МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ
УНІВЕРСИТЕТ

В.Д. ТАРАРАКА

ПРИКЛАДНА ТЕОРІЯ ЦИФРОВИХ АВТОМАТІВ

Навчальний посібник

За рішенням Вченої ради
ЖДТУ
протокол № 3
від 25. 03. 2019 р.

ЖДТУ
2019

УДК 681.3
Т19

Рецензенти: професор кафедри комп'ютерних інтегрованих технологій Житомирського військового інституту імені С.П.Корольова, Заслужений працівник освіти України, доктор технічних наук, професор ***Пількевич І.А.***;

керівник навчально-наукового центру інформаційних технологій Житомирського національного агроекологічного університету, доктор технічних наук, доцент ***Молодецька К.В.;***

завідувач кафедри метрології та інформаційно-вимірювальної техніки Житомирського державного технологічного університету, доктор технічних наук, професор ***Подчашинський Ю.А.***

Тарарака В.Д.

Т19 Прикладна теорія цифрових автоматів: навчальний посібник. – Житомир: ЖДТУ, 2019. – 183с.

Навчальний посібник присвячений питанням теоретичних і практичних основ комп'ютерної арифметики іалгебри логіки, вивчення методів подання чисел в ЕОМ, алгоритмів виконання основних арифметичних та логічних операцій з числами в різних системах числення,аналізу та синтезу цифрових операційних та керуючих автоматів.

Навчальний посібникдає студентам необхідну теоретичну і практичну підготовку для того, щоб вміти розробляти і аналізувати алгоритми переробки дискретної інформації складних процесів, складати структурні схеми комбінаційних логічних схем та автоматів з пам'яттю, ефективно розв'язувати практичні задачі з прикладної теорії цифрових автоматів.

Для студентів, що спеціалізуються в області інформаційних технологій, і які вивчають дисципліни "Прикладна теорія цифрових автоматів", "Електроніка та мікропроцесорна техніка", "Основи побудови автоматизованих систем управління", "Архітектура комп'ютерних систем".

УДК 681. 3

ЗМІСТ

СПИСОК УМОВНИХ СКОРОЧЕНЬ.....5
ПЕРЕДМОВА.....7
РОЗДІЛ 1. АРИФМЕТИЧНІ ОСНОВИ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ.....11
1.1.Системи числення, які використовуються в обчислювальній техніці.....11
1.1.1. Загальні поняття про системи числення.....11
1.1.2. Переведення чисел з одних систем числення в інші.....18
1.1.2.1. Переведення цілих чисел з одної позиційної системи числення в іншу діленням на основу нової системи числення.....19
1.1.2.2. Переведення дробових чисел з однієї системи числення в іншу множенням на основу нової системи числення.....20
1.1.2.3. Метод безпосереднього заміщення.....24
1.1.2.4. Переведення чисел з вісімкової і шістнадцяткової систем числення у двійкову та навпаки.....26
1.1.3. Арифметичні операції в двійковій системі числення.....29
1.2. Подання чисел в засобах обчислювальної техніки.....31
1.2.1. Подання чисел у формі з фіксованою комою.....31
1.2.2. Подання чисел у формі з плаваючою комою.....34
1.3. Способи кодування двійкових чисел.....37
1.3.1. Прямий код двійкових чисел.....38
1.3.2. Зворотний код двійкових чисел.....39
1.3.3. Доповняльний код двійкових чисел.....41
1.3.4. Модифіковані коди.....43
1.4. Алгоритми виконання арифметичних операцій у цілочисельних операційних пристроях.....45
1.4.1. Алгоритми виконання операцій додавання і віднімання.....46
1.4.2. Алгоритми множення кодів чисел в АЛП з фіксованою комою.....48
1.4.3. Алгоритми ділення кодів чисел в АЛП з фіксованою комою.....57
1.4.4. Особливості виконання арифметичних операцій в АЛП з плаваючою комою.64
РОЗДІЛ 2. ЛОГІЧНІ ОСНОВИ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ.....69
2.1. Поняття логічної функції.....69
2.2. Основні закони і рівносильності алгебри логіки.....73
2.3. Форми подання логічних функцій.....75
2.4. Мінімізація логічних функцій.....77
2.4.1. Мінімізація логічних функцій методом безпосереднього перетворення.....78
2.4.2. Мінімізація логічних функцій методом Квайна.....79
2.4.3. Мінімізація логічних функцій табличним методом.....82
2.4.4. Мінімізація неповністю заданих логічних функцій.....85
2.4.5. Поняття про сумісну мінімізацію логічних функцій.....87
2.5. Основи синтезу комбінаційних схем.....88
2.6. Цифрові автомати з пам’яттю.....91
2.6.1. Загальні відомості про цифрові автомати з пам’яттю.....91
2.6.2. Способи задавання цифрових автоматів.....93
2.6.3. Канонічний метод структурного синтезу цифрових автоматів.....95
РОЗДІЛ 3. ФУНКЦІОНАЛЬНІ ВУЗЛИ ЦИФРОВИХ ЕЛЕКТРОННИХ ОБЧИСЛЮВАЛЬНИХ МАШИН.....104
3.1. Загальні відомості про системи елементів та їх характеристики.....104
3.2. Тригери.....107
3.2.1. Класифікація схем тригерів.....107

3.2.2. Синтез RS-тригера 108
3.2.3. Синтез Т-тригера 112
3.2.4. Синтез комбінованого тригера типу RST 115
3.2.5. Синтез D-тригера 117
3.2.6. Синтез JK-тригерів 119
3.2.7. Модифікації синхронних тригерів 125
3.3. Функціональні вузли цифрових ЕОМ комбінаційного типу 126
3.3.1. Синтез двійкового шифратора 126
3.3.2. Синтез двійкових дешифраторів 129
3.3.3. Однорозрядний напівсуматор комбінаційного типу 135
3.3.4. Однорозрядний суматор комбінаційного типу 137
3.3.5. Поняття про метод композиції вузлів 140
3.3.6. Багаторозрядні комбінаційні суматори послідовної дії 142
3.3.7. Багаторозрядні комбінаційні суматори паралельної дії 144
3.4. Функціональні вузли цифрових ЕОМ накопичуючого типу 146
3.4.1. Суматор накопичуючого типу з послідовним переносом 146
3.4.2. Суматор накопичуючого типу з наскрізним переносом 149
3.4.3. Синтез підсумовуючого лічильника 151
3.4.4. Синтез віднімаючого лічильника 156
3.4.5. Лічильники з довільним модулем рахунку 157
3.4.6. Загальні відомості про регістри 164
3.4.7. Синтез регістрів паралельної дії 165
3.4.8. Синтез регістрів послідовної дії 173
Список літератури 181

СПИСОК УМОВНИХ СКОРОЧЕНЬ

АЛП – арифметико – логічний пристрій
АЦП – аналого – цифровий перетворювач
АСК – архітектура системи команд
БОЗ – блок обробки знаків
БОМ – блок обробки мантис
БОП – блок обробки порядків
ВВ – ввід/вивід
ВІС – велика інтегральна схема
ЕОМ – електронна обчислювальна машина
ДНФ – досканала нормальна форма
ЗЕ – запам'ятовуючий елемент
ЗЗП – зовнішній запам'ятовуючий пристрій
ЗП – запам'ятовуючий пристрій
ІМС – інтегральна мікросхема
КС – комп'ютерна система
ЛФ – логічна функція
Лч – двійковий лічильник
ЛЕ – логічний елемент
МК – мікрокоманда
МО – мікрооперація
МП – мікропроцесор
ОБ – операційний блок
ОЗП – оперативний запам'ятовуючий пристрій
ОМ – обчислювальна машина
ОП – основна пам'ять
ОПр – операційний пристрій
ОС – обчислювальна система
ОТ – обчислювальна техніка
ПЕ – процесорний елемент
ПЗП – постійний запам'ятовуючий пристрій
ПК – персональний комп'ютер
ПЛМ – програмована логічна матриця
ПМП – пам'ять мікропрограм
ПП – периферійний пристрій
Рг – регістр
РЧ – регістр частки
См – суматор
СУ – сигнал управління
СЧД – сума часткових добутків
ФБ – функціональний блок

ФПС ЛЕ – функціонально повна система логічних елементів
ЦА – цифровий автомат
ЦАП – цифро – аналоговий перетворювач
ЦП – центральний процесор
ЧД – частковий добуток
ЧЗ – частковий залишок

CISC – Complex Instruction Set Computer – архітектура з повним набором команд
FIFO – First In First Out – першим прийшов – першим вийшов
LIFO – Last In First Out –останнім прийшов – першим вийшов
OHS – напівсуматор
RISC – Reduced Instruction Set Computer – архітектура зі скороченим набором команд

ПЕРЕДМОВА

Створення комп'ютера варто вважати найвищим досягненням людства за минулий час, оскільки немає жодної сфери діяльності від будь-якого виробництва до різноманітного творчості, де б з його допомогу не відбувалася справжня революція. Комп'ютерна техніка допомогла значно підвищити ефективність розумової праці людини, що пов'язана з обчислювальними і кібернетичними процесами, з моделюванням і дослідженнями. Вона поставила на новий, вищий технічний рівень питання, пов'язані з автоматизацією технологічних виробничих процесів, контролем і управлінням, накопиченням і обробкою інформації. Важко знайти область техніки або науки, куди не проникла або не могла б проникнути обчислювальна техніка (ОТ).

Засоби обчислювальної техніки призначені для переробки інформації. Будь-яке керування ґрунтується на інформації. Поняття інформації має узагальнений філософський смисл і відображає одну з об'єктивних властивостей матеріального світу. Воно використовується в різних значеннях і має ряд визначень. Дамо, на наш погляд, більш узагальнене означення інформації.

Інформація – це сукупність повідомлень про властивості об'єктів, зміну їх стану, про протікання процесів у природі, виробництві та суспільстві.

Взагалі, інформація – це те, що зменшує невідомість, невизначеність (ентропію). Чим більша ентропія системи (повідомлення), тим більше інформації можна отримати, якщо зняти невизначеність (наприклад, нове відкриття (нове знання) в науці знімає невизначеність відносно об'єкта дослідження). Термін "ентропія" увів німецький фізик Клаузіус для опису невизначеності стану речовини. Широке використання цей термін набув у термодинаміці Больцмана, звідки і був запозичений у теорію інформації.

Інформація передається за допомогою повідомлень (показники приладів, команди керування, слова, картинки, звуки тощо). Матеріальною оболонкою повідомлень є сигнали різних видів, тобто повідомлення передаються за допомогою сигналів (радіосигнали, звукові сигнали, кольорові сигнали тощо).

Отже, загальне означення інформації, яке було б строго науковим і задовольняло усі області науки і техніки дати дуже важко.

Можна виділити основні загальні властивості інформації: її можна приймати, передавати, зберігати та перетворювати. Ці властивості

використовуються для побудови різноманітних інформаційних (кібернетичних) систем, центральним елементом яких є обчислювальні системи.

Таким чином, перш ніж перейти до вивчення обчислювальних засобів обробки інформації необхідно ближче познайомитися з поняттям інформації та її представленням в комп'ютерних системах.

Для використання терміна "інформація" в комп'ютерних (кібернетичних) системах необхідна одиниця вимірювання, тобто кількісна міра інформації. В сучасних технічних системах для кількісної оцінки інформації використовують статистичний підхід, в рамках якого інформація розглядається як сукупність відомостей, повідомлень про поведінку деякої системи, яка випадково може знаходитись в одному з можливих станів. Така система має деяку ступінь невизначеності (ентропію) і фактичний стан її до отримання повідомлення залишається невідомим. Повідомлення про фактичний стан системи, яку ми розглядаємо, і є інформація про неї.

Якщо стан системи визначений і не може змінюватись або всі можливі зміни станів відомі до появи повідомлення про них, то смислу передавати повідомлення про систему немає. Воно не представляє інтересу, не дає нічого нового, тобто не несе інформацію.

Статистичний підхід до кількісної оцінки інформації, який запропонував американський математик Клод Шеннон (формула Шеннона), широко використовується в наукових дослідженнях та при розв'язуванні багатьох важливих практичних питань. Наприклад, задачі оптимального кодування повідомлень в системах передачі інформації, визначення пропускної спроможності каналів з завадами, розрахунок ємності запам'ятовуючих пристроїв для зберігання інформації тощо. При цьому кількість інформації можна вимірювати в бітах.

Інформація передається за допомогою сигналів, які можуть бути представленні в аналоговій або цифровій формах. Різниця між ними у тому, що перші безперервні, а другі дискретні. Сучасні комп'ютерні системи використовують цифрові сигнали, тому для обробки інформації необхідно перетворювати безперервні сигнали (в природі існують тільки аналогові сигнали) в дискретні. Цей процес в технічних системах отримав назву «аналого-цифрове перетворення (АЦП)». Він містить в собі три операції:

- дискретизацію (квантування) за часом;
- дискретизацію (квантування) за рівнем;
- кодування (представлення у вигляді цифрового двійкового коду).

В сучасних комп'ютерних системах для обробки інформації використовують таку одиницю, як байт – це група з восьми бітів (1байт = 8біт).

За допомогою 1 байта можна кодувати 256 різних значень, тобто представити (передати, зберігати) інформацію про 256 можливих станів системи:

$I = \log_2 256 = 8 біт = 1 байт$.

Таким чином, різноманітна інформація (текст, таблиці, звуки, малюнки тощо) представляються у вигляді байтів. Однак комп'ютер повинен розрізняти, яку інформацію несе кожен байт, де вона починається і закінчується, який формат мають дані, тобто кожна послідовність байтів інформації визначеного типу повинна реєструватись. Після реєстрації (присвоєння ім'я та типу даних) послідовність байтів організується у файл. Тільки файл можна зберігати у комп'ютері як інформацію.

Метою дисципліни "Прикладна теорія цифрових автоматів" є теоретична підготовка бакалаврів спеціальності 151 "Автоматизація та комп'ютерно-інтегровані технології", що включає в себе вивчення студентами арифметичних, логічних і схемотехнічних основ побудови цифрових пристроїв обробки інформації та принципів їх аналізу і синтезу, а також отримання практичних навичок з розробки, створення і використання цифрових автоматів (ЦА) різного призначення та їх окремих вузлів.

Під цифровим автоматом розуміють пристрій, що оперує з цифровою дискретною інформацією і характеризується кінцевою множиною внутрішніх станів, в які він переходить під впливом множин вхідних сигналів, при цьому є кінцева множина правил переходу з одного стану в інший.

На відміну від звичайного цифрового пристрою (комбінаційної схеми) цифровий автомат має пам'ять.

Вихідний сигнал ЦА залежить не тільки від сукупності вхідних сигналів а також від того стану, в якому автомат перебував в попередній момент часу і який зберігається в пам'яті. Вміст пам'яті автомату називають його станом. Функціонування автомату розглядають в дискретні моменти часу.

Будь-який пристрій ЕОМ можна представити як цифровий мікропрограмний автомат.

Навчальний посібник з даної дисципліни містить матеріал для вивчення систем числення і методів подання чисел в ЕОМ, алгоритмів виконання основних арифметичних та логічних операцій з числами в різних системах числення, основ математичної логіки, аналізу та синтезу цифрових операційних та керуючих автоматів. Посібник дає студентам необхідну теоретичну і практичну підготовку для того, щоб вміти розробляти і аналізувати алгоритми переробки дискретної інформації складних процесів, складати структурні схеми комбінаційних логічних схем та автоматів з пам'яттю, ефективно розв'язувати практичні задачі з прикладної теорії цифрових автоматів різного призначення

та їх окремих вузлів при проектуванні сучасних комп’ютеризованих систем управління і автоматики.

Навчальний посібник складається з трьох розділів:

- розділ 1 – «Арифметичні основи обчислювальної техніки»;
- розділ 2 - «Логічні основи обчислювальної техніки».
- розділ 3 - «Функціональні вузли комп’ютерних систем».

У першому розділі посібника висвітлюються форми і методи представлення чисел в сучасних комп’ютерних системах, алгоритми перетворення чисел різних систем числення, методи виконання основних арифметичних операцій в різних системах числення, структурні і операційні схеми основних арифметичних пристроїв.

У другому розділі посібника висвітлюються логічні основи цифрових автоматів, математичний апарат булевої алгебри, методи мінімізації булевих рівнянь, описи цифрових автоматів (ЦА), методи синтезу і аналізу ЦА (в т.ч. канонічні) на логічних елементах різних базисів, графи і граф-схеми автоматів (ГСА).

У третьому розділі посібника висвітлюються принципи побудови і функціонування основних типових вузлів, які широко використовуються в засобах обчислювальної техніки.

РОЗДІЛ 1

АРИФМЕТИЧНІ ОСНОВИ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

1.1. СИСТЕМИ ЧИСЛЕННЯ, ЯКІ ВИКОРИСТОВУЮТЬСЯ В ОБЧИСЛЮВАЛЬНІЙ ТЕХНІЦІ

1.1.1. Загальні поняття про системи числення

Системою числення називається сукупність заходів і правил для найменування і позначення чисел.

Умовні знаки, що використовуються для позначення чисел називаються цифрами. Далі будемо припускати, що кількість цифр кінцева, тобто абетка, на основі якої складаються числа в деякій системі числення, складається з кінцевого числа елементів (цифр).

Звичайно всі системи числення поділяються на два класи: *непозиційні* і *позиційн*і.

Непозиційною називають систему числення, в якій значенню кожної цифри у будь – якому місці запису числа існує один і той же кількісний еквівалент. Такі системи з'явилися раніше в історичному плані, наприклад, загальновідома римська нумерація. Але непозиційні системи числення знаходять обмежене застосування в ОТ, так як вони характеризуються дуже складними і громіздкими алгоритмами подання чисел і виконання арифметичних операцій.

Системи, в яких значення кожної цифри залежить від місця (позиції) у послідовності цифр при записі числа, носять назву ***позиційних***. Позиційною системою числення є широко розповсюджена звичайна десяткова система числення. Звичайно позиційні системи числення мають найменування, яке співпадає з кількістю цифр, що в ній використовуються. Наприклад, в десятковій системі числення використовується десять цифр від 0 до 9. а число 123 визначається як $1\cdot10^2 + 2\cdot10^1 + 3\cdot10^0$. З цього прикладу ми бачимо, що позиція кожної цифри має свій ваговий коефіцієнт, і перша позначає кількість сотень, а остання – кількість одиниць.

У сучасних цифрових ЕОМ використовуються головним чином позиційні системи числення, тому що в них простіше реалізуються правила, ніж в непозиційних системах [10].

Основними характеристиками позиційних систем числення є:

- основа системи числення – P;
- абетка цифр системи числення (кількість використовуваних в системі цифр);
- вага розрядів – R_i , де $i = \overline{0, n-1}$ - розрядність числа.

Основа позиційної системи числення є число, яке виявляє у скільки разів одиниця старшого розряду більша одиниці сусіднього молодшого розряду.

Абетка цифр позиційної системи числення характеризує значення цифр, які використовуються для зображення чисел у даній системі числення. Звичайно цифри обираються таким чином, щоб вони складали відрізок натурального ряду чисел, включаючи число “0”. У цьому випадку максимальне значення цифри, яка використовується у системі числення, дорівнює P – 1. У десятковій системі числення абетка характеризується значеннями цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Вага розряду визначає кількісне значення цифри у зображенні числа. Вагою розряду називається відношення кількісного еквівалента цифри, яка стоїть в *i*-му розряді a_i, до кількісного еквіваленту тієї ж цифри, яка стоїть у нульовому розряді.

$$R_i = \frac{a_i}{a_0} = \frac{p^i}{p^0} = p^i. \qquad (1.1)$$

В якості прикладу в табл. 1 наведено десяткове число 34043,9145, вказані номери його розрядів і вага цифр у відповідних розрядах.

Таблиця 1.1

Десяткове число	3	4	0	4	3	9	1	4	5
Номери розрядів	4	3	2	1	0	-1	-2	-3	-4
Вага розрядів	10^4	10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}

В даному прикладі одні й ті ж самі цифри повторюються у декількох розрядах, але їх кількісне значення різне. Так, вага цифри 4 у третьому розряді дорівнює 10^3, що надає їй кількісний вміст у даному числі в чотири тисячі одиниць, а вага тієї ж цифри 4 у мінус третьому розряді дорівнює 10^{-3} і кількісний вміст її дорівнює чотирьом тисячам часток одиниці.

Таким чином, в позиційній системі числення будь – яке число **A** може бути подане у вигляді:

$$A_{(p)} = a_{n-1}p^{n-1} + a_{n-2}p^{n-2} + \cdots + a_1p^1 + a_0p^0 + a_{-1}p^{-1} + a_{-2}p^{-2} + \cdots + a_{-m}p^{-m} = \sum_{i=-m}^{n-1} a_i\ p^i\ , \tag{1.2}$$

де a_i – значення цифри в і-м розряді;

p – основа системи числення;

m – кількість розрядів дрібної частини числа;

n – кількість розрядів цілої частини числа.

Для скорочення запису числа **A** вага розрядів не пишеться і вираз (1.2) можна подати у вигляді:

$$A_{(p)} = a_{n-1}a_{n-2}\ldots a_1a_0, a_{-1}a_{-2}\ldots a_{-m}. \tag{1.3}$$

Приклад. Десяткове число 3125,267 можна подати у вигляді:

$$3125{,}267 = 3\cdot10^3 + 1\cdot10^2 + 2\cdot10^1 + 5\cdot10^0 + 2\cdot10^{-1} + 6\cdot10^{-2} + 7\cdot10^{-3}\ .$$

В якості основи систем числення може бути взяте будь – яке відмінне від одиниці ціле число.

У залежності від того, яке число обране у якості основи системи числення, розрізняють: двійкову, трійкову, п’ятіркову, вісімкову, шістнадцяткову та ін. системи числення.

Систем числення можна побудувати незліченну кількість, але для того, щоб обрати систему числення для використання у цифрових ЕОМ необхідно враховувати цілу низку вимог. Основні з них такі [10]:

- однозначність подання чисел;
- можливість подання будь – якого числа із заданого діапазону чисел;
- простота виконання арифметичних і логічних операцій;
- зручність вводу початкових даних і виводу результатів обчислень;
- зручність відтворення кожної цифри стійкими станами декотрої фізичної системи. Кількість стійких станів фізичної системи повинно дорівнювати кількості цифр в системі числення, яку передбачають використовувати у цифровій ЕОМ.

Перерахованим вище вимогам у великій мірі задовольняє двійкова система числення. Вона потребує тільки два стійких стани для подання цифр **0** і **1**.

Разом з двійковою системою числення, яка знайшла широке застосування у цифрових ЕОМ, для зручності вводу і виводу початкових даних, а також для інших допоміжних операцій застосовують вісімкову і шістнадцяткову системи числення, а для обробки економічної інформації у малих обчислювальних машинах – двійково–десяткову.

Двійкова система числення

Основа цієї позиційної системи числення P = 2, тобто старший розряд числа у два рази більший сусіднього молодшого розряду. В цій системі використовуються тільки дві цифри: 0 і 1. Тому будь – яке число у двійковій системі числення записується як комбінація цифр 0 і 1. Основа двійкової системи числення записується як 10, тобто $2_{(10)} = 10_{(2)}$ і читається: "один, нуль".

Будь – яке число в цій системі числення записується у вигляді (1.2):

$$A_{(2)} = \sum_{i=-m}^{n-1} a_i \cdot 10^i_{(2)}.$$

Приклад.

$$A_{(2)} = 1101{,}101 = 1 \cdot 10^{011}_{(2)} + 1 \cdot 10^{010}_{(2)} + 0 \cdot 10^{001}_{(2)} +$$
$$+ 1 \cdot 10^{000}_{(2)} + 1 \cdot 10^{-001}_{(2)} + 0 \cdot 10^{-010}_{(2)} + 1 \cdot 10^{-011}_{(2)}.$$

Для зручності кількісного аналізу двійкових чисел у наведеному прикладі основа 10 і показники ступеню основи допускається подавати у десятковій системі числення, тоді, виконавши відповідні арифметичні операції, можна отримати кількісний еквівалент двійкового числа у десятковій системі числення. Для нашого випадку маємо:

$$A_{(10)} = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} =$$
$$= 8 + 2 + 0 + 1 + \frac{1}{2} + 0 + \frac{1}{8} = 13\frac{5}{8} = 13{,}625.$$

Відповідно, кількісно

$$1101{,}101_{(2)} = 13{,}625_{(10)}.$$

Основними перевагами двійкової системи числення перед іншими системами є простота і надійність фізичних елементів, які використовуються для подання розрядів двійкового числа у машині (будь – який простий елемент, який має два стійких стани: тригер, феритове кільце, магнітна поверхня та ін.), що створює більші зручності виконання арифметичних операцій.

Одним з суттєвих недоліків двійкової системи числення є необхідність використання спеціальних підпрограм переводу початкових даних, які подані у десятковій системі, у двійкову систему числення, і підпрограм переводу результатів обчислень із двійкової системи у звичайну для людини десяткову систему числення.

Двійкова система числення використовується для подання внутрішньої інформації в цифрових ЕОМ.

Вісімкова система числення

В цій системі числення для запису будь – якого числа застосовується перші вісім цифр десяткової абетки: 0, 1, 2, 3, 4, 5, 6, 7.

У відповідності з виразом (1.3) зображення числа у вісімковій системі числення має вигляд:

$$A_{(8)} = a_{n-1}a_{n-2}\ldots a_к \ldots a_1a_0, a_{-1}a_{-2}\ldots a_{-m}.$$

Приклад.

$$A_{(8)} = 3726{,}145,$$

і читається таким чином: три, сім, два, шість, кома, один, чотири, п’ять. Основою системи є число вісім (P = 8), тобто одиниця старшого розряду у вісім разів більша одиниці сусіднього молодшого розряду.

Використовуючи формулу (1.2), можна знайти кількісне значення будь – якого вісімкового числа у звичайній для нас десятковій системі числення.

Приклад.

$$A = 175{,}36_{(8)} = 1\cdot 8^2 + 7\cdot 8^1 + 5\cdot 8^0 + 3\cdot 8^{-1} + 6\cdot 8^{-2} = 125\frac{15}{32_{(8)}}.$$

$$B = 10_{(8)} = 1\cdot 8^1 + 0\cdot 8^0 = 8_{(10)}.$$

Вісімкова система числення використовується при ручному програмуванні в кодах особливо у спеціалізованих цифрових ЕОМ.

Шістнадцяткова система числення

Основою системи числення є число шістнадцять (P = 16). Це означає, що одиниця старшого розряду у шістнадцять разів більша одиниці сусіднього молодшого розряду. В записі числа, який поданий виразом (1.3), на місці коефіцієнта a_i може знаходитися будь – який з шістнадцяти символів, що застосовуються у шістнадцятковій абетці.

Звичайно ними є перші шістнадцять чисел десяткової абетки. Але для того, щоб не було двозначних символів, в шістнадцятковій абетці використовуються такі шістнадцять знаків:

$$a_i = 0,1,2,3,4,5,6,7,8,9,\bar{0},\bar{1},\bar{2},\bar{3},\bar{4},\bar{5},$$

де знаки з рискою зверху означають: $\bar{0}$ – десять, $\bar{1}$ – одинадцять, $\bar{2}$ – дванадцять, $\bar{3}$ – тринадцять, $\bar{4}$ – чотирнадцять, $\bar{5}$ – п'ятнадцять.

Найчастіше для зображення додаткових символів використовуються початкові букви латинської абетки: **A, B, C, D, E, F**.

Згідно виразу (1.3) зображення числа у шістнадцятковій системі числення матиме такий вигляд:

$$A_{(16)} = A37B,6D5,$$

яке читається таким чином: десять, три, сім, одинадцять, кома, шість, тринадцять, п'ять.

Десяткове подання шістнадцяткового числа можна визначити, використовуючи формулу (1.2).

Приклад.

$$A_{(16)} = A5D,2B,$$

тоді

$$A_{(10)} = 10 \cdot 16^2 + 5 \cdot 16^1 + 13 \cdot 16^0 + 2 \cdot 16^{-1} + 11 \cdot 16^{-2} =$$

$$= 2560 + 80 + 13 + \frac{2}{16} + \frac{11}{256} = 2653\frac{43}{256},$$

тобто кількісно $A5D,2B_{(16)} = 2653\frac{43}{256}$

Шістнадцяткова система числення використовується для подання декотрих специфічних видів інформації в спеціалізованих ЕОМ.

Двійково – десяткова система числення (код Д 1)

Так як у більшості сучасних цифрових ЕОМ використовуються елементи, що придатні для подання цифр двійкової системи числення, а поза машинами завжди використовується десяткова система числення, то значний інтерес має двійкове кодування десяткових цифр. Системи числення, в яких використовується двійкове кодування десяткових цифр називають двійково – кодованими десятковими системами.

Для двійкового кодування десяткових цифр необхідно мати чотири двійкових розряди (двійкова тетрада). Найбільше застосування в теперішній час знайшов код під шифром “8 – 4 – 2 – 1”, де цифри шифру позначають вагу існуючих двійкових цифр у коді. Система числення, що використовує такий код, називається двійково – десятковою системою числення.

Приклад. Число $A_{(10)} = 849, 05$ в двійково – десятковій системі числення матиме такий вигляд:

$$849{,}05_{(10)} = 1000\ \ 0100\ \ 1001,\ \ 0000\ \ 0101_{(2\text{-}10)}.$$

Для зручності читання ми записали тетради з проміжками між ними. Але всі цифри можуть бути поставлені і рядом.

Двійково – десяткова система числення (код “8 – 4 – 2 – 1” або код прямого заміщення) зручна для машинних перетворень з десяткової системи у двійкову і навпаки.

Але ця система незручна для виконання арифметичних операцій над десятковими числами. Це пов’язано з труднощами виявлення перенесення в наступний десяткових розряд (більш старшу тетраду). Крім того, код прямого заміщення характеризується складністю переходу до зворотних і додаткових кодів для десяткових чисел, що полегшують виконання алгебраїчного додавання. Це пояснюється тим, що код прямого заміщення не є тим, що самодоповнюється, тобто інверсія його двійкових цифр не дає коду доповнення десяткової цифри до 9.

Двійково – десяткова система числення з надлишком 3 (код Д 4)

Формування цифр в цій системи числення відбувається методом додавання до десяткової цифри числа 3 і подальшим поданням результату у вигляді двійкових тетрад. Код з надлишком 3 є зручним для виконання арифметичних операцій над числами, так як є тим, що самодоповнюється, тобто додавання до дев’яти отримується шляхом заміни одиниць на нулі і навпаки нулів на одиниці.

Перевагою кода з надлишком 3 є також те, що легко визначається перенесення в старшу двійкову тетраду. Але код з надлишком 3 незручний для перетворення чисел з однієї системи числення в іншу.

Треба відзначити, що разом з двійково – десятковою системою числення з надлишком 3 в обчислювальних машинах широке застосування знаходять і системи числення з надлишком 6. Побудова цих систем відбувається по тій же схемі, що і система числення з надлишком 3.

В табл. 1.2 показано подання десяткових чисел в коді “8 – 4 – 2 – 1”, коді з надлишком 3 і коді з надлишком 6.

Таблиця 1.2

	Код “8 – 4 – 2 – 1”	*Код з надлишком 3*	*Код з надлишком 6*	*Доповнення коду з надлишком 3 до 9*
0	0000	0011	0110	1100
1	0001	0100	0111	1011
2	0010	0101	1000	1010
3	0011	0110	1001	1001
4	0100	0111	1010	1000
5	0101	1000	1011	0111
6	0110	1001	1100	0110
7	0111	1010	1101	0101
8	1000	1011	1110	0100
9	1001	1100	1111	0011

1.1.2. Переведення чисел з одних систем числення в інші

При розв’язанні задач на цифрових ЕОМ початкові дані задаються звичайно в десятковій системі числення і в той же системі, як правило, потрібно отримати і кінцеві результати. Але, якщо ЕОМ працює в будь – якій іншій системі числення, наприклад, у двійковій, то виникає необхідність переводу чисел з однієї системи числення і другу. Переведення чисел відбувається або вручну – на аркуші паперу, або самою ЕОМ – шляхом обчислення спеціальної програми переводу.

Перевести число з одної системи числення в іншу – це означає знайти зображення цифр числа заданої системи в необхідній системі числення [10]. Якщо задана система має основу P_1 , а необхідна P_2, то з певним ступенем точності повинна виконуватися рівність $A_{(P_1)} = B_{(P_2)}$.

Якщо $A_{(P_1)} = \sum_{i=-m}^{n-1} a_i \cdot P_1^i$, а $B_{(P_2)} = \sum_{j=-l}^{K-1} b_j \cdot P_2^j$, то

$$\sum_{i=-m}^{n-1} a_i \cdot P_1^i = \sum_{j=-l}^{K-1} b_j \cdot P_2^j, \qquad (1.4)$$

причому $\{a_i\} = \overline{0, P_1 - 1}; \{b_j\} = \overline{0, P_2 - 1}$.

Далі задачу переводу чисел $A_{(P1)}$ у $B_{(P2)}$ будемо позначати

$$A_{(P_1)} \rightarrow B_{(P_2)}.$$

Існують різноманітні методи переведення чисел із одної системи числення в другу. Розглянемо декотрі з них.

1.1.2.1. Переведення цілих чисел з одної позиційної системи числення в іншу діленням на основу нової системи числення

Нехай задане ціле число в системі числення з основою P_1 – $A_{(P1)}$. Необхідно розв'язати задачу $A_{(P1)} \rightarrow B_{(P2)}$.

Ціле число $A_{(P1)}$ в системі з основою P_2 буде записане у вигляді

$$B_{(P_2)} = b_K \cdot P_2^K + b_{K-1} \cdot P_2^{K-1} + ... + b_1 \cdot P_2^1 + b_0 \cdot P_2^0.$$

Переписавши цей вираз за схемою Горнера, отримаємо

$$B_{(P_2)} = (...((b_K \cdot P_2 + b_{K-1}) \cdot P_2 + b_{K-2}) \cdot P_2 + b_{K-2}) \cdot P_2 + ... + b_1)P_2 + b_0. \qquad (1.5)$$

Праву частину виразу (1.5) розділимо на величину основи P_2. В результаті визначимо перший залишок b_0 і цілу частину $(...((b_K \cdot P_2 + b_{K-1}) \cdot P_2 + ... + b_1)$. Розділивши цілу частину на P_2, знайдемо другий залишок b_1. Повторюючи процес ділення К + 1 раз, отримаємо останній цілий залишок b_K, який за умовою, менше основи системи P_2 і є старшою цифрою числа, поданого в системі з основою P_2.

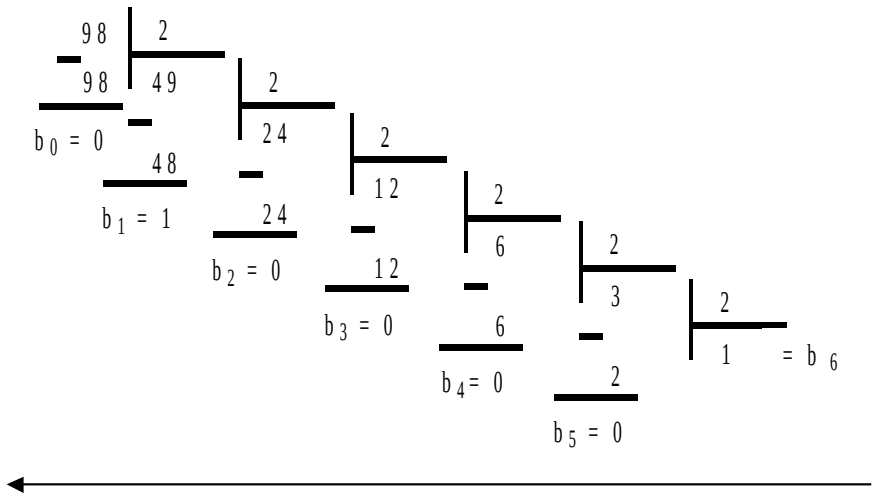
ПРАВИЛО. Для переведення цілого числа з одної системи числення в іншу необхідно послідовно ділити це число і проміжкові частки на основу нової системи числення до тих пір, доки проміжкова частка не буде менше основи нової системи числення. Остання частка і залишки у порядку зворотному їх отриманню є зображеннями цифр числа в новій системі числення [13].

Так як всі операції виконуються в старій, тобто в P_1-річній системі числення, тому коефіцієнти, які шукаємо будуть отримані у цій же системі числення. Для кінцевого запису числа $A_{(P1)}$ в P_2-річній системі числення необхідно кожний з отриманих коефіцієнтів b_i записати одною P_2-річною цифрою.

Розглянутий метод переведення, як правило, застосовується для переводу чисел з десяткової системи числення в інші системи числення.

Приклад. Перевести десяткове число $A_{(10)} = 98$ у двійкову систему числення ($P_2 = 2$).

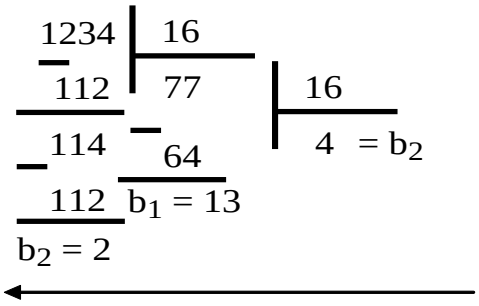
Розв'язання.



Відповідь: $B_{(2)}$=1100010.

Приклад. Перевести число $A_{(10)}$=1234 в шістнадцяткову систему числення.

Розв'язання.



Таким чином, $b_2=4_{(10)}$, $b_1=13_{(10)}$, $b_0=2_{(10)}$.

Для закінчення запису числа $A_{(10)}$ в шістнадцятковій системі треба кожний з коефіцієнтів записати однією шістнадцятковою цифрою.

Відповідь: $B_{(16)}$=4D2.

1.1.2.2. Переведення дробових чисел з однієї системи числення в іншу множенням на основу нової системи числення

Нехай початкове число, яке записане в системі числення з основою P_1 має вигляд

$$A_{(P_1)} = a_{-1}P_1^{-1} + a_{-2}P_1^{-2} + ... + a_{-m}P_1^{-m}.$$

Тоді в новій системі з основою P_2 це число буде зображено як $0, b_{-1}, b_{-2}...b_{-l}$, або

$$B_{(P_2)} = b_{-1}P_2^{-1} + b_{-2}P_2^{-2} + ... + b_{-l}P_2^{-l}.$$

Якщо переписати цей вираз за схемою Горнера, то отримаємо

$$B_{(P_2)} = P_2^{-1}(b_{-1} + P_2^{-1}(b_{-2} + ... + P_2^{-1}(b_{-(l-1)} + P_2^{-1}b_{-l}))...) \quad (1.6)$$

Якщо праву частину виразу (1.6) помножити на величину основи P_2, то знайдемо новий десятковий дріб, в цілій частині якої буде число b_{-1}.

Потім помножити дробову частину яка залишилася на величину основи P_2, отримаємо дріб, в цілій частині якої буде b_{-2}.

Повторюючи процес множення l раз, знайдемо все l цифр числа в новій системі числення. При цьому всі дії повинні виконуватися за правилами P_1-річної арифметики і, відповідно, в цілій частині отриманих дробів будуть проявлятися еквіваленти цифр нової системи числення, які записані в початковій системі числення.

При переведенні десяткових дробів з однієї системи числення в іншу можна отримати дроби у вигляді нескінченності, або рядка, який розходиться. Процес переведення можна закінчити, якщо появиться дробова частина, яка має у всіх розрядах нулі, або буде досягнута задана точність переведення (отримано необхідна кількість розрядів результату).

Викладене означає, що при переведенні дробів необхідно вказувати кількість розрядів числа в новій системі числення. На основі викладеного можна записати загальне правило, яке дозволить переводити десяткові дроби з одної позиційної системи в іншу.

ПРАВИЛО. Для переведення десяткового дробу з однієї позиційної системи числення в іншу його необхідно послідовно множити на основу нової системи числення до того часу, поки в новому дробі не буде потрібної кількості цифр, яка визначається потрібною точністю представлення дробу.

Десятковий дріб в новій системі числення записується з цілих частин добутків, які отримані при послідовному множенні, причому перша ціла частина буде старшою цифрою нового дробу [13].

Приклад. Перевести десятковий дріб $A_{(10)} = 0{,}625$ із десяткової в двійкову систему числення ($P_2=2$) з точністю до четвертого знаку.

Розв'язання:

0,	625
x	2
b_{-1}=1,	250
x	2
b_{-2}=0,	500
x	2
b_{-3}=1,	000
x	2
b_{-4}=0	000

Відповідь: $B_{(2)} = 0{,}1010$.

Приклад. Перевести десяткове число $A_{(10)}=0{,}12$ у вісімкову систему числення з точністю до шостого знака.

Розв'язання:

0,	12
x	8
b_{-1}=0,	96
x	8
b_{-2}=7,	68
x	8
b_{-3}=5,	44
x	8
b_{-4}=3,	52
x	8
b_{-5}=4,	16
x	8
b_{-6}=1,	28

Відповідь: $B_{(8)} = 0{,}075341$.

Треба відзначити, що якщо основа нової системи числення $P_2<P_1$, то коефіцієнти (цілі частини добутку є цифрами P_2-річної системи числення, як це було в приведених вище прикладах. Якщо $P_2>P_1$, то коефіцієнти b_i представляють собою числа в P_1-річній системі числення, які необхідно замінити цифрами P_2-річної системи.

Приклад. Перевести десяткове число $A_{(10)}$=0,87 у шістнадцяткову систему числення.

Розв'язання: Так як основа нової системи числення $P_2>P_1$, то переведення треба робити в такій послідовності:

1. Основа нової системи $P_2=10_{(16)}$ подається в початковій системі числення P_1=10:

$$10_{(16)} = 16_{(10)};$$

2. Виконується послідовне множення дробової частини на основу $P_2=16_{(10)}$

0,	87
x	16
b_{-1}=13,	22
x	16
b_{-2}=14,	72
x	16
b_{-3}=11,	52
x	16
b_{-4}=8,	32

3. Цілі частини (коефіцієнти b_{-i}) переводяться у шістнадцяткову систему числення

$$13_{(10)} = D_{(16)};\quad 14_{(10)} = E_{(16)};\quad 11_{(10)} = B_{(16)};\quad 8_{(10)} = 8_{(16)}.$$

Відповідь: $B_{(16)}$ = 0,DEB8 переведення визначено четвертим знаком.

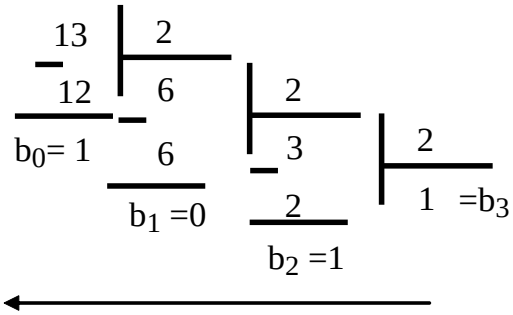
Треба відзначити універсальність цього методу переведення, але на практиці він частіше всього використовується для переведення чисел з десяткової системи числення в інші системи, так як арифметичні дії в цьому випадку робляться у звичній для нас десятковій системі числення.

Переведення звичайних дробів в двійкову систему числення робиться за розглянутим вище правилом після переведення їх в десяткові дроби [10].

Для переведення в двійкову систему числення звичайних дробів, знаменники яких є ціла ступінь двійки (основа двійкової системи числення), потрібно перевести їх чисельник як ціле число, відділивши комою, починаючи з молодшого розряду, кількість цифр, що дорівнює ступеню двійки в знаменнику дробу.

Приклад. Перевести число $A_{(10)} = \frac{13}{16}$ в двійкову систему числення.

Розв'язання: переводимо чисельник дробі як ціле число, методом ділення на основу нової системи числення



Так як число $16 = 2^4$, то, відокремивши комою чотири розряди, починаючи з молодшого отримаємо:

$$\frac{13}{16_{(10)}} = 0{,}1101_{(2)}.$$

Відповідь: $B_{(2)} = 0{,}1101$.

Для переведення змішаного дробу з однієї системи числення в іншу необхідно окремо переводити цілу і дробову частини числа відповідно до методу ділення і методу множення на основу нової системи числення.

1.1.2.3. Метод безпосереднього заміщення

Відповідно до виразу (1.2) числа в різних системах числення можна подавати таким чином:

$$A_{(P1)} = a_{n-1}P_1^{n-1} + a_{n-2}P_1^{n-2} + \cdots + a_1P_1^1 + a_0P_1^0 + a_{-1}P_1^{-1} + \cdots + a_{-m}P_1^{-m} =$$
$$= b_{k-1}P_2^{k-1} + b_{k-2}P_2^{k-2} + \cdots + b_1P_2^1 + b_0P_2^0 + b_{-1}P_2^{-1} + \cdots + b_{-l}P_2^{-l} = B_{(P2)}$$

Отже, у загальному вигляді задачу переведення числа з системи числення з основою P_1 в систему числення з основою P_2 можна подати як задачу визначення коефіцієнтів b_j нового ряду, що зображує числа в системі з основою P_2.

Правило переводу чисел цим методом полягають у наступному [10]:

1. Задане число в системі числення з основою P_1 у відповідності з виразом (1.2) подаються у вигляді зваженої суми.

2. Всі цифри a_i і основа P_1 в правій частині виразу (1.2) записуються (заміщуються) в системі числення з основою P_2. Якщо $P_2 > P_1$, то зображення цифр в P_2-річній системі числення співпадає з їх зображенням в P_1-річній системі.

3. Виконуються всі арифметичні операції у відповідності з виразом (1.2) в системі числення з основою P_2.

Приклад. Перевести десяткове число $A_{(10)}$ = 35, 25 у двійкову систему числення.

Розв'язання. 1. Подамо число $A_{(10)}$ у відповідності з виразом (1.2):

$$35{,}25_{(10)} = 3 \cdot 10^1 + 5 \cdot 10^0 + 2 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

2. Замінивши в правій частині усі десяткові цифри двійковими, отримаємо:

$$B_{(2)} = 111010 + 101 + \frac{10}{1010} + \frac{101}{1010 \cdot 1010}.$$

3. Виконавши усі арифметичні операції у двійковій системі числення, знайдемо двійковий запис числа

$$B_{(2)} = 100011{,}01.$$

Відповідь. $B_{(2)}$ = 100011,01

Приклад. Перевести вісімкове число $A_{(8)}$ = 623,2 у десяткову систему числення.

Розв'язання.

$$A_{(8)} = 6 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0 + 2 \cdot 8^{-1} = 403{,}25_{(10)}.$$

Відповідь. $B_{(10)}$ = 403,25.

Приклад. Перевести двійкове число $A_{(2)}$ = 11001,011 у десяткову систему числення.

Розв'язання.

$$A_{(2)} = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 25{,}375_{(10)}.$$

Відповідь. $B_{(10)}$ = 25,375.

Метод безпосереднього заміщення інколи називають методом додавання з урахуванням ваги розрядів. Метод безпосереднього заміщення зручний при переведенні чисел з будь-якої системи числення в ту систему, в якій найпростіше виконуються арифметичні операції.

Так при ручному розрахунку цим методом зручно переводити числа в десяткову систему числення, а в цифрових ЕОМ, що використовують для виконання арифметичних операцій двійкову систему, метод безпосереднього заміщення застосовується для переведення чисел у двійкову систему числення.

1.1.2.4. Переведення чисел з вісімкової і шістнадцяткової систем числення у двійкову та навпаки

Якщо основа одної системи числення є цілим степенем двійки, тобто $P = 2^K$, то при цьому значно спрощується перетворення інформації з системи числення з основою $P = 2^K$ в двійкову систему і навпаки. Перетворення фактично зводиться до того, що символи вхідної інформації, які задані в системі з основою $P = 2^K$, замінюються відповідними двійковими еквівалентами [10].

Зворотне перетворення із двійкової системи в систему з основою $P = 2^K$ зводиться до того, що двійковий код розбивається на групи по К - двійкових розрядів в кожній. Ці групи замінюються відповідними символами вхідної системи числення.

Розглянемо перехід між вісімковою і двійковою системами числення.

Нехай задано число А у вісімковій системі числення:

$$A_{(8)} = a_{n-1}a_{n-2}\cdots a_1a_0 = a_{n-1}\cdot 8^{n-1} + a_{n-2}\cdot 8^{n-2} + \cdots + a_1\cdot 8^1 + a_0\cdot 8^0 \quad (1.7)$$

де $a_{n-1}, a_{n-2}, \ldots, a_1, a_0$ – цифри вісімкової системи числення.

Припустимо, що знайдено подання цього ж числа у двійковій системі числення.

$$B_{(2)} = b_{k-1}b_{k-2}\cdots b_1b_0 = b_{k-1}\cdot 2^{k-1} + b_{k-2}\cdot 2^{k-2} + \cdots + b_1\cdot 2^1 + b_0\cdot 2^0 \quad (1.8)$$

де $b_{k-1,}b_{k-2},\cdots,b_1,b_0$– цифри двійкової системи числення.

Тому що вирази(1.7) і (1.8) подають однакове число, то

$$\begin{aligned} &a_{n-1}\cdot 8^{n-1} + a_{n-2}\cdot 8^{n-2} + \cdots + a_1\cdot 8^1 + a_0 = \\ &= b_{k-1}\cdot 2^{k-1} + b_{k-2}\cdot 2^{k-2} + \cdots + b_1\cdot 2^1 + b_0 \end{aligned} \quad (1.9)$$

Розділивши ліву і праву частини рівняння (1.9) на вісім, отримаємо однакові частки:

$$a_{n-1}8^{n-2} + a_{n-2}8^{n-3} + \cdots + a_1 = b_{k-1}2^{k-4} + b_{k-2}2^{k-5} + \cdots + b_5 2^2 + b_4 2 + b_3 \quad (1.10)$$

і однакові остачі :

$$a_0 = b_2\cdot 2^2 + b_1\cdot 2^1 + b_0 = b_2b_1b_0 \quad (1.11)$$

З виразу (1.11) виходить, що молодша вісімкова цифра a_0 виражається трирозрядним двійковим числом (триадою) b_2, b_1, b_0. Якщо розділити на 8 ліву і праву частини рівняння (1.10), то отримаємо подання наступної вісімкової цифри у вигляді трирозрядного двійкового числа:

$$a_1 = b_5 \cdot 2^2 + b_4 \cdot 2^1 + b_3 = b_5 b_4 b_3$$

Аналогічно можна отримати зображення інших вісімкових цифр у вигляді трирозрядних двійкових чисел (двійкових триад).

При переведенні шістнадцяткового числа у двійкову систему числення всі наведені вище міркування справедливі. Але потрібно мати на увазі, що кожній шістнадцятковій цифрі відповідає чотирирозрядне двійкове число (двійкова тетрада), тобто

$$a_0 = b_3 b_2 b_1 b_0$$
$$a_1 = b_7 b_6 b_5 b_4$$

і так далі.

ПРАВИЛО. Для переведення вісімкового (шістнадцяткового) числа у двійкову систему числення достатньо кожну вісімкову (шістнадцяткову) цифру замінити рівною їй двійковою триадою (тетрадою) [10].

Приклад. Перевести вісімкове число $A_{(8)}$ = 765,163 у двійкову систему числення.

Розв'язання.

$$B_{(2)} = \underbrace{111}_{7}\ \underbrace{110}_{6}\ \underbrace{101}_{5},\ \underbrace{001}_{1}\ \underbrace{110}_{6}\ \underbrace{011}_{3}$$

Відповідь. $B_{(2)}$ = 111110101,001110011.

Приклад. Перевести шістнадцяткове число $A_{(16)}$ = 3B7E,5A6 у двійкову систему числення.

Розв'язання.

$$B_{(2)} = \underbrace{0011}_{3}\ \underbrace{1011}_{B}\ \underbrace{0111}_{7}\ \underbrace{1110}_{E},\ \underbrace{0101}_{5}\ \underbrace{1010}_{A}\ \underbrace{0110}_{6}$$

Відповідь. $B_{(2)}$ = 11101101111110, 01011010011

ПРАВИЛО. Для переведення двійкового числа у вісімкову (шістнадцяткову) систему числення достатньо розбити його направо та наліво від коми на триади (тетради) і замінити кожну триаду (тетраду) відповідною їй вісімковою (шістнадцятковою) цифрою. Якщо при розбиванні крайні триади (тетради) виявляться неповними, то їх потрібно доповнити нулями [10].

Приклад. Перевести двійкове число $A_{(2)}$ = 11010011101,11001011 у вісімкову систему числення.

Розв'язання.

$$A_{(2)} = \underbrace{011}_{3}\ \underbrace{010}_{2}\ \underbrace{011}_{3}\ \underbrace{101}_{5},\ \underbrace{110}_{6}\ \underbrace{010}_{2}\ \underbrace{110}_{6}$$

Відповідь. $B_{(2)} = 3235{,}626$.

Приклад. Перевести двійкове число $A_{(2)} = 11010011101{,}11001011$ у шістнадцяткову систему числення.

Розв'язання.

$$A_{(2)} = \underbrace{0110}_{6}\ \underbrace{1001}_{9}\ \underbrace{1101}_{D},\ \underbrace{1100}_{C}\ \underbrace{1011}_{B}$$

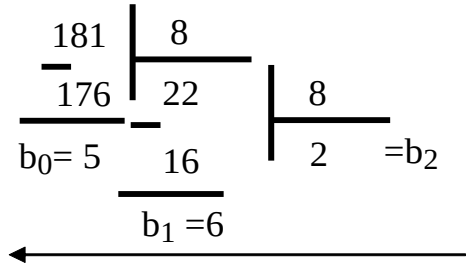
Відповідь. $B_{(2)} = 69D{,}CB$

Простоту переходу від вісімкової (шістнадцяткової) системи числення до двійкової можна використовувати для скорочення кількості операцій при ручному переведенні чисел у двійкову систему числення. При цьому вісімкова (шістнадцяткова) система використовується як проміжкові.

Приклад. Перевести десяткове число $A_{(10)} = 181{,}71875$ у двійкову систему числення.

Розв'язання. Для цілої частини:

переведення у вісімкову систему числення:



$$181_{(10)} = 265_{(8)}$$

переведення у двійкову систему числення:

$$B_{1(2)} = \underbrace{010}_{2}\ \underbrace{110}_{6}\ \underbrace{101}_{5} = 10110101$$

Для дробової частини:

переведення у вісімкову систему счислення:

$$0{,}71875$$
$$b_{-1} = \frac{\times 8}{5{,}75000}$$

$$b_{-2} = \frac{\times 8}{6{,}00000}$$

$$0{,}71875_{(10)} = 0{,}56_{(8)}$$

переведення у двійкову систему числення:

$$B_{2(2)} = 0, \underbrace{101}_{5}\ \underbrace{110}_{6} = 0{,}10111$$

Остаточно $B_{(2)} = B_{1(2)} + B_{2(2)} = 10110101,\ 10111$

Відповідь. $B_{(2)} = 10110101,\ 10111$.

1.1.3. Арифметичні операції в двійковій системі числення

Всі арифметичні операції в двійковій системі числення проводяться відповідно до відомих правил виконання арифметичних операцій в загальноприйнятій десятковій системі числення, але при цьому використовуються таблиці додавання і множення, складені для двійкової системи числення.

При складанні таблиці додавання в будь-якій системі числення можна користуватися таким правилом: якщо при підсумовуванні двох цифр a_i і a_j

$$a_i + a_j < P, \quad \text{то} \quad a_i + a_j = a_k,$$

де a_k – цифра даної системи числення; P – основа системи числення, якщо $a_i + a_j \geq P,$ то $a_k = a_i + a_j - P$ і з'являється одиниця перенесення в наступний старший розряд.

Правило складання таблиці додавання можна записати в наступному вигляді:

$$a_i + a_j = \begin{cases} a_k, & \text{якщо} \quad a_i + a_j < P; \\ 1a_k, & \text{якщо} \quad a_i + a_j \geq P, \end{cases}$$

де $a_k = a_i + a_j - P.$

Слід пам'ятати, що в двійковій системі числення кожен старший розряд містить дві одиниці сусіднього молодшого розряду.

Таблиця двійкового додавання	Таблиця двійкового віднімання	Таблиця двійкового множення
$0 + 0 = 0$	$0 - 0 = 0$	$0 \times 0 = 0$
$0 + 1 = 1$	$1 - 0 = 1$	$0 \times 1 = 0$
$1 + 0 = 1$	$1 - 1 = 0$	$1 \times 0 = 0$
$1 + 1 = 10$	$10 - 1 = 1$	$1 \times 1 = 1$
↑	↑	
одиниця перенесення в сусідній старший розряд	одиниця позички з сусіднього старшого розряду	

Додавання чисел в двійковій системі можна виконувати стовпчиком, починаючи з молодшого розряду. У кожному розряді у відповідності з правилами, зазначеними таблицею двійкового додавання, проводиться складання двох чисел доданків і одиниці перенесення з сусіднього молодшого розряду (якщо вона є). В результаті виходить цифра відповідного розряду суми, і, можливо, одиниця перенесення в сусідній старший розряд.

Приклад. Скласти число $A_{(2)} = 1100101$ і число $B_{(2)} = 1010011$ в двійковій системі числення.

Розв'язання:

одиниці перенесення

$$\begin{array}{r} \overset{1}{}\overset{1\,1\,1}{} \\ A = 1\,1\,0\,0\,1\,0\,1 \\ + \\ B = 1\,0\,1\,0\,0\,1\,1 \\ \hline A + B = 1\,0\,1\,1\,1\,0\,0\,0 \end{array}$$

Відповідь: $A + B = 10111000_{(2)}$.

При відніманні чисел в двійковій системі числення здійснюється зайняття одиниці з сусіднього старшого розряду кожен раз, коли цифра в розряді від'ємника більше цифри в тому ж розряді зменшуваного. Ця займана одиниця дорівнює двом одиницям даного розряду.

Приклад. Провести операцію віднімання числа $B_{(2)} = 1010101$ з числа $A_{(2)} = 1101011$ в двійковій системі числення.

Розв'язання:

одиниці позички

$$\begin{array}{r} \overset{1\ \ 1}{} \\ A = 1\,1\,0\,1\,0\,1\,1 \\ - \\ B = 1\,0\,1\,0\,1\,0\,1 \\ \hline A - B = 0\,0\,1\,0\,1\,1\,0 \end{array}$$

Відповідь: $A - B = 10110_{(2)}$.

При множенні двійкових чисел так само, як і при множенні десяткових, спочатку отримують часткові добутки, після чого знаходять їх суму. Відповідно до таблиці двійкового множення кожний частковий добуток дорівнює нулю, якщо у відповідному розряді множника стоїть нуль, або одне множене, зсунуте на відповідне число розрядів вліво, якщо в розряді множника стоїть одиниця.

Таким чином, операція множення багаторозрядних двійкових чисел зводиться до операцій зсуву і додавання. Комою відділяється кількість розрядів добутку, яке дорівнює сумі розрядів дробових частин співмножників.

Ділення чисел в двійковій системі числення проводиться аналогічно діленню десяткових чисел.

Операції множення та ділення виконуються в комп'ютері за спеціальними алгоритмами, які будуть розглянуті далі.

Завдяки простоті правил двійковій арифметики застосування в комп'ютерах двійкової системи числення дозволяє істотно спростити схеми арифметичних пристроїв.

1.2. ПОДАННЯ ЧИСЕЛ В ЗАСОБАХ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Подання чисел в обчислювальній машині зводиться до подання чисел кожного розряду окремо.

Для зберігання одного розряду коду двійкового числа використовується один фізичний елемент, який має два стійких і чітко виражених станів і володіє великою швидкістю переходу з одного стану в інший. Набір відповідної кількості таких елементів служить для подання багаторозрядного двійкового коду числа. У цифрових ЕОМ розрядність числа завжди скінченна.

Сукупність двійкових розрядів, призначених для зберігання і обробки чисел, являє розрядну сітку машини. Розрядна сітка машини визначає формати чисел, якими можна оперувати при обробці інформації. У машині не може бути представлено число, що містить більшу кількість двійкових розрядів, ніж їх є в розрядної сітці машини.

У цифрових ЕОМ використовуються дві форми подання чисел: ***природна*** (подання чисел з фіксованою комою) і ***нормальна*** (подання чисел з плаваючою комою) [9].

1.2.1. Подання чисел у формі з фіксованою комою

Подання числа X у формі з *фіксованою комою* (ФК) включає знак числа і його модуль в q-ічному коді. Тут q – *основа системи числення* або *база*. Для сучасних ОМ характерна двійкова система (q = 2), але іноді використо-

вуються також вісімкова (q = 8) або шістнадцяткова (q = 16) системи числення. Знак позитивного числа кодується двійковою цифрою 0, а знак негативного числа – цифрою 1.

Числам з ФК відповідає запис вигляду $X = \pm a_{n-1}\dots a_1 a_0 a_{-1} a_{-2}\dots a_{-r}$. Розряд коду числа, в якому розміщується знак, називається *знаковим розрядом коду*. Розряди, де розташовуються значущі цифри числа, називаються *цифровими розрядами коду*. Знаковий розряд розміщується лівіше старшого цифрового розряду. Положення коми однакове для всіх чисел і в процесі вирішення завдань не міняється. Хоча кома і фіксується, в коді числа вона ніяк не виділяється, а тільки мається на увазі. У загальному випадку розрядна сітка ОМ для розміщення чисел у формі з ФК має вигляд, показаний на рис. 2.1, де ***n*** розрядів використовуються для запису цілої частини числа і ***r*** розрядів – для дробової частини.

Якщо число є змішаним (містить цілу і дробову частини), воно обробляються як ціле, хоча і не є таким (в цьому випадку застосовують термін *масштабоване ціле)*. Обробка змішаних чисел у ОМ зустрічається украй рідко. Як правило, використовуються ОМ з дробовою (n = 0) або цілочисельною (r = 0) арифметикою [9].

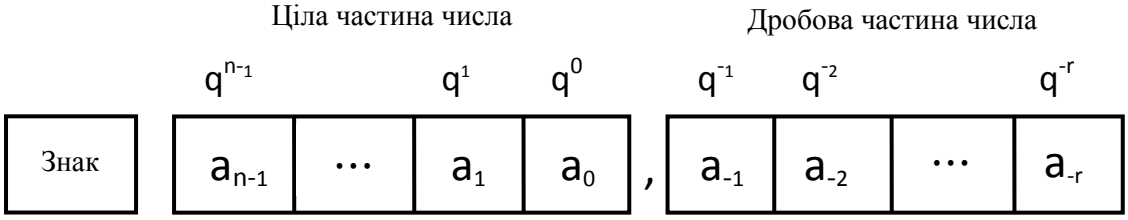


Рис. 2.1. Формат подання чисел з фіксованою комою

Під час фіксації коми перед старшим цифровим розрядом можуть бути подані тільки правильні дроби. Під час фіксації коми після молодшого розряду подаються лише цілі числа (рис. 2.2).

Це найбільш поширений спосіб, тому надалі поняття ФК зв’язуватиметься виключно з цілими числами, а операції над числами у формі з ФК характеризуватимуться як цілочисельні. Тут можливі числа із знаком і без знака [9].

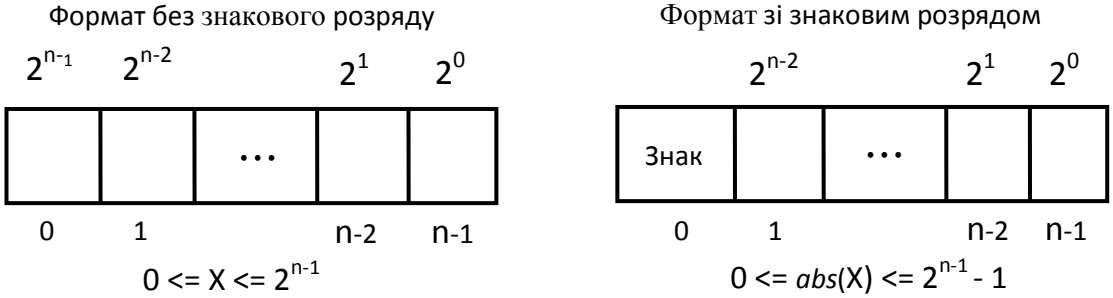


Рис. 2.2. Подання цілих чисел у форматі з ФК

Подання чисел у форматі з ФК спрощує апаратурну реалізацію ОМ і скорочує час виконання машинних операцій, проте під час вирішення завдань необхідно постійно стежити за тим, щоб усі початкові дані, проміжні і остаточні результати не виходили за допустимий діапазон формату, інакше можливе переповнення розрядної сітки і результат обчислень буде невірним.

Вихід числа за межі розрядної сітки вліво називається переповненням. В цьому випадку спотворюється не тільки величина числа, яке бере участь в обчисленнях, але і його знак, що абсолютно неприпустимо.

Для подання чисел, що не укладаються в діапазон чисел представимих в цифровий ЕОМ, програміст повинен вводити *масштабні множники*, тобто замінювати справжні величини, які беруть участь у вирішенні задачі, їх добутками на коефіцієнти, підібрані так, щоб не вийти за межі діапазону чисел, представлених в машині .

Максимальна кількість різних чисел ($N_{макс.фз.}$), яке можна записати в розрядній сітці машини з кроком $h_{фз} = 2^{-m}$, дорівнює

$$N_{макс.фз.} = 2^m,$$

де m – кількість розрядів, виділених для запису в розрядній сітці цифрової частини числа.

Розглянемо точність представлення чисел в машині з фіксованою комою.

Оцінимо максимальну похибку уявлення чисел [11].

Абсолютною похибкою уявлення чисел назвемо різницю $(A^1 - A)$ між точним значенням A^1 деякого числа і тим значенням A, яке може бути введено в конкретну обчислювальну машину.

Зауважимо, що мінімальне число, яке може бути представлено в обмеженій розрядній сітці машини, дорівнює

$$A_{мін} = 2^{-m}.$$

Абсолютна похибка уявлення чисел в цьому випадку може з'явитися тільки через похибку округлення, яка не перевищує половини ціни молодшого розряду:

$$|\Delta A_{фз}| = \frac{\mathbf{2^{-m}}}{\mathbf{2}} = 2^{\mathbf{-m-1}}. \qquad (2.1)$$

Ця помилка є постійною для всього діапазону представимих чисел.

Однак ***відносна похибка*** при поданні різних по модулю чисел, що дорівнює відношенню абсолютної похибки до значення числа, не буде постійною. Визначимо діапазон зміни цієї помилки:

$$\delta_{\text{мін.фз.}} = \frac{\Delta A_{фз}}{A_{макс}} = \frac{2^{-m-1}}{1-2^{-m}} \approx 2^{-m-1}$$

$$\delta_{\text{макс.фз.}} = \frac{\Delta A_{фз}}{A_{мін}} = \frac{2^{-m-1}}{2^{-m}} = 2^{-1} = -\frac{1}{2} \qquad (2.2)$$

З виразів (2.2) видно, що *відносна похибка* уявлення чисел в цифровий ЕОМ з фіксованою комою змінюється в широких межах і вона тим більше, чим менше значення числа і при $A = A_{мін}$ досягає свого максимального значення (рис. 2.5). Ця обставина може призвести до отримання недостовірних результатів при виконанні операцій над малими за величиною числами. Зазначений недолік усувається правильним масштабуванням вихідних даних.

1.2.2. Подання чисел у формі з плаваючою комою

Від недоліків ФК в значній мірі вільна форма подання чисел з плаваючою комою (ПК), відома також під назвами нормальної або напівлогарифмічної форми. У даному варіанті кожне число розбивається на дві групи цифр. Перша група цифр називається мантисою, друга – порядком. Число подається у вигляді добутку $X = \pm m\, q^{\pm p}$,

де m – мантиса числа X, p – порядок числа, q – основа системи числення.

Для подання числа у формі з ПК потрібно задати знаки мантиси і порядку, їх модулі в q-ічному коді, а також основу системи числення (рис. 2.3). Нормальна форма неоднозначна, оскільки взаємна зміна m і p приводить до «плавання» коми, чим і обумовлена назва цієї форми [9].

Рис. 2.3. Форма подання чисел з плаваючою комою

Діапазон і точність подання чисел з ПК залежать від числа розрядів, що відводяться під порядок і мантису.

У більшості обчислювальних машин для спрощення операцій над порядками останні приводять до цілих позитивних чисел, застосовуючи так званий зміщений порядок. Для цього до дійсного порядку додається ціле позитивне число – зсув (рис. 2.4). Наприклад, у системі із зсувом 128 порядок –3 подається як 125 (–3 + 128).

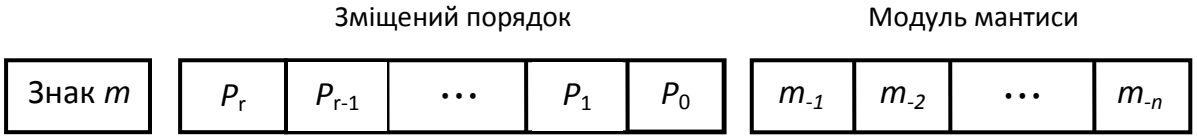


Рис. 2.4. Формат числа з ПК із зміщеним порядком

Звичайний зсув вибирається рівним половині уявного діапазону порядків. Відзначимо, що зміщений порядок займає всі біти поля порядку, у тому числі і той, який раніше використовувався для запису знака порядку.

Мантиса в числах з ПЗ зазвичай подається в *нормалізованій формі*. Це означає, що на мантису накладаються такі умови, щоб вона по модулю була менше одиниці ($|q| < 1$), а перша цифра після крапки відрізнялася від нуля. Отримана таким чином мантиса називається *нормалізованою*. Для двійкової системи числення можна записати: $X = q\,2^{p}$, $(1 > |q| = S)$.

Якщо перші i цифр мантиси рівні нулю, для нормалізації її потрібно зсунути відносно коми на i розрядів вліво з одночасним зменшенням порядку на i одиниць.

Визначимо діапазон представлення чисел в машині з плаваючою комою.

Нехай для зберігання мантиси числа відведено m – розрядів, а для відображення порядку числа – k розрядів сітки машини.

Тоді найменше нормалізоване число, яке можна представити в розрядній сітці дорівнює

$$A_{мін.пз} = \underbrace{0,10\ldots00}_{m} \cdot 2^{\overbrace{-11\ldots1}^{k}} = 2^{-1} \cdot 2^{-(2^{k}-1)} = 2^{-2^{k}}.$$

Максимальне позитивне число матиме вид:

$$A_{макс.пз} = \underbrace{0,11\ldots11}_{m} \cdot 2^{\overbrace{11\ldots1}^{k}} = (1 - 2^{-m}) \cdot 2^{(2^{k}-1)} \approx 2^{2^{k}}.$$

Таким чином, діапазон представлення чисел лежить в межах

$$2^{-2^{k}} \le A \le 2^{2^{k}}. \qquad (2.3)$$

З виразу (2.3) видно, що практично діапазон чисел залежить тільки від кількості розрядів порядку [11].

Визначимо загальну кількість нормалізованих чисел, яку можна записати в розрядну сітку машини з плаваючою комою.

Так як в мантисі нормалізованого двійкового числа в старшому розряді завжди стоїть одиниця, то кількість різних мантис, які можуть бути записані в $m - 1$ розрядах, дорівнюватиме 2^{m-1}.

Так як максимальна кількість позитивних порядків, включаючи нульовий, дорівнює 2^k, а кількість негативних порядків 2^{k-1}, то загальна кількість різних нормалізованих чисел одного знака дорівнює:

$$N_{макс.пз} = (2^{k+1} - 1) \cdot 2^{m-1}. \qquad (2.4)$$

Знайдемо крок чисел в машинах з плаваючою комою. Для цього зауважимо, що мантиси двох сусідніх чисел A_1 і A_2 відрізняються на одиницю молодшого розряду 2^{-m}, тобто

$$A_1 = M \cdot 2^k; \qquad A_2 = (M + 2^{-m}) \cdot 2^k.$$

Звідси

$$h_{пз} = 2^{-m} \cdot 2^k. \qquad (2.5)$$

Таким чином, в машинах з плаваючою комою числа розподілені по діапазону нерівномірно. Крок чисел залежить від їх порядку.

Проаналізуємо точність представлення чисел в розрядної сітці машини з плаваючою комою. *Абсолютна максимальна похибка* залежить від порядку числа k і визначається на підставі співвідношення (2.5) виразом

$$A_{макс.пз} = \frac{h_{пз}}{2} = \frac{2^{-m+k}}{2} = 2^{-m+k-1}. \qquad (2.6)$$

Відносна похибка

$$\delta_{пз.} = \frac{\Delta A_{макс.пз}}{A} = \frac{2^{-m+k-1}}{M \cdot 2^k} = \frac{2^{-m-1}}{M} \qquad (2.7)$$

досягає максимуму при $M = M_{мін} = 2^{-1}$ і мінімуму при $M = M_{макс} = 1 - 2^{-m} \approx 1$

$$\delta_{макс.пз} = \frac{2^{-m-1}}{2^{-1}} = 2^{-m}, \qquad (2.8)$$

$$\delta_{мін.пз} = \frac{2^{-m-1}}{1-2^{-m}} \approx 2^{-m-1}, \qquad (2.9)$$

Так як зазвичай $m » 1$, то можна вважати, що відносна похибка подання чисел в машинах з плаваючою комою приблизно постійна у всьому діапазоні чисел.

Характер розподілу *відносних похибок* за діапазоном для природної і нормальної форм представлення чисел в обчислювальних машинах показаний на рис. 2.5.

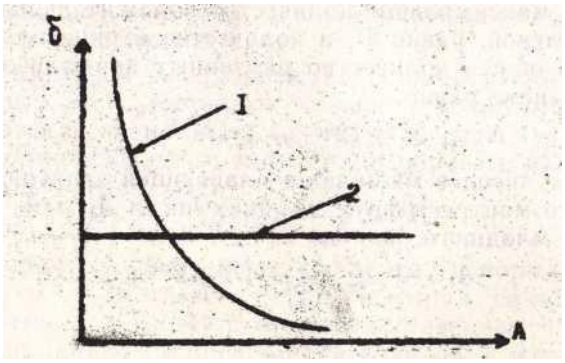


Рис. 2.5. Розподіл відносних похибок для природної (1)
і нормальної (2) форм представлення чисел

З рисунка видно, що у цифрових ЕОМ з фіксованою комою спостерігається нерівномірність розподілу помилок, яка компенсується правильним масштабуванням, а у машин з плаваючою комою помилки представлення чисел розподілені рівномірно.

На закінчення зробимо порівняння машин з фіксованою і плаваючою комою [11].

Машини з фіксованою комою мають більшу швидкодію, тому що не вимагають вирівнювання порядків чисел при обчисленні арифметичних операцій.

Машини з фіксованою комою мають більш прості арифметико-логічні пристрої в порівнянні з машинами з плаваючою комою, так як останні вимагають обладнання як для виконання операцій над мантиси, так і над порядками.

У машинах з фіксованою комою складно проводити масштабування.

У машинах з фіксованою комою низька точність при виконанні арифметичних операцій над малими за абсолютною величиною числами.

Подання чисел в нормалізованому вигляді в машинах з плаваючою комою підвищує точність обчислень. При однаковій довжині розрядної сітки діапазон представимих чисел в машинах з фіксованою комою значно вужче діапазону представимих чисел в машинах з плаваючою комою.

З огляду на те, що природна і нормальна форми подання чисел мають свої переваги, і недоліки, останнім часом створені цифрові ЕОМ, які можуть працювати в режимах, з плаваючою і фіксованою комою.

1.3. СПОСОБИ КОДУВАННЯ ДВІЙКОВИХ ЧИСЕЛ

Під час виконання арифметичних операцій в арифметико-логічних пристроях необхідно враховувати можливість обробки як додатних, так і від’ємних чисел, а також виникнення переповнення розрядної сітки. Ця задача розв’язується застосуванням спеціальних кодів, за допомогою яких операція

віднімання зводиться до арифметичного додавання, що призводить до спрощення арифметичних пристроїв комп'ютера. Взагалі для подання двійкових чисел в комп'ютерах застосовують прямий, зворотний і доповняльний коди.

В усіх цих кодах додатні числа мають один і той же вигляд, а від'ємні − різний. Двійкові коди чисел будемо записувати в квадратних дужках. Розглянемо ці коди.

1.3.1. Прямий код двійкових чисел

В прямому коді число подається у вигляді абсолютного значення числа з кодом відповідного знаку.

Правило формування прямого коду може бути записане у вигляді

$$[A]_{п} = \begin{cases} A, & \text{якщо } A \geq 0; \\ 1 + |A|, & \text{якщо } A \leq 0. \end{cases} \qquad (3.1)$$

При переході від двійкового зображення числа до зображення його в прямому коді зміст мантиси збігається з дробовою частиною двійкового числа.

Знак додатного числа в прямому коді зображується цифрою **0,** а від'ємного – **1**. Знаки відділяються від мантиси крапкою.

Приклад. Подати число $A_{(2)} = 0{,}101101$ і $B_{(2)} = -0{,}110111$ в прямому коді.

$$A_{(2)} = 0{,}101101 \;\rightarrow\; [A]_{п} = 0.101101;$$

$$B_{(2)} = -0{,}110111 \;\rightarrow\; [B]_{п} = 1.110111.$$

З формули (3.1) видно, що нуль в прямому коді має два зображення:

$$A_{(2)} = +0{,}00\ldots0 \;\rightarrow\; [A]_{п} = 0.00\ldots0;$$

$$A_{(2)} = -0{,}00\ldots0 \;\rightarrow\; [A]_{п} = 1.00\ldots0.$$

Це треба мати на увазі при порівнюванні чисел.

Додавання чисел, які подані в прямому коді і мають однакові знаки, виконується досить просто. Мантиси чисел додаються і сумі привласнюється код знаку доданків.

Приклад. Подати два числа $A_{(2)} = -0{,}10010$ і $B_{(2)} = -0{,}01011$ в прямому коді з перевіркою додавання в двійковій системі числення.

Розв'язання:

1 Спочатку проведемо додавання в двійковій системі числення.

$$\begin{array}{r} A_{(2)} = -0{,}10010 \\ + \quad B_{(2)} = -0{,}01011 \\ \hline (A + B)_{(2)} = -0{,}11101 \end{array}$$

2. Проведемо додавання чисел A і B в прямому коді

$$\begin{array}{r} [A]_п = 1.10010 \\ + \quad [B]_п = 1.01011 \\ \hline [A + B]_п = 1.11101 \end{array}$$

3. Перейдемо від прямого коду до двійкової системи числення, отримаємо

$$(A + B)_{(2)} = -\,0{,}11101$$

що підтверджує правильність виконання обчислень.

Операція додавання чисел з різними знаками (операція алгебраїчного додавання) є більш складною. В цьому випадку приходиться визначати більше за модулем число, з мантиси більшого числа виконати віднімання мантиси меншого числа, а потім різниці привласнити знак більшого (за модулем) числа.

Таким чином, для реалізації операції віднімання (алгебраїчного додавання) чисел в прямому коді необхідне додаткове обладнання, що конструктивно ускладнює арифметико-логічний пристрій комп'ютера. Це є недоліком прямого коду.

В комп'ютерах прямий код взагалі застосовується для запису і зберігання чисел в запам'ятовуючому пристрої, при вводі інформації та виводі результатів, а також для виконання операцій множення і ділення, в ході яких можуть використовуватись абсолютні значення чисел [11].

Операцію віднімання в комп'ютерах часто заміняють операцією додавання чисел в спеціальних кодах. Такими кодами є зворотний і доповняльний.

1.3.2. Зворотний код двійкових чисел

В зворотному коді операція віднімання зводиться до операції простого арифметичного додавання. Зворотний код утворюється відповідно до формули

$$[A]_з = \begin{cases} A, & \text{якщо } A \geq 0; \\ 10^{n+1} - 10^{-m} + A, & \text{якщо } A \leq 0, \end{cases} \qquad (3.2)$$

де n – кількість розрядів у цілій частині числа;

m – кількість розрядів дробової частини числа;

10^{-m} – одиниця молодшого розряду числа A;

10 – число 2 в двійковій системі числення.

Для чисел меньших одиниці

$$[A]_з = \begin{cases} A, & \text{якщо } A \geq 0; \\ 10 - 10^{-m} + A, & \text{якщо } A \leq 0. \end{cases} \qquad (3.3)$$

В зворотному коді знаковий розряд розглядається як цифровий розряд цілої частини числа і при проведенні арифметичних операцій ним оперують як звичайним цифровим.

Зворотний код додатного числа збігається з його прямим кодом.

Зворотний код від'ємного числа утворюється заміною (інвертуванням) значущих цифр початкового числа на зворотні і встановленням в знаковий розряд одиниці.

Приклад. Подати числа $A_{(2)} = 0{,}1011$ і $B_{(2)} = -0{,}1011$ в зворотному коді.

Розв'язання:

$$[A]_з = 0.1011, \qquad [B]_з = 10 - 0.1011 = 1.0100.$$

З формули (3.3) видно, що нуль в зворотному коді має два зображення:

$$A_{(2)} = +0{,}00\ldots0 \;\rightarrow\; [A]_з = 0.00\ldots0;$$

$$A_{(2)} = -0{,}00\ldots0 \;\rightarrow\; [A]_з = 1.11\ldots1.$$

Для переходу від зворотного коду до двійкового зображення числа необхідно замість знакового розряду записати мінус (якщо в знаковому розряді стоїть одиниця), а мантису числа інвертувати.

Приклад. Знайти двійкове зображення числа $[A]_з = 1.0100$.

Відповідь: $A_{(2)} = -0{,}1011$.

Для від'ємних чисел перехід від прямого коду до зворотного здійснюється за таким правилом.

ПРАВИЛО. Для отримання зворотного коду від'ємного числа з прямого коду необхідно всі розряди мантиси про інвертувати, а в знаковому розряді залишити одиницю.

Це ж правило справедливе і при переведенні від'ємних чисел із зворотного коду в прямий.

При алгебраїчному додаванні чисел, поданих в зворотному коді, виконується арифметичне додавання цих кодів, включаючи розряди знаків, які при цьому розглядаються як звичайні цифрові розряди. При виникненні одиниці, що вийшла за знаковий розряд, вона додається до молодшого розряду суми кодів. Таке перенесення називається циклічним [11].

Пояснимо це. Згідно з (3.3) від'ємні числа в зворотному коді можуть бути подані таким чином:

$$[A]_з = 10 - 10^{-m} - |A| \quad і \quad [B]_з = 10 - 10^{-m} - |B|.$$

Знайдемо суму чисел і (-A) (-B).

$$[A]_з + [B]_з = (10 - 10^{-m} - |A|) + (10 - 10^{-m} - |B|) = \underbrace{10}_{\text{одиниця переносу із знакового розряду}} + (10 - 10^{-m} - 10^{-m} - |A + B|).$$

Вираз $(10 - 10^{-m} - 10^{-m} - |A + B|)$ являє собою зменшену на одиницю молодшого розряду суму чисел A і B, яка подана в зворотному коді.

Тому, щоб отримати правильний результат суми, необхідно одиницю переносу із знакового розряду додати до молодшого розряду суму. Тоді сума чисел A і B буде виражена згідно з (3.3) і матиме вигляд:

$$[A + B]_з = 10 - 10^{-m} - |A + B|).$$

Приклад. Додати два числа $A_{(2)} = -0{,}011010$ і $B_{(2)} = -0{,}100011$ в зворотному коді.

Розв'язання:

$$\begin{array}{r} [A]_з = 1.100101 \\ + \\ [B]_з = 1.011100 \\ \hline 11.000001 \\ \hline [A + B]_з = 1.000010 \end{array} \quad \begin{array}{l} \\ \\ \\ \\ \curvearrowright 1 \\ \\ \end{array}$$

Відповідь: $[A + B]_з = 1.000010$

Приклад. Додати два числа $A_{(2)} = -0{,}110110$ і $B_{(2)} = 0{,}100101$ в зворотному коді. Результат подати в прямому коді.

Розв'язання:

$$\begin{array}{r} [A]_з = 1.001001 \\ + \\ [B]_з = 1.100101 \\ \hline [A + B]_з = 1.101110 \end{array}$$

Для отримання прямого коду необхідно інвертувати мантису.

Відповідь: $[A + B]_п = 1.010001$.

Перевагою зворотного коду є можливість звести операцію *віднімання* до операції *додавання.*

Недоліки зворотного коду:

− виникнення переповнення розрядної сітки в тому випадку, коли абсолютне значення суми більше одиниці, що призводить до зміни результатів обчислень. Це явище буде розглянуте нижче;

− виникнення циклічного переносу збільшує час додавання чисел і тим самим зменшує швидкодію комп'ютерів. Цей недолік відсутній при додаванні чисел в доповняльному коді.

1.3.3. Доповняльний код двійкових чисел

Доповняльний код двійкових чисел утворюється відповідно до формули

$$[A]_д = \begin{cases} A, & \text{якщо } A \geq 0; \\ 10^{n+1} + A, & \text{якщо } A < 0, \end{cases}$$

де n – кількість розрядів у цілій частині числа;

10 – число 2 в двійковій системі числення.

Для чисел, менших одиниці,

$$[A]_д = \begin{cases} A, & \text{якщо } A \geq 0; \\ 10 + A, & \text{якщо } A < 0, \end{cases} \qquad (3.4)$$

Доповняльний код додатного числа збігається з його прямим кодом.

Для отримання доповняльного коду від'ємного числа перетворимо вираз (3.4) таким чином:

$$[A]_д = 10 - |A| = 10 - 10^{-m} - |A| + 10^{-m} = [A]_з + 10^{-m}. \qquad (3.5)$$

Із виразу виходить що для отримання доповняльного коду від'ємного числа необхідно отримати його зворотний код, а потім до молодшого розряду числа, яке подане в зворотному коді, додати одиницю.

Приклад. Подати число $A_{(2)} = -0{,}11010$ в доповняльному коді.

Розв'язання:

$$[A]_з = 1.00101$$

$$[A]_д = 1.00110$$

Відповідь: $[A]_д = 1.00110$

Якщо при виконанні арифметичних операцій в доповняльному коді виникає одиниця, яка вийшла за знаковий розряд, то вона відкидається.

Пояснимо це. Нехай треба додати два від'ємних числа A і B в доповняльному коді. Згідно з (3.4) доповняльний код від'ємних чисел буде мати наступний вигляд:

$$[A]_д = 10 - |A| \qquad \text{і} \qquad [B]_д = 10 - |B|.$$

Тоді

$$[A]_д + [B]_д = (10 - |A| + 10 - |B|) = \underbrace{10}_{\text{одиниця переносу відкидається}} + (10 - |A + B|).$$

Вираз в дужках $(10 - |A + B|)$ є сумою від'ємних чисел A і B, яка подана в доповняльному коді. Отже щоб отримати правильний результат суми в доповняльному коді, необхідно відкинути одиницю переносу із знакового розряду.

Відповідно до виразу (3.4) нуль в доповняльному коді має одне зображення:

$$[0]_д = 0.00\ldots0,$$

$$[0]_д = [0]_з + 10^{-m} = 1.11\ldots1 + 0.00\ldots1 = \underbrace{10}_{\text{одиниця відкидається}}.00\ldots0 = 0.00\ldots0.$$

Сформулюємо правило додавання чисел в доповняльному коді.

ПРАВИЛО. При алгебраїчному додаванні двійкових чисел, які подані в доповняльному коді, виконується додавання цих кодів, включаючи розряди знаків, які при цьому розглядаються як звичайні цифрові розряди. При виникненні переносу із знакового розряду одиниця переносу відкидається.

Сума отримується в прямому коді, якщо вона додатна, і в доповняльному коді, якщо вона від'ємна. Якщо сума від'ємна, то для отримання прямого коду необхідно взяти двійковий додаток від доповняльного коду [11].

Приклад. Додати два числа $A_{(2)} = -\ 0{,}10011$ і $B_{(2)} = -\ 0{,}01001$ в доповняльному коді. Результат подати в прямому коді.

Розв'язання:

$$\begin{array}{r} [A]_д = 1.01101 \\ + \quad [B]_д = 1.10111 \\ \hline [A + B]_д = 11.00100 \end{array}$$

одиниця відкидається

$$[A + B]_п = \left[[A + B]_д\right]_д = 1.11100.$$

Відповідь: $[A + B]_п = 1.11100.$

Таким чином, використання доповняльного коду дозволяє позбавитись від циклічної передачі одиниці переносу, що сприяє підвищенню швидкодії комп'ютера, але отримати доповняльні коди доданків важче, тому, що спочатку необхідно отримати зворотний код, а потім додати до нього одиницю молодшого розряду.

1.3.3. Модифіковані коди

В процесі виконання арифметичних операцій можливий варіант, коли модуль отриманого результату перевищує одиницю, тобто перевищує максимально допустиме число, яке може бути записане в розрядну сітку комп'ютера. Це явище, яке називається переповненням розрядної сітки, приводить до спотворення результатів. Покажемо це.

Приклад. Додати два числа $A_{(2)} = -\ 0{,}10101$ і $B_{(2)} = -\ 0{,}11001$ в зворотному і додатковому кодах.

Розв'язання:

Модуль суми цих чисел більший за одиницю, тобто $|A + B| > 1$.

Виконаємо додавання цих чисел:

$$\begin{array}{r} [A]_з = 1.01010 \\ + \quad [B]_з = 1.00110 \\ \hline 10.10000 \\ \hline [A + B]_з = 0.10001 \end{array} \qquad \begin{array}{r} [A]_д = 1.01011 \\ + \quad [B]_д = 1.00111 \\ \hline 10.10010 \\ \hline [A + B]_д = 0.10010 \end{array}$$

→ 1

відкидається

Отриманий результат є невірним з двох позицій.

По-перше, в знаковому розряді стоїть нуль, який показує, що отримана сума додатна, а насправді результат повинен бути від'ємним.

По-друге, в цифрових розрядах числа також отримане невірне значення.

Це виходить через те, що результат додавання не вкладається в розрядну сітку комп'ютера, тобто спостерігається переповнення розрядної сітки. Таке явище відбувається кожного разу, коли абсолютне значення суми більше одиниці, тобто при $|A + B| > 1$.

Щоб результат обчислень вийшов правильним, необхідно виключити переповнення розрядної сітки.

Для цього використовують модифіковані коди.

Модифіковані коди відрізняються від розглянутих вище тим, що для зображення знака числа відводиться два розряди.

Якщо число додатне, то в знаковому розряді записується два нулі, якщо від'ємне – дві одиниці [11].

Приклад. Подати число $A_{(2)} = -0{,}101011$ в прямому, зворотному і додатковому модифікованих кодах.

Розв'язання:

$[A]^{м}_{п} = 11.101011$; $[A]^{м}_{з} = 11.010100$; $[A]^{м}_{д} = 11.010101$.

Алгебраїчне додавання двійкових чисел в модифікованих кодах виконується за такими ж правилами, як і в звичайних кодах, при цьому знакові розряди розглядаються як розряди цілої частини числа.

Приклад. Виконати додавання чисел $A_{(2)} = -0{,}00101$ і $B_{(2)} = -0{,}10011$ в зворотному і додатковому модифікованих кодах. Результат подати в прямому коді.

Розв'язання:

$[A]^{м}_{з} = 11.11010$	$[A]^{м}_{д} = 11.11011$
$[B]^{м}_{з} = 11.01100$	$[B]^{м}_{д} = 11.01101$
111.00110	111.01000
↓ ⟵ 1	↓ відкидається
$[A + B]^{м}_{з} = 11.00111$	$[A + B]^{м}_{д} = 11.01000$
$[A + B]_{п} = 1.11000$	$[A + B]_{п} = 1.11000$

Відповідь: $[A + B]_{п} = 1.11000$

При використанні модифікованих кодів ознакою переповнення розрядної сітки є наявність в знакових розрядах різних цифр: 01 або 10. Якщо в старшому знаковому розряді суми стоїть 0 (комбінація 01), то це означає, що отриманий

результат додатний. Якщо в старшому знаковому розряді суми стоїть 1 (комбінація 10), то це означає, що отриманий результат від’ємний.

У випадку переповнення розрядної сітки в комп’ютерах з фіксованою комою він зупиняється. Це свідчить про те, що масштабування виконане невірно. Для усунення переповнення розрядної сітки необхідно знову виконати масштабування.

Якщо здійснюється переповнення розрядної сітки в комп’ютері з плаваючою комою, то в цьому випадку комп’ютер здійснює так звану операцію нормалізації вправо на один розряд. В результаті виконання цієї операції зсувається вправо на один розряд код результату алгебраїчного додавання і додається одиниця до порядку числа. При цьому модуль числа залишається без змін.

<u>*Приклад.*</u> Додати два числа, записані в формі з плаваючою комою з використанням доповняльного модифікованого коду:

$$A_{(2)} = -\,0{,}110110 \cdot 10^{011} \text{ і } B_{(2)} = -\,0{,}001101 \cdot 10^{011}.$$

<u>*Розв’язання:*</u>

$$
\begin{array}{r}
[A]_{д}^{м} = 11.001010 \quad 0.011 \\
[B]_{д}^{м} = 11.110011 \quad 0.011 \\
\hline
\not{1}10.111101 \quad 0.011 \\
\hline
[A+B]_{д}^{м} = 10.111101 \quad 0.011
\end{array}
$$

Видно, що відбулося переповнення розрядної сітки. Виконуємо зсув мантиси і її знаку на один розряд вправо і збільшення порядку на одиницю

$$[A+B]_{д}^{м} = 11.0111101 \quad 0.100.$$

Таким чином, модифіковані коди застосовуються для скорочення часу на визначення переповнення розрядної сітки.

1.4. АЛГОРИТМИ ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ У ЦІЛОЧИСЕЛЬНИХ ОПЕРАЦІЙНИХ ПРИСТРОЯХ

Для більшості сучасних комп’ютерів загальноприйнятим є такий формат з фіксованою комою (ФК), коли кома фіксується праворуч від молодшого розряду коду числа. З цієї причини відповідні операційні пристрої називають цілочисельними ОПр. У формі з ФК можуть бути представлені як числа без знака, коли все $\boldsymbol{n}$ позицій числа відводяться під значущі цифри, так і зі знаком. У останньому випадку старший (n – 1)-й розряд числа займає знак числа (0 – плюс, 1 – мінус), а під значущі цифри відведені розряди з (n – 2)-го по 0-й. Під час запису негативних чисел використовується доповняльний код.

Цілочисельний ОПр повинний забезпечувати виконання таких арифметичних операцій над числами без знака і зі знаком: *додавання/віднімання; множення; ділення.*

1.4.1. Алгоритми виконання операцій додавання і віднімання

На рис. 4.1 наводяться приклади додавання цілих чисел, поданих у доповняльному коді (нагадаємо, що під час додавання в доповняльному коді знаковий розряд бере участь в операції нарівні з цифровими).

Під час додавання *n*-розрядних двійкових чисел (біт знака і *n* – 1 значущих цифр) можливий результат, який містить *n* значущих цифр.

Ця ситуація відома як *переповнення*. «Зайвий» біт займає позицію знака, що приводить до некоректності результату. Природно, що ОПр повинний виявляти факт переповнення і сигналізувати про нього. Для цього використовується наступне правило: *якщо додаються два числа і вони обидва позитивні або обидва негативні, переповнення має місце тоді і тільки тоді, коли знак результату протилежний знаку доданків* [9].

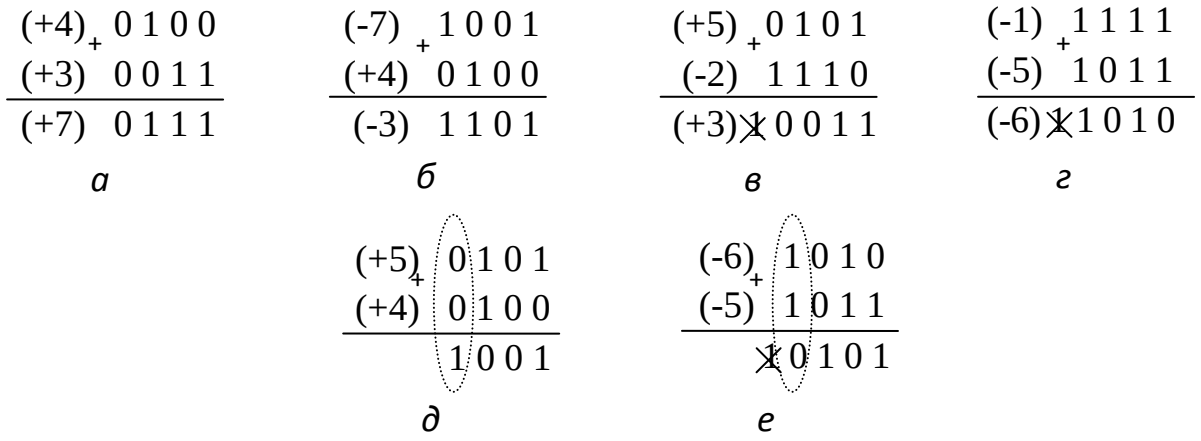


Рис. 4.1. Приклади виконання операції додавання в доповняльному коді:
а, б, в, г – додавання без виникнення переповнення;
д, е – додавання з переповненням

Рисунки 4.1, *д* і 4.1, *е* показують приклади переповнення. Звернемо увагу, що переповнення не завжди супроводжується перенесенням із знакового розряду.

Віднімання виконується відповідно до правила: *для віднімання одного числа (від'ємника) з іншого (зменшуваного) необхідно взяти доповнення від'ємника і додати його до зменшуваного*. Під доповненням тут розуміється від'ємник з протилежним знаком, поданий у доповняльному коді. Віднімання ілюструється прикладами (рис. 4.2). Два останні приклади (рис. 4.2, *д* і 4.2, *е*) демонструють раніше розглянуте правило виявлення переповнення.

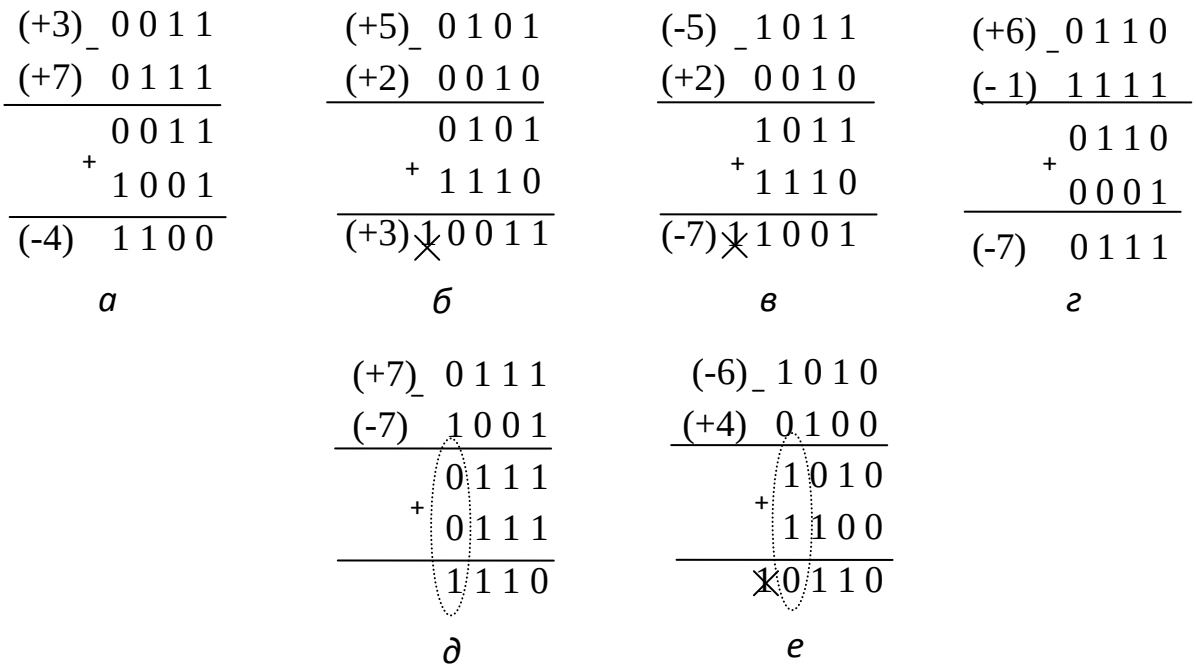


Рис. 4.2. Приклади виконання операції віднімання в доповняльному коді:
а, *б, в, г* - віднімання без виникнення переповнення;
д, е - віднімання з переповненням

Щоб спростити виявлення ситуації переповнення, часто застосовується так званий *модифікований доповняльний код*, коли для зберігання знака відводяться два розряди, причому обидва беруть участь в арифметичній операції нарівні з цифровими розрядами.

На рис. 4.3 показана можлива структура операційного блока (ОБ) для додавання і віднімання чисел із знаком у форматі з фіксованою комою.

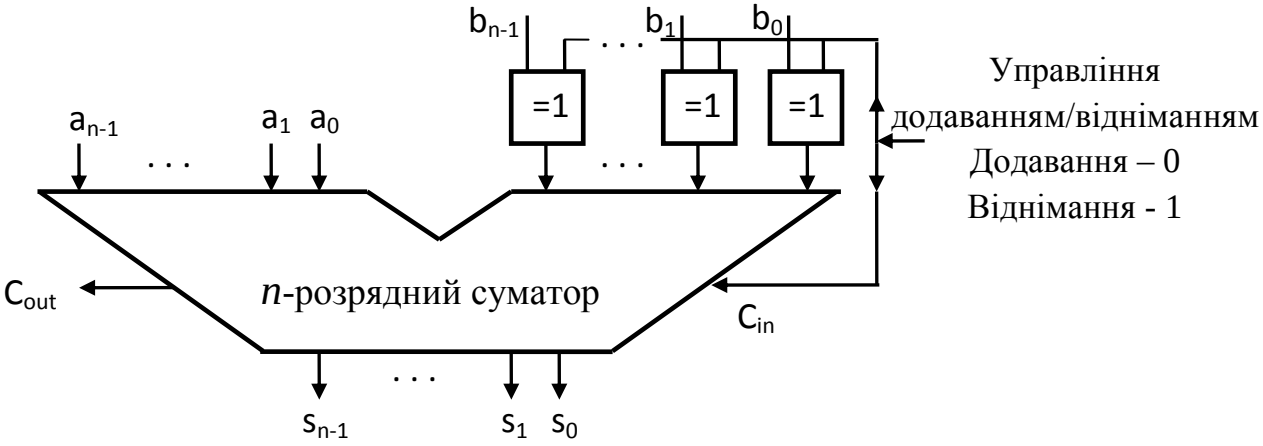


Рис. 4.3. Структура операційного блока для додавання і віднімання

Центральною ланкою пристрою є *n*-разрядный двійковий суматор.

Операнд *A* поступає на вхід суматора без змін. Операнд *B* заздалегідь пропускається через схеми додавання по модулю 2, тому вид коду *B*, що поступає на інший вхід суматора, залежить від виконуваної операції. Якщо задана операція додавання (управляючий код – 0), то результат на виході ОБ

визначається виразом $S = A + B$. Якщо задана операція віднімання (управляючий код – 1), на вхід суматора подаються інверсні значення всіх розрядів B і, крім того, на вхід перенесення в молодший розряд суматора C_{in} поступає 1. В результаті на виході ОБ буде $S = A + \overline{B} + 1$, що відповідає додаванню до A числа B з протилежним знаком, тобто відніманню.

1.4.2. Алгоритми множення кодів чисел в АЛП з фіксованою комою

У порівнянні з додаванням і відніманням, множення – складніша операція як у разі програмного, так і в разі апаратного втілення. В комп'ютерах застосовуються різні алгоритми реалізації операції множення і, відповідно, декілька схем побудови операційних блоків, що забезпечують виконання операції множення.

Традиційна схема множення схожа на відому процедуру запису «в стовпчик». Обчислення добутку P **(** p_{2n-1} p_{2n-2} **...** p_1 p_0) двох ***n***-розрядних двійкових чисел без знака A **(** a_{n-1} a_{n-2} **...** a_1 a_0) і B **(** b_{n-1} b_{n-2} **...** b_1 b_0) зводиться до формування часткових добутків (ЧД) W_i, поодинці на кожну цифру множника, з подальшим підсумовуванням отриманих ЧД. Перед підсумовуванням кожен частковий добуток повинен бути зсунутим на один розряд щодо попереднього згідно з вагою цифри множника, якій цей ЧД відповідає. Оскільки операндами є двійкові числа, обчислення ЧД спрощується – якщо цифра множника b_i дорівнює 0, то W_i теж дорівнює 0, а при $b_i = 1$ частковий добуток дорівнює множеному ($W_i = A$). Множення двох ***n***-розрядних двійкових чисел $P = A \times B$ приводить до отримання результату, що містить ***2n*** бітів. Таким чином, алгоритм множення припускає послідовне виконання двох операцій – додавання і зсуву (рис. 4.3).

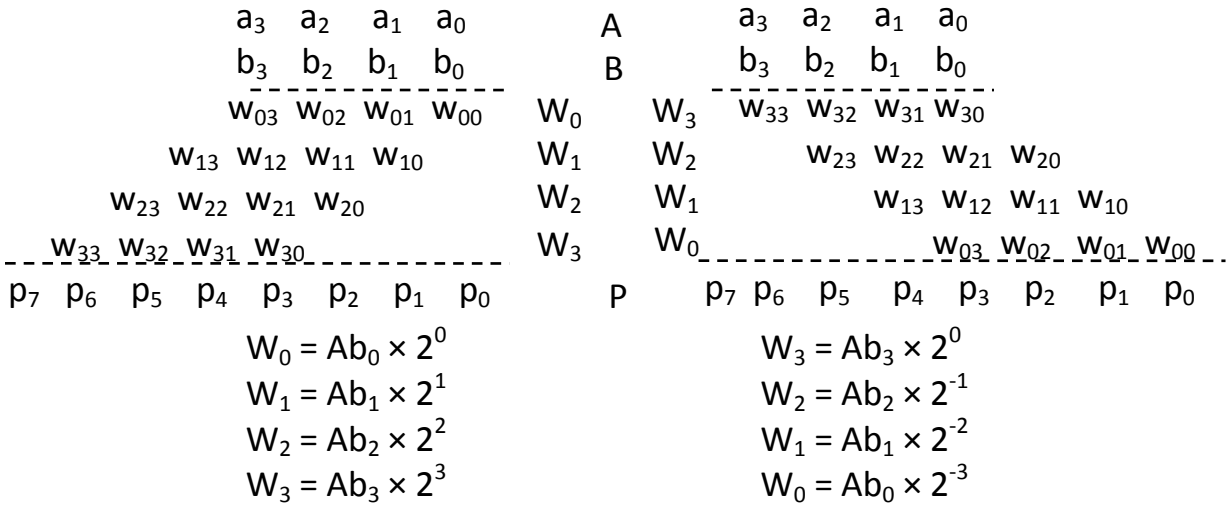


Рис. 4.3. Загальна схема множення із зсувом суми часткових добутків вліво або вправо

Підсумовування ЧД зазвичай проводиться не на завершальному етапі, а в міру їх отримання. Це дозволяє уникнути необхідності зберігання всіх ЧД, тобто скорочує апаратні витрати. Згідно з даною схемою пристрій множення припускає наявність регістрів множеного, множника і суми часткових добутків, а також суматора ЧД і схем зсуву.

Залежно від способу отримання суми часткових добутків (СЧД) можливі чотири варіанти реалізації «традиційної» схеми множення [9]:

1. Множення починається з молодших розрядів множника, і зсув суми часткових добутків відбувається вправо у разі нерухомого множеного.

2. Множення починається із старших розрядів множника, і зсув суми часткових добутків відбувається вліво у разі нерухомого множеного.

3. Множення починається з молодших розрядів множника, і зсув множеного відбувається вліво у разі нерухомої суми часткових добутків.

4. Множення починається із старших розрядів множника, і зсув множеного відбувається вправо у разі нерухомої суми часткових добутків.

Варіанти із зсувом множеного на практиці не використовуються, оскільки для їх реалізації регістр множеного, регістр СЧД і суматор повинні мати розрядність $2n$, тому зупинимося на найбільш розповсюдженому 1-му варіанті, який має назву *алгоритм із зсувом вправо* (*алгоритм А).*

Множення чисел без знака

Загальну процедуру традиційного множення спочатку розглянемо стосовно чисел без знака, тобто таких чисел, в яких всі n розрядів представляють значущі цифри. Процедура множення ілюструється прикладом обчислення добутку 10 × 11 (рис. 4.4).

A		1 0 1 0
B	×	1 0 1 1
0		0 0 0 0
$W_0 = Ab_0$	+	1 0 1 0
$P_0 = 0 + W_0$		0 1 0 1 0
$P_0 \times 2^{-1}$	⇒	0 1 0 1 0
$W_1 = Ab_1$	+	1 0 1 0
$P_1 = P_0 \times 2^{-1} + W_1$		0 1 1 1 1 0
$P_1 \times 2^{-1}$	⇒	0 1 1 1 1 0
$W_2 = Ab_2$	+	0 0 0 0
$P_2 = P_1 \times 2^{-1} + W_2$		0 0 1 1 1 1 0
$P_2 \times 2^{-1}$	⇒	0 0 1 1 1 1 0
$W_3 = Ab_3$	+	1 0 1 0
$P_3 = P_2 \times 2^{-1} + W_3$		0 1 1 0 1 1 1 0
$P_3 \times 2^{-1}$	⇒	0 1 1 0 1 1 1 0

Рис. 4.4. Приклад множення із зсувом суми часткових добутків вправо

Алгоритм може бути реалізований за допомогою схеми, яка показана на рис. 4.5.

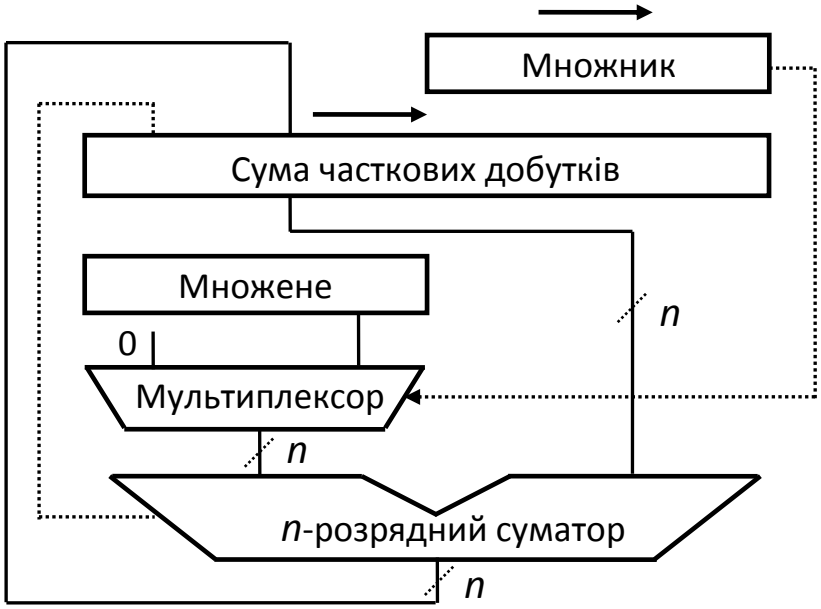


Рис. 4.5. Схема пристрою множення за алгоритмом ***A***

Спочатку множене і множник заносяться в ***n***-розрядні регістри множеного (РМн) і множника (РМк) відповідно, а всі розряди 2***n***-розрядного регістра суми часткових добутків (РЧД) встановлюються в 0. Множення відбувається за ***n*** кроків. На кожному кроці, залежно від стану молодшого розряду регістра множника, який керує мультиплексором, на один з входів ***n***-разрядного суматора подається або множене, або 0. На другий вхід поступає вміст ***n*** старших розрядів РЧД. Новий частковий добуток з суматора пересилається в старші розряди РЧД. Далі вміст РЧД зсовується на один розряд вправо, причому в старший розряд регістра, що звільнився, заноситься значення переносу із старшого розряду суматора. Оскільки мультиплексор управляється молодшим розрядом РМк, то і вміст цього регістра також зсовується на один розряд вправо. Описана послідовність повторюється ***n*** разів [9].

Економічнішим у плані апаратури є інше рішення, коли замість двох регістрів – ***n***-розрядного РМк і 2***n***-розрядного РЧД – використовується один комбінований 2***n***-розрядний регістр. Множник спочатку заноситься в молодші *n* розрядів цього регістра, а старші розряди обнуляються. В міру зсуву вправо молодші, вже проаналізовані розряди множника виштовхуються з регістра, звільняючи місце для чергової цифри СЧД. Зазвичай такий регістр будується з двох *n*-розрядных регістрів, об'єднаних ланцюгами зсуву. Додатково відзначимо, що якщо чергова цифра множника дорівнює 1, то для обчислення суми ЧД потрібні операції додавання і зсуву, а при нульовій цифрі множника в принципі можна обійтися без додавання, обмежившись тільки зсувом.

Множення чисел із знаком

Дещо складніше йде справа з множенням чисел із знаком, коли *n*-розрядні співмножники містять знак (у старшому розряді слова) і *n*–1 значущу цифру. В подальшому умовимося відокремлювати знаковий розряд крапкою, не забуваючи, проте, що знаковий розряд бере участь в операції разом з цифровими розрядами.

Найбільш очевидна думка – набути абсолютних значень операндів і перемножити їх як числа без знака. Справедливість такого рішення видно з прикладу, приведеного на рис. 4.6, де показаний процес множення чисел +13 і +10.

У всіх ОМ загальноприйнято подавати числа із знаком у формі з фіксованою комою в доповняльному коді. Додатні числа в цьому уявленні не відрізняються від запису в прямому коді, від'ємні записуються у вигляді $2^n - x$, де x – фактичне значення числа. В двійковій системі запис від'ємного числа в доповняльному коді зводиться до інвертування всіх цифрових розрядів числа, представленого в прямому коді, і додавання одиниці до молодшого розряду зворотного коду, що вийшов після інвертування.

A	×	0. 1 1 0 1	+13
B		0. 1 0 1 0	+10
0		0. 0 0 0 0	
$W_0 = Ab_0$	+	0. 0 0 0 0	
$P_0 = 0 + W_0$		0. 0 0 0 0	
$P_0 \times 2^{-1}$	⇨	0. 0 0 0 0 0	
$W_1 = Ab_1$	+	0. 1 1 0 1	
$P_1 = P_0 \times 2^{-1} + W_1$		0. 1 1 0 1 0	
$P_1 \times 2^{-1}$	⇨	0. 0 1 1 0 1 0	
$W_2 = Ab_2$	+	0. 0 0 0 0	
$P_2 = P_1 \times 2^{-1} + W_2$		0. 0 1 1 0 1 0	
$P_2 \times 2^{-1}$	⇨	0. 0 0 1 1 0 1 0	
$W_3 = Ab_3$	+	0. 1 1 0 1	
$P_3 = P_2 \times 2^{-1} + W_3$		1. 0 0 0 0 0 1 0	
$P_3 \times 2^{-1}$	⇨	0. 1 0 0 0 0 0 1 0	+130

Рис. 4.6. Множення чисел при додатних співмножниках

Зупинимося на особливостях операції множення у випадку різних поєднань знаків співмножників. Перша з них виявляється під час виконання операції арифметичного зсуву вправо для суми часткових добутків – цифрові позиції, що звільнилися у разі зсуву, повинні заповнюватися не нулем, а значенням знакового розряду зсунутого числа. Тут, проте, слід враховувати, що це правило заповнення цифрових розрядів, що звільнилися, починає діяти лише з моменту, коли серед аналізованих розрядів множника з'являється перша одиниця [9].

Множене довільного знака, множник додатний. У разі від'ємного множеного процедура множення протікає аналогічно розглянутому, з урахуванням зробленого зауваження про арифметичний зсув СЧД (рис. 4.7).

A	×	1. 0 0 1 1	−13
B		0. 1 0 1 0	+10
0		0. 0 0 0 0	
$W_0 = Ab_0$	+	0. 0 0 0 0	
$P_0 = 0 + W_0$		0. 0 0 0 0	
$P_0 \times 2^{-1}$	⇒	0. 0 0 0 0 0	
$W_1 = Ab_1$	+	1. 0 0 1 1	
$P_1 = P_0 \times 2^{-1} + W_1$		1. 0 0 1 1 0	
$P_1 \times 2^{-1}$	⇒	1. 1 0 0 1 1 0	
$W_2 = Ab_2$	+	0. 0 0 0 0	
$P_2 = P_1 \times 2^{-1} + W_2$		1. 1 0 0 1 1 0	
$P_2 \times 2^{-1}$	⇒	1. 1 1 0 0 1 1 0	
$W_3 = Ab_3$	+	1. 0 0 1 1	
$P_3 = P_2 \times 2^{-1} + W_3$		1 0. 1 1 1 1 1 1 0	
$P_3 \times 2^{-1}$	⇒	1. 0 1 1 1 1 1 1 0	−130

Рис. 4.7. Множення чисел з від'ємним множеним і додатним множником

Оскільки результат множення від'ємний, він виходить у доповняльному коді.

Множене довільного знака, множник від'ємний. На рис. 4.8 і 4.9 приведені приклади множення додатного і від'ємного множеного на від'ємний множник.

A	×	0. 1 1 0 1	+13
B		1. 0 1 1 0	−10
0		0. 0 0 0 0	
$W_0 = Ab_0$	+	0. 0 0 0 0	
$P_0 = 0 + W_0$		0. 0 0 0 0	
$P_0 \times 2^{-1}$	⇒	0. 0 0 0 0 0	
$W_1 = Ab_1$	+	0. 1 1 0 1	
$P_1 = P_0 \times 2^{-1} + W_1$		0. 1 1 0 1 0	
$P_1 \times 2^{-1}$	⇒	0. 0 1 1 0 1 0	
$W_2 = Ab_2$	+	0. 1 1 0 1	
$P_2 = P_1 \times 2^{-1} + W_2$		1. 0 0 1 1 1 0	
$P_2 \times 2^{-1}$	⇒	0. 1 0 0 1 1 1 0	
$W_3 = Ab_3$	+	0. 0 0 0 0	
$P_3 = P_2 \times 2^{-1} + W_3$		0. 1 0 0 1 1 1 0	
$P_3 \times 2^{-1}$	⇒	0. 0 1 0 0 1 1 1 0	
	+	1. 0 0 1 1	Корекція
		1. 0 1 1 1 1 1 1 0	−130

Рис. 4.8. Множення чисел у разі додатного множеного і від'ємного множника

A		1. 0 0 1 1	−13
B	×	1. 0 1 1 0	−10
0		0. 0 0 0 0	
$W_0 = Ab_0$	+	0. 0 0 0 0	
$P_0 = 0 + W_0$		0. 0 0 0 0	
$P_0 \times 2^{-1}$	⟹	0. 0 0 0 0 0	
$W_1 = Ab_1$	+	1. 0 0 1 1	
$P_1 = P_0 \times 2^{-1} + W_1$		1. 0 0 1 1 0	
$P_1 \times 2^{-1}$	⟹	1. 1 0 0 1 1 0	
$W_2 = Ab_2$	+	1. 0 0 1 1	
$P_2 = P_1 \times 2^{-1} + W_2$		0. 1 1 0 0 1 0	
$P_2 \times 2^{-1}$	⟹	1. 0 1 1 0 0 1 0	
$W_3 = Ab_3$	+	0. 0 0 0 0	
$P_3 = P_2 \times 2^{-1} + W_3$		1. 0 1 1 0 0 1 0	
$P_3 \times 2^{-1}$	⟹	1. 1 0 1 1 0 0 1 0	
	+	0. 1 1 0 1	Корекція
		0. 1 0 0 0 0 0 1 0	+130

Рис. 4.9. Множення чисел у разі від’ємного множеного і від’ємного множника

Оскільки множник від’ємний, він записується в доповняльному коді: $[B]_Д = 2^n - |B|$, і в цифрових розрядах коду буде подане число $2^{n-1} - |B|$. У разі типового множення (як у випадку $B \geq 0$) отримаємо $P' = A \times (2^{n-1} - |B|) = -|B| \times A + A \times 2^{n-1}$. Псевдодобуток P' більше дійсного добутка P на величину $A \times 2^{n-1}$, що і необхідно враховувати під час формування остаточного результату. Для цього перед останнім зсувом з отриманого псевдодобутку необхідно відняти надлишковий член. У приклади множення є згадана корекція результату.

Розглянуті процедури множення чисел із знаком у принципі можуть бути реалізовані за допомогою раніше розглянутого пристрою (див. рис. 4.5). На практиці для перемножування чисел із знаком застосовують інші алгоритми. Найбільш поширеним з них є алгоритм Бута, що має додаткову перевагу, – він прискорює процес множення в порівнянні з розглянутим раніше.

В основі алгоритму Бута [9] лежить таке співвідношення, характерне для послідовностей двійкових цифр:

$$2^m + 2^{m-1} + \cdots + 2^k = 2^{m+1} - 2^k ,$$

де m і k – номери крайніх розрядів у групі з послідовних одиниць.

Наприклад, $011110 = 2^5 - 2^1$. Це означає, що за наявності в множнику груп з декількох одиниць (комбінацій вигляду 011, 110), послідовне додавання до СЧД множеного з наростаючою вагою (від 2^k до 2^m) можна замінити

відніманням з СЧД множеного з вагою 2^k і додаванням до СЧД множеного з вагою 2^{m+1}.

Як видно, алгоритм припускає три операції: зсув, додавання і віднімання. Крім скорочення числа додавань (віднімань) у нього є ще одна перевага – він у рівній мірі застосовний до чисел без знака і зі знаком.

Розглянуті алгоритми відносилися до подання чисел з фіксованою комою, тобто, як це прийнято в більшості ОМ, до цілих чисел. При перемножуванні чисел із знаком необхідно брати до уваги, що добуток двох n-розрядних чисел із знаком (знак і n–1 значущий розряд) може мати $2(n–1)$ значущих розрядів і для його зберігання зазвичай використовують регістр подвійної довжини ($2n$ розрядів). Оскільки число ітерацій в операції множення визначається кількістю цифрових розрядів множника, кінцевий результат може розміщуватися в розрядній сітці подвійного слова невірно. Молодший розряд добутку цілих чисел, що має вагу 2^0, розміщується в позиції подвійного слова, яка має вагу 2^1. Таким чином, для правильного розташування добутку в розрядній сітці подвійного слова необхідний додатковий зсув вправо. Такий зсув можна врахувати як в апаратурі помножувача, так і програмним способом.

З іншого боку, при перемножуванні правильних дробів додатковий зсув не потрібний [9]. Цю обставину необхідно брати до уваги під час побудови помножувача для чисел у формі з плаваючою комою, де мантиси, що беруть участь в операції, подані в нормалізованому вигляді, тобто правильними дробами.

Під час множення правильних дробів часто обмежуються результатом, що має одинарну довжину. В цьому випадку може застосовуватися або відкидання зайвих розрядів, або округлення.

Прискорення цілочисельного множення

Методи прискорення множення можна умовно розділити на апаратні і логічні. Ті та інші вимагають додаткових витрат обладнання, які під час використання апаратних методів зростають із збільшенням розрядності співмножників. Апаратні способи приводять до ускладнення схеми помножувача, але не зачіпають схеми управління. Додаткові витрати обладнання під час реалізації логічних методів не залежать від розрядності операндів, але схема управління помножувача при цьому ускладнюється. На практиці прискорення множення часто досягається комбінацією апаратних і логічних методів.

Логічні методи прискорення множення. Логічні підходи до прискорення множення можна поділити на дві групи:

- методи, що дозволяють зменшити кількість додавань у ході множення;

- методи, що забезпечують обробку декількох розрядів множника за крок.

Реалізація і тих і інших вимагає введення додаткових ланцюгів зсуву в регістри.

Розглянемо другу групу логічних методів [9].

Зупинимося на множенні з обробкою за крок двох розрядів множника (IBM 360/370). В принципі це ефективніша версія алгоритму Бута. Аналіз множника починається з молодших розрядів. Залежно від вхідної дворозрядної комбінації передбачаються такі дії:

- 00 – звичайний зсув на два розряди вправо суми часткових добутків (СЧД);
- 01 – до СЧД додається одинарне множене, після чого СЧД зсовується на 2 розряди вправо;
- 10 – до СЧД додається подвоєне множене, і СЧД зсовується на 2 разряди вправо;
- 11 – з СЧД віднімається одинарне множене, і СЧД зсовується на 2 розряди вправо. Отриманий результат повинен бути скоректований на наступному кроці, що фіксується в спеціальному тригері ознаки корекції.

Оскільки в разі пари 11 з СЧД віднімається одинарне множене замість збільшення потрійного, для коректування результату до СЧД перед виконанням зсуву треба було б додати збільшене вчетверо множене. Але після зсуву на два розряди вправо СЧД зменшується в чотири рази, так що на наступному кроці досить додати одинарне множене. Це враховується під час обробки наступної пари розрядів множника, шляхом обробки пари 00 – як 01, пари 01 – як 10, 10 – як 11, а 11 – як 00. В останніх двох випадках фіксується ознака корекції.

Після обробки кожної комбінації вміст регістра множника і суматора часткових добутків зсовується на 2 розряди вправо. Даний метод множення вимагає коректування результату, якщо старша пара розрядів множника дорівнює 11 або 10 і стан ознаки корекції одиничний. У цьому випадку до отриманого добутку повинне бути додане множене.

Апаратні методи прискорення множення. Традиційний метод множення за рахунок зсувів і додавань, навіть у разі його апаратної реалізації, не дозволяє досягти високої швидкості виконання операції множення. Зв’язано це, головним чином, з тим, що під час додавання до СЧД чергового часткового добутку перенос повинен розповсюдитися від молодшого розряду СЧД до старшого. Затримка через розповсюдження переносу відносно велика, причому вона повторюється під час додавання кожного ЧД.

Один із способів прискорення множення полягає в зміні системи кодування співмножників, за рахунок чого можна скоротити кількість підсумовуваних часткових добутків. Прикладом такого підходу може служити алгоритм Бута.

Ще один ресурс підвищення продуктивності помножувача – використання ефективніших способів підсумовування ЧД, що виключають витрати часу на розповсюдження переносів. Досягається це за рахунок подання ЧД у надмірній формі, завдяки чому підсумовування двох чисел не пов'язане з розповсюдженням переносу уздовж всіх розрядів числа. Найбільш уживаною формою такого надмірного кодування є так звана форма *із збереженням переносу*. В ній кожен розряд числа подається двома бітами *cs*, відомими як перенос (*c*) і сума (*s*). Під час підсумовування двох чисел у формі із збереженням переносу перенос розповсюджується не далі, чим на один розряд. Це робить процес підсумовування значно швидшим, ніж у разі додавання з розповсюдженням переносу уздовж всіх розрядів числа.

Нарешті, третя можливість прискорення операції множення полягає в паралельному обчисленні всіх часткових добутків. Якщо розглянути загальну схему множення [9], то неважко відмітити, що окремими розрядами ЧД є добутки виду $a_i b_j$, тобто добуток певного біта множеного на певний біт множника. Це дозволяє обчислити всі біти часткових добутків одночасно, за допомогою n^2 схем «І». Під час перемножування чисел в доповняльному коді окремі розряди ЧД можуть мати вид $a_i \overline{b_j}$, $\overline{a_i} b_j$ або $\overline{a_i b_j}$.

Тоді елементи «І» замінюються елементами, що реалізують відповідну логічну функцію.

Таким чином, апаратні методи прискорення множення зводяться:

- до паралельного обчислення часткових добутків;
- до скорочення кількості операцій додавання;
- до зменшення часу розповсюдження перенесень під час підсумовування часткових добутків.

Всі три підходи в будь-якому їх поєднанні зазвичай реалізуються за допомогою комбінаційних пристроїв.

Паралельне обчислення ЧД має багато різновидів. Відмінності виявляються в основному в способі підсумовування отриманих часткових добутків, і з цих позицій використовувані схеми множення можна поділити на *матричні* і *з деревовидною структурою*. В обох варіантах підсумовування здійснюється за допомогою масиву взаємозв'язаних однорозрядних суматорів. У матричних помножувачах суматори організовані у вигляді матриці, а в деревовидних вони реалізуються у вигляді дерева того або іншого типу.

Відмінності в рамках кожної з цих груп виражаються в кількості використовуваних суматорів, їх вигляді і способі розповсюдження переносів, що виникають у процесі додавання.

В *матричних помножувачах* підсумовування здійснюється матрицею суматорів, що складається з послідовних лінійок (рядків) однорозрядних суматорів із збереженням переносів (СЗП). В міру руху даних вниз по масиву суматорів кожен рядок СЗП додає до СЧД черговий частковий добуток. Оскільки проміжні СЧД представлені в надмірній формі із збереженням переносу, у всіх схемах, аж до останнього рядка, де формується остаточний результат, розповсюдження переносу не відбувається. Це означає, що затримка в помножувачах відштовхується тільки від «глибини» масиву (числа рядків суматорів) і не залежить від розрядності операндів, якщо тільки в останньому рядку матриці, де формується остаточна СЧД, не використовується схема з послідовним переносом.

Разом з високою швидкодією важливою перевагою матричних помножувачів є їх регулярність, що особливо істотно під час реалізації таких помножувачів у вигляді інтегральної мікросхеми. З іншого боку, подібні схеми займають велику площу на кристалі мікросхеми, причому із збільшенням розрядності співмножників ця площа збільшується пропорційно квадрату числа розрядів. Друга проблема з матричними помножувачами – низький рівень утилізації апаратури. У міру руху СЧД вниз кожен рядок задіюється лише одноразово, коли її перетинає активний фронт обчислень. Ця обставина, проте, може використовуватись для підвищення ефективності обчислень шляхом конвеєризації процесу множення, при якій в міру звільнення рядка суматорів остання може бути використана для множення чергової пари чисел.

1.4.3. Алгоритми ділення кодів чисел в АЛП з фіксованою комою

Ділення дещо складніша операція, ніж множення, але базується на тих же принципах. Основу складає загальноприйнятий спосіб ділення за допомогою операцій віднімання або додавання і зсуву (рис. 4.10).

Z ділене	$z_7\ z_6\ z_5\ z_4\ z_3\ z_2\ z_1\ z_0$	$d_3\ d_2\ d_1\ d_0$	D – дільник
$-\ D\ q_3 \times 2^3$	$r_3\ r_2\ r_1\ r_0$	$q_3\ q_2\ q_1\ q_0$	Q – частка
$-\ D\ q_2 \times 2^2$	$r_3\ r_2\ r_1\ r_0$		
$-\ D\ q_1 \times 2^1$	$r_3\ r_2\ r_1\ r_0$		
$-\ D\ q_0 \times 2^0$	$r_3\ r_2\ r_1\ r_0$		
	$s_3\ s_2\ s_1\ s_0$		S – остача

Рис. 4.10. Загальна схема операції ділення

Задача зводиться до обчислення частки Q і залишку S:

$$Q = \text{int}\left(\frac{Z}{D}\right), \qquad S = Z - QD, \qquad S < D.$$

Ділення виражається як послідовність віднімань дільника спочатку з діленого, а потім з часткових залишків (ЧЗ), що утворюються в процесі ділення. Ділене Z ($z_{2n-1}\ z_{2n-2}\ \dots\ z_1\ z_0$) зазвичай записується подвійним словом ($2n$ розрядів), дільник D ($d_{2n-1}\ d_{2n-2}\ \dots\ d_1\ d_0$), частка Q ($q_{2n-1}\ q_{2n-2}\ \dots\ q_1\ q_0$) і залишок S ($s_{2n-1}\ s_{2n-2}\ \dots\ s_1\ s_0$) мають розрядність n.

Операція виконується за n ітерацій і може бути описана таким чином:

$$S^{(i)} = 2S^{(i-1)} - q_{n-i}(2^n D),\ \text{якщо}\ S^{(0)} = Z\ \text{ і }\ S^{(n)} = 2^n S;$$

$$q_{n-i} = \begin{cases} 1, \text{якщо } (2S^{(i-1)} - 2^n D) \geq 0, \\ 0, \text{якщо } (2S^{(i-1)} - 2^n D) < 0. \end{cases}$$

Після n ітерацій виходить

$$S^{(n)} = 2^n S^{(0)} - Q(2^n D) = 2^n[Z - (Q \times D)] = 2^n S.$$

Частка від ділення $2n$-розрядного числа на n-розрядне може містити більше, ніж n розрядів. У цьому випадку виникає переповнення, через що перед виконанням ділення необхідна перевірка умови

$$Z < (2^n - 1)D + D = 2^n D$$

З виразу виходить, що переповнення не буде, якщо число, що міститься в старших n розрядах діленого, менше дільника.

Крім цієї вимоги, перед початком операції необхідно виключити можливість ситуації ділення на 0.

Реалізувати ділення можна двома основними способами:

- з нерухомим діленим і зсовуваним вправо дільником;
- з нерухомим дільником і зсовуваним вліво діленим.

Недоліком першого способу є потреба мати в пристрої ділення суматор і регістр подвійної довжини. Другий спосіб дозволяє будувати пристрій ділення з суматором одинарної довжини. Нерухомий дільник D зберігається в регістрі одинарної довжини, а ділене Z, що зсовується відносно D, знаходиться в двох таких же регістрах. Цифри частки Q, що утворюються, заносяться в розряди одного з регістрів Z, які звільняються при зсуві розрядів Z.

Нижче на прикладі чисел без знаку розглядаються два основні алгоритми цілочисельного ділення [9].

Ділення з відновленням залишку. Найбільш очевидний алгоритм носить назву алгоритму *ділення з нерухомим дільником і відновленням залишку*. Він дуже схожий на загальноприйнятий спосіб ділення стовпчиком.

На рис. 4.11 показаний процес ділення з відновленням залишку числа 41 на 7.

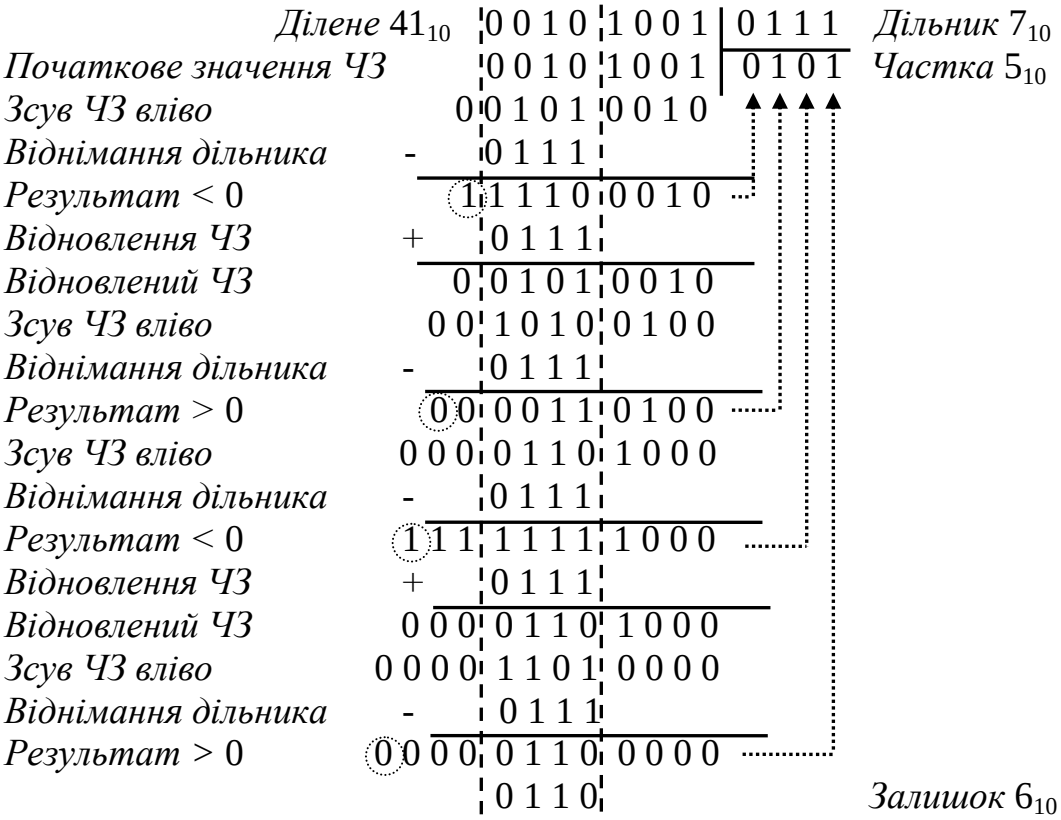


Рис. 4.11. Приклад ділення з відновленням залишку

Ділення без відновлення залишку. Недолік розглянутого алгоритму полягає в необхідності виконання на окремих кроках додаткових операцій додавання для відновлення часткового залишку. Це збільшує час виконання ділення, яке в цьому випадку може мінятися залежно від конкретного поєднання кодів операндів. Через вказані причини реальні пристрої ділення будуються на основі алгоритму *ділення з нерухомим дільником без відновлення залишку*. Процес ділення без відновлення залишку для раніше розглянутого прикладу демонструється на рис. 4.12.

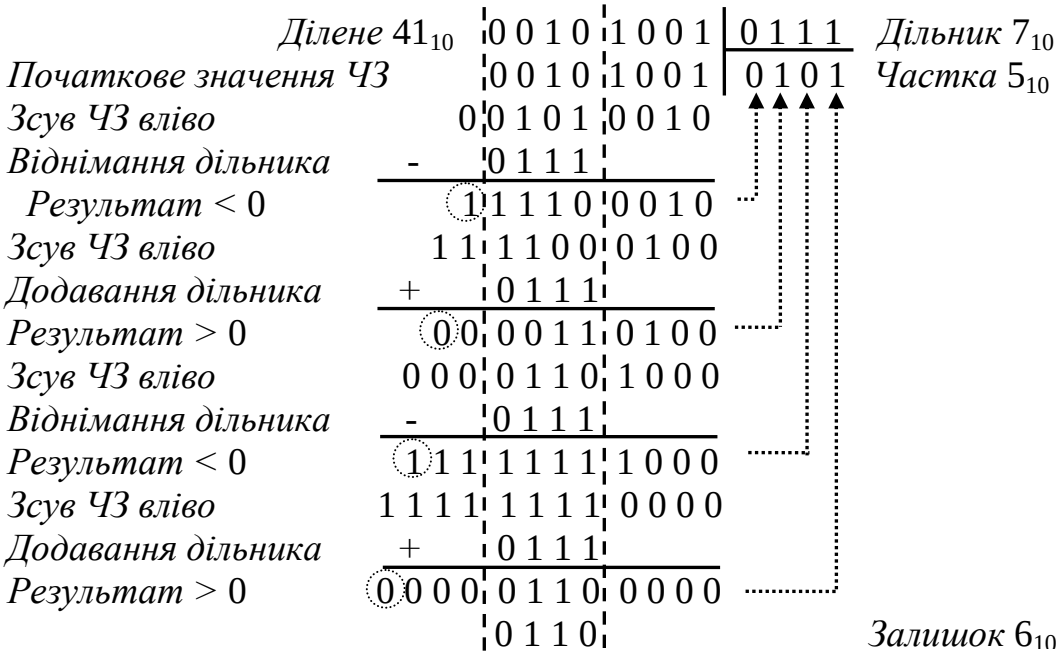


Рис. 4.12. Приклад ділення без відновлення залишку

Приведемо опис цього алгоритму [9].

1. Початкове значення часткового залишку вважається рівним старшим розрядам діленого.

2. Частковий залишок подвоюється шляхом зсуву на один розряд вліво. При цьому в молодший розряд ЧЗ, що звільняється під час зсуву, заноситься чергова цифра частки.

3. Із зсунутого часткового залишку віднімається дільник, якщо залишок додатний, і до зсунутого часткового залишку додається дільник, якщо залишок від'ємний.

4. Чергова цифра модуля частки дорівнює одиниці, коли результат віднімання додатний, і нулю, якщо він від'ємний.

5. Пункти 2…4 послідовно виконуються для отримання всіх цифр модуля частки.

Як бачимо, пункти 1, 2, 5 повністю збігаються з відповідними пунктами попереднього алгоритму ділення.

Ділення чисел із знаком

Як і у разі множення, ділення чисел із знаком може бути виконане шляхом переходу до абсолютних значень діленого і дільника, з подальшим привласненням частки знака «плюс» у разі збігу знаків діленого і дільника або «мінус» – інакше.

Ділення чисел, представлених у доповняльному коді, можна здійснювати не переходячи до модулів. Розглянемо необхідні для цієї зміни в алгоритмі без відновлення залишку.

Оскільки ділене і дільник не обов'язково мають однакові знаки, то дії з частковим залишком (додавання або віднімання D) залежать від знаків залишку і дільника і визначаються вмістом табл. 4.1:

Таблиця 4.1

Операція, яка виконується в черговій ітерації ділення

Знак залишку	Знак дільника	Дія
+	+	Віднімання
+	−	Додавання
−	+	Додавання
−	−	Віднімання

- Якщо знак залишку збігається із знаком дільника, то чергова цифра частки − 1, інакше − 0.
- Якщо $Z > 0$ і $D < 0$, частку необхідно збільшити на 1.

• Якщо $Z < 0$ і $D > 0$, то при ненульовому залишку від ділення частку потрібно збільшити на одиницю.

• Якщо $Z < 0$ і $D < 0$, то у разі нульового залишку від ділення частку потрібно збільшити на одиницю.

Залишок завжди приводиться до додатного числа, тобто якщо після закінчення ділення він від'ємний, до нього слід додати модуль дільника [9].

Пристрій ділення

Розглянутий алгоритм ділення без відновлення залишку може бути реалізований за допомогою пристрою, схема якого приведена на рис. 4.13.

Процедура починається із занесення діленого в 2*n*-розрядний регістр діленого (РДн) і дільника в *n*-розрядний регістр дільника (РДк). В лічильник циклу (ЛчЦ – на схемі не показаний), який служить для підрахунку кількості отриманих цифр частки, поміщається початкове значення, яке дорівнює *n*.

На кожному кроці вміст регістра діленого (РДн) і регістра частки (РЧ) зсовується на один розряд вліво. Залежно від поєднання знаків часткового залишку і дільника визначається значення чергової цифри частки і необхідна дія: віднімання або додавання дільника. Віднімання дільника проводиться за допомогою додавання доповняльного коду дільника. Перетворення в доповняльний код здійснюється за рахунок передачі дільника на вхід суматора зворотним (інверсним) кодом з подальшим додаванням одиниці до молодшого розряду суматора.

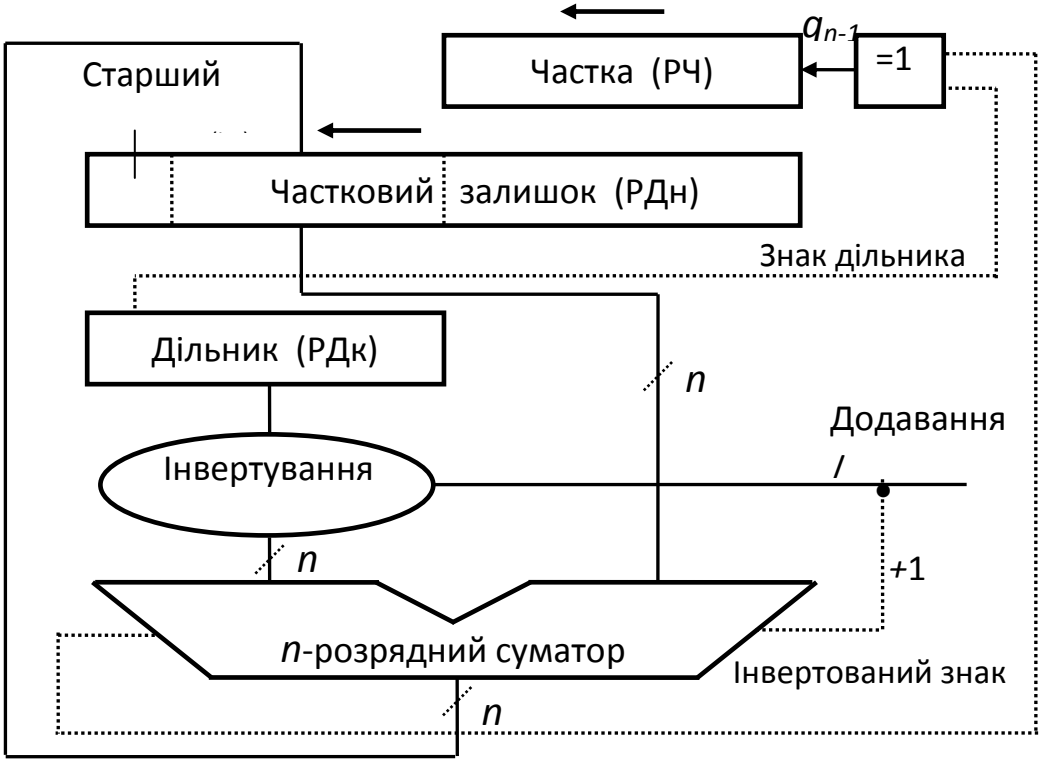


Рис. 4.13. Схема ділення за алгоритмом без відновлення залишку

Описана процедура повторюється до вичерпання всіх цифр діленого, про що свідчить нульовий вміст лічильника циклів ЛчЦ. Після закінчення операції ділення частка розташовується в регістрі частки, а в регістрі діленого буде залишок від ділення.

На завершальному етапі, якщо це необхідно, проводиться коректування отриманого результату.

На практиці для накопичення і зберігання частки замість окремого регістра використовують молодші розряди регістра діленого, що звільняються в процесі зсувів [9].

Прискорення цілочисельного ділення

Слід зазначити, що операція ділення надає не дуже багато шляхів для своєї оптимізації за часом. Проте певні можливості для прискорення ділення існують, і їх можна звести до таких:

- заміна дільника зворотною величиною, з подальшим її множенням на ділене;
- скорочення часу обчислення часткових залишків в традиційних методах ділення (з відновленням або без відновлення залишку) за рахунок прискорення операцій підсумовування (віднімання);
- скорочення часу обчислення за рахунок зменшення кількості операцій підсумовування (віднімання) під час розрахунку ЧЗ;
- обчислення частки в надмірній системі числення.

Детальніше розглянемо перший з перерахованих підходів, всі інші фактично є модифікаціями традиційного способу ділення [9].

Операцію множення можна проводити порівняно швидко, якщо взяти на озброєння комбінаційні схеми паралельного множення. Дану обставину можна використовувати, замінивши операцію ділення на D множенням на

$$Q = \frac{1}{D} = Z \times \frac{1}{D}.$$

У цьому випадку проблема зводиться до ефективного обчислення $1/D$. Зазвичай задача вирішується одним з двох методів: за допомогою ряду Тейлора або методу Ньютона-Рафсона. В обох випадках основний час витрачається на множення, тому даний метод прискорення ділення має сенс за наявності швидких схем множення.

У разі реалізації першого методу дільник D подається у вигляді: $D = 1 + X$. Тоді для двійкового подання D можна записати:

$$\frac{1}{D} = (1 - X) \times (1 + X^2) \times (1 + X^4) \times (1 + X^8) \times (1 + X^{16})...$$

Метод був використаний у моделі 91 обчислювальної машини IBM 360 для обчислення 32-розрядної величини $1/D$. Можливі значення співмножників у правій частині виразу витягувалися з таблиці ємністю 28 байт, яка зберігалася в пам'яті. Операція обчислення $1/D$ вимагає шести множень.

Обчислення величини $1/D$ методом Ньютона-Рафсона зводиться до знаходження кореня рівняння

$$f(X) = \frac{1}{X} - D = 0,$$

тобто $X = 1/D$. Рішення може бути отримане із залученням рекурентного співвідношення: $X_{i+1} = X_i\,(2 - X_i D)$. Кількість ітерацій визначається необхідною точністю обчислення $1/D$. Реалізація методу для n-розрядних чисел вимагає $2\ \mathrm{int}\,(\log_2 n) - 1$ операцій множення.

Загалом, заміна операції ділення на множення характерніша для чисел з плаваючою комою.

Можливості *прискорення обчислення часткових залишків* значно обмежені і пов'язані в основному з прискоренням операцій додавання (віднімання). Способи досягнення цієї мети нічим не відрізняються від тих, що застосовуються, наприклад, під час виконання множення. Це різні прийоми для прискорення розповсюдження переносів, матричні схеми додавання і тому подібне.

В основі *третьої групи методів прискорення операції ділення*, згідно з приведеною вище класифікацією, лежить так званий алгоритм SRT [25]. Свою назву алгоритм отримав по прізвищах авторів (Sweeney, Robertson, Tocher), що розробили його незалежно один від одного приблизно в один і той же час. Цей алгоритм є модифікацією ділення без відновлення залишку. В стандартній процедурі на кожному кроці крім зсуву часткового залишку проводиться додавання або віднімання дільника. В SRT-алгоритмі зсув ЧЗ також є в кожній ітерації, проте додавання або віднімання, залежно від ЧЗ, що виходить, на окремих кроках може не виконуватися, що, природно, позитивно впливає на швидкодію ділення.

Алгоритм був орієнтований на операції над мантисами чисел з плаваючою комою і спирається на ту обставину, що мантиси в таких числах нормалізовані. Вперше SRT-алгоритм був реалізований у моделі 91 обчислювальної машини IBM 360. В даний час він широко застосовується в блоках обробки чисел з плаваючою комою, зокрема в мікропроцесорах фірми Intel.

Найбільш поширені методи прискорення операції ділення засновані на застосуванні алгоритмів, де ділення виконується в *надмірних системах числення*, тобто в системах числення, відмінних від двійкової. Це означає, що

цифри частки можуть мати більше, ніж два значення, наприклад {-1,0,1}, як це було в алгоритмі множення Бута, або {-2,-1,0,1,2}. В таких системах одне і те ж число може бути записане декількома способами, через що системи називають надмірними. Чергова цифра частки в надмірній системі числення, залежно від бази цієї системи, відповідає двом або більше цифрам у двійковому уявленні частки, і для потрібної кількості двійкових цифр частки і залишку потрібно менше ітерацій. У той же час реалізація такого підходу веде до ускладнення апаратури дільника, зокрема, надбудовується логіка визначення операції, що виконується в черговій ітерації. Для цієї мети до складу пристрою ділення включається спеціальна пам'ять, що зберігає таблицю, яка визначає необхідні дії, залежно від поточної комбінації цифр у частковому залишку і дільнику. Проте виграш у швидкодії є вирішальним моментом.

1.4.4. Особливості виконання арифметичних операцій в АЛП з плаваючою комою

Операції над числами у форматі з плаваючою комою (ПК) мають істотні відмінності від аналогічних операцій цілочисельної арифметики, тому їх зазвичай реалізують за допомогою самостійного операційного пристрою. Як і цілочисельний ОПр, операційний пристрій для чисел у форматі ПК як мінімум повинен забезпечувати виконання чотирьох арифметичних дій: додавання, віднімання, множення і ділення.

Після ухвалення стандарту IEEE 754 негласною вимогою до всіх ОМ є забезпечення операцій з числами, поданими у форматах, які визначені даним стандартом. Нагадаємо основні положення запису чисел у стандарті IEEE 754. Мантиси чисел *M* подаються у нормалізованому вигляді, при цьому діє прийом прихованого розряду, коли старша цифра мантиси, яка завжди дорівнює одиниці, в записі числа відсутня, тобто в полі мантиси старшою є друга старша цифра нормалізованої мантиси [9].

На відміну від загальноприйнятої умови нормалізації S = |*M*| < 1, в стандарті IEEE 754 використовується умова 1 = |M| < 2.

Запис числа містить зміщений порядок, тобто порядок, збільшений на величину зміщення, яке в стандарті IEEE 754 для одинарного формату дорівнює 127, а для подвійного − 1023.

З урахуванням перерахованих особливостей арифметичну операцію над числами у форматі з плаваючою комою можна записати у вигляді:

$$\pm Z_M \times 2^{Z_{СП}} = (\pm X_M \times 2^{X_{СП}}) \Diamond (\pm Y_M \times 2^{Y_{СП}}),$$

де X_M, Y_M, Z_M − нормалізовані мантиси операндів і результату; $X_{СП}$, $Y_{СП}$, $Z_{СП}$ − зміщені порядки операндів і результату; ◇ знак арифметичної операції.

При всіх відмінностях у виконанні різних арифметичних операцій підготовчий і завершальний етапи у всіх випадках збігаються, через що має сенс розглянути їх окремо [9].

Підготовчий етап. Першою особливістю операційних пристроїв для чисел з плаваючою комою є те, що в них операції над трьома складовими чисел з ПК (знаками, мантисами і порядками операндів) виконуються роздільно: блоком обробки знаків (БОЗ), блоком обробки порядків (БОП) і блоком обробки мантис (БОМ). Для зберігання операндів і результату в ОП передбачені відповідні регістри. Хоч ці регістри можуть бути фізично реалізовані у вигляді єдиних пристроїв, кожен з них логічно розглядати як сукупність трьох регістрів: знака, порядку і мантиси. Таким чином, на етапі завантаження операндів у регістри ОП здійснюється «розпаковка» чисел з ПК, їх розбиття на три складові. У ході такого розпаковування в старшому розряді регістра мантиси відновлюється одиниця, яка в записі числа була відсутня (була прихована).

На попередньому етапі може бути також виконана перевірка на рівність нулю одного або обох операндів (у стандарті IEEE 754 для представлення нульового значення використовується такий запис числа, в якому нулю дорівнюють усі розряди порядку). Це дозволяє виключити непотрібні операції. Так, в операціях множення і ділення, якщо нулю рівні множник, множене або ділене, як результат відразу ж можна прийняти нульове значення, обійшовши наказані даними операціями дії.

Завершальний етап. Дії на завершальному етапі виконання будь-якої арифметичної операції ідентичні і зводяться до виявлення нульового значення мантиси (втрати значимості мантиси), нормалізації мантиси, виявлення від'ємного переповнювання порядку, "упаковки" складових результату.

Нульове значення мантиси може вийти в результаті операції, наприклад під час складання або віднімання мантис. Другою причиною може стати зсув мантиси вправо для усунення переповнення. В обох випадках має місце ситуація *втрати значимості мантиси*, і результат операції приймається рівним нулю. Для стандарту IEEE 754 це означає, що всі цифри порядку результату необхідно обнулити, а також те, що нормалізацію мантиси результату проводити не потрібно.

Нормалізація мантиси результату зводиться до послідовного її зсуву вліво до тих пір, поки старшу позицію не займе одиниця. Кожен зсув супроводжується зменшенням на одиницю порядку результату. В ході

зменшення порядок може стати від'ємним, що для зміщених порядків свідчить про отримання числа, неуявного в даному форматі. В такій ситуації результат приймається рівним нулю і одночасно формується ознака *втрати значимості порядку*.

На завершення мантиса результату округляється і, якщо це передбачено форматом ПК, з неї видаляється прихований розряд.

В останній фазі здійснюється «упаковка» всіх складових результату (знака, порядку і мантиси), після чого сформований результат заноситься у вихідний регістр ОП.

Додавання і віднімання

В арифметиці з плаваючою комою додавання і віднімання − складніші операції, ніж множення і ділення. Обумовлено це необхідністю вирівнювання порядків операндів. Алгоритм додавання і віднімання включає такі основні фази:

1. Підготовчий етап.
2. Визначення операнда, що має менший порядок, і зсув його мантиси вправо на число розрядів, яке дорівнює різниці порядків операндів.
3. Прирівнювання порядку результату більшому з порядків операндів.
4. Додавання або віднімання мантис і визначення знака результату.
5. Перевірку на переповнювання.
6. Завершальний етап.

Додавання і віднімання виконуються ідентично, але у разі віднімання необхідно змінити знак другого операнда на протилежний [9].

Множення

На початковому етапі множення чисел з ПК проводиться перевірка на рівність нулю одного із співмножників. Якщо один з операндів дорівнює нулю, як результат видається 0, поданий у даному форматі чисел з ПК. Наступний крок − додавання порядків. Результатом дій з порядками може стати як переповнення порядку, так і втрата значимості. У обох випадках виконання операції припиняється і видається відповідне повідомлення (виникає переривання).

Якщо порядок результату лежить у допустимих межах, на наступному кроці проводиться перемножування мантис з урахуванням їх знака, яке виконується так само, як для чисел з фіксованою комою. Під час розміщення добутку мантис у розрядній сітці необхідно враховувати, що мантиси подані не цілими числами, а правильними дробами. Хоч результат множення мантис має подвоєну в порівнянні з операндами довжину, він округляється до довжини поля мантиси.

На останньому кроці проводиться нормалізація і компоновка результату, аналогічно тому, як це має місце під час додавання і віднімання [9].

Ділення

Спочатку також проводиться перевірка на 0. Якщо нулю дорівнює дільник, залежно від реалізації видається повідомлення про ділення на 0, або як результат приймається нескінченність. Коли нулю дорівнює ділене, результат також приймається рівним нулю.

Далі виконується віднімання порядку дільника з порядку діленого, що приводить до видалення зсуву з порядку результату. Отже, для отримання зміщеного порядку результату до різниці повинен бути доданий зсув. Після виконання цих дій необхідна перевірка на переповнювання порядків і втрату значимості.

Наступний крок − ділення мантис, за яким йдуть нормалізація, округлення і компоновка числа з мантиси і порядку [9].

КОНТРОЛЬНІ ПИТАННЯ

1. Які системи числення використовуються в обчислювальній техніці?
2. Назвіть та поясніть основні характеристики позиційних систем числення.
3. Які вимоги необхідно враховувати при виборі системи числення для застосування в ЕОМ?
4. Як виконується переведення цілих чисел з одних систем числення в інші?
5. Як виконується переведення дробових чисел з одних систем числення в інші?
6. Поясніть сутність метода безпосереднього заміщення.
7. Поясніть сутність переведення чисел з двійкової системи числення у вісімкову.
8. Поясніть сутність переведення чисел з двійкової системи числення у шистнадцяткову.
9. У якій системі числення записано число 101011100110001, 100101?
10. У якій системі числення записано число B25F7,3CE?
11. Поясніть сутність подання чисел у формі з фіксованою комою.
12. Поясніть сутність подання чисел у формі з плаваючою комою.
13. Скільки розрядів потрібно використати, щоб записати двійкове число +0,0001011101 у формі з плаваючою комою ?

14. Скільки розрядів потрібно використати, щоб записати двійкове число +1011,01011011 у формі з плаваючою комою ?

15. Зробіть порівняльну характеристику подання чисел у формі з фіксованою і плаваючою комою.

16. Для чого використовується прямий код двійкових чисел?

17. Який код використовується для виконання операції віднімання?

18. Який код використовується для виконання операції множення?

19. Поясніть правило формування зворотного коду двійкових чисел.

20. Поясніть правило формування доповняльного коду двійкових чисел.

21. Поясніть сутність виконання операції додавання в зворотному коді.

22. Поясніть сутність виконання операції додавання в доповняльному коді.

23. Для чого використовуються модифіковані коди двійкових чисел?

24. Поясніть сутність виконання арифметичних операцій в модифікованих кодах.

25. Поясніть сутність виконання операції додавання в ЕОМ з плаваючою комою.

26. Виконайте додавання чисел $A = - 0{,}1001101*10^{010}$ і $B = 0{,}1110011*10^{100}$ в зворотному модифікованому коді. Результат подайте в прямому коді.

27. Поясніть сутність виконання операції множення в ЕОМ з фіксованою комою.

28. Поясніть особливість виконання операції множення в ЕОМ з плаваючою комою.

29. Поясніть сутність прискореного виконання операції множення.

30. Поясніть сутність виконання операції ділення з відновленням залишку.

31. Поясніть сутність виконання операції ділення без відновлення залишку.

32. Поясніть особливість виконання операції ділення в ЕОМ з плаваючою комою.

33. Як визначається знак добутку і частки під час виконання операцій множення та ділення відповідно?

РОЗДІЛ 2

ЛОГІЧНІ ОСНОВИ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

2.1. ПОНЯТТЯ ЛОГІЧНОЇ ФУНКЦІЇ

Змінна, яка приймає тільки два значення, позначені символами 0 або 1, називається *логічною* або *булевою* змінною.

Різні логічні змінні можуть бути зв'язані функціональними залежностями подання. Наприклад, вираз y=f (x_1,x_2) вказує на функціональну залежність логічної змінної Y від логічних змінних x_1 та x_2, які називаються аргументами (або вхідними змінними).

Функцію логічних змінних, область значень якої має тільки два значення: 0 або 1, будемо називати логічною або булевою, або перемикальною. Її позначають P_1(A,B,…) або f(x_1,x_2,…,x_n). Оскільки логічні змінні можуть приймати тільки два значення (значення істинності або значення хибності), можна показати, що в загальному випадку логічна функція ***n***-змінних включає 2^n можливих наборів змінних. А всього в цьому випадку існує 2^{2^n} логічних функцій. Для n=1 кількість наборів змінної дорівнює двом, а всього логічних функцій чотири. Для n=2 кількість наборів змінних дорівнює чотирьом, а логічних функцій – шістнадцять і т.д. З ростом числа змінних число функцій росте дуже швидко.

Логічна функція вважається повністю визначеною, якщо задані її значення на всіх наборах аргументів. Таблиця, за допомогою якої задається логічна функція, називається таблицею істинності.

Дві логічні функції називаються еквівалентними, якщо їх значення на всіх наборах збігаються, а якщо значення двох функцій не збігаються хоча б на одному наборі, то такі функції вважаються різними.

Логічні функції двох змінних

Розглянемо логічні функції двох змінних Y = f(A,B). Для n=2 число можливих різних наборів дорівнює 2^2=4, а кількість різних логічних функцій 2^{2^n} = 16.

Ці 16 логічних функцій подані в таблиці 2.1.

Із цих 16-ти функцій 6 – це функції однієї змінної: f_0 – константа 0; f_3 – змінна A; f_5 - змінна B; f_{10}- інверсія B; f_{12} - інверсія A; f_{15} - константа 1.

Таблиця 2.1

A	0	0	1	1	Умовне позначення функції	Найменування функції
B	0	1	0	1		
$f_0(A,B)$	0	0	0	0	0	Константа 0
$f_1(A,B)$	0	0	0	1	$A\cdot B=A\&B=A\wedge B$	Кон'юнкція
$f_2(A,B)$	0	0	1	0	$A\cdot\overline{B}$	Заборона по B
$f_3(A,B)$	0	0	1	1	A	Змінна A
$f_4(A,B)$	0	1	0	0	$\overline{A}\,B$	Заборона по A
$f_5(A,B)$	0	1	0	1	B	Змінна B
$f_6(A,B)$	0	1	1	0	$\overline{A}B\vee A\overline{B}=A\oplus B$	Нерівнозначність
$f_7(A,B)$	0	1	1	1	$A\vee B$	Диз'юнкція
$f_8(A,B)$	1	0	0	0	$\overline{A\vee B}=A\downarrow B$	Стрілка Пірса
$f_9(A,B)$	1	0	0	1	$\overline{A}\,\overline{B}\vee AB=\overline{A\oplus B}$	Рівнозначність
$f_{10}(A,B)$	1	0	1	0	$\overline{B}$	Інверсія B
$f_{11}(A,B)$	1	0	1	1	$A\vee\overline{B}=A\leftarrow B$	Зворотна імплікація
$f_{12}(A,B)$	1	1	0	0	$\overline{A}$	Інверсія A
$f_{13}(A,B)$	1	1	0	1	$\overline{A}\vee B=A\rightarrow B$	Імплікація
$f_{14}(A,B)$	1	1	1	0	$\overline{AB}=A\&B$	Операція Шеффера
$f_{15}(A,B)$	1	1	1	1	1	Константа 1

Розглянемо інші десять функцій.

1. Функція $f_1(A,B) = A\cdot B = A\wedge B = A\&B$ називається кон'юнкцією або логічним множенням, або функцією "І":

$$f_1(A,B)=\begin{cases}1, & A=B=1;\\ 0, & \text{в інших випадках.}\end{cases}$$

Логічний елемент, який реалізує функцію "кон'юнкція" називається кон'юнктором, або елементом "І" і має таке графічне зображення (рис. 2.1).

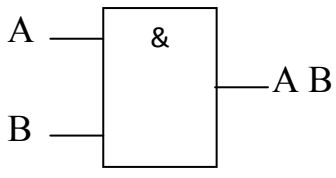


Рис. 2.1. Логічний елемент І

Сигнал на виході такого елемента з'являється тоді і тільки тоді, коли є сигнал на обох його входах.

2. Функції $f_2(A,B) = A\cdot\overline{B}$ і $f_4(A,B) = \overline{A}\cdot B$ називаються функціями заборони по B і по A відповідно.

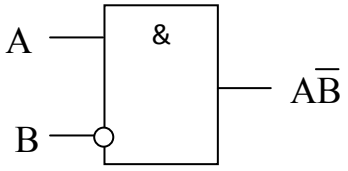


Рис. 2.2. Елемент заборони по B

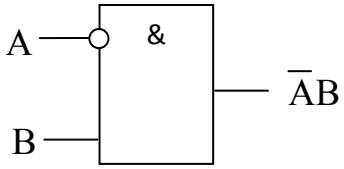


Рис. 2.3. Елемент заборони по A

$$f_2(A,B) = \begin{cases} 1, & A = 1, B = 0; \\ 0, & \text{в інших випадках.} \end{cases}$$

$$f_4(A,B) = \begin{cases} 1, & B = 1, A = 0; \\ 0, & \text{в інших випадках.} \end{cases}$$

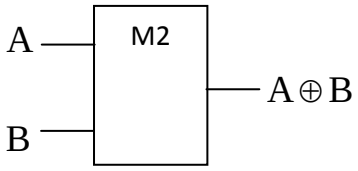
3. Функція $f_6(A,B) = \overline{A}B \vee A\overline{B} = A \oplus B = \overline{A \sim B}$ – нерівнозначність або сума по модулю два:

$$f_6(A,B) = \begin{cases} 1, & A \neq B; \\ 0, & A = B. \end{cases}$$

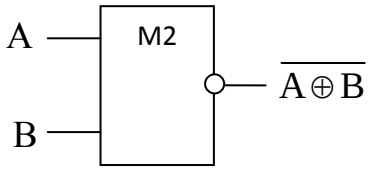
4. Функція $f_9(A,B) = \overline{A}\,\overline{B} \vee AB = A \sim B = \overline{A \oplus B}$ – рівнозначність:

$$F_9(A,B) = \begin{cases} 1, & A = B; \\ 0, & A \neq B. \end{cases}$$

Функції f_6 і f_9 реалізуються на елементах, які показані на рис. 2.4.



а



б

Рис. 2.4. Елементи, що реалізують операції
а-нерівнозначність, б-рівнозначність

5. Функція $f_7(A,B) = A \vee B = A+B$ – називається диз'юнкцією або логічним додаванням, або функцією "АБО":

$$f_7(A,B) = \begin{cases} 0, & A = B = 0; \\ 1, & \text{в інших випадках.} \end{cases}$$

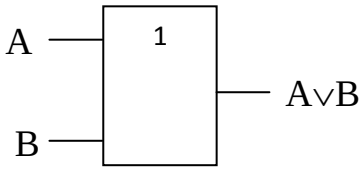
Логічний елемент, який реалізує цю функцію, називається диз'юнктором або елементом "АБО", який має вигляд, що показаний на рис. 2.5.

Сигнал на виході такого елементу з'являється тоді, коли хоч би на одному вході або на обох входах одночасно з'являється сигнал.

6. Функція $f_8(A,B) = \overline{A \vee B} = A \downarrow B$ – називається операцією (стрілкою) Пірса або запереченням диз'юнкції:

$$f_8(A,B) = \begin{cases} 1, & A = B = 0; \\ 0, & \text{в інших випадках.} \end{cases}$$

Ця функція реалізується на елементах Пірса (рис. 2.6).



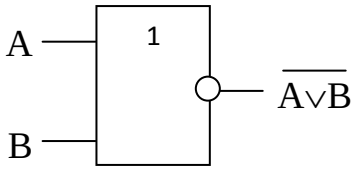


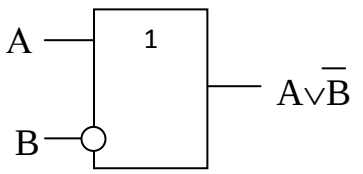
Рис. 2.5. Логічний елемент "АБО" Рис. 2.6. Елемент Пірса

7. Функції $f_{11}(A,B)=A \vee \overline{B} = A \leftarrow B$ та $f_{13}(A,B) = \overline{A} \vee B = A \rightarrow B$ – називаються зворотною імплікацією і імплікацією відповідно:

$$f_{11}(A,B) = \begin{cases} 0, & A = 0, B = 1; \\ 1, & \text{в інших випадках.} \end{cases}$$

$$f_{12}(A,B) = \begin{cases} 0, & A = 1, B = 0; \\ 1, & \text{в інших випадках.} \end{cases}$$

Ці функції графічно зображуються, як показано на рис. 2.7.



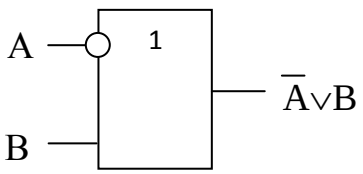


Рис. 2.7. Елементи імплікації

8. Функція $f_{14}(A,B) = \overline{AB} = \overline{A \wedge B} = \overline{A\&B}$ – називається операцією Шеффера:

$$F_{14}(A,B) = \begin{cases} 0, & A = B = 1; \\ 1, & \text{в інших випадках.} \end{cases}$$

Ця функція реалізується на елементі, який називається елементом Шеффера (рис. 2.8).

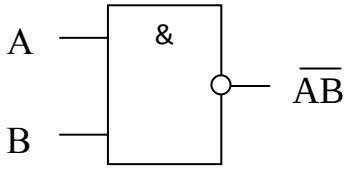


Рис. 2.8. Елемент Шеффера

2.2. ОСНОВНІ ЗАКОНИ І РІВНОСИЛЬНОСТІ АЛГЕБРИ ЛОГІКИ

В алгебрі логіки є чотири основні закони [7]:

- переміщувальний (властивість комутативності);
- сполучний (властивість асоціативності);
- розподільчий (властивість дистрибутивності);
- інверсії (правило де Моргана).

Відношення, які відображають ці закони для двох чи трьох змінних, приведені в табл. 2.2.

Таблиця 2.2

	Закон	Логічне додавання	Логічне множення
1	Переміщувальний	$A \vee B = B \vee A$	$AB = BA$
2	Сполучний	$(A \vee B) \vee C = A \vee (B \vee C)$	$(AB)C = A(BC)$
3	Розподільчий	$(A \vee B)C = AC \vee BC$	$AB \vee C = (A \vee C)(B \vee C)$
4	Інверсії	$\overline{A \vee B} = \overline{A} \cdot \overline{B}$	$\overline{A \cdot B} = \overline{A} \vee \overline{B}$

Більшість названих законів зустрічається у звичайній алгебрі і не викликає сумніву. Розподільчого закону для множення і закону інверсії у звичайній алгебрі немає. Доведення цих законів може бути виконано шляхом складання таблиць істинності для правої та лівої частин рівняння, що описують той чи інший закон, або ж методом безпосередніх перетворень. Для прикладу доведемо за допомогою таблиці істинності справедливість закону інверсії при додаванні двох змінних A та B.

В табл. 2.3 вказані значення лівої і правої частини рівняння, яке характеризує закон інверсії при додаванні. Збіг таблиць істинності для лівої і правої частин підтверджує справедливість закону інверсії для двох змінних. Неважко показати справедливість основних законів алгебри логіки і для більшого числа змінних.

Таблиця 2.3

Ліва частина $\overline{A \vee B}$		Права частина $\overline{A}\,\overline{B}$	
A	0 0 1 1	$\overline{A}$	1 1 0 0
B	0 1 0 1	$\overline{B}$	1 0 1 0
$A \vee B$	0 1 1 1	-	- - - -
$\overline{A \vee B}$	1 0 0 0	$\overline{A} \cdot \overline{B}$	1 0 0 0

Використовуючи основні закони алгебри логіки, можна скласти ряд правил, які застосовуються при аналізі складних логічних виразів з метою перетворення виразів до більш простого і зручного виду.

Ці правила зведені в табл. 2.4.

Перші три правила фактично описують властивості операції "НІ", "І", "АБО". Четверте правило вказує на те, що множення на коефіцієнти, які відрізняються від логічного нуля і логічної одиниці, а також піднесення до степеня не мають смислу в алгебрі логіки.

Таблиця 2.4

	Правило	Логічне додавання	Логічне множення
1	Інверсії	$\overline{0} = 1$	$\overline{1} = 0$
2	Незмінності	$A \vee 0 = A$	$A \cdot 1 = A$
3	Універсальної і нульової множини	$A \vee 1 = 1$	$A \cdot 0 = 0$
4	Повторення	$A \vee A = A$	$A \cdot A = A$
5	Додатковості	$A \vee \overline{A} = 1$	$A \cdot \overline{A} = 0$
6	Склеювання	$AB \vee A\overline{B} = A$	$(A \vee B)(A \vee \overline{B}) = A$
7	Поглинання	$A \vee AB = A$	$A(A \vee B) = A$
8	Подвійного заперечення	$\overline{\overline{A}} = A$	$\overline{\overline{A}} = A$

Правила склеювання і поглинання широко використовуються при мінімізації логічних функцій, яка проводиться з метою їх спрощення.

Доведемо правила склеювання методом безпосередніх перетворень. При цьому методі виходять із вірності деяких законів, і застосовуючи ці вірні закони, доводять вірність інших.

Будемо вважати, що вірні відношення:

$$A \vee \overline{A} = 1; \qquad A \cdot 1 = A; \qquad A \cdot A = A; \qquad A \cdot \overline{A} = 0.$$

Тоді для логічного додавання

$$AB \vee A\overline{B} = A(B \vee \overline{B}) = A \cdot 1 = A.$$

Для логічного множення

$$(A \vee B)(A \vee \overline{B}) = A \cdot A \vee A\overline{B} \vee AB \vee B \cdot \overline{B} = A \vee A\overline{B} \vee AB \vee 0 = A \vee A(\overline{B} \vee B) = A \vee A \cdot 1 = A \vee A = A.$$

Аналогічно доводяться правила поглинання. Варто звернути увагу на властивість симетрії, яка властива основним законам і правилам булевої алгебри. Всі правила (крім останнього) подані в таблиці 2.4 парою співвідношень. В кожній парі одне співвідношення витікає з іншого заміною додавання множенням і навпаки. Крім того, всі значення 0 замінюються на 1 і, навпаки, всі значення 1 – на 0. Ця властивість симетрії в булевій алгебрі відома як принцип двоїстості.

2.3. ФОРМИ ПОДАННЯ ЛОГІЧНИХ ФУНКЦІЙ

Логічні функції можуть мати різну форму подання. Вибір тієї або іншої форми подання визначається цілями завдання функції. Звичайно кінцевою метою подання є синтез цифрового пристрою, який реалізує це подання. Ось найбільш вживані подання логічних функцій [10]:

таблиця істинності;

номери наборів, на яких логічна функція дорівнює одиниці або нулю;

аналітичне довільне подання;

аналітичне канонічне подання.

В таблиці істинності вказуються номери наборів змінних, самі набори змінних і значення логічних функцій на цих наборах.

В табл. 2.5 приведене значення істинності для деякої логічної функції трьох змінних.

Для подання функції можна використовувати номери наборів. Наприклад, для табл. 2.5 функцію можна задавати у вигляді наборів 2,3,5 і 6, де вона дорівнює одиниці.

Довільне аналітичне подання логічної функції найбільш компактне і наочне. Крім того, деякі спеціальні методи спрощення (мінімізації) логічних функцій потребують їх подання в спеціальному (канонічному) вигляді.

Таблиця 2.5

Номер набору	A	B	C	f(A,B,C)	$K_i(1)$	$K_i(0)$
0	0	0	0	0	$K_0(1)= \overline{A}\ \overline{B}\ \overline{C}$	$K_0(0)=A\vee B\vee C$
1	0	0	1	0	$K_1(1)= \overline{A}\ \overline{B}\ C$	$K_1(0)=A\vee B\vee \overline{C}$
2	0	1	0	1	$K_2(1)= \overline{A}\ B\ \overline{C}$	$K_2(0)=A\vee \overline{B}\vee C$
3	0	1	1	1	$K_3(1)= \overline{A}\ B\ C$	$K_3(0)=A\vee \overline{B}\vee \overline{C}$
4	1	0	0	0	$K_4(1)= A\ \overline{B}\ \overline{C}$	$K_4(0)=\overline{A}\vee B\vee C$
5	1	0	1	1	$K_5(1)= A\ \overline{B}\ C$	$K_5(0)=\overline{A}\vee B\vee \overline{C}$
6	1	1	0	1	$K_6(1)= A\ B\ \overline{C}$	$K_6(0)=\overline{A}\vee \overline{B}\vee C$
7	1	1	1	0	$K_7(1)= A\ B\ C$	$K_7(0)= \overline{A}\vee \overline{B}\vee \overline{C}$

Розглянемо більш докладно засоби переходу від різних форм подання логічних функцій до канонічного вигляду.

Нормальною формою булевої функції називається така форма, яка складається тільки з диз'юнкції елементарних кон'юнкцій або з кон'юнкції елементарних диз'юнкцій.

В свою чергу *елементарною кон'юнкцією* називається вираз, який складається із довільного кінцевого числа змінних або їх інверсій, що логічно помножені одна на одну.

Наприклад, A або $\overline{B}$, або $B\overline{C}$, або $ABC\overline{D}E$ і т.д.

Елементарною диз'юнкцією називається логічний вираз, який складається із довільного кінцевого числа змінних або їх інверсій, що логічно додаються.

Наприклад, A або $A\vee\overline{B}$, або $\overline{A}\vee B\vee C$, або $A\vee\overline{B}\vee C\vee\overline{D}\vee E$ і т.д.

Розрізняють *диз'юнктивні нормальні форми* (ДНФ) подання логічних функцій і *кон'юнктивні нормальні форми* (КНФ).

ДНФ являє собою диз'юнкцію кінцевої множини елементарних кон'юнкцій. Наприклад,

$$AB\vee A\overline{B}\overline{C}\vee\overline{A}DE\vee F$$

КНФ являє собою кон'юнкцію кінцевої множини елементарних диз'юнкцій. Наприклад,

$$(A\vee\overline{B})\cdot C\cdot(\overline{D}\vee E)$$

Елементарна кон'юнкція, яка складається з усіх змінних, причому деякі або всі з них можуть бути з запереченням, називається *конституентою одиниці* і позначається $K_i(1)$, де i – номер набору, на якому логічна функція дорівнює одиниці.

Оскільки для n змінних всього існує 2^n наборів, то таке ж число конституент містить і вся множина 2^{2^n} функцій. Розглянемо запис конституент

одиниці на прикладі функції трьох змінних. В табл. 2.5 показані номери наборів, можливі набори функції трьох змінних і вирази для $K_i(1)$. Вираз для $K_i(1)$ складається таким чином: записується елементарна кон'юнкція всіх змінних. Потім ті змінні, які на даному наборі приймають значення 0, беруться з запереченням. Наприклад, $K_i(3)=\overline{A}\cdot B\cdot C$.

Елементарна диз'юнкція, яка складається з усіх змінних, причому деякі або всі з них можуть бути з запереченням, називається *конституентою нуля* і позначається $K_i(0)$, де – і номер набору, на якому $K_i(0) = 0$. В табл. 2.5 показано запис $K_i(0)$ для функції трьох змінних. Вираз для $K_i(0)$ складається таким чином: записується елементарна диз'юнкція всіх змінних. Потім ті змінні, які на даному наборі приймають значення 1, беруться з запереченням. Наприклад $K_3(0) = A\vee\overline{B}\vee\overline{C}$.

$$K_i(1) = \begin{cases} 1, & \text{тільки на і - му наборі;} \\ 0, & \text{на всіх інших наборах.} \end{cases} \qquad (2.1)$$

$$K_i(0) = \begin{cases} 0, & \text{тільки на і - му наборі;} \\ 1, & \text{на всіх інших наборах.} \end{cases} \qquad (2.2)$$

З (2.1) і (2.2) виходить що

$$K_i(1) = \overline{K_i}(0); \qquad K_i(0) = \overline{K_i}(1).$$

Досконалою диз'юнктивною нормальною формою (ДДНФ) логічної функції називається диз'юнкція конституент одиниці тих наборів, на яких логічна функція приймає значення 1.

Досконалою кон'юнктивною нормальною формою (ДКНФ) логічної функції називається кон'юкція конституент нуля тих наборів, на яких логічна функція приймає значення 0.

ДДНФ і ДКНФ є канонічними формами завдання логічних функцій.

Приклад. Подати функцію, що задана в табл. 2.5, в ДДНФ і ДКНФ.

$f_{\text{дднф}}(A,B,C) = \overline{A}B\overline{C}\vee\overline{A}BC\vee\overline{A}BC\vee AB\overline{C}$;

$f_{\text{дкнф}}(A,B,C) = (A\vee B\vee C)\,(A\vee B\vee\overline{C})\,(\overline{A}\vee B\vee C)\,(\overline{A}\vee\overline{B}\vee\overline{C})$.

2.4. МІНІМІЗАЦІЯ ЛОГІЧНИХ ФУНКЦІЙ

Одна і та ж логічна функція може бути задана багатьма способами. Тому, природно, виникає проблема пошуку найбільш раціональної, з точки зору технічної реалізації, форми подання логічних функцій. Процес пошуку таких форм називається мінімізацією.

Мінімізація логічних функцій є однією з основних задач синтезу і аналізу функціональних вузлів цифрових обчислювальних пристроїв. Метою мінімізації є отримання найбільш простих кінцевих форм логічних функцій, які

легко можна реалізувати за допомогою набору логічних елементів, на яких будуються вузли цифрових ЕОМ. Тому задача мінімізації завжди враховує специфіку реальних схем. Проте в більшості випадків ця специфіка враховується на кінцевому етапі мінімізації.

Мінімізацію форм логічних функцій звичайно використовують в такій послідовності:

- знаходять скорочену диз'юнктивну (кон'юнктивну) нормальну форму;
- із скороченої форми будують тупикові нормальні форми;
- серед тупикових вибирають мінімальні форми.

Логічна функція, що отримана з досконалої диз'юнктивної (кон'юнктивної) форми в результаті усіх можливих склеювань і поглинань, називається скороченою.

Кон'юнкції (диз'юнкції), які входять в скорочену ДНФ (КНФ) називаються імплікантами (імпліцентами).

Диз'юнкція (кон'юнкція) простих імплікант (імпліцент) логічної функції, якщо ні одна з них не є лишньою, називається тупиковою.

Диз'юнктивна або кон'юнктивна нормальна форма називається мінімальною, якщо вона містить менше число знаків двійкових змінних і їх заперечень, ніж будь-яка інша диз'юнктивна або кон'юнктивна нормальна форма тієї ж функції.

Для мінімізації логічних функцій використовуються різні методи, що основані на законах і співвідношеннях алгебри логіки.

2.4.1. Мінімізація логічних функцій методом безпосереднього перетворення

За допомогою законів і співвідношень алгебри логіки можна виконувати мінімізацію логічних функцій, заданих не тільки в канонічних, але і в довільних аналітичних формах.

Особливістю метода безпосередніх перетворень є велика залежність конкретного алгоритму мінімізації від форми початкового виразу, а також від знань і досвіду особи, яка виконує цю роботу [10].

На практиці доцільно використовувати наступну послідовність перетворення:

1. Виявлення груп змінних або груп членів вихідної форми, до яких послідовно можна застосовувати основні закони і співвідношення алгебри логіки, які приводять початковий вираз до більш простої форми.

2. Спрощення вихідної логічної формули шляхом застосування до виявлених груп відповідних законів і співвідношень.

3. Перетворення проміжної логічної формули з метою формування груп змінних або членів формули, аналогічних загальним співвідношенням, що виражають закони та співвідношення алгебри логіки. Ці перетворення призводять до групування членів, які застосовують дії, що розкривають дужки або виводять за дужки, додавання членів, які еквівалентні нулю, множення членів на диз'юнкцію змінної і Її заперечення і т. п.

4. Спрощення проміжної перетвореної логічної формули з отриманням скороченої диз'юнктивної або кон'юнктивної нормальної форми.

5. Виявлення і видалення лишніх членів з скороченої форми.

Щоб перевірити логічну функцію на лишні члени, потрібно для кожного члена знайти набір, на якому він перетворюється в одиницю, і визначити значення суми залишкових членів на цьому наборі. Якщо вона рівна одиниці, то член, що перевіряється є лишнім. Виявлення і усунення лишніх членів необхідно проводити послідовно.

Використання законів і рівносильностей для мінімізації розглянемо на прикладі.

Приклад. Необхідно мінімізувати логічну функцію

$$f(A,B,C) = ABC \vee AB\overline{C} \vee A\overline{B}C \vee A\overline{B}\overline{C} \vee \overline{A}BC \vee \overline{A}B\overline{C}.$$

На підставі розподільного закону групуємо кон'юнкції 1 і 2, 3 і 4, 5 і 6. Виносимо загальні множники за дужки, отримуємо:

$$f(A,B,C) = (ABC \vee AB\overline{C}) \vee (A\overline{B}C \vee A\overline{B}\overline{C}) \vee (\overline{A}BC \vee \overline{A}B\overline{C}) =$$

$$= AB(C \vee \overline{C}) \vee A\overline{B}(C \vee \overline{C}) \vee \overline{A}B(C \vee \overline{C}).$$

Так як $C \vee \overline{C} = 1$, то $f(A,B,C) = AB \vee A\overline{B} \vee \overline{A}B$.

Якщо застосувати формулу склеювання для перших двох членів, отримуємо:

$$f(A,B,C) = A \vee \overline{A}B.$$

Використовуючи розподільний закон для логічного множення, найдемо:

$$f(A,B,C) = (A \vee \overline{A}) \vee (A \vee B) = A \vee B.$$

2.4.2. Мінімізація логічних функцій методом Квайна

Метод Квайна є різновидом мінімізації логічних функцій методом безпосереднього перетворення [2]. Практичне застосування цього методу полягає в наступному.

Логічна функція повинна бути задана в ДДНФ (ДКНФ) або приведена до них. Над конституентами заданої функції необхідно зробити всі можливі склеювання (кожна конституента склеюється з кожною). Це відповідає операціям склеювання і поглинання, так як склеюватися можуть тільки кон'юнкції одного рангу. В результаті виходять кон'юнкції (диз'юнкції)

(n−1)-го рангу, над якими необхідно знову зробити всі можливі операції склеювання, отримавши кон'юнкції (диз'юнкції) (n−2)-го рангу і т.д. Цей Процес триває до отримання мінімальної форми логічної функції. Розглянемо використання методу Квайна на прикладі.

Приклад. Знайти мінімальну форму логічної функції

$$\overset{1}{}\quad\overset{2}{}\quad\overset{3}{}\quad\overset{4}{}\quad\overset{5}{}\quad\overset{6}{}$$
$$f(A,B,C) = \overline{A}BC \vee \overline{A}\overline{B}C \vee ABC \vee AB\overline{C} \vee A\overline{B}C \vee \overline{A}B\overline{C}.$$

Пронумеруємо конституенти.

Провівши почергове склеювання кожної конституенти з усіма подальшими, отримаємо:

$1–2 — \overline{A}C$ (по B) — 1',
$1–3 — BC$ (по A) — 2',
$1–6 — \overline{A}B$ (по C) — 3',
$2–5 — \overline{B}C$ (по A) — 4',
$3–4 — AB$ (по C) — 5',
$3–5 — AC$ (по B) — 6',
$4–6 — B\overline{C}$ (по A) — 7'.

Зробимо подальше склеювання отриманих кон'юнкцій (n−1)-го рангу

1'–6' — C (по A),
2'–4' — C (по B),
2'–7' — B (по C),
3'–5' — B (по A).

В результаті отримуємо мінімальну форму логічної функції

$$f(A,B,C) = C \vee C \vee B \vee B = C \vee B$$

Приклад. Знайти мінімальну форму логічної функції

$$\overset{1}{}\quad\overset{2}{}\quad\overset{3}{}\quad\overset{4}{}\quad\overset{5}{}\quad\overset{6}{}$$
$$f(A,B,C) = (A \vee B \vee \overline{C})(A \vee \overline{B} \vee \overline{C})(\overline{A} \vee \overline{B} \vee C)(A \vee B \vee C)(\overline{A} \vee B \vee C)(\overline{A} \vee \overline{B} \vee \overline{C}).$$

Пронумеруємо конституенти і зробимо всі операції склеювання:

$1–2 — A \vee \overline{C}$ (по B),
$1–4 — A \vee B$ (по C),
$2–6 — \overline{B} \vee \overline{C}$ (по A),
$3–5 — \overline{A} \vee C$ (по B),
$3–6 — \overline{A} \vee \overline{B}$ (по C),
$4–5 — B \vee C$ (по A).

Отримані елементарні диз'юнкції 2-го рангу між собою не склеюються, отже, скорочена кон'юнктивна нормальна форма буде мати вигляд:

$$f(A,B,C) = (A \vee \overline{C})(A \vee B)(\overline{B} \vee \overline{C})(\overline{A} \vee C)(\overline{A} \vee \overline{B})(B \vee C).$$

Елементарні диз'юнкції, що входять до складу скороченої кон'юнктивної нормальної форми, звуться простими імпліцентами. Подальший етап мінімізації полягає в виявленні в складі формули зайвих членів, який зручно проводити за допомогою імпліцентної (для ДДНФ – імплікантної) матриці (табл. 2.6).

Таблиця 2.6

Імпліценти	Конституенти					
	$A \vee B \vee \overline{C}$	$A \vee \overline{B} \vee \overline{C}$	$\overline{A} \vee \overline{B} \vee C$	$A \vee B \vee C$	$\overline{A} \vee B \vee C$	$\overline{A} \vee \overline{B} \vee \overline{C}$
	1	2	3	4	5	6
$A \vee \overline{C}$	*	*				
$A \vee B$	*			*		
$\overline{B} \vee \overline{C}$		*				*
$\overline{A} \vee C$			*		*	
$\overline{A} \vee \overline{B}$			*			*
$B \vee C$				*	*	

Стовпці табл. 2.6 відповідають конституентам заданої функції, рядки – простим імпліцентам (імплікантам) скороченої форми. У ті клітини, які показують, що дана імпліцента (імпліканта) є частиною відповідної констітуенти, ставляться зірочки. Для визначення мінімальних форм заданої логічної функції необхідно вибрати мінімальну кількість імпліцент (ймплікант), що перекривають всі стовпці табл. 2.6.

Перекрити всі стовпці імпліцентної матриці даної функції можна трьома елементарними диз'юнкціями. При цьому існує два варіанти мінімальних форм:

1. $A \vee \overline{C}$ перекриває 1-у та 2-у колонки;
   $\overline{A} \vee \overline{B}$ перекриває 3-ю та 6-у колонки;
   $B \vee C$ перекриває 4-у та 5-у колонки.
2. $A \vee B$ перекриває 1-у та 4-у колонки;
   $\overline{B} \vee \overline{C}$ перекриває 2-у та 6-у колонки;
   $\overline{A} \vee C$ перекриває 3-ю 5-у колонки.

Таким чином, дана функція має дві мінімальні кон'юнктивні форми:

$$f_1(A,B,C) = (A \vee \overline{C})(\overline{A} \vee \overline{B})(B \vee C);$$

$$f_2(A,B,C) = (A \vee B)(\overline{B} \vee \overline{C})(\overline{A} \vee C).$$

Іноді мінімальну форму можна ще дещо спростити за допомогою законів і співвідношень алгебри логіки.

2.4.3. Мінімізація логічних функцій табличним методом

Найбільш зручним методом знаходження мінімальних ДНФ при невеликій кількості змінних (порядка 4-5) є табличний метод або метод площинних діаграм Вейча (карт Карно) [2].

Початковим виразом для мінімізації за допомогою діаграм Вейча є досконала ДНФ логічних функцій. Функцію, що задана в ДДНФ, можна подати за допомогою таблиці, яка складається з клітинок, число яких дорівнює числу конституент одиниці. Кожна клітинка відповідає одній конституенті, причому клітинки розташовуються таким чином, що конституенти суміжних клітинок відрізняються значенням тільки однієї змінної. Ця вимога обумовлена тим, що при такому розташуванні до конституент двох суміжних клітинок можна застосувати формулу склеювання виду $AB \vee A\overline{B} = A$, що приводить до спрощення функції.

У випадку логічної функції двох змінних при записі її в ДДНФ береться логічна сума чотирьох конституент. Отже, діаграма Вейча повинна містити чотири клітинки (за кількістю конституент) і мати такий вигляд і позначення клітинок, як показано на рис. 2.9.

Верхній рядок клітинок відповідає змінній $\overline{A}$, нижній – A; лівий стовпець клітинок відповідає змінній $\overline{B}$, правий – B.

В клітинках вказані двійкові номери наборів змінних. Отже, кожній конституенті, яка є кон'юнкцією двох змінних, відповідає одна клітинка. Внаслідок цього конституенти в діаграмі будуть розташовані відповідно до номерів наборів змінних таким чином (рис. 2.10).

A	B	
	0	1
0	00	01
1	10	11

Рис. 2.9. Діаграма Вейча

A	B	
	0	1
0	$\overline{A}\overline{B}$	$\overline{A}B$
1	$A\overline{B}$	AB

Рис.2.10. Діаграма Вейча для функції двох змінних

Для того щоб на діаграмі подати логічну функцію, задану в ДДНФ, необхідно позначити клітинки, що відповідають тим конституентам одиниці, які утворюють задану функцію. Це можна зробити, записуючи одиниці у відповідні клітинки. Решту клітинок можливо заповнити нулями.

Наприклад, логічна функція трьох змінних виду

$$f(A,B,C) = \overline{A}\overline{B}\overline{C} \vee \overline{A}\overline{B}C \vee \overline{A}B\overline{C} \vee ABC$$

буде подана діаграмою Вейча таким чином (рис. 2.11). Для логічної функції трьох змінних діаграма Вейча має $N = 2^3 = 8$ клітинок.

Характерним для діаграм Вейча функції трьох змінних є те, що нумерація стовпців йде не за порядком зростання їх номерів, а так, як показано на рис.2.11.

A	BC			
	00	01	11	10
0	1	1	0	1
1	0	0	1	0

Рис. 2.11. Діаграма Вейча для функції трьох змінних

В цьому випадку сусідніми є не тільки суміжні по горизонталі, а також і ті, які стоять по краях таблиці. До суміжних клітинок застосовується формула склеювання конституент, яка використовується при мінімізації логічних функцій.

Накінець, покажемо вид діаграми Вейча для функції чотирьох змінних, позначення її стовпців і рядків (рис. 2.12), а також приклад подання в ній функції виду

$$f(A,B,C,D) = \overline{A}\overline{B}\overline{C}\overline{D} \vee \overline{A}\overline{B}CD \vee A\overline{B}C\overline{D} \vee ABC\overline{D} \vee ABCD.$$

AB	CD			
	00	01	11	10
00	1	0	1	0
01	0	0	0	0
11	0	0	1	1
10	0	0	0	1

Рис. 2.12. Діаграма Вейча для функції чотирьох змінних

В діаграмі Вейча для функції чотирьох змінних сусідніми клітинками, крім безпосередньо суміжних, є клітинки як крайніх стовпців, так і крайніх рядків. Принцип нумерації рядків і стовпців залишається попереднім: номери сусідніх клітинок по рядках і стовпцях відрізняються тільки одним розрядом.

Для функції п’яти змінних кількість клітинок буде 32, для шести змінних – 64 і т.д. В таких діаграмах сусідніми клітинками, крім названих, є дзеркально відбиті відносно вертикальної і горизонтальної середніх лінії таблиці. Із цього виходить, що зі збільшенням числа змінних, діаграми Вейча стають дуже

великими і наочними, тому мінімізація функцій табличним методом більш ніж від п’яти – шести змінних є недоцільною.

Розглянемо суть мінімізації логічних функцій з допомогою площинних діаграм Вейча. Суть методу полягає в записі скороченої ДНФ (КНФ) логічної функції шляхом виявлення сусідніх відмічених на діаграмі клітинок і об’єднання суміжних конституент за такими правилами:

1. Якщо дві помічені клітинки діаграми є сусідніми по рядку або по стовпцю, то відповідна їм пара конституент замінюється однією імплікантою (імпліцентою) рангом на одиницю нижче, ніж ранг конституенти, і включає змінні з однаковими показниками інверсування.

2. Якщо чотири відмічені клітинки діаграми є сусідніми або по рядку, або по стовпцю чи утворюють квадрат, то відповідна четвірка конституент замінюється однією імплікантою (імпліцентою) рангом на дві одиниці нижче, ніж ранг конституент, і включає змінні з однаковими показниками інверсування.

Приклад. Мінімізувати логічну функцію трьох змінних

$$f(A,B,C) = \overline{A}\overline{B}C \vee \overline{A}BC \vee AB\overline{C} \vee ABC.$$

Будуємо діаграму Вейча для заданої функції і відмічаємо клітинки, які відповідають конституентам одиниці і нуля, що утворюють дану функцію (рис. 2.13).

A	BC			
	00	01	11	10
0	0	1	1	0
1	0	0	1	1

Рис. 2.13. Діаграма Вейча для функції f(A,B,C)

Находимо сусідні конституенти (обведені пунктиром) і об’єднуючи їх, записуємо скорочену ДНФ логічної функції, яка складається з імплікант другого рангу (оскільки конституенти третього рангу)

$$f(A,B,C) = \overline{A}C \vee BC \vee AB$$

В даному прикладі в результаті мінімізації логічної функції за допомогою діаграми Вейча отримання так звана тупикова ДНФ логічної функції.

Тупиковою ДНФ логічної функції називається така скорочена ДНФ, яка включає диз’юнкції всіх можливих імплікант логічної функції.

Щоб отримати мінімальні ДНФ з тупикових, використовується метод конституентно- імплікантних матриць, за допомогою якого виявляються лишні імпліканти.

Побудуємо конституентно- імплікатну матрицю (рис. 2.14).

Імпліканти	Конституенти			
	$\overline{A}\overline{B}C$	$\overline{A}BC$	$AB\overline{C}$	ABC
$\overline{A}C$	*	*		
BC		*		*
AB			*	*

Рис. 2.14. Імплікантна матриця

Аналізуючи матрицю, можна побачити, що імпліканта BC є лишньою, оскільки дві інші імпліканти охоплюють всі конституенти. Отже, тепер скорочену форму логічної функції можна записати так:

$$f(A,B,C) = \overline{A}C \vee AB.$$

Дана скорочена ДНФ є мінімальною ДНФ, тому що подальше спрощення неможливе.

Якщо логічна функція задана в кон'юнктивній формі, то мінімізація виконується аналогічно, тільки об'єднуються конституенти нуля. Діаграма Вейча дозволяє наочно бачити лишні варіанти склеювань і зразу записувати мінімальну форму логічної функції.

Приклад. Мінімізувати логічну функцію трьох змінних

$$f(A,B,C) = (A \vee B \vee C)(A \vee \overline{B} \vee C)(\overline{A} \vee B \vee C)(\overline{A} \vee B \vee \overline{C}).$$

A	BC			
	00	01	11	10
0	0	1	1	0
1	0	0	1	1

Рис. 2.15. Діаграма Вейча для функції f(A,B,C)

Скорочена форма логічної функції має наступний вигляд

$$f(A,B,C) = (A \vee C)(\overline{A} \vee B).$$

Вміле використання методу мінімізації за допомогою діаграм Вейча може сприяти отриманню більш простих логічних формул при умові, якщо не робити лишніх склеювань.

2.4.4. Мінімізація неповністю заданих логічних функцій

В деяких задачах структурного синтезу вузлів цифрових ЕОМ зустрічаються випадки, коли деякі комбінації вхідних сигналів ніколи не

подаються. Такі комбінації сигналів називаються забороненими. Значення вихідних сигналів, які відповідають забороненим комбінаціям вхідних сигналів, не визначаються і їх можна вибирати довільно, при цьому задані умови функціонування схеми вузла не порушуються [2].

Робота схеми з забороненими комбінаціями вхідних сигналів описується неповністю заданими логічними функціями, значення яких визначені не на всіх наборах змінних. Інколи такі функції називаються частково визначеними або частково заданими.

Для мінімізації неповністю заданих логічних функцій можна використовувати аналітичні та табличні методи, враховуючи при цьому, що на заборонених наборах змінних значення логічних функцій можна вибрати рівними або нулю, або одиниці. На практиці на цих наборах змінних задають такі значення функції, при яких можна отримати найбільш прості формули логічних функцій при їх мінімізації [10].

Приклад. Нехай частково визначена логічна функція чотирьох змінних в процесі мінімізації буде представлена діаграмою Вейча як показано на рис. 2. 16. На наборах 6, 7, 9, 14, 15 функція не визначена, що в діаграмі показано знаком —. Необхідно виконати мінімізацію цієї функції.

Для мінімізації довизначемо функцію як показано на рис. 2. 17.

AB	CD			
	00	01	11	10
00	0	0	1	1
01	1	1	—	—
11	1	0	—	—
10	0	—	1	1

Рис. 2. 16. Діаграма Вейча заданої функції

AB	CD			
	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	0	1	1
10	0	0	1	1

Рис. 2. 17. Діаграма Вейча довизначеної функції

Після довизначення функції одиницями на 6, 7, 14 і 15 наборах, а також нулем на 9 наборі, в результаті мінімізації отримаєм:

$$f(A,B,C,D) = \overline{A}B \vee B\overline{D} \vee C.$$

Недоліком мінімізації неповністю заданих логічних функцій розглянутим методом є необхідність їх до визначення на заборонених наборах. Це значно ускладнює процес мінімізації при збільшенні числа змінних і ускладнює вибір правильного рішення. Вирази скороченої диз'юнктивної або кон'юнктивної нормальної форми логічних функцій не завжди є єдиними. Склад формул залежить від способу довизначення і може містити зайві члени.

2.4.5. Поняття про сумісну мінімізацію логічних функцій

При синтезі цифрових автоматів комбінаційного типу з декількома виходами доводиться описувати його роботу за допомогою не однієї, а кількох логічних функцій. У цьому випадку цілеспрямовано виконувати мінімізацію логічних функцій не окремо, приводячи їх до мінімальних форм, а сумісно. Результат сумісної мінімізації логічних функцій дозволяє використовувати одні і ті ж логічні елементи для формуванні сигналів з різних виходів. При цьому необхідно враховувати факт, що сумісна мінімізація функцій призводить до збільшення глибини схеми

Найбільш наочним і зручним способом сумісної мінімізації логічних функцій є метод площинних діаграм Вейча [10]. Розглянемо сутність даного способу на конкретному прикладі.

Приклад. Здійснити сумісну мінімізацію логічних функцій виду:

$$f_1(A,B,C) = \overline{A}\,\overline{B}\,\overline{C} \vee \overline{A}\,\overline{B}C \vee ABC \vee AB\overline{C};$$

$$f_2(A,B,C) = \overline{A}\,\overline{B}\,\overline{C} \vee \overline{A}\,\overline{B}C \vee ABC \vee \overline{A}B\overline{C}.$$

Побудуємо діаграми Вейча для кожної з цих функцій (рис. 2.18)

A	BC			
	00	01	11	10
0	1	1	0	0
1	0	0	1	1

A	BC			
	00	01	11	10
0	1	1	0	1
1	0	0	1	0

Рис. 2.18. Діаграми Вейча двох функцій f_1 і f_2

З діаграм видно, що функція $f_2(A,B,C)$ відрізняється від функції $f_1(A,B,C)$ наявністю зайвої конституенти I з номером 010 і конституенти нуля з номером 110.

Отже, функцію $f_2(A,B,C)$ можна виразити через функцію $f_1(A,B,C)$ наступним чином:

$$f_2(A,B,C) = f_1(A,B,C) \cdot К_{110}(0) \vee К_{010}(1) = f_1(A,B,C) \cdot (\overline{A} \vee \overline{B} \vee C) \vee (\overline{A}B\overline{C}).$$

Після мінімізації функції $f_1(A,B,C)$ будемо мати:

$$f_1(A,B,C) = \overline{A}\,\overline{B} \vee AB;$$

Таким чином, виразивши одну функцію через іншу, і виконав спрощення логічного виразу, ми здійснили сумісну мінімізацію обох логічних функцій.

Реалізуючи тепер функцію $f_1(A,B,C)$ за допомогою ФПС ЛЕ, ми одночасно реалізуємо і частину функції $f_2(A,B,C)$

$$f_2(A,B,C) = f_1(A,B,C) \cdot (\overline{A} \vee \overline{B} \vee C) \vee (\overline{A}B\overline{C}).$$

2.5. ОСНОВИ СИНТЕЗУ КОМБІНАЦІЙНИХ СХЕМ

Однією з різновидностей комбінаційних схем є вузол цифрових ЕОМ. Звичайно вузол складається з функціональних елементів і виконує задачу з перетворення інформації.

Вузлом *комбінаційного типу* називається вузол, в якому те або інше значення сигналу на виході з'являється при цілком визначеній комбінації значень сигналів на його входах і, як правило, не зберігається при знятті вхідних сигналів [11].

Процес розробки логічних схем, які реалізують задані умови функціонування, являє собою послідовність етапів синтезу і аналізу.

Задача синтезу логічної схеми полягає у визначенні такого способу з'єднання логічних елементів, при якому побудована схема реалізує поставлену задачу з перетворення вхідної двійкової інформації.

Слід відзначити, що задача синтезу розв'язується неоднозначно. Різноманітність форм логічних функцій обумовлює множину логічних схем, що реалізують одну й ту ж логічну функцію. Для того щоб зробити задачу синтезу визначеною, накладають деякі обмеження на спосіб побудови логічної схеми і вводять критерії оптимальності, яким повинна відповідати побудована схема.

Задача аналізу логічних схем складається з оцінки якості синтезованої структурної схеми за певними критеріями. Найбільш часто як критерії оптимальності використовують такі: *глибина схеми* і *структурна складність*, яка визначається ціною за Квайном.

Під глибиною схеми (H) розуміють максимальну кількість послідовно з'єднаних логічних елементів. Глибина схеми визначає її швидкодію. Чим менша глибина схеми, тим менша в ній затримка сигналу, а отже, вища швидкодія.

Ціна за Квайном (C) визначається як загальне число входів усіх логічних елементів, з яких побудована схема, що реалізує дану функцію.

Синтез логічних схем прийнято поділяти на два етапи: *етап абстрактного синтезу* та *етап структурного синтезу*.

На етапі абстрактного синтезу робляться такі операції:

завдання умов функціонування логічної схеми шляхом складання таблиці істинності, яка іноді називається таблицею умов роботи логічної схеми;

отримання за таблицею істинності аналітичних подань логічних функцій у вигляді досконалих диз'юнктивних або кон'юнктивних нормальних форм

На етапі структурного синтезу робляться такі операції:

мінімізація логічних функцій і приведення їх до вигляду, який відповідає найбільш простій їх реалізації за допомогою заданого функціонального повного набору логічних елементів;

побудова за остаточними формулами логічної схеми з врахуванням заданих критеріїв оптимальності.

Приклад. Побудувати схему порівнювання двох дворозрядних двійкових кодів, яка реалізує наступну умову: $N \leq M$. Синтез виконати в базисі "АБО-НІ" (базис Пірса).

Етап абстрактного синтезу

Позначимо розряди порівнювальних кодів: N = (A, B); M = (C, D). Сформулюємо умову функціонування пристрою за допомогою таблиці істинності (табл.2.7).

Таблиця 2.7

A	**B**	**C**	**D**	**f(A,B,C,D)**
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Оскільки за умовою синтез необхідно виконати в базисі Пірса, доцільно функцію f(A,B,C,D) подати в ДКНФ.

$$f(A,B,C,D)=(A\vee\overline{B}\vee C\vee D)(\overline{A}\vee B\vee C\vee D)(\overline{A}\vee B\vee C\vee\overline{D})(\overline{A}\vee\overline{B}\vee C\vee D)(\overline{A}\vee\overline{B}\vee C\vee\overline{D})(\overline{A}\vee B\vee\overline{C}\vee D).$$

Етап структурного синтезу

За допомогою діаграми Вейча (рис. 2.19) виконаємо мінімізацію логічної функції f(A,B,C,D).

AB	CD			
	00	01	11	10
00	1	1	1	1
01	0	1	1	1
11	0	0	1	0
10	0	0	1	1

Рис. 2.19. Діаграма Вейча для функції f(A,B,C,D)

$f(A,B,C,D) = (\overline{A} \vee C)(\overline{B} \vee C \vee D)(\overline{A} \vee \overline{B} \vee D)$.

Використовуючи інверсний закон, перетворимо отриману функцію

$$f(A,B,C,D) = \overline{\overline{(\overline{A} \vee C)(\overline{B} \vee C \vee D)(\overline{A} \vee \overline{B} \vee D)}} = \overline{\overline{(\overline{A} \vee C)} \vee \overline{(\overline{B} \vee C \vee D)} \vee \overline{(\overline{A} \vee \overline{B} \vee D)}}$$

Побудуємо схему пристрою, використовуючи логічні елементи “АБО-НІ” (рис. 2.20).

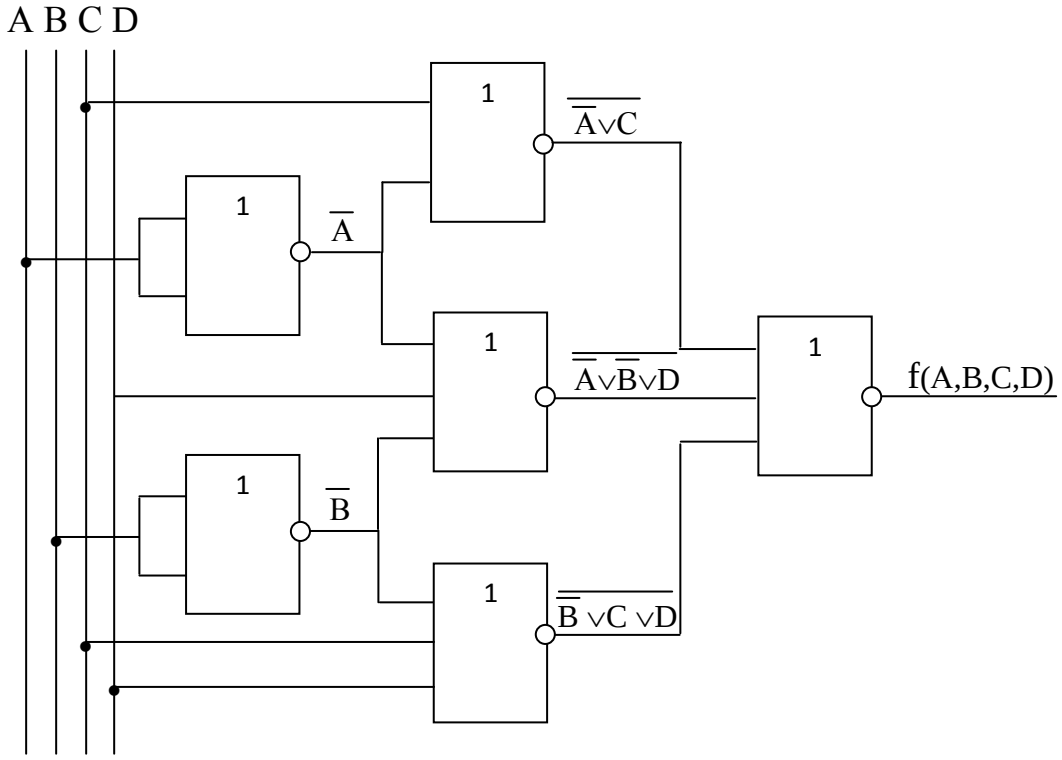


Рис. 2.20. Структурна схема вузла

Виконаємо аналіз побудованої схеми.

Глибина схеми – H = 2. Ціна за Квайном – C = 11.

2.6. ЦИФРОВІ АВТОМАТИ З ПАМ'ЯТТЮ

2.6.1. Загальні відомості про цифрові автомати з пам'яттю

Цифровим автоматом називається дискретний пристрій з пам'яттю. Вихідні сигнали в такому пристрої в данний момент часу залежать не тільки від значень вхідних сигналів, але і від їх попередніх значень. Структурну схему такого автомата можна подати так, як це показано на рис. 2.18.

Вихідні сигнали (Y) визначаються зв'язаною послідовністю перетворень над цифровими даними (X), в яких результати попередніх перетворень Q використовуються в наступних.

В комбінаційній частині (I) автомата здійснюється перетворення інформації за допомогою комбінаційних схем, а в пам'яті (II) зберігається результат перетворення інформації, отриманий на попередньому кроці.

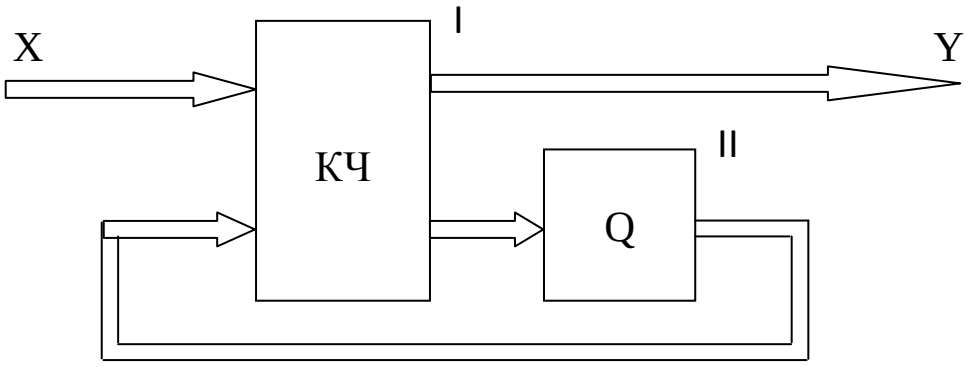


Рис. 2.21. Структурна схема цифрового автомата

Математичною моделлю цифрового автомата з пам'яттю є абстрактний автомат, який задається множиною із шести елементів:

$$A = \{X, Y, Q, \delta, \lambda, q_0\}, \qquad (2.3)$$

де $X = \{x_1, x_2, \ldots, x_n\}$ –множина вхідних сигналів;

$Y = \{y_1, y_2, \ldots, y_n\}$ – множина вихідних сигналів;

$Q = \{q_1, q_2, \ldots, q_n\}$ – множина внутрішніх станів;

δ – функція переходів;

λ – функція виходів;

$q_0 \in Q$ – початковий стан автомата.

Якщо множини X, Y, Q кінцеві, то цифровий автомат називається кінцевим. Далі будуть розглянуті тільки кінцеві автомати [11].

Функції переходів δ і виходів λ описують закон функціонування кінцевого автомата. При цьому функціонування автомата розглядають в дискретні моменти часу 0, 1, 2,…, t, t+1,…

Функція переходів **δ** визначає стан автомата в момент часу (t+1) залежно від стану автомата і значення вхідного сигналу в момент часу t, тобто

$$Q(t+1) = \delta[Q(t), X(t)]. \qquad (2.4)$$

Функція виходів **λ** визначає залежність вихідного сигналу автомата від стану автомата і вхідного сигналу в момент часу t, тобто

$$Y(t) = \lambda[Q(t), X(t)]. \qquad (2.5)$$

Якщо функції (2.4) і (2.5) визначені на всіх значеннях Q(t) та X(t), то такі автомати називають повними або повністю визначеними. На практиці зустрічаються автомати, на входи яких не подаються деякі вхідні сигнали. Функції переходів та виходів таких автоматів визначені не на всіх значеннях Q(t) та X(t) і їх називають частковими або неповністю визначеними.

Функціонування кінцевого автомата полягає в наступному. В початковий момент часу $t = 0$ автомат встановлюється в початковий стан q_0. Після цього на його вхід в дискретні моменти часу 0, 1, 2, …, t, t+1, … подаються вхідні сигнали X(0), X(1), X(2), …, X(t), X(t+1), … Автомат згідно з виразом (2.5) формує послідовність вихідних сигналів.

$$Y(0), Y(1), Y(2), \ldots Y(t), Y(t+1),\ldots$$

а згідно з виразом (2.4) послідовно, починаючи з q_0, переходить в стан

$$q(1), q(2), \ldots, q(t), q(t+1),\ldots$$

В середині інтервалу (t, t+1) значення всіх сигналів і станів не змінюється.

На даний час розрізняють кілька узагальнених структур кінцевих автоматів, названих за іменем вчених, які їх досліджували: автомат Мілі, автомат Мура, автомат Глушкова і т.д. На практиці найбільше застосування отримали автомати Мілі та Мура. Закон функціонування автомата Мілі задається рівняннями (2.4) та (2.5), а автомати Мура рівняннями

$$Q(t+1) = \delta\,[Q(t), X(t)], \qquad (2.6)$$

$$Y(t) = \lambda\,[Q(t)]. \qquad (2.7)$$

Із рівняння (2.7) виходить, що в автоматі Мура вихідні сигнали залежать тільки від стану автомата і не залежить від вхідних сигналів (рис. 2.22).

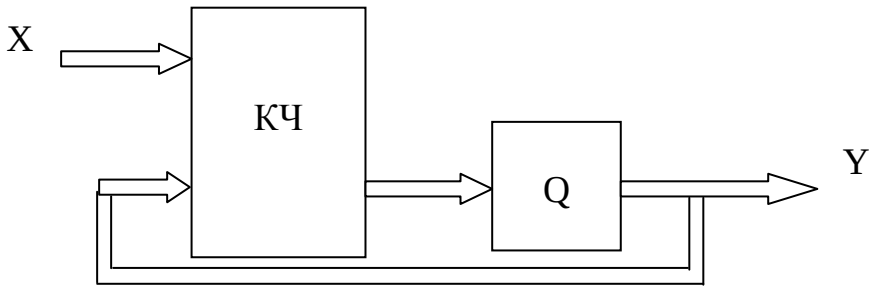


Рис. 2.22. Структурна схема автомата Мура

Варто відзначити, що в останні роки теорія кінцевих автоматів інтенсивно удосконалюється. Великий внесок в теорію кінцевих автоматів внесли радянські вчені В.М.Глушков, М.А.Гаврилов, М.А.Цетлін, а також закордонні вчені Мілі, Мур та інші.

2.6.2. Способи задавання цифрових автоматів

Задавання цифрового автомата полягає в описі елементів множини

$$A = \{X, Y, Q, \delta, \lambda, q_0\}.$$

В теперешній час найбільш широке застосування знайшли три способи задавання цифрових автоматів: табличний, графічний, матричний [11].

При табличному способі задавання автомата Мілі функції переходів $\delta(Q,X)$ та виходів $\lambda(Q,X)$ описуються таблицями переходів і виходів відповідно. Рядки цих таблиць містять всі можливі значення вхідних вигналів, а стовпчики – всі можливі стани автомата. Причому в крайньому лівому стовпчику звичайно розташовуються початкові значення q_0. На перетині рядка X_i та стовпця q_j в таблиці переходів записується стан, в який автомат переходить із стану q_j під дією вхідного сигналу X_i, а в таблиці виходів – вихідний сигнал, який формується при даному переході. Табл. 2.8 та табл. 2.9 ілюструють приклад табличного способу задавання повністю визначеного автомата Мілі із двома вхідними сигналами, чотирма станами і трьома вихідними сигналами.

Таблиця 2.8

Вхідний сигнал	Стан			
	q_0	q_1	q_2	q_3
X_1	q_1	q_2	q_3	q_0
X_2	q_0	q_1	q_2	q_3

Таблиця 2.9

Вхідний сигнал	Стан			
	q_0	q_1	q_2	q_3
X_1	Y_1	Y_2	Y_3	Y_3
X_2	Y_2	Y_3	Y_1	Y_3

Приведені вище таблиці можуть бути об’єднані в одну, яку називають об’єднаною таблицею переходів і виходів(табл. 2.10).

Використовуючи таблиці переходів та виходів, розглянемо порядок функціонування цифрового автомата. Нехай, починаючи з моменту часу t = 0, на вхід цифрового автомата, встановленого в початковий стан q_0, надходить послідовність вхідних сигналів X_1, X_2, X_2, X_1, X_2, X_2. Тоді під дією першого сигналу X_1 в момент часу t = 0 на виході автомата з’явиться вихідний сигнал $Y_1 = \lambda(q_0,X_1)$ і він до моменту часу t = 1 перейде в стан $q_1 = \delta(q_0,X_1)$. При

надходженні другого вхідного сигналу X_2 в момент часу t = 1 вихідний сигнал автомата буде дорівнювати $Y_3 = \lambda(q_1,X_2)$, а стан автомата не зміниться, оскільки з таблиці переходів виходить, що $\delta(q_1,X_2) = q_1$.

Аналізуючи аналогічним способом роботу автомата на задану послідовність вхідних сигналів, отримаємо послідовність вихідних сигналів

$$Y_1\ Y_3\ Y_3\ Y_2\ Y_1\ Y_1$$

і порядок зміни станів автомата

$$q_0\ q_1\ q_1\ q_1\ q_2\ q_2\ q_2.$$

Таблиця 2.10

Вхідний сигнал	Стан			
	q_0	q_1	q_2	q_3
X_1	q_1 / Y_1	q_2 / Y_2	q_3 / Y_3	q_0 / Y_3
X_2	q_0 / Y_2	q_1 / Y_3	q_2 / Y_1	q_3 / Y_3

Для задавання роботи автомата Мура використовується одна таблиця переходів, в якій кожному стовпчику, крім станів, приписується також вихідний сигнал, який відповідає цьому стану. Таку таблицю називають відзначеною таблицею переходів автомата Мура (табл.2.11).

Таблиця 2.11

Вхідний сигнал	Вихідний сигнал			
	Y_1	Y_2	Y_2	Y_3
	стан			
	q_0	q_1	q_2	q_3
X_1	q_1	q_2	q_3	q_0
X_2	q_2	q_3	q_1	q_3

Найбільш наочним способом задавання цифрового автомата є графічний, при якому автомат подається направленим графом. Направлений граф містить множину вершин, які ототожнюються із множиною станів автомата і

множиною направлених дуг (гілок), що вказують переходи автомата із одного стану в інший.

При цьому вершина q_j з'єднується направленою дугою з вершиною q_k, якщо є вхідний сигнал X_1, який забезпечує даний перехід.

В автоматі Мілі дана дуга відзначається вхідним сигналом X_1 та вихідним сигналом Y_j, який формується при цьому переході. Якщо цей перехід забезпечує кілька вхідних сигналів, то даній дузі приписуються всі відповідні вхідні та вихідні сигнали.

На рис. 2.23 приведено граф повністю визначеного автомата Мілі, функціонування якого задано табл. 2.8 та 2.9.

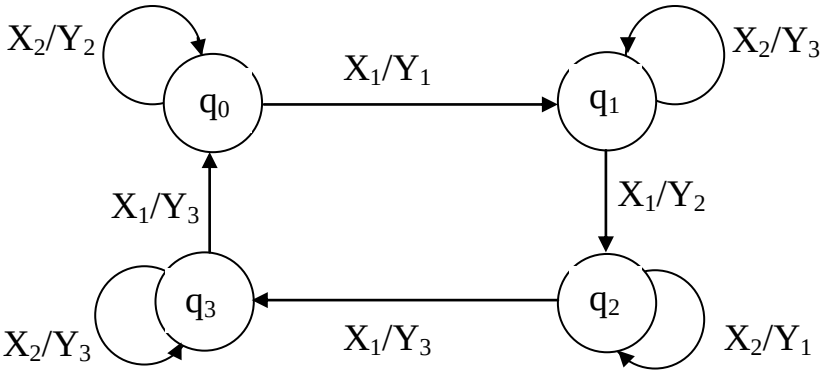


Рис. 2.23. Графічне задавання автомата Мілі

При графічному задаванні автомата Мура вихідні сигнали прийнято записувати всередині або поряд із відповідними вершинами. На рис. 2.24 приведено граф автомата Мура, заданий табл. 2.11.

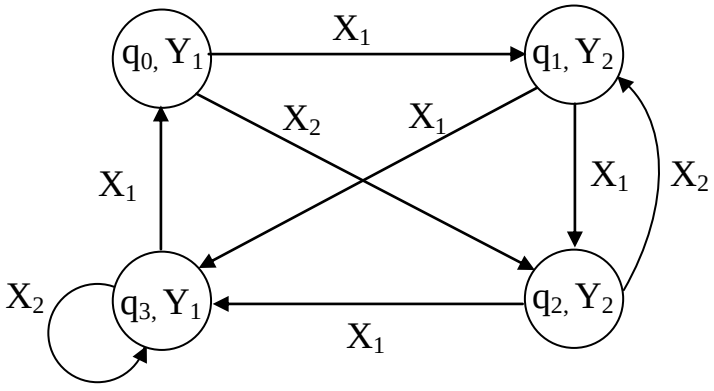


Рис. 2.24. Графічне задавання автомата Мура

2.6.3. Канонічний метод структурного синтезу цифрових автоматів

Описання цифрового автомата за допомогою таблиць і графів не дає можливості аналітичного перетворення функцій виходів і переходів.

Мінімізація цих функцій дозволяє будувати структуру цифрового автомата, тобто здійснювати його синтез.

Процес синтезу складних цифрових автоматів із пам'яттю звичайно ділиться на кілька етапів, основними з яких є абстрактний та структурний етапи [11].

Етап абстрактного синтезу передбачає перш за все опис умов роботи схеми, що проектується за допомогою абстрактного автомата, який заданий одним із розглянутих способів. Цей етап закінчується мінімізацією числа станів абстрактного автомата.

Основною метою *етапу структурного синтезу* є побудова структурної схеми автомата на основі композиції елементарних автоматів. При цьому умови функціонування побудованої схеми повинні відповідати поведінці абстрактного автомата, заданого на етапі абстрактного синтезу. На цьому етапі вхідні та вихідні сигнали, а також стан абстрактного автомата подаються набором значень фізичних сигналів, які кодуються в тій чи іншій системі числення. В сучасних дискретних пристроях стани, вхідні та вихідні сигнали звичайно подаються упорядкованими наборами двійкових змінних. Це дозволяє при синтезі автоматів використовувати розроблений апарат мулевих функцій.

Процес переходу від множини абстрактних вхідних сигналів, вихідних сигналів і станів до упорядкованих наборів двійкових змінних називають кодуванням відповідних значень. При кодуванні кожному із значень X_i, Y_r, і Q_j ставиться у відповідність свій, відмінний від інших, набір значень двійкових змінних. Розрядність кожного з наборів двійкових змінних для X_i, Y_r, Q_j визначається співвідношеннями:

$$n \geq |\log_2 N|,$$
$$m \geq |\log_2 R|,$$
$$l \geq |\log_2 P|,$$

де N – кількість вхідних сигналів;
R – кількість вихідних сигналів;
P – кількість внутрішніх станів;
дужки | | означають найближче більше ціле число.

Варто відзначити, що розрядність для X_i та Y_r визначає число вхідних ***n*** та вихідних ***m*** каналів схеми, що проектується, а число ***l*** визначає число запам'ятовуючих елементів для реалізації різних внутрішніх станів цифрового автомата. В якості запам'ятовуючих елементів використовують так звані елементарні автомати з пам'яттю. Під елементарним автоматом з пам'яттю розуміється автомат, в якому може запам'ятовуватись не більше однієї двійкової змінної або пари взаємно інверсних змінних. При побудові цифрових автоматів з пам'яттю, як елементарні

автомати, найбільш широке застосування знайшли елементи затримки та тригери різних типів. Перш за все відзначимо, що елементарні автомати є автоматами Мура, в яких кожному стану відповідає свій вихідний сигнал. В зв'язку з цим далі внутрішній стан і відповідний йому вихідний сигнал будемо позначати однією і тією ж подвійною змінною Q. Елемент затримки має один вхід і один вихід і зберігає під час одного такту роботи автомата одну двійкову змінну. Тригер може мати кілька двійкових входів і два виходи: прямий і інверсний, тобто він зберігає пару взаємно інверсних змінних.

Пам'ять цифрового автомата складається із елементарних автоматів Мура. Зміна стану пам'яті автомата проводиться за допомогою сигналів $\boldsymbol{q_k}$, які називаються функціями збудження елементарного автомата і формуються комбінаційною схемою цифрового автомата.

При канонічному методі структурного синтезу вважають, що елементарні автомати Мура попередньо вибрані. Тоді задача синтезу зводиться до синтезу комбінаційної схеми цифрового автомата. Комбінаційна схема формує функції збудження, під дією яких відбувається зміна стану пам'яті до наступного такту роботи автомата.

Для отримання значень функцій збудження пам'яті автомата використовується таблиця переходів елемента пам'яті, яка записується у вигляді таблиці функцій збудження елемента пам'яті. В цій таблиці для кожної пари станів { Q(t), Q(t+1) } визначається значення сигналу $\boldsymbol{q_k}$, який забезпечує перехід елементарного автомата із заданого поточного стану Q(t) в потрібний стан в наступному такті Q(t+1).

Таблиці функцій збудження пам'яті деяких елементарних автоматів будуть розглянуті нижче.

На основі викладеного вище можна запропонувати такий порядок дій при канонічному синтезі цифрових автоматів з пам'яттю:

1. Визначити умови роботи схеми, що синтезується, на основі яких скласти математичну модель у вигляді абстрактного кінцевого автомата, заданого таблицями переходів і виходів (направленим графом, матричним способом).
2. Провести двійкове кодування вхідних та вихідних сигналів, внутрішніх станів автомата. Згідно з результатами кодування визначити число вхідних та вихідних каналів, а також число елементарних автоматів пам'яті.
3. Перейти від таблиць переходів та виходів абстрактного автомата до закодованої таблиці переходів та виходів структурного автомата шляхом підставляння в таблиці закодованих значень станів, вхідних і вихідних сигналів.
4. Вибрати елементарні автомати пам'яті та скласти для них двійкові таблиці функцій збудження елементів пам'яті.

5. Доповнити закодовану таблицю переходів та виходів функціями збудження елементів пам'яті. Значення функцій збудження визначаються на основі таблиці переходів автомата і таблиці функцій збудження елементарних автоматів.

6. Визначити мінімальні форми всіх функцій збудження та функцій вихідного сигналу.

7. За отриманими формулами в заданому базисі логічних елементів побудувати структурну схему автомата, що проектується.

Приклад. Потрібно синтезувати автомат-екзаменатор, який ставить оцінку «залік – незалік» по двом відповідям на питання, що видаються у вигляді «так – ні». Якщо обидві відповіді правильні, пристрій ставить «залік», якщо хоч одна відповідь неправильна – ставить «незалік». Після отримання другої відповіді автомат повертається в початковий стан.

Рішення. Задамо умови функціонування автомата за допомогою графа. Очевидно, автомат повинен мати початковий стан, в якому він бере відповіді на перше питання, позначимо його q_0. Відповідей може бути два – правильний і неправильний. Позначимо правильну відповідь – x_1, а неправильну – x_2. Під впливом правильної відповіді автомат переходить в стан q_1, а під впливом неправильної відповіді – в стан q_2. При кожному з переходів формується вихідний сигнал y_1 – «незалік», або y_2 – «залік». Граф, що описує функціонування автомата, наведений на рис. 2.25.

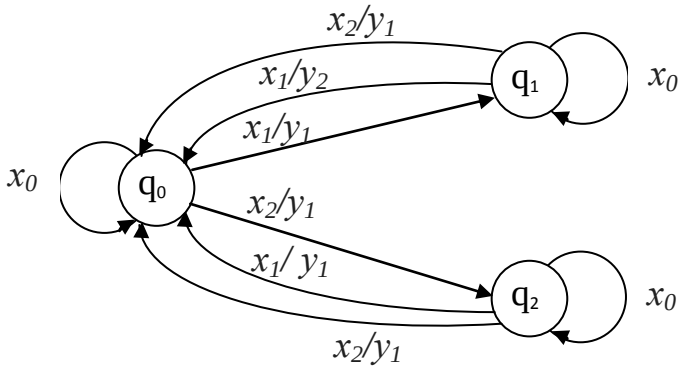


Рис. 2.25. Граф функціонування автомата

Крім сигналів x_1 і x_2 на вхід автомата повинен надходити сигнал, який свідчить про відсутність відповіді, інакше відсутність відповіді буде сприйматися автоматом як неправильна відповідь. Позначимо вхідний сигнал при відсутності відповіді – x_0. На цьому етап абстрактного синтезу закінчується.

Зробимо кодування вхідних, вихідних сигналів і станів синтезованого автомата.

Так як вхідних сигналів і станів автомата по три, то для кодування досить двох двійкових розрядів

x_0 — 0 0; q_0 — 0 0;
x_1 — 0 1; q_1 — 0 1;
x_2 — 1 0; q_2 — 1 0.

Позначимо розряди вхідних сигналів через Z_1 і Z_2, а розряди станів – A і B.

Для кодування вихідних сигналів досить одного розряду $Y_1 - 0$, $Y_2 - 1$.

Зробимо вибір типу і кількість елементарних автоматів. В якості елементарних автоматів найчастіше вибираються автомати Мура. У цьому випадку вихідний сигнал елементарного автомата ототожнюється з його внутрішнім станом. В якості елементарного автомата виберемо тригер типу RS. Тригер має два стани, тому для запам'ятовування будь–якого з трьох станів синтезованого автомата знадобиться два тригера. Для подальшого синтезу знадобляться таблиці функцій збудження елементарного автомата (табл. 2.13), які виходять з таблиці функціонування RS – тригера (табл. 2.12).

Таблиця 2.12

R	S	Q
0	0	Q*
0	1	1
1	0	0
1	1	—

Таблиця 2.13

Q	Q′	q_R	q_S
0	0	—	0
0	1	0	1
1	0	1	0
1	1	0	—

В табл. 2.12 і 2.13 прочерком (—) позначені невизначені стани, Q* – попередній стан, Q' – подальший стан, в який переходить елементарний автомат.

Складемо кодовану таблицю переходів і виходів і доповнимо її функціями збудження елементів пам'яті (табл. 2.14).

Функції збудження елементів пам'яті позначемо q_{RA}, q_{SA} і q_{RB}, q_{SB} відповідно. Значення функцій збудження визначаються на основі переходів автомата, що вказані в кодованій таблиці 2.13 і таблиці переходів елементарного автомата RS – тригера (табл. 2.12).

Значення A', і B', а також Y визначаються з умов функціонування синтезованого автомата (з графа).

Таблиця 2.14

Z_1	Z_2	A	B	A'	B'	Y	q_{RA}	q_{SA}	q_{RB}	q_{SB}
0	0	0	0	0	0	0	—	0	—	0
0	0	0	1	0	1	0	—	0	0	—
0	0	1	0	1	0	0	0	—	—	0
0	0	1	1	—	—	—	—	—	—	—
0	1	0	0	0	1	0	—	0	0	1
0	1	0	1	0	0	1	—	0	1	0
0	1	1	0	0	0	0	1	0	—	0
0	1	1	1	—	—	—	—	—	—	—
1	0	0	0	1	0	0	0	1	—	0
1	0	0	1	0	0	0	—	0	1	0
1	0	1	0	0	0	0	1	0	—	0
1	0	1	1	—	—	—	—	—	—	—

Визначимо мінімальні форми всіх функцій збудження і функції виходу.

Мінімізацію здійснимо за допомогою площинних діаграм Вейча (табл. 2.15 ... 2.19).

Таблиця 2.15

$Z_1 Z_2$	AB			
	00	01	11	10
00	—	—	—	0
01	—	—	—	1
11	—	—	—	—
10	0	—	—	1

$q_{RA} = Z_2 \vee Z_1 A$

Таблиця 2.16

$Z_1 Z_2$	AB			
	00	01	11	10
00	0	0	—	—
01	0	0	—	0
11	—	—	—	—
10	1	0	—	0

$q_{SA} = Z_1\overline{A}\overline{B}$

Таблиця 2.17

$Z_1 Z_2$	AB			
	00	01	11	10
00	—	0	—	—
01	0	1	—	—
11	—	—	—	—
10	—	—	—	—

$q_{RB} = Z_1 \vee Z_2 B$

Таблиця 2.18

$Z_1 Z_2$	AB			
	00	01	11	10
00	0	—	—	0
01	1	0	—	0
11	—	—	—	—
10	0	0	—	0

$q_{SB} = Z_2\overline{A}\overline{B}$

Таблиця 2.19

$Z_1 Z_2$	AB			
	00	01	11	10
00	0	0	—	0
01	0	1	—	0
11	—	—	—	—
10	0	0	—	0

$$Y = Z_2B$$

Зробимо побудову схеми за заданими критеріями оптимальності в обраному базисі логічних елементів.

Для побудови комбінаційної частини схеми задамося базисом "І, АБО, НІ".

Якщо заданий критерій найменшої структурної складності, то необхідно по можливості зробити спільну мінімізацію функцій збудження і вихідних функцій.

Структурна схема сінтезованого автомата приведена на рис. 2.26.

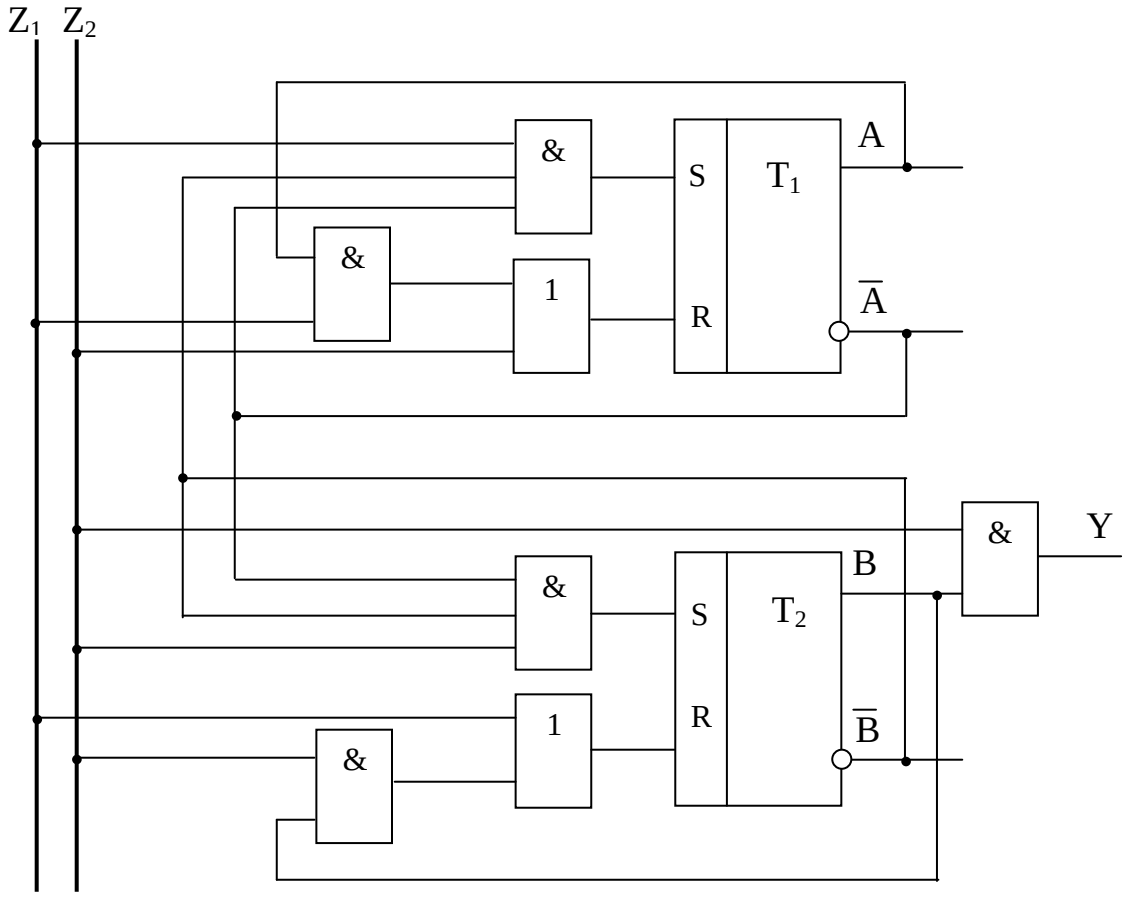


Рис. 2.26. Структурна схема цифрового автомата

КОНТРОЛЬНІ ПИТАННЯ

1. Дайте визначення логічної функції.
2. Скільки різних логічних функцій має функція Y = f(A.B)?
3. Складіть таблицю істинності функції «кон'юнкція».
4. Складіть таблицю істинності функції «диз'юнкція».
5. Складіть таблицю істинності функції «нерівнозначність».
6. Складіть таблицю істинності функції «операція Шеффера».
7. Складіть таблицю істинності функції «операціяПірса».
8. Сформулюйте і поясніть основні закони алгебри логіки.
9. У якому з основних базисів представлено виразF= $x_1\overline{x}_2 + x_3x_4$?
10. Сформулюйте і поясніть правило «склеювання» для логічного додавання і логічного множення.
11. Сформулюйте і поясніть правило «поглинання» для логічного додавання і логічного множення.
12. Назвіть найбільш вживані форми подання логічних функцій.
13. Яка форманазивається нормальною формою булевої функції.
14. Дайте визначення і поясніть поняття «конституента одиниці».
15. Дайте визначення і поясніть поняття «конституента нуля».
16. Дайте визначення і поясніть поняття «досконала диз'юнктивна нормальна форма (ДДНФ) логічної функції».
17. Дайте визначення і поясніть поняття «досконала кон'юнктивна нормальна форма (ДКНФ) логічної функції».
18. В чому полягає процес мінімізації логічних функцій?
19. Які існують способи мінімізації логічних функцій?
20. З якою метою використовують закон Де-Моргана при синтезі логічних схем?
21. Поясніть спосібмінімізації логічних функцій методом Квайна.
22. В чому полягає сутність табличного способу мінімізації логічних функцій?
23. Яке правило покладено в основу мінімізаціїлогічних функцій за допомогою діаграм Вейча?
24. Яку функцію виконує конституентно-імплікантна (імпліцентна) матриця?
25. Як виконується мінімізація неповністю заданих логічних функцій?
26. Поясніть і наведіть приклад сумісної мінімізації логічних функцій.
27. Поясніть порядок синтезу комбінаційних схем.
28. Які критерії оптимальності використовують для аналізу синтезованих схем?
29. Який пристрій називається цифровим автоматом з пам'яттю?
30. Яку функцію виконує комбінаційна частина цифрового автомата з пам'яттю?

31. Поясніть сутність математичної моделі цифрового автомата з пам’яттю.
32. Що визначає функція переходів цифрового автомата з пам’яттю?
33. Що визначає функція виходів цифрового автомата з пам’яттю?
34. Як задаються умови функціонування абстрактного цифрового автомату з пам’яттю?
35. Поясніть табличний спосіб задавання цифрового автоматаз пам’яттю.
36. Чому відповідають строки і стовпці при табличному способі задавання цифрового автомата з пам’яттю?
37. Поясніть графічний спосіб задавання цифрового автомата з пам’яттю.
38. Як будується граф функціонування цифрового автомата?
39. Яку інформацію містить дуга, яка з’єднує вершини направленого графа?
40. Який з логічних пристроїв є елементарним автоматом?
41. Для чого використовуються функції збудження елементарного автомату?
42. Що є метою етапу абстрактного синтезу цифрового автомата?
43. Які дії виконуються на етапі структурного синтезуцифрового автомата?
44. Як визначається число вхідних і вихідних каналів цифрового автомата?
45. Від чого залежить кількість тригерів, необхідних для побудови цифрового автомата?

РОЗДІЛ 3

ФУНКЦІОНАЛЬНІ ВУЗЛИ ЦИФРОВИХ ЕЛЕКТРОННИХ ОБЧИСЛЮВАЛЬНИХ МАШИН

3.1. ЗАГАЛЬНІ ВІДОМОСТІ ПРО СИСТЕМИ ЕЛЕМЕНТІВ ТА ЇХ ХАРАКТЕРИСТИКИ

Під системою елементів розуміється комплекс розрахованих на спільну роботу елементів, із яких може бути построєна будь яка схема ЕОМ.

У зв'язку з тим, що схеми ЕОМ оперують з двійковими символами, всі елементи з яких вони складаються отримали назву двійкових логічних елементів.

Відповідно з Держстандартом − двійковий логічний елемент – елемент, пристрій або функціональна група, реалізують функцію або систему функцій двійкової алгебри логіки, наприклад: логічний елемент І, тригер, цифровий елемент затримки, дешифратор, суматор і т.д.

До двійкових логічним елементам умовно відносяться також елементи, не виконуючі логічні функції, але ті що використовуються в логічних цілях з схемотехнічних міркувань, наприклад: посилювач, генератор, формувач і т.д.

При аналізі або синтезі логічних схем термін «елемент» відображає просту схему, яка реалізує логічну або допоміжну функцію над двійковими змінними або функцію їх запам'ятовування.

При технічному застосуванню вузлів і пристроїв обчислювальної машини термін «елемент» відображає неподільну конструктивну одиницю ЕОМ, яка може складатися з декількох логічних, допоміжних і запам'ятовувальних елементів і виконувати певні логічні функції.

Поняття «елемент» з логічної і конструктивної точки зору має тенденцію до розбіжності, так як досягнений в наший час рівень технології виробництва інтегральних мікросхем дозволяє виготовляти у виді неподільної конструктивної одиниці схеми, реалізуючи достатньо складні логічні функції, включаючи функції центрального процесора цифрової ЕОМ. Такі мікросхеми отримали назву мікропроцесорів.

По функціям, що виконуються при переробці інформації, елементи діляться на :

двійкові логічні елементи комбінаційного типу;

двійкові логічні елементи, які спроможні запам’ятовувати і зберігати інформацію;

допоміжні елементи.

Двійкові логічні елементи комбінаційного типу - це елементи, без пам’яті і які реалізують логічні функції або системи логічних функцій. Комбінаційними ці елементи називаються через те, що значення вихідного сигналу у них визначається комбінацією вхідних сигналів в момент приходу вихідного сигналу.

Двійкові логічні елементи, які спроможні запам’ятовувати і зберігати інформацію бувають двох типів. До першого типу відносяться елементи, в яких представлення інформації зв’язано з зміною електричного стану елемента (наприклад, тригери). До другого типу відносяться елементи, в яких уявлення інформації зв’язано з зміною фізичного стану елемента (наприклад, магнітні елементи);

Допоміжні елементи призначені для посилення, формування, генерації, індикації двійкових сигналів і т.д.

По способу представлення інформації або по виду сигналів на входах і виходах розрізняють імпульсні схеми, потенціальні і імпульсно-потенціальні.

В імпульсних системах інформація на вході і виході елементів є імпульсами напруги або струму; в потенціальних – наявність або відсутність відповідного рівня сигналу; в імпульсно – потенціальних – як імпульсами, так і потенціальними рівнями.

По технології виготовлення розрізняють двійкові логічні елементи, виготовлені з дискретних деталей (резисторів, конденсаторів, напівпровідникових приборів і т.д.), і елементи, виготовлені методами інтегральної технології.

Для раціонального будування вузлів і пристроїв ЕОМ системи елементів повинні відповідати деяким вимогами, які зумовлюють основні характеристики цих систем. Основними вимогами, яким повинні задовольняти системи елементів ЕОМ, є:

- повність системи елементів;
- сумісність вхідних і вихідних сигналів;
- здатність навантаження;
- швидкодія;
- завадостійкість;
- величина споживаної потужності;
- надійність.

Крім цього, пред’являються вимоги до мінімальності набору елементів, ефективності використання обладнання, побудованого на цій системі елементів,

технологічності, конструктивного оформлення, а також спеціальні вимоги: стійкість до збільшення температури, трясіння, вологості, радіації. Всі ці спеціальні вимоги зазвичай враховуються при визначеності надійності, так як ускладнення умов експлуатації приводить до збільшення інтенсивності відмов.

Розберемо коротко основні з цих вимог.

Повність системи елементів. Під логічною повністю розуміють здібність системи елементів реалізувати будь яку логічну функцію. Це поняття зв'язано з поняттям функціонально повних систем логічних функцій.

Сумісність вхідних і вихідних сигналів полягає в забезпеченні таких вхідних і вихідних сигналів, при яких кожний елемент може працювати на один або кілька входів інших елементів без застосування будь яких узгоджувальних ланцюгів.

Навантажувальна здатність визначається коефіцієнтом розгалуження по виходу і показує на яку кількість входів других елементів може бути навантажений цей елемент без зміни форми сигналу.

Швидкодія елемента визначається часом його переключення, тобто переходу елемента із стану I в стан 0 і навпаки. Швидкодія залежить від якості і режимів роботи елементів, величини паразитних ємностей, параметрів вхідних імпульсів і т.д.

Завадостійкість визначає степінь нечутливості елементів до впливу різного роду перешкод. Запас завадостійкості визначається величиною постійної напруги, додавання які до теперішнього рівня вхідної напруги не змінює значення сигналу.

Величина споживаної потужності визначає тепловий режим роботи елементів. Чим більше споживана потужність, тим більше нагрів і гірші умови роботи елементів.

Надійність є характеристикою якості і визначається як властивість елемента виконувати задані функції в визначених умовах експлуатації.

Одним з основних понять, що визначають кількісну оцінку надійності, є поняття «відмова». Під відмовою розуміють стан елемента, при якому він повністю або частково не виконує задані функції, або один чи декілька його параметрів мають значення, які виходять за встановлені норми. Відмова є випадковою подією, через це надійність елементів визначається ймовірністю безвідмовної роботи в період деякого відрізку часу.

Важливість тої або іншої характеристики елементів, яка визначається розглянутими вимогами, залежить від вимог до параметрів пристроїв, які потребується на цих елементах реалізувати.

В даному посібнику системи елементів розглядаються з точки зору їх використання для побудови схем різних вузлів цифрових ЕОМ.

3.2. ТРИГЕРИ

Тригер є елементарний закінчений автомат з пам'ятю, який має два стійких стани. Один з цих станів кодується цифрою 1, другий – цифрою 0. Тригер, як правило, має два виходи – прямий, позначимо його Q, і інверсний – $\overline{Q}$. Стан тригера прийнято визначати по значенню потенціалу на прямому виході, який часто називають одиничним. Домовимся одиничним виходом рахувати вихід, на якому є високий потенціал, коли тригер знаходиться в одиничному стані.

Складність і різновидність задач, вирішуваних тригерами в схемах ЕОМ, привели до великої кількості варіантів їх реалізації, які відрізняються між собою багатьма ознаками.

3.2.1. Класифікація схем тригерів

По особливостям логічного функціонування розрізняють:

- тригери з роздільними входами установки станів 0 і 1 (RS-тригери);
- тригери з рахунковим входом (Т – тригери);
- тригери з прийомом інформації по одному входу, тригери затримки (Д-тригери);
- універсальні тригери з роздільним встановленням станів 0 і 1 (JK – тригери);
- тригери з керованим прийомом інформації по одному входу (DV – тригери);
- комбіновані тригери (RST, JKRS, DRS – тригери і т.д.);
- тригери з складною вхідною логікою.

Існують і другі типи тригерів.

По способу зберігання інформації і кодування вихідних сигналів розрізняють статичні і динамічні тригери.

В статичних тригерах вихідні сигнали, відповідні коду 0 і 1, визначаються різними рівнями напруги. В динамічних тригерах код 1 фіксується наявністю послідовності імпульсів на виході тригера, а код 0 – відсутністю імпульсів. При цьому інформація зберігається в виді імпульсів, циркулюючих по замкнутому контуру тригера.

Статичні тригери по способу запису інформації діляться на дві групи: асинхронні, в яких зміни стану відбувається при подачі сигналів на інформаційні входи, і синхронні (або тактовані), в яких стан змінюється при подачі синхронних сигналів (у відповідності з сигналами на інформаційних входах).

По характеру вхідних і вихідних сигналів тригери діляться на імпульсні, імпульсно-потенціальні і потенціальні.

Умови функціонування тригерів і їх умовне графічне зображення, задаються Держстандартом. Відповідно з цим функціональне призначення входів тригерів наступні:

S – (set – установка) – для роздільної установки тригера в стан 1;

R – (reset – скидання або переустановлення) – для установки тригера в стан 0;

J – для установки універсального JK – тригера в стан 1;

K – для установки універсального JK – тригера в стан 0;

T – рахунковий вхід (від trigger – перемикач);

D – інформаційний вхід для встановлення тригера в стан 1 і 0 (від delay – затримка);

V – підготовчий керуючий вхід для дозволу прийому інформації;

C – виконуючий керуючий вхід для здійснення прийому інформації (вхід синхронізації).

При необхідності до букв дозволяється добавляти цифри.

Розглянемо особливості функціонування і методику побудови схем найбільше поширених типів тригерів.

3.2.2. Синтез RS - тригера

Таблиця станів для тригеру ***RS*** має наступний вигляд:

S	R	Q
0	0	Q*
0	I	0
I	0	I
I	I	—

Тут використано наступні символи:

S і ***R*** – сигнали, діючі на входах тригера;

Q – стан тригера;

Q* – стан тригера у попередній момент часу;

— – не визначено.

Умовне графічне зображення:

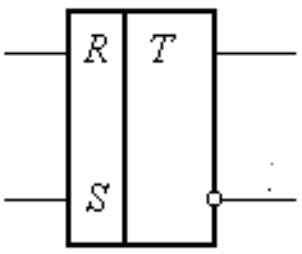


Рис. 3.1. Умовне графічне зображення ***RS*** – тригера

З таблиці станів слідує, що при нульових вхідних сигналах стан тригера не змінюється. При подачі одиничного сигналу на вхід $\boldsymbol{R}$ тригер переходить в нульовий стан, а при подачі одиниці на вхід $\boldsymbol{S}$ він переходить в одиничний стан. Одночасна подача двох одиничних сигналів на входи $\boldsymbol{S}$ і $\boldsymbol{R}$ є забороненою, так як при такій комбінації вхідних сигналів стан тригера не визначений [13].

Відповідно до методики синтезу кінцевих автоматів, викладеної в попередньому розділі, необхідно перед усім побудувати кодовану таблицю переходів та виходів синтезованого автомату. Здійснимо кодування вхідних сигналів і станів тригера $\boldsymbol{RS}$. Що стосується кодування вихідних сигналів, то, як відзначалось вище, тригер функціонує як автомат Мура і його вихідний сигнал однозначно визначається станом, в якому знаходиться тригер.

Вхідних сигналів два – позначимо їх як $\boldsymbol{X_S}$ та $\boldsymbol{X_R}$. Кожен з них може приймати два значення: 0 або I, тому досить одного двійкового розряду для кодування кожного вхідного сигналу. Ці самі міркування про кількість двійкових розрядів відносяться і до станів тригера. Позначимо стан тригера в момент часу $\mathbf{t}$ — $\mathbf{Q}$, а в момент часу $\mathbf{t+1}$ — $\mathbf{Q'}$.

В якості елементарного автомату виберемо елемент затримки. Для визначення значень функції збудження елементарного автомату необхідно використати матрицю переходів елемента затримки. Матриця переходів елементів затримки має наступний вигляд:

$\mathbf{Q}$	$\mathbf{Q'}$	$\mathbf{q_{ЕЗ}}$
0	0	0
0	I	I
I	0	0
I	I	I

Побудуємо таблицю кодованих переходів та виходів $\boldsymbol{RS}$- тригера (табл. 3.1):

Таблиця 3.1.

$\mathbf{X_S}$	$\mathbf{X_R}$	$\mathbf{Q}$	$\mathbf{Q'}$	$\mathbf{q_{ЕЗ}}$
0	0	0	0	0
0	0	I	I	I
0	I	0	0	0
0	I	I	0	0
I	0	0	I	I
I	0	I	I	I
I	I	0	-	-
I	I	I	-	-

Значення $\mathbf{Q'}$визначаються з таблиці станів $\boldsymbol{RS}$- тригера.

Доповнимо таблицю переходів та виходів стовпчиком значень функції збудження елементарного автомату. Таку таблицю часто називають узагальненою таблицею переходів, виходів та функцій збудження.

У зв'язку з тим, що автомат, який синтезується має тільки два стани, очевидно, що досить мати один елементарний автомат для запам'ятовування всіх його станів.

Визначаємо мінімальну форму функції збудження, розглядаючи її як функцію, аргументами якої являються розряди кодів та станів.

В залежності від того, з використанням якого функціонально повного набору логічних елементів вимагається побудувати схему автомата, повинна бути отримана мінімальна диз'юнктивна або кон'юнктивна нормальна форма.

Нехай потрібно синтезувати тригер ***RS*** в базисі "АБО-НІ". Для мінімізації використаємо діаграму Вейча. Доповнимо функцію на невизначених наборах нулями та отримаємо мінімальну кон'юнктивну нормальну форму функції збудження.

X_S	X_RQ			
	00	01	11	10
0	0	1	0	0
1	1	1	—	—

$$q_{E3} = \overline{X}_R(X_S \vee Q). \qquad (3.1)$$

Для реалізації функції в базисі "АБО-НІ" перетворимо вираз (3.1):

$$q_{E3} = \overline{\overline{\overline{X}_R(X_S \vee Q)}} = \overline{X_R \vee \overline{(X_S \vee Q)}}. \qquad (3.2)$$

Схема тригера буде мати вигляд:

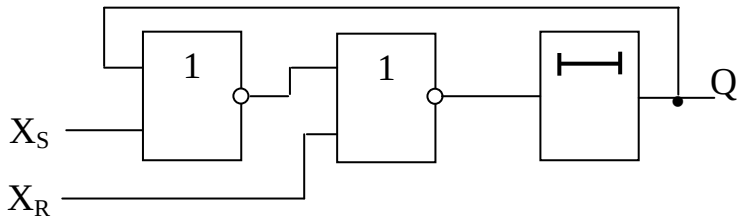


Рис. 3.2 Схема *RS*-тригера в базисі "АБО-НІ"

З урахуванням того, що для запам'ятовування стану схеми вистачить затримки в логічних елементах, можна виразити схему як показано на рис. 3.3.

Така схема симетричного ***RS***-тригера отримала назву бістабільної схеми , і вона часто використовується в якості елементарного автомата для побудови схем інших видів тригерів.

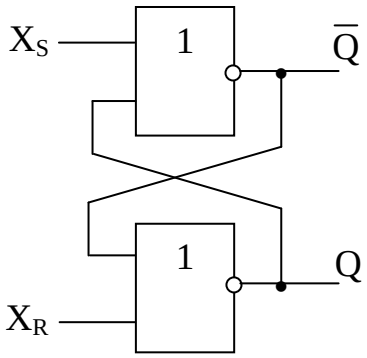


Рис. 3.3. Схема *RS*-тригера в базисі “АБО-НІ”

Для того щоб побудувати схему ***RS***-тригера в базисі “І-НІ”, необхідно отримати мінімальну диз’юнктивну нормальну форму функції збудження і її перетворити:

$$q_{E3} = X_S \vee \overline{X}_R Q = \overline{\overline{X_S \vee \overline{X}_R Q}} = \overline{\overline{X}_S \vee \overline{\overline{X}_R\, Q)}}. \qquad (3.3)$$

В цьому випадку схема буде мати такий само вид, як на рис. 3.3, але на входи повинні подаватись інверсні сигнали:

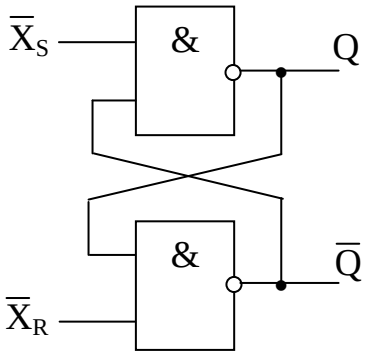


Рис. 3.4. Схема *RS*-тригера в базисі “І-НІ”

На основі асинхронного ***RS***-тригера можна збудувати схему синхронного або тактируємого ***RS***-тригера. Схема такого тригера має вигляд:

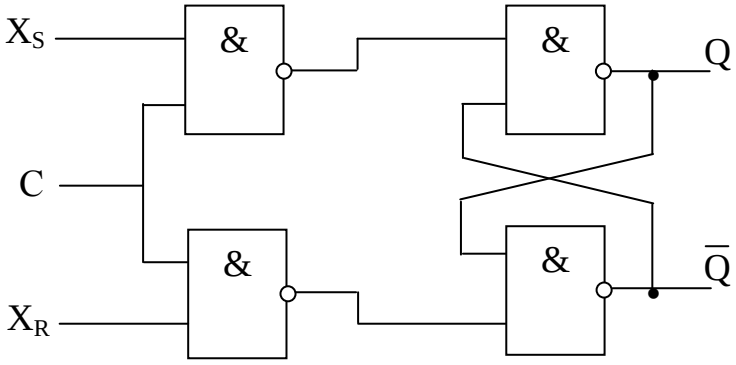


Рис. 3.5. Схема тактируємого *RS*-тригера

Умовне графічне позначення такого тригера показано на рис. 3.6.

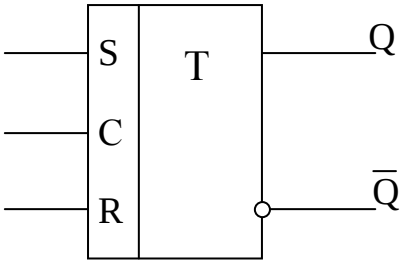


Рис 3.6 Умовне графічне позначення
синхронного *RS*-тригера

В таблиці 3.2 приведені стани синхронного ***RS***-тригера при різних комбінаціях сигналів на його входах.

Таблиця 3.2

C	S	R	Q	$\overline{Q}$
0	—	—	Q*	$\overline{Q}$*
1	0	0	Q*	$\overline{Q}$*
1	0	1	0	1
1	1	0	1	0
1	1	1	—	—

Перший і другий рядок таблиці станів синхронного ***RS***-тригера відповідають *режиму зберігання* (символ «*»), тому що при будь-якій комбінації сигналів на входах ***S*** і ***R*** (див. символи «—» в першому рядку) і нульових значеннях на входах ***S*** і ***R*** на виходах елементів "І-НІ" першого ступеня схеми формуються дві логічні «1».

При одиничному логічному рівні на *вході синхронізації С* схема функціонує як звичайний ***RS***-тригер.

Останній рядок таблиці станів (на всі входи подані логічні «1») відповідає режиму «розорваних зворотніх зв'язків».

3.2.3. Синтез T - тригера

Згідно таблиці станів, зображеній на рис. 3.7, ***T***-тригер змінює свій стан на протилежний при подачі на його вхід одиничного сигналу [13].

X	**Q**
0	Q*
1	$\overline{Q}$*

Рис. 3.7. Таблиця станів *T*-тригера

Умовне графічне позначення ***T***-тригера наступне:

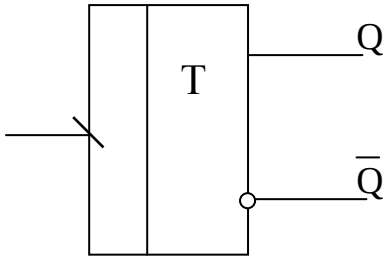


Рис. 3.8. Умовне графічне позначення *T*-тригера

T-тригер ще називають тригером з лічильним входом. Тригер змінює свій вихідний стан по фронту імпульса на вході. Похила риска вказує напрямок фронту вхідного імпульсу (із логічної «1» в «0»).

T-тригер можна реалізувати на основі тригера будь-якого типу.

Синтезуємо ***T***-тригер на основі ***RS***- тригера. Виберемо в якості елементарного автомата бістабільну схему синтезовану в базисі "АБО-НІ". Матриця переходів такої схеми має вигляд, зображений таблицею 3.3.

Таблиця 3.3

Q	**Q′**	q_R	q_S
0	0	–	0
0	1	0	1
1	0	1	0
1	1	0	–

Закодована таблиця переходів тригера типу ***T*** буде мати вигляд:

Таблиця 3.4

X	**Q**	**Q′**	q_R	q_S
0	0	0	–	0
0	1	1	0	–
1	0	1	0	1
1	1	0	1	0

Для отримання мінімальних форм функцій збудження довизначимо їх нулями. Тоді:

$$q_R = X\overline{Q};$$

$$q_S = XQ.$$

Схема тригера, побудована за отриманими виразами, зображена на рис. 3.9.

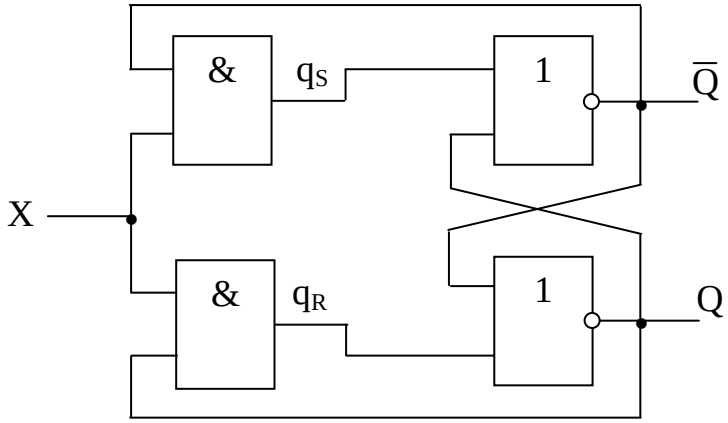


Рис. 3.9. Схема *T*-тригера

Якщо в якості елементарного автомата вибрати бістабільну схему на елементах “І-НІ”, то з врахуванням того, що на її входи подаються інверсні сигнали, функції збудження запишуться у вигляді:

$$\bar{q}_R = \overline{XQ};$$

$$\bar{q}_S = \overline{X\bar{Q}}.$$

Структурна схема тригера типу ***T***, побудована в базисі “І-НІ”, зображена на рис. 3.10.

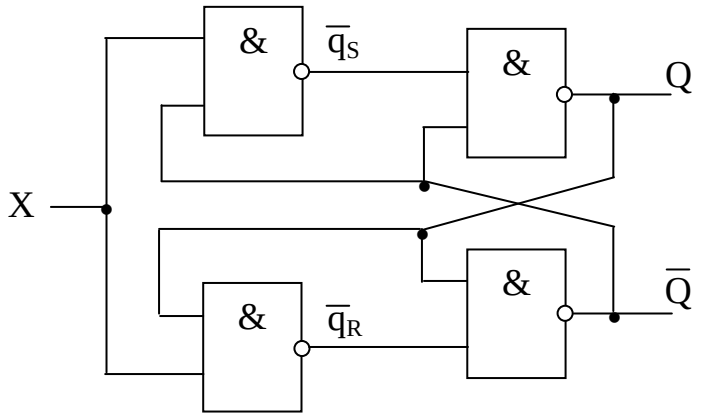


Рис. 3.10. Схема *T*-тригера в базисі “І-НІ”

Для побудови ***T***-тригерів потрібно мати на увазі той файкт, що тривалість імпульсів **X**, поступаючих на рахунковий вхід тригера не повинна перевищувати тривалості перехідних процесів в тригері. Це необхідно для того, щоб імпульс закінчився до моменту надходження інформації про зміни стану тригера.

З іншого боку, імпульси мають володіти достатньою енергією, щоб змінити стан тригера, що при заданій амплітуді імпульсів веде до обмеження мінімальної тривалості.

Для забезпечення вказаних умов з'являється необхідність в затримці проходження сигналів по ланцюгам тригера. Роль такої затримки може грати або дійсна затримка в логічних елементах, або спеціально введені елементи затримки

3.2.4. Синтез комбінованого тригера типу RST

RST – тригер має три входи: ***S*** і ***R*** такі ж як у ***RS***- тригера – вони служать для роздільної установки тригера в одиничний та нульовий стан, а при подачі одиничного сигнала на вхід ***T***-тригера, він змінює свій стан на протилежний. Одночасна подача двох і трьох сигналів являється забороненою комбінацією.

Позначимо вхідні сигнали тригерів $\mathbf{X_S}$, $\mathbf{X_T}$ и $\mathbf{X_R}$ та складемо кодовану таблицю переходів (табл. 3.5):

Таблиця 3.5

$\mathbf{X_S}$	$\mathbf{X_T}$	$\mathbf{X_R}$	**Q**	$\mathbf{Q'}$	q_R	q_S
0	0	0	0	0	-	0
0	0	0	1	1	0	-
0	0	1	0	0	-	0
0	0	1	1	0	1	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	-	-	-
0	1	1	1	-	-	-
1	0	0	0	1	0	1
1	0	0	1	1	0	-
1	0	1	0	-	-	-
1	0	1	1	-	-	-
1	1	0	0	-	-	-
1	1	0	1	-	-	-
1	1	1	0	-	-	-
1	1	1	1	-	-	-

Стан $\mathbf{Q'}$ визначаємо з умов функціонування тригера, тому у випадках, коли на входах працюють два і три сигнала, $\mathbf{Q'}$ не визначено.

В якості елементарного автомата вибираємо бістабільну схему на елементах "АБО-НІ". Доповнюємо кодовану таблицю значеннями сигналів збудження, використовуючи для їхнього визначення матрицю переходів бістабільної схеми.

Мінімальні форми функцій збудження отримаємо за допомогою площинних діаграм Вейча.

Для q_R діаграма Вейча має вигляд:

$X_S\ X_T$	$X_R\ Q$			
	00	01	11	10
00	—	0	1	—
01	0	1	—	—
11	—	—	—	—
10	0	0	—	—

$$q_R = X_T Q \vee X_R Q = Q(X_T \vee X_R).$$

Для $\boldsymbol{q_S}$ діаграма Вейча має вигляд:

$X_S\ X_T$	$X_R\ Q$			
	00	01	11	10
00	0	—	0	0
01	1	0	—	—
11	—	—	—	—
10	1	—		—

$$q_S = X_T \overline{Q} \vee X_S \overline{Q} = \overline{Q}(X_T \vee X_S).$$

Для реалізації схеми тригера в базисі “АБО-НІ” перетворемо отримані мінімальні форми наступним чином:

$$q_R = \overline{\overline{Q(X_T \vee X_R)}} = \overline{\overline{Q} \vee \overline{(X_T \vee X_R)}};$$

$$q_S = \overline{\overline{Q}(X_T \vee X_S)} = \overline{Q \vee \overline{(X_T \vee X_S)}}.$$

Побудуємо схему згідно з отриманими виразами (рис. 3.11):

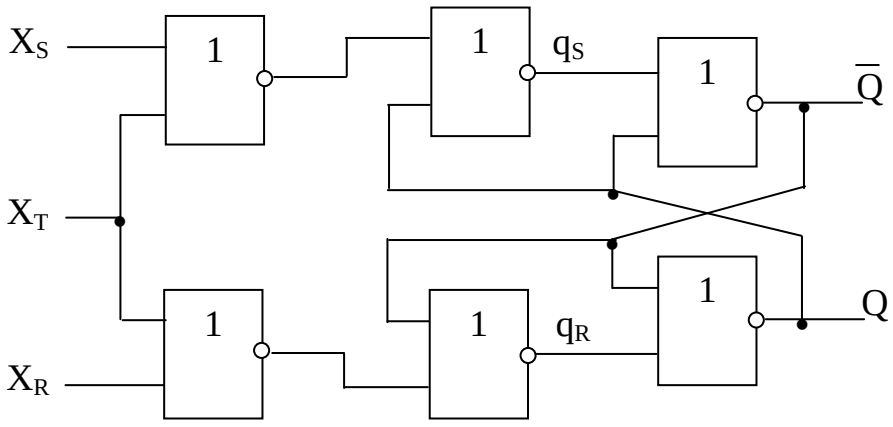


Рис. 3.11. Схема *RST*-тригера в базисі “АБО-НІ”

3.2.5. Синтез D - тригера

D-тригером називається елемент з двома стійкими станами та одним інформаційним входом. Тригери типу ***D*** зазвичай використовують при побудові складних вузлів накопичуючого типу в якості елемента затримки сигналів на один такт з метою підвищення стійкості функціонування цих вузлів. ***D***-тригери виконуються зазвичай синхронними, тому умовне графічне позначення тригера наступне (рис.3.12):

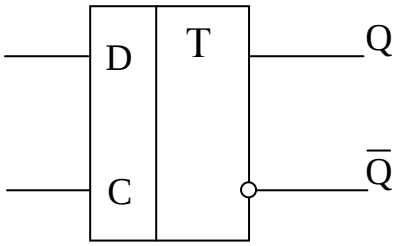


Рис. 3.12. Умовне графічне позначення *D*-тригера

Функціонування ***D***-тригера визначається наступною таблицею станів:

Таблиця 3.6

X_D	X_C	Q
0	0	Q*
0	1	0
1	0	Q*
1	1	1

Із таблиці видно, що стан тригера в момент часу ***t+1*** визначається сигналом, поданим на інформаційний вхід ***D*** в момент часу ***t*** (при наявності сигналу 1 на вході ***C***).

Складемо закодовану таблицю переходів і виходів ***D***-тригера (табл.3.8).

Обираємо в якості елементарного автомату бістабільну схему на елементах “І-НІ” з урахуванням того, що на входи такої схеми повинні подаватися інверсні значення вхідних сигналів. Матриця переходів елементарного автомата має вигляд (табл. 3.7):

Таблиця 3.7

Q	Q′	$\overline{q_R}$	$\overline{q_S}$
0	0	–	1
0	1	1	0
1	0	0	1
1	1	1	–

Для складання узагальненої таблиці переходів, виходів та сигналів збудження можна скористатися матрицею переходів для бістабільної схеми на елементах “АБО-НІ”, але в цьому випадку будуть отримані інверсні значення функцій збудження.

Закодована таблиця переходів, виходів та сигналів збудження ***D***-тригера маэ наступний вигляд:

Таблиця 3.8

X_D	X_C	Q	Q′	$\overline{q_R}$	$\overline{q_S}$
0	0	0	0	−	1
0	0	1	1	1	−
0	1	0	0	−	1
0	1	1	0	0	1
1	0	0	0	−	1
1	0	1	1	1	−
1	1	0	1	1	0
1	1	1	1	1	−

Мінімальні форми функцій збудження отримаємо за допомогою площинних діаграм Вейча.

X_D	X_CQ			
	00	01	11	10
0	−	1	0	−
1	−	1	1	1

X_D	X_CQ			
	00	01	11	10
0	1	−	1	1
1	1	−	−	0

$$\overline{q_R} = \overline{X_C} \vee X_D = \overline{X_C \overline{X_D}}\,; \qquad \overline{q_S} = \overline{X_C} \vee \overline{X_D} = \overline{X_C X_D}\,.$$

Не важко переконатися, що $\overline{\overline{q_S} X_C} = \overline{q_R}$.

Дійсно, $\overline{\overline{X_C X_D} X_C} = \overline{(\overline{X_C} \vee \overline{X_D}) X_C} = \overline{\overline{X_D} X_C}$.

Тоді схема ***D***-тригера, побудована в базисі “І-НІ”, буде мати вигляд:

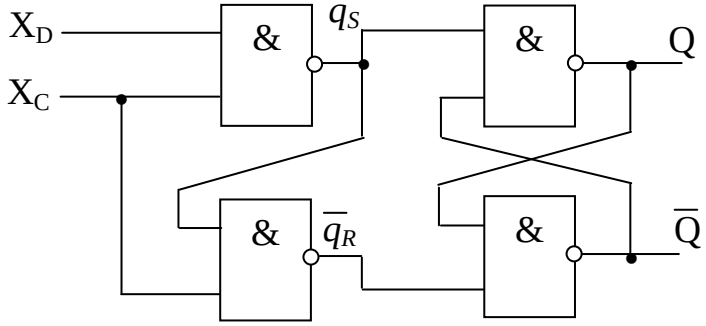


Рис. 3.13. Схема синхронного ***D***-тригера

3.2.6. Синтез JK - тригерів

JK-тригер є універсальним, на базі якого, шляхом зовнішніх комутацій можна отримати деякі інші типи тригерів.

Умовне графічне позначення асинхронного ***JK***-тригера з динамічними (імпульсними) входами показано на рис. 3.14, а умови функціонування тригера – в таблиці 3.9.

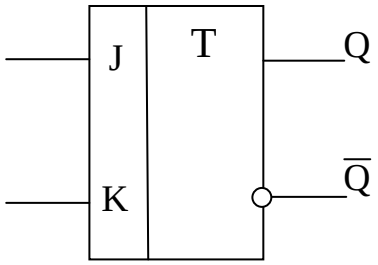


Рис.3.14. Асинхронний *JK*-тригер з динамічним управлінням

Таблиця 3.9

J	**K**	**Q**
0	0	Q*
0	1	0
1	0	1
1	1	$\overline{Q}$*

Із таблиці 3.9 видно, що ***JK***-тригер відрізняється логікою роботи від ***RS***-тригера тим, що при одночасній подачі двох одиниць на входи ***J*** і ***K*** тригер змінює свій стан на протилежний. Для решти сполучень вхідних сигналів переходи тригерів співпадають (при цьому вхід ***J*** аналогічний входу ***S***, а вхід ***K*** – входу ***R***). Проведемо синтез асинхронного ***JK***-тригера, використовуючи в якості елементарного автомата бістабільну схему на елементах “І-НІ” [13].

Використовуючи таблицю станів ***JK***-тригера (табл. 3.9) і матрицю переходів бістабільної схеми на елементах “І-НІ” (табл. 3.7), будуємо узагальнену таблицю переходів і функцій збудження.

Таблиця 3.10

$\mathbf{X_J}$	$\mathbf{X_K}$	**Q**	**Q′**	$\overline{q_R}$	$\overline{q_S}$
0	0	0	0	−	1
0	0	1	1	1	−
0	1	0	0	−	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	1	−
1	1	0	1	1	0
1	1	1	0	0	1

Мінімізуємо функції збудження за допомогою площинних діаграм Вейча.

X_J	$X_K Q$			
	00	01	11	10
0	–	1	0	–
1	–	1	0	1

X_J	$X_K Q$			
	00	01	11	10
0	1	–	1	1
1	0	–	1	0

$$\overline{q_R} = \overline{X_K} \vee \overline{Q} = \overline{X_K Q}\,; \qquad \overline{q_S} = \overline{X_J} \vee Q = \overline{X_J\,\overline{Q}}\,.$$

Після проведених перетворень, які заключалися в застосуванні до отриманих мінімальних форм функцій збудження інверсного закону, комбінаційну частину тригера зручно будувати на елементах “І-НІ”.

Схема тригера буде мати вигляд, представлений на рис.3.15.

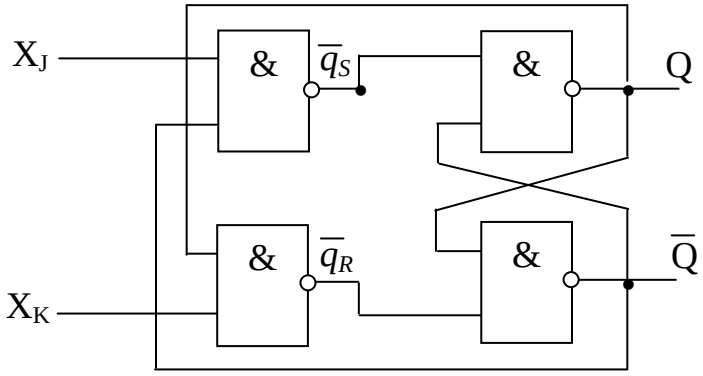


Рис. 3.15. Схема асинхронного *JK*-тригера

Вище зазначалось, що в тому випадку, коли тригер керується по рахунковому входу сигналами значної тривалості або потенціальними, необхідно забезпечити додаткову затримку проходження сигналів по внутрішнім ланцюгам. В цьому неважко переконатись, розглянувши граф функціонування тригера з рахунковим входом, приведений на рис. 3.16.

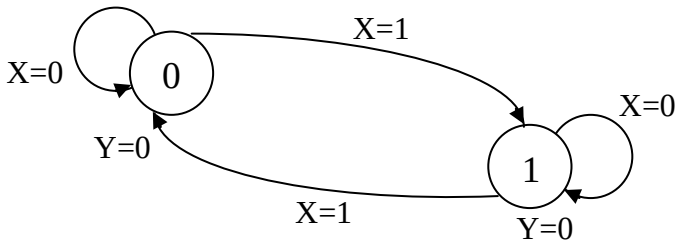


Рис. 3.16. Граф функціонування *T*-триггера

При сигналі **X** великої тривалості тригер перейде зі стану 0 в стан 1 и і почне знову переходити в стан 0. Це означає, що двух внутрішніх станів недостатньо для того, щоб при такому **X** зафіксувати одиночний стан.

Очевидно, потенційний тригер, керуємий по рахунковому входу, повинен функціонувати у відповідності з наступним графом (рис. 3.17).

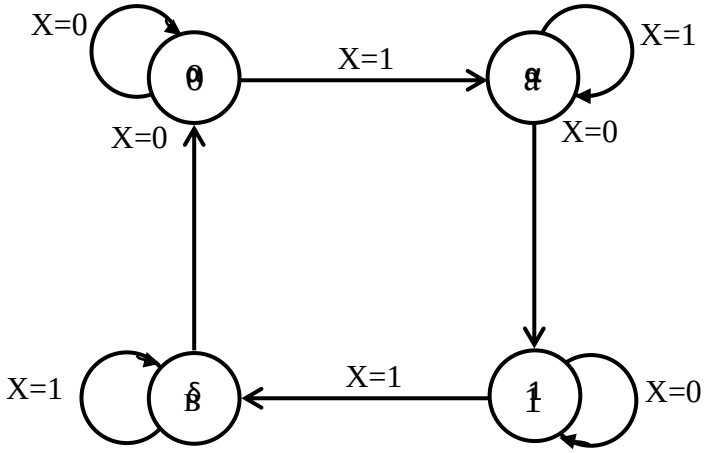


Рис. 3.17. Граф функціонування потенційного *T*-триггера

Якщо тригер знаходиться в нульовому стані, то при подачі одиничного сигнала на рахунковий вхід триггера він повинен перейти в стан, позначений на графі літерою ***a*** и названий проміжним, і залишатись в ньому на протязі всього часу дії одиничного сигналу. По закінченню дії одиничного сигналу, тобто при X=0, тригер переходить в одиночний стан, позначений на графі цифрою 1. Точно так само здійснюється перехід тригера з одиничного стану в нульовий.

Це означає, що для синтезу такого тригера одного елементарного автомата недостатньо. Для запам'ятовування чотирьох внутрішніх станів потенційного тригера знадобиться дві тригерні комірки, а це призводить до отримання структури тригера, яка називається двухступеневою. В основному полі умовного графічного позначення таких тригерів ставяться дві букви Т – ТТ.

Проведемо синтез двухступеневого асинхронного ***JK***-тригера. Умовне графічне позначення такого тригера приведено на рис. 3.18.

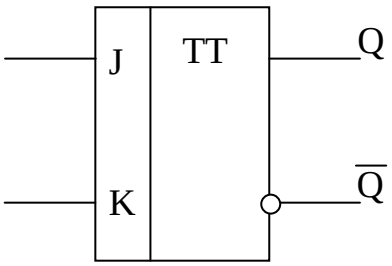


Рис. 3.18. Умовне графічне позначення
двухступеневого *JK*-триггера

Опишемо роботу двухступеневого *JK*- тригера за допомогою графа (рис. 3.19). В якості елементарного автомата выбираємо тригерну комірку на елементах "І-НІ". Для синтеза тригера потрібно дві комірки. Позначимо стан тригерної комірки першого ступеню буквою **A**, другого ступеня – **Q** .

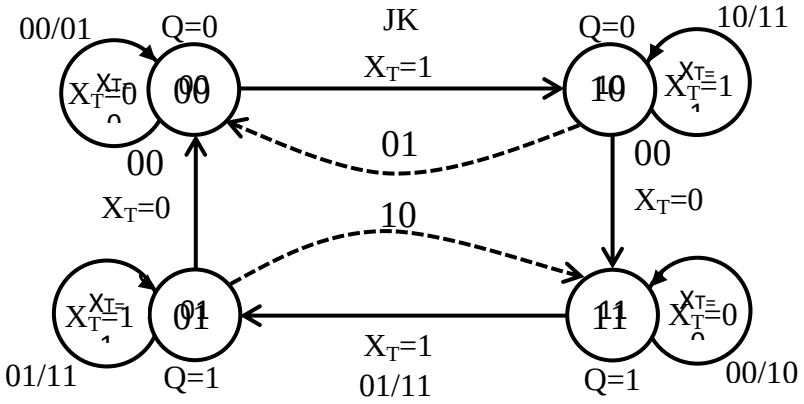


Рис.3.19. Граф функціонування двухступеневого *JK*-триггера

Для кодування чотирьох внутрішніх станів потрібно два двоїчних розряди, позначимо їх тими ж буквами A та Q.

Выберемо наступний варіант кодування внутрішніх станів:

нульовий стан тригера – 00;
перший допоміжний стан – 10;
одиничний стан триггера – 11;
другий допоміжний стан – 01.

Позначимо вхідні сигнали $\mathbf{X_J}$ та $\mathbf{X_K}$, а функції збудження $\boldsymbol{q_{AR}}$, $\boldsymbol{q_{AS}}$, $\boldsymbol{q_{QR}}$, $\boldsymbol{q_{QS}}$.

Складемо узагальнену таблицю переходів, виходів та функцій збудження, (табл. 3.19) визначаючи стан кожної тригерної комірки в наступному такті з графу, а значення функцій збудження – з матриці переходів, показаної в таблиці 3.7.

Переходи, показані на графі пунктирними дугами із розгляду виключаємо через те, що діючі на входах тригера сигнали не можуть одночасно мінятись на протилежні без переходу через нульові значення.

Таблиця 3.19

$\mathbf{X_J}$	$\mathbf{X_K}$	**A**	**Q**	**A′**	**Q′**	$\overline{\boldsymbol{q}}_{\boldsymbol{AR}}$	$\overline{\boldsymbol{q}}_{\boldsymbol{AS}}$	$\overline{\boldsymbol{q}}_{\boldsymbol{QR}}$	$\overline{\boldsymbol{q}}_{\boldsymbol{QS}}$
0	0	0	0	0	0	–	1	–	1
0	0	0	1	0	0	–	1	0	1
0	0	1	0	1	1	1	–	1	0
0	0	1	1	1	1	1	–	1	–
0	1	0	0	0	0	–	1	–	1
0	1	0	1	0	1	–	1	1	–
0	1	1	0	–	–	–	–	–	–
0	1	1	1	0	1	0	1	1	–
1	0	0	0	1	0	1	0	–	1
1	0	0	1	–	–	–	–	–	–
1	0	1	0	1	0	1	–	–	1
1	0	1	1	1	1	1	–	1	–
1	1	0	0	1	0	1	0	–	1
1	1	0	1	0	1	–	1	1	–
1	1	1	0	1	0	1	–	–	1
1	1	1	1	0	1	0	1	1	–

Визначимо мінімальні форми функцій збудження.

$X_J X_K$	AQ			
	00	01	11	10
00	–	–	1	1
01	–	–	0	–
11	1	–	0	1
10	1	–	1	1

$$\overline{q_{AR}} = \overline{X_K} \vee \overline{Q} = \overline{X_K Q};$$

$X_J X_K$	AQ			
	00	01	11	10
00	1	1	–	–
01	1	1	1	–
11	0	1	1	–
10	0	–	–	–

$$\overline{q_{AS}} = \overline{X_J} \vee Q = \overline{X_J \overline{Q}}.$$

$X_J X_K$	AQ			
	00	01	11	10
00	–	0	1	1
01	–	1	1	–
11	–	1	1	–
10	–	–	1	–

$$\overline{q_{QR}} = X_K \vee A = \overline{\overline{X_K}\,\overline{A}};$$

$X_J X_K$	AQ			
	00	01	11	10
00	1	1	–	0
01	1	–	–	–
11	1	–	–	1
10	1	–	–	1

$$\overline{q_{QS}} = X_J \vee \overline{A} = \overline{\overline{X_J}\,A}.$$

Схема двухступеневого асинхронного ***JK***-тригера представлена на рис. 3.20.

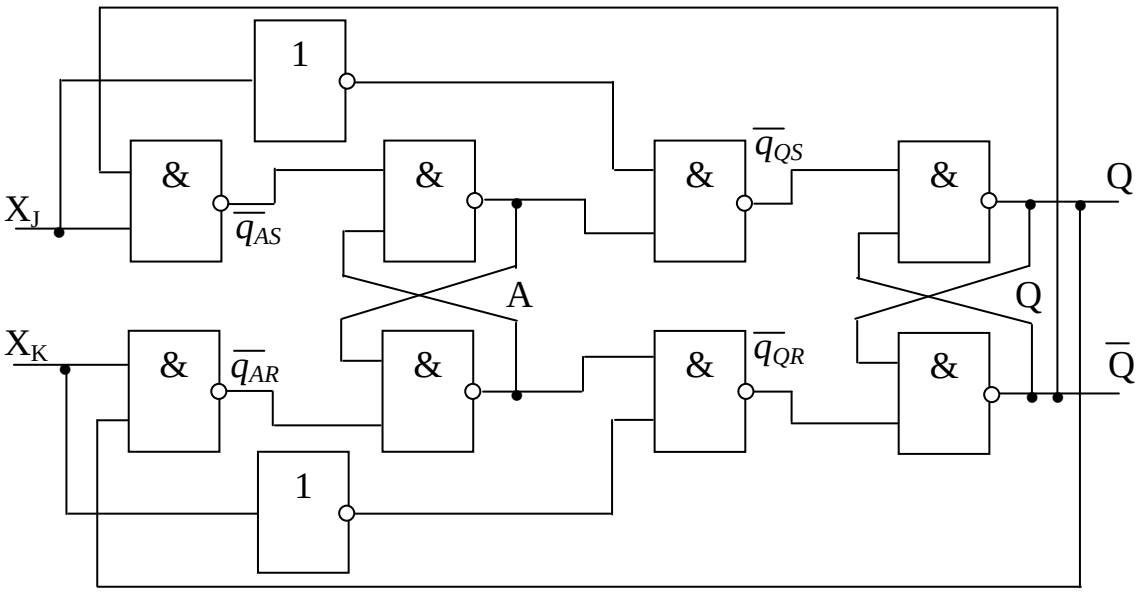


Рис. 3.20. Схема двухступеневого асинхронного *JK*-тригера

Розглянуті раніше тригери називаються *асинхронними*, тому що зміна стану на виході відбувається при будь-якої зміни вхідних сигналів. Якщо при формуванні вхідних сигналів виникають помилкові короткі імпульси за рахунок «ефекту гонок» (змагань), ці короткі імпульси можуть викликати помилкові спрацьовування тригера.

Для виключення помилкових спрацьовувань застосовують ***синхронні тригери***, у яких зміна вихідних станів відбувається в момент подачі спеціальних *синхроімпульсів*. Ці синхроімпульси подаються після завершення перехідних процесів в схемах формування вхідних сигналів тригера [13].

На рис. 3.21 показане умовне графічне позначення синхронного ***JK***-тригера, а в таблиці 3.20 – умови його функціонування.

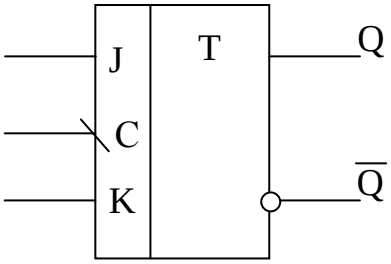


Рис. 3.21. Умовне графічне позначення
двухтактного *JK*-триггера

Таблиця 3.20

C	**J**	**K**	**Q**	$\overline{\mathbf{Q}}$	**Стан**
0	—	—	Q*	$\overline{Q}$*	Режим зберігання
1	—	—	Q*	$\overline{Q}$*	Режим зберігання
↓	0	0	Q*	$\overline{Q}$*	Режим зберігання
↓	0	1	0	1	Скидання в «нуль»
↓	1	0	1	0	Установка в «одиницю»
↓	1	1	$\overline{Q}$*	Q*	Інверсія вихідного стану

В табл. 3.20 умова зміни логічного рівня на синхровході з «1» на «0» позначена стрілкою «↓». Такий фронт називається «*спадаючим фронтом*» або «фронтом 1–0». Тригер, умови функціонування якого задані таблицею 3.20, має назву «*синхронний тригер, керований фронтом*», який також називається «*двотактним тригером*».

На відміну від функціонування синхронного ***RS***-тригера (табл. 3.2) у синхронного ***JK***-тригера усунена невизначеність стану при подачі на входи ***J*** і ***K*** логічних одиниць (див. останній рядок табл. 3.20). У ***JK*** –тригера в цій ситуації інвертується вихідний стан, тобто при подачі на входи ***J*** і ***K*** двох логічних одиниць цей тригер працює як рахунковий тригер.

На рис. 3.22 показана принципова схема *синхронного двотактного* ***JK****-тригера*.

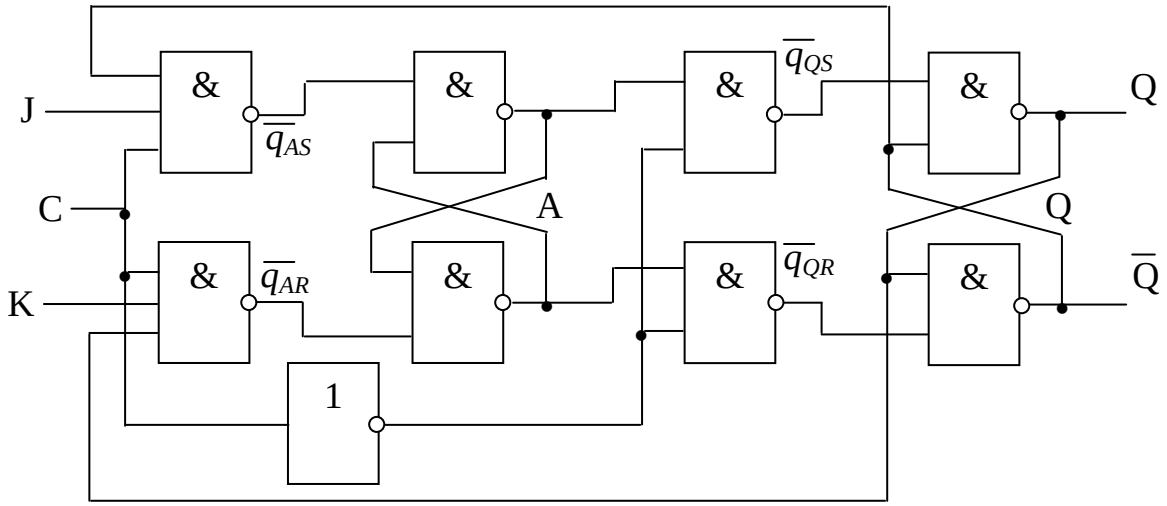


Рис. 3.22. Схема синхронного двотактного *JK*-тригера

3.2.7. Модифікації синхронних тригерів

На основі будь-якого ***D***-тригера, керованого фронтом (см. рис. 3.12), можна реалізувати лічильний ***T***-тригер, якщо з'єднати інверсний вихід **Q** з входом **D** (см. рис. 3.23). Аналогічну модифікацію можна виконати (реалізувати лічильний ***T***-тригер) на основі розглянутих раніше двотактних тригерів, керованих фронтом [13].

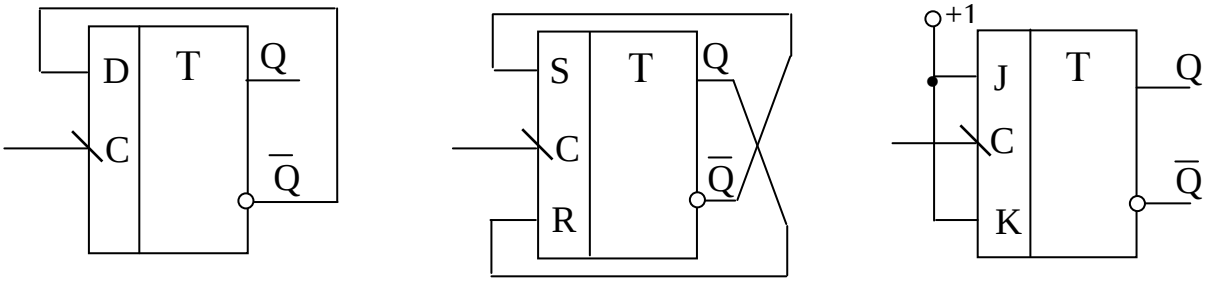


Рис. 3.23. Реалізація лічильного ***T***-тригера на основі *D*-тригера, *RS*- тригера, *JK*- тригера

Синхронний ***D***-тригер можна реалізувати на основі синхронних ***RS***- або ***JK***- тригерів і логічного інвертора (рис. 3.24).

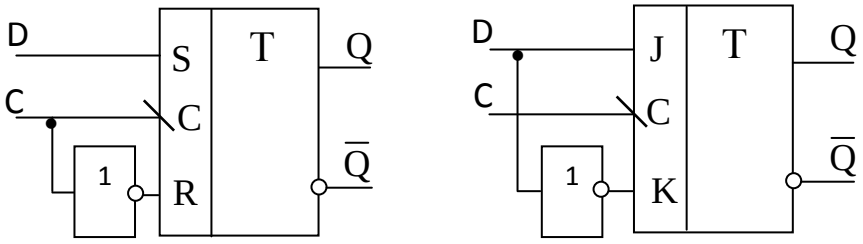


Рис. 3.24. Синхронні ***D***-тригери на основі *RS*- і *JK*- тригерів

Для реалізації однотактного ***D***-тригера, керованого потенціалом, необхідний однотактной ***RS***-тригер [13]. Однотактні синхронні тригери часто називають «*синхронними тригерами, керованими потенціалом*».

Більшість тригерів в інтегральному виконанні мають окрім розглянутих управляючих і синхронізуючих входів також і асинхронні входи установки в одиничний і нульовий початковий стан. На рис. 3.25 приведено умовне графічне позначення цих тригерів.

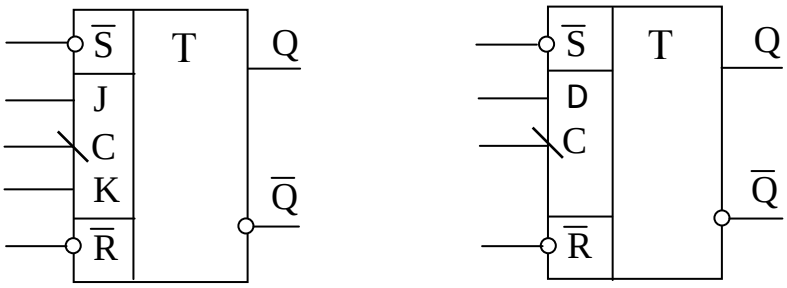


Рис. 3.25. Умовне графічне позначення *JK*- і *D*- тригерів с інверсними асинхронними входами установки

3.3. ФУНКЦІОНАЛЬНІ ВУЗЛИ ЦИФРОВИХ ЕОМ КОМБІНАЦІЙНОГО ТИПУ

3.3.1. Синтез двійкового шифратора

Перетворювачі кодів (кодові перетворювачі) призначені для перетворення двійкового коду одного типу в двійковий код іншого типу. Наприклад, двійковий код − в код Грея і т. п. Серед кодових перетворювачів необхідно виділити перетворювачі, які працюють з розподільними кодами. Двійковий розподільний код це код, в якому значення, рівне одиниці, може одночасно приймати тільки один з n розрядів. Наприклад, якщо $A_p = a_0\ a_1\ a_2\ a_3$ - чотирьох розрядний розподільний код, то A_p може приймати тільки чотири різні значення: 1000, 0100, 0010, 0001. Розподільний код ще називають унітарним. Перетворювачі, працюючі з такими кодами називаються шифраторами і дешифраторами.

Шифратором називається вузол комбінаційного типу, що перетворює двійковий розподільний код в двійковий позиційний або двійковий код іншого типу. Шифратор називають ще кодером, звідки і сталася скорочена назва, що поміщається в умовному графічному зображенні, яке приведене на рис. 3.26.

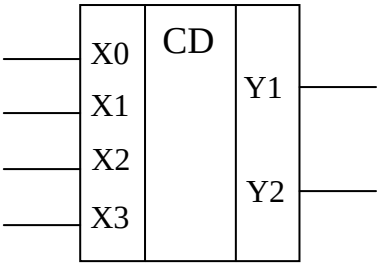


Рис. 3. 26. Умовне графічне позначення шифратора

Синтез шифраторів розглянемо на прикладі побудови схеми шифратора на чотири входи, що перетворює двійковий розподільний код X_0 X_1 X_2 X_3 в двійковий позиційний. При визначенні потрібного числа розрядів двійкового позиційного коду можна скористатися відомим співвідношенням

$$m = [\log_2 n]_{\text{н.б.ц.}},$$

де n – кількість розрядів розподільного коду (входів шифратора);

m – кількість розрядів двійкового позиційного коду (виходів шифратора);

н.б.ц. – найближче більше ціле число.

Позначимо розряди двійкового позиційного коду, що є виходами шифратора, Y_1 і Y_2, оскільки для нашого прикладу $m = 2$.

Сформулюємо умови роботи шифратора за допомогою таблиці істинності (табл. 3.21).

Таблиця 3.21

X_0	X_1	X_2	X_3	Y_1	Y_2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Функції Y_1 і Y_2, записані в ДДНФ матимуть вигляд:

$$Y_1 = \overline{X}_0\overline{X}_1X_2\overline{X}_3 \vee \overline{X}_0\overline{X}_1\overline{X}_2X_3,$$

$$Y_2 = \overline{X}_0X_1\overline{X}_2\overline{X}_3 \vee \overline{X}_0\overline{X}_1\overline{X}_2X_3.$$

Ці функції залежать від чотирьох змінних, але визначені тільки на чотирьох наборах, отже, при мінімізації цих функцій їх необхідно довизначити. Для мінімізації функцій Y_1 і Y_2 скористаємося площинними діаграмами Вейча (таблиця. 3.22 і 3.23 відповідно).

Таблиця. 3.22

X_0X_1	X_2X_3			
	00	01	11	10
00	–	1	–	1
01	–	–	–	–
11	–	–	–	–
10	0	–	–	–

$Y_1 = X_3 \vee X_2;$

Таблиця. 3.23

X_0X_1	X_2X_3			
	00	01	11	10
00	–	1	–	0
01	1	–	–	–
11	–	–	–	–
10	0	–	–	–

$Y_2 = X_3 \vee X_1.$

Після мінімізації функцій Y_1 і Y_2, представлених в ДКНФ, отримаємо

$$Y_1 = \overline{X}_1 \cdot \overline{X}_0; \qquad Y_2 = \overline{X}_0 \cdot \overline{X}_2.$$

З мінімальних форм вихідних функцій шифратора видно, що є два варіанти побудови схеми – на елементах “АБО” і елементах “І”. В другому випадку на вхід шифратора вимагається подавати інверсний розподільний код. Схема шифратора, на елементах “АБО” показана на рис. 3.27.

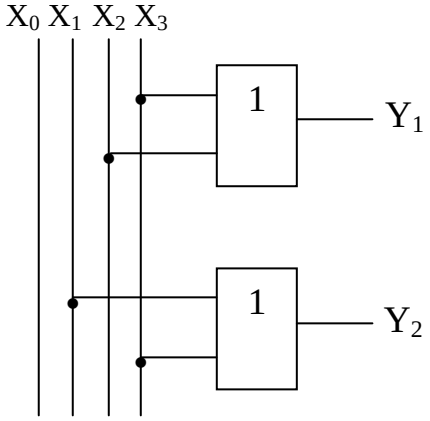


Рис. 3.27. Схема шифратора на елементах “АБО”

Аналізуючи схему, можна узагальнити принцип побудови шифраторів на багаторозрядні розподільні коди. Дійсно, на входи кожного елемента “АБО” підключаються розряди розподільного коду рівні одиниці на наборах, на яких функція **Y** дорівнює одиниці.

Реалізація схем шифраторів в інших базисах робиться шляхом перетворення вихідних функцій шифратора за відомими правилами. Наприклад, схема шифратора на чотири входи у базисі “І-НІ” з урахуванням того, що

$$Y_1 = \overline{\overline{X_3 \vee X_2}} = \overline{\overline{X}_3 \cdot \overline{X}_2}; \qquad Y_2 = \overline{\overline{X_1 \vee X_3}} = \overline{\overline{X}_1 \cdot \overline{X}_3}.$$

матиме вигляд (рис. 3.28).

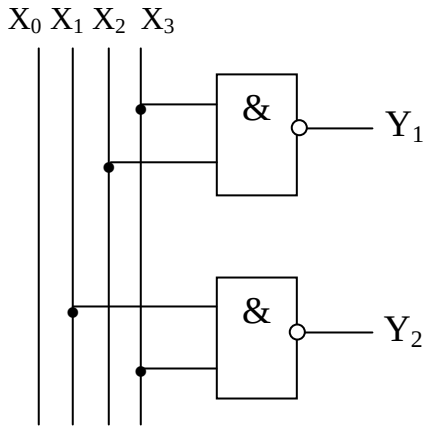


Рис. 3.28. Схема шифратора на елементах “І-НІ”

3.3.2. Синтез двійкових дешифраторів

Дешифратором (декодером) називається вузол, призначений. для перетворення двійкового позиційного коду в розподільний. Кожній комбінації сигналів на вході дешифратора відповідає одиничний сигнал тільки на одному, строго певному виході. Дешифратори знаходять застосування в пристроях, в яких необхідно з множини об'єктів вибрати один, тобто вирішити завдання вибору. Тому іноді дешифратори називають виборчими схемами [2].

У загальному випадку дешифратор, що має n входів, має $m = 2^n$ виходів, оскільки n-розрядне слово може приймати 2^n різних значень, кожному з яких повинен відповідати одиничний сигнал на одному з виходів дешифратора. Дешифратор називається повним, якщо він перетворює n -розрядний двійковий код в 2^n-розрядний розподільний. Інакше дешифратор називають неповним. Розрізняють прямі і інверсні дешифратори. Прямий дешифратор перетворює двійковий код в прямій розподільний, а інверсний − в інверсний розподільний. Умовне графічне зображення дешифратора показане на рис. 3.29.

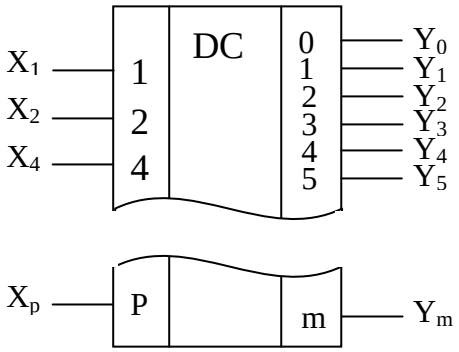


Рис. 3.29. Умовне графічне позначення дешифратора

Входи у дешифратора позначаються десятковими цифрами що зображують двійкові ваги $P = 2^{n-1}$, а виходи десятковими цифрами кодових комбінацій $m = 2^n-1$. Допускається зображення неповного набору входів і виходів як це показано на рис. 3.29. Якщо позначити позиційний двійковий код, що поступає на вхід дешифратора $\mathbf{X = X_1\ X_2 ...\ X_p}$, то на виході повного дешифратора має бути сформований $(m+1)$-розрядний розподільний код $\mathbf{Y = Y_0\ Y_1\ Y_2 ...\ Y_m}$. Тоді логічна функція виходів **Y** повного дешифратора може бути описана системою логічних функцій :

$$
\begin{aligned}
\mathbf{Y_0} &= \overline{\mathbf{X}}_1\,\overline{\mathbf{X}}_2 \ldots \overline{\mathbf{X}}_{P-1}\,\overline{\mathbf{X}}_P;\\
\mathbf{Y_1} &= \overline{\mathbf{X}}_1\,\overline{\mathbf{X}}_2 \ldots \overline{\mathbf{X}}_{P-1}\,\mathbf{X}_P;\\
\mathbf{Y_2} &= \overline{\mathbf{X}}_1\,\overline{\mathbf{X}}_2 \ldots \mathbf{X}_{P-1}\,\overline{\mathbf{X}}_P;\\
&\ldots\ldots\ldots\ldots\\
\mathbf{Y_{m-1}} &= \mathbf{X}_1\,\mathbf{X}_2 \ldots \mathbf{X}_{P-1}\,\overline{\mathbf{X}}_P;\\
\mathbf{Y_m} &= \mathbf{X}_1\,\mathbf{X}_2 \ldots \mathbf{X}_{P-1}\,\mathbf{X}_P.
\end{aligned}
\qquad (3.4)
$$

Кожна з функцій цієї системи складається з єдиної конституенти одиниці, що відповідає набору $\mathbf{X_i}$, на якому функція $\mathbf{Y_i}$ набуває значення рівне одиниці. Різні способи обчислення кон'юнкцій в системі функцій (3.4) породжують різні структури дешифраторів, що відрізняються за відомими критеріями структурної складності і глибини. Методику структурного синтезу дешифратора розглянемо на прикладі побудови дешифратора на три входи. Таблиця умов роботи дешифратора матиме вигляд:

Таблиця 3.24

X_4	X_2	X_1		Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0		1	0	0	0	0	0	0	0
0	0	1		0	1	0	0	0	0	0	0
0	1	0		0	0	1	0	0	0	0	0
0	1	1		0	0	0	1	0	0	0	0
1	0	0		0	0	0	0	1	0	0	0
1	0	1		0	0	0	0	0	1	0	0
1	1	0		0	0	0	0	0	0	1	0
1	1	1		0	0	0	0	0	0	0	1

Функції $\mathbf{Y_i}$ у відповідності до виразу (3.4) приймуть наступний вигляд:

$$
\begin{aligned}
&\mathbf{Y_0 = \overline{X}_4\,\overline{X}_2\,\overline{X}_1;}\\
&\mathbf{Y_1 = \overline{X}_4\,\overline{X}_2\,X_1;}\\
&\mathbf{Y_2 = \overline{X}_4\,X_2\,\overline{X}_1;}\\
&\mathbf{Y_3 = \overline{X}_4\,X_2\,X_1;}\\
&\mathbf{Y_4 = X_4\,\overline{X}_2\,\overline{X}_1;}\\
&\mathbf{Y_5 = X_4\,\overline{X}_2\,X_1;}\\
&\mathbf{Y_6 = X_4\,X_2\,\overline{X}_1;}\\
&\mathbf{Y_7 = X_4\,X_2\,X_1.}
\end{aligned}
\qquad (3.5)
$$

Пряма структурна реалізація функцій $\mathbf{Y_i}$ (3.5), тобто побудова схеми з елементів "І" на три входи (вважатимемо, що є вхідні змінні і їх заперечення), призводить до отримання структури дешифратора, що називається одноступеневим, паралельним або прямокутним дешифратором. Структурна схема такого дешифратора представлена на рис. 3.30. Дешифратор такого типу має один ступінь дешифрування, отже, глибина схеми дорівнює одному логічному елементу і його швидкодія максимальна. Кількість елементів "І" (структурна складність) $\boldsymbol{H = 2^n}$, а ціна по Квайну $\boldsymbol{C = n \cdot 2^n}$.

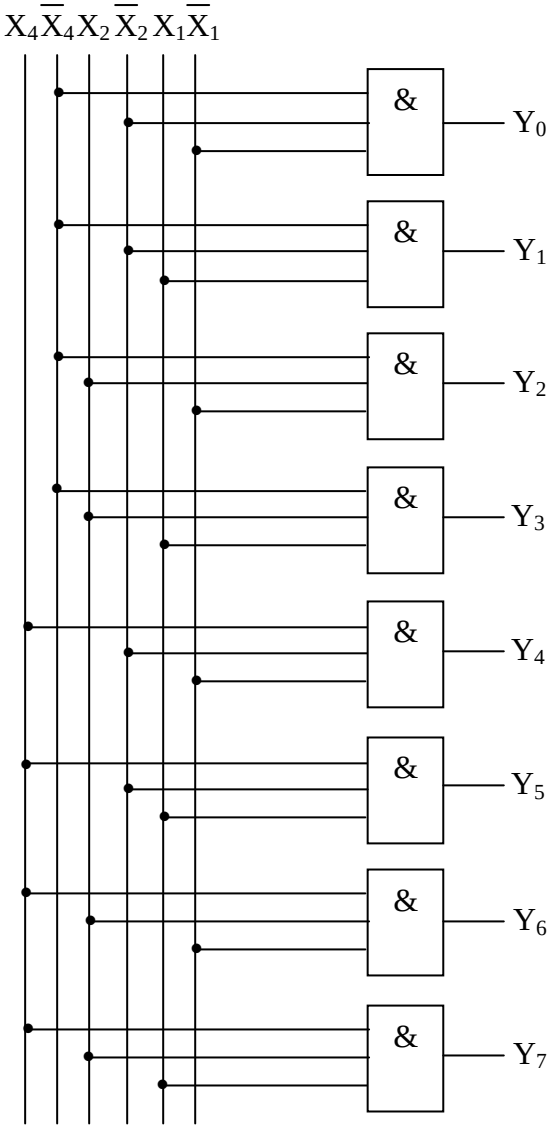


Рис. 3.30. Схема прямокутного дешифратора

При побудові схем дешифраторів на більшу кількість входів, із-за обмеженого числа входів логічних елементів, паралельні схеми дешифраторів реалізувати не представляється можливим. В цьому випадку будують пірамідальні, двох і багатоступеневі дешифратори.

Принцип побудови вказаних вище схем дешифраторів ґрунтований на тому, що вихідні функції дешифраторів підпорядковуються відносно своїх аргументів сполучному закону. Це дозволяє робити різне розбиття і групування аргументів з метою реалізації вихідних функцій дешифраторів схемами із елементів з обмеженим числом входів, а також для виявлення кон'юнкцій, які повторюються, що дає можливість використати одні і ті ж логічні елементи для реалізації декількох вихідних функцій.

Розглянемо принцип побудови пірамідального і двохступеневого дешифраторів на чотири входи.

Вирази логічних функцій виходів **Y** дешифратора запишемо таким чином:

$$
\begin{aligned}
&Y_0 = [(\overline{X}_8\,\overline{X}_4)\,\overline{X}_2]\,\overline{X}_1; && Y_8 = [(X_8\,\overline{X}_4)\,\overline{X}_2]\,\overline{X}_1;\\
&Y_1 = [(\overline{X}_8\,\overline{X}_4)\,\overline{X}_2]\,X_1; && Y_9 = [(X_8\,\overline{X}_4)\,\overline{X}_2]\,X_1;\\
&Y_2 = [(\overline{X}_8\,\overline{X}_4)\,X_2]\,\overline{X}_1; && Y_{10} = [(X_8\,\overline{X}_4)\,X_2]\,\overline{X}_1;\\
&Y_3 = [(\overline{X}_8\,\overline{X}_4)\,X_2]\,X_1; && Y_{11} = [(X_8\,\overline{X}_4)\,X_2]\,X_1;\\
&Y_4 = [(\overline{X}_8\,X_4)\,\overline{X}_2]\,\overline{X}_1; && Y_{12} = [(X_8\,X_4)\,\overline{X}_2]\,\overline{X}_1;\\
&Y_5 = [(\overline{X}_8\,X_4)\,\overline{X}_2]\,X_1; && Y_{13} = [(X_8\,X_4)\,\overline{X}_2]\,X_1;\\
&Y_6 = [(\overline{X}_8\,X_4)\,X_2]\,\overline{X}_1; && Y_{14} = [(X_8\,X_4)\,X_2]\,\overline{X}_1;\\
&Y_7 = [(\overline{X}_8\,X_4)\,X_2]\,X_1; && Y_{15} = [(X_8\,X_4)\,X_2]\,X_1.
\end{aligned}
\qquad (3.6)
$$

Якщо зробити групування аргументів відповідно до формул (3.6), то отримана структура дешифратора складатиметься з елементів “І” на два входи і при цьому реалізація кон’юнкцій в круглих дужках дозволяє використати один елемент “І” для формування чотирьох вихідних функцій, а кон’юнкції в квадратних дужках – для формування двох функцій (для дешифратора на чотири входи). Структура такого дешифратора матиме вигляд, представлений на рис. 3.31.

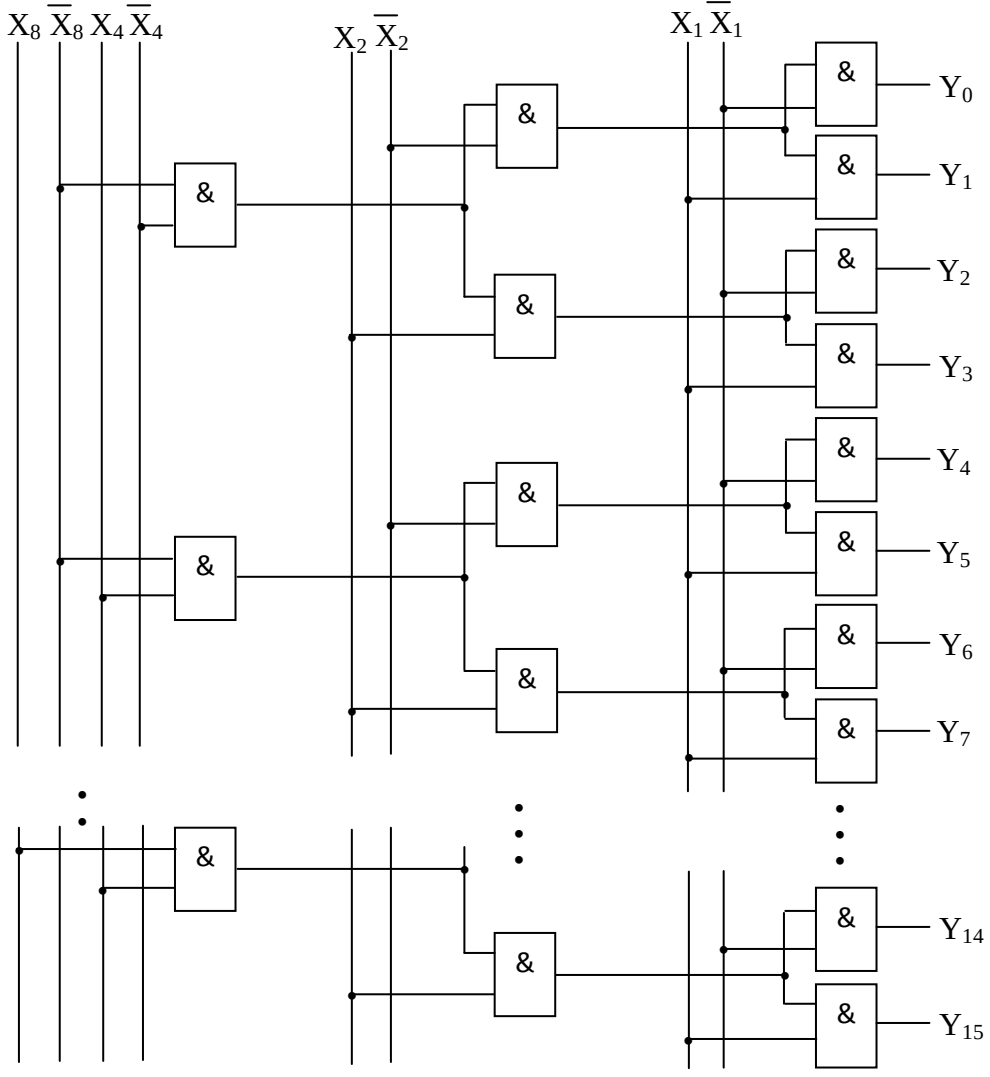


Рис. 3.31. Схема пірамідального дешифратора

Такі структури дешифраторів називаються пірамідальними або послідовними. Кількість ступенів пірамідального дешифратора (структурна глибина схеми) визначається із виразу

$$M_{пір} = n - 1.$$

Для визначення структурної складності необхідно визначити кількість логічних елементів в кожному рядку і підсумувати отримані значення, тобто

$$K_{пір} = 2^2 + 2^3 + 2^4 + ... + 2^n.$$

Скориставшись формулою для визначення кількості членів геометричної прогресії, отримаємо

$$K_{пір} = 4\,(2^{n-1} - 1).$$

У зв'язку з тим, що усі логічні елементи, на яких синтезована схема, двовходові, ціна по Квайну такого дешифратора визначається з виразу

$$C_{пір} = 2 \cdot K_{пір} = 8\,(2^{n-1} - 1).$$

При малих значеннях ***n*** ціна по Квайну пірамідального дешифратора і одноступеневого приблизно рівні. А при $\boldsymbol{n \geq 4}$ ціна пірамідального дешифратора буде менше ціни одноступеневого і економія устаткування може бути визначена з наступного співвідношення

$$\frac{C}{Cпір} = \frac{n \cdot 2^n}{8(2^{n-1} - 1)} \approx \frac{n \cdot 2^n}{8 \cdot 2^{n-1}} \approx \frac{n}{4}.$$

Швидкодія пірамідального дешифратора в $(n - 1)$ раз менша, ніж у одноступеневого.

Якщо зробити групування аргументів у вираженні (3.6) по парах, то отримані вирази дозволять побудувати структуру дешифратора, що складається з двох ступенів перетворення

$$\begin{aligned}
&\mathbf{Y_0 = (\overline{X}_8\,\overline{X}_4)\,(\overline{X}_2\,\overline{X}_1);} && \mathbf{Y_8 = (X_8\,\overline{X}_4)\,(\overline{X}_2\,\overline{X}_1);}\\
&\mathbf{Y_1 = (\overline{X}_8\,\overline{X}_4)\,(\overline{X}_2\,X_1);} && \mathbf{Y_9 = (X_8\,\overline{X}_4)\,(\overline{X}_2\,X_1);}\\
&\mathbf{Y_2 = (\overline{X}_8\,\overline{X}_4)\,(X_2\,\overline{X}_1);} && \mathbf{Y_{10} = (X_8\,\overline{X}_4)\,(\,X_2\,\overline{X}_1);}\\
&\mathbf{Y_3 = (\overline{X}_8\,\overline{X}_4)\,(X_2\,X_1);} && \mathbf{Y_{11} = (X_8\,\overline{X}_4)\,(X_2\,X_1);}\\
&\mathbf{Y_4 = (\overline{X}_8\,X_4)\,(\overline{X}_2\,\overline{X}_1);} && \mathbf{Y_{12} = (X_8\,X_4)\,(\overline{X}_2\,\overline{X}_1);}\\
&\mathbf{Y_5 = (\overline{X}_8\,X_4)\,(X_2\,X_1);} && \mathbf{Y_{13} = (X_8\,X_4)\,(\overline{X}_2\,X_1);}\\
&\mathbf{Y_6 = (\overline{X}_8\,X_4)\,(X_2\,\overline{X}_1);} && \mathbf{Y_{14} = (X_8\,X_4)\,(X_2\,\overline{X}_1);}\\
&\mathbf{Y_7 = (\overline{X}_8\,X_4)\,(X_2\,X_1);} && \mathbf{Y_{15} = (X_8\,X_4)\,(X_2\,X_1).}
\end{aligned} \qquad (3.7)$$

Схема двохступеневого дешифратора представлена на рис. 3.32.

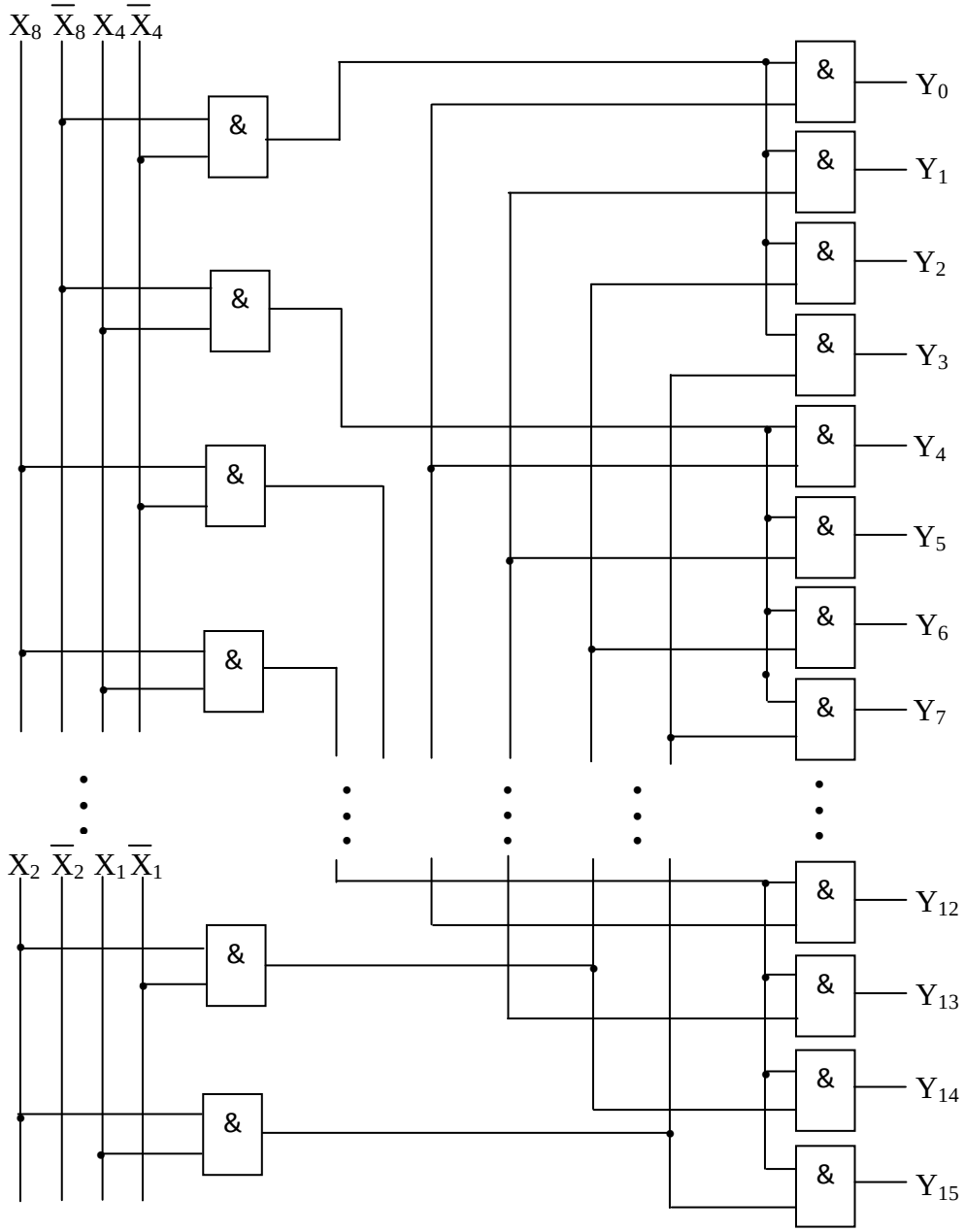


Рис. 3.31. Схема двохступеневого дешифратора

Структурна глибина двохступеневого дешифратора дорівнює 2. Для визначення структурної складності необхідно визначити кількість логічних елементів кожної ступені і підсумувати отримані значення. Для реалізації другої ступені дешифрації потрібно стільки ж логічних елементів, скільки виходних функцій Y_i дешифратора, тобто 2^n.

Проаналізувавши значення структурної складності і глибини різних типів дешифраторів, можна зробити наступні висновки:

максимальною швидкодією володіє прямокутний дешифратор, далі за ступенем зменшення швидкодії за ним слідують двоступеневий, багатоступеневий і пірамідальний дешифратори;

по структурній складності, що оцінюється ціною по Квайну, дешифратори розташовуються в наступному порядку – багатоступеневий, двоступеневий, пірамідальний і прямокутний.

3.3.3. Однорозрядний напівсуматор комбінаційного типу

Однорозрядним напівсуматором називається вузол, призначений для додавання двійкових чисел однойменних розрядів двох чисел без урахування перенесення з сусіднього молодшого розряду.

Однорозрядний напівсуматор здійснює реалізацію функції нерівнозначності, яка описує додавання двох або більше двійкових змінних по модулю два. Умовне зображення такого напівсуматора наведено на рис. 3.32.

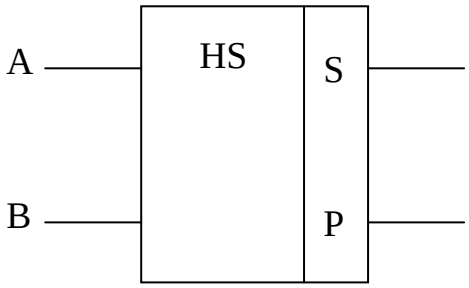


Рис. 3.32. Умовне зображення однорозрядного напівсуматора

Схема має два входи, на які надходять сигнали, що відповідають значенням розрядів двійкових чисел, і два виходи для значень суми і переносів, які виникають при додаванні.

Вирішимо задачу синтезу однорозрядного напівсуматора за критерієм найменшої структурної складності.

Етап абстрактного синтезу.

На цьому етапі необхідно сформулювати умови функціонування напівсуматора за допомогою таблиці істинності і записати функції **S** і **P** в аналітичному вигляді.

1. Опис умов функціонування однорозрядного напівсуматора (OHS).

Так як вхідних сигналів два, то таблиця істинності містить чотири набори змінних (табл. 3.25).

Таблиця 3.25

A	**B**	**S**	**P**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2. Подання функцій **S** і **P** в аналітичному вигляді.

По таблиці можна легко скласти булеві функції для значень суми **S** і перенесення **P** в ДДНФ і ДКНФ.

ДДНФ:

$$S_{(1)} = \overline{A}B \vee A\overline{B}; \qquad P_{(1)} = AB. \tag{3.8}$$

ДКНФ:

$$S_{(0)} = (A \vee B)(\overline{A} \vee \overline{B}); \qquad P_{(0)} = (A \vee B)(A \vee \overline{B})(\overline{A} \vee B). \tag{3.9}$$

На цьому етап абстрактного синтезу закінчується.

Етап структурного синтезу.

На цьому етапі потрібно провести аналіз отриманих досканалих нормальних форм (ДНФ) логічних функцій і перетворити їх у відповідності із заданими критеріями оптимальності і заданим функціонально повними наборами (ФПН) логічних елементів.

Так як в якості критерія оптимальності задана найменша структурна складність схеми, то необхідно провести спільну мінімізацію логічних функцій **S** і **P**, так як вузол формує два вихідних сигнали.

З аналізу ДДНФ ЛФ видно, що їх спростити не можна. ДДНФ цих функцій є і скорочені, і тупикові, і мінімальні форми функцій **S** і **P**.

З аналізу ДКНФ функцій випливає, що в вираженні для функції $\mathbf{S_{(0)}}$ можна виключити одну операцію заперечення, застосувавши до конституенти нуля третього набору інверсний закон

$$S_{(0)} = (A \vee B)(\overline{A} \vee \overline{B}) = (A \vee B)\overline{AB}. \tag{3.10}$$

Що стосується функції **P**, то її ДКНФ значно складніше ДДНФ. Отже, для побудови схеми OHS доцільно використовувати для реалізації функції **S** вираз (3.10), а для реалізації **P** вираз (3.8). Але $\overline{AB}$ в вираженні (3.8) є ні що інше, як $\overline{\mathbf{P}}$, значить, зробивши вищезазначені зміни функції $\mathbf{S_{(0)}}$, ми представимо цю функцію через функцію **P**, тобто виконаємо спільну мінімізацію функцій **S** і **P**.

Отже, для побудови схеми OHS необхідно реалізувати логічну функцію (3.10):

$$S_{(0)} = (A \vee B) \cdot \overline{P}_{(1)} = (A \vee B) \cdot \overline{AB}.$$

Якщо в якості ФПН ЛЕ заданий базис І, АБО, НІ, тоді схема OHS матиме вигляд, представлений на рис. 3.33.

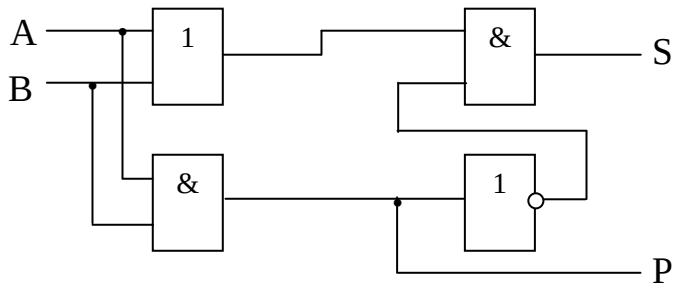


Рис. 3.33. Схема однорозрядного напівсуматора

Проаналізуємо схему за основними критеріями оптимальності:

структурна складність – 7;

структурна глибина по виходу S – 3, по виходу P – 1.

Якщо в якості ФПН ЛЕ заданий базис Шеффера (“І-НІ”), то виявляється, що складність схеми OHS, побудованої за формулами (3.8), така ж, як складність схеми, що реалізує логічний вираз (3.10). Покажемо це. Приведемо до базису “І-НІ” логічні функції (3.8).

$$S_{(1)} = \overline{A}B \vee A\overline{B} = \overline{\overline{\overline{A}B \vee A\overline{B}}} = \overline{\overline{\overline{A}B} \cdot \overline{A\overline{B}}}.$$

Для реалізації цієї функції потрібно п’ять двухвходових елементів “І-НІ”.

$$P_{(1)} = AB = \overline{\overline{AB}}.$$

Для реалізації цієї функції потрібно два двухвходових елемента “І-НІ”.

Приведемо до базису “І-НІ” логічний вираз (3.10).

$$S_{(0)} = (A \vee B) \cdot \overline{AB} = \overline{\overline{(A \vee B) \cdot \overline{AB}}} = \overline{\overline{A\overline{B}} \cdot \overline{AB}}.$$

Для реалізації цього виразу теж потрібно сім двовходових елементів “І-НІ” (з урахуванням того, що з $\overline{AB}$ необхідно отримати $P = AB$).

Таким чином, ФПН ЛЕ впливає на основні характеристики логічних схем – структурну складність і глибину. Цю обставину необхідно враховувати при виборі функціонально повного набору логічних елементів.

3.3.4. Однорозрядний суматор комбінаційного типу

Однорозрядним сумматором називається вузол арифметико-логічного пристрою (АЛП) цифрової ЕОМ, призначений для підсумовування одноймен-них розрядів двох чисел з урахуванням перенесення з сусіднього молодшого розряду.

У загальному вигляді однорозрядний суматор являє собою схему, яка має три входи і два виходи. Умовне позначення повного суматора приведено на рис. 3.34.

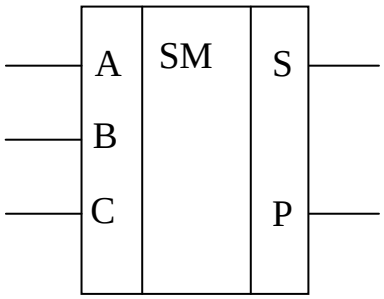


Рис. 3.34. Умовне зображення повного суматора

На входи **A** і **B** подаються сигнали, які відповідають однойменним розрядам доданків. На вхід **C** подається сигнал, що відповідає одиниці перенесення з сусіднього молодшого розряду. Сигнал на виході **S** відповідає сумі трьох цифр, які додаються по модулю два, а сигнал на виході **P** – переносу в старший розряд.

Умови роботи однорозрядного суматора на три входи представимо у вигляді таблиці істинності (табл. 3.26).

Таблиця 3.26

A	**B**	**C**	**S**	**P**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

З цієї таблиці запишемо функції **S** і **P** в аналітичному вигляді в ДДНФ

$$S = \overline{A}\overline{B}C \vee \overline{A}B\overline{C} \vee A\overline{B}\overline{C} \vee ABC , \qquad (3.11)$$

$$P = \overline{A}BC \vee A\overline{B}C \vee AB\overline{C} \vee ABC . \qquad (3.12)$$

Оскільки необхідно побудувати схему, що реалізує два вихідних сигнали, доцільно провести спільну мінімізацію функцій **S** і **P**. При спільної мінімізації висловлюють більш складну логічну функцію через простішу, якщо функції збігаються на половині або більше наборів і через заперечення простіший функції, якщо вони збігаються менше, ніж на половині кількості наборів. Для спільної мінімізацій скористаємося площинними діаграмами (табл. 3.27 і 3.28).

Таблиця 3.27

A	**BC**			
	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Таблиця 3.28

A	**BC**			
	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Простішею функцією є функція **P**, якій відповідає табл. 3.28. Функція **S** (табл. 3.27) практично не спрощується.

Функції **S** і **P** збігаються тільки на двох наборах – на 0-м і 7-м, отже, доцільно виражати функцію **S** через функцію $\overline{\mathbf{P}}$.

Для зручності спільної мінімізації представимо функцію $\overline{\mathbf{P}}$ площинною діаграмою (табл. 3.29).

Таблиця 3.29

A	**BC**			
	00	01	11	10
0	1	1	0	1
1	1	0	0	0

Аналіз табл. 3.29 показує, що функції **S** і $\overline{\mathbf{P}}$ не збігаються тільки на 2-х наборах. На нульовому наборі функція $\overline{\mathbf{P}}$ дорівнює 1, а **S** (табл. 3.27) дорівнює 0 і на 7-му наборі: **P** дорівнює 0, а **S** дорівнює 1.

Для того, щоб виразити функцію **S** через функцію $\overline{\mathbf{P}}$, тобто поставити між ними знак рівності, необхідно зробити так, щоб функція $\overline{\mathbf{P}}$ на нульовому наборі оберталася в нуль, а на сьомому наборі – в одиницю.

Для того, щоб функція на заданому наборі оберталася в нуль, досить помножити її на конституенту нуля цього набору. Отже, функцію $\overline{\mathbf{P}}$ необхідно помножити на $\mathbf{K_{000}(0)}$, тобто на **(AvBvC)**. Для того, щоб функція на заданому наборі дорівнювала одиниці, необхідно додати до неї конституенту одиниці для цього набору. Отже, до функції $\overline{\mathbf{P}}$ необхідно додати $\mathbf{K_{111}(1)}$, тобто **ABC**. Тоді функція **S** виразиться через функцію $\overline{\mathbf{P}}$ наступним чином:

$$S = \overline{P}\cdot(A \vee B \vee C) \vee ABC . \qquad (3.13)$$

За допомогою площинної діаграми (табл. 3.28) зробимо мінімізацію функції **P**

$$P = AB \vee AC \vee BC . \qquad (3.14)$$

Тоді вираз (3.13) з урахуванням (3.14) буде представлений у вигляді

$$S = (AB \vee AC \vee BC)\,(A \vee B \vee C) \vee ABC . \qquad (3.15)$$

Схема, що реалізує вираз (4.11) в базисі “І-АБО-НІ” має вигляд, представлений на рис. 3.35.

Синтезована структурна схема використовується в якості базової при побудові багаторозрядних суматорів комбінаційного типу. Основні характеристики цієї схеми:

структурна складність — C = 20;

структурна глибина — H = 5.

Наведена схема була побудована з умовою задоволення двох заданих критеріїв оптимальності – найменшої структурної складності і найменшої структурної глибини.

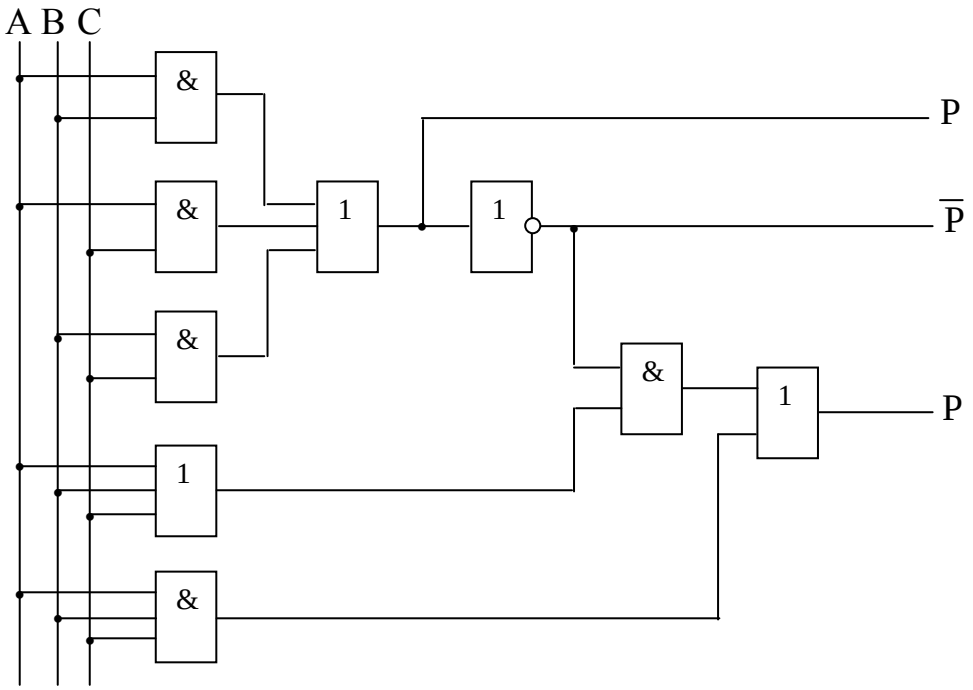


Рис. 3.35. Структурна схема однорозрядного суматора

Якби було потрібно реалізувати тільки критерій найменшої структурної складності, необхідно було б провести додаткові перетворення виразів (3.13) і (3.14) з метою виявлення в них загальних частин. Ці перетворення наступні:

$$S = \overline{P}\cdot(A \vee B \vee C) \ \vee \ ABC = \overline{P}[A \vee (B \vee C)] \vee A(BC)\,, \qquad (3.16)$$

$$P = AB \vee \ AC \vee BC = A(B \vee C) \vee BC\,. \qquad (3.17)$$

Схема суматора, побудована за виразами (3.16) і (3.17) буде мати структурну складність меншу, ніж попередня, але структурна глибина схеми збільшиться.

При необхідності реалізації однорозрядного суматора в універсальних базисах потрібно здійснити перетворення логічних функцій відповідно до раніше викладеної методикою. Так, наприклад, якщо потрібно реалізувати схему суматора в базисі “І-НІ”, то необхідно перетворити мінімальні диз’юнктивні форми логічних функцій за допомогою інверсного закону. Тоді вирази для функцій **S** і **P** будуть мати такий вигляд

$$S = \overline{\overline{A}\overline{B}C \vee \overline{A}B\overline{C} \vee A\overline{B}\overline{C} \vee ABC} = \overline{\overline{A}\overline{B}C}\cdot\overline{\overline{A}B\overline{C}}\cdot\overline{A\overline{B}\overline{C}}\cdot\overline{ABC};$$

$$P = \overline{\overline{AB \vee AC \vee BC}} = \overline{\overline{AB}}\cdot\overline{\overline{AC}}\cdot\overline{\overline{BC}}.$$

3.3.5. Поняття про метод композиції вузлів

Метод композиції вузлів полягає в тому, що на основі простих вузлів здійснюється синтез складніших вузлів. При цьому на етапі структурного синтезу перетворення логічних функцій, що описують функціонування

складного вузла, здійснюється таким чином, щоб вони були виражені через логічні функції, що описують функціонування простих вузлів [10].

Проілюструємо метод композиції вузлів на прикладі синтезу однорозрядного суматора з однорозрядних напівсуматорів.

Для побудови схеми однорозрядного суматора скористаємося виразами

$$S = \overline{A}\,\overline{B}C \vee \overline{A}B\overline{C} \vee A\overline{B}\overline{C} \vee ABC\ ;$$
$$P = \overline{A}BC \vee A\overline{B}C \vee AB\overline{C} \vee ABC\ . \qquad (3.18)$$

Як відомо напівсуматор реалізує для суми логічну залежність $S_{HS} = A\overline{B} \vee \overline{A}B$ і для переносу – $P_{HS} = AB$.

Застосувавши розподільний закон, перетворимо вираження (3.18) для функцій **S** і **P** однорозрядного суматора.

$$S = (\overline{A}B \vee A\overline{B})\overline{C} \vee (AB \vee \overline{A}\,\overline{B})C\ ;$$
$$P = AB(C \vee \overline{C}) \vee (\overline{A}B \vee A\overline{B})C\ . \qquad (3.19)$$

Тоді ці вирази з урахуванням логічних залежностей для суми і перенесу однорозрядного напівсуматора можна записати в наступному вигляді

$$S = S_{HS} \cdot \overline{C} \vee \overline{S}_{HS} \cdot C;$$
$$P = P_{HS} \vee S_{HS} \cdot C, \qquad (3.20)$$

де $\overline{S}_{HS} = \overline{\overline{A}B \vee A\overline{B}} = \overline{\overline{A}B} \cdot \overline{A\overline{B}} = (A \vee \overline{B})(\overline{A} \vee B) = AB \vee \overline{A}\,\overline{B}$.

З виразів (3.20) неважко помітити, що вихід **S** однорозрядного суматора може бути реалізований однорозрядним напівсуматором, якщо на один вхід його подати сигнал $\mathbf{S_{HS}}$, у якого на входи подано сигнали **A** і **B**, а на другий вхід сигнал **C**. На виході **P** такого напівсуматора буде сформований сигнал $\mathbf{S_{HS}} \cdot \mathbf{C}$. Схема однорозрядного суматора, яка реалізована на підставі виразів (3.20) має вигляд, представлений на рис. 3.36.

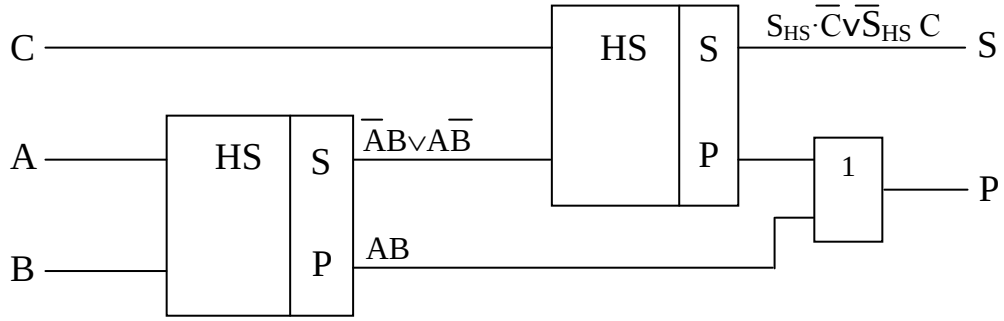


Рис. 3.36. Схема однорозрядного суматора на базі напівсуматора

Таким чином, синтез однорозрядного сумматора проведено на основі методу композиції вузлів. Для того, щоб схема була оптимальною за заданими критеріями необхідно, щоб прості вузли синтезувалися з урахуванням заданих

критеріїв. Очевидною перевагою методу є можливість використання для побудови складних вузлів цифрових ЕОМ більш простих, надійних і взаємозамінних типових вузлів.

3.3.6. Багаторозрядні комбінаційні суматори послідовної дії

Суматор послідовної дії повинен перетворювати послідовні коди доданків в послідовний код їх суми.

Розряди доданків на входи суматора повинні поступати послідовно в часі, починаючи з молодших і з урахуванням переносів з попереднього розряду. Значення перенесу, що виникає у будь-якому розряді, повинне зберігатися в суматорі до поступання на вхід чергових розрядів доданків. Тому суматор послідовної дії повинен мати два вхідні канали для сигналів доданків a_i і $в_i$, вхідний канал для сигналу переносу P_{i-1} з молодших розрядів, вихідний канал для видачі суми S і мати пам'ять для зберігання переносу в наступний розряд.

Залежно від типу елементарного автомата, який забезпечує зберігання переносу, комбінаційні суматори послідовної дії діляться на суматори «із затримкою переносу», якщо елементарні автомати є елементами затримки, або «із запам'ятовуванням переносу», якщо елементарним автоматом є тригер.

Виконаємо синтез комбінаційного суматора, в обох модифікаціях. Для цього складемо таблицю переходів P_i і виходів S_i такого суматора і доповнимо її функціями збудження елементарних автоматів, які будуть використані при синтезі (табл. 3.30). При синтезі суматора із затримкою переносу табл. 3.30 доповнюємо функцією збудження елемента затримки. Вона повинна дорівнювати значенню переноса в i-му розряді, оскільки саме це значення повинне враховуватися в наступному такті.

Таблиця 3.30

a_i	$в_i$	P_{i-1}	S_i	P_i	$q_{ез}$	q_S	q_R
0	0	0	0	0	0	0	–
0	0	1	1	0	0	0	1
0	1	0	1	0	0	0	–
0	1	1	0	1	1	–	0
1	0	0	1	0	0	0	–
1	0	1	0	1	1	–	0
1	1	0	0	1	1	1	0
1	1	1	1	1	1	–	0

Зробивши мінімізацію методом діаграм Вейча (табл. 3.31 – для S_i, 3.32 – для P_i, 3.33 – для $q_{ез}$), отримуємо:

Таблиця 3.31

a_i	$в_i P_{i-1}$			
	00	01	11	10
0		1		1
1	1		1	

Таблиця 3.32

a_i	$в_i P_{i-1}$			
	00	01	11	10
0			1	
1		1	1	1

Таблиця 3.33

a_i	$в_i P_{i-1}$			
	00	01	11	10
0			1	
1		1	1	1

$$S_i = \overline{a}_i \overline{в}_i P_{i-1} \vee \overline{a}_i в_i \overline{P}_{i-1} \vee a_i \overline{в}_i \overline{P}_{i-1} \vee a_i в_i P_{i-1};$$

$$P_i = a_i P_{i-1} \vee a_i в_i \vee в_i P_{i-1};$$

$$q_{ез} = a_i P_{i-1} \vee a_i в_i \vee в_i P_{i-1}.$$

Перші два рівняння представляють рівняння однорозрядного суматора, розглянутого раніше. Внаслідок того, що $q_{ез} = P_i$ можна використати вихід P однорозрядного суматора. Тоді схема багаторозрядного комбінаційного суматора послідовної дії із затримкою переноса матиме вигляд (рис. 3.37).

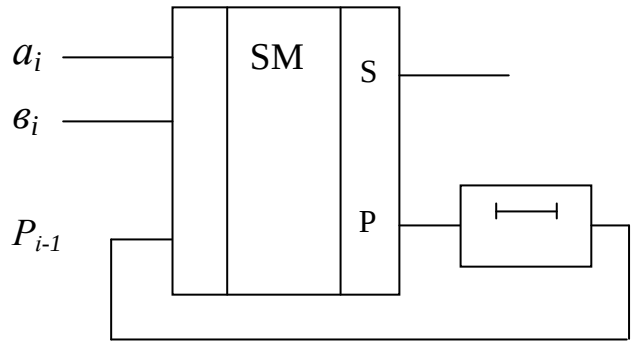


Рис. 3.37. Суматор з затримкою переноса

Побудуємо схему суматора, у якого, як елемент пам’яті використовується тригерний пристрій, наприклад, тригер *RS*. Доповнимо табл. 3.30 функціями збудження *RS*-тригера. Підставивши на невизначених наборах значення функцій q_S і q_R рівні одиниці і зробивши мінімізацію за допомогою діаграм Вейча (табл. 3.34, 3.35), отримаємо:

Таблиця 3.34

a_i	$в_i P_{i-1}$			
	00	01	11	10
0	0	0	–	0
1	0	–	–	1

Таблиця 3.35

a_i	$в_i P_{i-1}$			
	00	01	11	10
0	–	1	0	–
1	–	0	0	0

$$S_i = \overline{a}_i \overline{в}_i P_{i-1} \vee \overline{a}_i в_i \overline{P}_{i-1} \vee a_i \overline{в}_i \overline{P}_{i-1} \vee a_i в_i P_{i-1};$$

$$P_i = a_i P_{i-1} \vee a_i в_i \vee в_i P_{i-1};$$

$$q_S = a_i в_i; \qquad q_R = \overline{a}_i \overline{в}_i.$$

Відповідно до отриманих виразів схема матиме вигляд, показаний на рис. 3.38.

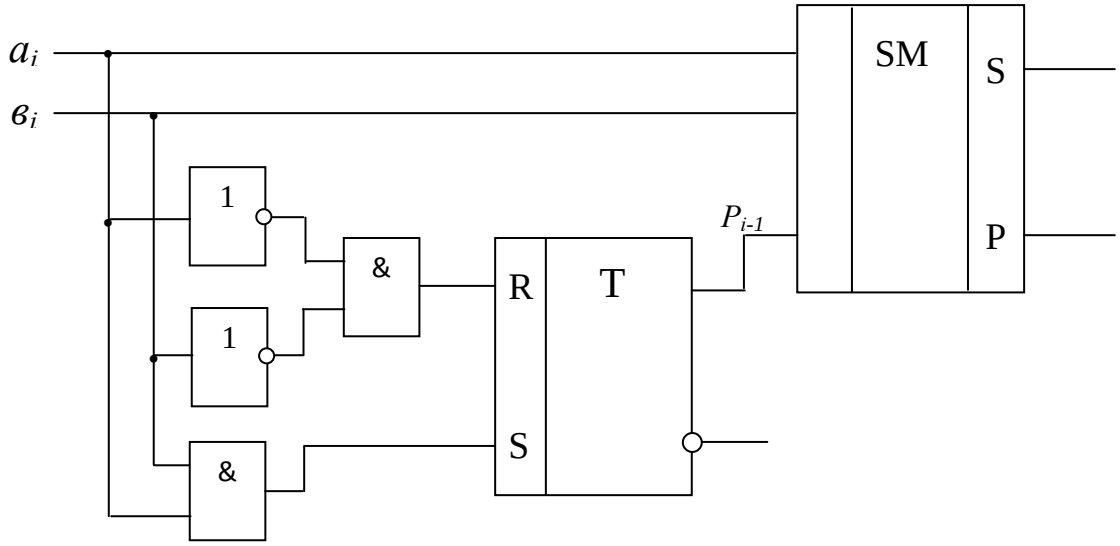


Рис. 3.38. Суматор з запам’ятовуванням переноса

Час додавання в суматорах послідовної дії може бути визначений із співвідношення

$$T_{\Sigma} = n \cdot T,$$

де n – кількість розрядів;

T – період слідування розрядів.

Очевидно, що час додавання в таких суматорах визначається розрядністю чисел, що додаються, і у зв’язку з цим швидкодія суматора послідовної дії не може бути високою. Проте такі суматори володіють тою перевагою, що потрібна кількість устаткування, яке необхідне для організації додавання багаторозрядних чисел, буде мінімальною в порівнянні з іншими типами суматорів.

3.3.7. Багаторозрядні комбінаційні суматори паралельної дії

У паралельних суматорах дії над всіма розрядами доданків виконуються одночасно.

Очевидно, що суматор паралельної дії може бути синтезований методом композиції вузлів з однорозрядних суматорів. Причому кількість їх має дорівнювати числу розрядів доданків, якщо переповнення розрядної сітки при додаванні виключено за рахунок організації обчислень і на одиницю більше, якщо переповнення можливо.

Зв’язок між однорозрядними суматорами повинен бути організований таким чином, щоб вихід P однорозрядного сумматора сусіднього молодшого

розряду з'єднувався зі входом *C* суматора сусіднього старшого розряду. Схема суматора паралельної дії комбінаційного типу представлена на рис. 3.39.

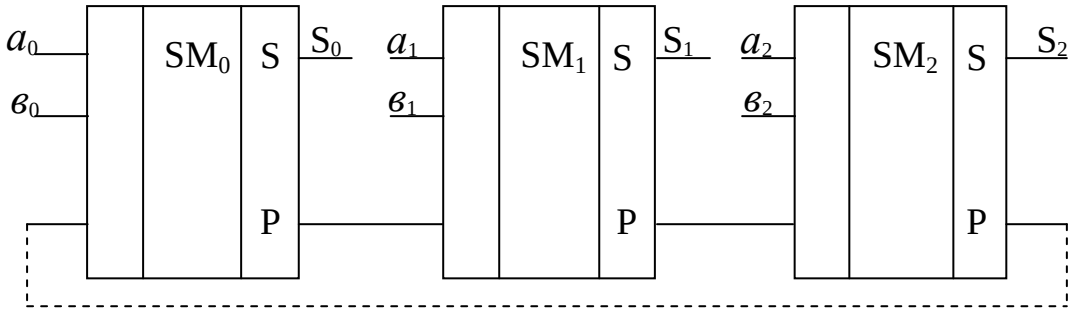


Рис. 3.39. Схема комбінаційного суматора паралельної дії з послідовним переносом

У таких суматорах перенос поширюється послідовно від розряду до розряду, тому суматори такого типу називаються суматорами паралельної дії з послідовним переносом. Суматор забезпечує додавання чисел в доповнювальному або зворотному кодах. При додаванні чисел в зворотному коді необхідно забезпечити циклічний перенос одиниці, що вийшла за знаковий розряд, в молодший розряд і додавання її за звичайними правилами арифметики. Це можна забезпечити зворотним зв'язком, показаним на рис. 3.39 пунктиром.

Час підсумовування в таких суматорах буде складатися з часу додавання цифр в одному розряді і часу поширення переносу, який залежить від кількості розрядів суматора, так як в самому незручному випадку перенос, що виник в самому молодшому розряді, може поширюватися до найстаршого розряду, тобто

$$T_{\sum} = \tau_s + \tau_{\text{п}}(n-1),$$

де n – кількість розрядів доданків;

τ_s – час формування суми в однорозрядному сумматорі;

$\tau_{\text{п}}$ – час формування сигнала переносу на виході однорозрядного суматора.

Переваги розглянутої схеми: висока швидкодія в порівнянні з комбінаційним суматором послідовної дії; простота в порівнянні з іншими схемами комбінаційних суматорів паралельної дії.

Недоліки схеми: значне збільшення обладнання в порівнянні з комбінаційним суматором послідовної дії; мала швидкодія в порівнянні з іншими комбінаційними суматорами паралельної дії.

З метою підвищення швидкості поширення переносу застосовуються спеціальні структурні методи прискорення переноса, які стосуються логіки побудови ланцюгів переносу.

3.4. ФУНКЦІОНАЛЬНІ ВУЗЛИ ЦИФРОВИХ ЕОМ НАКОПИЧУЮЧОГО ТИПУ

3.4.1. Суматор накопичуючого типу з послідовним переносом

В сумматорі накопичуючого типу значення суми може зберігатися скільки завгодно довго. Отже, в суматорі цього типу можна виконувати послідовне додавання декількох чисел. Накопичуючі суматори використовуються тільки як суматори паралельної дії. У якості однорозрядних суматорів в них використовуються тригерні пристрої, які і є елементами пам'яті. На вхід накопичуючого суматора доданки подаються послідовно: спочатку розряди першого доданка (вони запам'ятовуються), потім розряди другого доданка. Значення суми запам'ятовується як стан тригерів [10].

Здійснимо синтез накопичуючого суматора паралельної дії з послідовним переносом. Так як зв'язки між усіма розрядами суматора (крім наймолодшого і сусіднього старшого) однакові, можна обмежитися побудовою схеми одного розряду суматора, а потім за методом композиції вузлів побудувати багаторозрядний суматор. У однорозрядний суматор крім відповідних розрядів доданків подається сигнал переносу з сусіднього молодшого розряду, затриманий щодо моменту приходу другого доданка на час перехідних процесів в тригері. На виході **S** суматора формується сума двох доданків і переносу по модулю 2, а на виході **P** повинен бути сформований сигнал переносу в сусідній старший розряд, якщо ця сума більше або дорівнює двом. У зв'язку з тим, що умови функціонування однорозрядного сумматора визначені, приступимо відразу до етапу структурного синтезу.

В якості елементарного автомата виберемо **T**-тригер, так як на одиничному виході тригера формується сума по модулю 2 сигналів, що надходять на його лічильний вхід. Таблиця станів **T**-тригера має такий вигляд

X	**Q**
0	Q*
1	$\overline{Q}$*

Буквою **Q** позначено стан тригера, а **Q*** – стан тригера в попередній момент часу (до приходу вхідного сигнала). Тоді стан тригера в момент часу ***t+1*** (позначимо **Q'**) в залежності від стану в момент ***t*** і вхідного сигналу буде змінюватися наступним чином

X	**Q**	**Q′**
0	0	0
0	1	1
1	0	1
1	1	0

Але в стан **Q** тригер перейшов під впливом сигналу **X**, поданого в попередній момент часу ***t-1***. Слід зазначити. що початковим станом тригера є стан логічного нуля, в яке суматор переходить перед виконанням операції додавання. Якщо потенціал одиничного виходу тригера позначити буквою **S**, і врахувати, що в стан **Q'** тригер перейшов під впливом сигналу **X**, поданого в момент часу ***t***, то можна записати $\mathbf{Q' = S_{t+i}}$. Отже, потенціал одиничного виходу тригера дійсно є сума по модулю 2 сигналів $\mathbf{X_t}$ і $\boldsymbol{X_{t+1}}$, тобто $\mathbf{S_{t+1} = X_t \oplus X_{t+1}}$. А це означає, що якщо подавати на рахунковий вхід тригера цифри доданків, то на одиничному виході його буде сформовано сигнал, відповідний їх сумі. Але так як вхід у **T**-тригера один, то складові повинні подаватися послідовно в часі, включаючи і сигнал переносу з попереднього молодшого розряду, який необхідно сформувати.

У якості ФПС ЛЕ виберемо систему "І-АБО-НІ" і зробимо кодування вхідних, вихідних сигналів і станів автомата (суматора). Вхідних сигналів три, але перший доданок запам'ятовується тригером в попередній момент часу, тобто $\mathbf{Q_i = a_i}$.

Для кодування вхідних сигналів, що залишилися, досить по одному двійковому розряду, позначимо їх $\boldsymbol{в_i}$ і $\mathbf{P_{i-1}}$.

Вихід суматора **S** буде відповідати стану тригера, а сигнал переноса в сусідній старший розряд позначимо літерою $\mathbf{P_i}$.

Тоді закодована таблиця переходів, виходів і сигналів збудження матиме вигляд (табл. 3.36).

Таблиця 3.36

$\boldsymbol{в_i}$	$\boldsymbol{P_{i-1}}$	$\mathbf{Q_i}$	$\mathbf{Q_i'}$	$\mathbf{Q_i''}$	$\boldsymbol{q_{T1}}$	$\boldsymbol{q_{T2}}$	$\mathbf{P_i}$
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	1	0
0	1	1	1	0	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	0	1	0	1
1	1	0	1	0	1	1	1
1	1	1	0	1	1	1	1

Таблиця дещо незвичайна, так як необхідно розглядати дві функції переходів і дві функції збудження. Це пов'язано з тим, що доданки надходять на вхід суматора по черзі. Значить необхідно забезпечити переходи під дією другого доданка, а потім переходи під дією сигналу переносу. У табл. 3.36 введені наступні позначення:

Q' – стан після подачі розряду доданка $\boldsymbol{в_i}$;

$\boldsymbol{q_{T1}}$ – функція збудження елементарного автомата для забезпечення переходу під впливом сигналу $\boldsymbol{в_i}$;

Q" – стан після подачі сигналу $\mathbf{P}_{i-1}$;

$\boldsymbol{q_{T2}}$ – функція збудження для забезпечення переходу під впливом сигналу $\mathbf{P}_{i-1}$.

Сигнал переносу повинен формуватися, коли сума доданків по модулю 2 стане рівною нулю. У зв'язку з тим, що доданки на вхід тригера подаються послідовно в часі, то як тільки з'являється дві одиниці, повинен бути сформований перенос. Отже, перенос повинен формуватися при переході тригера зі стану 1 в стан 0. Цей перехід відповідає надходженню на вхід тригера двох одиниць.

За допомогою діаграм Вейча (табл. 3.37, 3.38) отримаємо мінімальні форми функцій $\boldsymbol{q_{T1}}$ і $\boldsymbol{q_{T2}}$.

Таблиця 3.37

$\boldsymbol{в_i}$	$\mathbf{P}_{i-1}\mathbf{Q}_i$			
	00	01	11	10
0	0	0	0	0
1	1	1	1	1

Таблиця 3.38

$\boldsymbol{в_i}$	$\mathbf{P}_{i-1}\mathbf{Q}_i$			
	00	01	11	10
0	0	0	1	1
1	0	0	1	1

$\boldsymbol{q_{T1}} = в_i$ $\qquad\qquad$ $\boldsymbol{q_{T2}} = \mathrm{P}_{i-1}$

Значить сигнал на вході **T**-тригера після подачі другого доданка повинен бути $\boldsymbol{в_i}$ а після подачі сигналу переносу – $\mathbf{P}_{i-1}$, а з урахуванням того, що на цей же вхід подається і перший доданок $\mathbf{a}_i$, функція збудження при подачі всіх доданків, матиме вигляд

$$\boldsymbol{q_{Ti}} = \mathbf{a}_i \vee \boldsymbol{в_i} \vee \mathrm{P}_{i-1}.$$

Для $(i+1)$-го розряду — $\boldsymbol{q_{Ti+1}} = \mathbf{a}_{i+1} \vee \boldsymbol{в_{i+1}} \vee \mathrm{P}_i$,

для $(i+2)$-го розряду — $\boldsymbol{q_{Ti+2}} = \mathbf{a}_{i+2} \vee \boldsymbol{в_{i+2}} \vee \mathrm{P}_{i+1}$ і т.д.

Вихід **S** у сумматора вже є, а перенос в сусідній старший розряд можна сформувати по різному. Один з варіантів формування сигналу переносу для випадку, коли тригер переводиться з одного стану в інший імпульсом негативної полярності, складається в підключенні до виходу **Q** тригера диференціюючого ланцюжка, імпульс на виході якого з'являється при переході тригера, з одиничного стану в нульовий. При такій організації переносів схема суматора накопичуючого типу з послідовним переносом має вигляд, представлений на рис. 3.40.

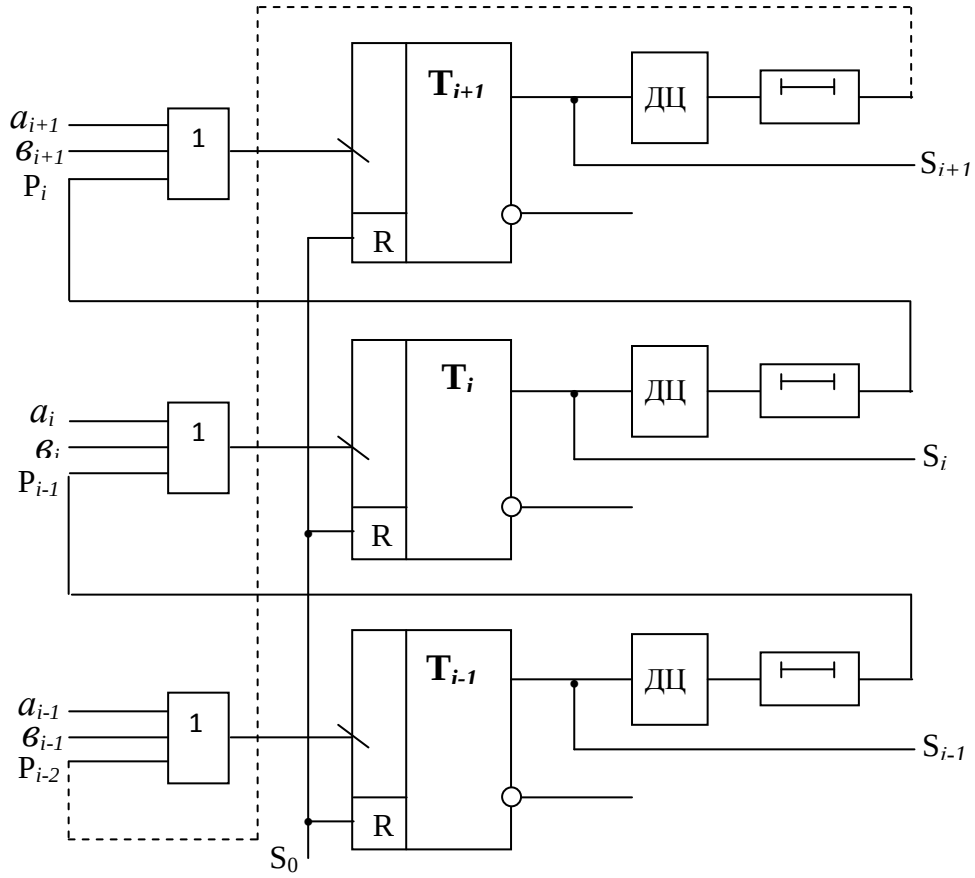


Рис. 3.40. Накопичуючий суматор з послідовним переносом

У ланцюг проходження переносу включені елементи затримки, що забезпечують запізнювання сигналу переносу не менше, ніж на час спрацьовування тригера. Для установки тригера в початковий стан перед додаванням нових чисел використовується сигнал обнулення $\mathbf{S_0}$, який подається на додатковий вхід **R**.

Накопичуючий суматор забезпечує простоту реалізації додавання як в зворотному так і в додатковому кодах. При додаванні чисел в зворотному коді в схемі замикається ланцюг зворотного зв'язку, показаний пунктиром (рис. 3.40).

Максимальний час додавання можна оцінити, провівши аналіз проходження сигналу переноса в найбільш несприятливому випадку

$$T_{\sum \text{макс}} = t_{\text{або}} + t_{\text{тр}} + \left(t_{\text{ез}} + t_{\text{або}} + t_{\text{тр}}\right) \cdot n\,.$$

Перші два доданки відповідають часу порозрядного додавання. Таким чином, швидкодія суматора накопичуючого типу виявляється значно менше, ніж комбінаціонного.

3.4.2. Суматор накопичуючого типу з наскрізним переносом

На підставі табл. 3.36 можна мінімізувати функцію $\mathbf{P_i}$ (табл. 3.39) і побудувати логічну схему для формування сигнала переноса.

Таблиця 3.39

$в_i$	$\mathbf{P_{i-1}Q_i}$			
	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$P_i = в_i Q_i \vee в_i P_{i-1} \vee P_{i-1} Q_i$$

Функція $\mathbf{P_i}$ має такий же вигляд, як і функція переноса однорозрядного суматора комбінаційного типу. Для суматора на три розряди запишемо вирази для функції переноса

$$P_0 = в_0 Q_0, \quad \text{так як} \quad P_{i-1} = 0;$$
$$P_1 = в_1 Q_1 \vee в_1 P_0 \vee P_0 Q_1;$$
$$P_2 = в_2 Q_2 \vee в_2 P_1 \vee P_1 Q_2.$$

Схема переноса, яка побудована відповідно до цих виразів утворює ланцюг наскрізного переносу.

Схема накопичуючого суматора з наскрізним переносом представлена на рис. 3.41.

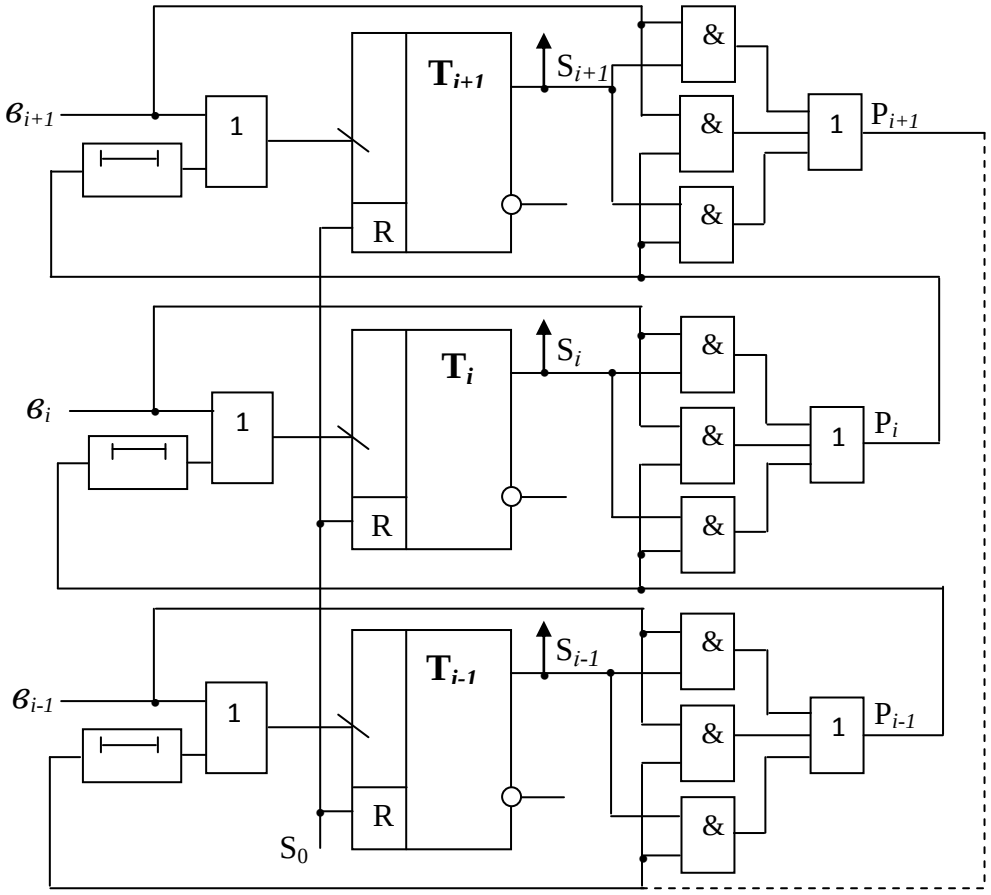


Рис. 3.41. Накопичуючий суматор з наскрізним переносом

Для додавання зворотних кодів чисел необхідно замкнути зворотний зв'язок, показаний пунктиром. Даний суматор відрізняється від попереднього тим, що в ньому паралельно з порозрядним переносом введений ще і наскрізний перенос, що забезпечує передачу імпульсу відразу через всі розряди, тригери яких знаходяться в одиничному стані. Цей імпульс переводить всі тригери, повз яких він проходить, в нульовий стан, а перший тригер, що зустрівся на його шляху і знаходиться в нульовому стані, – в одиничний стан. Час поширення переноса в таких суматорах в гіршому випадку визначається лише сумарною затримкою цих схем і зазвичай значно менше, ніж в суматорах з послідовним переносом. Максимальний час додавання при використанні зворотного коду в суматорі з наскрізним переносом становить

$$T_{\sum \text{макс}} = (t_{\text{і}} + t_{\text{або}}) \cdot n + t_{\text{ез}} + t_{\text{або}} + t_{\text{тр}} .$$

Час порозрядного додавання, рівний $t_{\text{або}}+t_{\text{тр}}$, в максимальному часі підсумовування, враховувати не потрібно, так як формування переноса починається одночасно з порозрядним додаванням.

Якщо у виразах для функції P_i висловити P_0 через $в_0 Q_0$, а P_1 через P_0 , P_2 через P_1 і т. д., то можна отримати схему ланцюга одночасного (паралельного) переноса.

3.4.3. Синтез підсумовуючого лічильника

Загальні відомості про лічильники

Лічильником називається вузол, призначений для підрахунку кількості імпульсних сигналів, що надходять на його вхід, і видачі результатів підрахунку в двійковому коді [2].

Лічильники застосовуються в цифрових ЕОМ для формування адрес команд, лічби кількості циклів при виконанні програми, для підрахунку кількості кроків при виконанні операції множення або ділення, в перетворювачах аналогових величин в код і коду в аналогову величину, в зовнішніх пристроях і т.д.

За видом операції лічби розрізняють підсумовуючі, віднімаючі і реверсивні лічильники. Підсумовуючі лічильники підсумовують одиничні сигнали. Їх показання з приходом кожного вхідного сигналу збільшуються на одиницю. Показання лічильників віднімання з приходом кожного сигналу зменшуються на одиницю. Реверсивні лічильники можуть працювати як в режимі додавання, так і в режимі віднімання сигналів, що надходять на вхід.

Основними параметрами лічильника є модуль лічби ($M_л$) і швидкодія. Модуль лічби визначається числом внутрішніх станів лічильника і дорівнює кількості його різних показань. Оскільки кількість різних двійкових цілих

чисел, які можна записати, маючи n розрядів, дорівнює 2^n, то $M_л = 2^n$. Звідси, кількість розрядів лічильника із заданим модулем лічби може бути отримана із співвідношення

$$n = [\log_2 M_л] \text{ н.б.ц},$$

де н.б.ц − найближче більше ціле число.

Так, при $M_л = 16$ кількість розрядів лічильника $n = 4$. Кількість розрядів, лічильника відповідає кількості елементарних автоматів, необхідних для синтезу лічильника.

Швидкодія лічильника визначається часом його встановлення в новий стан (максимальним часом між моментом надходження лічильного імпульсу і моментом встановлення коду в лічильнику). Швидкодія лічильника залежить від кількості розрядів, способу організації міжрозрядних зв'язків та швидкодії логічних елементів, що використовуються.

Умовне графічне зображення двійкового підсумовуючого лічильника з можливістю встановлення коду та асинхронним входом установки лічильника в 0-й стан приведено на рис. 3.42.

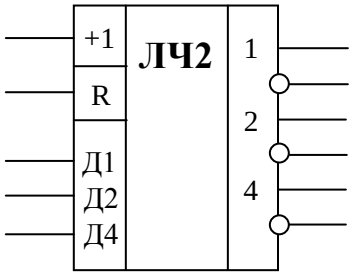


Рис. 3.42. Умовне графічне зображення двійкового лічильника

Синтез підсумовуючого лічильника

Розглянемо синтез підсумовуючого лічильника з $M_л = 8$. Задамо умови функціонування лічильника за допомогою графа. З огляду на те, що $M_л = 8$, лічильник має вісім станів, які змінюються з приходом кожного імпульсу від 000 до 111, граф має 8 вершин. Позначимо вхідні сигнали X, а вихідні − Y.

Граф функціонування підсумовуючого лічильника буде мати вигляд, показаний на рис. 3.43.

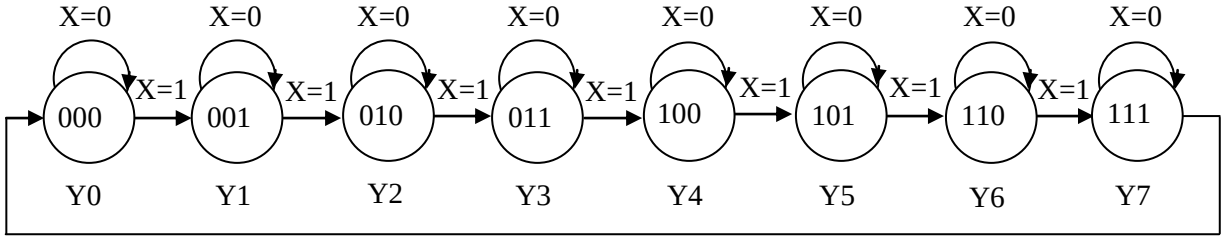


Рис. 3.43. Граф функціонування лічильника з $M_л = 8$

Переходимо до етапу структурного синтезу.

Як елементарний автомат виберемо **Т**-тригер.

Кількість тригерів дорівнює кількості розрядів $n = \log_2 8 = 3$. Як функціонально повний набір елементів використовуємо елементи І, АБО, НІ.

Для кодування вхідних сигналів достатньо одного двійкового розряду. Визначимо його як **X**. Для кодування восьми станів лічильника потрібно три двійкових розряди. Позначимо їх $\mathbf{Q_2}$, $\mathbf{Q_1}$, $\mathbf{Q_0}$. Варіанти кодування станів лічильника, як правило, вибирають так, щоб код кожного стану збігався з числом окремих сигналів, які переводять лічильник із початкового нульового стану в даний стан. В цьому випадку кожний стан лічильника відповідає кількості поданих на його вхід окремих сигналів, тобто відповідає показанням лічильника і, отже, вихідному сигналу лічильника. При такому підході відпадає необхідність кодування вихідних сигналів, оскільки вони відповідають станам.

Сигнали збудження елементарного автомата отримаєм на основі матриці переходів **Т**-тригера (табл. . 3.45).

Складаємо закодовану таблицю переходів та сигналів збудження (табл. 3.46).

Таблиця 3.45

Q	Q'	q
0	0	0
0	1	1
1	0	1
1	1	0

Таблиця 3.46

X	Q_2	Q_1	Q_0	Q'_2	Q'_1	Q'_0	q_2	q_1	q_0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	1	0	0	0	0
0	0	1	1	0	1	1	0	0	0
0	1	0	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0
0	1	1	0	1	1	0	0	0	0
0	1	1	1	1	1	1	0	0	0
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
1	0	1	0	0	1	1	0	0	1
1	0	1	1	1	0	0	1	1	1
1	1	0	0	1	0	1	0	0	1
1	1	0	1	1	1	0	0	1	1
1	1	1	0	1	1	1	0	0	1
1	1	1	1	0	0	0	1	1	1

Мінімізацію функцій збудження $\mathbf{q_1}$ та $\mathbf{q_2}$ проводимо за допомогою площинних діаграм (табл. 3.47 та 3.48).

Таблиця 3.47

XQ_2	Q_1Q_2			
	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	1	0

$q_2 = XQ_0Q_1$

Таблиця 3.48

XQ_2	Q_1Q_2			
	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	0
10	0	1	1	0

$q_1 = XQ_0$

Порівнюючи значення вхідного сигналу і сигналу збудження тригера $\mathbf{T_0}$, видно, що вони рівні, тобто $q_0 = X$.

Залежно від того, який критерій оптимальності заданий як основний, можуть бути отримані різні структури лічильників. Так, наприклад, якщо як основний критерій оптимальності задано максимальну швидкодію, то будувати комбінаційну частину потрібно відповідно до виразів, які отримані після мінімізації. Структура такого лічильника показана на рис. 3.44. Такі лічильники мають назву лічильників з паралельним (одночасним) переносом.

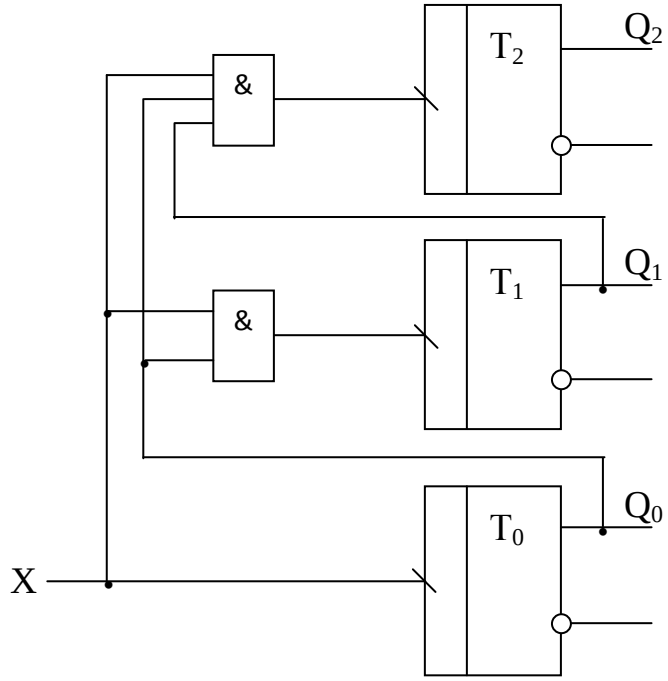


Рис. 3.44. Підсумовуючий лічильник з паралельним переносом

Швидкодія лічильника з паралельним переносом (час встановлення в новий стан) визначається із співвідношення

$$T_{вст} = \tau_{ле} + \tau_{тр},$$

де $\tau_{ле}$ – час перехідних процесів у логічному елементі;

$\tau_{тр}$ – час перехідних процесів у тригері.

Паралельна структура схеми переносів забезпечує максимальну швидкодію, однак в багаторозрядних лічильниках вимагає використання кон'юнкторів зі значним числом входів.

Якщо при синтезі лічильника як основний критерій оптимальності задана найменша структурна складність, то необхідно провести сумісну мінімізацію функцій збудження.

Виразивши одну функцію черев іншу, отримаємо:

$$q_0 = X; \qquad q_1 = q_0 Q_0; \qquad q_2 = q_1 Q_1 .$$

Структура лічильника, який побудований за цими виразами, має вигляд, показаний на рис. 3.45. Такі лічильники отримали назву лічильників з наскрізним переносом.

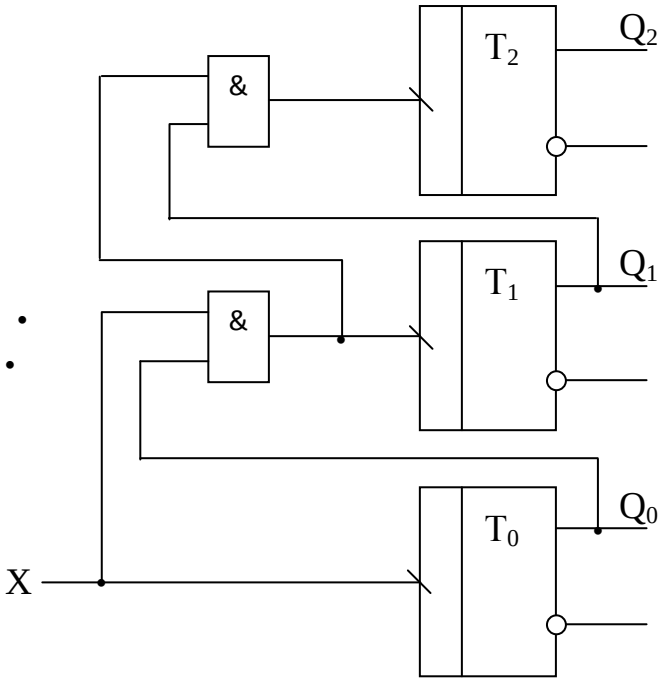


Рис. 3.45. Підсумовуючий лічильник з наскрізним переносом

Час встановлення такого лічильника в новий стан можна визначити із співвідношення:

$$T_{вст} = (n-1)\tau_{ле} + \tau_{тр} ,$$

де n – кількість розрядів лічильника.

Для того щоб мати можливість встановлювати схему в початковий стан (для підсумовуючих лічильників це нульовий стан), при побудові лічильників використовують тригери, які мають установлювальні входи. Структуру лічильника, можна спростити, якщо організувати порозрядний перенос. При цьому на лічильний вхід кожного Т-тригера підключається прямий вихід тригера сусіднього молодшого розряду. На вхід тригера наймолодшого розряду подаються вхідні сигнали. Лічильники, зібрані за таким принципом,

називаються лічильниками з послідовним переносом. Структура такого лічильника подана на рис. 3.46.

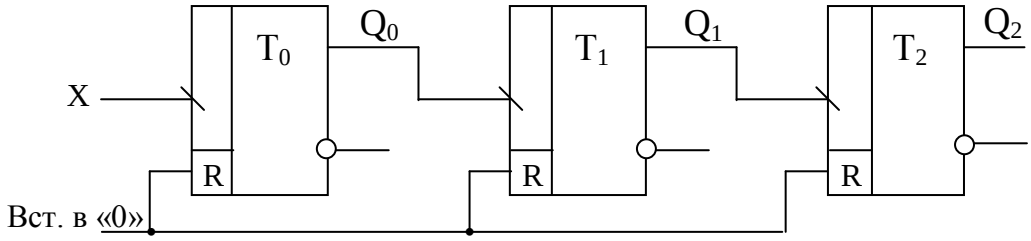


Рис. 3.46. Лічильник з послідовним переносом

Швидкодія лічильника визначається із співвідношення

$$T_{вст} = n\tau_{тр}.$$

У випадку використання імпульсних тригерів, до прямих виходів тригерів підключають диференціюючи ланцюги для формування імпульсів переносу.

3.4.4. Синтез віднімаючого лічильника

З приходом одиничного сигналу на вхід віднімаючого лічильника його показання (число, записане в лічильнику) зменшуються на одиницю.

Проведемо синтез віднімаючого трьохрозрядного лічильника. Оскільки синтез віднімаючого лічильника нічим не відрізняється від синтезу підсумовуючого лічильника, приступимо відразу до етапу структурного синтезу.

У якості елементарного автомата виберемо Т-тригер. Для побудови трьохрозрядного лічильника необхідно мати три тригера. Як функціонально повний набір елементів використовуємо елементи І, АБО, НІ. Зробимо кодування вхідних, вихідних сигналів і станів. Для кодування вхідних сигналів досить одного двійкового розряду. Позначимо його – X. Для кодування восьми станів лічильника потрібно три двійкових розряди. Позначимо, їх A, B, C.

Складемо кодовану таблицю переходів і сигналів збуждения. З огляду на те, що при вхідному сигналі рівному 0 стани лічильника не змінюються і сигнали збудження тригерів при цьому рівні 0, опустимо перші вісім наборів змінних. Кодована таблиця має вигляд (табл. 3.49).

Таблиця 3.49

X	A	B	C	A'	B'	C'	q_A	q_B	q_C
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

За аналогією з підсумовуючим лічильником визначаємо, що $\mathbf{q_C}$ = X. Для мінімізації функцій $\mathbf{q_A}$ і $\mathbf{q_B}$ скористаємося площинними діаграмами.

XA	**BC**			
	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	0	0
10	1	0	0	0

$q_A = X\overline{B}\overline{C}$

XA	**BC**			
	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	0	1
10	1	0	0	1

$q_B = X\overline{C}$

Побудуємо схему віднімаючого лічильника за критерієм максимальної швидкодії (рис. 3.47).

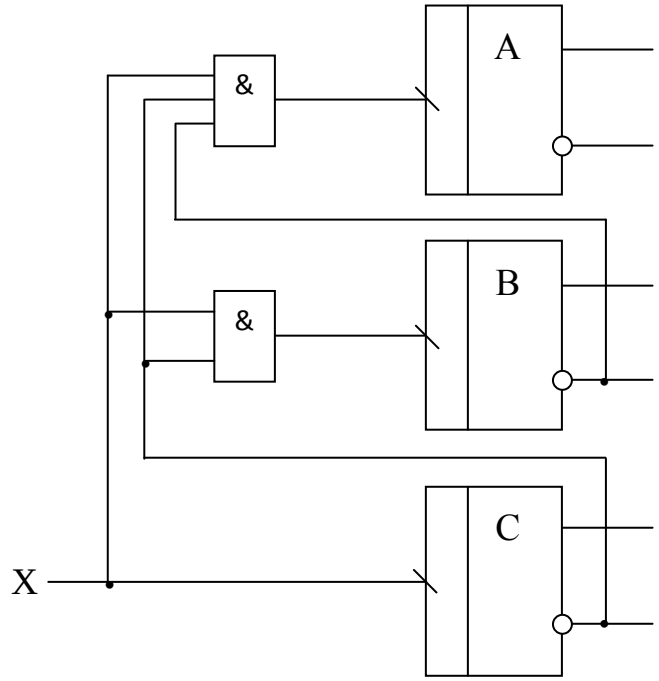


Рис. 3.47. Віднімаючий лічильник з паралельним переносом

Віднімаючі лічильники відрізняються від підсумовуючих тим, що для формування позики (в підсумовуючих лічильниках формували перенос) використовуються сигнали не з прямих виходів розрядів лічильника, а з інверсних.

3.4.5. Лічильники з довільним модулем рахунку

Всі розглянуті вище схеми лічильників мають модуль рахунку $\mathbf{M_л = 2^n}$, де **n** – число розрядів лічильника. На практиці часто виникає необхідність в лічильниках з довільним модулем рахунку, з $\mathbf{M_л \neq 2^n}$.

Загальний принцип побудови лічильників з довільним модулем рахунку полягає у виключенні деяких (зайвих) стійких станів звичайного лічильника, число яких визначається зі співвідношення $\mathbf{L = 2^n - К}$, де **L** – число зайвих станів; **К** – коефіцієнт перерахунку.

Надлишкові стани виключаються за рахунок введення зворотних зв'язків усередині лічильника. Зворотні зв'язки утворюють введенням додаткових логічних ланцюгів, що з'єднують входи і виходи відповідних тригерів [10].

Розрізняють лічильники з природним порядком рахунку і довільним порядком рахунку. У перших показання змінюються послідовно, як в підсумовуючих або віднімаючих лічильниках, наприклад: 000, 001, 010 і т.д. Показання других змінюються довільно, наприклад: 000, 011, 101, 111 і т.д.

Лічильники з довільним модулем рахунку часто використовуються в якості перерахункових схем (подільників частоти сигналів, що надходять на вхід лічильника), у яких вихідними сигналами не є показання лічильника, а тільки сигнал перенесення з старшого розряду. Коефіцієнт перерахунку **К** визначає, після якої кількості вхідних сигналів повинен формуватися вихідний сигнал.

Проведемо синтез перерахункової схеми з природним порядком рахунку з К = 5.

Кількість розрядів лічильника, яке необхідне для забезпечення заданого коефіцієнта перерахунку, визначається зі співвідношення

$$n = [\log_2 K] \text{ н.б.ц, або } 2^{n-1} < K \le 2^n \qquad (3.21)$$

де н.б.ц − найближче більше ціле число.

Для $K = 5 \rightarrow n = 3$.

Визначимо кількість зайвих станів: $L = 2^n - K = 2^3 - 5 = 3$.

Задамо умови функціонування перерахункової схеми за допомогою графа (рис. 3.48).

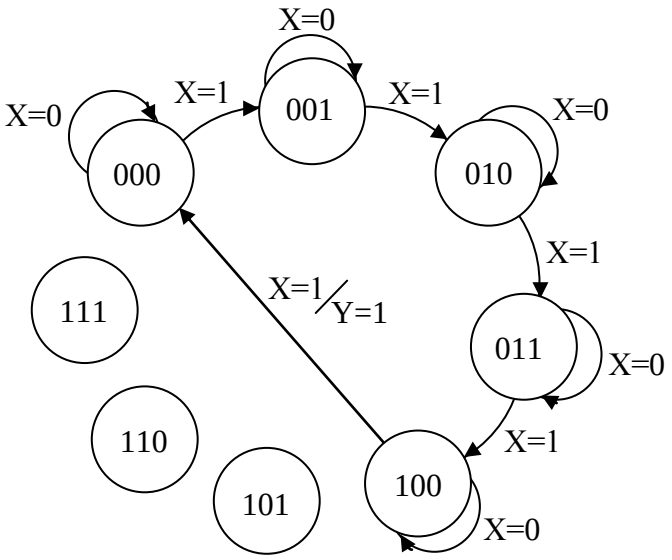


Рис. 3.48. Граф функціонування перерахункової схеми з К = 5

Існує кілька варіантів вибору п’яти послідовних станів. Ці варіанти розрізняються вихідним і кінцевим станами і порядком зміни станів.

Використовуємо в якості елементарного автомата тригер типу **ЈК**.

Для кодування вхідного і вихідного сигналів достатньо по одному двійковому розряду. Позначимо вхідний сигнал **X**, а вихідний – **Y**. Для кодування станів лічильника потрібно три двійкових розряди – позначимо їх **A**, **B**, **C**.

З умов функціонування **ЈК**-тригера, наведених в табл. 3.50, складаємо матрицю переходів елементарного автомата (див. табл. 3.51)

Таблиця 3.50

J	**K**	**Q**
0	0	Q*
0	1	0
1	0	1
1	1	$\overline{Q}$*

Таблиця 3.51

Q	**Q'**	$\mathbf{q_J}$	$\mathbf{q_K}$
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

Кодована таблиця переходів, виходів і сигналів збудження елементарних автоматів має вигляд:

Таблиця 3.52

X	**A**	**B**	**C**	**A'**	**B'**	**C'**	**Y**	$\mathbf{q_{JA}}$	$\mathbf{q_{KA}}$	$\mathbf{q_{JB}}$	$\mathbf{q_{KB}}$	$\mathbf{q_{JC}}$	$\mathbf{q_{KC}}$
0	0	0	0	0	0	0	0	0	–	0	–	0	–
0	0	0	1	0	0	1	0	0	–	0	–	–	0
0	0	1	0	0	1	0	0	0	–	–	0	0	–
0	0	1	1	0	1	1	0	0	–	–	0	–	0
0	1	0	0	1	0	0	0	–	0	0	–	0	–
0	1	0	1	1	0	1	0	–	0	0	–	–	0
0	1	1	0	1	1	0	0	–	0	–	0	0	–
0	1	1	1	1	1	1	0	–	0	–	0	–	0
1	0	0	0	0	0	1	0	0	–	0	–	1	–
1	0	0	1	0	1	0	0	0	–	1	–	–	1
1	0	1	0	0	1	1	0	0	–	–	0	1	–
1	0	1	1	1	0	0	0	1	–	–	1	–	1
1	1	0	0	0	0	0	1	–	1	0	–	0	–
1	1	0	1	–	–	–	–	–	–	–	–	–	–
1	1	1	0	–	–	–	–	–	–	–	–	–	–
1	1	1	1	–	–	–	–	–	–	–	–	–	–

Для мінімізації функції Y і функцій збудження скористаємося площинними діаграмами.

XA	BC			
	00	01	11	10
00	0	0	0	0
01	–	–	–	–
11	–	–	–	–
10	0	0	1	0

$q_{JA} = XBC$;

XA	BC			
	00	01	11	10
00	–	–	–	–
01	0	0	0	0
11	1	–	–	–
10	–	–	–	–

$q_{KA} = X$.

XA	BC			
	00	01	11	10
00	0	0	–	–
01	0	0	–	–
11	0	–	–	–
10	0	1	–	–

$q_{JB} = XC$;

XA	BC			
	00	01	11	10
00	–	–	0	0
01	–	–	0	0
11	–	–	–	–
10	–	–	1	0

$q_{KB} = XC$.

XA	BC			
	00	01	11	10
00	0	0	0	0
01	0	–	–	0
11	0	–	–	–
10	1	–	–	1

$q_{JC} = X\overline{A}$;

XA	BC			
	00	01	11	10
00	–	0	0	–
01	–	0	0	–
11	–	–	–	–
10	–	1	1	–

$q_{KC} = X$.

XA	BC			
	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	–	–	–
10	0	0	0	0

$Y = XA$.

Схема, побудована за отриманими виразами для функцій збудження, показана на рис. 3.49.

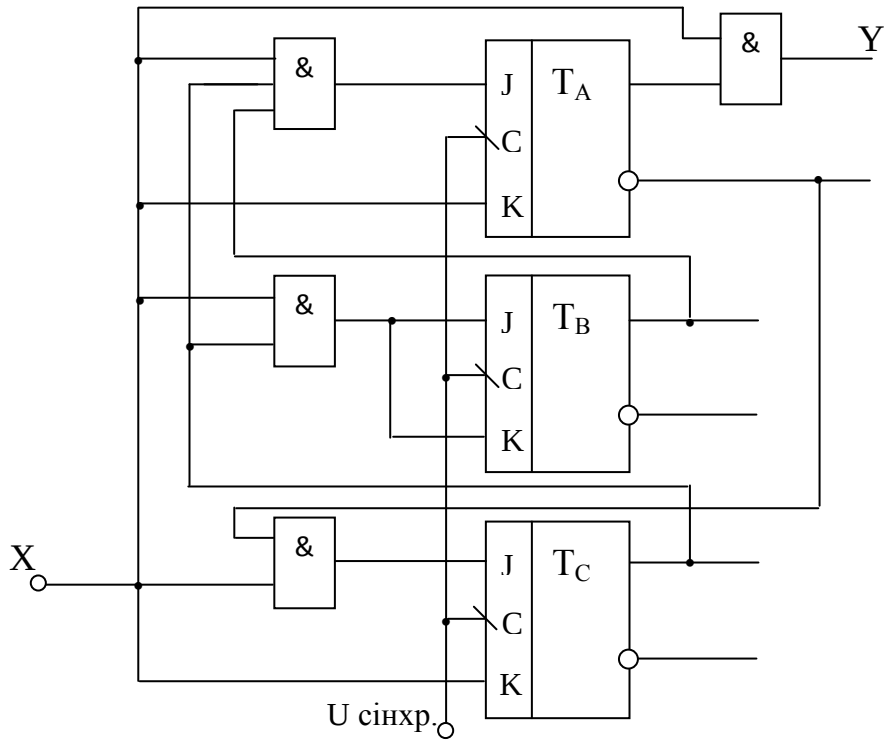


Рис. 3.49. Перерахункова схема з К = 5 на JK-тригерах

Методика побудови перерахункових схем з великим коефіцієнтом перерахунку

Розглянутий приклад показує, що методом структурного синтезу можна побудувати схеми лічильників з будь-яким коефіцієнтом перерахунку. Однак при досить великих ***K*** вихідний сигнал і функції збудження виявляються залежними від великого числа змінних, а це ускладнює їх мінімізацію. Тому на практиці зазвичай використовуються схеми, одержувані зі звичайних підсумовуючих або віднімаючих лічильників шляхом введення зворотних зв'язків, за допомогою яких виключаються зайві стани.

Методика побудови перерахункових схем при цьому наступна:

по заданому коефіцієнту перерахунку визначається необхідна кількість розрядів лічильника зі співвідношення

$$n = [\log_2 K] \quad н.б.ц, \quad або \quad 2^{n-1} < K \le 2^n;$$

визначається кількість зайвих станів, які необхідно виключити:

$$L = 2^n - K;$$

по числу виключених станів, записаному в довійковому ***n***-розрядному коді, визначається спосіб організації зворотних зв'язків і спосіб з'єднання тригерів лічильника між собою.

Залежно, від порядка рахунку розрізняють перерахункові схеми з однорідними зв'язками і комбінованими зв'язками.

У перерахункових схемах з однорідними зв’язками число виключених станів записується в лічильник при установці схеми в початковий стан, а потім кожен вихідний сигнал по ланцюгу зворотного зв’язку робить запис числа виключених станів в лічильнику. В таких схемах рахунок проводиться в природному порядку.

При побудові перерахункових схем з однорідними зв’язками число виключених станів, представлене в двійковому коді, показує, в які розряди лічильника необхідно записати одиниці для установки схеми в початковий стан і по ланцюгу зворотного зв’язку.

Нехай потрібно побудувати перерахункових схему з $K = 5$.

Визначимо кількість розрядів лічильника **n**:

$$2^{n-1} < 5 \le 2^n; \quad \rightarrow \quad 2^2 < 5 \le 2^3; \quad \rightarrow \quad n = 3.$$

Кількість зайвих станів дорівнює:

$$L = 2^n - K = 2^3 - 5 = 3.$$

Представляємо L в трьохрозрядному двійковому коді:

$$L = 011.$$

У якості елементарних автоматів при побудові таких схем дуже часто використовуються *RST*-тригери, так як вони дозволяють виконувати початкову установку лічильника в початковий стан. Перерахункова схема з К = 5 з однорідними зв’язками показана на рис. 3.50.

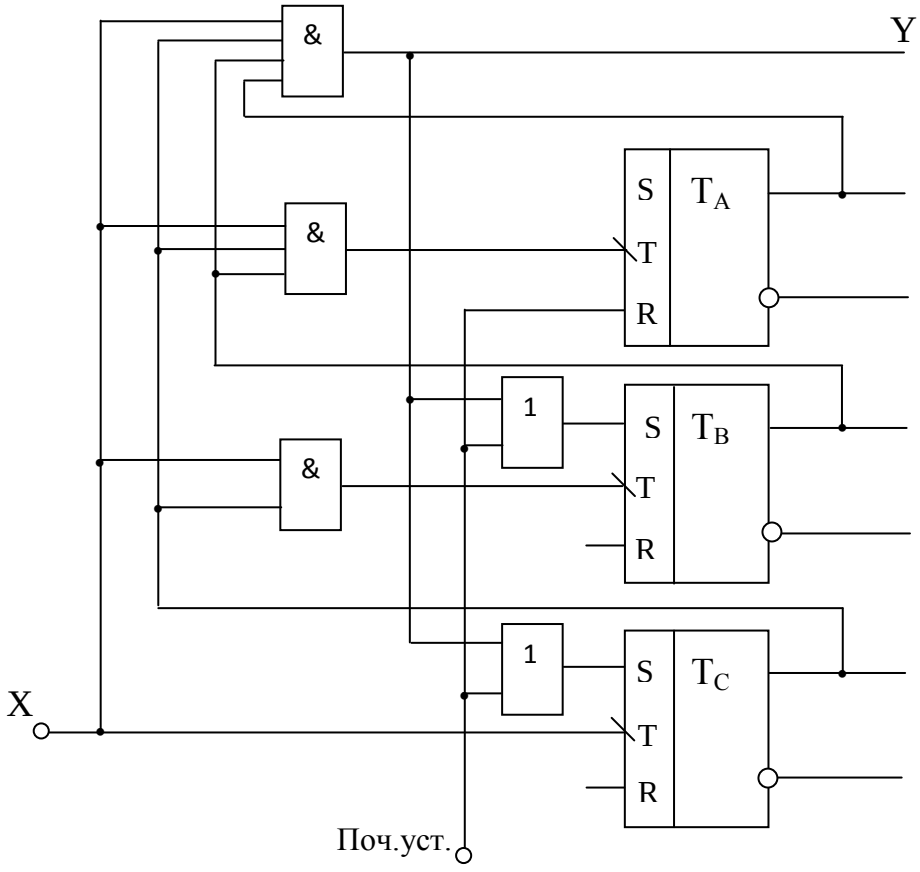


Рис. 3.50. Перерахункова схема з однорідними зв’язками з К = 5

Перёд початком роботи лічильника проводиться початкова установка його в стан 011. Таким чином, будуть виключені стани 000, 001 і 010. Перший імпульс, що надійшов на вхід схеми буде для лічильника четвертим, другий – п’ятим і т.д. Після приходу четвертого імпульсу лічильник переходить в стан 111, а п’ятий імпульс проходить на вихід **Y** схеми і по ланцюгу зворотного зв’язку через елементи АБО встановлює схему знову в стан 011. Таким чином, на виході **Y** формується одиничний сигнал після подачі на вхід схеми п’яти імпульсів.

Інший спосіб виключення надлишкового числа станів лічильника реалізується в перерахункової схемі з комбінованими зв’язками, а саме шляхом використання одиничних і нульових виходів тригерів для організації межразрядних з’єднань.

При побудові перерахункових схем з комбінованими зв’язками число виключених станів, представлене в ***n***-розрядному двійковому коді, показує:

по-перше, що в тих розрядах лічильника, які відповідають одиницям в числі виключених станів, перенос необхідно брати з нульового виходу тригера;

по-друге, що сигнал одиниці по ланцюгу зворотного зв’язку повинен бути поданий на нульові входи тригерів цих же розрядів.

Нехай потрібно побудувати перерахункову схему з комбінованими зв’язками з К = 6. Як і в попередньому випадку, кількість розрядів лічильника $n = 3$.

Кількість виключених станів дорівнює: $L = 2^n - K = 2^3 - 6 = 2$.

Представляємо L в трьохрозрядному двійковому коді: $L = 010$.

Перерахункова схема з комбінованими зв’язками з К = 6 показана на рис. 3.51.

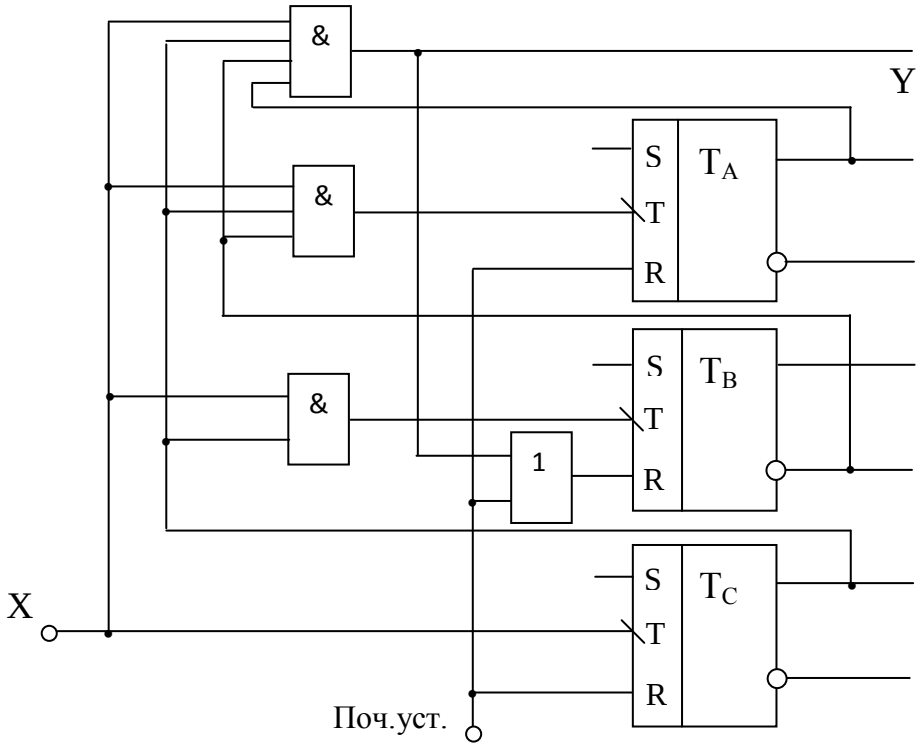


Рис. 3.51. Перерахункова схема з комбінованими зв’язками з К = 6

У перерахункових схемах з комбінованими зв’язками стани змінюються непослідовно. У цьому неважко переконатися, проаналізувавши роботу схеми, наведеної на рис. 3.51. Після приходу першого імпульсу в лічильнику буде записано двійковечисло 001, а після приходу другого імпульсу в лічильнику виявиться записаним число 111.

Промисловістю випускаються перерахункові схеми зі змінним коефіцієнтом ділення.

Подільники частоти із змінним коефіцієнтом ділення випускаються у вигляді окремих інтегральних мікросхем (ІМС) наприклад типу 155ИЕ8, умовне позначення якої показано на рис. 3.52.

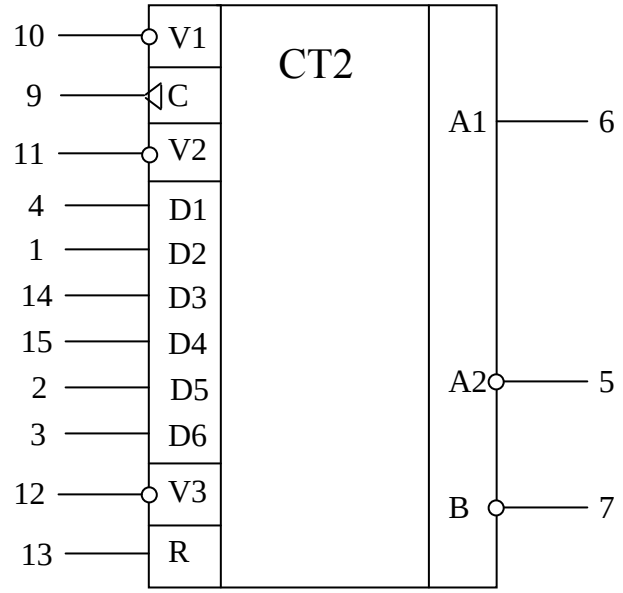


Рис. 3.52. Умовне позначення подільника частоти із змінним коефіцієнтом ділення (ІМС типу 155ИЕ8)

ІМС має: тактовий рахунковий вхід **С**; вхід установки лічильника в нульовий стан **R**; стробуючий вхід **V1**, який управляє дешифратором подільника частоти (дешифратор включається сигналом, логічного нуля, на вході **VI**); буферний дозволяючий вхід **V2** для дозволу режиму рахунку; шість інформаційних входів **01 - 06** для завдання коду коефіцієнта ділення частоти; вхід **V3** для управління виходом; виходи **A1** і **A2** для формування вихідних імпульсів в прямому і інверсному кодах; дозволяючий вихід **B**, який використовується при необхідності каскадного включення ІМС для збільшення розрядності коефіцієнта ділення частоти.

3.4.6. Загальні відомості про регістри

Регістром називається вузол, призначений для запам’ятовування машинного слова або окремих його частин. Крім того, за допомогою регістрів можуть виконуватися деякі операції над словами:

перетворення паралельного коду в послідовний і навпаки;

перетворення прямого коду числа в зворотний і навпаки;

зсув слова на задану кількість розрядів вправо або вліво;

порозрядне роз'єднання;

порозрядне логічне множення;

порозрядне порівняння слів.

Класифікація регістрів, пов'язана з особливостями їх функціонування, визначається в основному способами уявлення інформації, яка надходить на регістр і видається з регістра [13].

Двійкові слова по інформаційних каналах можуть передаватися паралельним, послідовним і паралельно-послідовним кодами.

При паралельному способі передачі все розряди n-розрядного двійкового слова передаються паралельно, одночасно по n каналах. Це найбільш швидкодіючий спосіб передачі.

Передача двійкового слова послідовним кодом виконується послідовно, розряд за розрядом, старшим або молодшим розрядом вперед. При цьому для передачі n-розрядного слова потрібно в n разів більше часу, ніж при передачі його паралельним кодом, але в n раз зменшуються витрати обладнання (підсилювачі, передаючі і прийомні логічні схеми і т.д.).

Проміжні оцінки по обладнанню та швидкодії дає паралельно-послідовний спосіб передачі. Багаторозрядне двійкове слово ділиться на групи по m розрядів, які передаються послідовно, група за групою. При цьому кожна група передається паралельним кодом. Прикладом передачі двійкового слова таким способом може служити потетрадна передача двійково-десяткового коду деякого числа.

За способом передачі двійкових слів до регістру і від регістра розрізняють регістри трьох типів:

регістри паралельної дії (паралельні регістри);

регістри послідовної дії (послідовні або зсувуючі регістри);

регістри паралельно-послідовної дії.

3.4.7. Синтез регістрів паралельної дії

У регістрах паралельної дії запис і видача інформації здійснюється паралельним кодом.

Частина регістра, призначена для зберігання значення одного двійкового розряду слова, називається розрядом регістра. Для зберігання n-розрядного двійкового числа регістр повинен мати n розрядів.

Залежно від способу записа значень двійкових розрядів слова розрізняють двотактні і однотактні регістри [10].

Для запису інформації в двотактний регістр потрібно два такти: такт стирання попередньої інформації (установка регістра в нульовий стан) і такт запису нової інформації. Запис інформації в однотактний регістр проводиться за один такт. Умовне графічне зображення двотактного регістра паралельної дії наведено на рис. 3.53.

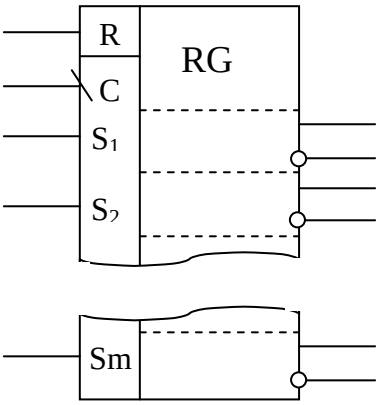


Рис. 3.53. Умовне графічне зображення двотактного регістра

При синтезі регістрів паралельної дії в якості елементарних автоматів використовуються тригери.

Зробимо синтез двотактного регістра паралельної дії. У зв’язку з тим, що міжрозрядні зв’язки в регістрі паралельної дії відсутні, досить побудувати схему одного розряду регістра.

Найменування входів:

R – вхід для установки регістра в стан 0;

C – виконавчий керуючий вхід для здійснення прийому інформації. Вхід синхронізації;

$S_1 \div S_m$ – входи регістра для прийому коду числа.

Причому, $m = 2^{n-1}$, тобто символ позначення вхідного сигналу відповідає вазі двійкового розряду, що подається на цей вхід.

Здійснимо кодування вхідних, вихідних сигналів і станів. Вхідних сигналів три. Для їх кодування досить по одному двійковому розряду. Позначимо їх так само, як вхідні сигнали: ***R***, ***C*** і ***S***. З умов функціонування регістра паралельної дії випливає, що він є автоматом Мура. У зв’язку з цим відпадає необхідність в кодуванні вихідного сигналу, він визначається станом автомата. Позначимо стан тригера розряду регістра **Q**, а стан в наступний момент часу – **Q'**. Тоді узагальнена таблиця переходів, виходів і сигналів збудження матиме вигляд (табл. 3.53).

Таблиця 3.53

R	C	S	Q	Q'	q_R	q_S
0	0	0	0	0	–	0
0	0	0	1	1	0	–
0	0	1	0	0	–	0
0	0	1	1	1	0	–
0	1	0	0	0	–	0
0	1	0	1	1	0	–
0	1	1	0	1	0	1
0	1	1	1	1	0	–
1	0	0	0	0	–	0
1	0	0	1	0	1	0
1	0	1	0	0	–	0
1	0	1	1	0	1	0
1	1	0	0	–	–	–
1	1	0	1	–	–	–
1	1	1	0	–	–	–
1	1	1	1	–	–	–

Для перших чотирьох наборів відсутні сигнали обнулення і запису і, отже, стан тригера не змінюється.

Для других чотирьох наборів відсутній сигнал обнулення, а це означає, що замість 1 не можна записати 0.

Для третьої четвірки наборів сигнал обнулення дорівнює 1, але сигнал запису дорівнює 0. Отже, майбутній стан тригера Q' для усіх цих наборів дорівнює 0.

Для останніх чотирьох наборів і сигнали обнулення, і сигнали запису дорівнюють одиниці, що відповідає одночасному поданню сигналів обнулення і запису, що управляють. Але по умові функціонування двотактного регістра має бути поданий сигнал обнулення, а потім сигнал запису. Тому на цих наборах стан Q' є невизначеним.

Мінімізацію функцій збудження проведемо за допомогою площинних діаграм.

RC	**SQ**			
	00	01	11	10
00	–	0	0	–
01	–	0	0	0
11	–	–	–	–
10	–	1	1	–

Довизначивши функцію $\boldsymbol{q_R}$ відповідним чином і склеївши вісім конституент одиниць, як показано на діаграмі, отримаємо

$$q_R = \text{R}.$$

RC	**SQ**			
	00	01	11	10
00	0	–	–	0
01	0	–	–	1
11	–	–	–	–
10	0	0	0	0

Виконавши аналогічні операції над функцією $\boldsymbol{q_S}$ отримаємо

$$q_S = \text{CS}.$$

Структурна схема одного розряду двотактного регістра паралельної дії матиме вигляд, показаний на рис. 3.54.

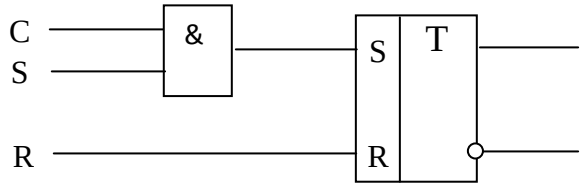


Рис. 3.54. Схема одного розряду двотактного регістра паралельної дії

Схема n-розрядного двотактного регістра матиме вигляд (рис. 3.55).

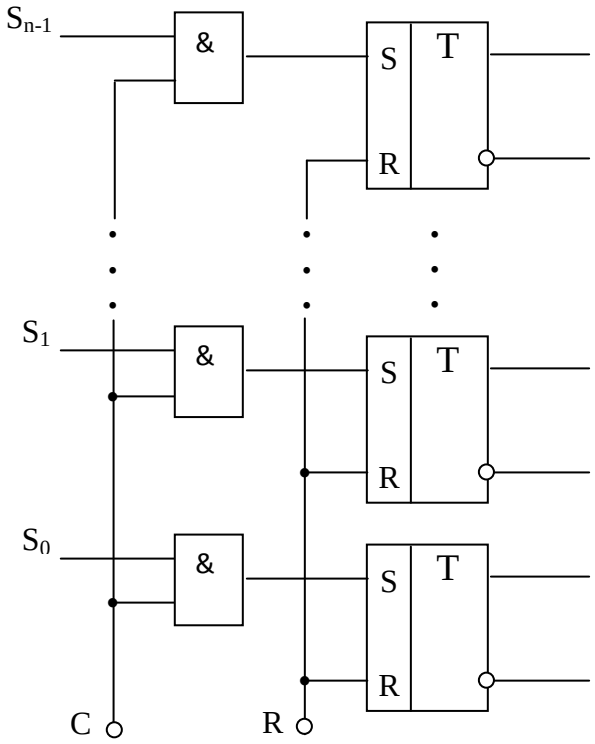


Рис. 3.55. Схема n-розрядного двотактного регістра паралельної дії

На практиці не лише прийом інформації в регістрі робиться по спеціальній команді, але і видача інформації з регістра здійснюється по спеціальному керівному сигналу.

Проведемо аналіз однотактного регістра паралельної дії з урахуванням організації видачі інформації з регістра. Позначимо буквою $\boldsymbol{K}$ управляючий сигнал видачі інформації, тоді узагальнена таблиця переходів, виходів і сигналів збудження тригера одного розряду такого регістра матиме вигляд, представлений табл. 3.54.

Таблиця 3.54

$\boldsymbol{C}$	$\boldsymbol{K}$	$\boldsymbol{S}$	Q	Q'	Y	q_R	q_S
0	0	0	0	0	0	–	0
0	0	0	1	1	0	0	–
0	0	1	0	0	0	–	0
0	0	1	1	1	0	0	–
0	1	0	0	0	0	–	0
0	1	0	1	1	1	0	–
0	1	1	0	0	0	–	0
0	1	1	1	1	1	0	–
1	0	0	0	0	0	–	0
1	0	0	1	0	0	1	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	0	–
1	1	0	0	–	–	–	–
1	1	0	1	–	–	–	–
1	1	1	0	–	–	–	–
1	1	1	1	–	–	–	–

Для перших чотирьох наборів немає сигналу запису і немає сигналу видачі коду, тому тригер залишатиметься в попередньому стані, а вихідний сигнал дорівнює 0.

Для наступних чотирьох наборів сигналу запису немає, а сигнал видачі коду дорівнює 1, отже, тригер залишиться в попередньому стані, а вихідний сигнал визначатиметься цим станом.

Для набору з 8 по 11 сигнал $\boldsymbol{C} = 1$, а сигнал $\boldsymbol{K} = 0$, отже, тригер перейде в стани, визначувані значенням коду вхідного сигналу, а вихідний сигнал дорівнює 0.

Для наборів з 12 по 15 сигнали $\boldsymbol{C}$ і $\boldsymbol{K}$ рівні 1, а це суперечить умовам функціонування регістра, тому і майбутній стан тригера, і вихідний сигнал можна вважати невизначеними.

Мінімізацію функцій збудження і функції **Y** проведемо за допомогою площинних діаграм.

CK	**SQ**			
	00	01	11	10
00	–	0	0	–
01	–	0	0	–
11	–	–	–	–
10	–	1	0	0

$q_R = C\overline{S}$;

CK	**SQ**			
	00	01	11	10
00	0	–	–	0
01	0	–	–	0
11	–	–	–	–
10	0	0	–	1

$q_S = CS$.

CK	**SQ**			
	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	–	–	–	–
10	0	0	0	0

$Y = KQ$.

Схема однорозрядного регістра паралельної дії представлена на рис. 3.56.

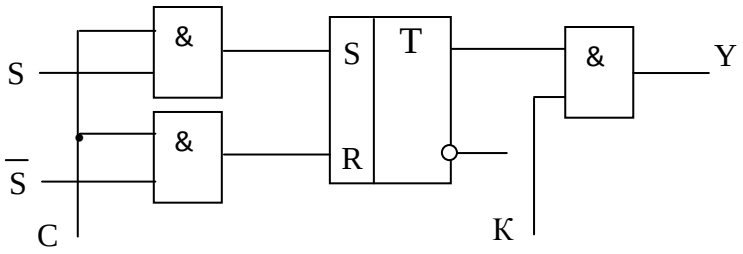


Рис. 3.56. Схема одного розряду однотактного регістра паралельної дії

В однотактних регістрах цифра кожного розряду повинна подаватися в прямому і зворотному коді. Такі коди називаються парафазними. По назві коду і регістри часто називають парафазними регістрами. Схема $\boldsymbol{n}$-розрядного однотактного регістра паралельної дії матиме вигляд, показаний на рис. 3.57.

За допомогою регістрів паралельної дії над машинними словами можна виконувати такі операції, як логічне додавання, логічне множення, порозрядне порівняння і перетворення прямого коду в зворотний і навпаки.

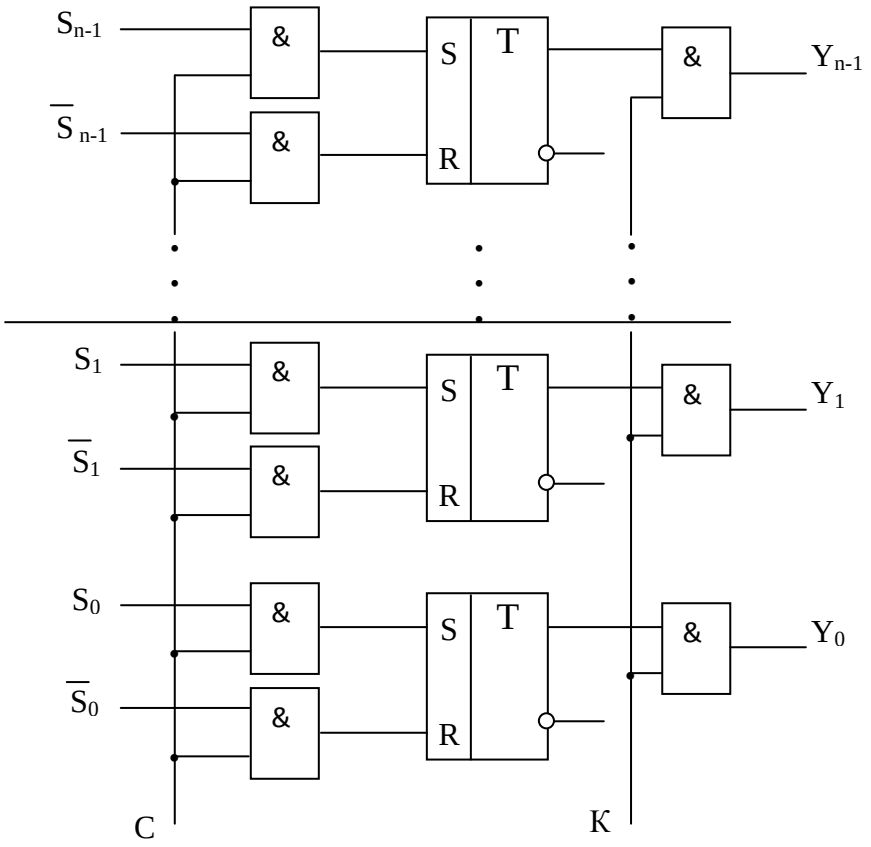


Рис. 3.57. Схема *n*-розрядного однотактного регістра паралельної дії

Розглянемо принципи виконання операцій, які можна виконувати за допомогою регістрів паралельної дії.

Логічне додавання (порозрядна диз'юнкція двійкових слів). Цю операцію над ***n***-розрядными словами можна виконати на ***n***-розрядном регістрі на ***RS***-тригерах, послідовно подаючи на ***S***-входи його тригерів однойменні розряди слів. Після подання останнього слова в регістрі опиниться результат, рівний порозрядній диз'юнкції усіх поданих на його вхід слів.

Логічне множення (порозрядна кон'юнкція слів). Операція виконується аналогічно попередній операції, якщо скористатися інверсним законом, відповідно до якого $\mathbf{X_1 \cdot X_2 \cdot ... \cdot X_n} = \overline{\overline{\mathbf{X}}_1 \vee \overline{\mathbf{X}}_2 \vee ... \vee \overline{\mathbf{X}}_n}$. Є два варіанти реалізації цього співвідношення. У першому варіанті на ***S***-входи заздалегідь встановленого в нульовий стан регістра вимагається подавати інверсні значення розрядів слів (зворотний код), а результат операції знімати з інверсних виходів розрядів регістра. Другий варіант вимагає попередньої установки усіх розрядів в одиничний стан і подання значень розрядів слів на ***R***-входи тригерів регістра, при цьому результат знімається з прямих виходів тригерів. Ці ж логічні операції над двома словами можуть виконуватися за допомогою двох регістрів, з'єднаних так, як показано на рис. 3.58.

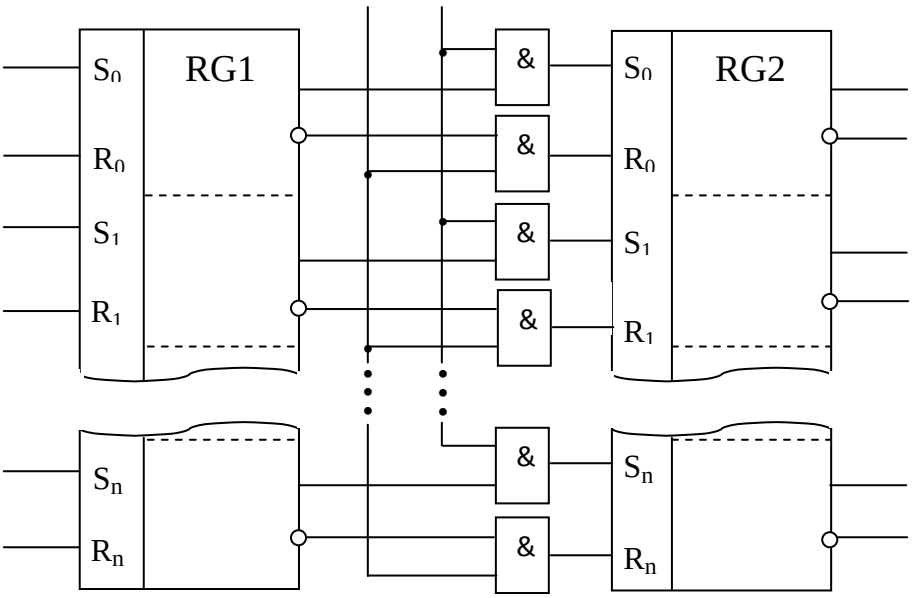


Рис. 3.58. Реалізація логічних операцій додавання і множення
за допомогою двох регістрів

Порівняння двох слів. Операцію визначення рівності (нерівності) двох слів можна виконати на регістрі, побудованому на ***T***-тригерах. Як відомо, ***T***-тригер виконує логічну операцію додавання по модулю два. Тому для визначення рівності двох слів необхідно подати їх одно за одним на ***T***-входи тригерів регістра. Якщо слова рівні, то усі тригери знаходитимуться в нульовому стані.

Перетворення прямого коду в зворотний і навпаки можна здійснити як при записі слів в регістр, так і при видачі їх з регістра. Схема одного розряду регістра з вхідною логікою, що перетворює по відповідних сигналах прямий код в зворотний і навпаки, показана на рис. 3.59.

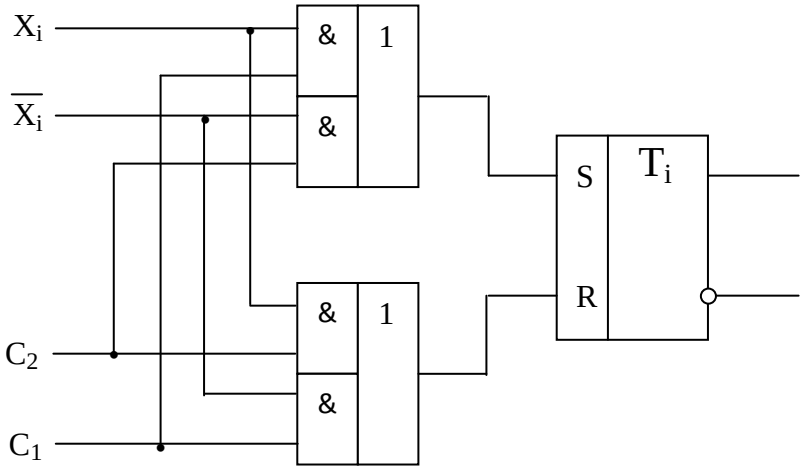


Рис. 3.59. Схема одного розряду регістра з інвертуванням числа на вході

Оскільки значення розрядів числа подаються на регістр порозрядно, то принцип перетворення полягає в перекомутації каналів, по яких поступають їх

прямі і інверсні значення. По сигналу $\mathbf{C_1}$ значення двійкового розряду $\mathbf{X_i}$ записується в тригер без інвертування, а по сигналу $\mathbf{C_2}$ - з інвертуванням.

Для перетворення коду при видачі з регістра необхідно організувати ланцюг видачи, аналогічний показаному на рис. 3.57, але в цьому випадку вентилі повинні управлятися потенціалами нульових виходів тригерів.

3.4.8. Синтез регістрів послідовної дії

Запис двійкового слова в регістр послідовної дії здійснюється послідовним кодом, а в регістр паралельно-послідовної дії − послідовним або паралельним кодом. Прийняте в регістр слово може бути зсунуте у бік старших (вліво) або молодших (управо) розрядів на задане число розрядів.

Регістри паралельно-послідовної дії і послідовно дії використовуються для перетворення паралельного коду в послідовний і навпаки. Перетворення послідовного коду в паралельний здійснюється шляхом зсуву інформації, що приймається послідовним кодом. Зсув здійснюється під впливом сигналу (імпульсу зсуву), що управляє, який подається на усі розряди регістра одночасно. Після того, як увесь код прийнятий в регістр, він видається усіма розрядами одночасно. Для перетворення паралельного коду в послідовний число, прийняте в регістр паралельним кодом, зсовується у бік старших або молодших розрядів. "Виштовхувані" з регістра розряди і утворюють послідовний код числа. Умовне графічне позначення паралельно-послідовного регістра приведене на рис. 3.60

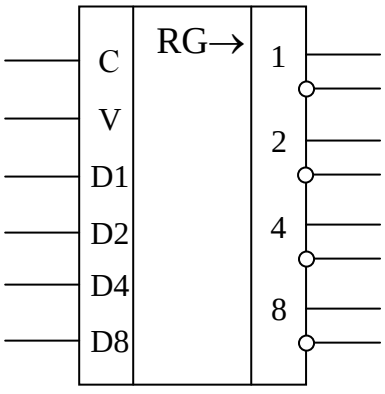


Рис. 3.60. Регістр зсуву

У пристроях обчислювальної техніки часто використовуються регістри, в яких є можливість зсовувати слова як вліво, так і вправо. Такі регістри називаються реверсивними. За способом управління зсувом коду, записаного в регістр, розрізняють регістри з однотактним і двотактним зсувом. Зробимо синтез однотактного регістра послідовної дії, в якому інформація зсовується вправо на один розряд з приходом кожного зсовуючого імпульсу. У зв’язку з

тим, що зв’язки між розрядами в такому регістрі однакові, досить побудувати схему двох розрядів регістра.

З огляду на те, що умови функціонування послідовного регістра визначені, можна відразу приступити до етапу структурного синтезу.

Виберемо в якості елементарного автомата ***RS***-тригер. Оскільки в дворозрядному регістрі можуть бути записані числа 00, 01, 10 і 11, тобто він має чотири стани, для синтезу регістра досить використати два тригери. В якості функціонально повного набору логічних елементів використовуємо набір І, АБО, НІ.

Наступний етап синтезу − кодування вхідних, вихідних сигналів і станів. Позначимо сигнал, що управляє зсувом (імпульс зсуву) буквою ***C***. Значення розряду послідовного коду, що приймається в регістр, буквою ***S***. Для кодування кожного з цих сигналів досить по одному двійковому розряду. Для кодування чотирьох станів регістра потрібно два двійкових розряди. Позначимо їх $\mathbf{Q_1}$ і $\mathbf{Q_0}$.

Вихідні сигнали регістра відповідають станам, тому відпадає необхідність в їх кодуванні.

Складемо кодовану таблицю переходів, виходів і сигналів збудження (табл. 3.55).

Таблиця 3.55

C	S	Q_1	Q_0	Q'_1	Q'_0	q_{RQ1}	q_{SQ1}	q_{RQ0}	q_{SQ0}
0	0	0	0	0	0	–	0	–	0
0	0	0	1	0	1	–	0	0	–
0	0	1	0	1	0	0	–	–	0
0	0	1	1	1	1	0	–	0	–
0	1	0	0	0	0	–	0	–	0
0	1	0	1	0	1	–	0	0	–
0	1	1	0	1	0	0	–	–	0
0	1	1	1	1	1	0	–	0	–
1	0	0	0	0	0	–	0	–	0
1	0	0	1	0	0	–	0	1	0
1	0	1	0	0	1	1	0	0	1
1	0	1	1	0	1	1	0	0	–
1	1	0	0	1	0	0	1	–	0
1	1	0	1	1	0	0	1	1	0
1	1	1	0	1	1	0	–	0	1
1	1	1	1	1	1	0	–	0	–

Мінімізуємо функції збудження за допомогою площинних діаграм.

CS	Q_1Q_0			
	00	01	11	10
00	–	–	0	0
01	–	–	0	0
11	0	0	0	0
10	–	–	1	1

$q_{RQ1} = C\overline{S}$;

CS	Q_1Q_0			
	00	01	11	10
00	0	0	–	–
01	0	0	–	–
11	1	1	–	–
10	0	0	0	0

$q_{SQ1} = CS$.

CS	Q_1Q_0			
	00	01	11	10
00	–	0	0	–
01	–	0	0	–
11	–	1	0	0
10	–	1	0	0

$q_{RQ0} = C\overline{Q}_1$;

CS	Q_1Q_0			
	00	01	11	10
00	0	–	–	0
01	0	–	–	0
11	0	0	–	1
10	0	0	–	–

$q_{SQ0} = CQ_1$.

Структурна схема двох розрядів регістра послідовної дії має вигляд (рис. 3.61).

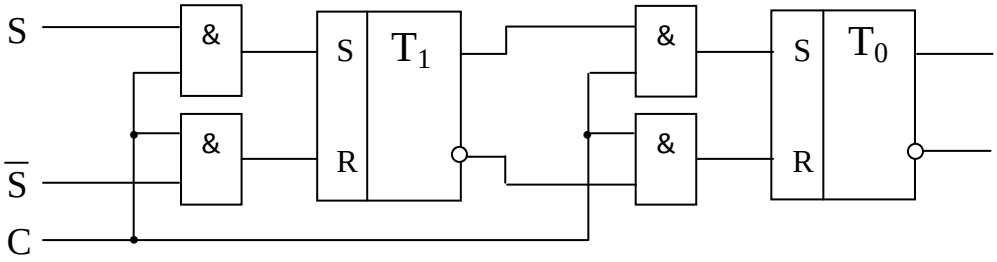


Рис. 3. 61. Схема регістра послідовної дії

Організуючи аналогічним чином міжрозрядні зв’язки, можна побудувати послідовний регістр на будь-яку кількість розрядів.

Часто потрібні складніші регістри: з паралельним синхронним записом інформації, реверсивні, з паралельно-послідовним синхронним записом. Такі регістри називаються *універсальними*.

Прикладом універсального регістра служить ІМС типу 155ИР1 (рис. 3.62).

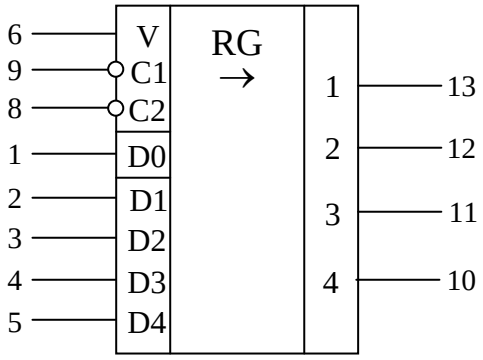


Рис. 3.62. Умовне графічне позначення регістра 155ИР1

Це чотирирозрядний зсовуючий регістр з можливістю послідовного і паралельного запису інформації. Його функціональна схема показана на рис. 3.63. Регістр виконаний на чотирьох ***RS***-тригерах і має два тактуючі входи **C1**, **C2** і один вхід **V**, який управляє режимом роботи регістра. Інформаційний вхід **D0** служить для занесення даних в послідовному коді, а входи **D1 ÷ D4** – для занесения даних в паралельному коді.

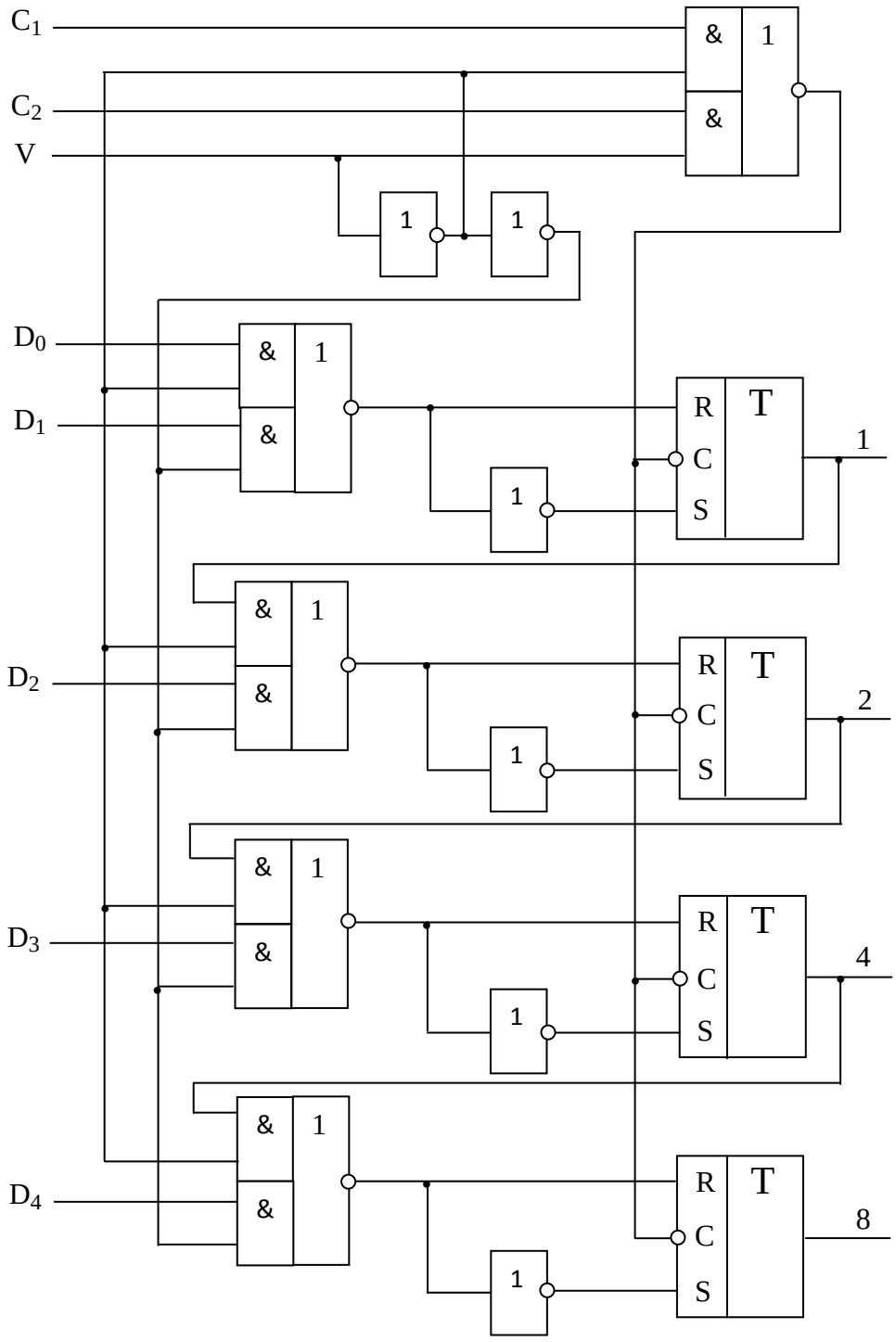


Рис. 3.63. Функціональна схема регістра 155ИР1

Регістр може працювати в чотирьох різних режимах, при яких виконуються: зсув кодів вправо, зсув кодів вліво, паралельне занесення даних, зберi-

гання інформації. Вибір того або іншого з них здійснюється поданням відповідного рівня логічного сигналу на управляючий вхід **V**. При **V = 0** робиться зсув кодів і зберігання інформації при зсуві вправо. Якщо **V = 1**, то відбувається або паралельне занесення інформації по входах **D1 ÷ D4**, або зсув кодів вліво.

При роботі регістра в режимі перетворення послідовного коду в паралельний із зсувом вправо (**V = 0**) відключаються входи **D1 ÷ D4** паралельного запису, дозволяється занесення даних в регістр по входу **D0** в послідовному коді і проходження тактуючих сигналів по входу **C1**, а також встановлюються зв'язки виходу кожного старшого розряду з входом подальшого молодшого. Зсув на один розряд вправо здійснюється при кожному спаді тактуючого імпульсу на вході **C1**. Інформація у вигляді чотирирозрядного паралельного коду з'явиться на виходах 1, 2, 4, 8 через чотири такти вхідного імпульсу (нумерація робиться від старших розрядів до молодших).

Паралельне занесення даних відбувається через входи **D1 ÷ D4** за наявності управляючого сигналу **V = 1** з приходом спаду імпульсу на вхід **C2.** При цьому вхід послідовного занесення **D0** і вхід тактуючих сигналів **C1** відключаються.

При організації зсуву вліво необхідно виконати з'єднання, показані на рис. 3.64.

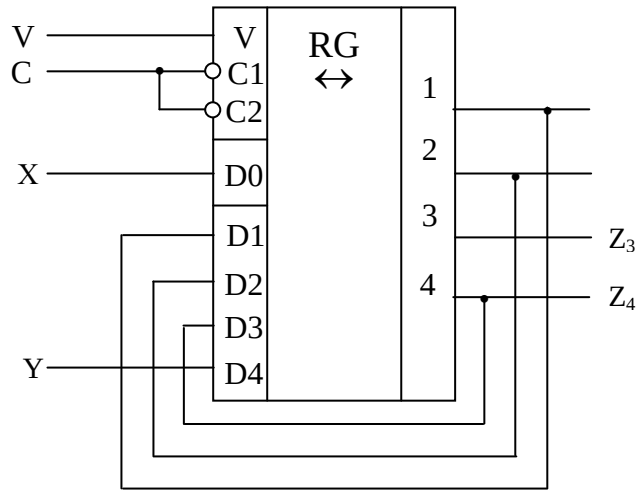


Рис. 3.64. Реверсивний зсовуючий регістр на ІМС типу 155ИР1

Послідовний запис в регістр здійснюється по входу **D4** при управляючому сигналі **V = 1**. Паралельний запис при зсуві кодів вліво немож-ливий, оскільки канали паралельного занесення використовуються для передачі даних від молодших розрядів до старших. Зсув кодів вправо здійснюється при кожному спаді тактуючого імпульсу **C2**. Помітимо, що у разі з'єднань, пока-заних нарис. 3.64, відсутня можливість лише паралельного занесення даних. Зсув кодів вправо можливий і, як і раніше, здійснюється поданням тактуючих сигналів на

вхід **С1** при низькому рівні логічного сигналу на управляючому вході **V**. Отже, зсовуючий регістр, зображений на рис. 3.64, є *реверсивним*.

При зсуві кодів вправо послідовні коди надходять по каналу **X** на вхід **D0** (**V = 0**), а при зсуві кодів вліво (**V = 1**) – по каналу **Y** на інформаційний вхід **D4**.

КОНТРОЛЬНІ ПИТАННЯ

1. Дайте визначення логічної функції.
2. Скільки різних логічних функцій має функція Y = f(A,B)?
3. Назвіть основні вимоги, яким повинні задовольняти системи елементів ЕОМ.
4. Поясніть сутність основних вимог, яким повинні задовольняти системи елементів ЕОМ.
5. Який логічний вузол називається тригером?
6. Наведіть таблицю станів для ***RS***-тригера.
7. Яким сигналом і в який стан встановлюється ***RS***-тригер по виходу за допомогою ***S***-входу?
8. Яким сигналом і в який стан встановлюється ***RS***-тригер по виходу за допомогою ***R***-входу?
9. Побудуйте таблицю кодованих переходів та виходів ***RS***- тригера.
10. Наведіть умовне графічне позначення і поясніть призначення виходів ***RS***-тригера.
11. Наведіть таблицю станів для ***T***-тригера.
12. Наведіть умовне графічне позначення ***T***-тригера.
13. Побудуйте таблицю кодованих переходів та виходів ***T***- тригера.
14. Наведіть умовне графічне позначення ***D***-тригера.
15. Наведіть таблицю станів для ***D***-тригера.
16. Наведіть умовне графічне позначення ***JK***-тригера.
17. Наведіть таблицю станів для ***JK***-тригера.
18. Яким сигналом і в який стан встановлюється ***JK***-тригер по виходу за допомогою ***J***- входу?
19. Яким сигналом і в який стан встановлюється ***JK***-тригер по виходу за допомогою ***K***- входу?
20. В який стан встановлюється ***JK***-тригер по виходу при подачі на входи ***J*** і ***K*** одиничних сигналів (1 1) ?

21. Чим відрізняється функціонування синхронного ***JK***-тригера від функціонування асинхронного ***JK***-тригера?
22. Реалізуйте лічильний ***T***-тригер на основі ***RS***-тригера.
23. Реалізуйте лічильний ***T***-тригер на основі ***D***-тригера.
24. Реалізуйте лічильний ***T***-тригер на основі ***JK***-тригера.
25. Яку функцію виконує двійковий шифратор?
26. Наведіть умовне графічне позначення шифратора.
27. Сформулюйте умови роботи шифратора за допомогою таблиці істинності.
28. Скільки виходів має повний класичний двійковий шифратор ***n***-розрядного вхідного коду ?
29. Який вузол називається дешифратором?
30. Наведіть умовне графічне позначення дешифратора.
31. Сформулюйте умови роботи дешифратора за допомогою таблиці істинності.
32. Скільки виходів має повний класичний двійковий дешифратор ***n***-розрядного вхідного коду ?
33. Запишіть вирази вихідних функцій пірамідального дешифратора на 4-и входи.
34. Запишіть вирази вихідних функцій двоступеневого дешифратора на 4-и входи.
35. Який вузол називається напівсуматором?
36. Наведіть умовне графічне позначення напівсуматора.
37. Запишіть умови функціонування однорозрядного напівсуматора (OHS) за допомогою таблиці істинності.
38. Який вузол називається суматором?
39. Наведіть умовне графічне позначення суматора.
40. Запишіть умови функціонування однорозрядного суматора за допомогою таблиці істинності.
41. Чим відрізняється функціонування однорозрядного напівсуматора від функціонування однорозрядного суматора?
42. Зарисуйте схему багаторозрядного комбінаційного суматора паралельної дії з послідовним переносом.
43. Які сигнали появляться на виході розрядної суми S_i і перенесення у наступний розряд P_i однорозрядного суматора при подачі на його інформаційні входи коду 11 і коду 0 на вхід перенесення з молодшого розряду ?
44. Які сигнали появляться на виході розрядної суми S_i і перенесення у наступний розряд P_i однорозрядного суматора при подачі на його інформаційні входи коду 01 і коду 1 на вхід перенесення з молодшого розряду ?

45. Чим відрізняється функціонування накопичуючого суматора від функціонування комбінаційного суматора?
46. Яку функцію виконують різні типи ланцюгів переносу в паралельних суматорах?
47. Яку функцію виконує двійковий лічильник?
48. Наведіть умовне графічне позначення двійкового лічильника.
49. Чим відрізняється підсумовуючий лічильник від віднімаючого?
50. Які функції виконує двійковий реверсивний лічильник?
51. Скільки тригерів потрібно використати для побудови 8-ми розрядного реверсивного лічильника ?
52. Як побудувати лічильник з довільним модулем рахунку?
53. Скільки тригерів потрібно використати для побудови лічильника з довільним модулем рахунку, якщо $M_{лч} = 7$?
54. Скільки тригерів потрібно використати для побудови лічильника з довільним модулем рахунку, якщо $M_{лч} = 12$?
55. Сформулюйте правило побудови перерахункових схем з однорідними зв’язками.
56. Сформулюйте правило побудови перерахункових схем з комбінованими зв’язками.
57. Який вузол називається регістром?
58. Наведіть умовне графічне позначення двотактного регістра.
59. Чим відрізняється регістр паралельної дії від регістра послідовної дії?
60. Скільки тригерів потрібно використати для побудови 8-ми розрядного паралельного регістра ?
61. Скільки тригерів потрібно використати для побудови 8-ми розрядного регістра зсуву (послідовного регістра)?
62. Які арифметичні операції можна виконувати за допомогою регістра?
63. Яку арифметичну операцію виконує регістр зсуву вправо?
64. Яку арифметичну операцію виконує регістр зсуву вліво?
65. Що розуміється під поняттям універсальний регістр?

СПИСОК ЛІТЕРАТУРИ

1. Вычислительные системы, сети и телекоммуникации / В. Л. Бройдо. – СПб.: Питер, 2004. – 703 с.
2. Жабін В.І. та ін. Прикладна теорія цифрових автоматів. – 2-ге вид., доопрац. – К.: НАУ, 2009. -360 с.
3. Кобайло А.С. Логические основы цифровых вычислительных машин: учеб.-метод. пособие. – Минск: БГТУ, 2010. – 90 с.
4. Ожиганов А.А. Теория автоматов. Учебное пособие. – СПб: НИУ ИТМО, 2013. – 84 с.
5. Поворознюк А.И. Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств: учеб. пособие. Ч.1. – Харьков: Торнадо, 2004. – 355 с.
6. Рябенький В.М., Жуйков В.Я., Гулий В.Д. Цифрова схемотехніка: навч. посібник. – Львів: «Новий світ – 2000», 2009. – 736 с.
7. Савельев А. Я. Прикладная теория цифрових автоматов. – М.: Высшая школа, 1987. – 272 с.
8. Схемотехника електронних систем: в 3 кн. Кн.3. Микропроцессоры и микроконтроллеры: учебник /В.И. Бойко, А.М. Гуржий, В.Я. Жуйков и др. – СПб.:БХВ - Петербург, 2004. – 455 с.
9. Тарарака В.Д. Архітектура комп’ютерних систем. Навчальний посібник. Ел. вид., перероб. і доп. – Житомир: ЖДТУ, 2018. – 384 с.
10. Тарарака В.Д. Основы вычислительной техники и программирование. Ч.2. Основы вычислительной техники: учебное пособие. – Житомир: ЖВУРЭ ПВО, 1992. – 500 с.
11. Тарарака В.Д. Обчислювальна техніка. Ч.І. Основи побудови ЕОМ: навчальний посібник. – Житомир: ЖВІРЕ, 2003. – 348 с.
12. Тарарака В.Д. Обчислювальна техніка. Ч. ІІ. Апаратні засоби персональних комп'ютерів: навчальний посібник. - Житомир ЖВІРЕ, 2004. – 308 с.
13. Торба А.А. Аналоговая и цифровая схемотехніка компьютеров: учеб. пособие / А.А. Торба, А.А. Бобкова. – Х., 2012. – 444 с.

14. Щербаков А.Н. та ін. Прикладна теорія цифрових автоматів.Ч.1. – Комп’ютерна арифметика. Конспект лекцій. – Запоріжжя: ЗНТУ, 2010. – 102 с.
15. Яцків В.В. Прикладна теорія цифрових автоматів. Методичні вказівки до лабораторних робіт. – Тернопіль: Економічна думка, 2005. – 60 с.

Навчальне видання

Тараракa Валерій Дмитрович

ПРИКЛАДНА ТЕОРІЯ ЦИФРОВИХ АВТОМАТІВ

Навчальний посібник

Електронне видання

Відповідальний редактор **Ю.А. Подчашинський**
Комп’ютерний набір та верстка **В.Д. Тараракa**
Макетування **В.Д. Тараракa**

Гарнітура Times New Roman