3).

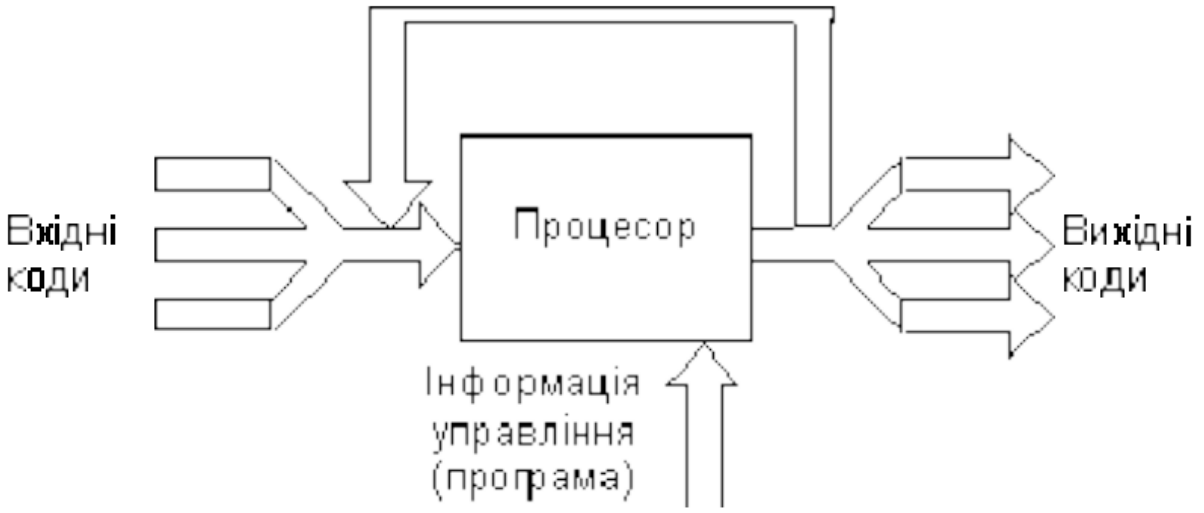


Рисунок 3.3 – Інформаційні потоки в мікропроцесорній системі

У традиційній цифровій системі можна легко організувати паралельну обробку всіх потоків інформації, щоправда, ціною ускладнення схеми [1].

Мікропроцесор здатний виконувати безліч операцій. Це визначається керівною інформацією – програмою. Це набір команд (інструкцій), тобто цифрових кодів, розшифрувавши які процесор визначає, що йому потрібно робити.

Усі команди, які можуть виконуватись процесором, утворюють систему команд процесора. Структура й обсяг системи команд процесора визначають його швидкодію, гнучкість, зручність використання. Усього команд у процесора може бути від декількох десятків до декількох сотень. Система команд може бути розрахована на вузьке коло задач (у спеціалізованих

процесорів) або на максимально широке коло задач (в універсальних процесорів). Коди команд можуть мати різну кількість розрядів (займати від одного до декількох байтів). Кожна команда має свій час виконання, тому час виконання всієї програми залежить не тільки від кількості команд у програмі, але і від того, які саме команди використовуються.

Для виконання команд у структуру процесора входять внутрішні регістри, арифметико-логічний пристрій (АЛП), мультиплексори, буфери, регістри й інші вузли. Робота усіх вузлів синхронізується загальним зовнішнім тактовим сигналом процесора. Тобто процесор є досить складним цифровим пристроєм (рис. 3.4).

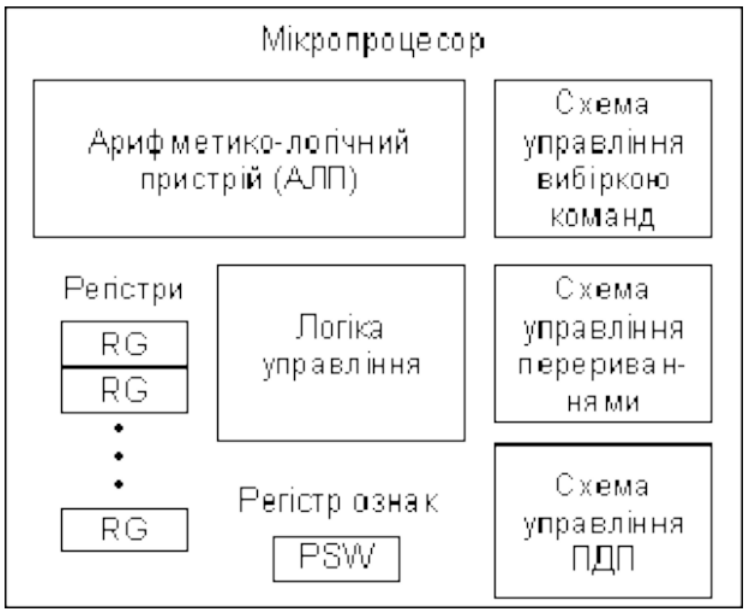


Рисунок 3.4 – Структура найпростішого процессора

3.3 Базова структура мікропроцесорної системи

Типова базова структура мікропроцесорної системи наведена на рис. 3.5.

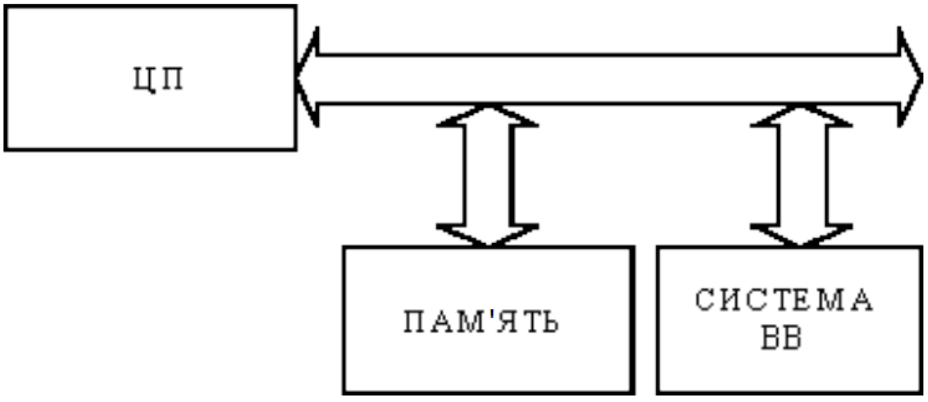


Рисунок 3.5 – Базова структура МПС

Вона містить у собі три основних типи пристроїв:

- ЦП – центральний процесор;
- пристрої пам'яті – оперативну запам'ятовувальну пам'ять (ОЗП) і постійну запам'ятовувальну пам'ять (ПЗП), що слугує для збереження даних і програм;
- ВВ – системи пристроїв введення/виведення, які слугують для зв'язку мікропроцесорної системи з зовнішніми пристроями, для прийому (введення, читання) вхідних сигналів і видачі (виведення, запису) вихідних сигналів.

Завдання управління системою покладається на центральний процесор, який пов'язаний з пам'яттю, і систему введення/виведення через канали пам'яті і введення/виведення, відповідно. ЦП прочитує з пам'яті команди, які утворюють програму і декодує їх. Відповідно до результату декодування команд він здійснює вибірку даних з пам'яті портів введення, обробляє їх і пересилає назад в пам'ять або порти виведення. Існує також можливість введення/виведення даних з пам'яті на зовнішні пристрої і назад в обхід ЦП. Цей механізм називається прямим доступом до пам'яті (ПДП).

Кожна складова частина мікропроцесорної системи має досить складну внутрішню структуру. Важливо враховувати, що пристрої введення/виведення зазвичай є пристроями на «жорсткій логіці». На них може бути покладена частина функцій, яка виконується мікропроцесорною системою. Тому у виробника завжди є можливість перерозподіляти функції системи між апаратною і програмною реалізаціями оптимальним чином. Апаратна реалізація прискорює виконання функцій, але має недостатню гнучкість. Програмна реалізація значно повільніша, але забезпечує високу гнучкість. Апаратна реалізація функцій збільшує вартість системи і її енергоспоживання, програмна – не збільшує.

Зазвичай застосовують комбінування апаратних і програмних функцій.

Іноді пристрої введення/виведення мають у своєму складі процесор, тобто є невеликою спеціалізованою мікропроцесорною системою. Це дозволяє перекласти частину програмних функцій на пристрої введення/виведення, розвантаживши центральний процесор системи [2, 3].

3.4 Поняття архітектури МПС

З точки зору користувача при виборі мікропроцесора доцільно мати в своєму розпорядженні деякі узагальнені комплексні характеристики можливостей мікропроцесора. Розробник потребує з'ясування і розуміння лише тих компонентів мікропроцесора, які явно відбиваються в програмах і мають бути враховані при розробці схем та програм функціонування системи. Такі характеристики визначаються архітектурою мікропроцесора.

Архітектура мікропроцесора – це його логічна організація, що розглядається з точки зору користувача; вона визначає можливості мікропроцесора з апаратної і програмної реалізації функцій, необхідних для побудови

мікропроцесорної системи [2]. Поняття архітектури мікропроцесора відображає:

- його структуру, тобто сукупність компонентів, складових мікропроцесора, і зв'язків між ними;
- формати даних;
- способи звернення до всіх програмно-доступних для користувача елементів структури (адресація до регістрів, елементів постійної і оперативної пам'яті, зовнішніх пристроїв);
- набір операцій, що виконуються мікропроцесором;
- характеристики керівних слів і сигналів, які виробляються мікропроцесором і які надходять на нього ззовні;
- реакцію на зовнішні сигнали (система обробки переривань і тощо).

За способом організації простору пам'яті мікропроцесорної системи розрізняють два основних типи архітектури. В будь-якому разі взаємодія з пам'яттю здійснюється по паралельно прокладених провідниках, які називають «шина». По кожному провіднику передається логічний сигнал «0» або «1».

Організація, при якій для зберігання програм і даних використовується один простір пам'яті, називається фон-нейманівською архітектурою. Програми і дані зберігаються в єдиному просторі, і немає жодних ознак, які вказують на тип інформації в елементі пам'яті (рис. 3.6).

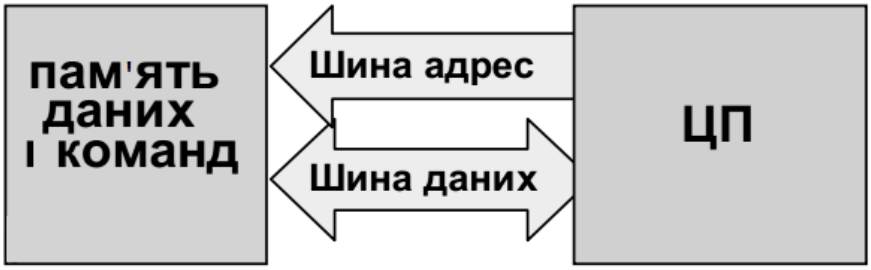


Рисунок 3.6 – Функціональна схема МПС з архітектурою фон Неймана

У архітектурі фон Неймана один набір шин для пам'яті програм і пам'яті даних. Для виконання арифметичної операції потрібно 4 фази:

1 – читання коду команди;
2 – читання першого операнда;
3 – читання другого операнда;
4 – запис результату.

Представниками такої архітектури є перші персональні комп'ютери.

Організація, при якій пам'ять програм ПП і пам'ять даних ПД розділені і мають свої власні адресні простори і способи доступу до них, називається гарвардською архітектурою. Обсяг ПД, як правило, значно менше обсягу ПП.

Така архітектура є складнішою і потребує додаткових керівних сигналів (рис. 3.7).

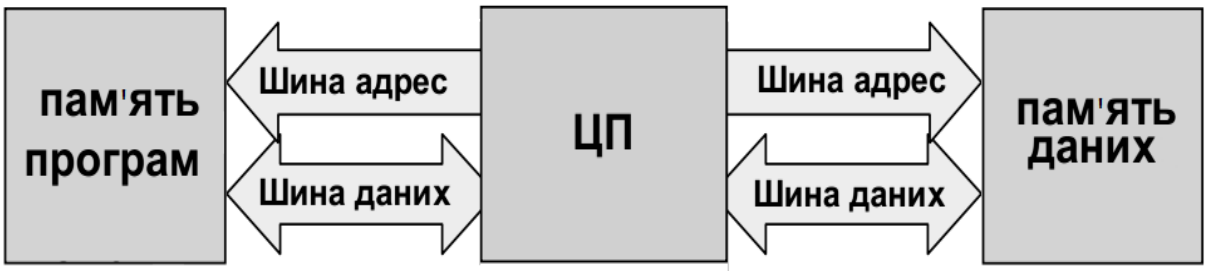


Рисунок 3.7 – Функціональна схема МПС з Гарвардською архітектурою

Проте, вона дозволяє здійснювати гнучкіші маніпуляції з інформацією, реалізовувати компактно кодований набір машинних команд і, у деяких випадках, прискорювати роботу мікропроцесора.

Головними відмінними рисами гарвардської архітектури організації пам’яті контроллера є:

- реалізація у вигляді різних пристроїв пам’яті для програм і пам’яті для даних;
- використання двох паралельних працюючих незалежних шин для читання даних і команд.

У гарвардській архітектурі два набори шин. Один набір – для добування кодів команди, інший – для добування даних. В цьому випадку для тієї ж самої операції буде потрібно лише 3 такти, тому що добування кодів команди і першого операнда може відбуватися одночасно. Представниками такої архітектури є мікроконтролери багатьох фірм.

Порівняємо розглянуті типи архітектур.

Основні переваги фон-нейманівської архітектури – це простота апаратної реалізації та універсальність виконання команд. Вона не потребує від процесора одночасного обслуговування двох шин, контролю обміну двома шинами відразу. Наявність єдиної пам’яті даних і команд дозволяє гнучко розподіляти її обсяг між кодами даних і команд.

Наприклад, у деяких випадках потрібна велика і складна програма, а даних у пам’яті треба зберігати не надто багато. В інших випадках, навпаки, програма проста, але необхідні великі обсяги збережених даних. Перерозподіл пам’яті не викликає ніяких проблем, головне – щоб програма і дані разом вміщувалися в пам’яті системи. Як правило, у системах з такою архітектурою пам’ять буває досить великого обсягу (до десятків і сотень мегабайтів). Це дозволяє вирішувати самі складні задачі.

Гарвардська архітектура з роздільними шинами даних і команд складніша, вона змушує процесор працювати одночасно з двома потоками кодів, обслуговувати обмін двома шинами одночасно. Програма може розміщуватися тільки в пам’яті команд, дані – тільки в пам’яті даних. Така вузька спеціалізація обмежує коло задач, які вирішуються системою, тому що не дає можливості гнучкого перерозподілу пам’яті. Пам’ять даних і пам’ять команд у цьому випадку мають порівняно невеликий обсяг, тому застосу-

вання систем з такою архітектурою обмежується переважно не надто складними задачами.

Перевагою гарвардської архітектури, в першу чергу, є швидкодія. Справа в тому, що при єдиній шині команд і даних процесор по одній цій шині має приймати дані (з пам'яті або пристрою введення/виведення) і передавати дані (у пам'ять або на пристрій введення/виведення), а також читати команди з пам'яті. Природно, що одночасно ці пересилання кодів магістралями відбуватися не можуть, вони мають відбуватися по черзі.

Сучасні процесори здатні поєднати в часі виконання команд і проведення циклів обміну системною шиною. Використання конвеєрних технологій і швидкої кеш-пам'яті дозволяє їм прискорити процес взаємодії з порівняно повільною системною пам'яттю. Підвищення тактової частоти й удосконалення структури процесорів дають можливість скоротити час виконання команд. Але подальше збільшення швидкодії системи можливе тільки при поєднанні пересилання даних і читання команд, тобто при переході до архітектури з двома шинами.

У випадку двошинної архітектури обмін між двома шинами може бути незалежним, паралельним у часі. Відповідно, структури шин (кількість розрядів коду адреси і коду даних, порядок і швидкість обміну інформацією тощо) можуть бути обрані оптимально для тієї задачі, яка виконується кожною шиною. Тому за інших рівних умов перехід на двошинну архітектуру прискорює роботу мікропроцесорної системи, хоча і потребує додаткових витрат на апаратуру, ускладнення структури процесора. Пам'ять даних у цьому випадку має свій розподіл адрес, а пам'ять команд – свій.

Найпростіше переваги двошинної архітектури реалізуються всередині однієї мікросхеми. У цьому випадку можна також істотно зменшити вплив недоліків цієї архітектури. Тому основне її застосування – у мікроконтролерах, від яких не потрібно вирішення надто складних задач, але зате необхідна максимальна швидкодія при заданій тактовій частоті [1 – 3].

Нині випускаються мікропроцесори зі змішаною архітектурою, в яких пам'ять програм CSEG і пам'ять даних DSEG мають єдиний адресний простір, проте різні механізми доступу до них. Конкретним прикладом є мікропроцесори сімейства 80×86 фірм Intel [4].

3.5 Шинна структура зв'язків МПС

Для досягнення максимальної універсальності і спрощення протоколів обміну інформацією в мікропроцесорних системах застосовується так звана шинна структура зв'язків між окремими пристроями, що входять у систему.

При класичній структурі зв'язків (рис. 3.8) усі сигнали і коди між пристроями передаються окремими лініями зв'язку.

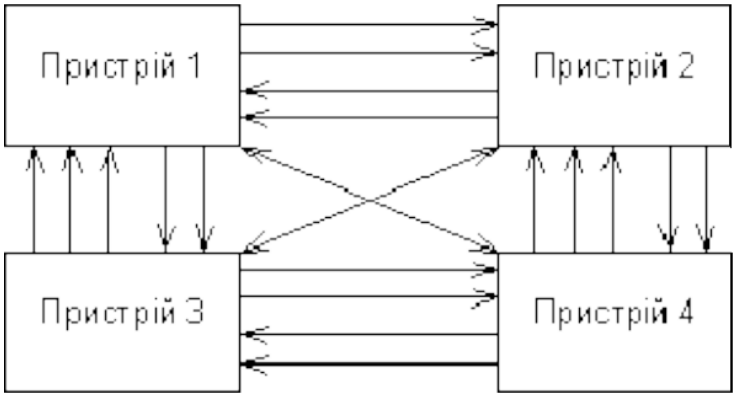


Рисунок 3.8 – Класична структура зв’язків

Кожен пристрій, який входить до системи, передає свої сигнали і коди незалежно від інших пристроїв. При цьому в системі виходить дуже багато ліній зв’язку і різних протоколів обміну інформацією.

При шинній структурі зв’язків (рис. 3.9) усі сигнали між пристроями передаються одними і тими ж лініями зв’язку, але в різний час (це називається мультиплексованою передачею).

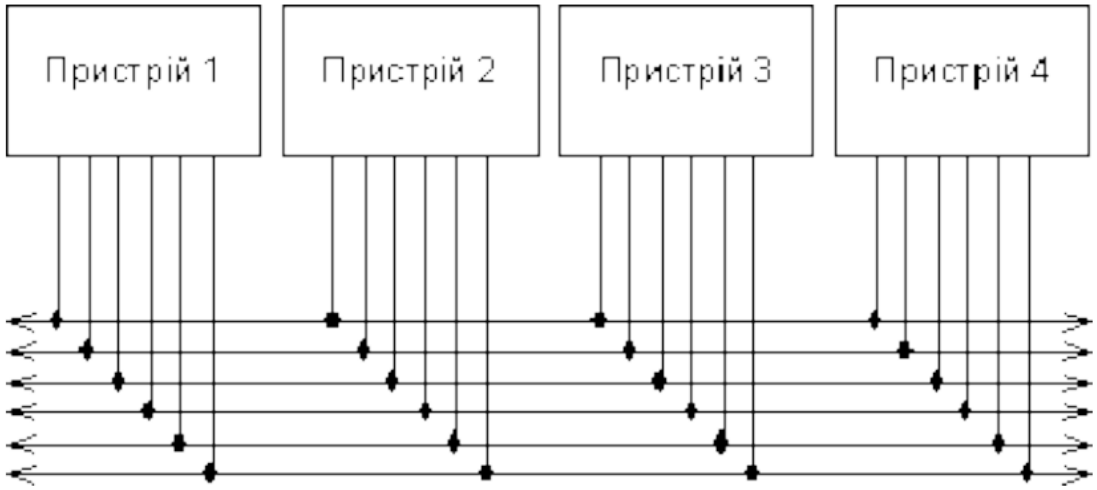


Рисунок 3.9 – Шинна структура зв’язків

Причому передача всіма лініями зв’язку може здійснюватися в обидва напрямки (двонапрямлена передача). У результаті кількість ліній зв’язку істотно скорочується, а правила обміну (протоколи) спрощуються. Група ліній зв’язку, якими передаються сигнали або коди, називається шиною.

При шинній структурі зв’язків легко здійснюється пересилання всіх інформаційних потоків у потрібному напрямку, наприклад, їх можна пропустити через один процесор, що дуже важливо для мікропроцесорної системи. Однак при шинній структурі зв’язків уся інформація передається лініями зв’язку послідовно в часі, по черзі, що знижує швидкодію системи порівняно з класичною структурою зв’язків.

Велика перевага шинної структури зв'язків полягає в тому, що всі пристрої, які під'єднані до шини, мають приймати і передавати інформацію за тими самими правилами (протоколам обміну інформацією із шини).

Відповідно, усі вузли, які відповідають за обмін із шиною в цих пристроях, мають бути однаковими, тобто уніфіковані.

Істотний недолік шинної структури пов'язаний з тим, що всі пристрої під'єднуються до кожної лінії зв'язку паралельно. Тому будь-яка несправність будь-якого пристрою може вивести з ладу всю систему, якщо вона псує лінію зв'язку. З цієї ж причини налагодження системи із шинною структурою зв'язків досить складне і, зазвичай, потребує спеціального устаткування.

У системах із шинною структурою зв'язків застосовують усі три існуючі різновиди вихідних каскадів цифрових мікросхем:

- стандартний вихід або вихід із двома станами (позначається 2С, 2S, рідше ТТЛ, TTL);
- вихід з відкритим колектором (позначається ВК, ОК, ОС);
- вихід із трьома станами або з можливістю відключення (позначається 3С, 3S).

Спрощено ці три типи вихідних каскадів можуть бути подані у виді схем (рис. 3.10).

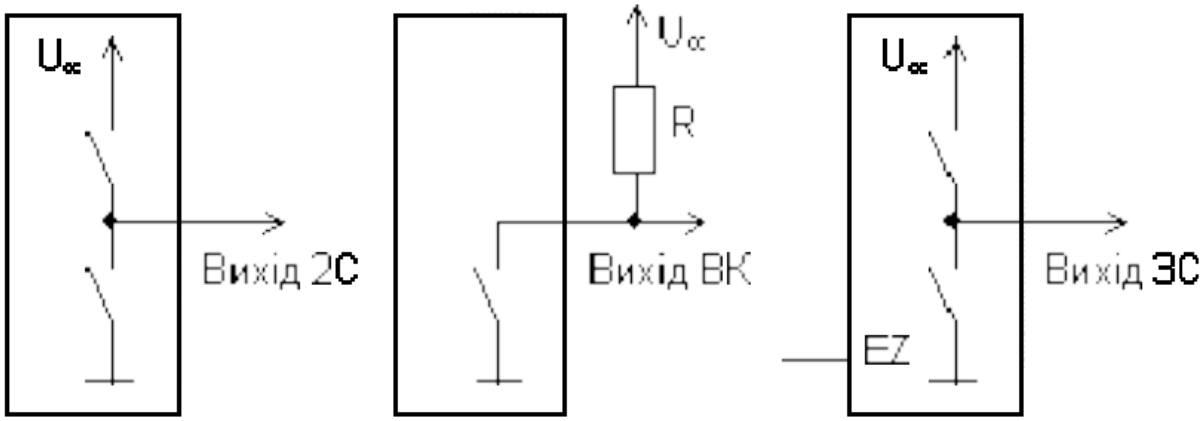


Рисунок 3.10 – Три типи виходів цифрових мікросхем

У виходу 2С два ключі замикаються по черзі, що відповідає рівням логічної одиниці (верхній ключ замкнуто) і логічного нуля (нижній ключ замкнуто). У виходу ВК замкнутий ключ формує рівень логічного нуля, розімкнутий – логічної одиниці. На виході 3С ключі можуть замикатися по черзі (як у випадку 2С), а можуть розмикатися одночасно, утворюючи тим самим третій, високоімпедансний стан. Перехід у третій стан (Z-стан) керується сигналом на спеціальному вході EZ.

Вихідні каскади типів 3С та ВК дозволяють з'єднувати кілька виходів мікросхем для отримання мультиплексованих (рис. 3.11) або двонапрямлених (рис. 3.12) ліній.

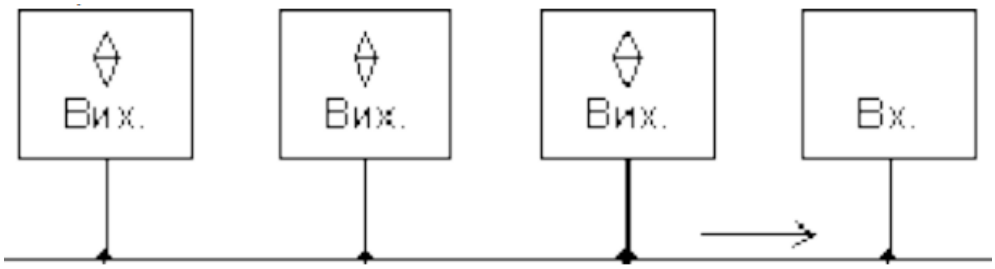


Рисунок 3.11 – Мультиплексована лінія

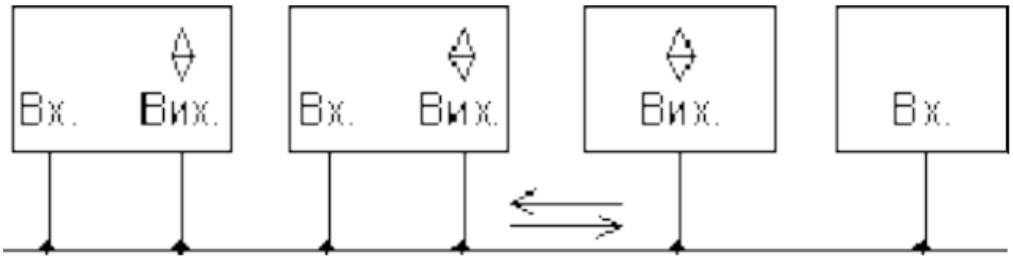


Рисунок 3.12 – Двонапрямлена лінія

При цьому у випадку виходів 3С необхідно забезпечити, щоб на лінії завжди працював тільки один активний вихід, а всі інші виходи знаходилися у цей час у третьому стані, інакше можливі конфлікти. Об'єднані виходи ВК можуть працювати всі одночасно, без усяких конфліктів [3].

3.5.1 Види системних магістралей

На фізичному рівні МПС мікропроцесор взаємодіє з пам'яттю і системою введення/виведення через єдиний набір системних шин – системну магістраль (загальна системна шина або канал). Вона в загальному випадку складається з:

- шини даних DB (Data Bus) – це основна шина, по якій відбувається обмін даними між ЦП, пам'яттю і системою ВВ. Переважно в пересиланні інформації бере участь процесор, що передає код даних у якийсь пристрій або в комірку пам'яті чи приймає код даних з якогось пристрою або з комірки пам'яті. Але можлива також і передача інформації між пристроями без участі процесора. Шина даних завжди двонапрямлена;
- шини адреси АВ (Address Bus), яка використовується для передачі адрес елементів пам'яті, і портів ВВ, до яких здійснюється звернення. Служить для визначення адреси (номера) пристрою, з яким процесор обмінюється інформацією в цей момент. Кожному пристрою (крім процесора), кожній комірці пам'яті в мікропроцесорній системі присвоюється власна адреса. Коли код якоїсь адреси виставляється процесором на шині адреси, пристрій, якому ця адреса приписана, визначає, що з ним зараз передбачається обмін інформацією. Шина адреси може бути однонапрямленою або двонапрямленою;

- шини управління СВ (Control Bus), по яких передаються сигнали управління, реалізовують цикли обміну інформацією і керують роботою системи. Шина управління, на відміну від шини адреси і шини даних, складається з окремих керівних сигналів. Кожний з цих сигналів під час обміну інформацією має свою функцію. Деякі сигнали служать для строба переданих або прийнятих даних (тобто визначають моменти часу, коли інформаційний код виставлений на шину даних). Інші керівні сигнали можуть використовуватися для підтвердження прийому даних, для скидання всіх пристроїв у вихідний стан (ініціалізації), для тактування всіх пристроїв тощо. Лінії шини управління можуть бути однонапрямленими або двонапрямленими. Виділяють такі основні сигнали: RD – читання з пам'яті; WD – записування в пам'ять; IORD – читання з порту; IOWD – записування в порт;

- шина живлення призначена не для пересилання інформаційних сигналів, а для живлення системи. Вона складається з ліній живлення і загального провідника. У мікропроцесорній системі може бути одне джерело живлення (частіше +5 В) або кілька джерел живлення (переважно -5 В, +12 В та -12 В). Кожній напрузі живлення відповідає своя лінія зв'язку. Усі пристрої під'єднані до цих ліній паралельно.

Цей самий набір шин застосовується для організації каналу ПДП. Магістраль такого типу носить назву тришинної з окремими шинами адреси і даних або демультиплексну (рис. 3.13).

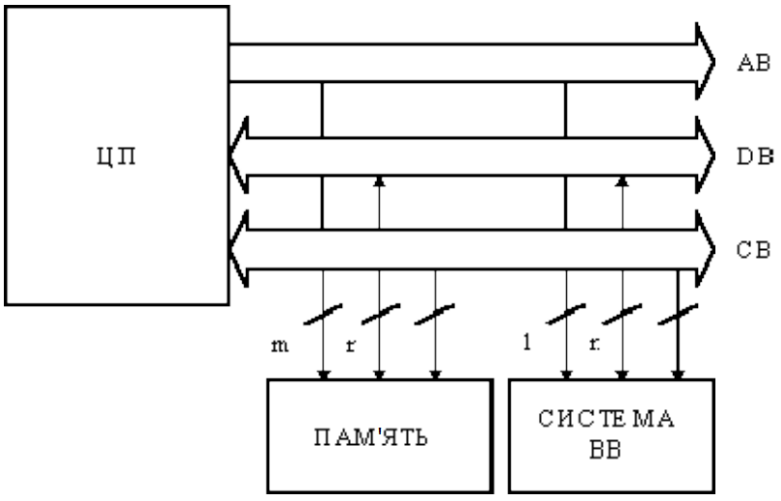


Рисунок 3.13 – МПС з демультиплексною магістраллю

У деяких мікропроцесорах з метою скорочення ширини фізичної магістралі вводять поєднану шину адреси-даних AD (Address/Data Bus), по якій передаються як адреси, так і дані. Етап передачі адресної інформації відокремлений за часом від етапу передачі даних і стробується спеціальним сигналом ALE (Address Latch Enable), який внесено до складу СВ. Таку магістраль зазвичай називають двошинною з поєднаними шинами адреси і даних або мультиплексною (рис. 3.14).

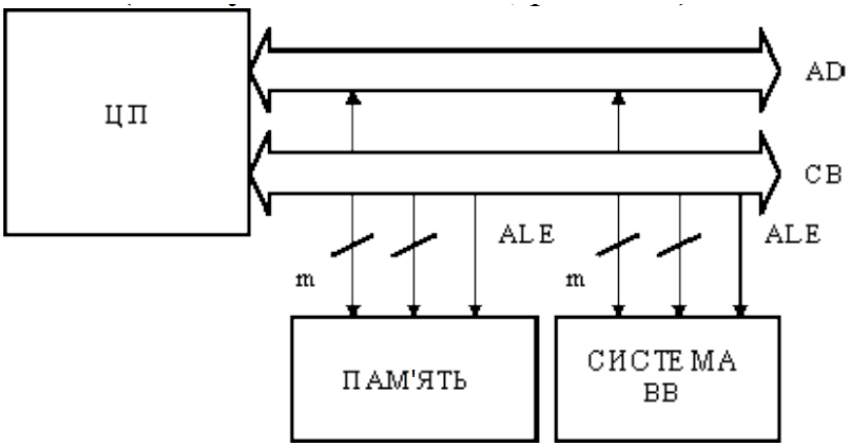


Рисунок 3.14 – МПС з мультиплексною магістраллю

Фізичний обмін даними через магістраль виконується байтами або словами у вигляді наступних один за одним звернень до каналу. За один цикл звернення до магістралі між ЦП, пам’яттю і системою ВВ передається одне слово або байт. Існують декілька типових циклів обміну. Серед них читання пам’яті і записування в пам’ять.

Обмін ЦП з пам’яттю демонструється рис. 3.15 – 3.18.

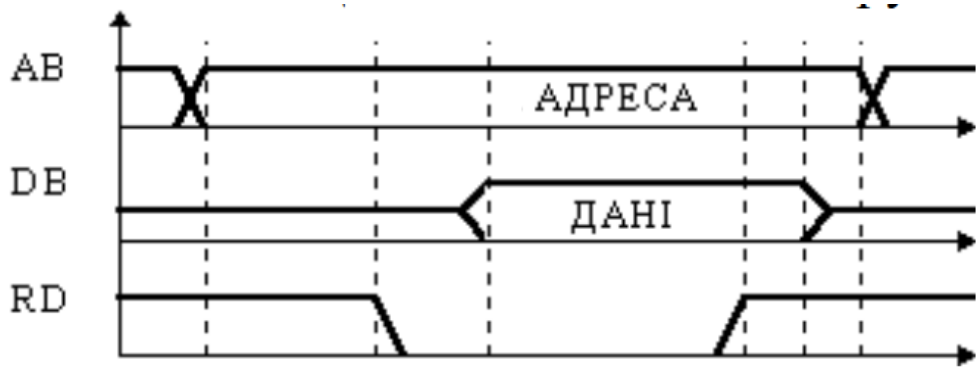


Рисунок 3.15 – Цикл читання пам’яті по демультиплексній магістралі

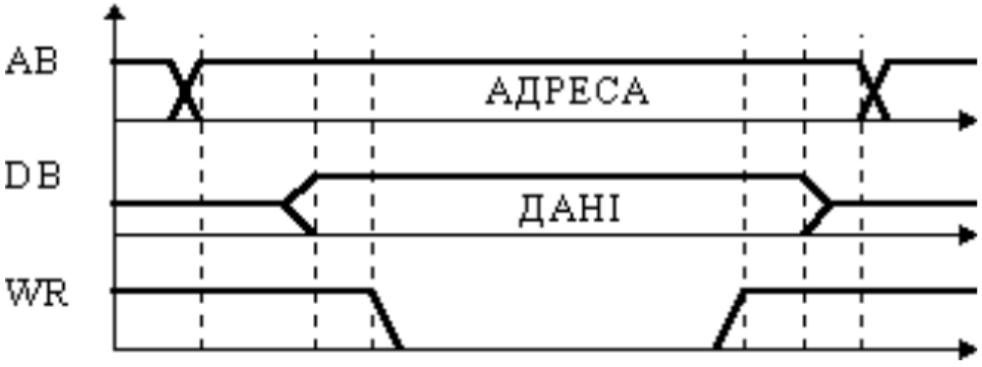


Рисунок 3.16 – Цикл записування в пам’ять по демультиплексній магістралі

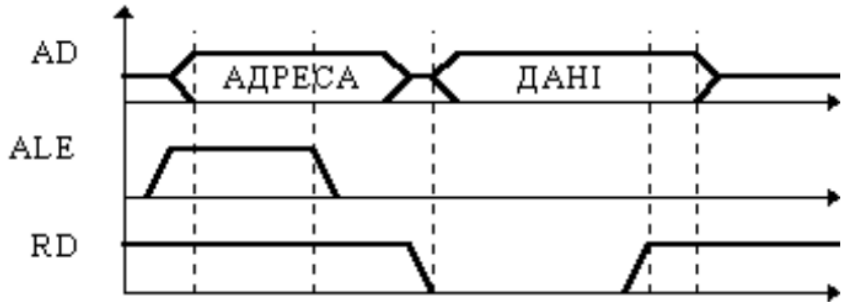


Рисунок 3.17 – Цикл читання пам’яті по мультиплексній магістралі

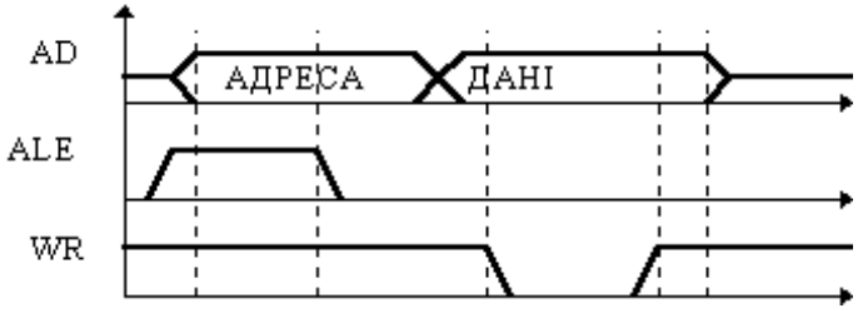


Рисунок 3.18 – Цикл записування в пам’ять по мультиплексній магістралі

При ізольованому просторі ВВ додаються цикли читання з порту ВВ і записування в порт ВВ (рис. 3.19 та 3.20).

Рисунок 3.19 – Цикл читання з порту ВВ по демультиплексній магістралі

Рисунок 3.20 – Цикл запису в порт ВВ по демультиплексній магістралі

В разі архітектури гарвардського типу, коли пам’ять програм і пам’ять даних розділені, вводиться також цикл читання пам’яті програм, що демонструє рис. 3.21.

Рисунок 3.21 – Цикл читання пам’яті програм по демультиплексній магістралі

В деяких випадках, коли на магістралі працюють пристрої, швидкодія яких поступається швидкодії ЦП, тривалості стробів RD та WR можуть виявитися недостатніми для правильного виконання операції обміну з боку периферійного модуля. Тоді для організації надійного завершення магістральної операції до складу СВ вводять спеціальний сигнал READY. У кожному циклі звернення до каналу перед закінченням строба RD або WR ЦП перевіряє стан сигналу READY (рис. 3.22).

Рисунок 3.22 – Цикл читання з циклами чекання

Якщо він до цього моменту ще не скинутий, то ЦП подовжує відповідний строб, вставляючи в нього такти чекання WS (Wait State). Максимальна кількість WS може бути обмежена або необмежена залежно від конкретної моделі мікропроцесора і режиму його роботи.

Якщо в мікропроцесорну систему потрібно увести вхідний код або вхідний сигнал, то процесор шиною адреси звертається до потрібного пристрою введення/виведення і приймає шиною даних вхідну інформацію. Якщо з мікропроцесорної системи потрібно вивести вихідний код або вихідний сигнал, то процесор звертається шиною адреси до потрібного пристрою введення/виведення і передає йому шиною даних вихідну інформацію.

Якщо інформація має пройти складну багатоступінчасту обробку, то процесор може зберігати проміжні результати в системній оперативній пам’яті. Для звернення до будь-якої комірки пам’яті процесор передає її адресу на шину адреси і надсилає до неї інформаційний код шиною даних або ж приймає з неї інформаційний код шиною даних. У пам’яті (оперативній і постійній) знаходяться також і керівні коди (команди, що виконуються процесором програми), які процесор також зчитує шиною даних з адресацією шиною адреси. Постійна пам’ять використовується, в основному, для збереження програми початкового пуску мікропроцесорної системи, що виконується щоразу після увімкнення живлення. Інформація в неї заноситься виробником раз і назавжди.

Таким чином, у мікропроцесорній системі всі інформаційні коди і коди команд передаються шинами послідовно, по черзі. Це визначає порівняно невисоку швидкодію мікропроцесорної системи. Вона обмежена зазвичай навіть не швидкодією процесора, яка теж є дуже важливою, і не швидкістю обміну системною шиною (магістраллю), а саме послідовним характером передачі інформації із системної шини (магістралі) [2, 3].

3.6 Режими роботи МП

Як уже відзначалося, мікропроцесорна система забезпечує високу гнучкість роботи, вона здатна програмуватись на будь-яку задачу. Гнучкість ця обумовлена насамперед тим, що функції, виконувані системою, визначаються програмою (програмним забезпеченням, software), яку виконує процесор.

Апаратура (апаратне забезпечення, hardware) залишається незмінною за будь-якої задачі. Записуючи в пам’ять системи програму, можна змусити мікропроцесорну систему виконувати будь-яку задачу, що підтримується цією апаратурою. До того ж шинна організація зв’язків мікропроцесорної системи дозволяє досить легко замінювати апаратні модулі, наприклад, замінювати пам’ять на нову більшого обсягу або вищої швидкодії, додавати або модернізувати пристрої введення/виведення, нарешті, замінювати процесор на потужніший. Це також дозволяє збільшити гнучкість системи, продовжити її життя при будь-якій зміні вимог до неї.

Гнучкість мікропроцесорної системи визначається не тільки цим, але і вибором режиму роботи системи, тобто режиму обміну інформацією із системною магістраллю (шиною).

Практично будь-яка розвинута мікропроцесорна система (зокрема і комп’ютер) підтримує три основних режими обміну магістраллю:

- програмний обмін інформацією;
- обмін з використанням переривань;
- обмін з використанням прямого доступу до пам’яті [1].

3.6.1 Програмний обмін інформацією

Програмний обмін інформацією є основним у будь-якій мікропроцесорній системі (рис. 3.23).

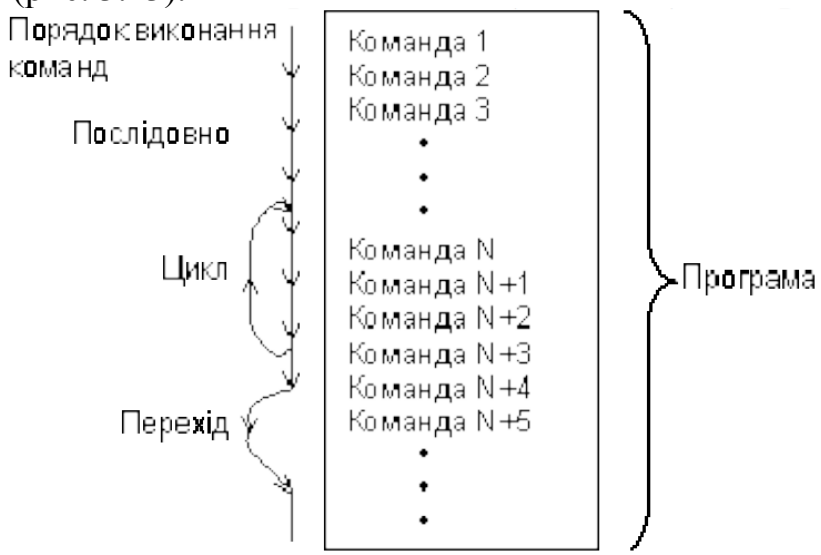


Рисунок 3.23 – Програмний обмін інформацією

Він передбачений завжди, без нього неможливі інші режими обміну. Всі операції (цикли) обміну інформацією в цьому випадку ініціюються тільки процесором, усі вони виконуються строго в порядку, запропонованому виконуваною програмою.

Процесор вибирає з пам'яті коди команд і виконує їх, зчитуючи дані з пам'яті або з пристрою введення/виведення, обробляючи їх, записуючи дані в пам'ять або передаючи їх у пристрій введення/виведення. Шлях процесора по програмі може бути лінійним, циклічним, може містити переходи (стрибки), але він завжди безупинний і цілком знаходиться під контролем процесора. На зовнішні події, не пов'язані з програмою, процесор не реагує [2].

3.6.2 Режим переривання

Іншим режимом роботи ЦП, що потребує від нього зміни нормального ходу виконання програми, є режим «переривання» (Interrupt).

Обмін з використанням переривань використовується тоді, коли необхідна реакція мікропроцесорної системи на якусь зовнішню подію, на прихід зовнішнього сигналу. У випадку комп'ютера зовнішньою подією може бути, наприклад, натискання на клавіші клавіатури або прихід локальною мережею пакета даних. Комп'ютер має реагувати на це, відповідно, виведенням символу на екран або ж читання й обробкою прийнятого з мережі пакета.

Практично всі сучасні мікропроцесори мають один або декілька входів зовнішніх переривань INT0, INT1, і так далі, на які надходять сигнали, що свідчать про деякі події в системі, на які ЦП має відреагувати певним чи-

ном. Під час виникнення активного рівня сигналу на один з таких входів мікропроцесор переривається нормальне виконання програми, запам'ятовує адресу команди, на якій він перервав роботу, переходить до виконання «підпрограми обробки переривання» (ПОП), записаної в CSEG за певною адресою. Адреса цієї підпрограми записана в спеціальному елементі пам'яті – «векторі переривання». Кожне окреме джерело переривання має свій власний вектор переривання.

Після виконання ПОП, за спеціальною командою, якою має закінчуватися ПОП, процесор повертається до виконання перерваної програми за адресою, що раніше запам'ятав.

Джерела переривань можуть бути як зовнішніми (тобто надходити на один з входів мікросхеми, які називаються «входами запиту переривання»), так і внутрішніми (тобто генеруватися усередині процесора за певними умовами).

Оскільки одночасно можуть надійти декілька різних запитів переривань, то існує певна процедура, що задає послідовність обслуговування окремих переривань. Цю процедуру забезпечує система «пріоритетного арбітражу переривань», реалізована або усередині ЦП, або за допомогою спеціального контролера пріоритетних переривань. Відповідно до цієї системи кожне джерело переривання має свій заданий пріоритет (постійний або змінний), що визначає черговість його обслуговування. При одночасному надходженні декількох запитів переривань спочатку обслуговується переривання з вищим пріоритетом, а потім з нижчим. Переривання з вищим пріоритетом може перервати підпрограму обробки переривання, що вже почалася, та має нижчий пріоритет, точно так, як воно перериває і основну програму. При цьому утворюються «вкладені переривання».

У загальному випадку організувати реакцію на зовнішню подію можна трьома різними шляхами:

- за допомогою постійного програмного контролю факту настання події (метод опитування прапорця або polling);
- за допомогою переривання, тобто примусового переведення процесора з виконання поточної програми на виконання необхідної програми;
- за допомогою прямого доступу до пам'яті, тобто без участі процесора при його відключенні від системної магістралі [2].

3.6.3 Режим прямого доступу до пам'яті

Прямий доступ до пам'яті (ПДП, DMA – direct memory access) – це режим, що принципово відрізняється від двох раніше розглянутих режимів тим, що обмін системною шиною відбувається без участі процесора. Зовнішній пристрій, що потребує обслуговування, сигналізує процесору, що режим ПДП необхідний, у відповідь на це процесор закінчує виконання поточної команди і відключається від усіх шин, сигналізуючи пристрою, який подав запит, що обмін у режимі ПДП можна починати (рис. 3.24).

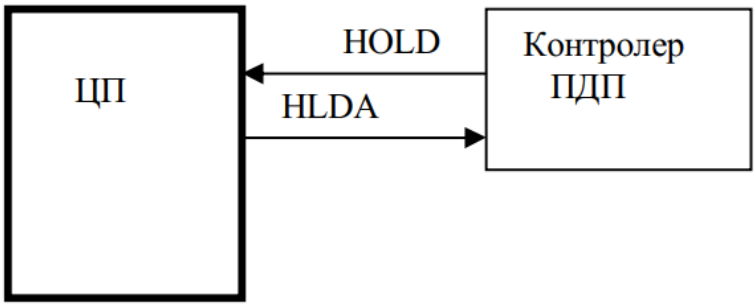


Рисунок 3.24 – Обслуговування переривання

Для того, щоб інший активний пристрій міг передати дані по магістралі, необхідно дезактивувати на цей час ЦП. Для здійснення цього режиму на СВ вводять додаткові сигнали HOLD і HLDA.

У звичайному режимі роботи на магістралі присутній єдиний активний пристрій (ЦП), який ініціює всі цикли обміну даних на магістралі.

Проте можливі випадки, коли на одній і тій самій магістралі присутні декілька активних пристроїв, які мають працювати з одним і тим самим блоком пам’яті і блоком ВВ.

При активному рівні на вхід HOLD мікропроцесор перериває виконання своєї програми, переводить виходи всіх своїх шин в стан високоімпедансу і виставляє активний рівень на виході HLDA, що має слугувати сигналом для іншого активного пристрою про те, що він може починати свої цикли обміну на магістралі. Коли цей пристрій закінчує свої цикли обміну, він скидає сигнал HOLD, після чого ЦП переходить в свій звичайний стан і продовжує виконувати програму.

Операція ПДП зводиться до пересилання інформації з пристрою введення/виведення в пам’ять або з пам’яті в пристрій введення/виведення. Коли пересилання інформації буде завершено, процесор знову повертається до перерваної програми, продовжуючи її з того місця, де його перервали. Це схоже на режим обслуговування переривань, але в цьому випадку процесор не бере участі в обміні. Як і у випадку переривань, реакція на зовнішню подію при ПДП істотно повільніша, ніж при програмному режимі.

Зрозуміло, що в цьому випадку потрібно введення в систему додаткового пристрою (контролера ПДП), що буде здійснювати повноцінний обмін системною магістраллю без всякої участі процесора. Причому процесор попередньо має повідомити цьому контролеру ПДП, звідки йому варто брати інформацію і/або куди її потрібно поміщати. Контролер ПДП може вважатися спеціалізованим процесором, який відрізняється тим, що сам не бере участі в обміні, не приймає і не видає інформацію (рис. 3.25, 3.26).

Контролер ПДП може входити до складу пристрою введення/виведення, якому потрібен режим ПДП або навіть обслуговувати декілька пристроїв введення/виведення.

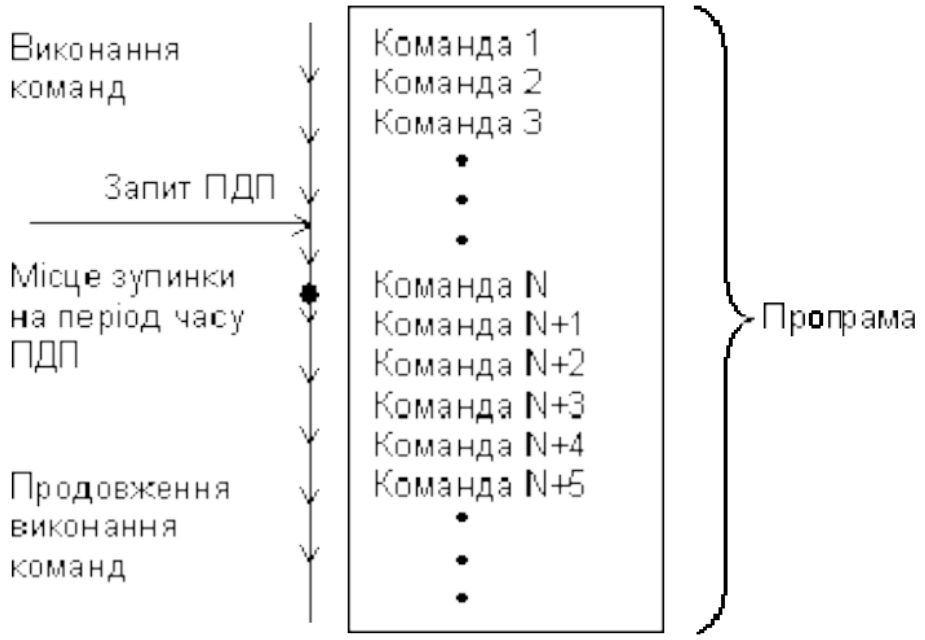


Рисунок 3.25 – Обслуговування ПДП

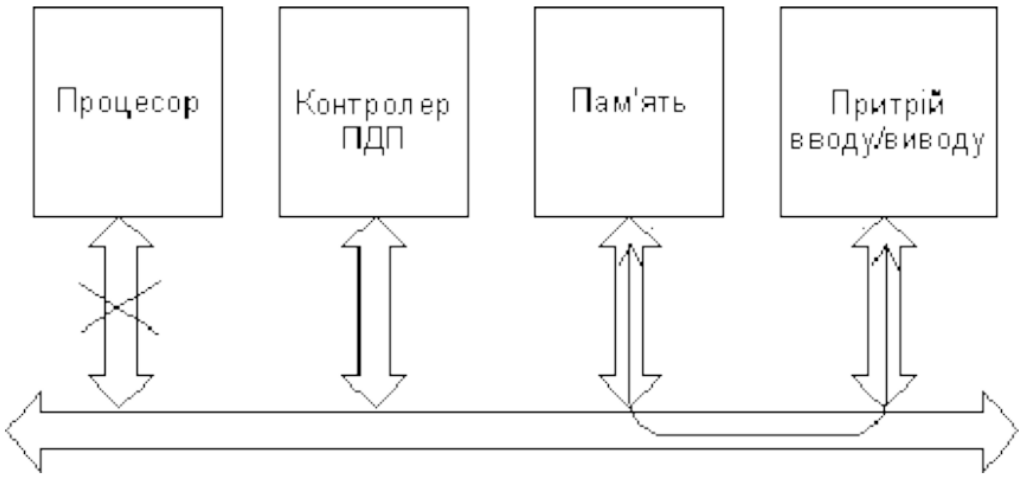


Рисунок 3.26 – Інформаційні потоки в режимі ПДП

Теоретично обмін за допомогою прямого доступу до пам’яті може забезпечити більш високу швидкість передачі інформації аніж програмний обмін, тому що процесор передає дані повільніше, ніж спеціалізований контролер ПДП. Однак на практиці ця перевага реалізується далеко не завжди. Швидкість обміну в режимі ПДП зазвичай обмежена можливостями магістралі. До того ж необхідність програмного задавання режимів контролера ПДП може звести нанівець виграш від більш високої швидкості пересилання даних у режимі ПДП. Тому режим ПДП застосовується рідко.

Якщо в системі вже існує самостійний контролер ПДП, то це може в ряді випадків істотно спростити апаратуру пристроїв введення/виведення, які працюють у режимі ПДП. У цьому полягає єдина безперечна перевага режиму ПДП [2].

3.7 Внутрішні регістри МП

Окрім CSEG і DSEG практично всі сучасні мікропроцесори мають спеціально виділений простір даних невеликого обсягу, який називається набором програмно-доступних регістрів RSEG.

На відміну від CSEG і DSEG регістри RSEG розташовуються всередині ЦП в безпосередній близькості від його АЛП, що забезпечує швидкий фізичний доступ до інформації, яка зберігається в них. У них, як правило, зберігаються проміжні результати обчислень, які часто використовуються ЦП. Область RSEG може бути повністю ізольована від простору даних DSEG, може частково перетинатися з нею, і може бути підчастиною DSEG. Внутрішня логічна організація RSEG дуже різноманітна і відіграє важливу роль при класифікації архітектури мікропроцесорів.

Регістри мікропроцесора функціонально неоднорідні: одні слугують для зберігання даних або адресної інформації, інші – для управління роботою ЦП. Відповідно до цього всі регістри можна розділити на регістри даних, покажчики і регістри спеціального призначення. Регістри даних беруть участь в арифметичних і логічних операціях як джерела операндів і приймачів результату, адресні регістри або покажчики використовуються для обчислення адрес даних і команд, розташованих в основній пам’яті. Спеціальні регістри слугують для індикації поточного стану ЦП і управління роботою його складових частин. Можлива архітектура, при якій одні і ті самі регістри використовуються для зберігання як даних, так і адресної інформації. Такі регістри називаються регістрами загального призначення (РЗП). Способи використання того або іншого типу регістрів визначають конкретні особливості архітектури мікропроцесора.

Серед регістрів даних часто виділяють один регістр, який називається акумулятором A (Accumulator), з яким пов’язують більшість команд арифметичної і логічної обробки даних. Це означає, що арифметичні і логічні команди використовують як один зі своїх операндів вміст акумулятора і зберігають в ньому результат операції. При цьому немає необхідності в коді команди виділяти спеціальну область для адрес операнда і результату. Такий тип архітектури мікропроцесора називається акумуляторним. До недоліків такої архітектури можна віднести відносно низьку швидкодію, яка пояснюється тим, що акумулятор є «вузьким місцем», в нього кожного разу необхідно спочатку занести операнд перед виконанням операції. Прикладом такої архітектури можуть слугувати мікроконтролери сімейства MCS-51 фірми Intel.

Іншим прикладом організації регістрів даних є регістри загального призначення R0, R1 тощо. В цьому випадку операнди і результати арифметичних і логічних операцій можуть зберігатися не в одному, а в декількох регістрах, що розширює можливості із маніпуляції даними. На відміну від акумулятора робочі регістри адресуються явно в коді команди.

Такий тип архітектури мікропроцесора називається регістровим. Прикладом такої організації можуть слугувати мікропроцесори сімейства 80×86 фірм Intel. У ряді мікропроцесорів, призначених для роботи в реальному часі, передбачено не один, а декілька наборів робочих регістрів. У кожен момент часу доступний лише один з наборів регістрів, вибір якого здійснюється записом відповідної інформації в певний службовий регістр. Прикладом таких пристроїв можуть слугувати мікроконтролери сімейства MCS-51 фірми Intel.

Архітектура, при якій процесор здатний використовувати як адреси операндів, так і результати операції елемента основної пам'яті, називається архітектурою типу «пам'ять–пам'ять». При цьому виключаються тимчасові витрати на переписування вмісту робочих регістрів при переході від однієї процедури до іншої. Проте при цьому втрачається швидкий доступ до проміжних даних, оскільки вони зберігаються не у внутрішніх регістрах, а в DSEG. Вирішенням цієї проблеми може бути розміщення частина DSEG на одному кристалі з ЦП і використання як робочі області цього внутрішнього сегмента ОЗП. Прикладом такої організації можуть слугувати мікроконтролери сімейства MCS-96 фірми Intel.

Практично у всіх сучасних мікропроцесорах виділяють окрему область пам'яті під так званий «стек», який використовується, в загальному випадку, для передачі параметрів процедурам і збереження адрес повернення з них. Стек може бути розташований усередині мікропроцесора або поза ним. Він може займати частину адресного простору DSEG або RSEG, а може бути розташований і окремо від них. У останньому випадку це є апаратний стек. Передача функцій акумулятора вершині стека утворює стекову архітектуру. Стекова організація дає можливість використовувати безадресні команди, код яких має найменшу довжину.

Безадресні команди оперують даними, що знаходяться на вершині стека і безпосередньо під ним. При виконанні операції вихідні операнди добуваються зі стека, а результат передається на вершину стека. Стекова архітектура має високу обчислювальну ефективність. Існує спеціальна мова високого рівня FORTH, яка базується на основі безадресних команд. Така архітектура використовується в спеціалізованих процесорах високої продуктивності і, зокрема, в RISC-процесорах.

Службові регістри, розташовані всередині мікропроцесора, призначені для різних функцій управління його роботою та індикації стану його складових частин. Їхній склад і організація залежать від конкретної архітектури процесора і розрізняються у кожному конкретному випадку. Найбільш використовуваними регістрами спеціальних функцій, що часто зустрічаються, є «програмний лічильник» PC (Program Counter), «вказівник стека» SP (Stack Pointer) і «слово стану програми» PSW (Program Status Word). Програмний лічильник PC в кожен конкретний момент часу містить адресу команди, наступної в CSEG за тією, яка в певний момент виконується. Покажчик стека SP зберігає поточну адресу вершини стека. Слово стану про-

грами PSW містить набір поточних ознак результату виконання операції. З кожною ознакою результату зв'язується однорозрядна змінна-прапорець, відповідна певному біту PSW. До типових прапорців-ознак відносяться:

- CF (Carry Flag) – прапорець перенесення із старшого розряду АЛП. Дорівнює 1, якщо в результаті виконання арифметичної операції або операції зсуву сталося перенесення із старшого розряду результату;

- ZF (Zero Flag) – прапорець нуля. Дорівнює 1, якщо результат операції дорівнює 0;

- SF (Sign Flag) – прапорець знака результату. Дублює знаковий розряд результату операції;

- AF (Auxilinary Carry Flag) – прапорець додаткового перенесення. Дорівнює 1, якщо в результаті виконання арифметичної операції або операції зсуву сталося перенесення з молодшої тетради результату в старшу. Часто використовується в двійково-десятковій арифметиці;

- OF (Owerfow Flag) – прапорець переповнення. Дорівнює 1, якщо в результаті виконання арифметичної операції сталося переповнення розрядної сітки результату;

- PF (Parity Flag) – прапорець парності. Дорівнює 1, якщо кількість одиниць в результаті останньої операції була парною.

- IF (Interrupt Flag) – прапорець дозволу переривання. Відображає дозвіл переривання в системі.

Конкретні прапорці використовуються програмою для аналізу результату попередньої команди і ухвалення рішення про подальший хід виконання програми. Спеціальні регістри можуть займати частину адресного простору DSEG або RSEG, а можуть бути розташовані і окремо від них [1 – 3].

3.8 Адресація команд і даних

Адресні регістри або покажчики використовуються для реалізації тих або інших методів адресації операндів, які використовуються в конкретних командах мікропроцесора. Їх конкретний набір і функції залежать від того, які методи адресації реалізовані в певній моделі мікропроцесора.

Під методом адресації розуміється метод кодування адреси операнда або результату операції в коді команди.

У загальному випадку код команди мікропроцесора можна записати в такому вигляді

КОП	АОП1	АОП2	АР

КОП – код операції;
АОП1 – поле адреси першого операнда;
АОП2 – поле адреси другого операнда;
АР – поле адреси результату.

Наявність окремих полів, окрім КОП, визначається конкретною командою і типом мікропроцесора. Інформація в полях АОП і АР визначається конкретним методом адресації, який використовується в цій команді.

Найбільш поширеними методами адресації, які використовуються в сучасних моделях мікропроцесорів є:

- регістрова адресація. Операнд знаходиться в регістрі. Адреса регістра внесена в код операції. Поле адреси в команді відсутнє;
- пряма адресація. Фізична адреса операнда розташована у відповідному полі адреси;
- безпосередня адресація. Безпосереднє значення операнда розташоване у відповідному полі адреси;
- непряма регістрова адресація. Фізична адреса операнда розташована в регістрі непрямої адреси DP (Data Pointer). Адреса регістра внесена в код операції. Поле адреси в команді відсутнє;
- непряма автоінкрементна/автодекрементна адресація. Фізична адреса операнда розташована в регістрі непрямої адреси DP. Адреса регістра внесена в код операції. Поле адреси в команді відсутнє. Після (або до) виконання операції вміст DP автоматично інкрементується/декрементується аби вказувати на наступний елемент таблиці;
- адресація по базі із зсувом. Базова адреса операнда розташована в регістрі бази BP (Base Pointer). Адреса регістра внесена в код операції. Зсув адреси операнда відносно базової адреси розташований у відповідному полі адреси;
- індексна адресація. Базова адреса операнда розташована у відповідному полі адреси. Зсув адреси операнда відносно базової адреси розташований в індексному регістрі X (Index);
- адресація по базі з індексуванням. Базова адреса операнда розташована в регістрі бази BP, зсув адреси операнда відносно базової адреси розташований в індексному регістрі X. Адреси регістрів внесені в код операції. Поле адреси в команді відсутнє;
- сегментна адресація. Вся пам’ять розбита на сегменти певного обсягу;
- адреса сегмента. Зберігається в сегментному регістрі SR (Segment Register), зсув адреси відносно початку сегмента розташовано у відповідному полі адреси або в індексному регістрі X.

Залежно від того, які методи адресації реалізовано в конкретному процесорі, в ньому є ті або інші адресні регістри. Складніші методи адресації потребують більшого часу для обчислення адреси операнда. Один з сучасних напрямів розвитку архітектури мікропроцесорів оснований на тому, щоб шляхом скорочення числа можливих команд і методів адресації досягти виконання будь-якої команди за один машинний цикл. Такі процесори називаються RISC-процессорами (Reduced Instruction Set Computer).

Конкретним прикладом такого пристрою може слугувати мікропроцесор POWERPC фірми Motorola.

У складі системи ВВ також можна виділити ряд функціонально закін-

чених пристроїв, які оформляються у вигляді модулів, що підключаються безпосередньо до єдиної магістралі системи. У простому випадку це буферні регістри, що адресуються ЦП, – порти ВВ. Складніші програмно-керовані підсистеми ВВ, що містять блоки портів, отримали назву периферійних адаптерів. У разі, якщо засоби ВВ призначаються для управління спеціальним зовнішнім устаткуванням і реалізації спеціальних функцій ВВ, їх називають периферійними контролерами. Найбільш складними з сучасних засобів обміну із зовнішніми пристроями ВВ вважають співпроцесори ВВ, які працюють за власними програмами, що зберігаються у власній пам'яті, і за своєю суттю є окремими мікропроцесорними системами. Прикладом такої системи може слугувати векторний співпроцесор ADMC-200 фірми Analog Devices, призначений для сполучення мікропроцесорної системи з вентильним перетворювачем, що керує приводом змінного струму. Він містить декілька каналів АЦП, багатоканальний ШІМ і обчислювальний блок, що реалізовує векторні перетворення, необхідні для здійснення алгоритму векторного управління синхронним і асинхронним двигуном змінного струму. Проте, незалежно від складності конкретної підсистеми ВВ, з боку ЦП всі вони подаються тим або іншим набором адресовуваних регістрів, який, як правило, є частиною DSEG.

Розрядністю мікропроцесорної системи прийнято вважати кількість бітів інформації, яку її ЦП може обробити за допомогою однієї команди. Розрядність мікропроцесора визначається розрядністю його АЛП, внутрішніх регістрів даних і зовнішньої шини даних. На сьогоднішній день існують 8-, 16-, 32- і 64-розрядні мікропроцесори. Для того, щоб обробляти інформацію з розрядністю більшою, ніж розрядність мікропроцесора, необхідно реалізовувати спеціальні алгоритми обчислень з підвищеною розрядністю. Ці алгоритми потребують додаткового часу для свого виконання. Тому підвищення розрядності мікропроцесора при заданій розрядності обчислень, безпосередньо пов'язано із збільшенням швидкодії системи.

Залежно від того, в якому форматі процесор здатний сприймати і обробляти дані, розрізняють мікропроцесори з фіксованою крапкою і мікропроцесори з плаваючою крапкою. При заданій точності обчислень і розрядності, діапазон чисел у форматі з плаваючою крапкою значно перевищує діапазон чисел у форматі з фіксованою крапкою. Тому обчислення з плаваючою крапкою використовуються для забезпечення підвищеної точності результату. Реалізація подібних алгоритмів на процесорах з фіксованою крапкою призводить до довготривалих обчислень і, отже, до зниження швидкодії системи. Процесори з плаваючою крапкою здатні виконувати арифметичні операції над числами з плаваючою крапкою за допомогою однієї команди. Тому вони виконують подібні обчислення значно швидше, ніж процесори з фіксованою крапкою.

Існують мікропроцесори, архітектура яких адаптована для виконання обчислень певного роду. До таких процесорів відносяться «процесори цифрової обробки сигналів» DSP (Digital Signal Procesor). Їх архітектура має

особливості, що дозволяють їм з найбільшою продуктивністю здійснювати алгоритми рекурентної обробки даних, які використовуються в багатьох завданнях, що вимагають їх виконання в реальному часі, таких як аудіо- і відео-кодування, регулювання, цифрова фільтрація, цифровий зв'язок тощо. Всі ці процесори побудовані, як правило, по Гарвардській архітектурі. Сучасні DSP мають окремі шини адреса/дані для CSEG і DSEG, що дозволяє їм за допомогою однієї команди здійснити доступ до різних видів пам'яті і виробити декілька операцій над даними. Основною особливістю DSP є те, що окрім звичайного АЛП, яке присутнє у всіх процесорах, вони мають ще декілька обчислювальних пристроїв. До таких пристроїв в першу чергу відноситься «помножувач-акумулятор» MAU (Multiple-Accumulator Unit), здатний за допомогою однієї команди помножити два багаторозрядні числа і скласти результат подвоєної розрядності з результатом попередньої команди.

Подібна операція «множення–додавання» використовується у всіх рекурентних алгоритмах. Наявність MAU у поєднанні з вищезгаданими особливостями організації шин процесора дозволяє DSP за одну команду повністю виконати один крок рекурентного алгоритму і підготувати вихідні дані для наступного кроку. Іншим додатковим обчислювальним пристроєм є «багаторозрядний регістр зрушення» S (Shifter), здатний виконувати операції зрушення над числами, розрядність яких перевищує розрядність АЛП. Спільна робота цих обчислювальних пристроїв дозволяє досягти на виконанні рекурентних алгоритмів обчислювальної продуктивності, незрівнянної з будь-якими іншими процесорами [1 – 3].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Назвіть і охарактеризуйте компоненти, що складають МПС.
2. Визначте мінімальну конфігурацію та призначення кожного елемента МПС.
3. У якому вигляді команди зберігаються в пам'яті?
4. Дайте означення шинної організації МПС. У чому її переваги?
5. Поясніть проходження сигналів у режимі читання пам'яті.
6. Поясніть проходження сигналів у режимі записування в пам'ять.
7. Поясніть проходження сигналів у режимі читання зовнішніх пристроїв.
8. Поясніть проходження сигналів у режимі записування в зовнішній пристрій.

4 ФІЗІЧНА ОРГАНІЗАЦІЯ ПРИСТРОЇВ ПАМ'ЯТІ

4.1 Запам'ятовувальний пристрій

Запам'ятовувальний пристрій (ЗП) складається з величезного числа елементів пам'яті, кожен з яких може знаходитися в одному з двох станів, що кодуються двійковою цифрою 1 або 0. Елемент пам'яті (ЕП) є областю, де зберігається біт інформації [1]. Елементи пам'яті ЗП групуються в слова інформації, тобто такі порції інформації, які можуть одночасно пересилатися між ЗП та МП і оброблятися останнім. Структуру ЗП, що складається з *n* комірок, кожна з яких зберігає 1 байт, подано на рис. 4.1.

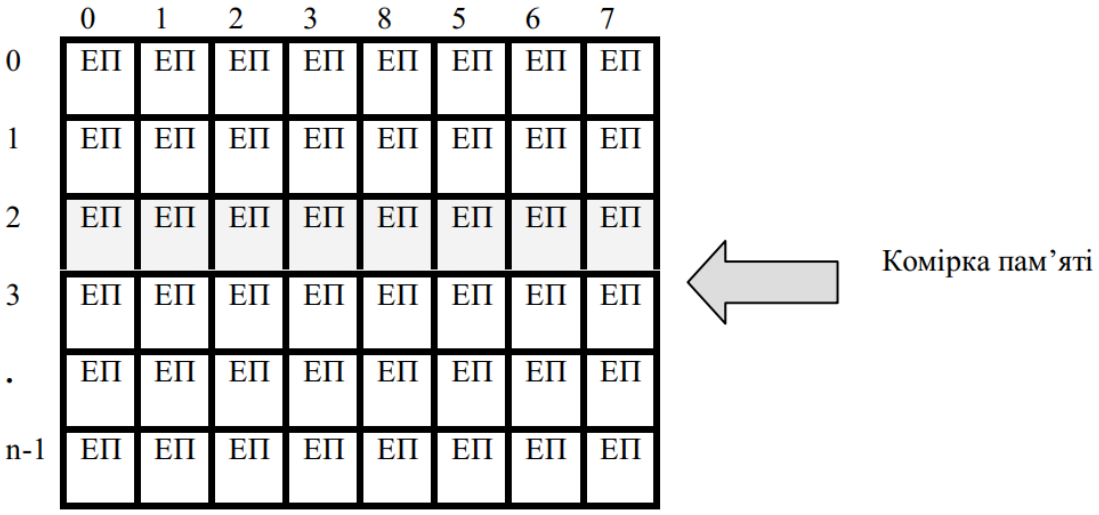


Рисунок 4.1 – Структура запам'ятовувального пристрою

Пошук потрібного слова ЗП можна здійснювати або за його адресою (адресні ЗП), або за його вмістом (асоціативні ЗП). У адресних ЗП звернення до комірок пам'яті виробляється за їх фізичними координатами, задаваним двійковим кодом – адресою. Вони бувають з довільним зверненням (вибіркою), тобто допускають будь-який порядок дотримання адрес, і з послідовним зверненням, де вибірка елементів пам'яті можлива лише в певному порядку зростання або спадання адрес. Архітектура МП орієнтована, в першу чергу, на використання адресних ЗП з довільною вибіркою.

Ємкість ЗП виражається в кількості бітів (б), байтів (Б) або слів, що складаються з певного числа бітів. Оскільки ця ємкість може бути дуже великою, то зазвичай використовують одиниці кіло- (к), мега- (М) або гіга- (Г):

2^{10} = 1К =1024;

2^{20} = 1М =1 048 676;

2^{30} = 1Г =1 073 741 824.

Нині найбільшого поширення набули напівпровідникові ЗП, побудовані на великих інтегральних схемах (ВІС), кожна з яких містить велике число елементів пам'яті. Ці елементи зазвичай об'єднуються у комірки розміром 1, 4 або 8 бітів. Так ВІС, що містять 2К (2048) елементів пам'яті, можна виготовляти для зберігання 2К 1-бітових слів, 512 4-бітових слів або 256 8-бітових слів (2К×1, 512×4 або 256×8) [4].

Напівпровідникові ЗП підрозділяються на незалежні від живлення – постійні запам'ятовувальні пристрої (ПЗП), і енергозалежні – оперативні запам'ятовувальні пристрої (ОЗП). На рис. 4.2 показано типи напівпровідникових ЗП [2].

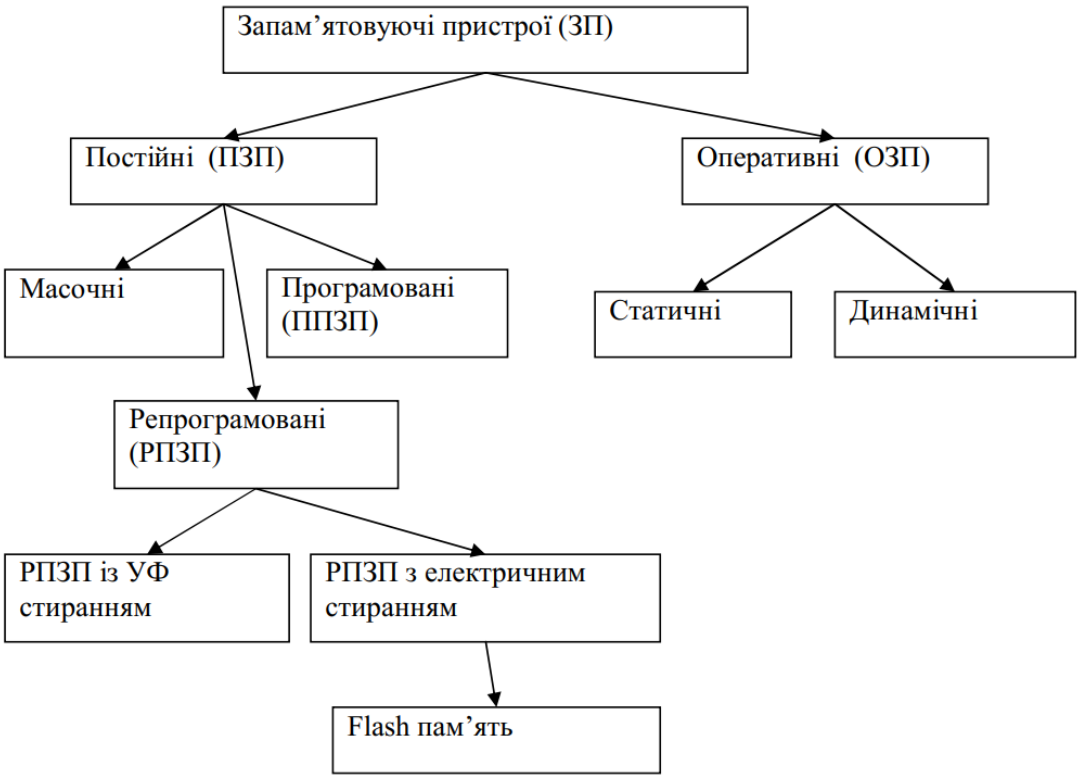


Рисунок 4.2 – Класифікація запам'ятовувальних пристроїв

4.2 Постійні запам'ятовувальні пристрої

Завдяки незалежності ПЗП застосовуються для зберігання керівних та ініціалізувальних програм, різних таблиць констант тощо. Мікропроцесор в процесі роботи може лише зчитувати (вибирати) інформацію з ПЗП, але не в змозі змінити його вміст.

Процес занесення інформації в ПЗП називається програмуванням і здійснюється, як правило, поза МПС, в якій передбачається їх використовувати. Для цього слугують програматори, які виконуються у вигляді автономних або периферійних пристроїв.

Виключенням є ПЗП, виконані за Flash-технологією, яка дозволяє здійснювати загальне стирання і записування інформації в елементи пам'яті

безпосередньо мікропроцесором. Ця властивість наближає його до ОЗП, але на відміну від них Flash ПЗП є незалежним [4].

На рис. 4.3 наведений приклад внутрішньої структури ПЗП з організацією 256 комірок по 8 біт (256×8).

Для адресації елементів пам’яті слугує 8 вхідних ліній A0-A7. По них можна задавати до 256 різних адрес в двійковому коді. Дешифратор адреси перетворить двійкову адресу на позиційний код для вибору одного з 256 рядків матриці елементів пам’яті (ЕП). З виходів вибраних ЕП на лінії читання виробляються сигнали 0 або 1. Таким чином, на виходи даних D0–D7, сполучених з вертикальними лініями читання через підсилювачі, надходить код, відповідний інформації з адресованих елементів пам’яті.

Керівні входи CS (вибір кристала) і OE (дозвіл за виходом) є інверсними, тобто активний рівень – логічний нуль. Вхід CS управляє загальним вибором мікросхеми, тобто при подачі нуля/одиниці дозволяється/ забороняється дешифрування адреси і вибір елемента пам’яті. У вибраного ПЗП за допомогою входу OE активуються вихідні буфери-підсилювачі. За відсутності сигналу CS або OE виходи D0–D7 знаходяться у відключеному (високоімпедансному) стані – в третьому стані.

Існує декілька різновидів ПЗП, які розрізняються принципом програмування, а також технологією виготовлення.

Програмовані ПЗП (ППЗП) з плавними перемичками надходять до споживача в первинному незапрограмованому стані, відповідному 0 або 1 у всіх елементах пам’яті.

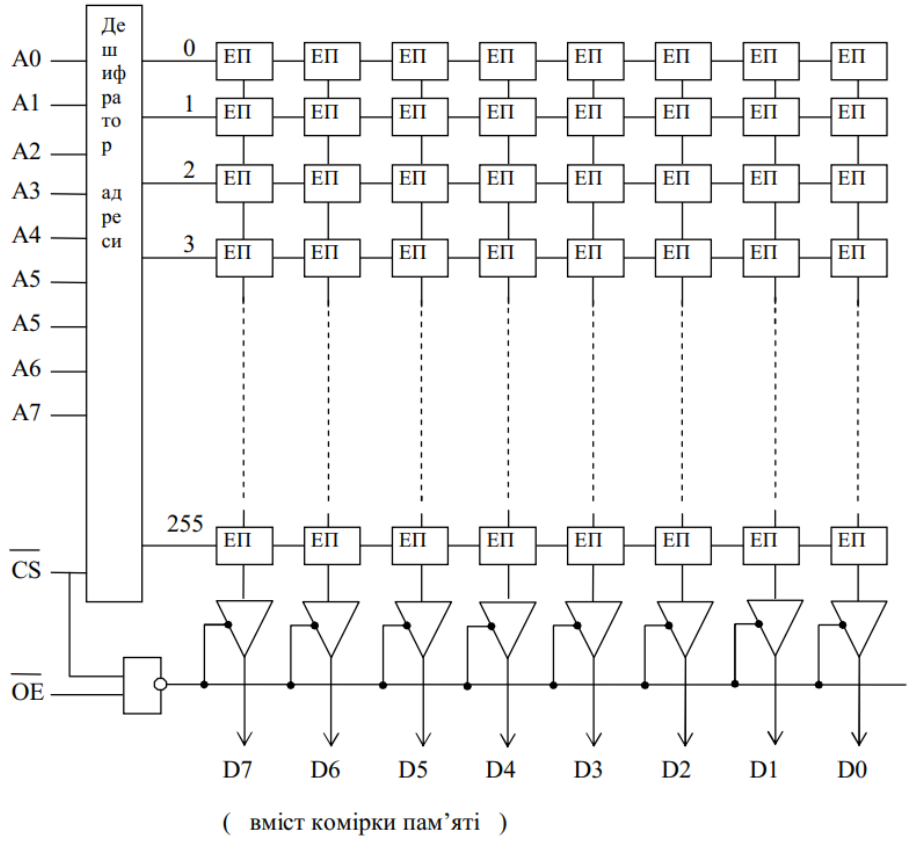


Рисунок 4.3 – Структура ПЗП з довільною вибіркою

У режимі програмування потрібну інформацію записують в ППЗП шляхом перепалювання перемичок, що відіграють роль ЕП, електричним струмом за допомогою програматора. Надалі зміна інформації, занесеної в ППЗП, можлива лише шляхом перепалювання перемичок, що залишилися після попереднього програмування.

Репрограмовані ПЗП (РПЗП) з ультрафіолетовим стиранням інформації нині найширше використовуються в МПС. У цих ВІС кожен біт інформації, що зберігається, відображається станом відповідного МОН-транзистора з плаваючим затвором, що є конденсатором. Заряджаючи і розряджаючи його, виконують записування і стирання інформації. Накопичений заряд в таких конденсаторах може зберігатися дуже довго (більше 10 років) за рахунок високої якості ізолювального шару.

Незапрограмована мікросхема РПЗП з ультрафіолетовим стиранням має на виходах по всіх адресах рівень логічної одиниці. Для записування в необхідні розряди логічного нуля на відповідні виводи даних подається рівень 0, а на останні – 1. Можна виконувати корегування раніше записаної інформації, змінюючи стан 1 на 0 (але не навпаки).

Для стирання інформації протягом 30 – 60 хв опромінюють кристал ВІС крізь прозоре вікно в корпусі ультрафіолетовим випромінюванням (рис. 4.4) люмінесцентної лампи, яке збільшує струм витоку в ізолювальному шарі, наводячи до розсмоктування заряду, що зберігається на плаваючих затворах.

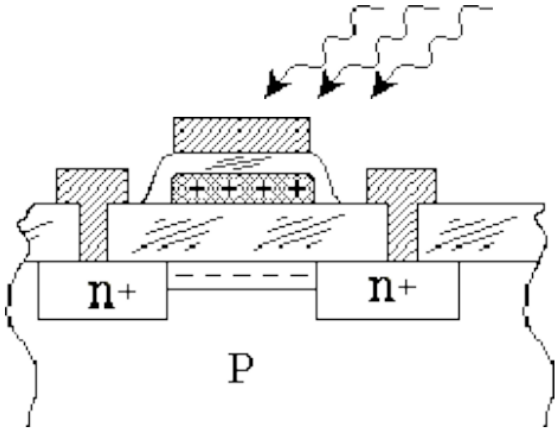


Рисунок 4.4 – Елемент пам’яті ПЗП з ультрафіолетовим і електричним стиранням

Число циклів перезапису лежить зазвичай в межах 10...100 (для різних типів), оскільки із перепрограмуванням поступово погіршуються діелектричні властивості ізолювального шару.

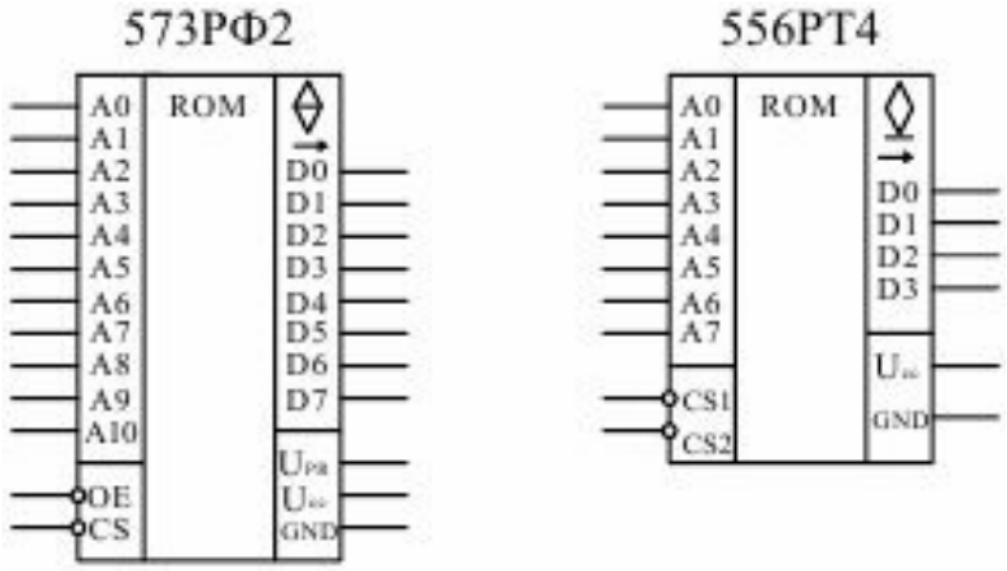
РПЗП з електричним стиранням дозволяє виконувати як записування, так і стирання інформації за допомогою електричних сигналів. Завдяки цьому з’являється можливість змінювати вміст постійної пам’яті безпосередньо в МПС, якщо там передбачено пристрої формування сигналів сти-

рання і програмування. Особливістю таких РПЗП є блокове стирання, тобто неможливість стирання інформації в окремих елементах пам'яті.

Перевагою РПЗП з електричним стиранням є не лише зручність і висока швидкість перезаписування інформації, але і значне допустиме число циклів перезаписування. Сучасні технології гарантують не менше 100000 циклів.

Flash пам'ять є різновидом РПЗП з електричним стиранням. Нині Flash ПЗП широко використовуються для організації програмної пам'яті мікроконтролерів. Допускається не менше 1000 циклів перезапису.

На рис. 4.5 подано приклади графічного позначення ПЗП на принципових схемах.



а б

Рисунок 4.5 – Умовне графічне позначення ПЗП: а – вихід з трьома станами; б – вихід з відкритим колектором

Абревіатура ROM (Read Only Memory – пам'ять лише для читання) використовується для ПЗП. Зліва розташовано виводи вхідних сигналів адреси і управління, справа – виходи даних, подачі живлення (Ucc, GND) і напруги програмування (UPR). Знак інверсії на входах управління показує, що включення режиму виконується логічним рівнем 0, а виключення – 1.

Програматори слугують для занесення інформації в програмованих і репрограмованих ПЗП. Програматори виконують або у вигляді автономних пристроїв, або на базі комп'ютерів. Автономні програматори мають обмежені функціональні можливості і застосовуються, в основному, для копіювання інформації з ПЗП. Досконалішими є програматори, побудовані на базі комп'ютерів. Можливі два варіанти їх підключення: через стандартний інтерфейс (COMPORT, LPT, USB) та через системну шину

комп’ютера. У першому випадку програматор є зовнішнім блоком комп’ютера, в другому – його внутрішнім модулем [3].

4.3 Оперативні запам’ятовувальні пристрої

ОЗП слугують для зберігання тимчасової і змінної інформації, наприклад налагоджуваних програм і проміжних даних користувача. Його головна перевага перед ПЗП – можливість записування і читання інформації безпосередньо мікропроцесором. При цьому не потрібно попереднє стирання вмісту елементів пам’яті. Час записування в ОЗП виходить наймінімальнішим серед інших типів ЗП, а кількість операцій запису необмежена.

ОЗП залежно від структури елементів пам’яті підрозділяються на статичні та динамічні.

Елементи пам’яті в статичних ОЗП будуються на основі статичних багатотранзисторних тригерних кіл. Статичні ОЗП програють в 4 – 8 разів в інформаційній ємкості на одиницю обсягу кристала динамічним ОЗП, в яких запам’ятовувальний елемент виконується однотранзисторним. Інформація в такому елементі зберігається у вигляді заряду на запам’ятовувальному конденсаторі. У режимі зберігання інформації необхідно періодично виробляти регенерацію заряду для компенсації природних витоків. Регенерація виробляється читанням вмісту кожної комірки ОЗП з періодом не більше 1...2 мс.

Регенерація може вироблятися мікропроцесором програмно, наприклад, за допомогою спеціальних переривань. Але частіше вона реалізується апаратно за допомогою спеціального контролера регенерації.

Завдяки високій щільності розміщення, динамічні структури використовують для створення мікросхем ОЗП найбільшого обсягу.

Хоча статичні ОЗП мають менший обсяг пам’яті, вони не потребують постійної регенерації. Це дозволяє спростити апаратні і програмні засоби МПС.

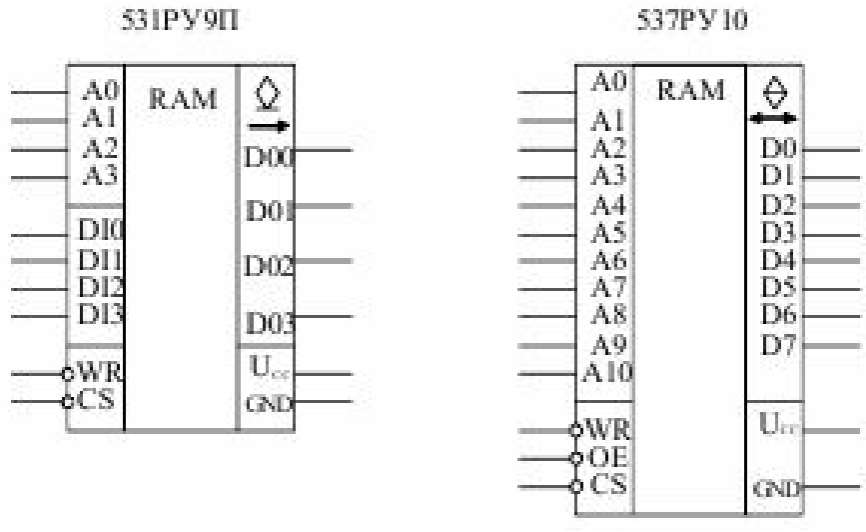
Крім того, статичні ОЗП споживають значно менше енергії, ніж динамічні, і можуть застосовуватися в пристроях, що працюють від автономних джерел (акумуляторів або батарей).

Нині розробляються нові типи ОЗП, що забезпечують незалежне зберігання інформації. Прикладом такого пристрою є статичні ОЗП фірми «FRAMTON», елементи пам’яті яких виконано на основі сегнетоелектриків.

Подібні ОЗП не поступаються ПЗП за часом зберігання інформації, перевершують їх за швидкодією і кількістю операцій записування. Проте широке їх використання стримують висока вартість і недостатня ємкість.

Внутрішня структура статичного ОЗП близька до схеми, наведеної на рис. 4.3. Відмінності полягають в тому, що ЕП є тригерами, і є пристрій, що керує їх перемиканням по зовнішніх сигналах дозволу записування і вхідних даних.

На рис. 4.6 подано приклади графічного позначення ОЗП на принципових схемах. Для них використовується абревіатура RAM (Random asess memory – пам’ять з довільною вибіркою).



а б

Рисунок 4.6 – Умовне графічне позначення ОЗП: а – з окремими входами і виходами даних; б – з поєднаними входами/виходами даних

Існує декілька варіантів організації виводів даних ОЗП. На рис. 4.6, а показано умовне графічне позначення ОЗП з окремими входами (DI0-DI3) і виходами (DO0-DO3) даних, а на рис. 4.6, б – ОЗП з поєднаними входами/виходами (D0-D7) даних. Крім того, виводи даних є двонапрямленими. Вхід WR слугує для управління записом інформації.

Оскільки він має знак інверсії, запис виробляється нульовим рівнем сигналу.

Запам’ятовувальні пристрої характеризуються такими основними параметрами:

- розрядністю даних (визначається розрядністю елемента пам’яті);

- інформаційною ємкістю (визначається кількістю одиниць інформації в бітах, яку ЗП може зберігати одночасно). Інформаційну ємкість часто позначають $M \times n$, де n – розрядність шини даних; M – кількість n-разрядных слів. Так, запис значення інформаційної ємкості 2048 × 8 або 2К × 8 означає, що ЗП може зберігати 2048 байт;

- часом вибірки (визначається як інтервал від моменту видачі запиту на передачу даних з пам’яті до моменту появи інформації на виході ЗП);

- тривалістю циклу звернення (визначається мінімально допустимим інтервалом часу між двома послідовними зверненнями до ЗП);

- напругою живлення;

- потужністю енергоспоживання (визначається струмом споживання і напругою джерела живлення). Для деяких типів ЗП наводять два значення потужності енергоспоживання – одне для режиму звернення, коли здійснюється записування або читання, інше – для режиму зберігання. Останнє, як правило, істотно менше потужності енергоспоживання в режимі звернення;

- питомою вартістю (визначається відношенням вартості ЗП до його інформаційної ємкості) [3].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. З чого складається внутрішня пам’ять МПС?
2. Що таке елемент пам’яті?
3. Який обсяг інформації називається словом?
4. Які існують способи пошуку даних в ЗП?
5. Які існують способи адресації ЗП?
6. Як розрізняють ЗП за часом зберігання даних?
7. Для чого застосовують ПЗП в МПС?
8. Для чого застосовують ОЗП в МПС?
9. Які існують типи ОЗП?
10. Що являє собою елемент оперативної пам’яті?
11. Яку внутрішню структуру мають ЗП?
12. Як розрізняють ПЗП за способом програмування?
13. Як розрізняють РПЗП за способом стирання?
14. Що таке програматор і які бувають варіанти його виконання?
15. Як здійснюється регенерація динамічних ОЗП?
16. Які існують варіанти організації введення/виведення даних ОЗП?

5 ЛОГІЧНА ОРГАНІЗАЦІЯ ПАМ'ЯТІ

5.1 Сегментування пам'яті

Організація пам'яті, коли кожній адресі відповідає вміст однієї комірки пам'яті, називається лінійною. З часів появи перших персональних комп'ютерів застосовують сегментну організацію пам'яті, яка характеризується тим, що в поточний час роботи програмно доступною є не вся пам'ять, а лише деякі сегменти, тобто області пам'яті. Усередині сегмента використовують лінійну адресацію.

В пам'яті зберігаються як байти, так і двобайтові слова. Слова розміщуються в двох сусідніх комірках пам'яті: старший байт зберігається в комірці зі старшою адресою, молодший – з молодшою. Адресою слова вважається адреса його молодшого байта. На рис. 5.1 показано приклад, коли з адресою 00000 зберігається байт 35H, а з адресою 00001 – слово 784AH.

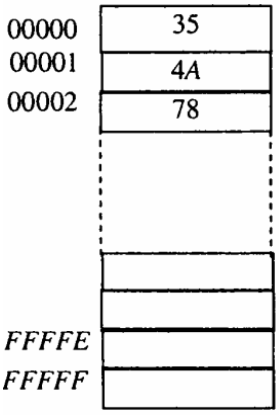


Рисунок 5.1 – Програмна модель фізичної пам'яті 1 Мбайт

Мікропроцесор i8086 являє собою 16-розрядний процесор, тобто він має 16-розрядну внутрішню шину, 16-розрядні регістри і суматори. Прагнення розробників ВІС адресувати якомога більший масив фізичної пам'яті зумовило використання 20-розрядної шини даних.

Для формування 20-розрядної адреси в 16-розрядному процесорі використовують інформацію з двох 16-розрядних регістрів. В МП i8086 20-розрядна адреси формується з двох 16-розрядних частин, які називають логічною адресою.

Перша частина адресує сегмент. Доповнена праворуч чотирма нулями, вона являє собою початкову фізичну адресу сегмента ємністю 64 Кбайт.

Друга частина адреси визначає зміщення в сегменті, тобто відстань від початку сегмента до адресованої комірки. Якщо вона дорівнює 0000, то адресується початкова комірка сегмента, якщо FFFF – то остання. Отже, логічний адресний простір розділено на блоки суміжних адрес розміром 64 Кбайт, тобто сегменти.

Такий підхід до організації пам'яті зручний ще й тому, що пам'ять зазвичай логічно поділяється на області програмного коду, даних і стека. Фізична 20-розрядна адреса комірки пам'яті формується з двох 16-розрядних адрес – адреси сегмента *Seg* і виконавчої адреси (зміщення) *EA*, які додаються із зсувом на чотири розряди (рис. 5.2).

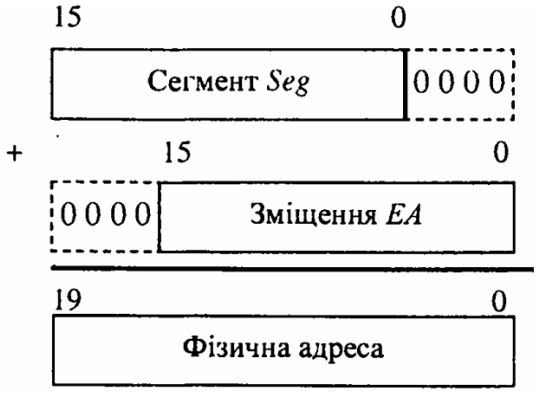


Рисунок 5.2 – Формування фізичної адреси

Зсув адреси сегмента на 4 розряди ліворуч еквівалентний його множенню на 16. Тоді фізична адреси дорівнює $16 \times Seg + EA$.

Перетворення логічних адрес на фізичні завжди однозначне, тобто парі *Seg* і *EA* відповідає єдина фізична адреса. Обернене перетворення не є однозначним: фізичну адресу можна подати за допомогою 4096 логічних адрес. Логічну адресу позначають Seg : EA.

Приклад 1. Знайти значення фізичної адреси за значенням логічної адреси B002:317A, тобто Seg=B002 і EA=317A (числа шістнадцяткові).

Допишемо до значення Seg праворуч шістнадцятковий 0, тобто чотири двійкових нулі.

Тоді B0020 – це адреси початкової (базової) комірки в сегменті. Виконавши операцію додавання цієї величини зі значенням EA, отримаємо фізичну адресу:

```
  1011 0000 0000 0010 0000
+      0011 0001 0111 1010
  1011 0011 0001 1001 1010   =   B319A
```

Отже, запис Seg = B002, EA = 317A відповідає фізичній адресі B319A.

Приклад 2. Знайти значення логічної адреси, яка б відповідала фізичній адресі B319A і не дорівнювала логічній адресі прикладу 1.

Значення фізичної адреси B319A можна одержати додаванням двох інших логічних адрес B100:219A :

```
  1011 0001 0000 0000 0000
+      0010 0001 1001 1010
  1011 0011 0001 1001 101A   =   B319A
```

Ємність пам’яті 1 Мбайт, починаючи з фізичної нульової адреси, розбивається на параграфи по 16 байт. Сегмент може починатися тільки на межі параграфа, тобто в адресі сегмента молодші чотири біти адреси – нульові. Розміщення сегментів в пам’яті довільне: сегменти можуть частково або повністю перекриватися або не мати спільних частин. Змінюючи значення як сегмента, так і зміщення в логічній адресі, можна адресувати будь-яку комірку із загальної пам’яті ємністю 1 Мбайт [5].

На рис. 5.3, а показано розміщення в просторі 1 Мбайт чотирьох сегментів по 64 кбайт без перекриття. Початкові адреси сегментів визначаються вмістом 16-розрядних сегментних регістрів, які доповнено праворуч чотирма нульовими бітами. Сегмент коду визначається вмістом регістра CS; сегмент даних – вмістом регістра DS, додатковий сегмент даних – вмістом регістра ES, а сегмент стека – вмістом регістра SP.

В сегментах кодів розміщено коди команд, тобто програма в машинних кодах; в решті сегментів – дані. Програма може звертатися тільки до даних в сегментах (рис. 5.3), позначених заштрихованими областями.

Змінюючи вміст сегментних регістрів, можна пересувати сегменти в межах усієї пам’яті – 1 Мбайт. На рис. 5.3, б показано розташування сегментів кодів, даних, стека та додаткового сегмента із частковим перекриттям. Такий випадок виникає тоді, коли вміст сегментних регістрів відрізняється менш ніж на 64 кбайт/16 = 4096 байт.

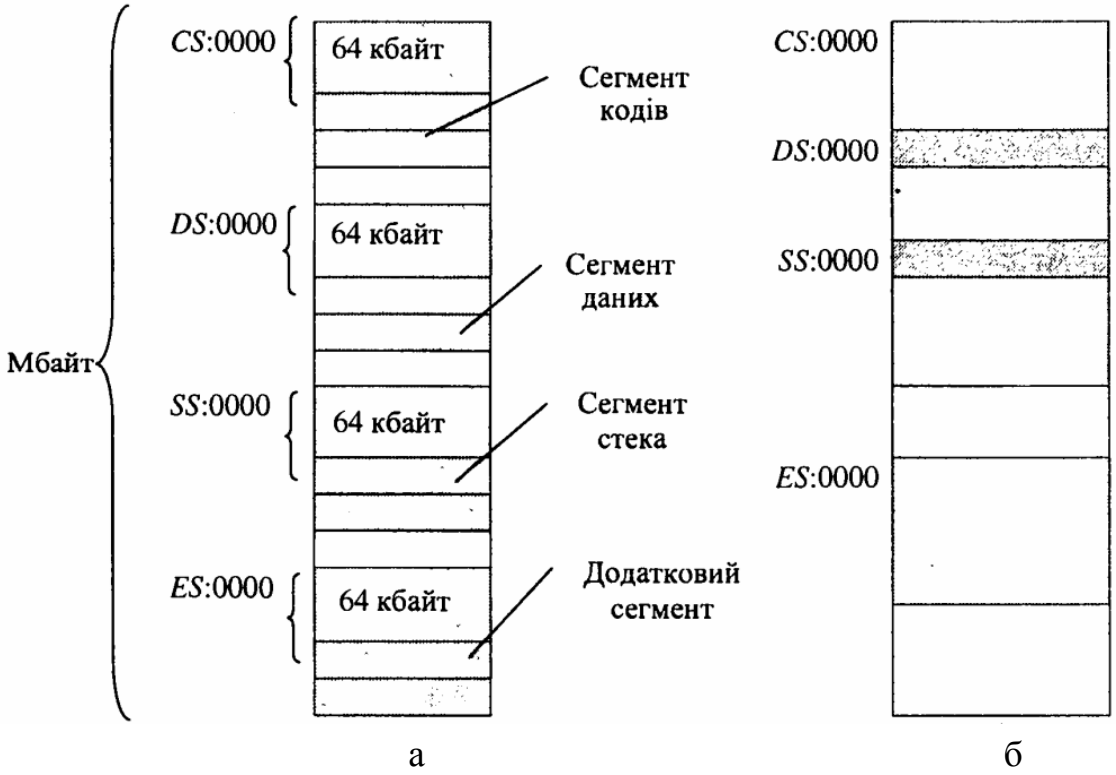


Рисунок 5.3 – Розміщення сегментів в просторі пам’яті 1 Мбайт: а – без перекритгя; б – з частковим перекриттям

5.2 Принципи організації стекової пам’яті

Стековою пам’яттю, або стеком, називають пам’ять, в якій реалізовано принцип «останній увійшов – перший вийшов», тобто дані, записані останніми, прочитуються першими [2]. У МПС стекова пам’ять використовується для викликів підпрограм, зокрема вкладених, і обробки переривань.

За способом реалізації розрізняють апаратний і апаратно-програмний стеки.

Апаратний стек є сукупністю регістрів, зв’язок між якими організовано таким чином, що під час записування і читання даних вміст стека автоматично зміщується. Принцип роботи апаратного 8-рівневого стека ілюструє рис. 5.4.

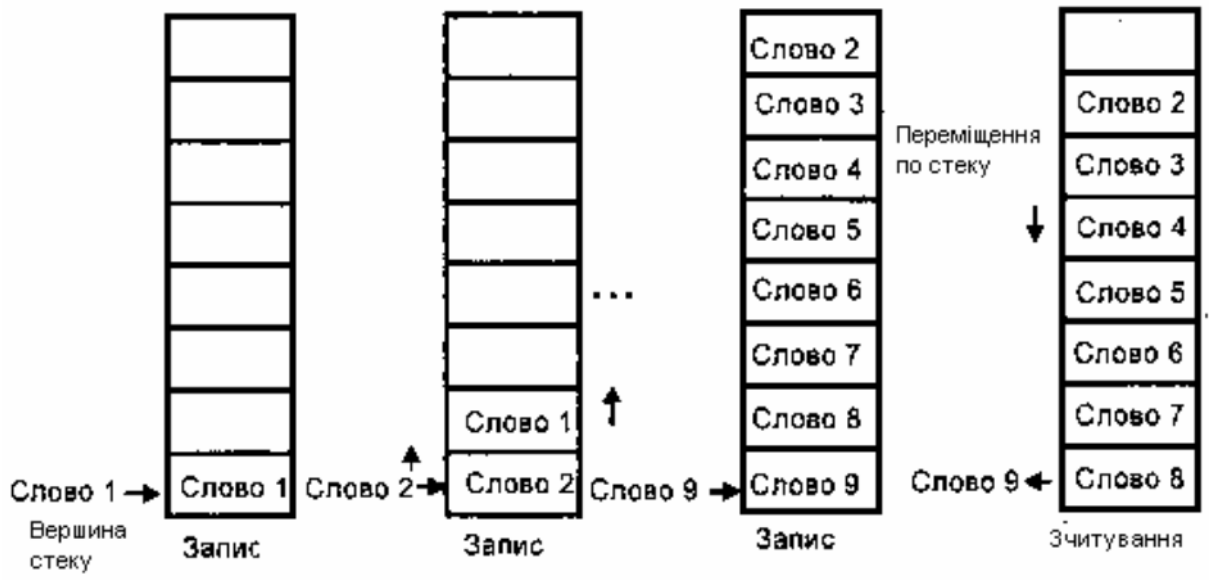


Рисунок 5.4 – Принцип роботи апаратного стека

При записі Слова 1 в стек воно розташовується в першому вільному елементі пам’яті (у першому регістрі) – вершині стека. Наступне слово зрушує вершину на одну комірку вгору, займаючи його місце, і так далі. Запис Слова 9 наводить до переповнення стека і втрати Слова 1. Зчитування слів із стека здійснюється в зворотному порядку, тобто спочатку прочитується Слово 9, яке записане останнім, а прочитування Слова 6 неможливо, поки не будуть зчитані Слова 8 і 7.

Інформаційна ємкість апаратного стека визначається як $M \times n$, де кількість n-розрядних слів M дорівнює кількості регістрів і може становити декілька десятків. Апаратні стеки, які використовуються, наприклад, в PIC-процесорах, мають 2, 8 або 16 регістрів (M = 2, 8, 16), в яких розміщуються 12-, 14- або 16-розрядні слова (n = 12, 14, 16). Основною перевагою апаратного стека є висока швидкодія, а недоліком – обмежена інформаційна ємкість.

Апаратно-програмний стек реалізується шляхом використання частини ОЗП статичного типу і спеціального регістра SP (Stack Pointer – вказівник або покажчик стека), який містить адресу останньої зайнятої комірки стека.

Принцип роботи апаратно-програмного стека для мікропроцесорів 80×86 показано на рис. 5.5. У апаратно-програмному стеку фізичного зрушення даних при записуванні і читанні не відбувається. Проте автоматична зміна вмісту SP еквівалентна зрушенню даних, що відбувається в апаратному стеку.

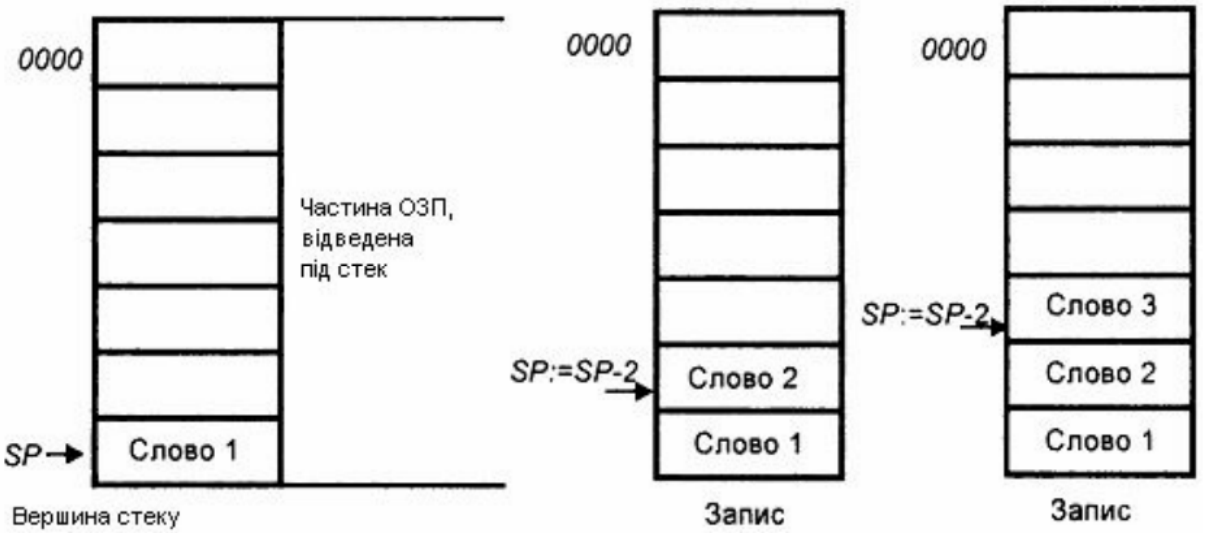


Рисунок 5.5 – Принцип роботи апаратно-програмного стека

На початку програми в регістр SP заносять адресу вершини стека. Після кожної операції записування (читання) вміст регістра SP автоматично змінюється. Для МП 80×86 одночасно можна записувати в стек або прочитувати з нього 2-байтові слова, тому SP змінюється на два. Під час записування в стек значення SP зменшується на два (стек «зростає» в область малих адрес), при прочитуванні із стека – збільшується на два. Таким чином, покажчик стека SP завжди містить адресу комірки, до якої відбулося останнє звернення. У деяких командах, наприклад, в командах викликів підпрограм, переривань, повернення з підпрограм, звернення до стека здійснюється автоматично [1, 2].

5.3 Кеш-пам’ять

Через те, що дані, які обробляються локально, можуть виникати в динаміці обчислень і не бути сконцентрованими в одній області при статичному розміщенні в основній пам’яті, буферну пам’ять організовують як асоціативну, в якій дані містяться в сукупності з їх адресою в основній пам’яті. Асоціативна пам’ять дозволяє вибирати серед елементів даних, що зберігаються в ній, ті, які мають збіжні з шаблоном вибірки, що називаються ключем значення розрядів. Наприклад, якщо ключ має значення 1 в

третьому розряді і значення 0 в п’ятому, то при вибірці будуть знайдені всі елементи асоціативної пам’яті, в яких третій розряд дорівнює 1, а п’ятий дорівнює 0.

Звернення до такої пам’яті з використанням поля адреси як ключа дозволяє вибрати дані незалежно від місця їх розташування в асоціативній пам’яті.

Зберігання адрес разом з даними дозволяє однозначно ідентифікувати дані їх адресами при передачі між рівнями пам’яті і легко знаходити дані на будь-якому рівні пам’яті. Така буферна пам’ять отримала назву кеш-пам’яті. Кеш-пам’ять дозволяє гнучко погоджувати структури даних, потрібні в динаміці обчислень, із статичними структурами даних основної пам’яті.

Кеш-пам’ять має сукупність рядків, кожен з яких складається з фіксованої кількості одиниць пам’яті (байтів, слів), що адресуються та зберігаються в основній пам’яті як комірки з послідовними адресами. Типовий розмір кеш-рядка: 16, 64, 128, 256 байтів [4].

Оскільки в кеш-пам’яті знаходяться копії елементів основної пам’яті, робота з кеш-пам’яттю при доступі по читанню і запису розрізняється. Чи тання даних з кеш-пам’яті не викликає жодних проблем. Дані, якщо вони є, передаються процесору. В цьому випадку йдеться про потрапляння процесора в кеш-пам’ять. Якщо копії даних з необхідною адресою немає, то йдеться про промах процесора і виконують доставку копій необхідних даних в кеш-пам’ять.

Однак в разі доступу за записом має бути забезпечена когерентність (узгодженість) кеш-пам’яті і основної пам’яті: відповідність між даними в оперативній пам’яті і кеш-пам’яті, що забезпечується внесенням змін до тих областей оперативної пам’яті, дані в яких в кеш-пам’яті піддалися модифікації. Когерентність даних забезпечується паралельно з основними обчисленнями.

Існує декілька способів її реалізації, відповідно, декілька режимів роботи кеш-пам’яті.

Один, найпростіший в реалізації, але не найефективніший за швидкодією, спосіб передбачає внесення змін до основної пам’яті відразу після зміни даних в кеш-пам’яті. При цьому процесор простоює в очікуванні завершення записування в основну пам’ять. У основній пам’яті підтримується правильна копія даних кеш-пам’яті, і при заміні рядків не потрібно жодних додаткових дій. Кеш-пам’ять, що працює в такому режимі, називається пам’яттю з наскрізним записуванням.

Інший спосіб передбачає відображення змін в основній пам’яті лише у момент заміни кеш-рядка в кеш-пам’яті. Якщо дані за адресою пам’яті, в яку необхідно здійснити запис, знаходяться в кеш-пам’яті, то йде записування лише в кеш-пам’ять. За відсутності даних в кеш-пам’яті виконується записування відразу в основну пам’ять. Такий режим роботи кеш-пам’яті отримав назву зворотного записування.

Існують також проміжні варіанти, при яких запити на зміну в основній пам’яті буферизуються і не затримують процесор на час операції записування в пам’ять. Це записування виконується за можливості доступу контролера кеш-пам’яті до основної пам’яті [1 – 3].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Дайте означення сегментної організації пам’яті.
2. Як сегмент і зміщення формують фізичну адресу?
3. Наведіть приклади визначення логічної адреси на підставі фізичної.
4. Дайте означення стекової пам’яті.
5. Яке призначення стекової пам’яті?
6. Поясніть принцип дії апаратного стека.
7. Поясніть принцип дії апаратно-програмного стека.
8. Дайте порівняльну характеристику апаратного і програмного стеків.
9. Пояснить призначення регістра SP.
10. Що таке асоціативна пам’ять?
11. Яку роль в МП відіграє кеш-пам’ять?
12. Поясніть механізми роботи кеш-пам’яті.

6 ПРИСТРОЇ ВВЕДЕННЯ/ВИВЕДЕННЯ

6.1 Інтерфейс введення/виведення

Одним з найважливіших завдань проектування МПС є організація взаємодії із зовнішніми пристроями – джерелами і приймачами даних. Прикладами пристроїв введення/виведення (ПВВ), що є як джерелами, так і приймачами інформації, є накопичувачі на гнучких і твердих магнітних дисках. До пристроїв введення належать перемикачі, клавіатура, аналого-цифрові перетворювачі (АЦП), датчики двійкової інформації, а до пристроїв виведення – індикатори, світлодіоди, дисплеї, друкувальні пристрої, цифро-аналогові перетворювачі (ЦАП), транзисторні ключі, реле, комутатори. ПВВ відрізняються: розрядністю даних, швидкодією, протоколами, тобто певним порядком обміну, керівними сигналами. Дані у ПВВ змінюються у довільний або чітко визначений момент часу. Зазвичай інтерфейс складається з одного або декількох портів введення/виведення та схем керування ними.

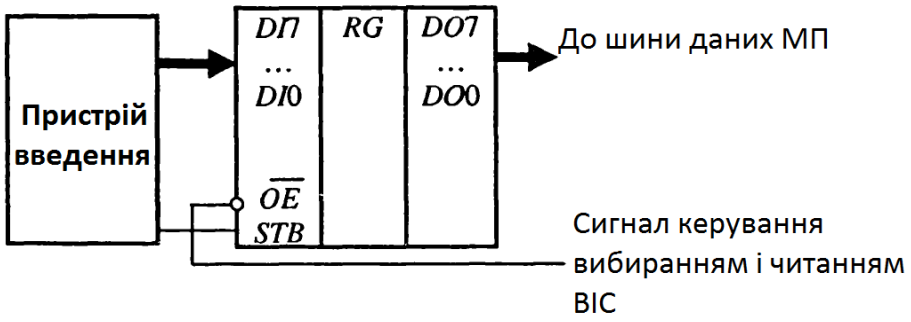
Проектуючи інтерфейс введення/виведення (ІВВ), необхідно забезпечити:

- зберігання інформації, яка надходить від ПВВ;
- доступ до інформації з боку МП;
- керування обміном;
- перетворення форматів даних [1].

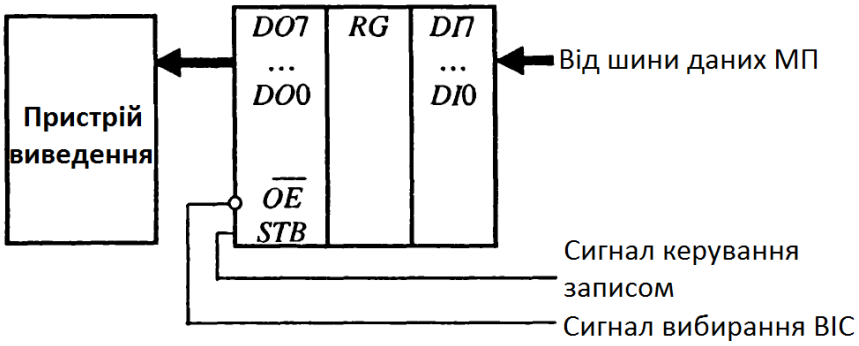
6.2 Зберігання інформації та доступ до неї з боку МП

Введення та виведення інформації виконується за допомогою портів введення/виведення, які являють собою 8- або 16-розрядні регістри зі схемами вибирання та керування читанням/записуванням. Як порти можуть бути використані буферні регістри, наприклад, і8282, і8285, КР580ИР82, КР589ИР12, КР580ВВ55. Використання регістра КР580ИР82 для з’єднання з пристроєм введення та пристроєм виведення показано на рис. 6.1.

Якщо регістр використовується як порт введення (рис. 6.1, а), то дані від пристрою введення надходять у регістр по лініях *DI*7-*DI*0 і записуються за стробом STB. Вихідні дані *DO*7-*DO*0 порту надходять у МПС по шині даних. Мікропроцесор формує також сигнал керування читанням і вибиранням порту, який надходить на вхід $\overline{OE}$.



а



б

Рисунок 6.1 – Використання регістра КР580ИР82 для з'єднання:
а – з пристроєм введення; б – з пристроєм виведення

Якщо регістр використовується як порт виведення (рис. 6.1, б), то дані від МП надходять по шині даних на входи *DI*7-*DI*0 порту і супроводжуються сигналами керування записуванням і вибиранням ВІС. Вихідні дані *D*07-*DO*0 порту надходять у пристрій виведення.

Введення або виведення даних можна здійснювати двома способами:

- з використанням окремого адресного простору ПВВ;
- з використанням спільного з пам'яттю адресного простору, тобто з відображенням на пам'ять [1].

6.3 Керування обміном

Існують три способи керування обміном: програмний обмін, обмін за перериванням, обмін у режимі ПДП.

6.3.1 Програмний обмін

Програмний обмін ініціюється МП і здійснюється під його керуванням. Розрізняють простий програмний обмін і програмний обмін за стробом готовності. При простому програмному обміні вважається, що ПВВ у будь-який момент готовий до обміну. У разі обміну за стробом готовності ПВВ сповіщає про свою готовність до обміну стробом. Наприклад, видача 8-розрядних даних супроводжується дев'ятим бітом – стробом. За такого обміну схема інтерфейсу містить тригер або порт керування для зберігання

інформації про готовність зовнішнього пристрою до обміну. Процесор опитує відповідний розряд порту керування для визначення стану готовності зовнішнього пристрою [4].

6.3.2 Обмін за перериванням

Обмін за перериванням ініціюється ПВВ і здійснюється під керуванням МП. У цьому разі сигнал готовності ПВВ до обміну використовується як запит переривання і надходить до програмовного контролера переривань (ПКП) (рис. 6.2).

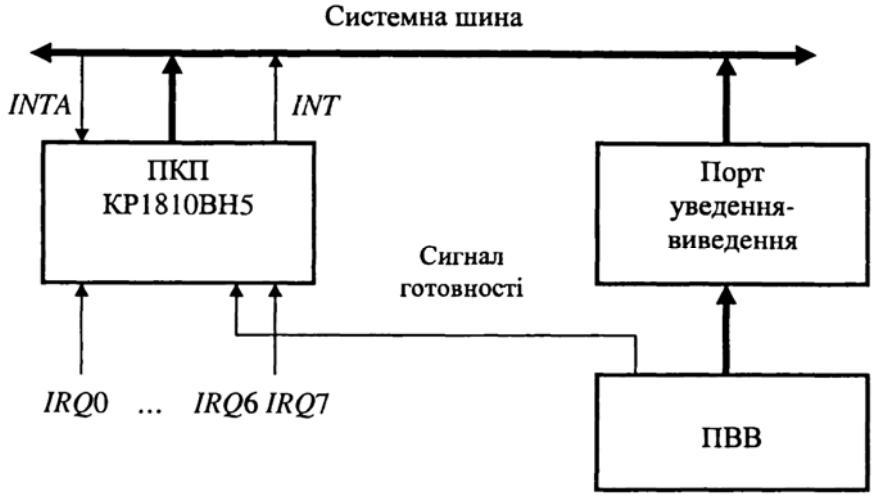


Рисунок 6.2 – Схема обміну за перериванням

Уведення або виведення здійснюється за підпрограмою обробки запиту переривання.

Пристрій введення/виведення формує сигнал готовності *IRQ*, коли він готовий до обміну. ПКП (рис. 6.2) здатний сприйняти 8 сигналів *IRQ*1-*IRQ*0. На рис. 6.2 сигнал готовності ПВВ надходить на вхід *IRQ*6. Сигнал готовності ПВВ являє собою вихідний сигнал тригера, який фіксує стан готовності *READY*. На виході ПКП асинхронно з діями МП формується сигнал *INT*. Заздалегідь не відомо, у який момент і які периферійні пристрої ініціюють переривання. Реагуючи на сигнал *INT*, МП перериває виконання програми, ідентифікує пристрій, переходить до підпрограми обслуговування переривань роботи цього пристрою, а після її завершення відновлює виконання перерваної програми. За командою *INT* вміст програмного лічильника та прапорців автоматично запам’ятовується у стеку. Вміст акумулятора та РЗП необхідно занести у стек за допомогою команд *PUSH* у підпрограмі обробки переривання.

У кожному МП реалізовано особливу структуру системи переривань. Однак загальна послідовність обміну за перериванням містить такі дії:

- ПВВ генерує сигнал готовності, який викликає появу сигналу переривання, що подається на вхід *INT* МП;

- МП завершує виконання поточної команди і, якщо переривання дозволені (не замасковані), формує сигнал *INTA* підтвердження переривання;

- МП здійснює запам'ятовування вмісту акумулятора, програмного лічильника, РЗП у стеку;

- МП ідентифікує пристрій, що викликав переривання, і виконує відповідну підпрограму обслуговування переривання;

- за допомогою команд *POP* відновлюються значення вмісту акумулятора та РЗП зі стека;

- за командою повернення з переривання *RET*, що є останньою командою підпрограми обслуговування переривання, відновлюються значення програмного лічильника та прапорців, і продовжується виконання перерваної програми.

Обмін за перериванням продуктивніший від програмного обміну, оскільки не потребує часу для опитування стану готовності ПВВ до обміну [1].

6.3.3 Обмін у режимі ПДП

Обмін у режимі ПДП ініціюється ПВВ і здійснюється під керуванням контролера ПДП без участі МП. За необхідності обміну між ПВВ і пам'яттю немає потреби у пересиланні даних через МП. Дані за допомогою контролера ПДП пересилаються безпосередньо з ПВВ у пам'ять або навпаки. ПДП при виконанні операцій введення/виведення дозволяє значно збільшити швидкість передавання даних і підвищити ефективність використання засобів МП. Схему обміну в режимі ПДП показано на рис. 6.3.

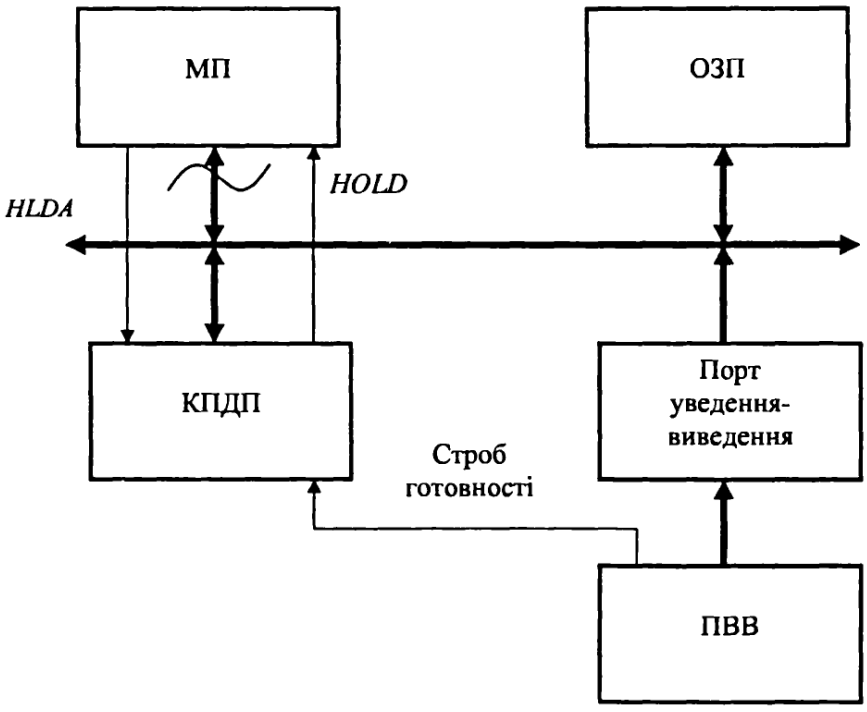


Рисунок 6.3 – Схема обміну у режимі ПДП

Контролер прямого доступу приймає запит від ПВВ, формує сигнал запиту захоплення шин МП *HOLD* і, отримавши від МП дозвіл *HLDA*, формує адреси пам'яті та керувальні сигнали $\overline{MEMR}$, $\overline{IOW}$ у разі читання пам'яті або $\overline{MEMW}$, $\overline{IOR}$ у разі запису в пам'ять.

Інформацію про область пам'яті, що використовується при обміні у вигляді початкової адреси і довжини масиву, завантажують у контролер ПДП під час програмування. Продуктивність обміну в режимі ПДП є найвищою [1].

6.4 Перетворення форматів даних

Якщо розрядність даних, якими оперує МП, менша від розрядності даних, якими оперує ПВВ, то для узгодження розрядності збільшується кількість портів введення/виведення. Якщо розрядність даних, якими оперує МП, більша від розрядності даних, якими оперує ПВВ, то для узгодження розрядності виконується пакування даних, отриманих із декількох джерел, у одне слово потрібної розрядності або доповнення нулями. Для перетворення послідовного коду на паралельний і навпаки використовують контролер послідовного обміну [3, 4].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як забезпечується зберігання інформації, що надходить від ПВВ?
2. Назвіть способи адресування портів введення/виведення.
3. Наведіть порівняльну характеристику видів обміну.
4. Які існують типи програмного обміну?
5. Наведіть структурну схему обміну за стробом готовності.
6. Наведіть структурну схему обміну за перериванням.
7. Наведіть структурну схему обміну в режимі ПДП.
8. Як відбувається запам'ятовування вмісту акумулятора, РЗП, програмного лічильника та прапорців при обміні за перериванням?
9. У яких випадках є доцільним застосування ПДП?

7 СИГНАЛЬНІ ПРОЦЕСОРИ

7.1 Загальні відомості

Для побудови систем цифрової обробки сигналів використовуються спеціалізовані процесори – цифрові сигнальні процесори. Неможливість або неефективність застосування універсальних процесорів для вирішення задач цифрової обробки сигналів пов'язана, з одного боку, з їх низькою продуктивністю на зазначених завданнях , а з іншого боку – з їх надмірністю для даних завдань.

Особливостями сигнальних процесорів є малоразрядна (40 і менше) обробка чисел з плаваючою крапкою, використання переважно чисел з фіксованою крапкою розрядності 32 і менше, а також орієнтація на нескладну обробку великих масивів даних. Відмінною особливістю задач цифрової обробки сигналів є потоковий характер обробки великих обсягів даних у реальному режимі часу, що потребує високої продуктивності процесора і забезпечення можливості інтенсивного обміну з зовнішніми пристроями. Відповідність цим вимогам досягається нині завдяки специфічній архітектурі сигнальних процесорів і проблемно-орієнтованій системі команд.

Сигнальні процесори мають високий ступінь спеціалізації. У них широко використовуються методи скорочення тривалості командного такту (характерні і для універсальних RISC-процесорів), такі як конвеєризація на рівні окремих команд, розміщення операндів більшості команд в регістрах, використання прихованих (тіньових) регістрів для збереження стану обчислень при перемиканнях, поділ пам'яті команд і даних (гарвардська архітектура). У той же час для сигнальних процесорів характерною є наявність апаратного помножувача, що дозволяє виконувати множення двох чисел за один такт. В універсальних процесорах множення зазвичай реалізовується протягом кількох тактів, як послідовність операцій зсуву та складання. Іншою особливістю сигнальних процесорів є внесення в систему команд таких операцій, як множення з накопиченням, реверсування порядку розташування бітів адреси, операції над бітами. У сигнальних процесорах реалізується апаратна підтримка програмних циклів, кільцевих буферів, обробки переривань.

Реалізація однотактного множення, а також команд, що використовують як операнди вміст комірок пам'яті, обумовлює порівняно низькі тактові частоти роботи цих процесорів.

Сигнальні процесори різних компаній-виробників утворюють два класи, що істотно відрізняються за ціною: дешевші процесори обробки даних в форматі з фіксованою крапкою і дорожчі процесори, які апаратно підтримують операції над даними у форматі з плаваючою крапкою.

Використання в сигнальній обробці даних у форматі з плаваючою крапкою обумовлено декількома причинами. Для багатьох завдань, пов'язаних з виконанням інтегральних і диференціальних перетворень, особливе значення має точність обчислень, забезпечити яку дозволяє експонентний формат подання даних. Алгоритми компресії, декомпресії, адаптивної фільтрації в цифровій обробці сигналів пов'язані з визначенням логарифмічних залежностей і вельми чутливі до точності подання даних в широкому динамічному діапазоні значень. Робота з даними у форматі з плаваючою крапкою істотно спрощує обробку, оскільки не потребує виконання операцій округлення і нормалізації даних, відстеження ситуацій втрати точності та переповнення.

Недоліком, який випливає з цього, є висока складність функціональних пристроїв, що виконують обробку даних в форматі з плаваючою крапкою, необхідність використання більш складних технологій виробництва мікросхем і, як наслідок, висока вартість процесорів.

Нині став популярний і інший підхід до отримання високої продуктивності. Велика кількість транзисторів на кристалі може бути використана для створення симетричної мультипроцесорної системи з більш простими процесорами.

Ці процесори, які називаються медійними процесорами, створювалися, виходячи з потреби обробки в реальному часі відео- і аудіоінформації в мультимедійних ПК, ігрових приставках, побутових радіоелектронних приладах. Вартість медійних процесорів досить низька (близько $ 100), а значення показника продуктивність/вартість на 2 – 3 порядки більше. Пікове значення продуктивності медійних процесорів становить кілька мільярдів цілочисельних операцій в секунду [5, 6].

7.2 Основні елементи архітектури сигнальних процесорів

7.2.1 Помножувач-акумулятор

Помножувач-акумулятор (MAC) являє собою сукупність помножувального пристрою і накопичувального суматора. Поєднання помножувального пристрою (помножувача) з акумулятором (суматором результату) ефективно для виконання основної процедури обробки сигналів.

Особливості помножувача-акумулятора в DSP:

- виконання операції множення-накопичення за один цикл;
- достатнє число охоронних бітів (бітів розширення) в акумуляторі;
- детектування переповнення до його настання;
- округлення без зміщення;
- арифметика насичення;
- внутрішні магістралі зворотного зв'язку для скорочення часових втрат в циклах, прискорення обчислень.

Вихід MAC здійснюється через систему регістрів, які реалізують «насичення» при перевищенні розрядності результату операції, приводять до результату у форматі шини даних та/або передають число подвійної точності у вигляді двох слів MSW, LSW (most-significant word, least-significant, відповідно) . Для цього регістр акумулятора розділений на частини LSW, MSW і EX (розширення). Для детектування майбутнього переповнення весь регістр EX і старший біт MSW тестуються логікою детектора переповнення. Крім цього, використовується зсування, яке призначене для двох випадків: лівий зсув усуває надлишкові знакові біти (для чисел в додатковому коді) для подальших обчислень значення, а правий зсув знижує масштаби чисел в блокових операціях з плаваючою крапкою. Логіка «насичення» перетворює дані, які отримані в результаті обчислень і виходять за розрядну сітку (за формат) шини даних. Якщо є переповнення, логіка «насичення» встановлює в регістр пристроїв максимальне/мінімальне значення (залежно від знака). Функція насичення аналогічна насиченню аналогових сигналів, наприклад, в лінійних підсилювальних схемах і запобігає серйозним помилкам в роботі DSP систем .

Розширення обчислювальних можливостей MAC ґрунтується на застосуванні додаткових проміжних регістрів і трактів зворотного зв’язку [7, 8].

7.2.2 Арифметико-логічний пристрій

У цифрових сигнальних процесорах арифметико-логічний пристрій (ALU), взагалі-то, такий, як в універсальних процесорах і виконує звичайний набір команд. Відрізняє їх лише можливість виконання команд за один такт, конвеєризація повторюваних операцій, але зі збереженням прозорості для наскрізних потоків команд. Крім того, в DSP ALU тісно пов’язані з пристроями, такими як зрушувач, які для збереження динамічного діапазону підтримують необхідний масштаб даних. Вимоги, які має задовольняти ALU DSP:

- достатня точність з можливістю нарощування розрядності для високоточних обчислень;
- виконання арифметичних функцій: додавання, віднімання, додавання з перенесенням, віднімання з позикою, формування абсолютного значення;
- виконання логічних функцій: AND, OR, XOR, логічне заперечення;
- формування прапорців: нуль, дорівнює, більше ніж, менше ніж, перенос;
- можливість виконання поділу;
- ефективне пересилання даних і обчислення: пересилання даних і арифметика в єдиному командному циклі;
- можливість завантаження двох операндів в ALU в кожному циклі;
- достатній розмір реєстрових файлів: подвійний набір для перемикання контексту (збереження даних на час переривань) або наявність реєстрового файлу для швидкого доступу і зберігання проміжних результатів;

- виконання умовних операцій: складання і зсув, якщо встановлено прапорець;

- наявність необхідних магістралей зворотного зв'язку: вихід на вхід (акумулятор) і, якщо встановлюється зв'язок зі з двигуном, то входи ALU зв'язуються з виходом зрушувача або входи зрушувача з виходом ALU;

- опції конвеєрного і наскрізного проходів на вхідні регістри;

- тісна взаємодія з двигуном для операцій масштабування;

- можливості виконання обчислень з подвійною точністю з мінімальними тимчасовими витратами.

Пересилання даних і арифметичні операції можуть виконуватися в єдиному циклі за допомогою двопортових регістрів. У цих регістрах записування і читання виконуються за один цикл. Вміст регістра зчитується в ALU на початку циклу (по передньому фронту тактового імпульсу) і забезпечує ALU операндом, а нове значення завантажується з шини даних по задньому фронту. Для максимальної продуктивності обидва операнди ALU (з X і Y регістрів) мають завантажуватися в єдиному циклі [8, 9].

7.2.3 Зрушувач

Зрушувач в DSP призначений для масштабування чисел з метою запобігання переповнення і втрати значущих розрядів, для виконання перетворень чисел з фіксованою крапкою в числа з плаваючою крапкою і навпаки. На відміну від процесорів загального призначення, де зсув здійснюється на один біт за цикл, в DSP зрушувач має бути здатним виконувати зсув слова на задану кількість бітів за один цикл.

Можна виконувати зрушення за допомогою помножувального пристрою (помножувача), домножаючи на 2, 4, ... 2n, проте повний комплект операцій зсуву краще виконувати з двигуном, спеціально спроектованим для цих цілей.

Зрушувач виконує арифметичні і логічні зрушення.

До основних операцій зрушувача відносяться:

- n-розрядні арифметичні і логічні зрушення;

- циклічні зрушення слів (обертання і злиття);

- тестування /встановлення /очищення біта.

Функції зрушувача DSP :

- нормалізація;

- денормалізація;

- блокові операції з плаваючою крапкою.

Це розширені функції зрушувача. Основним елементом зрушувача є детектор порядку числа. Він реалізований як пріоритетний шифратор та інформує про наявність ідентичних старших розрядів слова, і, тим самим, вказує, на яку кількість розрядів вліво може бути зрушене слово без змінення мантиси.

Нормалізація – перетворення числа з фіксованою крапкою в число з плаваючою крапкою, яке формує мантису і порядок. Вхідне значення зсувається вліво так, щоб крайній правий знаковий біт зайняв свою позицію. Кількість розрядів, на яку потрібно зрушити число, перетворюється на двійкове слово (код) за допомогою «детектора порядку» і записується в регістр порядку (EXP). Схема, яка перетворює число, являє собою пріоритетний шифратор, тобто тривіальну комбінаційну логічну схему; для $2b$ входів і b виходів такої схеми вихідне значення буде являти собою двійковий код, відповідний номеру (адреси) найвищого встановленого розряду.

Нормалізація – двокрокова операція: по-перше, детектор порядку обчислює необхідне значення зсуву, і потім це значення застосовується для управління з двигуном.

Денормалізація або перетворення числа з плаваючою крапкою в число з фіксованою крапкою – операція, обернена нормалізації. Порядок числа, завантаженого через шину даних в регістр порядку формує керівний код для зрушення. Молодші біти мантиси, а отже і точність, при цьому втрачаються. Додаткові функції механізму управління денормалізації дозволяють виконувати логічні зрушення, зрушення для слів подвійної довжини, циклічні зрушення, перестановку байтів в слові.

Блокові перетворення з плаваючою крапкою визначають порядок найбільшого числа з деякого масиву даних, і цей єдиний порядок числа потім пов’язують з усім масивом (блоком) і управляють його масштабом. Для цих операцій порядок визначається за допомогою спеціальної логіки. При проходженні першого елемента масиву його порядок визначається і замикається в регістрі – BLOC EXP REG. Потім визначається порядок наступного елемента масиву, поміщається в регістр EXP REG і порівнюється з поточною величиною, що знаходиться в BLOC EXP REG.

Найменше з двох цих значень (оскільки порядок від’ємний) запам’ятовується в BLOC EXP REG як новий блоковий порядок. І так далі до кінця масиву, поки не буде знайдено найменше число.

Очевидно, що застосування блокових перетворень з плаваючою крапкою можливо тільки тоді, коли дані являють собою досить однорідний за масштабом масив. Сигнали (або їх фрагменти), вагові коефіцієнти зазвичай задовольняють цій умові. В інших випадках використовують безпосередні перетворення з плаваючою крапкою [9, 10].

7.2.4 Генератор адрес даних

Генератор адрес даних (DAG) призначений для формування адрес операндів, розташованих у пам’яті даних (DM) і в пам’яті програм (PM).

DAG має один або більше вихідних регістрів з можливістю записування і читання, що містять числа, які вказують на розташування в пам’яті даних і констант (коефіцієнтів). Він забезпечує швидку і гнучку адресацію, дуже істотну в алгоритмах цифрової обробки сигналів, які зазвичай потре-

бують кілька операцій пересилань, записування і читання з елементами обробки в кожному командному циклі. Спеціальні формувачі адреси використовуються, наприклад, при скануванні масиву даних та генерації адрес для алгоритмів швидкого перетворення Фур'є. Тут істотно підвищується швидкість при переміщенні та формуванні адрес із ALU в спеціальний пристрій.

Сканування масиву даних відноситься до найбільш універсальних операцій, які використовуються в алгоритмах цифрової обробки сигналів. DAG зі своїм власним ALU робить можливим сканування за заданою схемою. Більшість «таблиць даних» у цифровій обробці сигналів виглядають як кільцеві (або циклічні); таблиці мають скінченну довжину, програма обробки циклюється: за досягнення кінця таблиці виконується перехід на її початок. Автоматичне визначення кінцевого пункту циклу звільняє процесор від множини додаткових обчислень і контрольних кроків. У високопродуктивних DSP генератор адреси даних комбінує ці функції в межах одного складеного виразу:

$$Y = Y + R; \text{ IF } R \geq \text{END THEN } Y = \text{BEGIN},$$

де Y – значення адреси,

R – величина зсуву,

BEGIN і END задають межі таблиці.

Властивості DAG:

- зрушення – забезпечує алгоритм обертання таблиці за рахунок формуванням парних адрес;

- біт-реверсування – перестановка бітів в адресному слові, що використовується в обчисленнях для пересортовування даних за необхідним порядком;

- логічні функції і маскування – використовуються при зверненні до таблиці даних, наприклад, в алгоритмі інтерполяції за методом Ньютона;

- гнучка модульна арифметика – при скануванні масиву даних DAG обчислюється прирощення покажчика і при досягненні кінця буфера здійснюється його скидання до базового значення [10, 11].

7.2.5 Формувач послідовності команд

Формувач послідовності (потоку) команд (PSqr) призначений для генерації адреси команд, які визначають хід програми. Він усуває тимчасові витрати , пов'язані з відстеженням зміни (нарощування) вмісту лічильника команд (враховуючи умовну логіку і цикли), управління підпрограмами і зовнішніми перериваннями. Наприклад, програма може мати лічильники, які тестуються для визначення моментів завершення циклів. Ці функції підтримуються в PSqr регістровими стеками (стек лічильника або стек циклу). Адреси повернення відновлюються зі стека лічильника команд після

завершення підпрограми та обслуговування запиту на переривання. PSqr має також доступ до статусної інформації (знаки, переповнення, заповнення стеків), генерується різними обчислювальними модулями. Ця статусна інформація використовується в умовних командах (в умовних переходах – Jump If ...). Таким чином, для PSqr визначено декілька головних функцій:

- управління нормальним ходом програми, інкрементація лічильника команд в кожному циклі;
- відстеження адресації при зверненні до підпрограм, управління стеком повернення;
- управління програмними циклами, контроль лічильників циклів;
- перехід на відповідні спеціальні програми при виявленні переповнення або необхідності перемасштабування даних;
- обслуговування переривань від пристроїв введення/виведення, переходи на відповідні підпрограми і повернення до перерваної задачі після завершення.

Для реалізації цих функцій PSqr має містити такі компоненти:

- лічильник команд;
- логіка перевірки умов;
- стеки для адрес повернення підпрограм і переривань, вмісту лічильників циклів, адрес переходів, станів (прапорів).

Стеки мають бути досить великими, щоб забезпечити вкладення підпрограм і кілька рівнів переривання. Потрібно зазначити, що конвеєрні схеми для підвищення продуктивності в PSqr застосовувати не рекомендується, оскільки затримки конвеєризації створюють труднощі для розробників програмного забезпечення системи [11, 12].

7.2.6 Пам’ять

Очевидно, що вибір типу пам’яті, її конфігурації для високошвидкісної обробки сигналів є одним з ключових рішень DSP і потребує розгляду безлічі особливостей.

Відзначимо основні з них на прикладі пам’яті сімейства ADSP2106x.

Процесори ADSP2106x мають два незалежних блоки двопортової пам’яті. Є також можливість підключення зовнішньої пам’яті з адресним полем до 4-х гігаслів. Для зберігання даних з плаваючою крапкою звичайної довжини використовуються 32-бітові слова, 48-бітові слова призначені для зберігання команд.

Крім того, в процесорах цього сімейства підтримується формат 16-бітових слів, які можуть використовуватися для цілих або дробових значень даних з фіксованою крапкою. ADSP 2106x має три внутрішніх шини, пов’язані з двопортовою пам’яттю. Внутрішні шини пам’яті даних і пам’яті програм керуються процесором, тоді як шини введення/виведення керуються контролером введення/виведення, який розташований в тому ж чипі. Шини введення/виведення забезпечують одночасне передавання да-

них між комунікаційними портами (послідовні порти, паралельні порти, порти зв'язку). Завдяки цій двопортовій структурі звернення до внутрішньої пам'яті з боку процесора і контролера введення/виведення є незалежними і прозорими один відносно одного. У кожному тактовом циклі до кожного блока пам'яті можуть звертатися і процесор, і контролер введення/виведення; звернення до одного і того ж блока не потребує яких-небудь додаткових циклів. Процесор і контролер введення/виведення мають доступ до зовнішньої пам'яті по зовнішній шині через зовнішній порт. Така схема магістралей дозволяє мати єдиний уніфікований адресний простір для зберігання коду і даних [12, 13].

7.3 Система команд

Систему команд сигнальних процесорів розглянемо на прикладі команд процесора ADSP-2100. Описувана нижче система команд дозволяє ефективно реалізовувати такі властивості як:

- цикли з нульовими тимчасовими витратами;
- збереження контексту за один цикл: перемикання повного набору регістрів;
- мультифункціональні команди: арифметика плюс передача даних за один такт;
- завантаження двох операндів ззовні за один такт;
- паралельний доступ до команд і операнд;
- 5 режимів адресації;
- 50 регістрів загального призначення в чипі.

Система команд сприяє виконанню інтенсивних обчислень, забезпечуючи переміщення даних з мінімальними тимчасовими витратами між обчислювальними блоками.

Команди подані у вигляді єдиного 24-бітового слова і виконуються за один такт, за винятком тих випадків, коли дані (тобто коефіцієнти) з програмної пам'яті і наступна команда ще не завантажені в програмний кеш. Мультифункціональні команди комбінують арифметичні команди з командами переміщення даних і реалізуються в одному і тому ж такті.

Регістри, які обчислюються в першій половині такту, можуть використовуватися для запису в другий або, за бажанням, може виконуватися незалежна задача. При написанні програми для ADSP-2100 спочатку записуються арифметичні кроки, потім окремо виписуються всі операції введення/виведення і з них комбінуються мультифункціональні інструкції.

Виклик підпрограм і обслуговування переривань завдяки збереженню контексту за один такт виконані компактно, і, незважаючи на можливість використання альтернативного набору регістрів, вміст вихідних регістрів зберігається незмінним.

Система команд:

- арифметичні команди;

- команди АЛП;
- команди МАС;
- функції зрушувача;
- команди пересилання даних;
- мультифункціональні команди;
- команди управління ходом програми;
- змішані команди.

Арифметичні команди за типом команд поділяються на стандартні, спеціальні та умовні. Стандартні команди, що входять до блоку арифметичних інструкцій можуть виконуватися умовно за результатами тестування арифметичного регістра стану, а також можуть комбінуватися з передачею даних в однотактних мультифункціональних командах. Спеціальні команди являють собою невелику підсистему. При використанні умовних команд перевіряється результат попередньої команди ALU на рівність нулю (EQ), нерівність нулю (NE), більше нуля (GT), більше або дорівнює нулю (GE), менше нуля (LT), менше або дорівнює нулю (LE), перевіряється наявність перенесення (AC, not-AC), знак за х-входом (POS, NEG) і переповнення ALU (AV, not-AV) або MAC (MV, not-MV), перевіряються лічильники циклів на рівність нулю (NOT CE) .

Команди АЛП – це додавання, віднімання, логічні функції, інкрементація, декрементація, очищення, формування абсолютного значення.

Стандартні команди МАС: множення, множення з накопиченням, множення з відніманням, пересилання, насичення, очищення. Умовні операції базуються на вмісті статусних регістрів.

Спеціальна команда МАС – насичення акумулятора. Команда тестує біт переповнення помножувача-акумулятора і якщо біт встановлений, то реалізує функцію насичення регістра (за один такт).

Стандартні функції зрушувача: арифметичні і логічні зрушення, а також операції масштабування чисел (перетворення з плаваючою крапкою, блокові перетворення). Умовні операції грунтуються на вмісті статусного регістра.

Спеціальна функція зрушувача – це безпосереднє зрушення.

Команди пересилання даних містять в собі процедури записування та зчитування між регістрами та між регістрами і обома блоками пам’яті (програмної та даних). Великий набір з 50 регістрів доступний по шині DMD. Сюди входять регістри, що «обрамляють» ALU та MAC, в яких зберігаються вхідні і вихідні дані, а також регістри генератора адреси даних і формувача послідовності команд.

Є такі режими переміщення даних:

- регістрові пересилання (регістр–регістр);
- команди читання/записування для пам’яті даних, операції читання/записування за адресою певного покажчика, читання в регістри з пам’яті, записування у пам’ять даних, записування у пам’ять програм.

Мультифункціональні команди – це команди, що виконуються одночасно, в єдиному циклі. До них відносять:

- мультифункціональні команди з читанням даних;
- мультифункціональні арифметичні команди, які включають арифметики з пересиланням регістр–регістр, з читанням, із записуванням.

Команди управління ходом програми забезпечують:

- обслуговування переривань;
- виклик підпрограм;
- цикли типу DO ;
- команди TRAP.

Змішані команди використовуються для модифікації вмісту індексних регістрів, управління стеками і перемикання альтернативних реєстрових банків [10, 13, 14].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Дайте означення сигнальних процесорів.
2. Які особливості сигнальних процесорів забезпечують їх високий ступінь спеціалізації?
3. Які виділяють класи сигнальних процесорів?
4. Дайте означення формувача послідовності команд.
5. Дайте означення генератора адрес даних.
6. Дайте означення зрушувача.
7. Наведіть функції зрушувача.
8. Дайте означення арифметико-логічного пристрою.
9. Яким вимогам має відповідати арифметико-логічний пристрій?
10. Дайте означення помножувача-акумулятора.
11. Які особливості помножувача-акумулятора?

8.1 Класифікація програмованих логічних інтегральних схем

Найбільш просту будову серед мікросхем програмованої логіки мали програмовані логічні матриці – ПЛМ. Ці мікросхеми складалися з програмованих матриць «І» та «АБО». Як приклад таких мікросхем можна навести мікросхеми РТ1, РТ2, РТ21 серії 556. На рисунку 8.1 зображено внутрішню структуру мікросхеми ПЛМ.

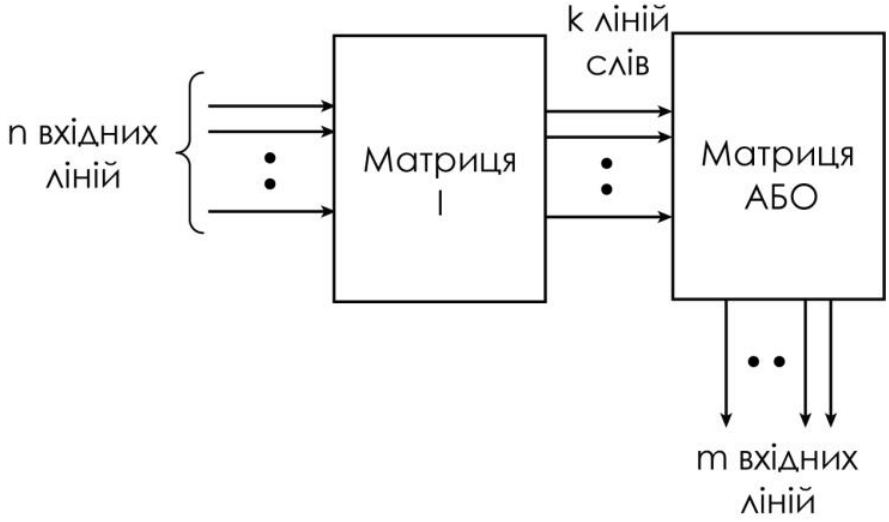


Рисунок 8.1 – Спрощена архітектура ПЛМ

Вхідний сигнал розрядністю *n* потрапляє до першої матриці, де над вхідними даними виконується логічна функція «І». Результати *k* кон'юнкцій входять до матриці «АБО», де над ними виконується операція логічного додавання (кон'юнкції). Таким чином у ПЛМ можна побудувати *m* функцій виду:

$$y = x_1\overline{x_2 x_3 x_4} + \overline{x_1 x_2 x_3 x_4}.$$

В подальшому архітектура мікросхем програмованої логіки ускладнювалась. Змінювалась її структура, кількість та схема зв'язків, додавались нові функції та модулі.

Розглянемо основні архітектурні особливості програмованих логічних інтегральних схем (ПЛІС).

Схематично мікросхему та шляхи проходження сигналів показано на рисунку 8.2.

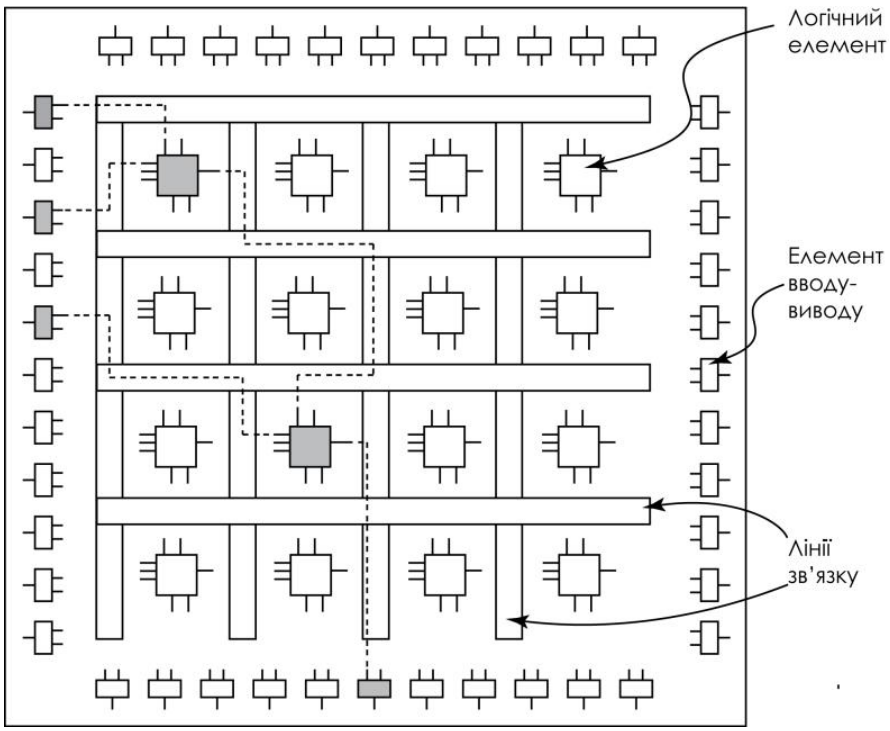


Рисунок 8.2 – Спрощена архітектура ПЛІС

Мікросхема містить елементи введення/виведення, логічні елементи та лінії зв'язку – рядки і стовпці. Спрощений механізм роботи пристрою на ПЛІС виглядає таким чином: сигнал через елементи введення/виведення потрапляє в мікросхему, де в логічному елементі над ним виконуються необхідні операції. Потім по лініях зв'язку через елементи введення/виведення сигнал виходить з мікросхеми.

Програмування та перепрограмування впливає на режими роботи логічних елементів, елементів введення/виведення, зв'язки елементів в мікросхемі – як і в якому порядку з'єднуються логічні елементи, як розташовуються функції в логічних елементах. Для збереження всіх цих параметрів в мікросхемі ПЛІС існує конфігураційна пам'ять, доступ до якої користувач не може отримати, і програмування якої виконується спеціалізованим програмним забезпеченням.

Архітектура та можливості сучасних ПЛІС дуже різноманітні, але у більшості випадків можна навести декілька класифікаційних ознак:

- ступінь інтеграції або логічна ємність;
- архітектура логічного елемента мікросхеми;
- тип запам'ятовувального елемента для пам'яті конфігурації;
- наявність вбудованих модулів (внутрішньої пам'яті, помножувачів, блоків цифрової обробки сигналів, інтерфейсів для різноманітних протоколів та інших).

Ступінь інтеграції або логічна ємність – найбільш важливий показник мікросхем ПЛІС. Сучасні мікросхеми мають у своєму складі сотні мільйо-

нів транзисторів, але мікросхема ПЛІС має дуже велику надлишковість структури, тому для більш коректного визначення обсягу мікросхем виробники використовували такий параметр як кількість еквівалентних логічних вентилів типу 2І-НЕ або 2АБО-НЕ, які б використовувались для реалізації пристроїв тієї ж самої складності, що і на ПЛІС. Такий спосіб розрахунку обсягу ПЛІС використовувався приблизно до 2004 року. Мікросхеми, які випускались на той час, мали обсяг від 10 тисяч еквівалентних логічних елементів (для мікросхеми Flex 10K10 фірми ALTERA) до 5,3 мільйонів еквівалентних логічних елементів (для мікросхеми APEX II фірми ALTERA). Потрібно відмітити, що кожна з фірм, які випускають мікросхеми програмованої логіки, розраховує кількість еквівалентних вентилів за власною методикою, що ускладнює порівняння мікросхем різних типів. Крім того, в структуру мікросхеми крім власне логічних елементів входить ще дуже багато різноманітних вузлів (елементи введення/виведення інформації, блоки пам'яті, ресурси трасування сигналів тощо), кожен з яких описується різною кількістю еквівалентних логічних елементів. Це також ускладнює розрахунок у еквівалентних вентилях.

Тому сьогодні для вимірювання ємності мікросхеми використовують такий параметр як кількість логічних елементів. Але, оскільки архітектура логічних елементів для різних мікросхем навіть одного виробника відрізняється від іншої, то порівняння мікросхем стає ще більш ускладненим.

За архітектурою логічного елемента найбільш часто мікросхеми розділяють на два типи [15, 16]:

- CPLD – Complex Programmable Logic Devices – пристрій зі складною програмованою логікою;
- FPGA – Field Programmable Gate Array – матриця програмованих логічних елементів.

Мікросхеми з архітектурою CPLD реалізовують в логічному елементі складну логічну функцію у вигляді логічного рівняння, яке формується за допомогою функцій «І» та «АБО». Це ускладнений варіант ПЛМ мікросхем. Конфігураційні дані в мікросхемах цього виду зберігаються у вбудованій пам'яті типу Flash, тому для своєї роботи мікросхема не потребує зовнішнього завантажувача конфігурації. Найбільш характерними представниками такого виду ПЛІС можуть слугувати мікросхеми сімейств MAX7000 та MAX3000 фірми ALTERA або CoolRunner та XC9500 фірми XILINX [17,18].

Основна ідея мікросхем FPGA полягає в тому, що будь-яка функція може бути описана таблицею істиності. Тому мікросхеми цього типу реалізують логічну функцію на основі таблиці перекодування, яка за принципом роботи подібна до постійного запам'ятовувального пристрою. Класичним прикладом такого типу мікросхем можна назвати сімейство FLEX 10K фірми Altera.

Тип запам’ятовувального елемента пам’яті конфігурація визначає можливості ПЛІС щодо програмування, перепрограмування і збереження інформації при відключенні живлення.

Сучасні мікросхеми використовують EEPROM, Flash або SRAM технології для виготовлення запам’ятовувального елемента, що дає можливість перепрограмовувати мікросхему. Мікросхеми, що використовують технологію EEPROM або Flash, забезпечують енергонезалежне збереження конфігурації, а також і багаторазове програмування мікросхеми. Мікросхема на SRAM має кожен раз програмуватися при ввімкненні живлення. Конфігураційні дані в цьому випадку зберігаються у зовнішньому ПЗП або у персональному комп’ютері. Окрім стандартних логічних елементів ПЛІС може містити також апаратні блоки, які виконують спеціалізовані функції. Найбільш часто в мікросхемах розміщують блоки пам’яті, помножувачі, блоки фазового автопідстроювання частоти. Кількість та наявність таких блоків визначається складністю та призначенням мікросхеми. Наявність таких блоків, з одного боку, ускладнює мікросхему, підвищуючи її вартість. З іншого боку, апаратні ядра значно потужніші за швидкодією порівняно з такими самими пристроями, створеними на логічних елементах [16].

8.2 Архітектура мікросхем сімейства Cyclone II

Мікросхеми Cyclone відносяться до так званих «дешевих» мікросхем ПЛІС. Перші мікросхеми сімейства Cyclone з’явились у 2002 році. З того часу було випущено кілька сімейств і на сьогоднішній день фірма ALTERA випускає мікросхеми сімейства Cyclone V. Розглянемо архітектуру сімейства Cyclone II.

Загальну архітектуру мікросхеми Cyclone II показано на рисунку 8.3.

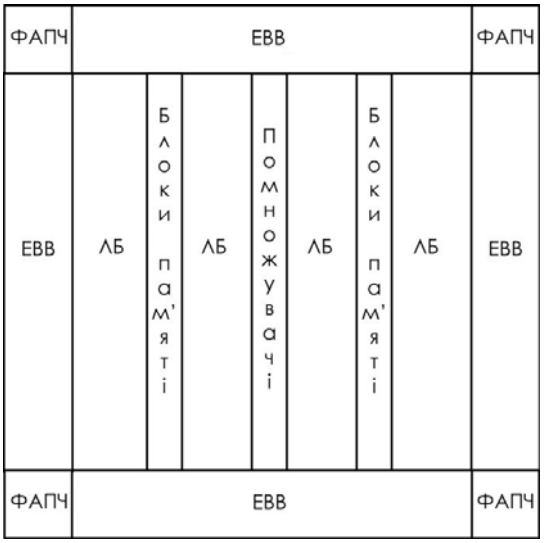


Рисунок 8.3 – Загальна архітектура мікросхеми Cyclone II

В структурі мікросхеми показано лише ті ресурси, які розробник може використовувати в своїх проектах і які доступні йому для програмування. Основний елемент мікросхеми – це логічний блок (ЛБ), який містить в собі логічні елементи (ЛЕ). В масиві логічних блоків розташовано стовпці блоків пам’яті та помножувачів. По периметру мікросхеми розміщуються елементи введення/виведення (ЕВВ), за допомогою яких виконується обмін інформацією мікросхеми з навколишнім світом. По кутах кристала розміщено блоки фазового автопідналаштування частоти (ФАПЧ). Крім показаного на рисунку 8.3 на кристалі розміщується ще вузол конфігурування та конфігураційна пам’ять, які недоступні користувачу. Перелік ресурсів найменшої та найбільшої мікросхем сімейства Cyclone II наведено в таблиці 8.1.

Таблиця 8.1 – Параметри мікросхем сімейства Cyclone II

Параметр	EP2C5	EP2C70
Кількість логічних елементів	4 608	68 416
Кількість блоків пам’яті M4K	26	250
Загальний обсяг пам’яті, біт	119 808	1 152 000
Кількість блоків ФАПЧ	2	4
Кількість вбудованих помножувачів	13	150
Кількість виводів, доступних користувачу	158	622

8.2.1 Архітектура логічного елемента

ПЛІС Cyclone II – це FPGA мікросхема і основою логічного блока в ній є таблиця перекодування. Архітектура логічного блока показана на рисунку 8.4.

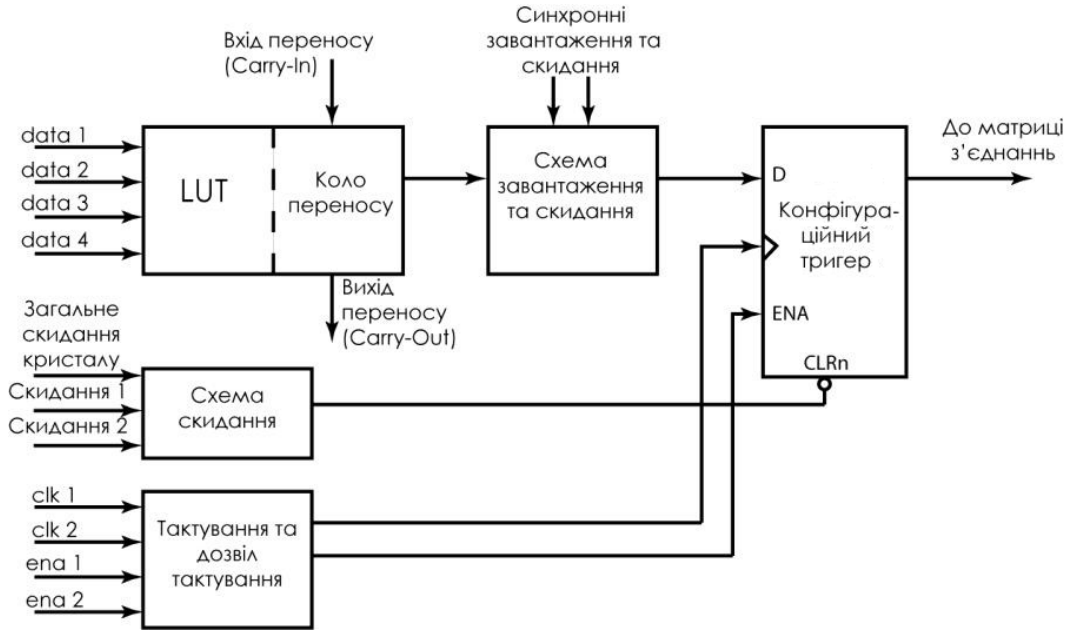


Рисунок 8.4 – Загальна архітектура логічного елемента FPGA мікросхеми

Таблиця перекодування може генерувати будь-яку логіку від чотирьох змінних. Крім таблиці перекодування логічний елемент містить також конфігураційний тригер та схему перенесення.

Логічний елемент може працювати у двох режимах роботи:

- нормальному режимі;
- арифметичному режимі.

В нормальному режимі роботи (рисунок 8.5) на входи таблиці перекодування надходять чотири сигнали даних (data 1, ... data 4), а з виходу дані можуть надходити або на вхід конфігураційного тригера, або через матрицю з'єднань на інші елементи мікросхеми. Крім вхідних даних в логічний елемент з логічного блока надходять також лінії двох локальних сигналів синхронізації (clk 1, clk 2), двох сигналів дозволу роботи (ena 1, ena 2), двох сигналів скидання та сигналу загального скидання кристала, а також сигнали синхронного завантаження та скидання. Конфігураційний тригер, залежно від режиму роботи, може працювати як D, T, JK або RS тригер. Вихідний сигнал тригера надходить в матрицю з'єднань і може подаватися через лінії зв'язку на будь-який елемент мікросхеми.

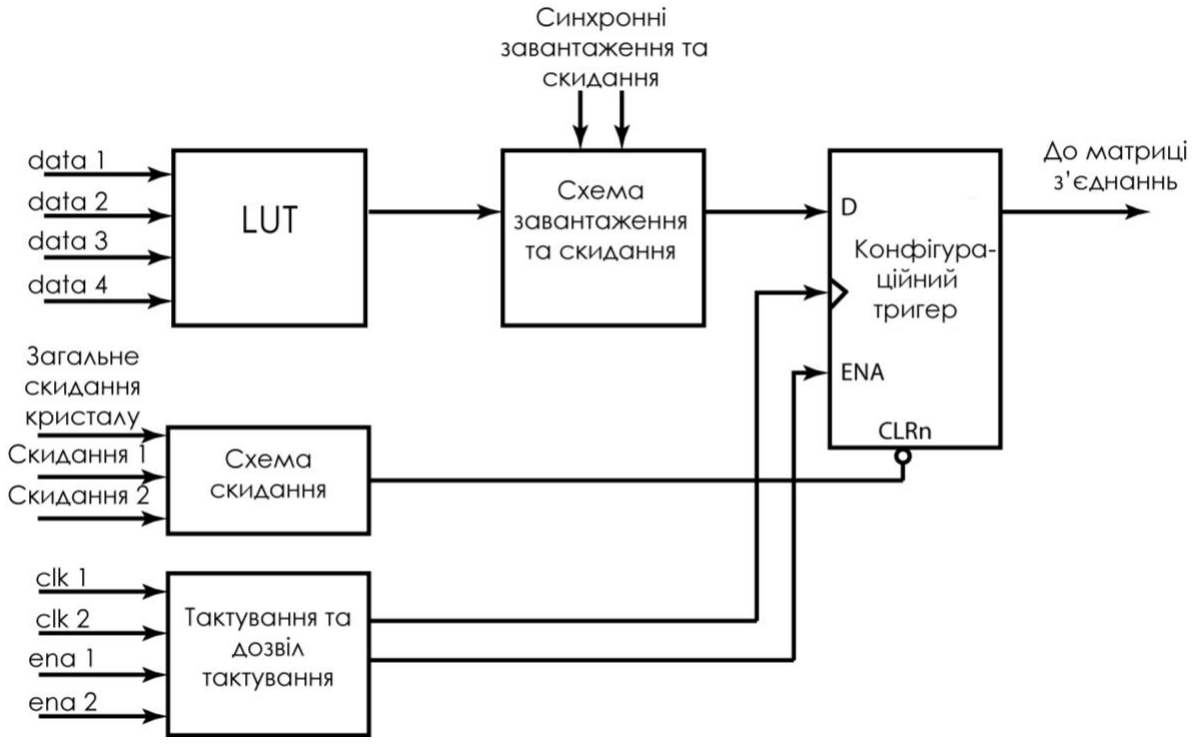


Рисунок 8.5 – Логічний елемент FPGA мікросхеми в нормальному режимі роботи

Перед вивченням арифметичного режиму роботи пригадаємо як виконується операція додавання у двійковому коді.

Для додавання двох багаторозрядних чисел над кожним розрядом (крім молодшого) необхідно виконати такі операції:

$$S_i = A_i \oplus B_i \oplus P_{i-1};$$
$$P_i = A_i B_i + P_{i-1}(A_i \oplus B_i),$$

де S_i – сума і-го розряду,

P_i – вихід перенесення з і-го розряду,

P_{i-1} – вхід перенесення з і-1-го розряду,

A_i, B_i – і-й розряд доданка A та B.

З наведених вище формул видно, що для виконання операції додавання необхідно використати три змінні: A, B, P, які мають надійти до логічного елемента мікросхеми. Крім того, необхідно організувати перенесення з молодшого розряду і до старшого розряду. Таким чином, необхідна інша архітектура логічного елемента, яка показана на рисунку 8.6.

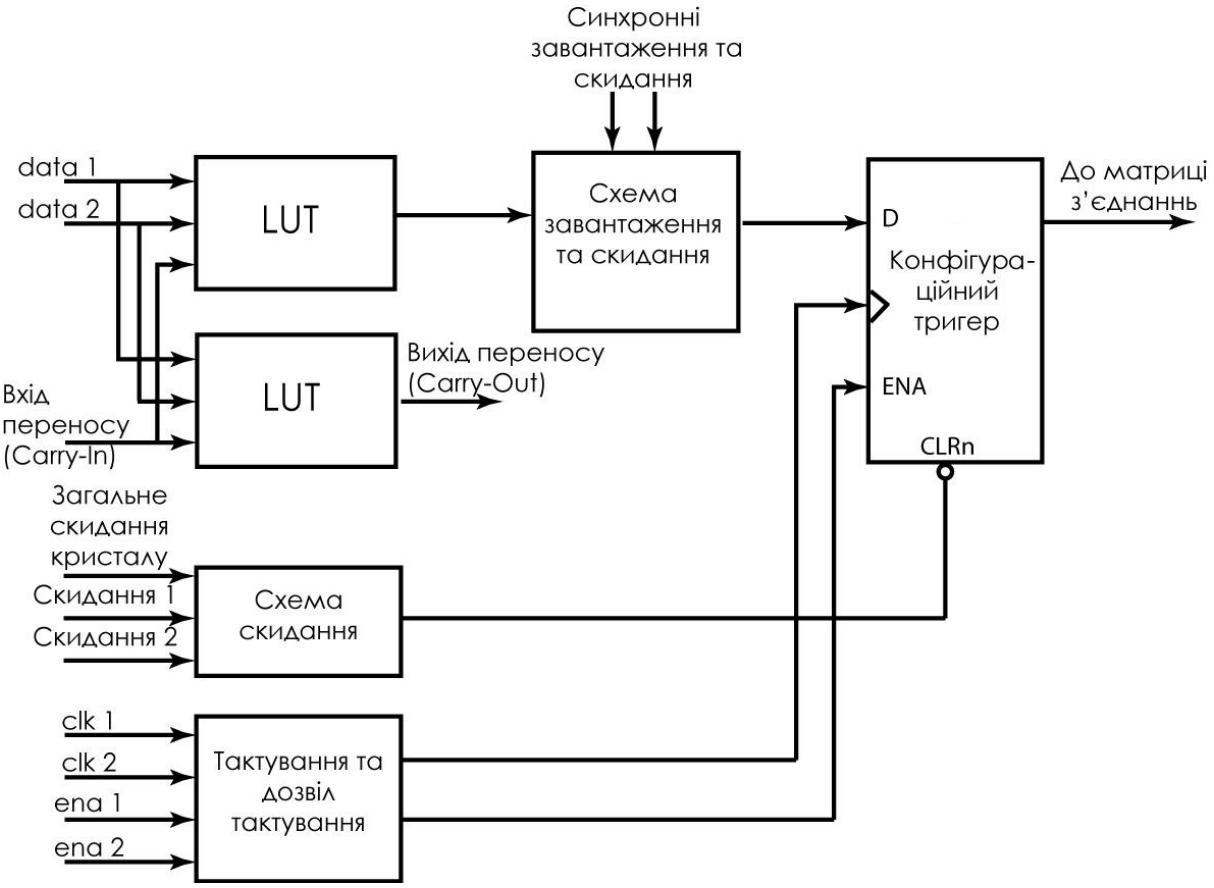


Рисунок 8.6 – Логічний елемент FPGA мікросхеми в арифметичному режимі роботи

Така архітектура використовує таблицю перекодування, що поділена на дві тривходові таблиці перекодування. Одна таблиця використовується для формування сигналу суми, а друга – для формування сигналу перенесення.

Як сигнали даних використовуються два входи даних (data1, data2) та вхід перенесення (Carry-In).

Для складання багаторозрядного числа логічні елементи повинні бути об’єднані у загальну структуру, структуру якої показано на рисунку 8.7. З рисунку видно, що окремі розряди доданків надходять до сусідніх ЛЕ, які формують біти результату. Якщо розрядність доданків перевищує кількість логічних елементів у логічному блоці (для мікросхем сімейства Cyclone II – це 16), то формується сигнал перенесення з одного логічного блоку до іншого (LAB Carry-Out).

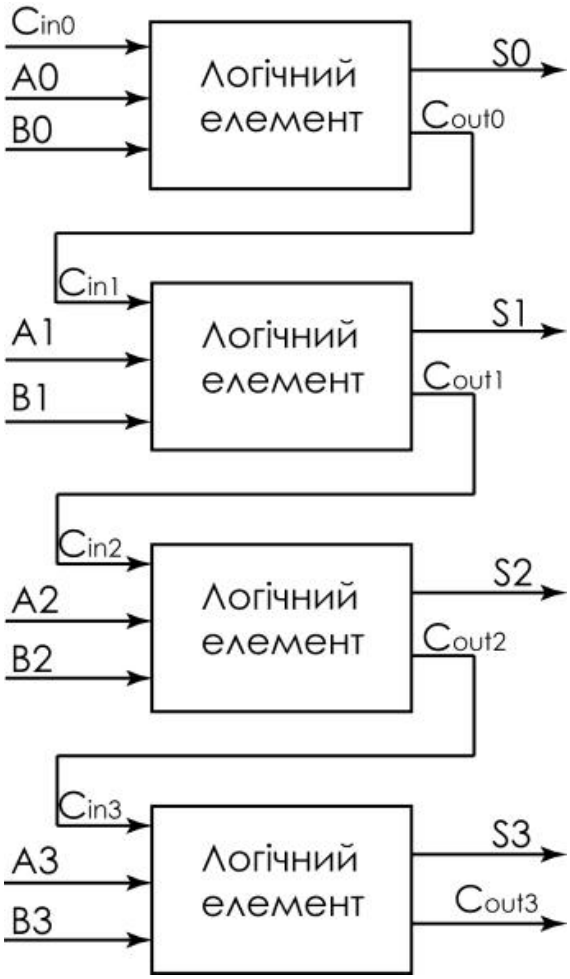


Рисунок 8.7 – Об’єднання логічних елементів для додавання чотирирозрядних чисел

8.2.2 Логічний блок

Логічний блок мікросхеми містить у собі шістнадцять логічних елементів, які з’єднані локальною матрицею з’єднань, кола сигналів керування логічним блоком, коло перенесення для логічних елементів блока та коло перенесення з одного логічного блока до іншого. На рисунку 8.8 показано два логічних блоки та з’єднання їх з глобальною матрицею з’єднань.

Логічні блоки розташовані рядками та стовпцями і з’єднуються між собою за допомогою глобальної матриці з’єднань.

8.2.3 Конфігураційний блок пам’яті М4К

Мікросхеми сімейства Cyclone II містять в собі блоки пам’яті типу М4К, обсяг яких становить 4 098 біт. Загальний обсяг вбудованої пам’яті мікросхеми становить від 104 кбіт до 1 Мбіт, а максимальна тактова частота роботи сягає 250 МГц.

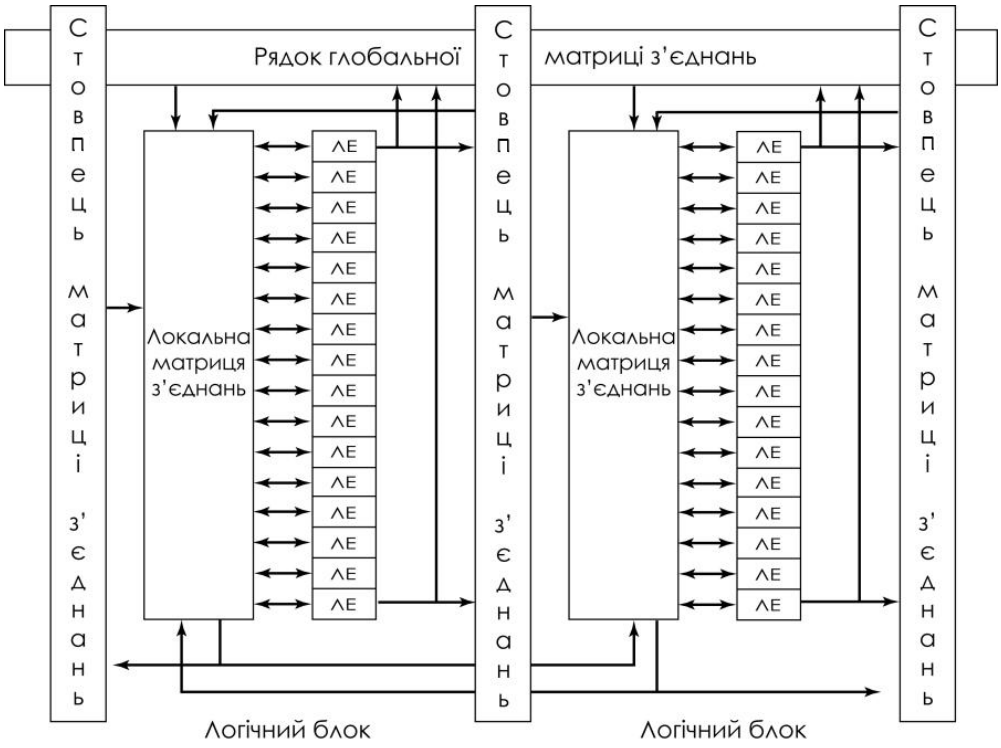


Рисунок 8.8 – Логічні блоки мікросхеми Cyclone II

Блок пам’яті М4К може конфігуруватись в такі види пам’яті:
- оперативна пам’ять;
- постійний запам’ятовувальний пристрій;
- двопортовий ОЗП;
- регістр зсуву;
- блок FIFO.

Крім того, можлива робота з бітом парності.

Основною відмінністю вбудованих блоків пам’яті ПЛІС є їх конфігураційність, тобто блок пам’яті може мати кілька різних варіантів організації, обсяг яких не перевищує обсяг самого блока пам’яті. При роботі блок М4К допускає такі конфігурації: 4К×1, 2К×2, 1К×4, 512×8, 512×9, 256×16, 256×18, 128×32, 128×36.

Якщо ж необхідно використовувати модуль вбудованої пам’яті більшого обсягу, то кілька блоків об’єднуються в один модуль. У мікросхемах сімейства Cyclone є можливість використовувати один модуль М4К для двох однопортових блоків пам’яті, що значно підвищує використання блоків пам’яті.

8.2.4 Архітектура блока ФАПЧ

Блоки фазового автопідналаштування частоти можуть використовуватись для множення та ділення частоти, зсуву фази тактового сигналу відносно основної частоти та отримання програмованої шпаруватості тактового сигналу.

Розглянемо основний принцип роботи системи ФАПЧ, структурна схема якої зображена на рисунку 8.9.

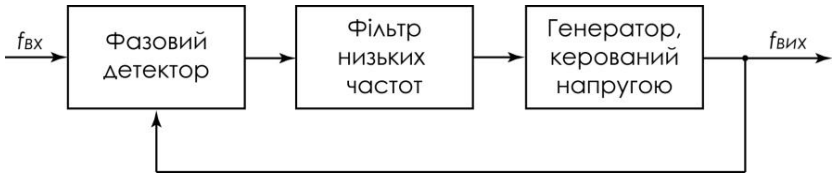


Рисунок 8.9 – Структурна схема ФАПЧ

До складу системи входить фазовий детектор, який визначає різницю в фазі та частоті вхідного сигналу (*fвх*) та сигналу зворотного зв'язку. Вихідний сигнал формується генератором, що керується напругою. Фільтр низьких частот необхідний для запобігання самозбудженню системи вцілому.

В результаті роботи такої системи в ідеальному випадку можна отримати на виході сигнал, що збігається з вхідним сигналом.

При подальшому ускладненні системи (рисунок 8.10) до неї додаються три лічильники – подільники частоти: вхідний (коефіцієнт ділення M), вихідний (коефіцієнт ділення K) та лічильник у ланці зворотного зв'язку (коефіцієнт ділення N).

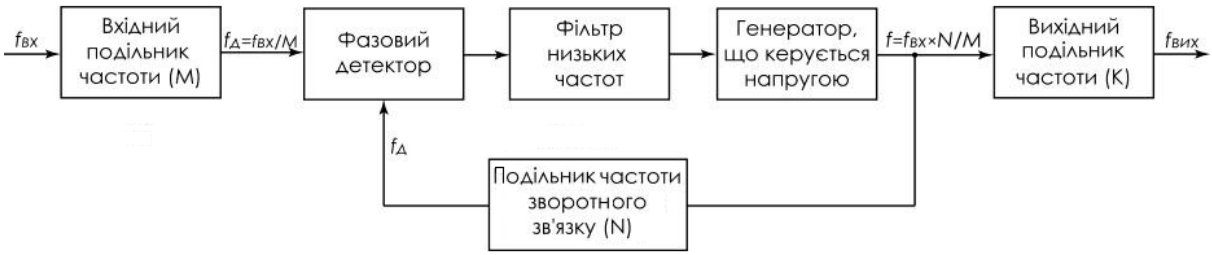


Рисунок 8.10 – Структурна схема ФАПЧ в ПЛІС Cyclone II

Якщо увімкнути у ланку зворотного зв'язку подільник зі змінним коефіцієнтом рахунку N, то частота на виході цього подільника зменшиться в N разів порівняно з сигналом *f*д. Система фазового автопідналаштування частоти буде підтримувати значення частот на вході фазового детектора однаковими. А це означає, що частота на виході генератора, що керується напругою, буде збільшуватись в N разів порівняно з вхідною частотою *f*д. Змінюючи коефіцієнт рахунку подільника зворотного зв'язку можна отримувати різні значення частот генератора.

Додавання до схеми вхідного подільника частоти з постійним коефіцієнтом рахунку M дозволяє отримати низьку частоту для порівняння на

входах фазового детектора. На виході генератора, що керується напругою, додається ще один подільник, але вже зі змінним коефіцієнтом рахунку К. В результаті вихідна частота буде визначатися формулою:

$$f_{вих} = \frac{f_{вх}}{M} N/K .$$

8.2.5 Архітектура вбудованого помножувача і блока цифрової обробки сигналу

Для підвищення швидкодії виконання операцій цифрової обробки сигналів у сучасних ПЛІС вбудовують або помножувачі, або блоки цифрової обробки сигналу.

Структура вбудованого помножувача наведена на рисунку 8.11.

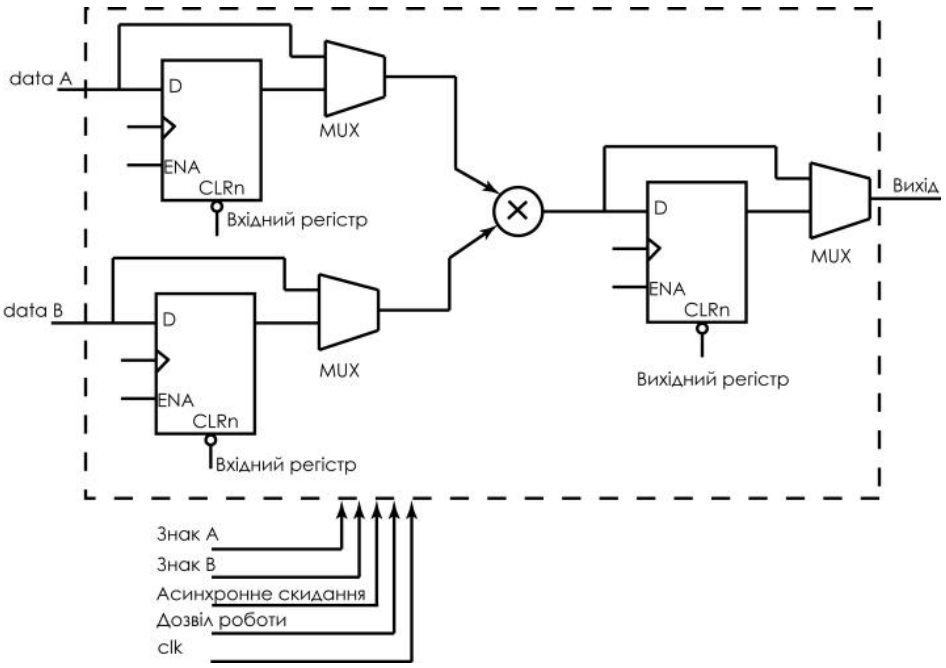


Рисунок 8.11 – Стуктурна схема помножувача у ПЛІС Cyclone II

Вбудований помножувач може працювати в одному з двох режимів: один 18-бітовий помножувач або два незалежних 9-бітових помножувача. Максимальна тактова частота роботи помножувача сягає 250 МГц. Кількість вбудованих 18-бітових помножувачів в мікросхемах сімейства Cyclone II може доходити до 150.

Як видно з рисунка, помножувач містить регістри за входом та виходом, що дає можливість створювати синхронні схеми. За бажанням ці регістри можуть і не використовуватись в схемі. Для цього в схемі використовуються мультиплексори, які формують сигнали на вході та виході помножувача.

В мікросхемах сімейств Stratix замість вбудованих помножувачів використовуються більш складні блоки цифрової обробки сигналу (ЦОС). Ос-

новним призначенням блока цифрової обробки сигналу є оптимізація алгоритмів ЦОС, які потребують високої пропускної спроможності. До таких алгоритмів відносяться цифрова фільтрація, швидке перетворення Фур'є, кодування даних. Типовими задачами, які потребують виконання таких операцій, є цифрове телебачення, IP-телефонія та телекомунікації. Для всіх цих задач використання вбудованих блоків ЦОС дозволяє значно підвищити пропускну спроможність алгоритму та зменшити ресурси, які необхідні для реалізації алгоритму.

Блок ЦОС складається з двох однакових частин, одна з яких показана на рисунку 8.12.

Блок містить помножувачі з програмувальною розрядністю, суматори та регістри для збереження проміжних результатів розрахунків. Помножувачі можуть працювати у декількох режимах – 9-, 12-, 18- та 36-розрядному.

При використанні такого блока при роботі помножувачів у 18-розрядному режимі на виході кожного суматора ∑ можна отримати такий вираз для операції MAC:

$$P0[36..0] = A0[17..0] \times B0[17..0] \pm A1[17..0] \times B1[17..0].$$

Крім помножувачів та суматорів блоки ЦОС містять кола для перенесення результатів з одного блока ЦОС до іншого, що значно прискорює обчислення у основних операціях ЦОС.

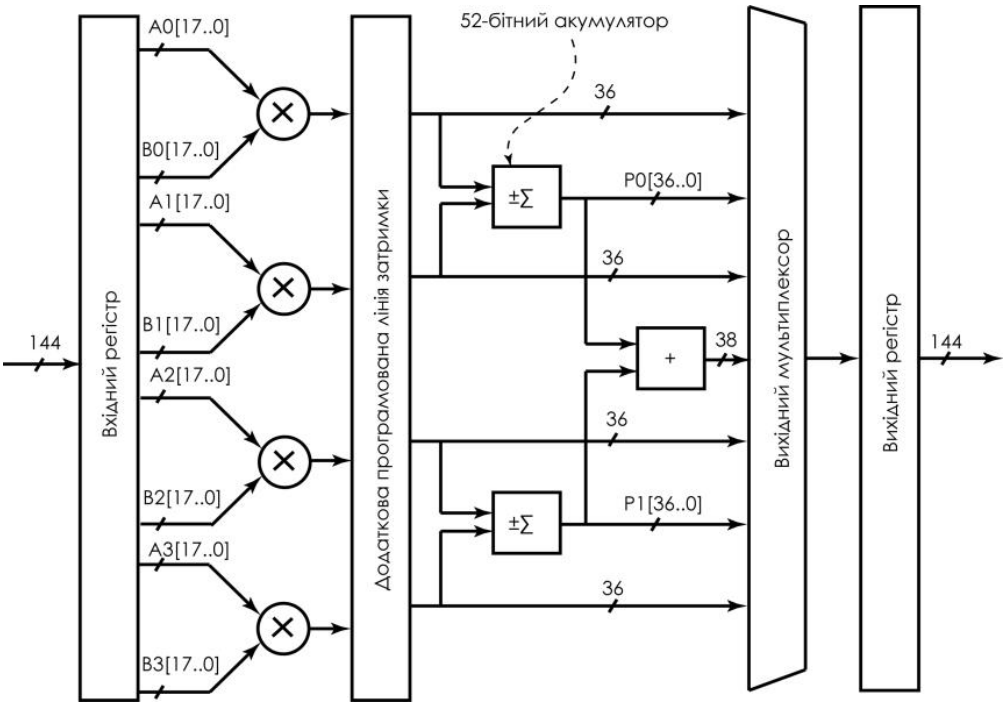


Рисунок 8.12 – Структурна схема половини блока ЦОС

8.2.6 Елементи введення/виведення і банки введення/виведення

Елемент введення/виведення мікросхем сімейства Cyclone II може працювати в одному з режимів, який задається при програмуванні мікросхеми: введення, виведення, двонаправлене виведення, виведення у високоімпедансному стані. Крім того, елемент введення/виведення забезпечує також підтримку різноманітних стандартів, увімкнення підтягувальних резисторів під час конфігурації мікросхеми, роботу кіл утримання шини, програмовану затримку сигналу для введення та виведення, підтримку необхідної сили струму для деяких стандартів, програмоване ввімкнення підтягувальних резисторів.

Елемент введення/виведення містить вихідний буфер, вхідний, вихідний регістри та регістр дозволу роботи, який переводить вихід мікросхеми в Z-стан, що забезпечують двонаправлений режим роботи, введення даних в ПЛІС та виведення даних з мікросхеми. Архітектуру елементів введення/виведення показано на рисунку 8.13.

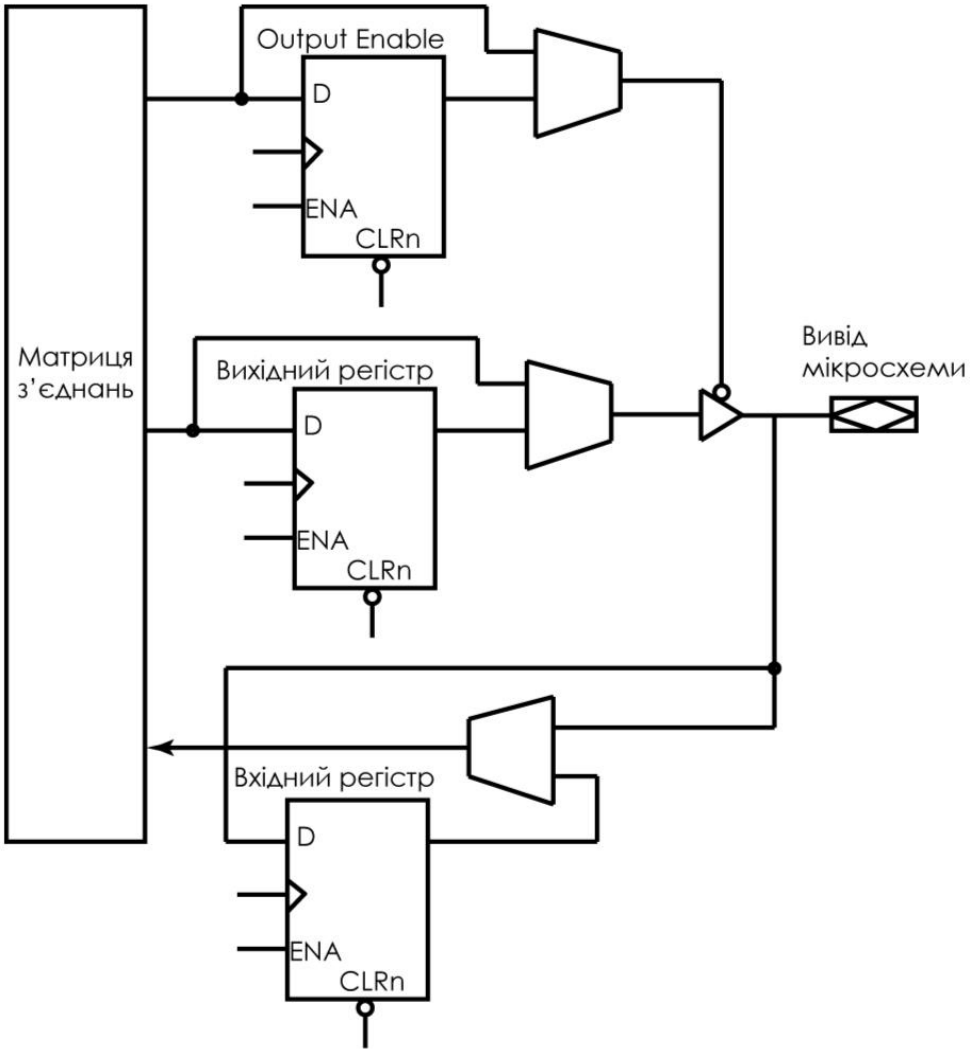


Рисунок 8.13 – Архітектура блока введення/виведення

Сучасні ПЛІС можуть працювати одночасно з кількома стандартами введення/виведення (рисунок 8.14). Це дає можливість використовувати ПЛІС в системах з різними стандартами введення/виведення. Так, наприклад, до мікросхеми ПЛІС можуть бути підключені сигнали з напругою живлення 2,5 В, 3,3 В, диференційні сигнали, сигнали різноманітних протоколів: PCI, PCI-X, DDR. Але потрібно звернути увагу на те, що ПЛІС не можуть забезпечити повноцінну роботу зі стандартними ТТЛ-рівнями. В більшості випадків мікросхеми працюють з сигналами до 3,3 В. Для роботи з сигналами ТТЛ та 5В КМОН необхідно використовувати перетворювачі рівня.

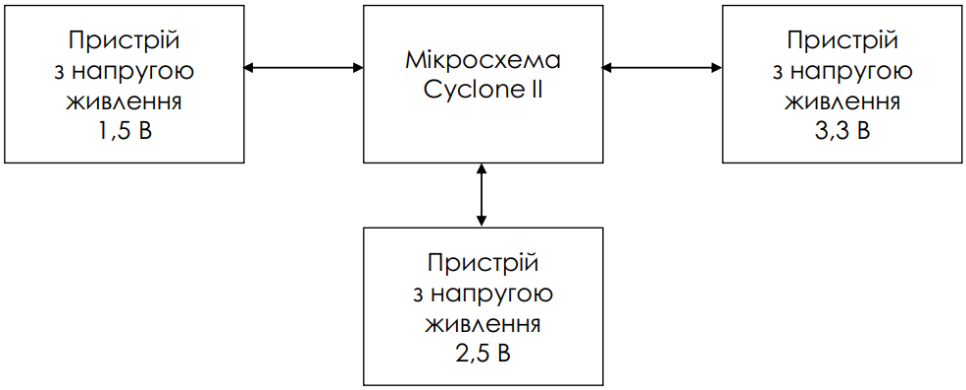


Рисунок 8.14 – Робота мікросхеми з різними стандартами введення/виведення

Забезпечення підтримки декількох стандартів введення/виведення досягається завдяки архітектурі ПЛІС та окремим лініям живлення для різних банків – об’єднання в групи елементів введення/виведення (рисунок 8.15).

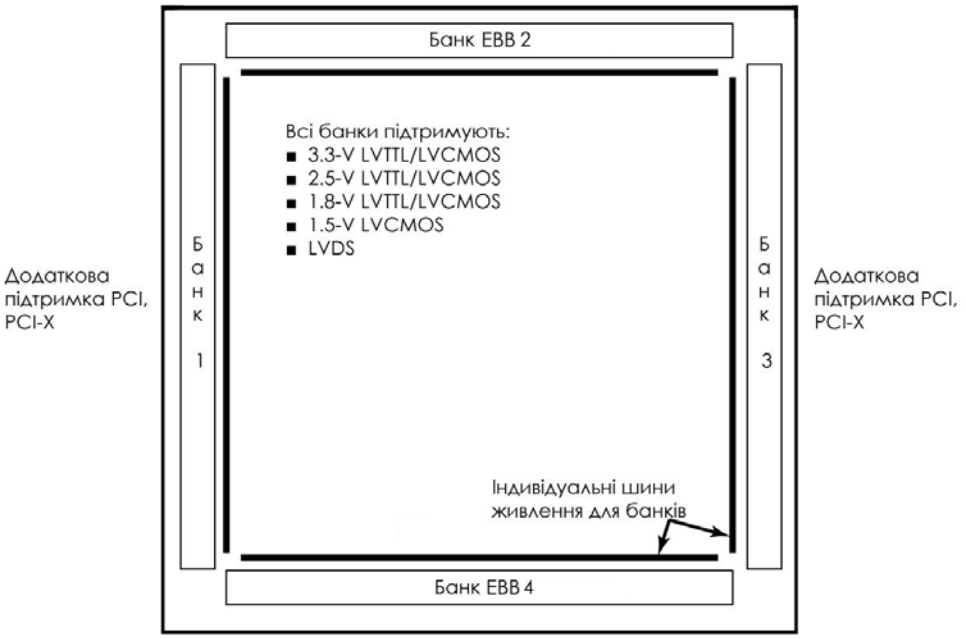


Рисунок 8.15 – Банки введення/виведення в мікросхемі ПЛІС

Особливістю банків є використання окремих ліній живлення для кожного з банків. Для живлення банків елементів введення/виведення може бути використано декілька рівнів напруг: 1,5 В; 1,8 В; 2,5 В; 3,3В. Тобто окрема лінія не може бути сконфігурована на інший стандарт, а лише весь банк в цілому, який містить кілька десятків виводів. Рівень напруги визначається стандартами введення/виведення, які будуть використані в елементах введення/виведення. В мікросхемах сімейства Cyclone II може бути 4 або 8 банків. Кількість банків залежить від обсягу мікросхеми та кількості виводів у корпусі (таблиця 8.2).

Таблиця 8.2 – Кількість ліній, доступних користувачу залежно від типу корпусу мікросхеми

Тип мікросхеми	Тип корпусу					
	TQFP, 100 виводів	TQFP, 144 виводи	PQFP, 240 виводів	BGA, 256 виводів	BGA, 324 виводи	BGA, 400 виводів
EP1C3	65	104	–	–	–	–
EP1C4	–	–	–	–	249	301
EP1C6	–	98	185	185	–	–
EP1C12	–	–	173	185	249	–
EP1C20	–	–	–	–	233	301

Використання різних типів корпусів та мікросхем різної логічної ємності дає можливість так званої вертикальної міграції, коли один і той самий корпус може використовуватись мікросхемами різної логічної ємності. При цьому розташування службових виводів та виводів живлення у мікросхем різної ємності буде однаковим. В цьому випадку можна використовувати ту ж саму друковану плату, замінивши лише мікросхему на більшу або меншу.

Крім ліній програмування та живлення мікросхеми ПЛІС мають також виводи для підключення тактових сигналів. Сигнали з цих виводів підключаються до ліній глобального поширення сигналу. Кількість спеціалізованих тактових сигналів залежить від типу корпусу мікросхеми і може дорівнювати 8 або 20. Лінії глобального поширення сигналів дозволяють прискорити проходження сигналів по мікросхемі порівняно з проходженням сигналу через стандартні лінії зв’язку. Це дозволяє досягти майже одночасного спрацювання тригерів, які розташовані в різних місцях мікросхеми.

Крім того, джерелом для глобальних сигналів можуть слугувати також виводи подвійного використання DPCLK (Dual-Purpose Clock), кількість яких дорівнює 8 або 16, виходи вбудованих блоків ФАПЧ (PLL) та внутрішня логіка проекту. Підключення сигналів до ліній глобального поширення може виконуватись або автоматично самим компілятором пакета Quartus II, або розробником за допомогою примітива Global як в графічному редакторі так і при використанні мови опису апаратури [17].

8.3 Архітектура мікросхем Spartan

Мікросхеми сімейств Spartan належать до так званих дешевих мікросхем. Архітектуру FPGA мікросхем компанії Xilinx розглянемо на прикладі мікросхеми Spartan 3. Основу мікросхеми складають конфігурувальні логічні блоки, які складаються з чотирьох логічних комірок, що організовуються у вигляді двох однакових секцій (Slice). Одна з таких секцій показана на рисунку 8.16.

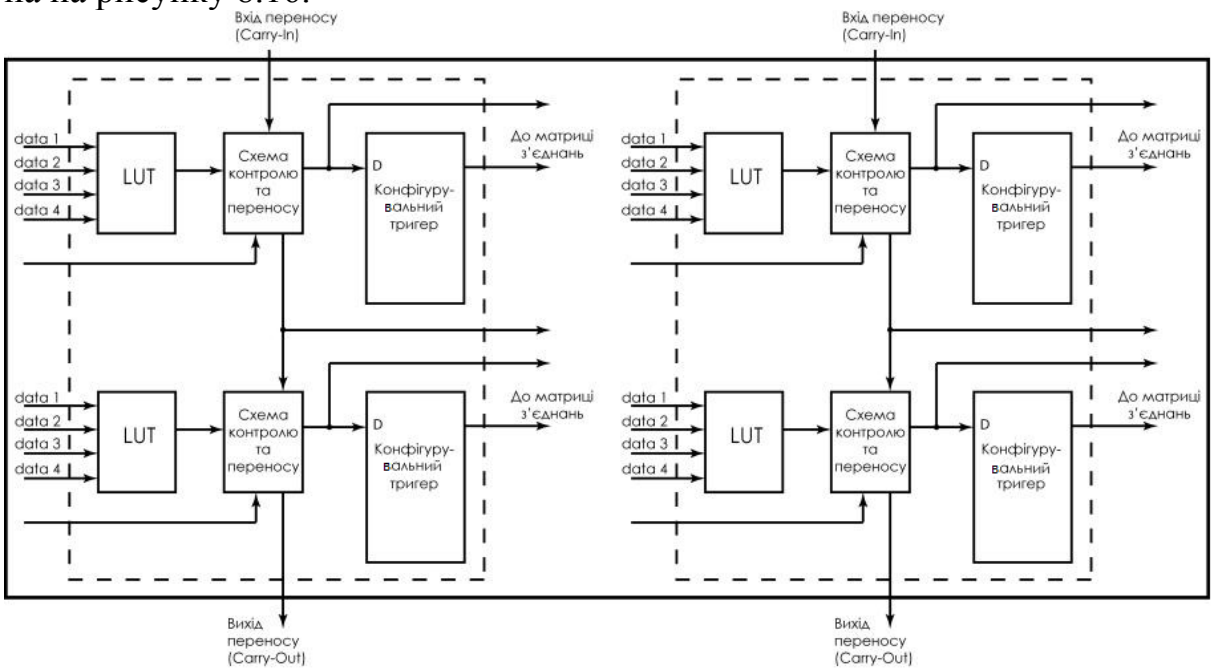


Рисунок 8.16 – Архітектура секції (Slice) мікросхеми Spartan 3

Комірка має в своєму складі таблицю перекодування (LUT) та конфігурувальний тригер. Тобто архітектура логічного елемента мікросхем сімейства Spartan дуже подібна до архітектури логічного елемента мікросхем сімейства Cyclone. Відмінність полягає в тому, що до складу логічного блока входять чотири комірки. Комірки згруповані в пари SliceM та SliceL, які мають спільні кола перенесення. Як і в мікросхемах Cyclone комірки можуть працювати в звичайному та арифметичному режимах [18].

Параметри найменшої та найбільшої з мікросхем сімейства Spartan 3 наведено в таблиці 8.3.

Таблиця 8.3 – Параметри мікросхем сімейства Spartan 3

Параметр	XC3S50	XC3S5000
Логічних елементів	1 728	74 880
Загальний обсяг пам’яті, кбіт	84	2 392
Блоків ФАПЧ	2	4
Вбудованих помножувачів	4	104
Виводів, доступних користувачу	124	784

8.4 Архітектура мікросхем CPLD

Архітектуру CPLD мікросхем розглянемо на прикладі сімейства MAX 3000 компанії Altera. На сьогоднішній день компанія Altera випускає два сімейства з класичною CPLD архітектурою: MAX 3000 та MAX 7000. В номенклатурі компанії Xilinx подібну архітектуру мають мікросхеми сімейств XC9500 та CoolRunner. Параметри мікросхем сімейства MAX 3000 наведено в таблиці 8.4. Часові параметри, що використовуються в таблиці, мають таке значення: t_{PD} – затримка проходження сигналу без тактування крізь мікросхему, t_{CO} – час між надходженням тактового сигналу до появи сигналу на виході мікросхеми, f_{CNT} – максимальна тактова частота для тригерів.

Таблиця 8.4 – Параметри мікросхем MAX 3000

Параметр	EPM3032A	EPM3064A	EPM3512A
Макрокомірка	32	64	512
Логічних блоків	2	4	32
t_{PD}, нс	4,5	4,5	7,5
t_{CO}, нс	3,0	3,1	4,7
f_{CNT}, МГц	227,3	222,2	116,3

Як видно з рисунка 8.17, основою мікросхеми MAX 3000 є макрокомірка, яка складається з матриці розподілу термів та конфігураційного тригера.

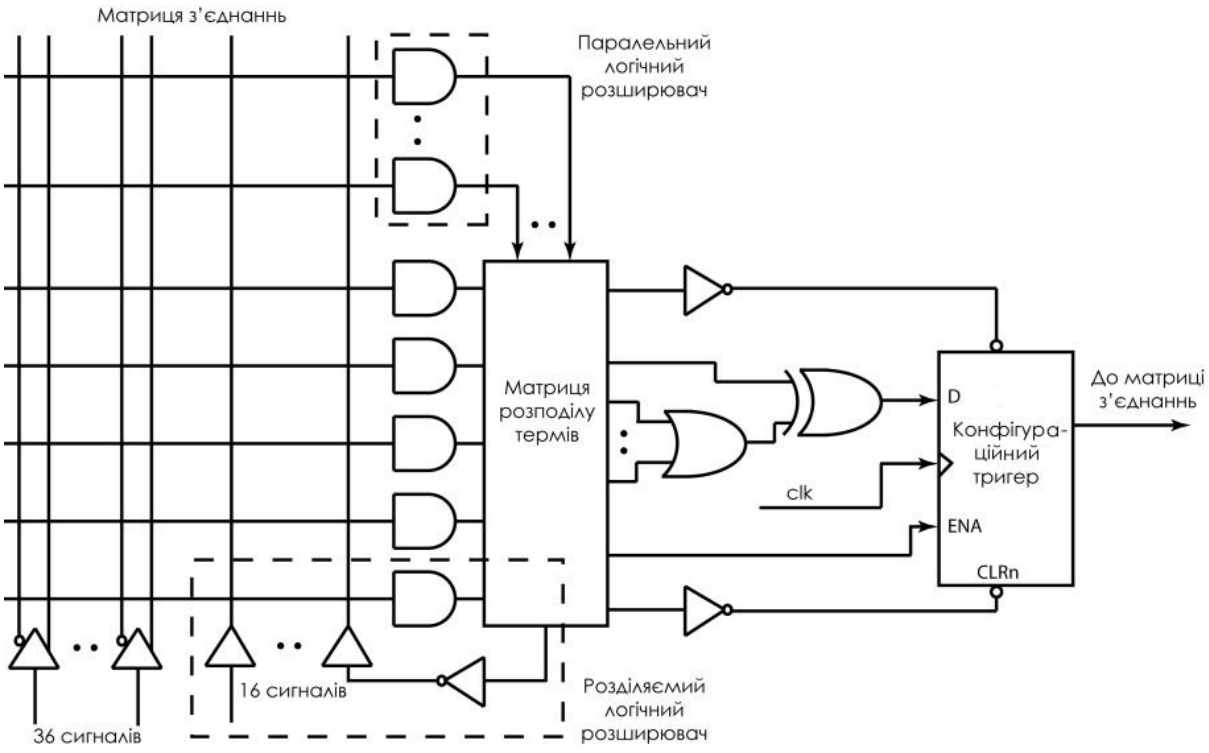


Рисунок 8.17 – Макрокомірка мікросхеми MAX 3000

Для реалізації логічної функції використовуються локальна програмна матриця з'єднань та матриця розподілу термів, які можуть об'єднувати за допомогою функції «АБО» або «Виняткове АБО» результати логічних добутків.

Для реалізації логічної функції на основі декількох макрокомірок одного логічного блока використовується логічний розширювач. Окремий логічний розширювач формує інверсію терма і подає його значення на локальну програмну матрицю, де він може використовуватись будь-якою макрокоміркою логічного блока. Паралельний логічний розширювач дозволяє використовувати терми з сусідніх макрокомірок для реалізації складних функцій, до складу яких входить більше 5 термів.

Як видно з рисунка 8.17, в матрицю з'єднань надходять 36 сигналів з інших логічних блоків та 16 сигналів з окремого паралельного логічного розширювача. Тобто на основі одного логічного блока можна реалізувати логічну функцію з 52 термів.

Макрокомірки (макрочарунки) об'єднуються в логічні блоки (рисунок 8.18) які, так само, з'єднані між собою та зовнішніми контактами ПЛІС за допомогою програмної матриці зв'язків (ПМЗ).

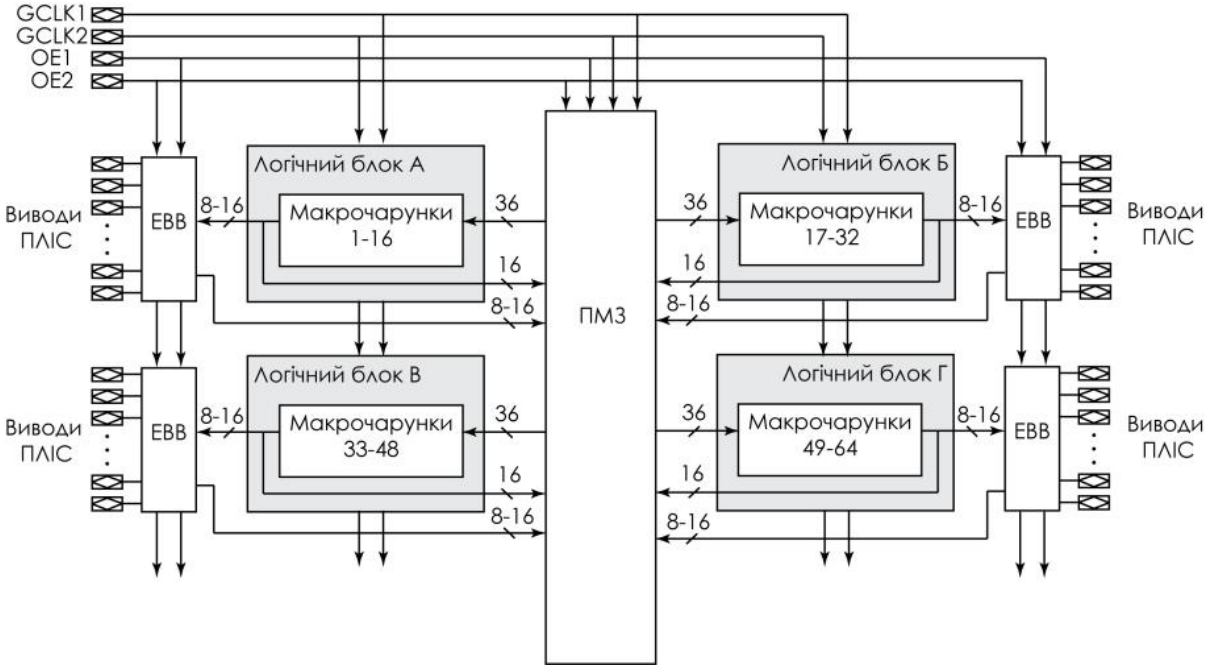


Рисунок 8.18 – Архітектура мікросхеми MAX 3000

Як і в мікросхемах типу FPGA MAX 3000 дозволяє використовувати глобальні сигнали. Але їх кількість значно менша – лише дві лінії тактування (GCLK1, GCLK2), сигнал скидання та дві лінії дозволу роботи (OE1, OE2).

Елементи введення/виведення мають архітектуру, подібну до архітектури цих блоків в мікросхемах Cyclone II [17].

8.5 Архітектура мікросхем MAX II

Мікросхеми сімейства MAX II займають проміжну ланку між класичними CPLD та FPGA мікросхемами. Як у всіх CPLD мікросхем, у MAX II конфігураційна інформація зберігається у внутрішній Flash пам'яті і для цих мікросхем зовнішній ПЗП для зберігання конфігурації не потрібен. Логічний елемент мікросхем MAX II подібний до логічних елементів FPGA мікросхем. Також в мікросхемах сімейства MAX II використовується архітектура програмованої матриці зв'язків, подібна до FPGA. Для забезпечення зв'язку між логічними блоками використовуються рядки і стовпці матриці зв'язків замість однієї загальної матриці. Це дозволяє значно зменшити розмір кристала при збільшенні кількості логічних блоків порівняно з класичними CPLD (рисунок 8.19).

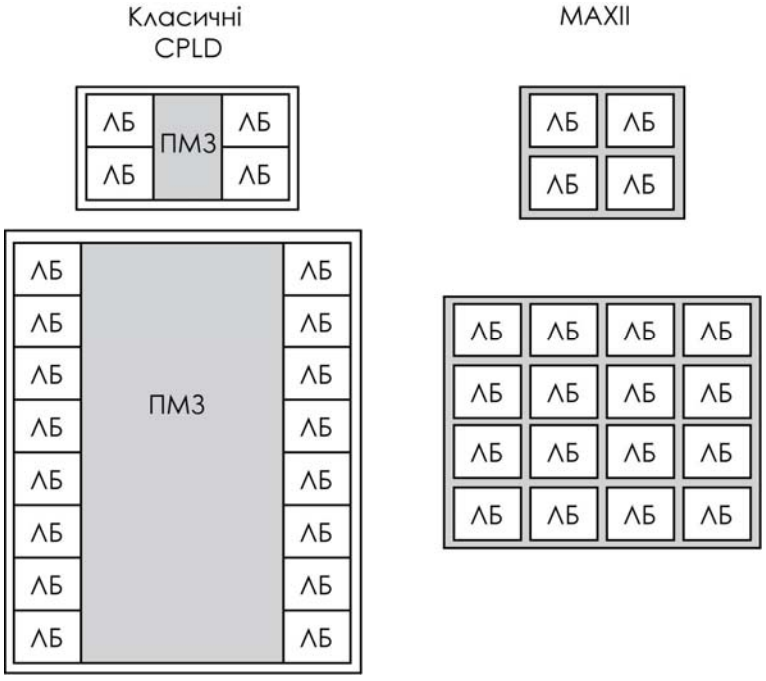


Рисунок 8.19 – Порівняння площі кристала для класичних CPLD та MAX II

Мікросхеми сімейства MAX II Z мають низьку споживану потожність у режимі STANDBY – 29 мкА (для мікросхеми EPM240Z).

Параметри мікросхем сімейства MAX II наведено в таблиці 8.5.

Таблиця 8.5 – Параметри мікросхем сімейства MAX II

Параметр	EPM240	EPM240G	EPM2210	EPM240Z
Логічних елементів	240	240	2210	240
t_{PD}, нс	4,7	4,7	7,0	7,5
t_{CO}, нс	4,3	4,3	4,6	6,5
f_{CNT}, МГц	304	304	304	152
$I_{CC\ STANDBY}$	12 мА	2 мА	12 мА	29 мкА

8.6 Конфігурування мікросхем ПЛІС

Для роботи ПЛІС необхідно завантажити в неї конфігурацію.

Конфігурація може зберігатись як в статичній пам'яті (SRAM – Flex10K, Cyclone, Stratix, Arria компанії Altera та Virtex і Spartan компанії Xilinx), так і в постійному запам'ятовувальному пристрої (EEPROM або Flash – MAX 3000, MAX 7000, MAX II компанії Altera та CoolRunner і XC9500 компанії Xilinx) [17, 18]. Для програмування та конфігурування мікросхем ПЛІС можуть використовуватись програматори, зовнішні ПЗП та завантажувальні кабелі. Завантажувальні кабелі можуть використовуватись при програмуванні мікросхеми, яка вже встановлена на друковану плату. Така технологія носить назву внутрішньосхемного програмування (In-System programming – ISP). При використанні зовнішніх ПЗП конфігураційна інформація зберігається в спеціалізованих мікросхемах і при включенні живлення завантажується до ПЛІС. Потрібно відзначити, що конфігураційні ПЗП відрізняються від звичайних ПЗП тим, що крім збереження інформації вони ще мають працювати з мікросхемою ПЛІС за стандартним протоколом конфігурування. Протокол залежить від режиму конфігурування. Для мікросхем ПЛІС можливо декілька режимів програмування:

- пасивний послідовний;
- асинхронний пасивний послідовний;
- пасивний паралельний;
- швидкий пасивний паралельний;
- програмування через JTAG-інтерфейс;
- активний послідовний.

Вибір режиму конфігурування визначається комбінацією на входах MSEL. Для різних сімейств мікросхем кількість входів може бути 2 чи 3. В таблиці 8.6 наведено режими конфігурування мікросхем сімейства Stratix.

Таблиця 8.6 – Режими конфігурування мікросхеми Stratix

Режим конфігурації	MSEL2	MSEL1	MSEL0
Швидкий паралельний пасивний (FPP)	0	0	0
Пасивний паралельний асинхронний (PPA)	0	0	1
Пасивний послідовний (PS)	0	1	0
Оновлення конфігурації (FPP)	1	0	0
Оновлення конфігурації (PPA)	1	0	1
Оновлення конфігурації (PS)	1	1	0
Програмування за допомогою JTAG інтерфейсу	X	X	X

На рисунку 8.20 наведено схему конфігурування мікросхем ПЛІС за допомогою послідовного конфігураційного пристрою.

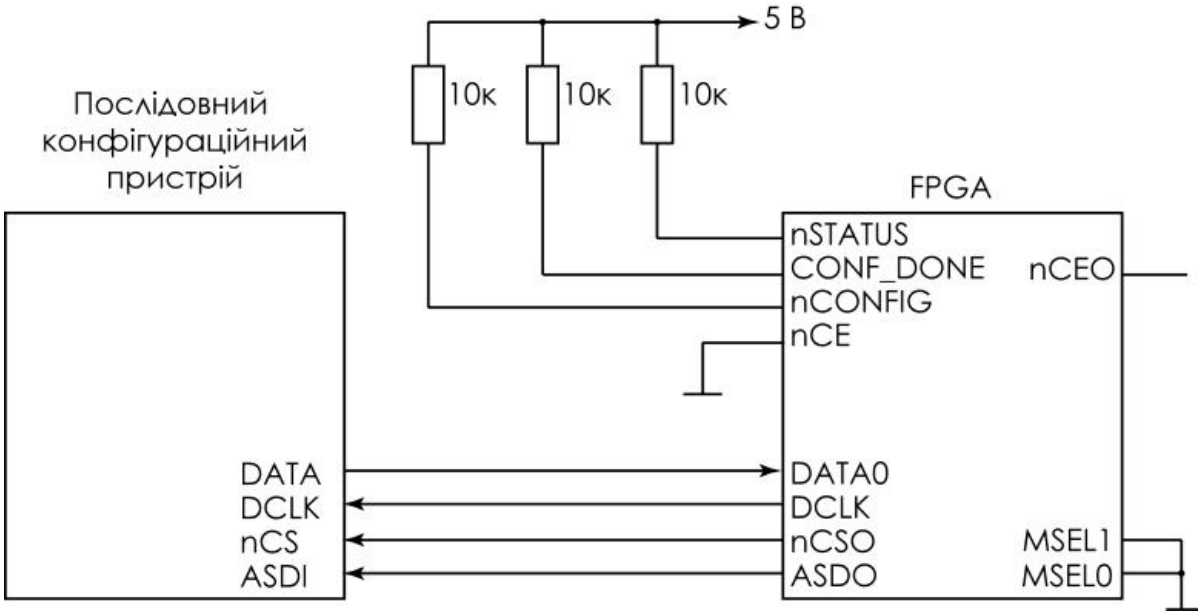


Рисунок 8.20 – Конфігурування за допомогою послідовного конфігураційного пристрою

В такому режимі процес конфігурування контролюється мікросхемою ПЗП. Тобто мікросхем ма ПЗП крім власне пам’яті також містить контролер конфігурації, який виробляє команди для конфігурування ПЛІС.

У тому випадку, коли необхідно провести конфігурування кількох мікросхем ПЛІС схема трохи змінюється і конфігурування мікросхем відбувається послідовно, одне за одним.

Використання активних конфігураційних пристроїв підвищувало вартість системи, тому при розробці мікросхем сімейства Cyclone було вирішено перенести контролер конфігурації до мікросхеми ПЛІС, а зовнішню пам’ять використовувати тільки для збереження конфігураційної інформації.

Конфігурування за допомогою завантажувального кабелю є основним режимом при проведенні експериментальних досліджень та лабораторних практикумів. Схема підключення завантажувального кабелю до ПЛІС наведена на рисунку 8.21.

В цьому випадку необхідно під’єднати кабель до паралельного інтерфейсу персонального комп’ютера, використовується завантажувальний кабель ByteBlaster II або ByteBlaster MV.

Сучасні мікросхеми ПЛІС також можуть програмуватися через порт порт USB за допомогою кабелю USB Blaster.

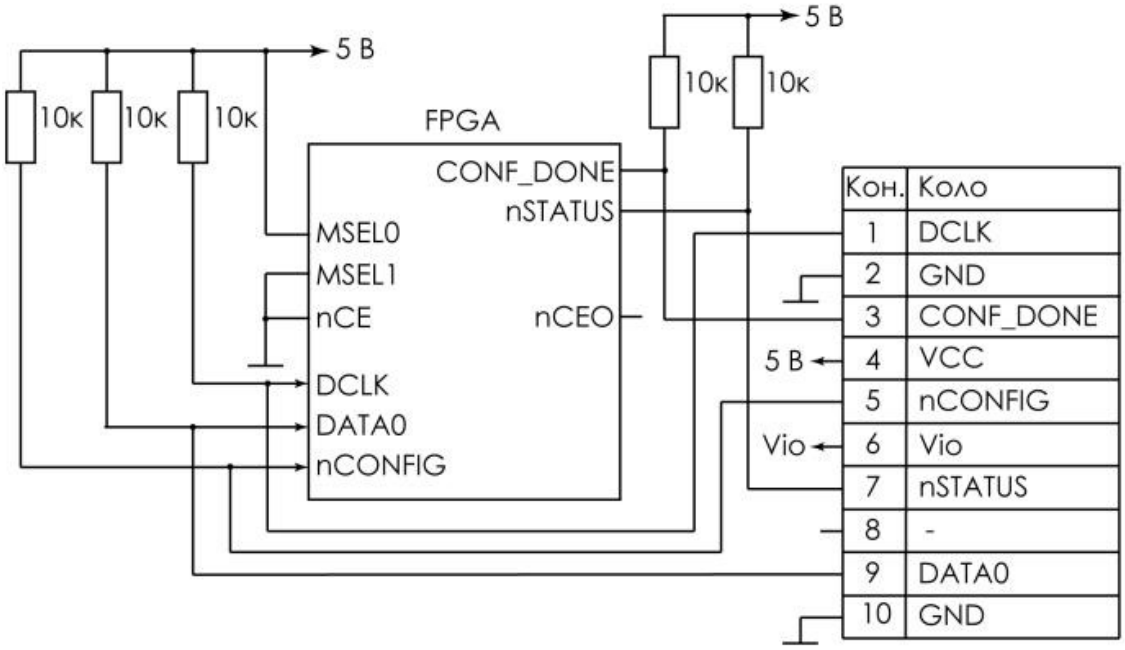


Рисунок 8.21 – Конфігурування ПЛІС за допомогою завантажувального кабелю

Якщо в системі викростовується ПЛІС великого обсягу, то необхідно використовувати і велику за обсягом конфігураційну пам’ять. Для зменшення обсягу конфігураційного файлу можна використати компресію конфігураційних даних, що зменшує обсяг конфігураційного файлу приблизно на 50%. Для декомпресії даних в ПЛІС необхідно встановити спеціальну схему, яка буде виконувати цю операцію.

Дуже часто виникає питання: а чи можливо, прочитавши конфігураційну інформацію, отримати схему проекту або файл мовою опису апаратури? Відповідь на це питання негативна. Можна при конфігуруванні пристрою прочитати бітовий потік, що йде з конфігураційного пристрою до ПЛІС, але це дозволить отримати лише конфігураційний файл, а не схему проекту. Конфігураційна інформація не є аналогом машинних кодів у програмі для комп’ютера або контролера. Тому навіть маючи конфігураційний файл неможливо з нього отримати схему проекту або програму мовою опису апаратури.

Для унеможливлювання зчитування інформації з мікросхеми ПЛІС в мікросхемах сімейств MAX 7000, MAX 3000, MAX II використовується біт секретності, який неможливо зняти. Для мікросхем сімейств Stratix використовується кодування даних за допомогою 128-розрядного ключа, який зберігається в спеціальній пам’яті самої мікросхеми, а з конфігураційного пристрою йде кодований потік даних, який декодується вже в самій мікросхемі [13, 14].

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Наведіть класифікаційні ознаки програмованих логічних інтегральних схем.
2. В чому особливість архітектури мікросхем сімейства Cyclone II?
3. Наведіть архітектуру логічного елемента.
4. Яке призначення логічного блока?
5. Які особливості конфігураційного блока пам’яті М4К?
6. Наведіть особливості архітектури блока ФАПЧ.
7. Поясніть архітектуру вбудованого помножувача і блока цифрової обробки сигналу.
8. Наведіть елементи введення/виведення і банки введення/виведення.
9. Які особливості архітектури мікросхем Spartan?
10. Поясніть архітектуру мікросхем CPLD.
11. Наведіть відмінні риси архітектури мікросхем MAX II.
12. В чому полягають особливості конфігурування мікросхем програмованих логічних інтегральних схем?

ГЛОСАРІЙ

Аналого-цифровий перетворювач – analog-to-digital converter.
Арифметико-логічний пристрій – arithmetic and logical unit.
Біт – bit.
Гарвардська архітектура – harvard architecture.
Двійкова система – binary number system.
Двійково-десяткова система – binary-decimal system.
Запам’ятовувальний пристрій – memory device.
Зрушувач – shifter.
Мантиса – mantissa.
Мікроконтролер – microcontroller.
Мікропроцесор – microprocessor.
Мікропроцесорна система – microprocessor system.
Мікропроцесорна техніка – microprocessor technology.
Оперативна пам’ять – random-access memory.
Пам’ять – memory.
Помножувач-акумулятор – multiply-accumulate operation.
Постійна пам’ять – read-only memory .
Прапорець додаткового перенесення – auxilinary carry flag.
Прапорець дозволу переривання – interrupt flag.
Прапорець знака результату – sign flag.
Прапорець нуля – zero flag.
Прапорець парності – parity flag.
Прапорець перенесення – carry flag.
Прапорець переповнення – owerfow flag.
Програмована логічна інтегральна схема –programmable logic integrated circuit.
Програмована логічна матриця – programmable logical matrix
Прямий доступ до пам’яті – direct memory access.
Фон-нейманівська архітектура – von Neumann architecture.
Формувач послідовності команд – form builder of teams.
Цифро-аналоговий перетворювач – digital-to-analog converter.
Шина адреси – address bus.
Шина даних – data bus.
Шина управління – control bus.

ТЕСТ ДЛЯ САМОКОНТРОЛЮ

1. Який пристрій призначений для обробки цифрової інформації та управління процесом цієї обробки?
   а – мікроконтролер;
   б – мікропроцесор;
   в – сигнальний мікропроцесор;
   г – програмована логічна інтегральна система.
2. Число 101011 в десятковій системі числення відповідає:
   а – 41;
   б – 39;
   в – 43;
   г – 45.
3. Які внутрішні регістри мікропроцесора використовуються для зберігання як даних, так і адресної інформації?
   а – регістри даних;
   б – покажчики;
   в – регістри загального призначення;
   г – регістри адрес.
4. Який діапазон хвиль використовується для стирання інформації в РПЗП?
   а – ультрафіолетовий;
   б – інфрачервоний;
   в – видимий;
   г – весь діапазон.
5. Знайти значення фізичної адреси за логічною адресою B002:317A.
   а – B002A;
   б – B317A;
   в – B219A;
   г – B319A.
6. Які регістри використовують як порти введення/виведення?
   а – 8-розрядні;
   б – 16-розрядні;
   в – 8-, 16-, 32-розрядні;
   г – 8- та 16-розрядні.
7. Які логічні функції виконує арифметико-логічний пристрій?
   а – OR, XOR;
   б – AND, OR;
   в – AND, OR, XOR;
   г – AND, OR, XOR, логічне заперечення.
8. Чи можна за конфігураційною інформацією отримати схему проекту?
   а – так;
   б – ні;
   в – можна отримати схему, не зчитуючи конфігураційну інформацію;
   г – можна отримати файл мовою опису апаратури.

СПИСОК ЛІТЕРАТУРИ

1. Мікропроцесорна техніка / [Якименко Ю. І., Терещенко Т. О., Сокол Є. І.та ін.]. – К. : Політехнік, 2003. – 440 с.
2. Кушков В. М. Мікропроцесорна техніка: Курс лекцій для студ. напряму 6.050202 «Автоматизація та комп'ютерно-інтегровані технології» ден. та заоч. форм навч. / Кушков В. М. – К. : НУХТ, 2011. – 148 с.
3. Бродин В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин, А. В. Калинин. – М. : ЭКОМ, 2002. – 400 с.
4. Симаков Г. М. Цифровые устройства и микропроцессоры в автоматизированном электроприводе / Г. М. Симаков, Ю. В. Панкрац – Новосибирск : НГТУ, 2013. – 210 с.
5. Корабельников Е. А. Самоучитель по программированию PIC микроконтроллеров / Корабельников Е. А. – СПб. : Питер, 2008. – 343 с.
6. Яценкоп В. С. Микроконтроллеры Microchip с аппаратной поддержкой USB / Яценкоп В. С. – СПб. : Питер, 2008. – 340 с.
7. Кенинг А. Полное руководство по PIC микроконтроллерам / Кенинг А. – СПб. : Питер, 2007. –430 с.
8. Тавернье К. PIC микроконтроллеры. Практика применения / Тавернье К. – СПб. : Питер, 2006. – 360 с.
9. Семенов Б. Ю. Микроконтроллеры MSP430 / Семенов Б. Ю. – СПб. : Питер, 2006. – 480 с.
10. Ревич Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке Ассемблера / Ревич Ю. В. – СПб. : Питер, 2008. – 356 с.
11. Трамперт В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров / Трамперт В. – СПб. : Питер, 2006. – 343 с.
12. Парр Э. Программируемые контроллеры: руководство для инженера / Парр Э. – СПб. : Питер, 2008. – 300 с.
13. Кузяков О. Н. Проектирование систем на микропроцессорах и микроконтроллерах / Кузяков О. Н. – Тюмень : ТюмГНГУ, 2014. – 104 с.
14. Русанов В. В. Микропроцессорные устройства и системы / Русанов В. В., Шевелёв М. Ю. – Томск : ТУСУР, 2012. – 184 с.
15. Грушвицкий Р. И. Проектирование систем на микросхемах программируемой логики / Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П. – СПб. : БХВ-Петербург, 2002. – 608 с.
16. Максфилд К. Проектирование на ПЛИС. Курс молодого бойца / Максфилд К. – М. : Додэка-XXI, 2007. – 408 с.
17. Стешенко В.Б. ПЛИС фирмы «ALTERA»: элементная база, система проектирования и языки описания аппаратуры / Стешенко В. Б. – М. : Додэка-XXI, 2002. – 576 с.
18. Кузелин М. О. Современные семейства ПЛИС фирмы Xilinx / Кузелин М. О., Кнышев Д. А., Зотов В. Ю. – М. : Горячая линия-Телеком, 2004. – 440 с.

Навчальне видання

Огородник Костянтин Володимирович
Книш Богдан Петрович

МІКРОПРОЦЕСОРНА ТЕХНІКА

Навчальний посібник

Рукопис оформлено Б. Книшом

Редактор Т. Старічек

Оригінал-макет виготовлено О. Ткачуком.

Підписано до друку 18.06.2018
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 6,36.
Наклад 50 (1-й запуск 1–20) пр. Зам. № 2018-122.

Видавець та виготовлювач
Вінницький національний технічний університет,
інформаційний редакційно-видавничий центр.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
E-mail: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.