Лекція 5. Арифметико-логічні пристрої та пристрої управління

5.1 Класифікація арифметико-логічних пристроїв

Арифметико-логічний пристрій (АЛП) функціонує на основі мікропрограмного управління. Кожна машинна операція розділяється на послідовність елементарних дій (передача слів, інверсія слів і ін.), які реалізовуються в тактах. Елементарне функціональне обчислення, яке виконується в одному машинному такті, називається мікрооперацією. Кожна мікрооперація ініціюється відповідним управляючим сигналом. Сукупність мікроопераций, які виконуються в одному такті, називається мікрокомандою. Зокрема, мікрокоманда може містити одну мікрооперацію або жодній.

Для вибору порядку проходження мікроопераций аналізуються логічні умови, які набувають значення одиниці (так) або нуля (не) залежно від значень операндів і результатів обчислень. Мікроалгоритм операції, записаний в термінах мікроопераций і логічних умов, називається мікропрограмою. Кожна машинна операція має свою мікропрограму.

Будь-який цифровий обчислювач, у тому числі і АЛП, може бути представлений композицією операційних управляючих пристроїв. У операційному пристрої виконуються арифметико-логічні операції. Управляючий пристрій забезпечує виконання операцій за допомогою послідовності управляючих сигналів, яку він виробляє залежно від мікропрограми. У математичних моделях АЛП перший пристрій представлений операційним автоматом, а другий – управляючим автоматом (рис. 8.1).

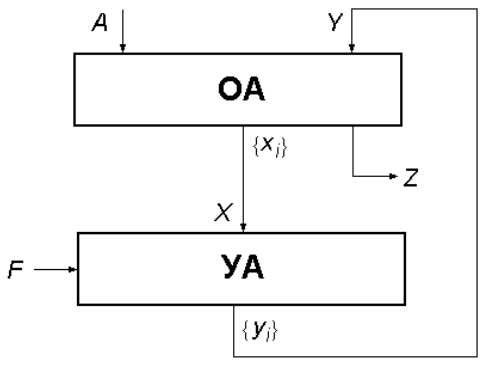


Рисунок 5.1 – Структура математичної моделі АЛП

Операційний автомат (ОА) приймає по входу А операнди, по входу Y – управляючі сигнали {yi}, передає на вихід Z результати операції і формує безліч значень логічних умов {xi}.

Управляючий автомат (УА приймає по входу X логічні умови {xi} і залежно від їх значень і коду операції по входу F формує послідовність управляючих сигналів.

Арифметико-логічні пристрої класифікують по наступних ознаках:

- способу обробки даних – паралельні, послідовні, паралельно-послідовні;

- системі числення – двійкові, вісімкові, десяткові, шістнадцятирічні, а також пристрої на основі спеціальних систем (залишкових класів, з штучним порядком ваги, чисел Фібоначчі) і др.;

- формі представлення чисел – з плаваючою комою, з фіксованою комою, цілі двійкові і десяткові числа;

- часу виконання операцій – синхронні і асинхронні;

- способу виконання мікрооперацій – із закріпленими мікрооперaціями, із загальними операціями;

- типу управляючого автомата – з схемною або програмованою логікою;

- методу побудови – багатофункціональні і блокові.

У синхронних АЛП на виконання різних операцій відводиться один і той же інтервал часу, а в асинхронних час виконання залежить від типа операції.

У АЛП із закріпленими мікрооперaціями кожен з регістрів за допомогою додаткових комбінаційних схем виконує певний набір мікрооперацій. При цьому комбінаційні схеми часто повторюються, що вимагає значних апаратних витрат.

У АЛП із загальними мікрооперaціями виділяють частину яка запам'ятовує – блок регістрів, в яких виконуються однорідні мікрооперації (прийом операндів, їх зберігання і видачу), і комбінаційну частину, в якій зосереджені

всі схеми для виконання мікрооперацій (формування коду, зрушення, складання і ін.). Обидві частини з'єднуються між собою за допомогою мультиплексорів і демультиплексорів. Арифметико-логічні пристрої із загальними мікрооперaціями часто називають магістральними (до числових магістралей по черзі підключаються регістри).

Багатофункціональні (універсальні) АЛП використовуються для виконання всього списку операцій, що досягається відповідним налаштуванням і комутацією вузлів. Блокові АЛП складаються з окремих блоків, орієнтованих на виконання окремих типів операцій (наприклад, блок множення чисел з плаваючою комою). Такі структури використовуються у високопродуктивних комп'ютерах.

У АЛП можливі два типи УА:

- з схемною (жорсткою) логікою, яка складається з елементів пам'яті (тригерів) і комбінаційних схем. Вони генерують відповідні управляючі сигнали $\{y_i\}$ в машинні такти залежно від коду операції;

- з програмованою (яка зберігається в пам'яті) логікою: для кожної операції в спеціальній пам'яті (найчастіше – різні види ПЗП) записується мікропрограма у вигляді послідовностей управляючих слів – мікрокоманд. Вони містять інформацію про мікрооперації, які повинні виконуватися в даному такті, і адреса наступної мікрокоманди.

Узагальнена і найбільш поширена структура АЛП показана на рисунку 5.2.

До складу ОА універсальних комп'ютерів входять:

- арифметико-логічний блок (АЛБ);

- набір регістрів загального призначення (РЗП);

- блок контролю.

У АЛБ виділяють комбінаційний суматор SM, вхідні регістри A і B для прийому операндів і вихідний регістр C для запису результату. У АЛБ, є логічні схеми, які виробляють безліч $\{x_i\}$ сигналів логічних умов (ознак результату), наприклад, нульовий або негативний результат і ін.

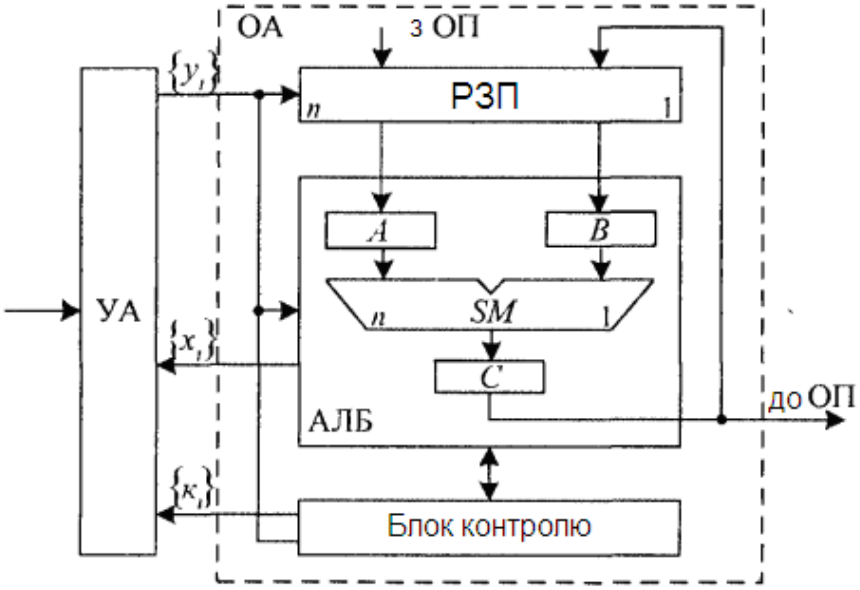


Рисунок 5.2 – Узагальнена структура АЛП

Регістри загального призначення використовують для прийому і зберігання операндів, проміжних і кінцевих результатів. Блок контролю забезпечує перевірку правильності виконання арифметико-логічних операцій, одночасною реалізацією тієї ж команди, дублюючою апаратурою і порівнянням результатів, або шляхом виконання дій над спеціальними кодами, отриманими від операндів при складанні по модулю два, три і ін.

При виявленні помилок і збоїв в роботі ОА блок контролю посилає в УА код помилок $\{k_i\}$.

У АЛП поступає код операції від центрального пристрою управління. Використання в АЛП пристроїв управління з схемною логікою прискорює виконання операцій. Використання УА з програмованою логікою забезпечує гнучкість мікропрограмування, дозволяє змінювати склад мікропрограм при введенні нових команд. У сучасних АЛП можуть об'єднуватися обидва типа УА.

Для опису операційних пристроїв на різних рівнях використовуються відповідні мови, а саме:

- мова електричних рівнянь для струмів і напруги в колах схем, сякі складвються з резисторів, діодів, транзисторів і т.д.;

- мова булевих функцій для опису логічних запам'ятовувальних елеме-

нтів;

- мова мікрооперацій для опису типових функціональних комбінаційних і послідовністних вузлів;

- мова мікропрограм для опису роботи операційних пристроїв на рівні мікроалгоритмів машинних операцій;

- алгоритмічні мови для опису обчислювального процесу в комп'ютері на рівні програм (Асемблер, Паскаль, Сі та ін.).

Термін “автомат” використовують двояко. У техніці з поняттям “автомат” зв'язують деякий пристрій, здатний виконувати певні функції без втручання людини або з його обмеженою участю. У іншому, широкому аспекті, автомат – це математична модель, яка відображує фізичні або абстрактні явища найрізноманітнішої природи (обчислювальні машини, системи управління і зв'язку, лінгвістика і ін.). Універсальність теорії автоматів дозволяє розглядати з єдиної точки зору різні об'єкти, встановлювати зв'язки і аналогії між ними, переносити результати досліджень з однієї області в іншу. Узагальненим прикладом цифрового автомата є комп'ютер, який виконує прийом, зберігання і перетворення дискретної інформації по заданих алгоритмах.

Загальна теорія автоматів підрозділяється на абстрактну і структурну. Абстрактна теорія вивчає поведінку автомата відносно зовнішнього середовища і не розглядає способи його побудови. Структурна теорія автоматів вивчає способи побудови логічних схем автоматів на основі алгоритму, заданого на абстрактному рівні.

5.2 Арифметичні суматори оперативного блоку АЛП

Арифметичні суматори призначені для арифметичного цифрового підсумовування, в загальному випадку, багаторозрядних чисел.

По внутрішній структурі суматори діляться на комбінаційних (що не містять елементів пам'яті) і нагромаджуючих (суматори з пам'яттю по модулю 2).

Суматори діляться по довжині оброблюваних слів (рис.5.3).

Рисунок 5.3 – Класифікація суматорів по довжині оброблюваних слів

По використовуваній системі числення - на двійкові, двійково-десяткові і які використовують інші системи числення.

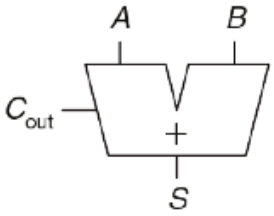
5.2.1 Арифметичні операції оперативної частини мікропроцесора

Арифметичні схеми є основним функціональним вузлом будь-якого комп'ютера. Комп'ютери і цифрові схеми виконують безліч арифметичних операцій: складання, віднімання, порівняння, зрушення, множення і ділення.

Складання - одна з найпоширеніших операцій в цифрових системах. Розглянемо складання двох одноразрядних двійкових чисел і потім розширимо цю процедуру до *N*-розрядних чисел. Суматори демонструють компроміс між швидкістю і складністю реалізації.

Спроектуємо одноразрядний напівсуматор (*half adder*). Як показано на рисунку 5.4 напівсуматор має два входи (A і B) і два виходи (*S* і C_{out}).

S - це сума *A* і *B*. Якщо і *A*, і *B* рівні 1, то вихід *S* повинен стати рівним 2, таке число не може бути представлене у вигляді одного двійкового розряду. В цьому випадку результат вказується разом з перенесенням C_{out} в наступний розряд. Напівсуматор може бути побудований з елементів XOR («виключне АБО») і AND («логічне І»).



A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$
$$C_{out} = AB$$

Рисунок 5.4 – Однорозрядний напівсуматор

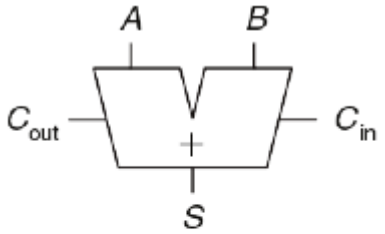
У багаторозрядному суматорі вихід C_{out} під'єднується до входу перенесення наступного розряду. Наприклад, на рисунку 5.5 біт перенесення показаний синім кольором, він є виходом C_{out} однорозрядного суматора 1-го розряду і входом C_{in} суматора наступного розряду.

```
  1
 0001
+0101
 0110
```

Рисунок 5.5 – Біт перенесення

Проте в напівсуматорі немає входу перенесення C_{in} для зв'язку з виходом C_{out} попереднього розряду. Як показано на рисунку 5.6 повний суматор (*full adder*) має вхід перенесення C_{in}. На рисунку також приведені рівняння для S і C_{out}.

N-розрядний суматор складає 2 N-розрядних числа (A і B), а також вхідне перенесення C_{in}, і формує N-розрядний результат S і вихідне перенесення C_{out}. Такий суматор називається суматором з перенесенням яке поширюється (*carry propagate adder, CPA*), оскільки вихідне перенесення одного розряду переходить в наступний розряд. Умовне позначення такого суматора показане на рисунку 5.7.



C_{in}	A	B	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

Рисунок 5.6 – Однорозрядний повний суматор

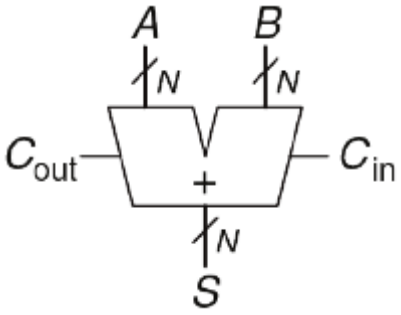


Рисунок 5.7 – Суматор з перенесенням, яке поширюється

Воно аналогічно позначенню повного суматора за винятком того, що входи/виходи *A*, *B*, *S* є шинами, а не окремими розрядами. Найпоширенішими реалізаціями *CPA* є: суматори з послідовним перенесенням (*ripple-carry adders*), з прискореним перенесенням (*carry-lookahead adders*) і префіксні суматори (*prefix adders*).

Найпростіший спосіб реалізації *N* розрядного суматора - це об'єднання в коло *N* повних суматорів. Вихід C_{out} деякого розряду поступатиме на вхід C_{in} наступного розряду і так далі (рис. 5.8).

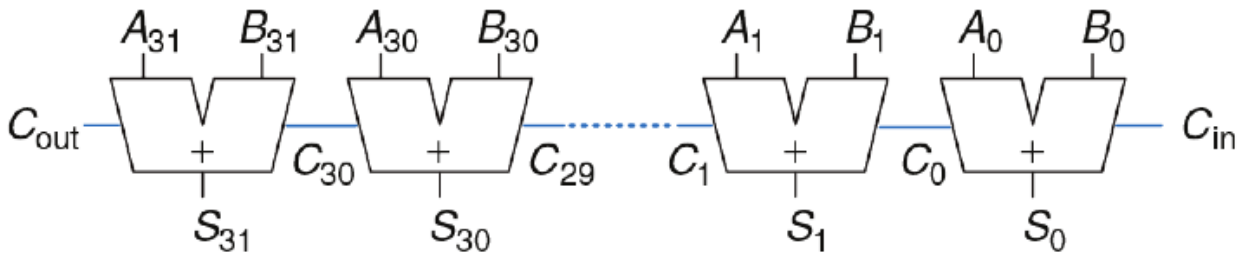


Рисунок 5.8 – 32- розрядний суматор з послідовним перенесенням

Така схема називається суматором з послідовним перенесенням (*ripple-carry adder*). При її проектуванні використовується принцип модульності і регулярності: модуль повного суматора багато разів використовується для формування більшої системи. Такий суматор має недолік: його швидкість падає при збільшенні числа розрядів *N*. S_{31} залежить від C_{30}, який залежить від C_{29}, який у свою чергу залежить від C_{28} і так далі до C_{in} (рис. 5.5). Перенесення проходить через все коло. Затримка такого суматора (*triple*) збільшується разом з кількістю розрядів, як показано в рівнянні (5.1), де t_{FA} - це затримка повного суматора.

$$t_{\text{ripple}} = Nt_{FA} \qquad (5.1)$$

Головною причиною того, що великі суматори з послідовним перенесенням працюють повільно, є те, що сигнал перенесення повинен пройти через всі біти суматора.

Суматори з прискореним перенесенням (*carry-lookahead adder, CLA*) - це інший тип суматорів з перенесенням, яке поширюється, який вирішує цю проблему шляхом розділення суматора на блоки і реалізовуючи схему так, щоб визначити вихідне перенесення блоку як тільки стало відомо його вхідне перенесення. Таким чином, ми дивимося вперед через блоки і не чекаємо проходження перенесення через всі повні суматори усередині блоку. Наприклад, 32-розрядний суматор може бути роздільний на 8 4-розрядних суматорів.

Суматори з прискореним перенесенням використовують сигнали генерації (*G*) і поширення (*P*), які описують, як блок (або розряд) визначає вихід перенесення. *i*-розряд суматора генерує перенесення, якщо він видає перене-

сення на своєму виході, незалежно від наявності перенесення на вході. *i*-розряд суматора генерує C_i в тому випадку, якщо і A_i, і B_i дорівнюють 1. Таким чином, сигнал генерації G_i можна обчислити як $G_i = A_iB_i$. Розряд називається поширюючим, якщо вихідний сигнал перенесення з'являється за наявності вхідного перенесення. Розряд поширюватиме вхідний сигнал перенесення C_{i-1}, якщо або A_i, або B_i дорівнюють 1. Таким чином, $P_i = A_i + B_i$. Використовуючи ці визначення, ми можемо переписати логіку формування сигналу перенесення для певного розряду. Розряд суматора формуватиме вихідний сигнал перенесення C_i, якщо він або генерує перенесення G_i або поширює вхідне перенесення P_iC_{i-1}. У вигляді рівняння це можна записати таким чином:

$$C_i = A_i B_i + (A_i + B_i) C_{i-1} = G_i + P_i C_{i-1} \qquad (5.2)$$

Визначення сигналів генерації і поширення відносяться і до багаторозрядних блоків. Блок називається генеруючим перенесення, якщо він створює вихідне перенесення незалежно від вхідного сигналу перенесення даного блоку. Блок називається поширюючим перенесення, якщо вихідне перенесення виникає під час вступу вхідного перенесення. $G_{i:j}$ і $P_{i:j}$ визначаються, як сигнали генерації і поширення для блоків, які охоплюють розряди з *i* до *j*.

Зазвичай в електронних схемах сигнали поширюються зліва направо. Арифметичні схеми порушують ці правила, оскільки перенесення йде справа наліво (від молодшого розряду до старшого).

Блок генерує перенесення, якщо самий старший розряд генерує перенесення або якщо старший розряд поширює перенесення, яке згенеровано попереднім розрядом і так далі

Наприклад, логіка блоку генерації для блоку, який охоплює розряди від 0 до 3, буде наступною:

$$G_{3:0} = G_3 + P_3 (G_2 + P_2 (G_1 + P_1 G_0)) \qquad (5.3)$$

Блок поширює перенесення, якщо всі розряди які до нього входять це перенесення поширюють. Логіка поширення для блоку, який охоплює розряди з 0 до 3:

$$P_{3:0} = P_3\ P_2\ P_1\ P_0 \qquad (5.4)$$

За допомогою блокових сигналів генерації і поширення можна швидко визначити вихідне перенесення блоку C_i, використовуючи його вхідне перенесення C_j.

$$C_i = G_{i:j} + P_{i:j}\ C_j \qquad (5.5)$$

На рисунку 5.9,а представлено 32-розрядний суматор з прискореним перенесенням, який складається з 8 4-розрядних блоків. Кожен блок містить 4-розрядний суматор з послідовним перенесенням і схему прискореного перенесення, яка визначає вихідне перенесення блоку по вхідному (рис. 5.9, б). На рисунку не показані елементи «І» і «АБО» необхідні для обчислення однорозрядних сигналів генерації і поширення G_i і P_i, по A_i, і B_i. Суматор з прискореним перенесенням демонструє модульність і регулярність.

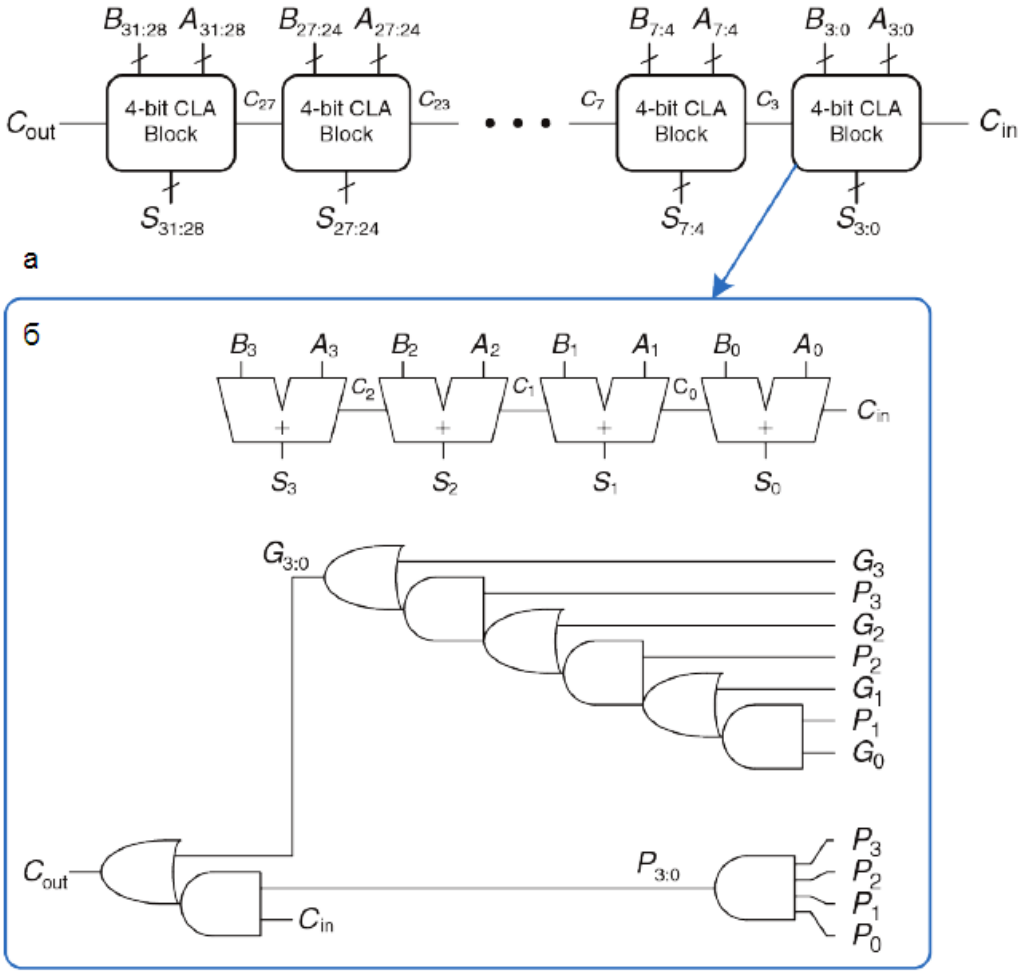


Рисунок 5.9 – 32-розрядний суматор з прискореним перенесенням (а); чотирьох-бітний блок (б)

Усі блоки суматора одночасно обчислюють однобітові і блокові сигнали генерації і поширення. Критичний шлях починається з обчислення G_0 і $G_{3:0}$ в першому блоці суматора. Сигнал C_{in} потім поширюється у напрямку до C_{out} через логічні елементи І/АБО всіх блоків. Для великого суматора це відбувається набагато швидше, ніж поширення перенесення через кожен подальший розряд суматора. Критичний шлях через останній блок містить невеликий суматор з послідовним перенесенням. Таким чином *N*-розрядний суматор, розділений на *k*- розрядні блоки, має затримку.

При $N > 16$ такий суматор працює набагато швидше, ніж суматор з послідовним перенесенням. Проте затримка суматора як і раніше лінійно зростає із зростанням *N*.

Префіксний суматор розвиває логіку генерації і поширення суматора з прискореним перенесенням для ще швидшого виконання операції складання. Спочатку він обчислює *G* і *P* для пар розрядів, далі для блоків з 4-х розрядів, потім для блоків з 8-мі, 16-ти і так далі розрядів, поки сигнал генерації не буде відомий для кожного розряду. Сума визначається всіма сигналами генерації.

Стратегія префіксного суматора полягає в обчисленні вхідного сигналу перенесення C_{i-1} для кожного розряду так швидко, наскільки це можливо. Потім за формулою обчислюється сума:

$$S_i = (A_i \oplus B_i) \oplus C_{i-1} \qquad (5.6)$$

Віднімання. Суматори можуть складати позитивні і негативні числа, використовуючи представлення числа в додатковому коді. Віднімання виконується майже також просто: міняється знак другого числа, потім числа складаються. Зміна знаку числа в додатковому коді виконується шляхом інверсії бітів і збільшення 1 (рис. 5.10).

Для обчислення Y = A – B спочатку створюється додатковий код числа B: інвертуються розряди B і додається 1; $-B = \overline{B} + 1$. Набутого значення складається з A. Ета сума може бути отримана одним суматором з перенесенням, що поширюється, шляхом складання $A + \overline{B}$ при $C_{in} = 1$. На рисунку

2. показано умовне позначення пристрою віднімання і базову апаратну реалізацію для обчислення Y = A – B.

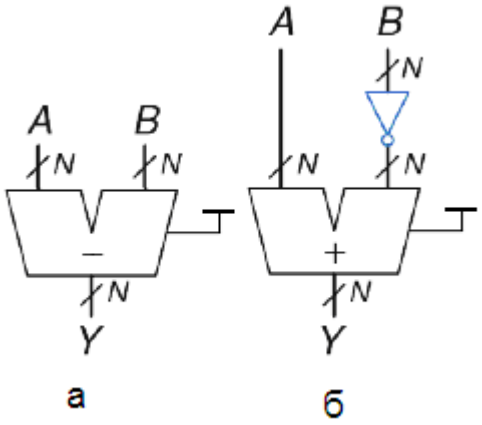


Рисунок 5.10 – Пристрій віднімання: а – умовне позначення; б - реалізація

5.3 Повний чотирьохрозрядний паралельний суматор

З принципу складання багаторозрядних двійкових чисел виходить, що в кожному і-розряді знаходиться сума S трьох чисел по модулю два: доданків A_i, B_i і перенесення, яке надійшло із молодшого розряду P_i, та формується сигнал перенесення в старший розряд P_{i+1}.

Проаналізуємо таблицю істинності однорозрядного суматора (табл. 5.1) і запишемо логічні вирази для вихідних величин.

$$S = A\overline{B}\overline{P_i} + \overline{A}B\overline{P_i} + \overline{A}\overline{B}P_i + ABP_i;$$

$$P_{i+1} = AB\overline{P_i} + A\overline{B}P_i + \overline{A}BP_i + ABP_i.$$

Таблиця 5.1 – Таблиця функціонування однорозрядного суматора

Вхід			Вихід	
Доданки		Перенесення	Сума	Перенесення
A	B	P_i	S	P_{i+1}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

По цих функціях можна побудувати суматор на елементах І, АБО (рис. 5.11).

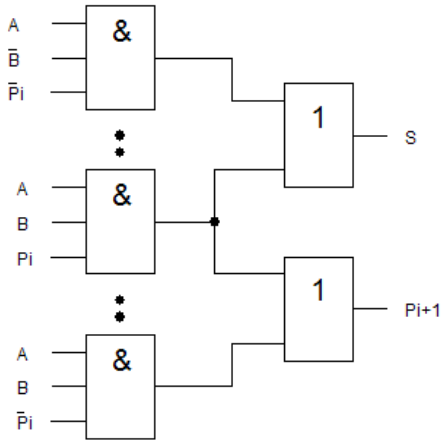


Рисунок 5.11 – Суматор на елементах І – АБО

У цифровій обчислювальній техніці використовуються однорозрядні схеми підсумовування з двома і трьома входами. Причому, перші називаються напівсуматорами, а другі – повними однорозрядними суматорами. Напівсуматори можуть використовуватися лише для підсумовування молодших розрядів чисел. Повні однорозрядні суматори мають додатковий третій вхід, на який подається перенесення з попереднього розряду при підсумовуванні багаторозрядних чисел.

У таблиці 5.2 приведена таблиця істинності напівсуматора, на підставі якої складена його структурна формула у вигляді ДДНФ.

Таблиця 5.2 – Таблиця істинності напівсуматора

A	B	S	P_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{A}B + A\overline{B}$$

$$P_{i+1} = AB$$

Логічний елемент “Виключне АБО” застосовується як суматор за модулем 2. Якщо до елементу “Виключне АБО” додати елемент, який є формувачем одиниці старшого розряду (генератор перенесення, який створює вихід

Р), то отримаємо однорозрядний напівсуматор (рис. 5.12)

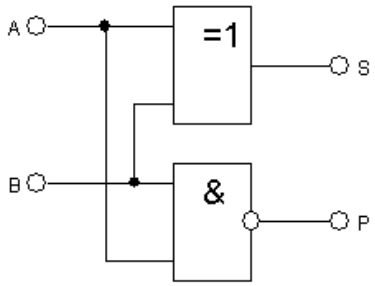


Рисунок 5.12 – Схема напівсуматора на елементі “Виключне АБО”

Схема дає при A = B = 1 результат S = 0 (це молодший розряд суми) і P = 1 (старший розряд, який тут має назву одиниці перенесення). У цьому випадку на обох виходах напівсуматора міститься двохрозрядне двійкове вихідне слово A + B = 01+ 01 = $10_{(2)}$. Його десятковий еквівалент 1+1 = $2_{(10)}$.

Схему повного однорозрядного суматора можна отримати на основі двох схем напівсуматорів і схеми «АБО», як показано на рисунку 5.13.

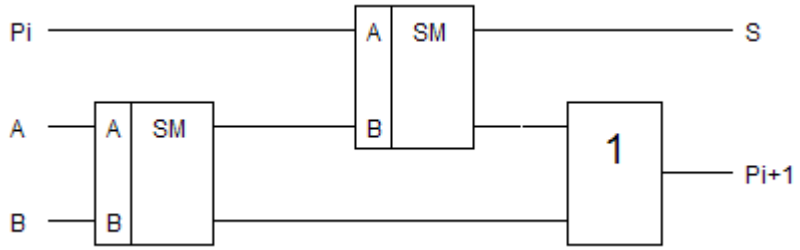


Рисунок 5.13 – Схема повного однорозрядного суматора

На основі одно розрядних суматорів будуються n-розрядні (паралельні) суматори. Алгоритм роботи чотирьохрозрядного суматора, побудований для випадку складання кодів, приведений на рисунку 5.14.

Алгоритм розглядається за умовами використання операції складання за модулем 2, та логічної операції АБО.

Схема паралельного суматора з послідовним перенесенням приведена на рисунку 5.15.

Кількість суматорів дорівнює числу розрядів чисел. Вихід перенесення P_{i+1} кожного суматора з'єднується з входом перенесення P_i наступного більш старшого розряду. На вході перенесення молодшого розряду встановлюється потенціал «0», оскільки сигнал перенесення сюди не поступає.

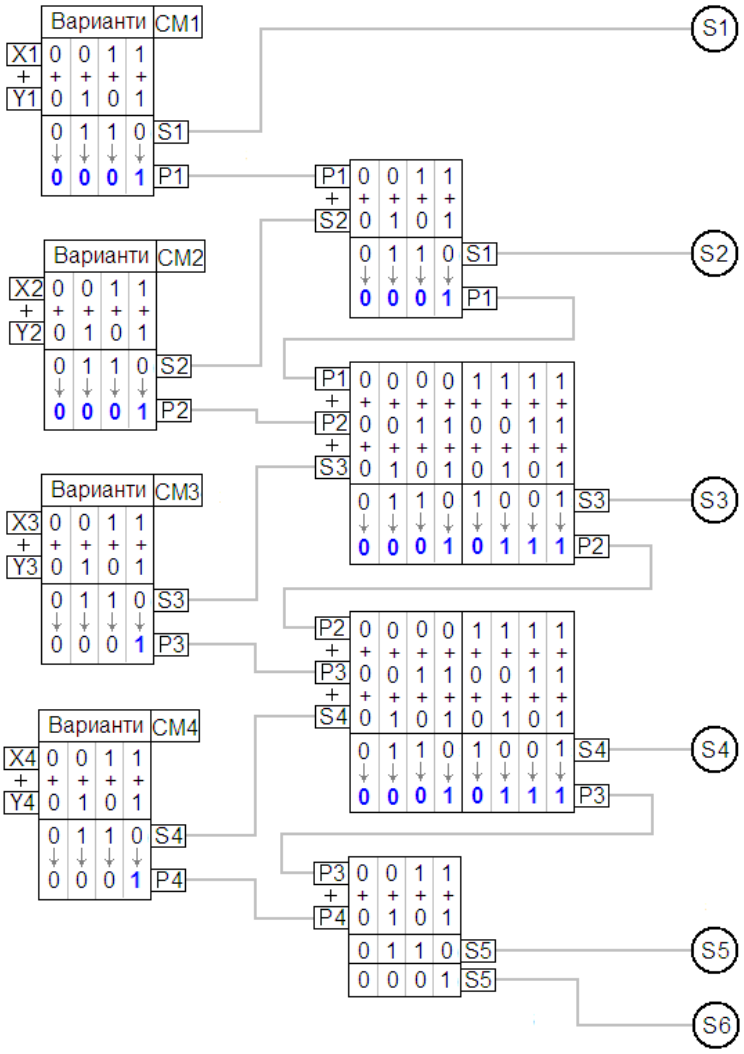


Рисунок 5.14 – Алгоритм роботи чотирьохрозрядного паралельного сумматора

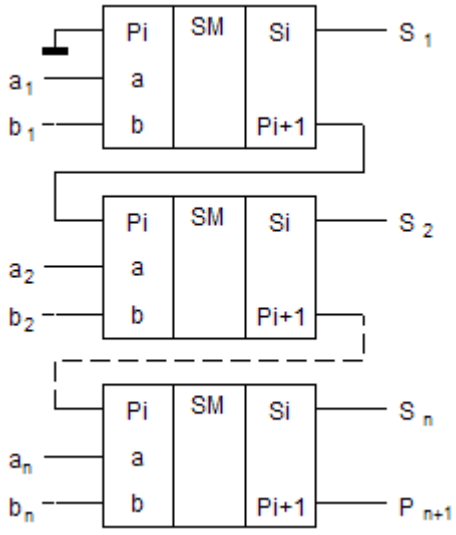


Рисунок 5.15 – Паралельне підсумовування багаторозрядних чисел з послідовним перенесенням

Доданки a_i і b_i підсумовуються у всіх розрядах одночасно, а перенесення P_{i+1} поступає із закінченням операції складання в попередньому розряді. Швидкодія таких суматорів обмежена затримкою перенесення, оскільки формування перенесення на виході старшого розряду не може статися до тих пір, доки сигнал перенесення не поширіться по всьому колу суматорів.

В чотирьохрозрядному паралельному суматорі з послідовним перенесенням число напівсуматорів дорівнює числу розрядів. Вихід перенесення P кожного суматора сполучений з входом перенесення наступного, більш старшого розряду. Доданки A_i і B_i складаються у всіх розрядах одночасно, а перенесення P поступає із закінченням операції складання в попередньому розряді. Схема функціонування чотирьох розрядного паралельного суматора представлена на рисунку 5.16.

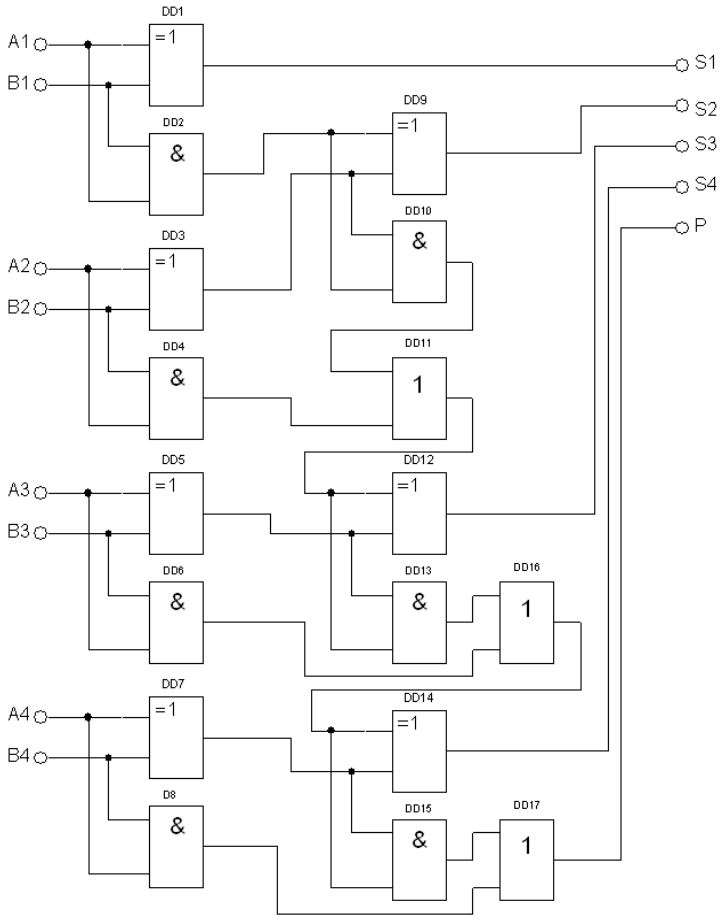


Рисунок 5.16 - Схема функціонування чотирьох розрядного суматора з послідовним перенесенням

У вигляді окремих мікросхем випускаються однорозрядні, двохрозрядні і чотирьохрозрядні суматори. У сімействі ТТЛ це мікросхеми відповідно К155ИМ1, ИМ2 і ИМ3.

Схема електрична принципова чотирьох розрядного паралельного суматора представлена на рисунку 5.17.

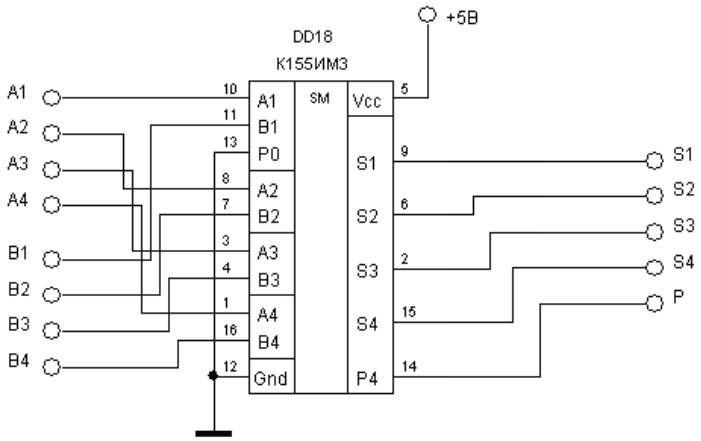


Рисунок 5.17 - Схема електрична принципова чотирьох розрядного паралельного суматора

5.4 Повний чотирьохрозрядний паралельний суматор-віднімач

Число напівсуматорів тут також дорівнює числу розрядів. Вихід перенесення P кожного суматора сполучений з входом перенесення наступного, більш старшого розряду (рис. 5.16). Доданки A_i і B_i складаються у всіх розрядах одночасно, а перенесення P поступає із закінченням операції складання в попередньому розряді.

Для того, щоб суматор виконував функції віднімача, число B слід представити в додатковому коді. Як результат отримаємо чотирьохрозрядний код. Стан перенесення 1 – позитивне число, 0 – негативне число. Таблиця перетворювача двійкового коду в додатковий (табл. 5.3) дозволяє отримати рівняння функціонування:

$$X1 = \overline{B4}\,\overline{B3}\,\overline{B2}B1 + \overline{B4}\,\overline{B3}B2B1 + \overline{B4}B3\,\overline{B2}B1 + \overline{B4}B3B2B1 + B4\,\overline{B3}\,\overline{B2}B1 + B4\,\overline{B3}B2B1 + B4B3\,\overline{B2}B1 + B4B3B2B1;$$

$$X2 = \overline{B4}\,\overline{B3}\,\overline{B2}B1 + \overline{B4}\,\overline{B3}B2\overline{B1} + \overline{B4}B3\,\overline{B2}B1 + \overline{B4}B3B2\overline{B1} + B4\,\overline{B3}\,\overline{B2}B1 + B4\,\overline{B3}B2\overline{B1} + B4B3\,\overline{B2}B1 + B4B3B2\overline{B1};$$

$$X3 = \overline{B4}\,\overline{B3}\,\overline{B2}B1 + \overline{B4}\,\overline{B3}B2\overline{B1} + \overline{B4}B3\,\overline{B2}B1 + \overline{B4}B3B2\overline{B1} + B4\,\overline{B3}\,\overline{B2}B1 + B4\,\overline{B3}B2\overline{B1} + B4B3\,\overline{B2}B1 + B4B3B2\overline{B1};$$

$$X4 = \overline{B4}\,\overline{B3}\,\overline{B2}B1 + \overline{B4}\,\overline{B3}B2\overline{B1} + \overline{B4}B3\,\overline{B2}B1 + \overline{B4}B3B2\overline{B1} + B4\,\overline{B3}\,\overline{B2}B1 + B4\,\overline{B3}B2\overline{B1} + B4B3\,\overline{B2}B1 + B4B3B2\overline{B1};$$

Таблиця 5.3 – Таблиця функціонування перетворювача

B4	B3	B2	B1	$\overline{B4}$	$\overline{B3}$	$\overline{B2}$	$\overline{B1}$	X4	X3	X2	X1
0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	0
0	0	1	1	1	1	0	0	1	1	0	1
0	1	0	0	1	0	1	1	1	1	0	0
0	1	0	1	1	0	1	0	1	0	1	1
0	1	1	0	1	0	0	1	1	0	1	0
0	1	1	1	1	0	0	0	1	0	0	1
1	0	0	0	0	1	1	1	1	0	0	0
1	0	0	1	0	1	1	0	0	1	1	1
1	0	1	0	0	1	0	1	0	1	1	0
1	0	1	1	0	1	0	0	0	1	0	1
1	1	0	0	0	0	1	1	0	1	0	0
1	1	0	1	0	0	1	0	0	0	1	1
1	1	1	0	0	0	0	1	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Спрощуємо рівняння методом карт Карно (рис. 5.18).

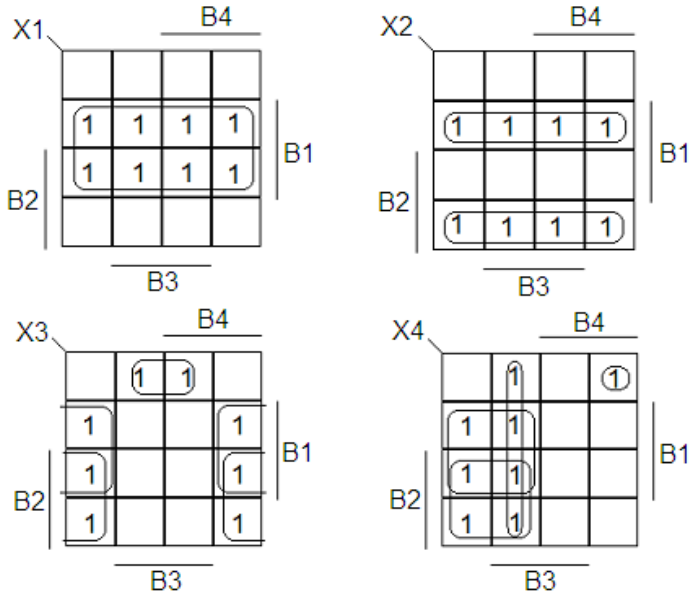


Рисунок 5.18 – Спрощення отриманих функцій

Рівняння функціонування перетворювача двійкового коду в додатковий код приведені до єдиного базису:

$$X1 = B1;$$

$$X2 = \overline{B2}B1 + B2\overline{B1} = \overline{\overline{B2}B1 + B2\overline{B1}} = \overline{\overline{B2}B1} \cdot \overline{B2\overline{B1}};$$

$$X3 = \overline{B3}B2 + \overline{B3}B1 + B3\overline{B2}\,\overline{B1} = \overline{\overline{B3}B2 + \overline{B3}B1 + B3\overline{B2}\,\overline{B1}} = \overline{\overline{B3}B2} \cdot \overline{\overline{B3}B1} \cdot \overline{B3\overline{B2}\,\overline{B1}};$$

$$X4 = \overline{B4}B2 + \overline{B4}B1 + \overline{B4}B3 + B4\overline{B3}\,\overline{B2}\,\overline{B1} = \overline{\overline{B4}B2 + \overline{B4}B1 + \overline{B4}B3 + B4\overline{B3}\,\overline{B2}\,\overline{B1}} =$$

$$= \overline{\overline{B4}B2} \cdot \overline{\overline{B4}B1} \cdot \overline{\overline{B4}B3} \cdot \overline{B4\overline{B3}\,\overline{B2}\,\overline{B1}}.$$

Згідно з рівняннями построїмо структурну схему перетворювача двійкового коду в додатковий (рис. 5.19). Схема має рівень складності по Квайну 34. Тому розглянемо варіант використання чотирьохрозрядного суматора на одному вході якого буде постійна одиниця.

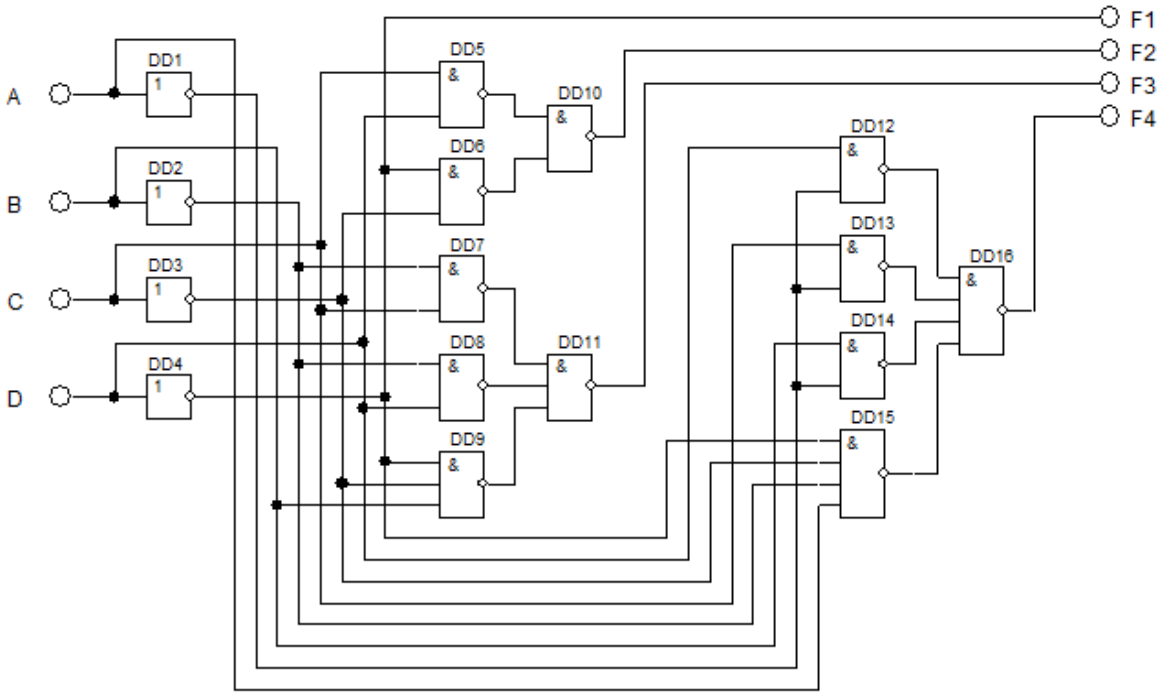


Рисунок 5.19 - Структурна схема перетворювача двійкового коду в додатковий

У вигляді окремих мікросхем випускаються однорозрядні, двохрозрядні і чотирьохрозрядні суматори. У сімействах ТТЛ і ТТЛШ це мікросхеми відповідно К555ИМ1, ИМ2 і ИМ3. Схема електрична перетворювача двійкового коду в додатковий представлена на рисунку 5.20.

Схема електрична принципова чотирьохрозрядного паралельного суматора – віднімача (рис. 5.21) містить: вхідне число B4B3B2B1 у інверсному представленні; перетворювач у додатковий код на мікросхемі К555ИМ3; вхідне число A4A3A2A1 у двійковому коді; чотирьохрозрядний паралельний суматор на мікросхемі К555ИМ3; вихідне число S4S3S2S1.

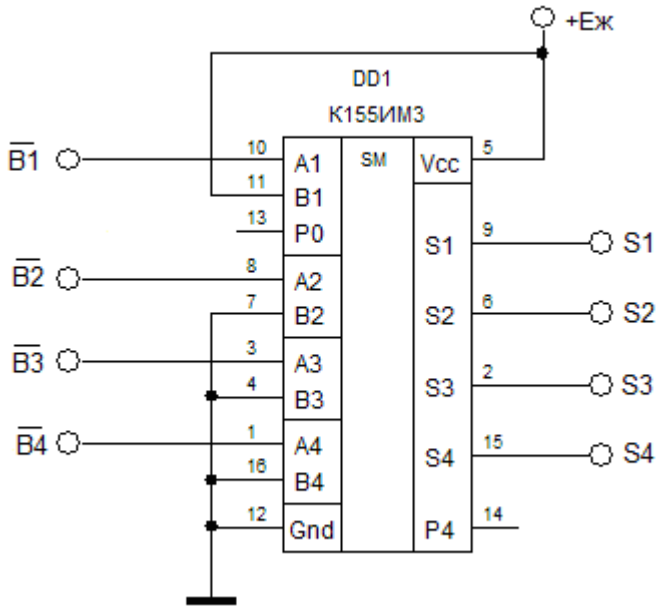


Рисунок 5.20 - Схема електрична перетворювача двійкового коду числа B4B3B2B1 в додатковий S4S3S2S1

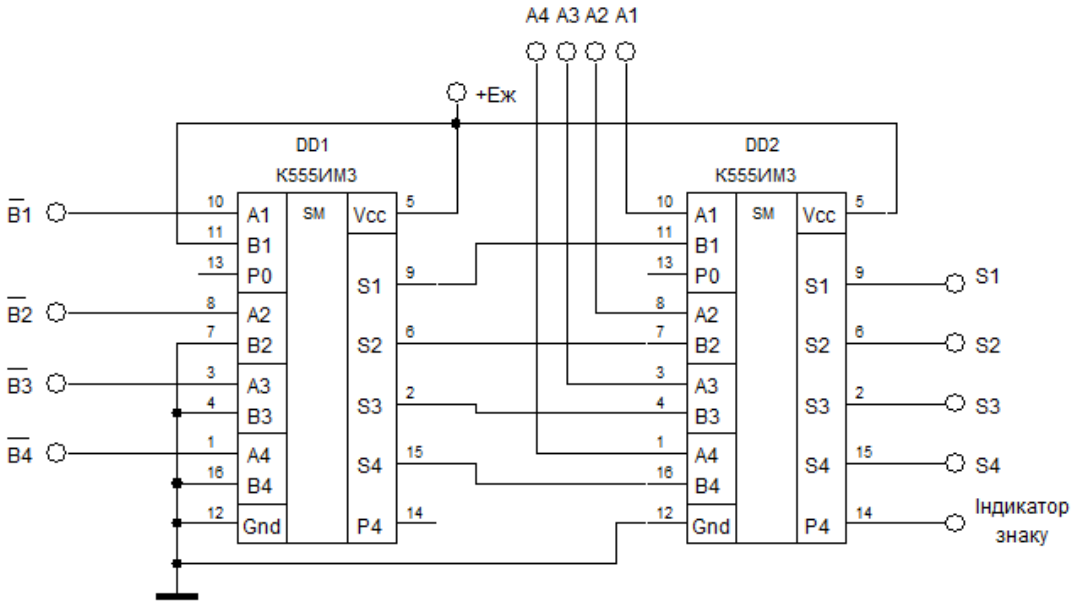


Рисунок 5.21 - Схема електрична принципова чотирьохрозрядного паралельного суматора – віднімача

5.5 Цифровий компаратор багаторозрядних чисел

Цифрові компаратори призначені для порівняння багаторозрядних чисел. При порівнянні багаторозрядних чисел використовують метод суперпозиції, тобто розбивають основне завдання на підзадачі. Алгоритм такий – спочатку порівнюють значення старших розрядів; якщо вони різні, то ці роз-

ряди і визначають результат порівняння. Якщо ж вони рівні, то необхідно порівнювати наступні за ним більш молодші розряди і так далі.

Дві кода А і В вважаються рівними, якщо попарно рівні їх однойменні розряди. Функція F(A = B) дорівнює 1, якщо $A_i = B_i$ для всіх і, інакше її значення дорівнює нулю. Компаратор повинен виробляти сигнали у випадках коли коди відрізняються між собою A > B і A < B. Порівняння проводиться із старших розрядів. Компаратор функціонує згідно таблиці 5.4.

Таблиця 5.4 - Функціональна таблиця цифрового компаратора

A3, B3	A2, B2	A1, B1	A0, B0	A>B	A<B	A=B
A3>B3	×	×	×	0	1	0
A3<B3	×	×	×	1	0	0
A3=B3	A2>B2	×	×	0	1	0
A3=B3	A2<B2	×	×	1	0	0
A3=B3	A2=B2	A1>B1	×	0	1	0
A3=B3	A2=B2	A1<B1	×	1	0	0
A3=B3	A2=B2	A1=B1	A0>B0	0	1	0
A3=B3	A2=B2	A1=B1	A0<B0	1	0	0
A3=B3	A2=B2	A1=B1	A0=B0	0	0	1

Розглянемо випадок порівняння одного і-розряда (таблиця. 5.5).

Таблиця 5.5 – Порівняння і-розряда

A_iB_i	$A_i>B_i$	$A_i<B_i$	$A_i=B_i$
00	0	0	1
01	0	1	0
10	1	0	0
11	0	0	1

Згідно таблиці 5.5 рівняння функціонування схеми порівняння і-розряда:

$$\begin{aligned} F(A>B) &= A\overline{B}; \\ F(A<B) &= \overline{A}B; \\ F(A=B) &= AB + \overline{A}\,\overline{B} = \overline{A \oplus B}. \end{aligned} \qquad (5.1)$$

Згідно рівняннями 5.1 і таблицею 5.5 складемо рівняння функціонування чотирьохрозрядного цифрового компаратора.

$$F_{A>B} = A_3\overline{B_3} + (\overline{A_3 \oplus B_3})A_2\overline{B_2} + (\overline{A_3 \oplus B_3})(\overline{A2 \oplus B_2})A_1\overline{B_1} +$$
$$+ (\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})(\overline{A_1 \oplus B_1})A_0\overline{B_0} = A_3\overline{B_3} + (\overline{A_3 \oplus B_3}) \cdot [A_2\overline{B_2} + (\overline{A2 \oplus B_2})A_1\overline{B_1} +$$
$$+ (\overline{A_2 \oplus B_2})(\overline{A_1 \oplus B_1})A_0\overline{B_0}] = A_3\overline{B_3} + (\overline{A_3 \oplus B_3}) \cdot [A_2\overline{B_2} + (\overline{A2 \oplus B_2}) \cdot [A_1\overline{B_1} +$$
$$+ (\overline{A_1 \oplus B_1})A_0\overline{B_0}]] = A_3\overline{B_3} + (\overline{A_3}\,\overline{B_3} + A_3B_3) \cdot [A_2\overline{B_2} + (\overline{A_2}\,\overline{B_2} + A_2B_2) \cdot [A_1\overline{B_1} +$$
$$+ (\overline{A_1}\,\overline{B_1} + A_1B_1)A_0\overline{B_0}]] = [A_3\overline{B_3} + (\overline{A_3}\,\overline{B_3} + A_3B_3)] \cdot [A_3\overline{B_3} + [A_2\overline{B_2} + (\overline{A_2}\,\overline{B_2} + A_2B_2)] \cdot$$
$$\cdot [A_2\overline{B_2} + [A_1\overline{B_1} + (\overline{A_1}\,\overline{B_1} + A_1B_1)] \cdot [A_1\overline{B_1} + A_0\overline{B_0}]]] =$$
$$= (A_3 + \overline{B_3}) \cdot [A_3\overline{B_3} + (A_2 + \overline{B_2}) \cdot [A_2\overline{B_2} + (A_1 + \overline{B_1}) \cdot (A_1\overline{B_1} + A_0\overline{B_0})]] =$$
$$= (A_3 + \overline{B_3}) \cdot [A_3\overline{B_3} + (A_2 + \overline{B_2}) \cdot [A_2\overline{B_2} + A_1\overline{B_1} + (A1 + \overline{B1})A_0\overline{B_0})]].$$

$$F_{A<B} = \overline{A}_3B_3 + (\overline{A_3 \oplus B_3})\overline{A}_2B_2 + (\overline{A_3 \oplus B_3})(\overline{A2 \oplus B_2})\overline{A}_1B_1 +$$
$$+ (\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})(\overline{A_1 \oplus B_1})\overline{A}_0B_0 = \overline{A}_3B_3 + (\overline{A_3 \oplus B_3}) \cdot [\overline{A}_2B_2 + (\overline{A2 \oplus B_2})\overline{A}_1B_1 +$$
$$+ (\overline{A_2 \oplus B_2})(\overline{A_1 \oplus B_1})\overline{A}_0B_0] = \overline{A}_3B_3 + (\overline{A_3 \oplus B_3}) \cdot [\overline{A}_2B_2 + (\overline{A2 \oplus B_2}) \cdot [\overline{A}_1B_1 +$$
$$+ (\overline{A_1 \oplus B_1})\overline{A}_0B_0]] = \overline{A}_3B_3 + (\overline{A_3}\,\overline{B_3} + A_3B_3) \cdot [\overline{A}_2B_2 + (\overline{A_2}\,\overline{B_2} + A_2B_2) \cdot [\overline{A}_1B_1 +$$
$$+ (\overline{A_1}\,\overline{B_1} + A_1B_1)\overline{A}_0B_0]] = [\overline{A}_3B_3 + (\overline{A_3}\,\overline{B_3} + A_3B_3)] \cdot [\overline{A}_3B_3 + [\overline{A}_2B_2 + (\overline{A_2}\,\overline{B_2} + A_2B_2)] \cdot$$
$$\cdot [\overline{A}_2B_2 + [\overline{A}_1B_1 + (\overline{A_1}\,\overline{B_1} + A_1B_1)] \cdot [\overline{A}_1B_1 + \overline{A}_0B_0]]] =$$
$$= (\overline{A}_3 + B_3) \cdot [\overline{A}_3B_3 + (\overline{A}_2 + B_2) \cdot [\overline{A}_2B_2 + (\overline{A}_1 + B_1) \cdot (\overline{A}_1B_1 + \overline{A}_0B_0)]] =$$
$$= (\overline{A}_3 + B_3) \cdot [\overline{A}_3B_3 + (\overline{A}_2 + B_2) \cdot [\overline{A}_2B_2 + \overline{A}_1B_1 + (\overline{A}_1 + B_1)\overline{A}_0B_0)]].$$

$$F_{A=B} = (\overline{A_3 \oplus B_3}) \cdot (\overline{A_2 \oplus B_2}) \cdot (\overline{A_1 \oplus B_1}) \cdot (\overline{A_0 \oplus B_0}) .$$

Розрахована структурна схема цифрового компаратора представлена на рисунку 5.22. Їй повністю відповідає мікросхема К555СП1. Мікросхема призначена для порівняння чотирьохрозрядних двійкових чисел, представлених в прямому коді. Мікросхема має засоби для нарощування розрядності порівнюваних чисел. На вхід Y = X при порівнянні самих молодших розрядів має бути поданий високий рівень напруги.

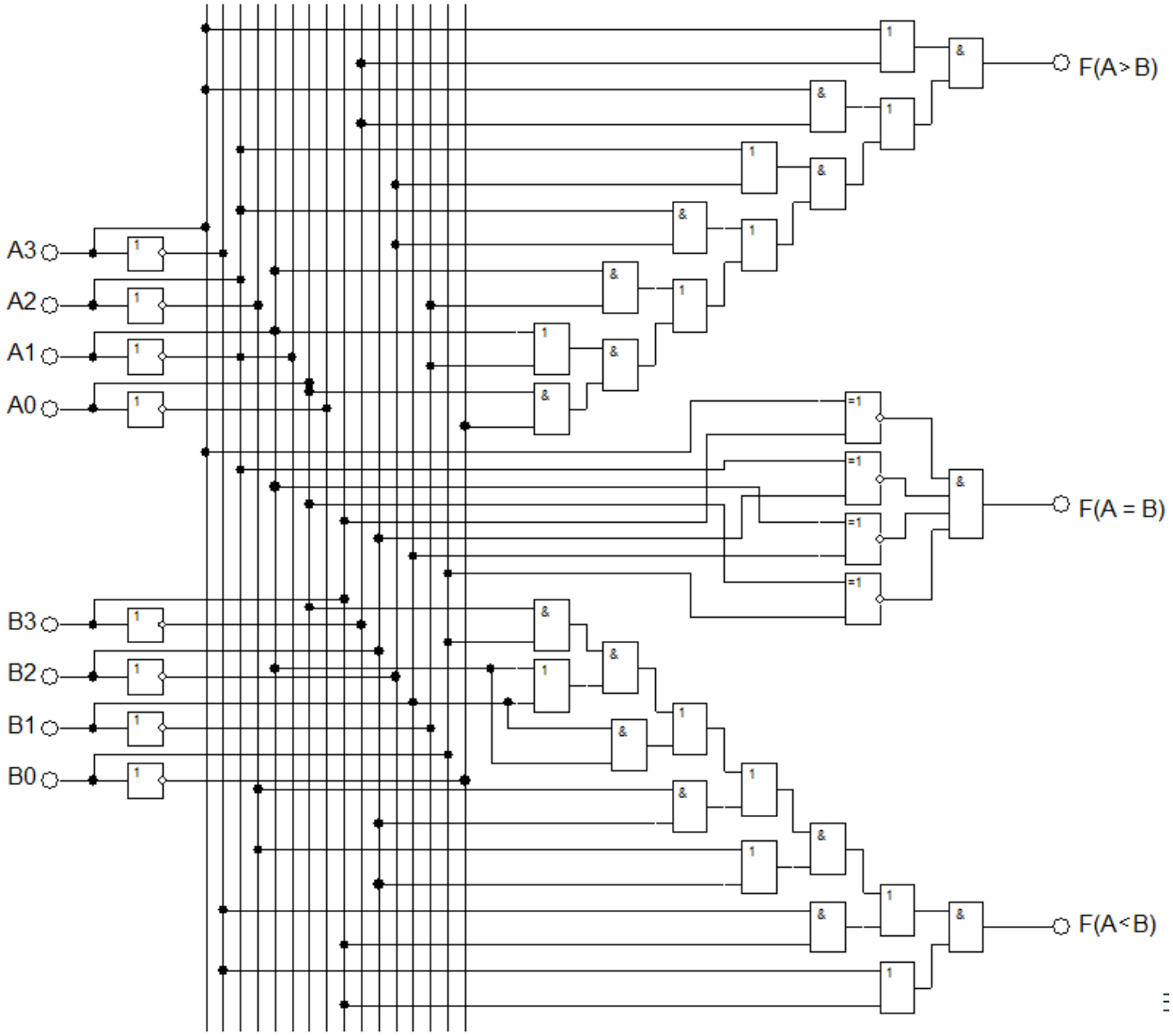


Рисунок 5.22 - Структурна схема чотирьохрозрядного цифрового компаратора

На рисунке 5.23 показано соединение двух таких схем, для увеличения разрядности сравниваемых чисел до восьми.

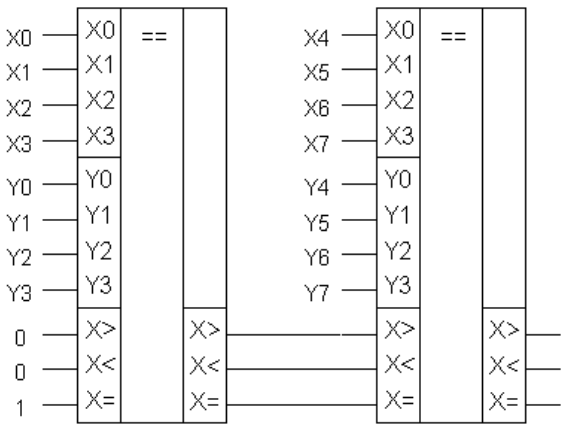


Рисунок 6.5 – Схема порівняння багаторозрядних чисел

Для правильного результату порівняння чисел X = (X7,X6...,X0) і Y=(Y7,Y6...,Y0) на вхід X = необхідно подати 1. Схема порівняння входить до складу АЛП мікропроцесора.