Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Функціонально-логічне проектування
Комбінаційні пристрої

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів, які навчаються за
спеціальністю 172 «Радіотехніка та телекомунікації»,
спеціалізацією «Інформаційно-обчислювальні засоби електронних систем»

Київ – 2017

УДК 004.312.2

Функціонально-логічне проектування: Комбінаційні пристрої [Електронний ресурс]: навч. посіб. для студ. спеціальності 172 «Телекомунікації та радіотехніка», спеціалізації «Інформаційно-обчислювальні засоби електронних систем» / КПІ ім. Ігоря Сікорського ; уклад.: А.Ю.Варфоломєєв. – Електронні текстові данні (1 файл). – К. : КПІ ім. Ігоря Сікорського, 2017 р. – 135 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 3 від 23.11.2017 р.) за поданням вченої ради факультету електроніки (протокол № 10/2017 від 30.10.2017 р.)

Електронне навчальне видання

Функціонально-логічне проектування. Комбінаційні пристрої

Укладач: *Варфоломєєв Антон Юрійович*, к.т.н., ст. викладач

Відповідальний редактор: *Лисенко О.М.*, д.т.н., проф., завідувач кафедри КЕОА ФЕЛ КПІ ім. Ігоря Сікорського

Рецензенти: *Віноградов М.А.*, д.т.н., проф. кафедри комп'ютерних інформаційних технологій НАУ

Петренко С.Ф., д.т.н., директор ТОВ «Лілея»

Наведено теоретичні основи проектування цифрових комбінаційних пристроїв. Посібник містить відомості про системи числення, включаючи дробове представлення двійкових чисел та формат чисел з плаваючою точкою IEEE-754, основи двійкової арифметики, алгебри логіки, теорії логічних функцій, способів їх представлення, аналітичного розкладу та основних методів мінімізації. На основі викладеного теоретичного матеріалу наведено приклади синтезу найпоширеніших комбінаційних логічних пристроїв, що застосовуються в сучасній електронно-обчислювальній апаратурі.

Посібник призначено для студентів спеціальності «Телекомунікації та радіотехніка», спеціалізації «Інформаційно-обчислювальні засоби електронних систем» проте він може бути корисний також і для інженерно-технічних спеціалістів та аспірантів, що працюють в галузі цифрової електроніки.

Зміст

Вступ **5**
1. Системи числення **7**
1.1. Переведення чисел з однієї позиційної системи числення в іншу ... 13
1.1.1. Переведення цілих чисел 13
1.1.2. Переведення правильних дробів 16
1.1.3. Переведення неправильних дробів 19
1.2. Вибір оптимальної системи числення для застосування в ЕОМ 19
Питання та завдання для самоперевірки і контролю засвоєння знань 21
2. Двійкова арифметика **23**
2.1. Додавання цілих двійкових чисел. 23
2.2. Віднімання цілих двійкових чисел. 24
2.3. Від'ємні числа та доповнювальний код 24
2.4. Множення двійкових чисел 29
2.5. Ділення двійкових чисел 32
2.6. Числа з фіксованою точкою 34
2.7. Числа з плаваючою точкою (IEEE-754/IEC 60559) 38
Питання та завдання для самоперевірки і контролю засвоєння знань 43
3. Логічні функції та логічні елементи **45**
3.1. Логічні функції 45
3.2. Логічні елементи 48
Питання та завдання для самоперевірки і контролю засвоєння знань 54
4. Аксіоми, теореми та тотожності алгебри логіки **56**
4.1. Аксіоми алгебри логіки 56
4.2. Теореми і тотожності алгебри логіки 57
4.3. Властивості операції «сума по модулю 2» (виключне АБО) 59
4.4. Пріоритетність операцій 60
Питання та завдання для самоперевірки і контролю засвоєння знань 61
5. Теореми розкладу перемикальних функцій **62**
5.1. Теорема розкладу Шеннона 62
5.2. Мультиплексні функції. Мультплексор 63
5.3. Розклад Рида 65
Питання та завдання для самоперевірки і контролю засвоєння знань 66
6. Первинні терми. Мінтерми і макстерми **67**
6.1. Первинні (елементарні) терми 67
6.2. Мінтерми 68
6.3. Макстерми 69
6.4. Дешифратор 70
Питання та завдання для самоперевірки і контролю засвоєння знань 71

7. Нормальні канонічні форми двійкових функцій 72
7.1. Досконала диз'юнктивна нормальна форма (ДДНФ) 72
7.2. Досконала кон'юнктивна нормальна форма (ДКНФ) 73
7.3. Досконалі нормальні форми в базисах І-НІ та АБО-НІ 74
7.4. Розклад Рида-Маллера 76
Питання та завдання для самоперевірки і контролю засвоєння знань 77
8. Системи логічних рівнянь 78
8.1. Системи логічних рівнянь з одним невідомим 78
8.2. Арифметичне представлення логічних рівнянь 83
8.3. Системи логічних рівнянь з більш ніж одним невідомим 84
Питання та завдання для самоперевірки і контролю засвоєння знань 87
9. Мінімізація логічних функцій 88
9.1. Метод Квайна 88
9.2. Мінімізація логічних функцій методом Квайна-Мак-Класкі 92
9.3. Мінімізація логічних функцій методом карт Карно-Вейча 94
9.4. Мінімізація неповністю визначених функцій 101
9.5. Спільна мінімізація декількох функцій 103
9.6. Факторизовані форми функції 105
9.7. Синтез комбінаційних схем на мажоритарних елементах 106
Питання та завдання для самоперевірки і контролю засвоєння знань 109
10. Схеми деяких найбільш поширених комбінаційних пристроїв 112
10.1. Демультиплексор 112
10.2. Шифратори 113
10.2.1. Простий шифратор 113
10.2.2. Пріоритетний шифратор 115
10.3. Схеми комбінаційного зсуву 118
10.4. Комбінаційні суматори 121
10.4.1. Однорозрядний суматор 121
10.4.2. Суматор з послідовним перенесенням 123
10.4.3. Суматор з паралельним перенесенням 124
10.5. Схема віднімання 126
10.6. Універсальна схема додавання та віднімання 127
10.7. Схема логічного множення 127
10.8. Схема логічного ділення 129
Питання та завдання для самоперевірки і контролю засвоєння знань 131
Перелік скорочень 132
Предметний покажчик 133
Література 135

Вступ

В основі всієї сучасної комп'ютерної техніки та електронно-обчислювальної апаратури лежать різноманітні цифрові пристрої. Згідно прийнятої на сьогодні методології, розробка даних пристроїв ведеться на базі програмованих логічних інтегральних схем (ПЛІС). Процедура створення програмного забезпечення для ПЛІС, передбачає використання спеціалізованих мов опису апаратури (HDL). Не зважаючи на те, що дані мови значно спрощують процес розробки, вони все ж не знімають з розробника необхідності розуміння основних законів функціонування та синтезу цифрових пристроїв. Окрім цього, в HDL-мовах разом із спрощенням опису апаратури, на жаль, втрачається і контроль над конкретною її реалізацією. Як приклад можна навести додавання чисел на мові Verilog: неможливо явно вказати, який саме тип суматора має використовуватись в даному випадку. Викладене вище, поза сумнівом, підтверджує той факт, що навіть на сьогодні інженери і конструктори електронної апаратури повинні володіти ґрунтовними знаннями щодо принципів функціонування, методів проектування та синтезу цифрових пристроїв. Саме цьому питанню і присвячено дане видання.

Основною метою посібника, як і дисципліни «Функціонально-логічне проектування» в рамках якої він застосовується, є формування у студентів, які навчаються за спеціальностю 172 «Телекомунікації та радіотехніка» здатностей реалізовувати прості перемикальні функції, здійснювати їх мінімізацію, проводити аналіз та синтез основних комбінаційних логічних схем. Посібник також має ознайомити студентів з готовими схемами найбільш поширених цифрових пристроїв.

Дане видання охоплює перший розділ дисципліни «Функціонально-логічне проектування», який присвячено розробці комбінаційних цифрових пристроїв. Матеріали посібника послідовно знайомлять читача з системами числення, включаючи дробове представлення двійкових чисел та формат чисел з плаваючою точкою IEEE-754, двійковою арифметикою, алгеброю логіки, з теорією логічних функцій, способами їх представлення, аналітичного розкладу та основними методами мінімізації. На основі викладеного теоретичного матеріалу, наприкінці посібника, наведено приклади синтезу основних комбінаційних логічних пристроїв, що застосовуються в сучасній електронно-обчислювальній апаратурі, зокрема: мультиплексорів і демультиплексорів, дешифраторів і

шифраторів, схем комбінаційного зсуву, суматорів, пристроїв віднімання множення та ділення двійкових чисел.

При цьому головною особливістю посібника є деталізоване викладення теоретичних основ систем числення, двійкової арифметики та перемикальних функцій, оскільки саме цього матеріалу наразі бракує більшості підручників, які видаються в останні роки. Таким чином, за основу даного видання було вирішено взяти перевірені часом класичні літературні джерела [1, 2]. Разом із тим, для забезпечення відповідності сучасному стану справ у цифровій електроніці до посібника включено й матеріали з сучасної літератури [3-5].

Слід відзначити, що для успішного застосування даного видання, читачу бажано, але не обов’язково володіти знаннями з вищої математики та інформатики. У свою чергу засвоєння матеріалу посібника буде важливим і доцільним при вивченні таких дисциплін, як: електронні пристрої, обчислювальні та мікропроцесорні засоби, проектування «систем на кристалі».

Навчальний посібник призначений головним чином для студентів радіоелектронних і телекомунікаційних спеціальностей, в той же час він може бути корисним також для інженерно-технічних спеціалістів та аспірантів, що працюють в галузі цифрової електроніки.

1. СИСТЕМИ ЧИСЛЕННЯ

Системою числення називається сукупність підходів запису чисел. В загальному випадку її можна вважати спеціальною мовою, алфавітом якої є символи, які називають *цифрами*, а синтаксисом – правила, що дозволяють однозначним чином сформувати запис чисел [2]. Запис числа в деякій системі числення називають *кодом* числа.

Формалізувати запис числа можна наступним чином:

$$A = a_n a_{n-1} \ldots a_2 a_1 a_0$$

Окрему позицію в зображенні числа прийнято називати *розрядом*, а номер позиції – *номером розряду*. Кількість розрядів у записі числа називається *розрядністю* і співпадає з його довжиною. Якщо алфавіт має p різних значень, то розряд a_i у числі може інтерпретуватись як p-кова (десяткова, двійкова тощо) цифра і відповідно приймати p значень.

Кожній цифрі a_i даного числа A однозначно відповідає її кількісний еквівалент – $K(a_i)$. Кількісний еквівалент числа A, заданого в певній системі числення, є деякою функцією еквівалентів всіх його цифр, тобто:

$$K(A) = f[K(a_n),\ldots, K(a_1)].$$

Діапазон представлення чисел у даній системі числення:

$$D = K(A)_{\max} - K(A)_{\min},$$

де $K(A)_{\min}$ та $K(A)_{\max}$ – мінімальне та максимальне число, представлене заданою розрядністю.

Класифікація систем числення наведена на рис. 1.1.

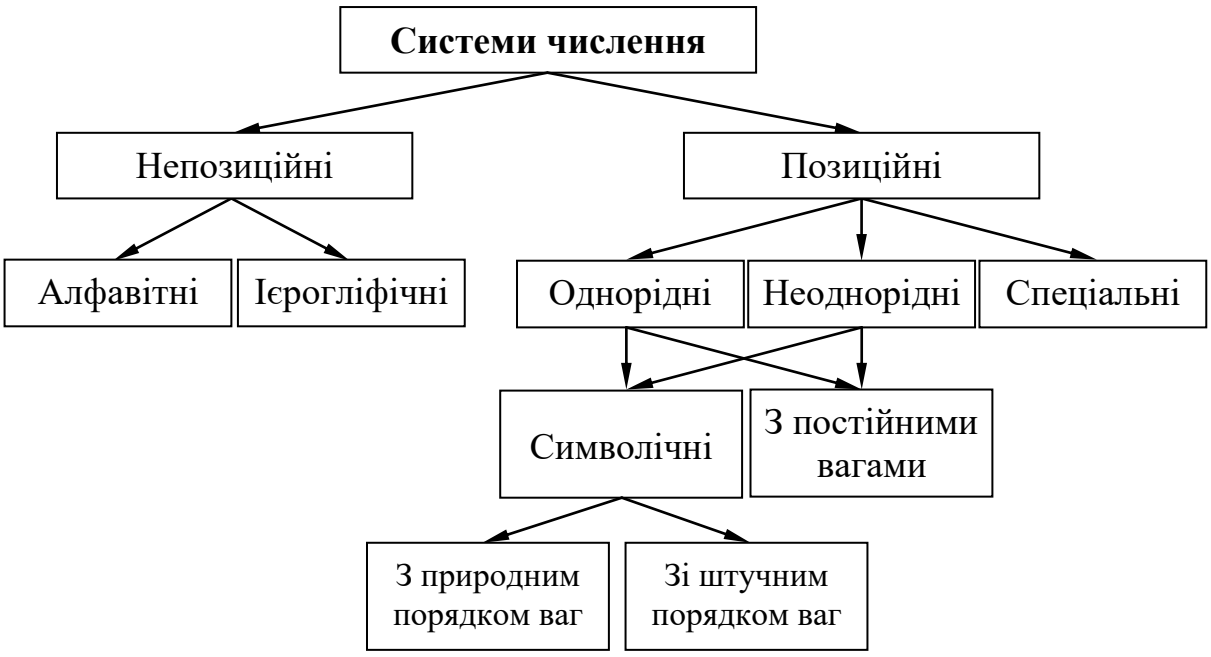


Рисунок 1.1 – Класифікація систем числення

База системи числення – послідовність цифр системи числення з p_i-ковим алфавітом. База системи числення може бути додатною або змішаною (містити цифри, що мають як додатну, так і від'ємну вагу).

Базис системи числення – сукупність всіх ваг окремих розрядів системи числення. Наприклад, базис десяткової системи є послідовністю: 1, 10, 10^2,…,10^n.

Більшість систем числення будуються за таким принципом, що їх кількісний еквівалент можна знайти за узагальненою формулою:

$$K(A_P) = a_0 \cdot p_0 + a_1 \cdot p_1 + a_2 \cdot p_2 + \ldots + a_{n-1} \cdot p_{n-1} + a_n \cdot p_n,$$

де A_P – запис числа в системі з базисом $P = \{p_i\}$; a_i – база.

Вага i-ого розряду w_i визначається відношення:

$$w_i = \frac{p_i}{p_0}.$$

Непозиційні системи числення. Непозиційними називаються такі системи числення, алфавіт яких має необмежену кількість символів (цифр), причому кількісний еквівалент будь-якої цифри є постійним і залежить тільки від її нарису, але не від положення в числі. Такі системи будуються за принципом адитивності, тобто кількісний еквівалент числа в них визначається як сума цифр, що знаходяться поруч:

$$K(A(Q)) = Q_1 + Q_2 + \ldots + Q_k = \sum_{i=1}^{k} Q_i,$$

де Q_i – символи, що утворюють базис системи $\{Q_1, Q_2, \ldots, Q_k\}$.

Непозиційні є найбільш ранніми системи числення. Знахідки археологів свідчать, що первісні люди позначали кількість предметів рівною кількістю певних знаків – зарубок, рисок, точок тощо, які часто групували по декілька штук (зазвичай 3 або 5) для полегшення підрахунків. Така система запису чисел називається *одиничною* або *унарною*, оскільки будь-яке число у ній утворюється шляхом повторення одного знаку, який символізує одиницю. Одинична система, очевидно, не є зручним способом представлення чисел, оскільки великі числа в ній виглядатимуть надто громіздко. Тим не менш унарна система числення досить часто застосовується в електроніці, зокрема, для представлення чисел кількістю імпульсів, що подаються на вхід цифрових пристроїв (наприклад $E = 11111_1 = 5_{10}$, де символ 1 позначає один імпульс). Прикладами таких цифрових пристроїв можуть бути різноманітні лічильники, частотоміри тощо.

Приблизно в третьому тисячолітті до н.е. давні єгиптяни придумали свою систему числення. Вона була десятковою і в ній для позначення ключових чисел 1, 10, 100 використовувались спеціальні ієрогліфи, зокрема: для тисяч – квітки лотоса, для сотень – пальмові листки, для десятків – дуги, для одиниць – шести. Так, для запису числа 3252 необхідно було намалювати три квітки лотоса, два пальмові листки, п’ять дуг і два шести. При цьому величина числа не залежала від того, в якому порядку записано його складові: їх можна було записати і зверху вниз, і справа наліво, і взагалі перемішати.

Одним із прикладів древніх непозиційних систем числення, які і досі використовуються є *римська*. В її основі лежати знаки I – один палець, що позначає 1, V – розкрита долоня – позначає 5, X – дві розкриті долоні, які позначають 10. Для позначення чисел 50, 100, 500 та 1000 застосовуються літери L, C, D та M відповідно (є припущення, що деякі з них є першими літерами латинських назв відповідних чисел: Centum – сто, Dimidium mille – півтисячі, Mille – тисяча). Щоб записати число в римській системі його необхідно розкласти на суму тисяч, півтисяч, сотень, півсотень, десятків, п’ятірок та одиниць. Зокрема, десяткове число 37 в римській системі матиме вигляд: XXXVII = 10 + 10 + 10 + 5 + 1 + 1. Для запису проміжних значень в римській системі використовується не тільки додавання, але й віднімання, для чого застосовуються наступне правило: кожен менший символ поставленій справа від символа з більшою вагою додається до його значення, а поставлений зліва – віднімається. Наприклад: IX = 10 − 1 = 9, а XI = 10 + 1 = 11, а десяткове число 99 записуватиметься так: XCIX = 100 − 10 + 10 − 1.

Більш досконалими непозиційними системами числення були алфавітні системи. До них можна віднести фінікійську, грецьку, слов’янську та інші. В цих системах числа від 1 до 9, цілі кількості десятків (від 10 до 90) і цілі кількості сотень (від 100 до 900) позначалися літерами алфавіту. Зокрема, в алфавітній системі числення Древньої Греції числа 1, 2,…, 9 позначались першими дев’ятьма літерами грецької абетки (α, β,…, ι), для позначення десятків 10, 20,…, 90 використовувались наступні дев’ять літер і так далі. У слов’янських народів числові значення букв встановлювались у порядку слов’янської абетки – спочатку глаголиці, а потім – кирилиці.

До сучасних непозиційних систем числення можна віднести біноміальну систему, що відповідно використовує біноміальні коефіцієнти, систему залишкових класів, основану на китайській теоремі про залишки.

Непозиційні системи мають ряд недоліків:

1) для представлення чисел довільної величини необхідно мати довільну кількість символів;
2) з ними складно виконувати арифметичні операції;
3) в них відсутній нуль.

Позиційні системи числення. Позиційними називаються системи числення, алфавіт яких містить обмежену кількість символів, причому значення кожного розряду числа визначається не тільки нарисом відповідної йому цифри, але й позицією в самому числі. До позиційних систем числення належить арабська десяткова система, якою ми зараз користуємось.

В загальному випадку в позиційній системі кількісний еквівалент визначається як:

$$K(A) = a_n p_n + a_{n-1} p_{n-1} + \ldots + a_2 p_2 + a_1 p_1 + a_0 p_0 = \sum_{i=0}^{n} a_i p_i , \quad (1.1)$$

де p_i – основи системи числення, причому $p_{i+1} > p_i$; a_i – цифра i-ого числа.

Вважається, що перша позиційна система числення була винайдена древніми шумерами та вавилонянами близько 3 тисяч років до н.е. Вона мала основу 60 і досі використовується для визначення мір кутів (1° = 60′, 1′ = 60″, тобто при вираженні кута в секундах $p_1 = 60$, $p_2 = 3600$).

Позиційні системи числення діляться на ряд підкласів.

Неоднорідні позиційні системи числення. В таких системах числення основи p_i – не залежать одна від одної і можуть приймати будь-які значення, тому їх також називають системами зі змішаними основами. Пошук кількісного еквівалента числа в даній системі числення здійснюється за формулою (1.1). Зауважимо, що в деяких неоднорідних системах кількість допустимих символів у різних розрядах може відрізнятись.

Прикладами неоднорідних позиційних систем числення є система числення часу, для якої $p_0 = 1$ (с), $p_1 = 60$ (с) [1 хв.], $p_2 = 3600$ (с) [1 год.], $p_3 = 24 \cdot 3600$ (с) [1 доба].

Іншим прикладом неоднорідної позиційної системи, що широко застосовується в електронній техніці є так званий *унітарний код*. Він містить символ «1» тільки в *одній* єдиній позиції n-розрядного числа – в тій позиції (розряді), номер якого дорівнює значенню даного числа.

Наприклад:

0_{10} = 000000001
1_{10} = 000000010
2_{10} = 000000100
…
7_{10} = 010000000
8_{10} = 100000000

Унітарний код формується на виходах повних дешифраторів.

Також до прикладів неоднорідних позиційних систем числення можна віднести двійково-десяткову систему з основою 5−4−2−1, яка застосовуються в декадних лічильниках, та спеціальну п'ятірково-двійкову систему з основою 5 – у непарних розрядах та 2 – у парних розрядах. Наразі такі системи майже не використовуються.

Однорідні позиційні системи числення. Однорідні позиційні системи числення є окремим випадком позиційних систем числення з основою $p_i = p_1^i$. В них ваги окремих розрядів є елементами геометричної прогресії зі знаменником $p = p_1$. Таким чином, кількісний еквівалент цілого числа може бути представлений поліномом [2]:

$$K(A) = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_2 p^2 + a_1 p^1 + a_0 p^0 = \sum_{i=0}^{n} a_i p^i . \qquad (1.2)$$

До однорідних позиційних систем числення належить сучасна десяткова система, якою ми користуємось, а також усі інші системи, які наразі широко застосовуються в обчислювальній техніці, зокрема двійкова, вісімкова та шістнадцяткова. Наведемо таблицю перших 17 чисел в цих системах починаючи з 0.

Таблиця 1.1 – Представлення чисел в основних системах числення

Десяткова	**Двійкова**	**Вісімкова**	**Шістнадцяткова**
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Враховуючи, що числа можуть бути дробовими, у однорідній позиційній системі їх запис можна узагальнити наступним чином:

$$A = \underbrace{a_n}_{\text{найбільш значущий розряд}} a_{n-1} \ldots a_2 a_1 a_0 . a_{-1} a_{-2} \ldots \underbrace{a_{-k}}_{\text{найменш значущий розряд}}$$

а кількісний еквівалент даного числа визначатиметься за формулою [2]:

$$K(A) = \underbrace{\sum_{i=0}^{n} a_i p^i}_{\substack{\text{ціла} \\ \text{частина}}} + \underbrace{\sum_{j=1}^{k} a_{-j} p^{-j}}_{\substack{\text{дробова} \\ \text{частина}}} = \sum_{i=-k}^{n} a_i p^i . \qquad (1.3)$$

Системи числення з непостійними вагами розрядів. В електронно-обчислювальній техніці також застосовуються системи числення з непостійними вагами розрядів. Найбільш поширеною серед них є так званий код Грея. Код Грея – це двійковий код, особливість якого полягає у тому, що будь-які два послідовно записані числа в ньому завжди відрізняється *тільки одним розрядом*. Такий спосіб представлення в деяких випадках дозволяє позбавитись помилок, зокрема, при реалізації електромеханічних датчиків (наприклад, датчиків кута повороту).

Для переходу з двійкового до коду Грея і назад використовується наступні правила:

1) із двійкового в код Грея:

$$G_i = B_i \oplus B_{i+1},$$

2) із коду Грея в двійковий:

$$B_i = \bigoplus_{j=i}^{n} G_j ,$$

де B_i – i-й розряд двійкового коду; G_i – i-й розряд коду Грея; якщо $i + 1$ розряд виходить за межі розрядної сітки, вважається, що він дорівнює 0; $a \oplus b$ – сума по модулю 2 («Виключне АБО»), яка дорівнює:

$$a \oplus b = \begin{cases} 1, & a \neq b \\ 0, & a = b \end{cases}$$

У табл. 1.2 представлено 4-бітний код Грея.

Таблиця 1.2 – Приклад 4-розрядного коду Грея

Десяткова	**Двійкова**	**Код Грея**	**Десяткова**	**Двійкова**	**Код Грея**
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

1.1. Переведення чисел з однієї позиційної системи числення в іншу

1.1.1. Переведення цілих чисел

Нехай задано число A в довільній позиційній системі числення і його необхідно перевести у нову систему з основою p, тобто привести до вигляду, коли кількісний еквівалент буде мати вигляд [2]:

$$K(A) = a_n p^n + a_{n-1} p^{n-1} + \ldots + a_1 p^1 + a_0 p^0, \qquad (1.4)$$

де $a_i = \overline{0, p-1}$ – база нової системи числення та її основа p представлені в старій системі числення.

Записаний вище вираз можна переписати наступним чином:

$$A = A_1 p + a_0,$$

де $K(A_1) = a_n p^{n-1} + a_{n-1} p^{n-2} + \ldots + a_2 p^1 + a_1 p^0$ – ціла частина частки; a_0 – остача від ділення A на p, що, очевидно, є цифрою молодшого розряду шуканого числа, записаного в символах старої системи числення.

Тепер, якщо переписати аналогічним чином A_1, отримаємо:

$$A_1 = A_2 p + a_1,$$

де $K(A_2) = a_n p^{n-2} + a_{n-1} p^{n-3} + \ldots + a_3 p^1 + a_2 p^0$. І так далі…, з чого випливає, що вираз (1.4) можна представити, застосовуючи схему Горнера:

$$K(A) = ((a_n p + a_{n-1})p + a_{n-2})p + \ldots + a_1)p + a_0.$$

Отже, в результаті серії ділень заданого цілого числа на основу нової системи числення p знаходимо коефіцієнти A_i та остачі від ділення a_i:

$$\begin{aligned}
A &= A_1 p + a_0;\\
A_1 &= A_2 p + a_1;\\
&\ldots\\
A_i &= A_{i+1} p + a_i;\\
&\ldots\\
A_{n-1} &= A_n p + a_{n-1};\\
A_n &= 0 \cdot p + a_n.
\end{aligned}$$

Ділення продовжується до тих пір, доки виконуються наступні умови:

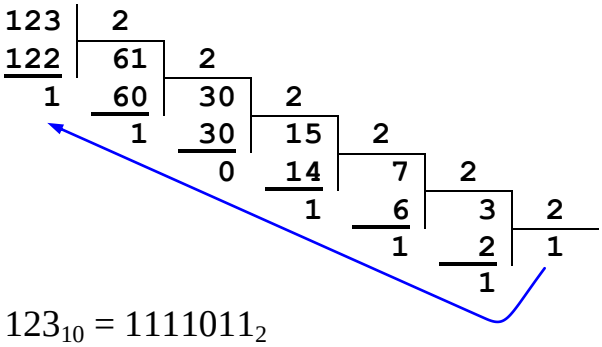
$$\begin{cases} A_n > p \\ A_{n+1} \neq 0 \end{cases}$$

Правило переводу цілих чисел із однієї позиційної системи числення у іншу можна сформулювати наступним чином: задане число необхідно поділити на основу нової системи числення та знайти частку й остачу від ділення, після цього повторно застосовувати дану процедуру до кожної нової отриманої частки доти, доки не буде отримано частку рівну нулю чи меншу за основу нової системи числення. Число в новій системі числення,

після цього, записується із знайдених остач, починаючи з останньої знайденої остачі. Основа нової системи числення при виконанні наведеної вище процедури повинна бути представлена числом старої системи числення.

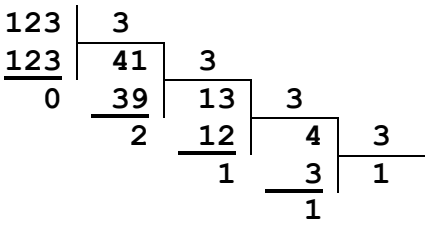
Приклади.

1) Перевести число 123_{10} до двійкової системи числення:



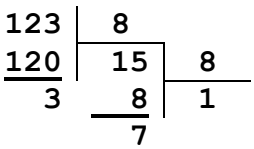
$123_{10} = 1111011_2$

2) Перевести число 123_{10} до трійкової системи числення:



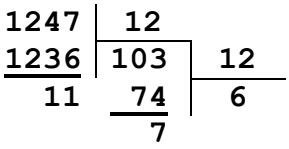
$123_{10} = 11120_3$

3) Перевести число 123_{10} до вісімкової системи числення:



$123_{10} = 173_8$

4) Перевести число 1247_8 до десяткової системи числення:



Враховуючи, що під час ділення всі дії і результати отримані у вісімковій системі, для запису числа в десятковій системі всі залишки також треба перевести в десяткову систему. Це зокрема стосується першого залишку 11_8 який в десятковій системі рівний 9_{10}. Отже остаточний результат буде:

$1247_8 = 679_{10}$

Зауважимо, що переведення чисел також можна виконувати безпосередньо за допомогою формули (1.4). Для цього кожну цифру та основу в початковій системі числення необхідно перевести у числа нової системи числення та знайти значення виразу (1.4), виконуючи арифметичні дії за правилами нової системи числення.

Приклади.

1) Для переведення числа $25B_{16}$ до десяткової системи маємо:
   - цифри: $a_2 = 2_{16} = 2_{10}$, $a_1 = 5_{16} = 5_{10}$, $a_0 = B_{16} = 11_{10}$
   - основа: $p = 10_{16} = 16_{10}$.

   Тоді:

   $25B_{16} = 2_{10} \cdot 16_{10} \cdot 16_{10} + 5_{10} \cdot 16_{10} + 11_{10} = 512_{10} + 80_{10} + 11_{10} = 603_{10}$
2) Для переведення числа 158_{10} до вісімкової системи маємо:
   - цифри: $a_2 = 1_{10} = 1_8$, $a_1 = 5_{10} = 5_8$, $a_0 = 8_{10} = 10_8$
   - основа: $p = 10_{10} = 12_8$.

   Тоді:

   $158_{10} = 1_8 \cdot 12_8 \cdot 12_8 + 5_8 \cdot 12_8 + 10_8 = 144_8 + 62_8 + 10_8 = 236_8$

Наведені способи переводу чисел будуть найбільш зручними для при переході з/до *десяткової* системи числення, оскільки в цьому випадку розрахунки в формулах виконуватимуться саме в десятковій системі, що для нас є більш звичним. При цьому для переходу з *десяткової в інші системи* краще використовувати перший спосіб, оснований на *схемі Горнера та діленні*, а для переходу з інших систем *до десяткової* – другий спосіб, оснований на *безпосередньому застосуванні виразу* (1.4).

Для швидкого переходу між двійковою, вісімковою та шістнадцятковою системами використовується ще один спосіб. Запишемо формулу (1.4) з основою $p = 2$, групуючи розряди наступним чином:

$$K(A) = (a_n 2^2 + a_{n-1} 2^1 + a_{n-2} 2^0) \cdot 2^{n-2} + \ldots + \\ + (a_5 2^2 + a_4 2^1 + a_3 2^0) \cdot 2^3 + (a_2 2^2 + a_1 2^1 + a_0 2^0) \cdot 2^0$$

та

$$K(A) = (a_n 2^3 + a_{n-1} 2^2 + a_{n-2} 2^1 + a_{n-3} 2^0) \cdot 2^{n-3} + \ldots + \\ + (a_7 2^3 + a_6 2^2 + a_5 2^1 + a_4 2^0) \cdot 2^4 + (a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0) \cdot 2^0$$

З наведених виразів видно, що суми в дужках є кількісними еквівалентами вісімкових та шістнадцяткових цифр, так як в першому випадку кожна дужка може представляти значення 0…7, а в другому – 0…15. Легко також переконатися, що множники винесені за дужки для першого виразу кратні степеню числа 8, а для другого – числа 16, тобто відповідають основам вісімкової та шістнадцяткової систем числення. Це означає, що якщо в двійковому числі, починаючи з найменш значущого розряду (справа наліво) утворювати групи по *три* (тріади) або по *чотири* (тетради) розряди, можна одразу записувати відповідні цим групам цифри вісімкового та шістнадцяткового чисел. Аналогічно, якщо число

представлене у вісімковій чи шістнадцятковій системах, то переписуючи кожен його розряд відповідними 3- або 4-розрядними двійковими числами, можна одразу отримати представлення цього числа у двійковій системі числення.

Приклади.

1) Переведення числа 1101011_2 до вісімкової та шітнадцяткової систем числення:

   $1101011_2 = \underbrace{001}_{1}\underbrace{101}_{5}\ \underbrace{011}_{3}{}_2 = 153_8$;

   $1101011_2 = \underbrace{0110}_{6}\ \underbrace{1011}_{B}{}_2 = 6B_{16}$.

   Якщо для коректного групування не вистачає розрядів додаємо зліва до числа довільну кількість нулів (це, очевидно, не вплине на його значення).

2) Переведення числа 123 з вісімкової та шітнадцяткової систем до двійкової системи числення:

   $123_8 = \underbrace{001}_{1}\underbrace{010}_{2}\ \underbrace{011}_{3}{}_2 = 1010011_2$

   $123_{16} = \underbrace{0001}_{1}\underbrace{0010}_{2}\ \underbrace{0011}_{3}{}_2 = 100100011_2$

1.1.2. Переведення правильних дробів

Нехай задано дріб A у довільній позиційній системі числення і його необхідно перевести у нову систему з основою p, тобто необхідно його привести до вигляду [2]:

$$K(A) = a_{-1}p^{-1} + a_{-2}p^{-2} + \ldots + a_{-k+1}p^{-k+1} + a_{-k}p^{-k}. \qquad (1.5)$$

Якщо аналогічно переводу цілих чисел у виразі (1.5) обидві частини виразу поділити на p^{-1}, що еквівалентно множенню на p, то отримаємо:

$$Ap = a_{-1} + A_1,$$

де $K(A_1) = a_{-2}p^{-1} + a_{-3}p^{-2} + \ldots + a_{-k}p^{-k+1}$ – дробова частина добутку; a_{-1} – ціла частина результату, яка буде першою цифрою дробу в новій системі числення.

Помноживши дробову частину A_1 результату знову на p отримаємо:

$$A_1p = a_{-2} + A_2,$$

де $K(A_2) = a_{-3}p^{-1} + a_{-4}p^{-2} + \ldots + a_{-k}p^{-k+2}$ – дробова частина добутку; a_{-2} – ціла частина результату, яка буде другою цифрою дробу в новій системі числення.

Таким чином, при переведенні числа до іншої системи числення вираз (1.5) представляється за схемою Горнера:

$$K(A) = p^{-1}(a_{-1} + p^{-1}(a_{-2} + \ldots + p^{-1}(a_{-k+1} + a_{-k}p^{-1} + \ldots)))$$

Отже, в результаті серії множень заданої дробової частини числа на основу нової системи числення p знаходимо коефіцієнти A_i та цілі частини добутків a_{-i}, які і є цифрами дробової частини дробу в новій системі числення.

Правило переводу правильного дробу із однієї позиційної системи числення у іншу можна сформулювати наступним чином: необхідно задану дробову частину послідовно множити на основу нової системи числення, записану в старій системі числення, повторюючи дану процедуру до отримання необхідної точності. Дріб в новій системі числення буде записуватись у вигляді цілих частин добутків, починаючи з першої отриманої цілої частини.

Приклади.

1) Перевести число $0{,}8125_{10}$ до двійкової системи числення:

Ціла частина	Дробова частина
0	8125 × 2
1	625 × 2
1	25 × 2
0	5 × 2
1	0

$0{,}8125_{10} = 0{,}1101_2$

2) Перевести число $0{,}101_2$ до десяткової системи числення:

Ціла частина	Дробова частина
0	101 × 1010
110	01 × 1010
10	1 × 1010
101	0

$0{,}101_2 = 0{,}625_{10}$

3) Перевести число $0{,}8125_{10}$ до вісімкової системи числення:

Ціла частина	Дробова частина
0	8125 × 8
6	5 × 8
4	0

$0{,}8125_{10} = 0{,}64_8$

4) Перевести число $0{,}525_{10}$ до двійкової системи числення:

Ціла частина	Дробова частина
0	525 × 2
1	05 × 2
0	1 × 2
0	2 × 2
0	4 × 2
0	8 × 2
1	6 × 2
1	2 × 2
...	...

$0{,}525_{10} = 0{,}1000011(0011)_2$

З наведених вище прикладів можна бачити, що деякі дробові числа в інших системах числення не можуть бути представлені в скінченному вигляді, зокрема розглянуте вище число $0{,}525_{10}$ у двійковій системі числення є нескінченним дробом.

Як і у випадку з цілими числами, переведення дробових чисел у іншу систему числення можна виконувати безпосередньо за допомогою формули представлення їх кількісного еквівалента (1.5).

Приклад.

Перевести число $0{,}12A_{16}$ у десяткову систему числення:

- цифри: $a_{-1} = 1_{16} = 1_{10}$, $a_{-2} = 2_{16} = 2_{10}$, $a_{-3} = A_{16} = 10_{10}$
- основа: $p = 10_{16} = 16_{10}$.

Тоді:

$$0{,}12A_{16} = 1_{10}\cdot 16^{-1} + 2_{10}\cdot 16^{-2} + 10_{10}\cdot 16^{-3} = 0{,}0727539_{10}$$

При цьому для переведення десяткових дробів до інших систем числення простіше використовувати процедуру основану на *схемі Горнера та множенні*, а для переходу з інших систем *до десяткової* – другий спосіб, оснований на *безпосередньому застосуванні виразу* (1.5).

Для переведення дробів між двійковою, вісімковою та шістнадцятковими системами можна виконувати взаємну заміну груп розрядів у двійковій системі числення вісімкові та шістнадцяткові цифри. Причому, вісімкові цифри замінюються на відповідні 3-розрядні двійкові числа, а вісімкові – на відповідні 4-розрядні двійкові числа, починаючи з найбільш значущого розряду (зліва направо). Якщо в крайній правій позиції не вистачає розрядів для формування групи, дробову частину будь-якого числа можна доповнити довільною кількістю нулів.

Приклади.

1) Перевести число $0{,}10010101_2$ до вісімкової та шістнадцятвової систем числення:

   $0{,}10010101_2 = 0{,}\underbrace{100}_{4}\underbrace{101}_{5}\underbrace{010}_{2}{}_2 = 0{,}452_8$;

   $0{,}10010101_2 = 0{,}\underbrace{1001}_{9}\underbrace{0101}_{5}{}_2 = 0{,}95_{16}$.

2) Перевести числа з вісімкової та шістнадцяткової систем до двійкової системи числення:

   $0{,}123_8 = 0{,}\underbrace{001}_{1}\underbrace{010}_{2}\underbrace{011}_{3}{}_2$;

   $0{,}1AF_{16} = 0{,}\underbrace{0001}_{1}\underbrace{1010}_{A}\underbrace{1111}_{F}{}_2$.

1.1.3. Переведення неправильних дробів

Для переведення неправильних дробів (чисел, що мають і цілу, і дробову частину) з однієї однорідної позиційної системи числення у іншу необхідно *окремо перевести цілу й дробову частини числа*, а далі з обох цих частин утворити число у заданій системі числення. Для переходу у десяткову систему числення можна одразу використовувати формулу (1.3) [2].

Перехід між *вісімковою та шістнадцятковою* системами найзручніше здійснювати *через двійкову систему, перегруповуючи розряди по тріадам і тетрадам*.

Приклади.

1) Перевести число $23{,}125_{10}$ до двійкової системи числення:

Переводимо цілу частину:

23	2			
22	11	2		
1	10	5	2	
	1	4	2	2
		1	2	1
			0	

Переводимо дробову частину:

0	125 × 2
0	25 × 2
0	5 × 2
1	0

Таким чином: $23_{10} = 10111_2$, $0{,}125_{10} = 0{,}001_2$

$23{,}125_{10} = 10111{,}001_2$

2) Перевести число $10101{,}101_2$ до десяткової системи числення:

$10101{,}101_2 = 1\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} =$

$= 16 + 4 + 1 + 0{,}5 + 0{,}125 = 21{,}625_{10}$.

3) Перевести число $10101{,}101_2$ до вісімкової системи числення:

$10101{,}101_2 = \underbrace{010}_{2}\,\underbrace{101}_{5},\underbrace{101}_{5}{}_2 = 25{,}5_8$

4) Перевести число $7B2{,}E_{16}$ до двійкової системи числення:

$7B2{,}E_{16} = \underbrace{7}_{0111}\,\underbrace{B}_{1011}\,\underbrace{2}_{0010}\,,\,\underbrace{E}_{1110}{}_{16} = 11110110010{,}111_2$

5) Перевести число $175{,}24_8$ до щістнадцяткової системи числення:

$175{,}24_8 = \underbrace{1}_{001}\,\underbrace{7}_{111}\,\underbrace{5}_{101},\,\underbrace{2}_{010}\,\underbrace{4}_{100}{}_8 = \underbrace{0111}_{7}\,\underbrace{1101}_{D},\underbrace{0101}_{5}{}_2 = 7D{,}5_{16}$.

1.2. Вибір оптимальної системи числення для застосування в ЕОМ

Для застосування в ЕОМ найбільш зручними є позиційні однорідні системи числення. Це обумовлено використанням у них однієї основи, однакової кількість символів у всіх розрядах, що, як наслідок, дозволяє

досягти найбільш раціонального використання апаратури та спростити алгоритми виконання арифметичних операцій [2].

При аналізі оптимальності використання систем числення в обчислювальній апаратурі слід враховувати ряд факторів, основним серед яких є *економічність системи числення* D_i – кількість деталей апаратури, необхідних для представлення числа в i-й системі числення. Можна вважати, що величина D_i пропорційна наступному виразу:

$$D_i \approx p_i \cdot n_i,$$

де p_i – основа i-ї системи числення, яка у виразі визначає кількість апаратури необхідної для реалізації цифр системи числення; n_i – кількість розрядів, необхідних для представлення заданого числа.

Кількість чисел, яку можна представити у i-й системі числення:

$$N_i = p_i^{n_i},$$

звідки:

$$n_i = \log_{p_i} N_i,$$

а економічність системи числення D_i:

$$D_i = p_i \log_{p_i} N_i.$$

Дослідимо цю функцію на екстремум відносно змінної p_i, вважаючи, що вона змінюється не дискретно, а неперервно:

$$D_i = p_i \frac{\ln N_i}{\ln p_i} \quad \Rightarrow \quad \frac{dD_i}{dp_i} = \ln N_i \left(\frac{1}{\ln p_i} + p_i \cdot (-1) \cdot \frac{1}{\ln^2 p_i} \cdot \frac{1}{p_i} \right) = \frac{\ln N_i (\ln p_i - 1)}{\ln^2 p_i},$$

$$\frac{dD_i}{dp_i} = 0 \quad \Rightarrow \quad \frac{\ln N_i (\ln p_i - 1)}{\ln^2 p_i} = 0 \quad \Rightarrow \quad \ln p_i - 1 = 0 \quad \Rightarrow \quad p_i = e.$$

Вигляд функції $D_i(p_i)$ при N_i 100, 1000 та 1000000 наведено на рис. 1.2.

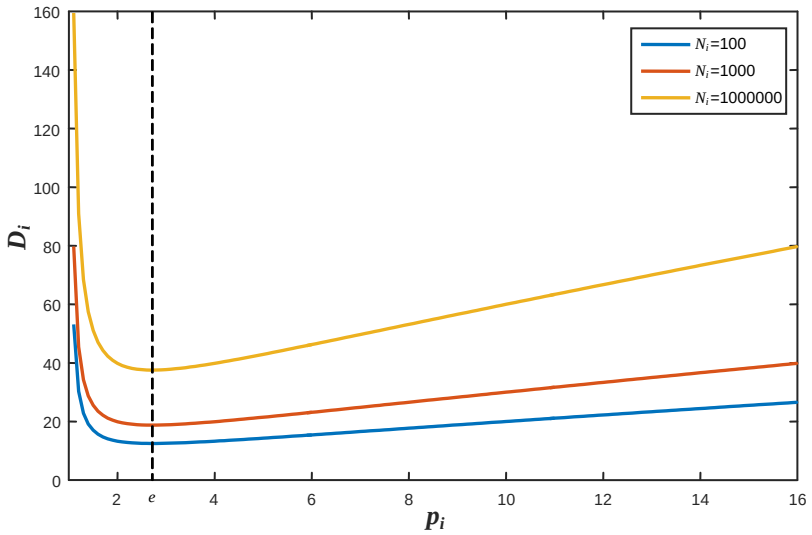


Рисунок 1.2 – Поведінка функції $D_i(p_i)$ для різних N_i

Очевидно, що використовувати систему числення з дробовою чи ірраціональною основою $p_{\text{опт}} = e = 2{,}7183\ldots$ менш практично, ніж з цілою. Досліджуючи поведінку функції $D_i(p_i)$ та обираючи цілі значення основ поблизу $p_{\text{опт}} = e$, можна з'ясувати, що найбільш оптимальними за критерієм економічності будуть системи числення з основами 3, потім 2 та 4. Водночас, враховуючи, що складність апаратних пристроїв зростає, а їх надійність знижується зі збільшенням кількості значень, які можуть приймати цифри в системі числення, обраної для даного обчислювального пристрою, виявляється, що для застосування у ЕОМ найбільш доцільною буде система числення з основою $p = 2$, тобто *двійкова система числення*.

Використання двійкової системи числення є більш доцільним з огляду не тільки на її економічність, але й виходячи з простоти виконання у ній арифметичних операцій.

Оскільки, ми вже визначилися з тим, що двійкова система числення є найбільш доцільною для застосування в ЕОМ, надалі працюватимемо саме з нею.

Зауважимо, що розряд у двійковій системі числення має спеціальну назву – *біт*, від англійського *bit* (**bi**nary digi**t**).

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Наведіть класифікацію систем числення. Які з цих систем знаходять застосування у електронно-обчислювальній техніці?
2. Що таке база, базис системи числення та кількісний еквівалент числа?
3. Наведіть формулу за якою розраховуються кількісні еквіваленти цілого, дробового числа та неправильного дробу.
4. Наведіть способи переведення цілих чисел між десятковою, двійковою, вісімковою та шістнадцятковою системи числення. Які з цих способів є більш простими та в якому випадку?
5. Наведіть способи переведення дробових чисел між десятковою, двійковою, вісімковою та шістнадцятковою системи числення.
6. Поясніть чому для застосування в електронно-обчислювальній апаратурі обрана саме двійкова система числення.
7. Переведіть числа до *десяткової* системи числення:
   а) $101001{,}101_2$;
   б) $322{,}12_8$;
   в) $3E{,}1A_{16}$.

8. Переведіть числа до *двійкової* системи числення:
    а) $210{,}6875_{10}$;
    б) $113{,}24_{8}$;
    в) $A3{,}7_{16}$.
9. Переведіть числа до *вісімкової* системи числення:
    а) $53{,}25_{10}$;
    б) $10011{,}01_{2}$;
    в) $2A{,}5C_{16}$
10. Переведіть числа до *шістнадцяткової* системи числення:
    а) $49{,}5625_{10}$;
    б) $100010{,}0101_{2}$;
    в) $27{,}2_{8}$.
11. Переведіть ціле двійкове число 10111_{2} у код Грея та представте число 10100_{G} записане у коді Грея у вигляді двійкового значення.

2. ДВІЙКОВА АРИФМЕТИКА

Правила виконання арифметичних дій над двійковими числами задаються таблицями двійкового додавання, віднімання та множення див. табл. 2.1 [2].

Таблиця 2.1 – Таблиці арифметичних дій у двійковій системі числення

Таблиця двійкового додавання	Таблиця двійкового віднімання	Таблиця двійкового множення
0 + 0 = 0	0 − 0 = 0	0 × 0 = 0
0 + 1 = 1	запозичення 0 − 1 = 1	0 × 1 = 0
1 + 0 = 1	1 − 0 = 1	1 × 0 = 0
1 + 1 = 0 перенесення	1 − 1 = 0	1 × 1 = 1

2.1. Додавання цілих двійкових чисел.

При додаванні будь-яких, в тому числі й двійкових чисел, в кожному розряді проводиться складання цифр доданків, причому якщо сума цифр a_i та b_i перевищує допустиме значення для цифри $a_i + b_i > p - 1$ в даній системі числення, то відбувається перенесення 1 у наступний розряд [1–3].

При цьому необхідно враховувати, що 1 + 1 дають 0 в даному розряді та перенесення 1 в наступний розряд.

Приклади.

1) Додати два двійкові числа $x = 1101_2$ та $y = 101_2$:

```
  1 1
  1101   x
+  101   y
 10010   x + y
```

Результат $x + y = 1101_2 + 101_2 = 10010_2$.

2) Додати двійкові числа $x = 1101_2$, $y = 101_2$ та $z = 111_2$:

```
  1
  1 1 1
  1101   x
+  101   y
   111   z
 11001   x + y + z = 1101₂ + 101₂ + 111₂= 11001₂.
```

2.2. Віднімання цілих двійкових чисел

При відніманні двійкових чисел у кожному розряді проводиться віднімання цифр, причому, якщо в даному розряді значення цифри зменшуваного менше, ніж від'ємника $a_i < b_i$, то із старшого(их) розрядів запозичується (займається) 1. Ця запозичена 1 для даного розряду, очевидно, матиме значення p в будь-якій системі числення [1–3].

Приклад.

Відняти від двійкового числа $x = 10010_2$ число $y = 101_2$:

```
  -1 -1
  10010   x
-   101   y
  01101   x − y
```

Результат $x - y = 10010_2 - 101_2 = 1101_2$.

2.3. Від'ємні числа та доповнювальний код

Яким чином у ЕОМ реалізувати від'ємні числа? Оскільки ми вже вміємо віднімати, можемо спробувати отримати від'ємне число, віднявши від 0 його додатній аналог (наприклад, від 0 відняти 1, щоб отримати −1):

```
   0000
-     1
(1)1111
```

Як бачимо, отримане число буде нескінченно довгим, оскільки доведеться робити безліч запозичень з кожної нульової цифри числа 0. Однак, при реалізації арифметики на реальній апаратурі розрядність чисел фізично не може бути нескінченною. Що ж тоді робити?

Спробуємо обмежити розрядність чисел, скажімо, 4-ма розрядами та дослідимо їх «поведінку». При цьому обмежений набір розрядів називатимемо *розрядною сіткою*, а кількість розрядів сітки – *довжиною розрядної сітки*. Всі розряди, які під час виконання арифметичних дій *виходитимуть за межі* розрядної сітки *ігноруватимемо*.

Наведемо числову шкалу додатних та від'ємних 4-розрядних чисел – див. рис. 2.1 (від'ємні числа отримані шляхом віднімання від 0 їх додатних еквівалентів) [3].

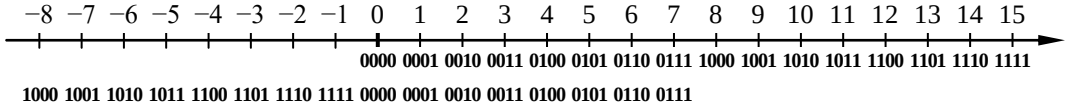


Рисунок 2.1 – Числова шкала двійкових 4-розрядних чисел

З рисунку 2.1 видно, що від'ємні двійкові числа відповідають додатним числам починаючи з числа $+8_{10}$. Нескладно також помітити, що найстарший розряд від'ємних чисел завжди дорівнює 1, тому коли надалі мова буде йти про знакові числа, тож називатимемо його *знаковим*. Можна також помітити, що двійковий еквівалент числа $+8_{10}$ відповідає числу -8_{10} і дорівнює 1000_2. Отже дане число можна було б віднести як до додатних, так і до від'ємних чисел, однак з огляду на зручність використання у якості ознаки «від'ємності» його старшого розряду, доцільніше 1000_2 вважати все ж від'ємним, тобто $1000_2 = -8_{10}$.

Оскільки для отримання від'ємних чисел лише було обмежено розмір розрядної сітки, розглянуті вище операції додавання і віднімання мають працювати і з від'ємними числами за умови, що операнди та результат знаходитимуться в межах числової шкали. В цьому нескладно переконатися, якщо, наприклад, до числа -1 додати $+1$, то отримаємо 0:

```
  1111
- 0001
1←0000
```

Тут «**1 ←** » символічно позначає вихід за розрядну сітку.

Перевіримо також чи отримаємо, наприклад, -5_{10}, якщо до -3_{10} додамо -2_{10} або $+4_{10}$, якщо від $+1_{10}$ віднімемо -3_{10}. Запишемо зазначені числа в двійковій системі числення:

$1101 = -3_{10}$ $\quad$ $1110 = -2_{10}$ $\quad$ $1011 = -5_{10}$

$0001 = +1_{10}$ $\quad$ $0100 = +4_{10}$

Тоді:

1) $-3_{10} + (-2_{10}) = -5_{10}$

```
  1101
+ 1110
1←1011 = -5₁₀
```

2) $1_{10} - (-3_{10}) = 4_{10}$

```
  0001
- 1101
1←0100 = +4₁₀
```

Очевидно, що збільшення розрядної сітки ніяк не вплине на спосіб виконання арифметичних дій додавання і віднімання як для додатних, так і для від'ємних чисел.

Для того, щоб отримати від'ємне число, необхідно виконати віднімання від 0 його додатного аналогу, що не є досить зручним. Розглянемо спосіб, який дозволяє дещо спростити цю процедуру.

Замість 0 виконуватимемо віднімання від числа, яке представляє собою *двійковий нуль* доповнений *зліва одиницею,* так щоб ця одиниця якраз знаходилась за межами розрядної сітки. Очевидно, що така маніпуляція не впливатиме на результат віднімання, оскільки з цієї одиниці за необхідності братиметься запозичення, а якщо в цьому не буде потреби (тоді і тільки тоді, коли віднімається 0), вона просто ігноруватиметься у зв'язку з нашою попередньою домовленістю про ігнорування розрядів за межами розрядної сітки. Саме дане число матиме вигляд $1\,|\,00\ldots000_2$, де символом « | » позначено межу розрядної сітки. Не складно переконатись, що якщо довжина розрядної сітки складає N, то наш «модифікований» нуль дорівнюватиме 2^N. Тоді від'ємне число $-A$ можна буде знайти як (див. також рис. 2.1):

$$2^N - A.$$

Правомірним буде такий запис:

$$2^N - A = \underbrace{(\underbrace{2^N - 1}_{11\ldots111})}_{\bar{A}} - A + 1.$$

Нескладно також переконатися, що число $2^N - 1$ – це число, виду $0\,|\,11\ldots111_2$, яке містить всі 1 у всіх розрядах в межах розрядної сітки, число $(2^N - 1) - A$ – це число $\bar{A}$, яке буде *інверсним* по відношенню до A, тобто число, яке містить нулі у всіх розрядах, в яких A містить одиниці й усі одиниці в розрядах, в яких A містить нулі. Тоді процедура пошуку від'ємного числа зводиться до двох простих кроків:

1) Інвертувати розряди числа A, що відповідатиме значенню $\bar{A} = (2^N - 1) - A$.
2) Додати до $\bar{A} + 1$, що дасть $A^* = 2^N - A$, наймолодші N розрядів якого є від'ємним числом $-A$.

Наведена вище процедура називається *доповненням до двох*, а результат, який при цьому утворюється – *доповнювальним кодом* (рос. дополнительным кодом, англ. two's complement). Пошук доповнювального коду числа еквівалентний обчисленню його від'ємного значення [1–3]:

$$A_{\text{доп}} = -A = \bar{a}_n \bar{a}_{n-1} \ldots \bar{a}_2 \bar{a}_1 \bar{a}_0 + 1,$$

де операція «+» є такою, що при її виконанні відкидається розряд, який утворюється поза розрядною сіткою.

Приклад.

Знайти від'ємне значення числа 78_{10} у вигляді 8-розрядного двійкового числа:

$x = \mathbf{78_{10} = 01001110_2}$

1) Інвертування

$\overline{x} = \mathbf{10110001_2}$

2) Додавання одиниці

```
  10110001
+        1
  10110010 = -78₁₀ = −x
```

Перевірка:

$78_{10} + (-78_{10}) = 0$

```
  01001110
+ 10110010
1←00000000
```

Зауважимо, що використання доповнювального коду дозволяє взагалі позбутися операції віднімання, так як останнє може бути замінене на додавання від'ємного числа. Сама ж процедура обчислення доповнювального коду передбачає виконання операцій інвертування та додавання одиниці.

Приклад.

Відняти в двійковій системі числення два числа: $115_{10} - 47_{10}$ користуючись доповнювальним кодом.

$115_{10} - 47_{10} = 68_{10}$

$\mathbf{115_{10} = 01110011_2}$

$\mathbf{47_{10} = 00101111_2}$

Доповнювальний код:

```
¬ 00101111      інвертування
  11010000
+        1      додавання 1
  11010001 = -47₁₀
```

Пошук різниці шляхом додавання від'ємного числа $115_{10} - 47_{10}$:

```
  01110011
+ 11010001
1←01000100 = 68₁₀
```

Відзначимо ще один факт. Якщо необхідно змінити розрядність (збільшити довжину розрядної сітки) *беззнакових* та *додатних* двійкових чисел, то до них *зліва* можна дописати *необхідну кількість нулів*, а для

від'ємних чисел – необхідно скопіювати знаковий розряд, інакше кажучи дописати *зліва необхідну кількість одиниць*. Дана операція називається знаковим розширенням.

Приклади.

1) Представити число -7_{10} у 8-розрядному форматі, якщо відомо, що в 4-розрядному воно виглядає як 1001_2.
   Скористаємось записаним вище правилом знакового розширення, тоді для 8-розрядної сітки число -7_{10} можна записати як:
   $$11111001_2.$$
2) Перевести до шістнадцяткової системи числення число 10110_2, якщо відомо, що це від'ємне число:
   Для запису шістнадцяткового числа необхідно утворити тетради – групи по 4 розряди, але оскільки число 10110_2 від'ємне, для утворення тетрад необхідно зліва до нього дописувати одиниці:
   $$10110_2 = \underbrace{1111}_{\text{F}}\ \underbrace{0110}_{6}{}_2 = \text{F6}_{16} = -10_{10}.$$

Для представлення від'ємних чисел також застосовується ще один поширений формат. Цей формат називають *прямим кодом* і його ідея полягає у тому, що двійкове число з N розрядами (бітами) використовує найбільш значущий розряд як знак, а інші $N-1$ розрядів представляють абсолютне значення даного числа. При цьому число вважається додатним, якщо знаковий розряд дорівнює 0 та є від'ємним, якщо він дорівнює 1. Шкала 4-розрядних двійкових чисел в прямому коді наведена на рис. 2.2 [3].

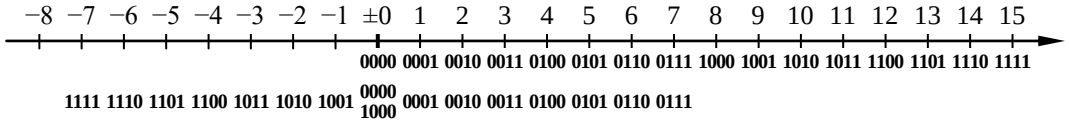


Рисунок 2.2 – Числова шкала 4-розрядних чисел в прямому коді

Особливістю прямого коду є наявність додатного і від'ємного нуля: -0 (1000_2) та $+0$ (0000_2).

Приклад.

Представлення числа $+5_{10}$ та -5_{10} у прямому коді:

$$\mathbf{0|101_2} = +5_{10}$$
$$\mathbf{1|101_2} = -5_{10}$$

знак | *абсолютне значення числа*

На перший погляд може здатися, що прямий код простіший, однак виконання дій у ньому є більш складним, зокрема додавання додатного та від'ємного числа не можливо виконати природним чином, наприклад:

$5_{10} + (-3_{10})$

0|101	5_{10}
+ 1|011	-3_{10}
1←0|000	0 (*невірно*)

Щоб операція виконувалась вірно необхідно в залежності від знакового розряду виконувати операцію віднімання або додавання. Такий аналіз, а також необхідність мати пристрої додавання та віднімання призводить до ускладнення апаратної частини, що є основним недоліком використання прямого коду.

Тим не менш, прямий код широко використовується в для представлення мантис чисел з плаваючою точкою.

Окрім доповнювального та прямого коду для представлення від'ємних чисел раніше застосовували інверсний код. При його використанні необхідно виконувати додаткове корегування, яке полягає у додаванні 1 до результату при виникненні перенесення 1 за розрядну сітку під час виконанні арифметичних дій (так зване циклічне перенесення). В інверсному коді, так само як і прямому наявний −0, який тут має вигляд двійкового числа виду $11\ldots111_2$. Наразі інверсний код для представлення від'ємних чисел майже не застосовується [3].

2.4. Множення двійкових чисел

Множення двійкових чисел виконується за тими ж правилами, що й десяткове, за виключенням того, що дії здійснюються над нулями та одиницями за допомогою таблиць двійкового множення та додавання [1–3].

Приклад.

Множення двох додатних цілих 4-розрядних двійкових чисел.

0101	множене
× 1101	множник
0101	
+ 0000	частинні добутки
0101	
0101	
1000001	результат

$0101_2 \times 1101_2 = 1000001_2$. В десятковій системі: $5_{10} \times 13_{10} = 65_{10}$.

Частинні добутки формуються шляхом множення окремих розрядів множника на все множене. Далі частинні добутки зсуваються та додаються, що дає остаточний результат.

В загальному випадку при множенні двох N-розрядних чисел формується $2N$-розрядний результат.

Множення знакових чисел можна виконувати за наведеним вище правилом, але враховуючи розрядність результату. Мається на увазі, що для отримання коректних частинних добутків, розрядність від'ємних множеного та/або множника *не повинна бути меншою* за розрядність результату. Повернімося до наведеного вище прикладу: якщо числа в ньому вважати знаковими, то множник 1101_2 мав би бути від'ємним числом -3_{10} і при виконанні множення мав би бути отриманий результат -15_{10}, який у двійковій системі записувався як 10001_2 (7-розрядним числом 1110001_2), натомість було отримано 1000001_2, що є невірним. Якщо ж врахувати, що розрядність для представлення результату (числа -15_{10}) має складати щонайменше 5 розрядів, то множник слід було б записати як 11101_2. Тоді, обмежуючи розрядну сітку результату п'ятьма розрядами отримаємо:

```
  00101     множене
× 11101     множник
-------
  00101
  0000
+ 101       частинні добутки
  01
  1
-------
1←10001     результат
```

Тепер результат є вірним.

Так само для перемноження від'ємних чисел -3_{10} та -7_{10} маємо отримати результат $+21_{10}$, для представлення якого необхідно 6 біт (разом із знаковим розрядом), отже у двійковій системі перемножуватимемо числа 111101_2 та 111001_2 відповідно, що дасть:

```
   111001
× 111101
--------
   111001
   00000
+ 1001
   001
   01
   1
--------
10←010101
```

Число 010101_2 є числом 21_{10}, що є вірним.

Для отримання коректного добутку чисел -2_{10} та -3_{10} необхідно щонайменше 4 розряди, тому в двійковій системі слід перемножувати числа $1110_2 \times 1101_2 = 0110_2$:

```
  1110
× 1101
  1110
  000
+ 10
  0
1←0110
```

Щоб однозначно не помилитися з вибором довжини розрядної сітки при обчисленні добутку N-розрядних знакових чисел у доповнювальному коді, можна прийняти розрядність результату рівною $2N$.

Більш просто операція множення виконується в прямому коді: вона виконується безпосередньо тільки над абсолютними значеннями числа, а знак числа визначається як сума по модулю 2 знакових розрядів множеного та множника (якщо розряди знаку чисел, що множаться однакові, то добуток є додатним, якщо знаки різні, то добуток – від'ємний). Розрядність результату N-розрядних знакових чисел у прямому коді становитиме $2N - 1$.

Приклад.

Знайти добуток чисел $+5_{10}$ та -7_{10} у прямому коді:

$$+5_{10} = 0|101_2,$$
$$-7_{10} = 1|111_2.$$

Знак добутку: 1, оскільки отримуване число є від'ємним – знаки множеного і множника відрізняються. Сам добуток:

```
  101
× 111
  101
+ 101
 101
100011
```

Отже в прямому коді:

$$0|101_2 \times 1|111_2 = 1|100011_2 = -35_{10}.$$

Нескладно переконатися, що *множення числа на основу* тої системи числення, в якій воно представлене *еквівалентно зсуву* цього числа *вліво*.

2.5. Ділення двійкових чисел

Так само як і множення ділення двійкових чисел аналогічне діленню десяткових чисел. Найпростіший алгоритм ділення полягає у тому, що дільник на кожному кроці віднімається від діленого стільки разів (починаючи зі старших розрядів), скільки це можливо для отримання найменшого додатного залишку. Кількість повторів показує, скільки разів дільник вкладається у ділене, і дорівнює частці.

Наприклад, поділимо 35_{10} на 7_{10}:

1) $35 - 7 = 28$;
2) $28 - 7 = 21$;
3) $21 - 7 = 14$;
4) $14 - 7 = 7$;
5) $7 - 7 = 0$.

Отже відповідь є 5_{10}, оскільки число 7_{10} можна 5 разів відняти від числа 35_{10}. Очевидно, що така процедура є досить тривалою, тому її можна скоротити: визначити скільки разів дільник вкладається в старшу частину діленого, а потім послідовно переходити до його молодших частин, враховуючи отримувані на кожному кроці остачі.

Таку процедуру у двійковій системі числення реалізують два найбільш поширені алгоритми, які називаються діленням *з відновленням* і *без відновлення остачі* [2]. Обидва ці алгоритми схожі, але алгоритм без відновлення остачі є дещо більш коротким, а отже швидкодіючим, тому зупинимось на ньому.

Крок алгоритму ділення без відновлення остачі має наступний вигляд [2]:

$$X - Y = a_0,$$

де X – ділене; Y – дільник, зсунутий до найбільш значущого розряду діленого; a_0 – залишок. Віднімання може бути замінене на операцію додавання доповнювального коду $Y_{\text{доп}}$. Для дільника має бути використано найменшу кількість розрядів, необхідну для його представлення.

Якщо $a_0 \geq 0$, то старший розряд частки $c_n = 1$, інакше – старший розряд частки $c_n = 0$.

Для визначення наступної цифри частки c_{n-1} необхідно знайти наступний залишок a_1, який визначається як:

$$a_1 = \begin{cases} 2a_0 - Y, & \text{якщо } a_0 \geq 0, \\ 2a_0 + Y, & \text{якщо } a_0 < 0. \end{cases}$$

Якщо $a_1 \geq 0$, то наступний, менший розряд частки $c_{n-1} = 1$, інакше – наступний розряд частки $c_{n-1} = 0$.

Наступний розряд частки визначається за допомогою залишку a_2, який знаходять аналогічно:

$$a_2 = \begin{cases} 2a_1 - Y, & \text{якщо } a_1 \geq 0, \\ 2a_1 + Y, & \text{якщо } a_1 < 0. \end{cases}$$

І так далі необхідно здійснити стільки ітерацій (зсувів), на скільки розрядність діленого більше дільника.

Як видно знак залишку визначає не тільки чергову цифру частки, але і характер наступної процедури: додавання дільника до збільшеного вдвічі (зсунутого вліво) діленого, якщо залишок менший 0, та віднімання дільника із зсунутого залишку, якщо він до зсуву був більшим або рівним 0.

Приклади.

1) Поділити 35_{10} на 5_{10}, тобто 100011_2 на 101_2.

Доповнимо числа знаковим розрядом (відділено знаком « | »), а дільник також представимо у доповнювальному коді для уникнення безпосередньої операції віднімання:

$35_{10} = 0|100011_2$
$+5_{10} = 0|101_2$
$-5_{10} = 1|011_2$

Процедура ділення:

	0|100011	ділене
+	1|011000	дільник у доп. коді (віднімання)
	1|111011	$a_0 < 0$, $C = 0_2$
←	1|110110	$2a_0$ (зсув вліво)
+	0|101000	дільник у прямому коді (додавання)
	0|011110	$a_1 > 0$, $C = 01_2$
←	0|111100	$2a_1$ (зсув вліво)
+	1|011000	дільник у доп. коді (віднімання)
	0|010100	$a_2 > 0$, $C = 011_2$
←	0|101000	$2a_2$ (зсув вліво)
+	1|011000	дільник у доп. коді (віднімання)
	0|000000	$a_3 = 0$, $C = 0111_2$

Отже: $100011_2 / 101_2 = 111_2$ ($35_{10} / 5_{10} = 7_{10}$)

2) Поділити 204_{10} на 12_{10}, тобто 11001100_2 на 1100_2.

Доповнимо числа знаковим розрядом (відділено знаком « | »), а дільник також представимо у доповнювальному коді для уникнення безпосередньої операції віднімання:

$204_{10} = 0|11001100_2$
$+12_{10} = 0|1100_2$
$-12_{10} = 1|0100_2$

Процедура ділення:

`0|11001100`	ділене
`+ 1|01000000`	дільник (віднімання)
`0|00001100`	$a_0 > 0$, $C = 1_2$
`← 0|00011000`	$2a_0$
`+ 1|01000000`	дільник (віднімання)
`1|01011000`	$a_1 < 0$, $C = 10_2$
`← 0|10110000`	$2a_1$
`+ 0|11000000`	дільник (додавання)
`1|01110000`	$a_2 < 0$, $C = 100_2$
`← 0|11100000`	$2a_2$
`+ 0|11000000`	дільник (додавання)
`1|10100000`	$a_3 < 0$, $C = 1000_2$
`← 1|01000000`	$2a_3$
`+ 0|11000000`	дільник (додавання)
`0|00000000`	$a_4 = 0$, $C = 10001_2$

Отже: $11001100_2 / 1100_2 = 10001_2$ ($204_{10} / 12_{10} = 17_{10}$)

Для виконання ділення знакових двійкових чисел найпростіше знайти частку їх абсолютних значень, після чого відповідним чином змінити знак результату. В цьому відношенні більш зручним способом представлення чисел буде прямий код.

Як і у випадку з множенням, нескладно впевнитись, що *ділення числа на основу* тої системи числення, в якій воно представлене *еквівалентно зсуву* цього числа *вправо*. При такому способі ділення слід зважати на те, що від'ємні числа у доповнювальному коді при зсуві вправо необхідно доповнювати *одиницями зліва*, щоб не змінити знак числа.

2.6. Числа з фіксованою точкою

Одним із способів представлення дробів у двійковій системі числення є числа з фіксованою точкою (комою) [2–3]. Цей спосіб є аналогічним представленню неправильних дробів, що розглядався вище. Він передбачає двійкову точку між бітами цілої та дробової частини числа, аналогічно десятковій точці між цілою і дробовою частинами звичайного десяткового числа.

Як і у всіх попередніх випадках представлення двійкових чисел (знакових/беззнакових), числа з фіксованою точкою є лише набором біт. Тому не існує способу дізнатися де саме знаходиться двійкова точка, окрім як про це заздалегідь домовитись і використовувати цю домовленість при інтерпретації чисел. Наприклад, ми можемо домовитись, що наші дробові

числа матимуть по чотири біти у цілій та дробовій частинах, тоді число $11{,}75_{10}$ в двійковій системі матиме вигляд:

$$1011{,}1100_2 = 2^3 + 2^1 + 2^0 + 2^{-1} + 2^{-1} = 8 + 2 + 1 + 0{,}5 + 0{,}25$$

Без точки, просто 10111100_2.

Знакові числа з фіксованою точкою можуть бути представлені як в прямому, так і доповнювальному коді. При цьому в прямому коді, як і раніше, найстарший біт виступає у ролі знакового, а доповнювальний код можна отримати за допомогою інверсії бітів абсолютного значення числа і додавання 1 до наймолодшого розряду. Наприклад, якщо ми тепер домовимось, що для представлення цілої частини числа будемо використовувати 8 біт, а для дробової – 4 біти, то число $-27{,}375_{10}$ можна представити як:

$\mathbf{00011011{,}0110_2}$ ← абсолютне значення
$\mathbf{10011011{,}0110_2}$ ← прямий код
$\mathbf{11100100{,}1010_2}$ ← доповнювальний код

Арифметичні дії з числами з фіксованою точкою

Операції додавання та віднімання чисел з фіксованою точкою виконуються аналогічно звичайним цілим числам.

Приклади.

Приймемо у прикладах розрядність цілої і дробової частин рівною 4.

1) Додати числа $1{,}625_{10}$ та $3{,}5_{10}$:

$$\mathbf{0001{,}1010_2 = 1{,}625_{10}}$$
$$\mathbf{0011{,}1000_2 = 3{,}5_{10}}$$

$$\begin{array}{r} \mathbf{00011010} \\ \mathbf{+\ 00111000} \\ \hline \mathbf{01010010} \end{array}$$

Відповідь: $0101{,}0010_2 = 5{,}125_{10}$.

2) Обчислити $1{,}625_{10} - 3{,}5_{10}$:

Виконаємо дану дію через додавання, представивши від’ємник у доповнювальному коді:

$$\mathbf{0001{,}1010_2 = 1{,}625_{10}}$$
$$\mathbf{1100{,}1000_2 = -3{,}5_{10}}$$

$$\begin{array}{r} \mathbf{00011010} \\ \mathbf{+\ 11001000} \\ \hline \mathbf{11100010} \end{array}$$

Відповідь: $1110{,}0010_2 = -1{,}875_{10}$.

3) Обчислити $-0{,}75_{10} - 2{,}125_{10}$:
Представимо обидва числа в доповнювальному коді:

$$1111{,}0100_2 = -0{,}75_{10}$$
$$1101{,}1110_2 = -2{,}125_{10}$$

$$\begin{array}{r} 11110100 \\ +\ 11011110 \\ \hline 1\leftarrow 11010010 \end{array}$$

Відповідь: $1101{,}0010_2 = -2{,}875_{10}$.

4) Знайти суму чисел $7{,}875_{10}$ та $-4{,}375_{10}$ у прямому коді:

$$0|0111{,}1110_2 = +7{,}875_{10}$$
$$1|0100{,}0110_2 = -4{,}375_{10}$$

$$\begin{array}{r} 0|01111110 \\ -\ 1|01000110 \\ \hline 0|00111000 \end{array}$$

Відповідь: $0|0011{,}1000_2 = +3{,}5_{10}$.

5) Знайти різницю чисел $-3{,}5_{10}$ та $-6{,}75_{10}$ у прямому коді:

$$1|0011{,}1000_2 = -3{,}5_{10}$$
$$1|0110{,}1100_2 = -6{,}75_{10}$$

$$1|00111000 - 1|01101100 \Rightarrow 0|01101100 - 1|00111000$$

$$\begin{array}{r} 0|01101100 \\ -\ 1|00111000 \\ \hline 0|00110100 \end{array}$$

Відповідь: $0|0011{,}0100_2 = +3{,}25_{10}$.

Як бачимо з останнього прикладу, при виконанні дій у прямому коді, зокрема при пошуку різниці, доводиться змінювати знак та порядок виконання операції так, щоб віднімання виконувалось від більшого за модулем числа. Якщо цього не зробити, доведеться мати справу з від'ємними числами у доповнювальному коді.

При виконання множення чисел з фіксованою точкою слід пам'ятати, що добуток, матиме вдвічі довшу розрядну сітку в порівнянні з довжиною розрядних сіток множеного і множника. При цьому збільшиться розрядність як цілої, так і дробової частин.

Приклад.

Знайти добуток чисел $10{,}25_{10}$ та $4{,}75_{10}$.

Виконаємо операцію множення і в десятковій і в двійковій системах.

```
  10,25          1010,01
× 04,75        × 0100,11
 ,5125           10,1001
7,175           101,001
41,00           0000,00
000,0          00000,0
48,6875         101001
                000000
                0110000,1011
```

Число $110000{,}1011_2 = 48{,}6875_{10}$.

З прикладу видно, що розрядність дробової частини зросла вдвічі як у десятковій, так і в двійкових системах числення. При цьому відбувся зсув коми (точки). Однак, якщо положення точки є фіксованим, то отриманий результат має бути скорегованим – *зсунутим вправо на кількість розрядів відведену під дробову частину*.

Зауважимо, що якщо вважати в наведеному прикладі дробову частину 2-х розрядною (як у множеного і множника), то результат має представлятись як $110000{,}10_2 = 48{,}5_{10}$, що означатиме втрату точності.

Множення чисел у прямому коді виконується за аналогічними принципами, причому робота зі знаковими числами тут виконуватиметься простіше.

У випадку з діленням чисел з фіксованою точкою навпаки розрядність результату буде зменшено вдвічі, тому для досягнення більшої точності процедуру ділення за необхідності можна продовжити.

Приклад.

$+2{,}75_{10} = 0|10{,}11_2$

$+1{,}25_{10} = 0|1{,}01_2$

$-1{,}25_{10} = 1|0{,}11_2$

```
   0|1011      ділене
+  1|0110      дільник у доп. коді (віднімання)
   0|0001      a0 > 0,      C = 12
←  0|0010      2a0
+  1|0110      дільник у доп. коді (віднімання)
   1|1000      a1 < 0,      C = 102
```

Результат ділення: 10_2

Продовжуємо процедуру для отримання вищої точності.

← 1|0000	$2a_1$
+ 0|1010	дільник у прямому коді (додавання)
1|1010	$a_2 < 0$, $C = 10{,}0_2$
← 1|0100	$2a_2$
+ 0|1010	дільник у прямому коді (додавання)
1|1110	$a_3 < 0$, $C = 10{,}00_2$
← 1|1100	$2a_3$
+ 0|1010	дільник у прямому коді (додавання)
0|0110	$a_4 > 0$, $C = 10{,}001_2$
← 0|1100	$2a_4$
+ 1|0110	дільник у доп. коді (віднімання)
0|0010	$a_5 > 0$, $C = 10{,}0011_2$
…	…

Уточнений результат: $10{,}0011_2$.

2.7. Числа з плаваючою точкою (IEEE-754/IEC 60559)

Числа з фіксованою точкою мають один значний недолік – для того, щоб представляти з їх допомогою одночасно малі дробові та великі цілі значення необхідно, щоб ці числа мали значну розрядність. На практиці ж, коли мова йде про значні числа, наприклад, порядку мільйона, зрідка бувають важливими мільйонні долі цього числа і навпаки – якщо необхідно працювати зі значеннями в діапазоні [0; 1], ціла частина чисел з фіксованою точкою виявиться незадіяною. Враховуючи це було запропоновано ще один формат представлення, який називається *числами з плаваючою точкою*. Даний формат дозволяє при сталій розрядності представляти з високою точністю як малі дробові, так і великі цілі значення.

До середини 80-х років кожен тип комп’ютерів використовував свій власний варіант чисел з плаваючою точкою, що створювало безліч незручностей при перенесенні програм. Через це було прийнято рішення стандартизувати формат представлення, в результаті чого в 1985 році був прийнятий стандарт IEEE-754 (трохи пізніше IEC 60559), який наразі використовують усі сучасні комп’ютери та мікроконтроллери.

Даний стандарт оснований на прийнятому в науці записі чисел в експоненціальному вигляді:

Заряд електрона: $-1{,}60217662 \times 10^{-19}$ Кл

Маса Землі 1 M⊕: $5{,}97219 \times 10^{24}$ кг

Експоненціальний запис числа передбачає, що для його представлення використовується неправильний дріб, який обов'язково має цілу частину, на яку відводиться один розряд – *мантису*. Для коректного задання числа його мантиса множиться на показникову функцію – *характеристику числа*:

$$\underbrace{\pm 1{,}23456789}_{\text{мантиса}} \times \underbrace{10^{\pm 123}}_{\text{характеристика}}$$

Запис мантиси у вигляді дробу з цілою частиною, на яку обов'язково відводиться 1 розряд називається *нормалізованим* виглядом чи записом.

За аналогією до експоненціального запису, число з плаваючою точкою представляється як показано на рис. 2.3.

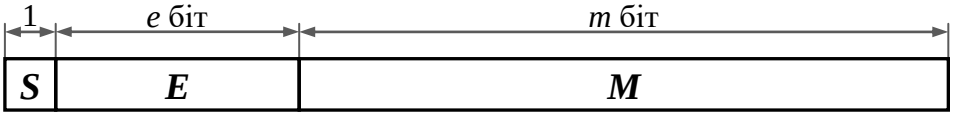


Рисунок 2.3 – Число з плаваючою точкою у стандартах IEEE-754/IEC 60559

Як видно з рисунка 2.3, число з плаваючою точкою має три поля:

- S – знак числа (1 біт);
- E – показник (e біт);
- M – мантиса (m біт).

На відміну від експоненціального запису, в числах з плаваючою точкою у ролі характеристики виступає показникова функція за основою 2, тобто 2^E, а мантиса зазвичай є нормалізованим двійковим дробом і тому знаходиться в діапазоні $1 \le M < 2$. Очевидно, що мантиса разом зі знаком утворюють неправильний дріб в прямому коді.

В загальному випадку кількісний еквівалент числа з плаваючою точкою визначається за формулою:

$$(-1)^S \cdot 2^{E-2^{e-1}+1} \cdot (1 + M \cdot 2^{-m}) = (-1)^S \cdot 2^{E-2^{e-1}+1} \cdot \left(1 + \sum_{i=1}^{m} 2^{-i} \cdot M_{[m-i]}\right),$$

де $M_{[i]}$ – позначає i-й біт мантиси.

При цьому нескладно помітити, що будучи представленою *в нормалізованому вигляді мантиса* завжди матиме *старший біт рівний* 1, тому його не зберігають, але враховують під час обчислень.

Водночас, відсутність старшого біту у мантисі створює неоднозначність при записі числа 0, тому *для запису нуля* передбачений спеціальний формат числа:

S = 0 або 1,
E = 0,
M = 0.

Як і для всіх чисел в прямому коді, число 0 тут може бути як додатнім, так і від'ємним.

Окрім нуля, числа з плаваючою точкою також мають додаткові спеціалізовані значення, такі як нескінченність ±∞ (Inf) та «*не число*» (NaN), які можуть свідчити про особливі ситуації, що виникають під час виконання операцій, зокрема: ділення на 0, множення нескінченності на 0, ділення нескінченності на нескінченність тощо.

Нескінченність ±∞ (Inf):

$S = 0/1,$
$E = 2^e - 1$ (всі 1),
$M = 0.$

«*Не число*» (NaN):

$S = 0/1,$
$E = 2^e - 1$ (всі 1),
M – будь-яке окрім 0.

Для підвищення точності в околі 0 також *може* використовуватись *денормалізований* (*субнормальний*) вигляд мантиси:

$$0 \le M < 1.$$

В цьому випадку кількісний еквівалент числа обчислюється за формулою:

$$(-1)^S \cdot 2^{E-2^{e-1}+2} \cdot (M \cdot 2^{-m}) = (-1)^S \cdot 2^{E-2^{e-1}+2} \cdot \sum_{i=1}^{m} 2^{-i} \cdot M_{[m-i]}.$$

Ознакою денормалізованого числа з плаваючою точкою є *нульовий показник та ненульова мантиса*:

$S = 0/1,$
$E = 0,$
M – будь-яке.

Відзначимо, що підтримка денормалізованих чисел реалізується не у всіх ЕОМ, оскільки вимагає ускладнення арифметичних пристроїв.

Арифметичні дії з числами з плаваючою точкою. Для виконання операцій *віднімання та додавання*, показники чисел з плаваючою точкою повинні бути вирівняні. Щоб виконати вирівнювання показників необхідно менший показник довести до значення більшого, для чого зсунути мантису меншого числа (числа з меншим показником) на таку кількість розрядів вправо, яка дорівнює абсолютному значенню різниці показників. Після цього можна виконати відповідну операцію (додавання чи віднімання) мантис. Віднімання та додавання мантис виконується за

правилами для чисел з фіксованою точною представлених в прямому коді. У якості показника результату слід взяти показник більшого за модулем числа. За необхідності слід виконати процедуру нормалізації результату – привести мантису результату до нормалізованого вигляду, відповідним чином корегуючи при цьому і показник.

Операції *множення та ділення* чисел з плаваючою точкою зводиться до відповідно додавання та віднімання показників із подальшим множенням та ділення мантис, після якого виконується нормалізація результату. Операції множення та ділення мантис виконуються за відповідним правилами виконання операцій над числами з фіксованою точкою в прямому коді.

При виконанні всіх дій передбачається, що мантиси чисел є повними, тобто доповнені розрядом, що відповідає за цілу частину мантиси – 1 зліва. Операцій зі спеціальними числами ±0, ±∞ та NaN виконуються за особливими правилами.

При виконанні операцій над числами з плаваючою точкою вводиться поняття *машинного нуля* – найменшого числа ε такого, що додавання до якого 1 ще може бути розпізнане машиною, тобто $1 + \varepsilon \neq 1$. Число ε характеризує точність виконання операцій. Поняття машинного ε не використовується у стандарті.

Розглянемо деякі конкретні варіанти чисел з плаваючою точкою визначені в стандарті IEEE-754/IEC 60559.

Число з плаваючою точкою одинарної точності (*single float*). Формат такого числа показано нижче на рис. 2.4.

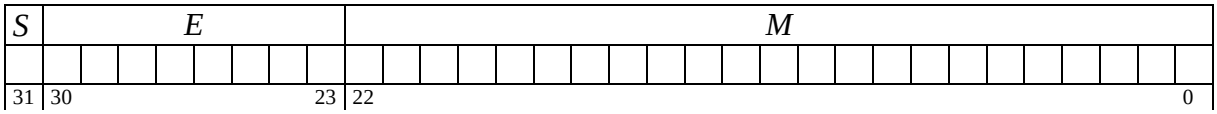


Рисунок 2.4 – Формат числа з плаваючою точкою одинарної точності

Число з плаваючою точкою одинарної точності має:

- *S* – знак 1 біт;
- *E* – показник 8 біт;
- *M* – мантису 23 біта.

Кількісні еквіваленти нормалізованого числа обчислюється за формулою:

$$(-1)^S \cdot 2^{(E-127)} \cdot (1 + M \cdot 2^{-23}),$$

денормалізованого числа – за формулою:

$$(-1)^S \cdot 2^{(E-126)} \cdot (M \cdot 2^{-23}).$$

Приклади представлення деяких чисел з плаваючою точкою одинарної точності.

1) Число 1:
   S = 0
   E = 01111111 (127_{10})
   M = [1.]00000000000000000000000

2) Число 3,25:
   S = 0
   E = 10000000 (128_{10})
   M = [1.]10100000000000000000000 ($1{,}625_{10}$)

3) Число −0,1:
   S = 1
   E = 01111011 (123_{10})
   M = [1.]10011001100110011001101 ($1{,}600000023841858_{10}$)

4) Число 0:
   S = 0 або **1**
   E = 00000000
   M = 00000000000000000000000

5) Нескінченність ±∞ (Inf):
   S = 0 або **1**
   E = 11111111
   M = 00000000000000000000000

6) «Не число» (NaN):
   S = 0 або **1**
   E = 11111111
   M = будь-яке окрім **0**

7) Деномалізоване число $8{,}816208 \cdot 10^{-39}$
   S = 0
   E = 00000000
   M = [0.]11000000000000000000000 ($0{,}75_{10}$)

8) Деномалізоване число $1{,}4 \cdot 10^{-45}$
   S = 0
   E = 00000000
   M = [0.]00000000000000000000001 ($0{,}00000011920928955078125_{10}$)

Число з плаваючою точкою подвійної точності (*double float*). Формат такого числа показано нижче на рис. 2.5.

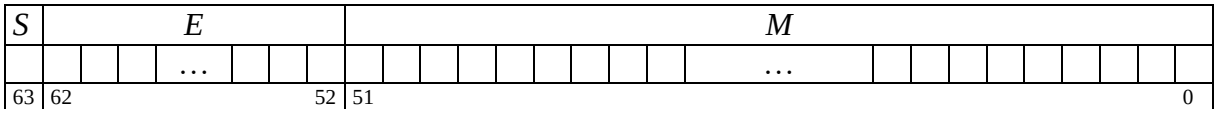


Рисунок 2.5 – Формат числа з плаваючою точкою одинарної точності

Число з плаваючою точкою подвійної точності має:

- S – знак 1 біт;
- E – показник 11 біт;
- M – мантису 52 біта.

Кількісні еквіваленти нормалізованого числа обчислюється за формулою:

$$(-1)^S \cdot 2^{(E-1023)} \cdot (1 + M \cdot 2^{-52}),$$

денормалізованого числа – за формулою:

$$(-1)^S \cdot 2^{(E-1022)} \cdot (M \cdot 2^{-52}).$$

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Наведіть правила за якими виконуються операції додавання, віднімання та множення двійкових чисел.
2. Які способи представляються від'ємних чисел використовуються в електронно-обчислювальній апаратурі? В чому полягають переваги і недоліки цих способів?
3. Наведіть процедуру переведення чисел у доповнювальний код.
4. Як коректно збільшити розрядність числа у доповнювальному коді?
5. Які особливості слід враховувати при виконанні операцій із від'ємними двійковими числами?
6. Наведіть алгоритм ділення двійкових чисел без відновлення остачі.
7. Якими є особливості виконання арифметичних дій над дробовими числами з фіксованою точкою? Як змінюється розрядність цілої і дробової частин при виконанні множення та ділення?
8. Яким арифметичним операціям відповідають зсув чисел вліво та вправо?
9. Який формат мають числа з плаваючою точкою згідно стандарту IEEE-754/IEC 60559? Як обчислюється кількісний еквівалент чисел з плаваючою точкою одиничної та подвійної точності?

10. Як представляються спеціальні значення 0, нескінченність, «не число» та денормалізоване число у форматі чисел з плаваючою точкою?
11. Представте числа у двійковому доповнювальному коді:
    а) 5_{10}
    б) -25_{10}
    в) 10110110_2
12. Виконайте дії у двійковій системі числення:
    а) $12_{10} + 17_{10}$
    б) $7_{10} - 11_{10}$
    в) $-3_{10} + 10_{10}$
    г) $-5_{10} - 7_{10}$
    д) $2_{10} \cdot 4_{10}$
    е) $-5_{10} \cdot 8_{10}$
    є) $11_{10} \cdot 2^3{}_{10}$
    ж) $36_{10} / 4_{10}$
    з) $14_{10} / 7_{10}$
    і) $12_{10} / -5_{10}$
13. Виконайте дії у двійковій системі числення над дробовими числами з фіксованою точкою:
    а) $3{,}5_{10} + 2{,}25_{10}$
    б) $1{,}75_{10} - 0{,}25_{10}$
    в) $2{,}5_{10} \cdot 6{,}25_{10}$
    г) $1{,}25_{10} - 0{,}5_{10}$
14. Надайте двійкове представлення дробових значень у форматі чисел з плаваючою точкою одинарної точності:
    а) -123
    б) $23{,}75$
    в) $12{,}625 \cdot 2^{-133} \approx 1{,}1594231 \cdot 10^{-39}$

3. ЛОГІЧНІ ФУНКЦІЇ ТА ЛОГІЧНІ ЕЛЕМЕНТИ

3.1. Логічні функції

Цифрові пристрої, які є основою сучасних електронно-обчислювальних засобів використовують двійкове представлення даних, у зв'язку з чим оперують сигналами, що приймають лише два рівні: низький (0) та високий (1). Таким чином, для аналізу та синтезу цифрових пристроїв знадобився б математичний апарат, який за аналогією оперував би лише двома значеннями. Виявляється, що такий математичний апарат вже існує досить давно і називається *алгеброю логіки*. Він був розроблений ще у 1847-1854 роках Джорджем Булем, тому на його честь також називається *Булевою алгеброю*. У 1938 році американський вчений *Клод Шеннон* вперше запропонував застосовувати алгебру логіки для синтезу цифрових схем [1, 7].

Алгебра логіки основана на функціях та змінних, які можуть приймати лише два значення 0 («*хибність*») або 1 («*істинність*»). Функції алгебри логіки називають *логічними* або *перемикальними*. Для логічної функції n змінних використовуватимемо наступне позначення:

$$f(\nu) = f(x_n,\ldots,x_1),$$

де $\nu = (x_n,\ldots,x_1)$ – сукупність змінних, яка може розглядатись як n-мірний вектор. Оскільки кожна змінна x_p, $i = 1, 2,\ldots, n$ може приймати лише два значення 0 та 1, кількість комбінацій $(x_n,\ldots,x_1)$ є обмеженою. В загальному випадку конкретне значення змінної $x_p = \{0$ або $1\}$ позначатимемо як e_p.

Областю визначення функції n змінних є сукупність точок $(x_n,\ldots,x_1)$ n-мірного простору, причому кожна точка задається певною комбінацією значень цих змінних. Конкретні точки, які задають область визначення функції $f(\nu)$ будемо позначати як:

$$\nu = (e_n,\ldots,e_p,\ldots,e_1),$$

де $i = e_n\ldots e_p\ldots e_1$ – всі точки області визначення функції n змінних, які можна пронумерувати за допомогою двійкових n-розрядних чисел або десяткових чисел i. Очевидно, що оскільки існує 2^n різних n-розрядних двійкових чисел, то і *область визначення функції n змінних* також складатиметься із 2^n точок:

$$\nu = \{v_0, v_1,\ldots, v_{2^n-1}\}.$$

Для задання функції n змінних необхідно вказати її значення у всіх точках області визначення, тобто у $m = 2^n$ точках. Якщо функція n змінних

повинна бути задана m двійковими значеннями, то з цих m значень можна сформувати 2^m різних комбінацій, а отже задати $2^m = 2^{2^n}$ різних функцій.

Якщо розглядати перемикальні функції *однієї* змінної, то таких функцій можна задати $2^{2^1} = 4$, вони наведені в табл. 3.1.

Таблиця 3.1 – Логічні функції однієї змінної

Функція $f_i(x)$	**Змінна** x		**Позначення**	**Назва**
	0	1		
f_0	0	0	$f(x) = 0$	константа *нуль*
f_1	0	1	$f(x) \equiv x$	тотожність x (тавтологія)
f_2	1	0	$\neg x$ або $\bar{x}$	заперечення x (логічне НІ)
f_3	1	1	$f(x) = 1$	константа *одиниця*

Також значний інтерес представляють функції *двох* змінних. Таких функцій може бути $2^{2^2} = 16$. Всі вони наведені у табл. 3.2.

Таблиця 3.2 – Логічні функції двох змінних

Функція $f_i(x_2, x_1)$	**Змінні** x_1x_2				**Позначення**	**Назва**
	00	01	10	11		
f_0	0	0	0	0	$f(x_2, x_1) = 0$	константа нуль
f_1	0	0	0	1	$x_2 \wedge x_1$ або $x_2 \cdot x_1$	кон’юнкція (логічне І)
f_2	0	0	1	0	$\bar{x}_2 \wedge x_1$	заборона x_2
f_3	0	0	1	1	$f(x_2, x_1) \equiv x_1$	тотожність x_1 (тавтологія x_1)
f_4	0	1	0	0	$x_2 \wedge \bar{x}_1$	заборона x_1
f_5	0	1	0	1	$f(x_2, x_1) \equiv x_2$	тотожність x_2 (тавтологія x_2)
f_6	0	1	1	0	$x_2 \oplus x_1$	сума по модулю 2 (виключне АБО)
f_7	0	1	1	1	$x_2 \vee x_1$	диз’юнкція (логічне АБО)
f_8	1	0	0	0	$x_2 \downarrow x_1 = \neg(x_2 \vee x_1)$	стрілка Пірса (АБО-НІ)
f_9	1	0	0	1	$x_2 \equiv x_1$	рівнозначність
f_{10}	1	0	1	0	$\bar{x}_2$	заперечення x_2 (НІ x_2)
f_{11}	1	0	1	1	$x_2 \rightarrow x_1$	імплікація із x_2
f_{12}	1	1	0	0	$\bar{x}_1$	заперечення x_1 (НІ x_1)
f_{13}	1	1	0	1	$x_1 \rightarrow x_2$	імплікація із x_1
f_{14}	1	1	1	0	$x_2 \mid x_1 = \neg(x_2 \wedge x_1)$	штрих Шеффера (І-НІ)
f_{15}	1	1	1	1	$f(x_2, x_1) = 1$	константа одиниця

Функції n змінних можуть залежати не від всіх змінних $(x_n,\ldots,x_1)$. Такі функції називаються *виродженими*. Прикладами вироджених функцій у наведених вище таблицях 3.1 та 3.2 є константи одиниць та нулів. Окрім того у таблиці 3.2 виродженими є функції тотожності та заперечення змінних (f_3, f_5, f_{10}, f_{12}).

Якщо деяка функція приймає взаємно протилежні значення у всіх відповідних точках області визначення по відношенню до деякої іншої

функції $f(\nu)$, то її називають *інверсною* до $f(\nu)$. Функцію інверсну, по відношенню до $f(\nu)$ будемо позначати як $\overline{f(\nu)}$. Одним із прикладів взаємно інверсних функцій у табл. 3.2 є кон'юнкція (І) та штрих Шеффера (І-НІ). Очевидно, що серед усіх функцій n змінних 2^{2^n-1} є взаємно інверсними.

Функція n змінних називається *повністю визначеною*, якщо її значення задані у всіх 2^n точках області визначення даної функції. Якщо значення функції не визначено хоча б в одній точці, то таку функцію називають *неповністю визначеною*. Якщо ж значення функції не задані в жодній з точок області визначення, то таку функцію називають *повністю невизначеною* та позначають як $\hbar$.

Якщо значення неповністю визначеної функції не задане в точці ν_i, то будемо його задавати довільним чином за допомогою коефіцієнта $c_i = \Ф$, де Ф – позначає суміщений символ 0 та 1. Формально це записуватиметься як $f(\nu_i) = c_i$. *Неповністю визначені функції можна довизначити* довільним чином, вважаючи c_i рівним 0 або 1. Якщо значення функції не задано в m точках, то її, очевидно, можна довизначити 2^m різними способами.

Будь-яка *функція багатьох змінних* може бути отримана із функцій *двох змінних* за допомогою *композиції* (*суперпозиції*) – шляхом підстановки функцій замість аргументів інших функції. Така підстановка можлива з огляду на те, що значення, які приймають і змінні, і функції співпадають (дорівнюють 0 або 1).

При використанні операції суперпозиції для отримання будь-якої функції з функцій 2-х змінних не обов'язково використовувати всі функції з таблиці 3.2. Набір логічних функцій, який забезпечує представлення будь-якої іншої функції шляхом суперпозиції функцій даного набору називають *функціонально повним набором* або *базисом* [1–3, 7]. Функціонально повні базиси утворюють наступні функції:

1) кон'юнкція (І) та заперечення (НІ);
2) диз'юнкція (АБО) та заперечення (НІ);
3) заборона та константа нуля;
4) заборона та константа одиниці;
5) сума по модулю 2 (виключне АБО) та імплікація;
6) стрілка Пірса (АБО-НІ);
7) штрих Шеффера (І-НІ);

Набір функцій кон'юнкції (І), диз'юнкції (АБО) та заперечення (НІ) отримав назву *основного функціонально-повного набору*.

3.2. Логічні елементи

Як вже зазначалось в минулому розділі для реалізації будь-якої перемикальної функції достатньо обмеженого набору функції двох змінних, які називають функціональним базисом. Таким чином, якщо створити набір реальних пристроїв, які здатні реалізувати функціональний базис, стане можливим створення і цифрових пристроїв довільної складності [1, 3, 4, 6]. Даний розділ якраз присвячений даному питанню.

Пристрої, що реалізують деяку логічну (перемикальну) функцію називаються *логічними елементами*. Насправді логічні елементи можуть бути реалізовані як механічні, оптичні, пневматичні, електромеханічні (реле, MEMS), електронні (діоди, тріоди, транзистори) та інші пристрої. Ми зі зрозумілих причин зупинимось на розгляді лише електромеханічних та електронних варіантах їх реалізації. Розглянемо найбільш вживані логічні елементи.

Логічний елемент І (*AND*)

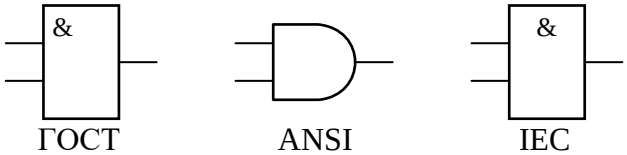
Логічний елемент І реалізує логічну операцію *кон'юнкції*. Словесний опис функціонування такого елемента може заданий наступним чином: на виході логічного елементу І формуватиметься *одиниця* тоді і тільки тоді, коли *на всі* його входи подано *одиниці*. Для аналітичного запису операції кон'юнкції використовують наступне позначення:

$$y = x_2 \wedge x_1 \text{ або } y = x_2 \cdot x_1 \text{ або } y = x_2 \,\&\, x_1$$

Опис логічних функцій та елементів також зручно представляти *таблицями істинності*. Таблиця істинності для логічного елемента І з двома входами матиме наступний вигляд:

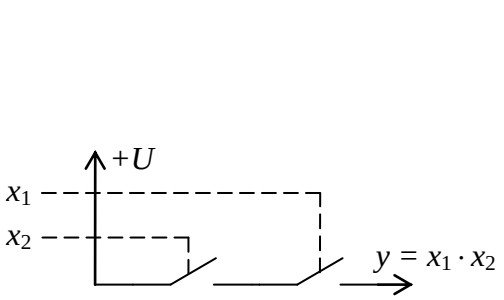
x_2	x_1	$y = x_2 \wedge x_1$
0	0	0
0	1	0
1	0	0
1	1	1

Для ідентифікації логічного елемента І на схемах використовують спеціальне умовне позначення. Це позначення визначено в основних стандартах ГОСТ, ANSI та IEC. В кожному стандарті використовується своє позначення, тому наведемо їх всі:

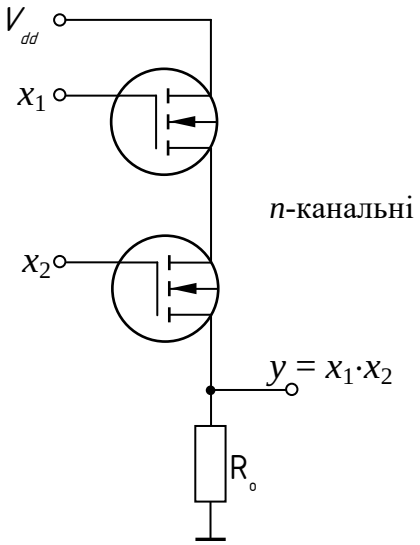


ГОСТ ANSI IEC

Як саме можна реалізувати логічний елемент І у вигляді електромеханічного пристрою на основі реле [6], а також елементарної електронної схеми на МДН-транзисторах показано нижче на рис. 3.1.



Електромеханічна реалізація елемента І на основі реле. Для спрощення керуючі кола, на які подаються вхідні сигнали показано пунктирною лінією



Спрощена електронна схема елемента І на основі *n*-канальних МДН-транзисторів

Рисунок 3.1 – Реалізація логічного елементу І на основі реле та *n*-канальних МДН-транзисторів

В обох схемах при подачі високого рівня напруги (1) на обидва керуючі входи x_2 та x_1 замикаються обидва контакти реле в електромеханічній схемі та виникає провідність в обох транзисторах електронної схеми, в результаті на виході формується напруга високого рівня (1). У випадку, якщо обидва або хоча б один контакт реле чи транзистор не спрацює – на виході формуватиметься напруга низького рівня (0), що матиме місце, коли хоча б на один вхід буде подано низький рівень напруги (0). Це повністю відповідає таблиці істинності логічного елемента І та операції кон'юнкції.

Логічний елемент АБО (*OR*)

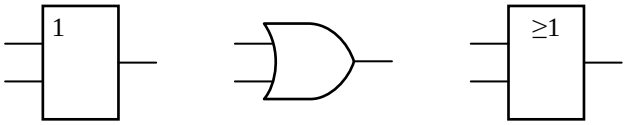
Логічний елемент АБО реалізує логічну операцію *диз'юнкції*. Словесний опис функціонування такого елемента є наступним: на виході логічного елементу АБО формуватиметься *одиниця* тоді і тільки тоді, коли *хоча б на один* його вхід буде подано *одиницю*. Для аналітичного запису операції диз'юнкції у рівняннях використовують наступне позначення:

$$y = x_2 \vee x_1.$$

Таблиця істинності для логічного елемента АБО з двома входами матиме наступний вигляд:

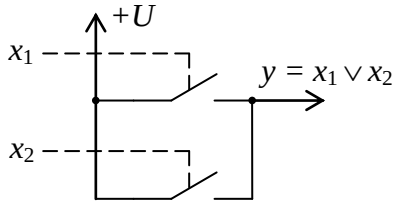
x_2	x_1	$y = x_2 \vee x_1$
0	0	0
0	1	1
1	0	1
1	1	1

Для позначення логічного елемента АБО на схемах, згідно стандартів ГОСТ, ANSI та IEC використовують наступні символи:

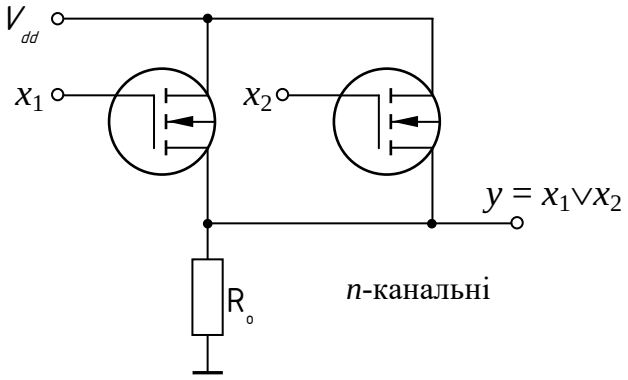


ГОСТ ANSI IEC

Реалізації елемента АБО у вигляді електромеханічного пристрою на основі реле [6] та елементарної електронної схеми на МДН-транзисторах показані на рис. 3.2.



Електромеханічна реалізація елемента АБО на основі реле. Для спрощення керуючі кола, на які подаються сигнали показано пунктирною лінією



Спрощена електронна схема елемента АБО на основі *n*-канальних МДН-транзисторів

Рисунок 3.2 – Реалізація логічного елементу АБО на основі реле та *n*-канальних МДН-транзисторів

В обох схемах при подачі високого рівня напруги (1) на обидва або хоча б на один керуючий вхід x_2 та x_1 реле або транзистори комутуватимуть вихід з напругою живлення, що дасть на виході високий рівень сигналу (1). У випадку, якщо обидва контакти реле не спрацюють чи не відкриються транзистори – на виході формуватиметься напруга низького рівня (0). Така ситуація виникатиме тоді, коли на обидва входи подаватиметься низький рівень напруги (0).

Логічний елемент НІ (*NOT*)

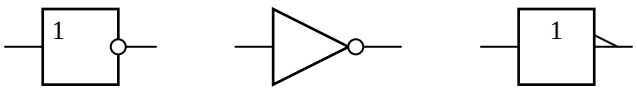
Логічний елемент НІ реалізує логічну операцію *заперечення*. Словесний опис функціонування такого елемента може бути наступним: логічний елемент НІ формує на своєму виході сигнал протилежний тому, що подано на його вхід. У зв'язку з цим, логічний елемент НІ також називають інвертором. Аналітично для запису операції заперечення використовують наступне позначення:

$$y = \neg x \text{ або } y = \overline{x}.$$

Таблиця істинності для логічного елемента НІ має наступний вигляд:

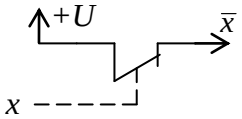
x_1	$y = \overline{x}$
0	1
1	0

Для позначення логічного елемента НІ на схемах, згідно стандартів ГОСТ, ANSI та IEC використовують наступні символи:

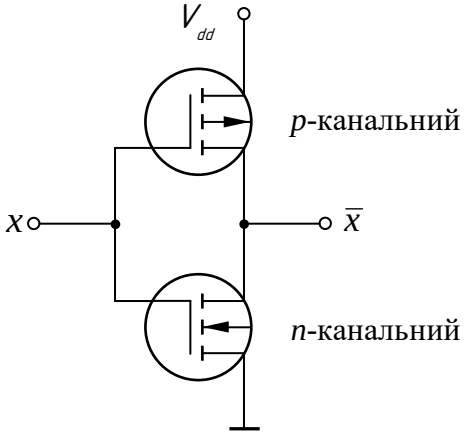


ГОСТ ANSI IEC

Реалізації елемента НІ у вигляді електромеханічного пристрою на основі реле [3, 6] та більш досконалої електронної схеми на комплементарних МДН-транзисторах показані на рис. 3.3.



Електромеханічна реалізація елемента НІ на основі реле розмикання. Для спрощення керуюче коло, на яке подається вхідний сигнал показано пунктирною лінією



Електронна схема елемента НІ на основі комплементарних МДН-транзисторів

Рисунок 3.3 – Реалізація логічного елементу НІ на основі реле розмикання та комплементарної пари МДН-транзисторів

В електромеханічній схемі при подачі високого рівня напруги (1) на керуючий вхід *x* реле розмикається, в результаті чого на виході

формується низький рівень сигналу (0). Якщо на вхід перестати подавати чи подати низьку напругу, то реле замкнеться, створюючи на виході високий рівень сигналу (1).

Електронна схема інвертора є більш складною в порівнянні з розглянутими вище схемами елементів І та АБО. Вона використовує комплементарну пару з двох *n*- та *p*-канального МДН-транзисторів. Дана схема працює наступним чином: коли на вхід поступає низький рівень сигналу (0), то верхній *p*-канальний транзистор відкривається, а нижній *n* – канальний закривається, при цьому вихід схеми комутується з напругою живлення, що призводить до появи на ньому високого рівня сигналу (1). Якщо на вхід поступає високий рівень сигналу (1), то відбувається все навпаки: верхній транзистор закривається, а нижній відкривається, комутуючи вихід зі спільним проводом, в результаті чого на виході формується низький рівень сигналу (0).

Наведена схема володіє однією суттєвою перевагою – при встановлені на виході необхідного сигналу через неї не протікає струм, чого не було у схемах І та АБО (при відкритих транзисторах у наведених вище схемах елементів І та АБО через обмежувальний резистор R_0 протікатиме постійний струм, що призведе до підвищення енерговитрат та збільшення виділення теплоти). Очевидно, що схеми елементів І та АБО можна модифікувати так, щоб вони також використовували комплементарні МДН-транзистори.

Як вже зазначалося кон’юнкція, диз’юнкція та заперечення (а отже і відповідні їм логічні елементи І, АБО та НІ) утворюють основний функціонально-повний набір, якого достатньо щоб реалізувати будь-яку іншу логічну функцію чи цифрову схему довільної складності. Тим не менш, розглянемо ще деякі логічні елементи, які досить широко застосовуються при проектуванні цифрових пристроїв.

Логічний елемент «Виключне АБО» (*XOR*)

Логічний елемент «Виключне АБО» реалізує логічну операцію *сума по модулю* 2. Словесний опис функціонування такого елемента може бути наступним: якщо елемент «Виключне АБО» має два вхідні сигнали, то на його виході формуватиметься *одиниця* тоді і тільки тоді, коли на обох його входах *сигнали відрізняються*. Для запису операції сума по модулю 2 використовують наступне позначення:

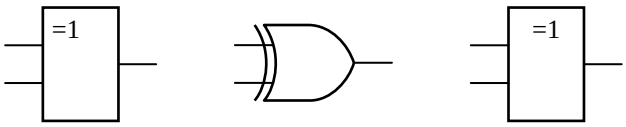
$$y = x_2 \oplus x_1 = \overline{x}_2 \cdot x_1 \vee x_2 \cdot \overline{x}_1.$$

Таблиця істинності логічного елемента «Виключне АБО» має наступний вигляд:

x_2	x_1	$y = x_2 \oplus x_1$
0	0	0
0	1	1
1	0	1
1	1	0

Нескладно помітити, що «Виключне АБО» реалізує таблицю двійкового додавання (див. табл. 3.2, функція f_6) без урахування перенесення. Звідси і походить назва відповідної йому операції «сума по модулю 2».

Для ідентифікації логічного елемента «Виключне АБО» на схемах, використовують спеціальне умовне позначення, яке згідно стандартів ГОСТ, ANSI та ISO/IEC матиме вигляд:



ГОСТ ANSI IEC

Логічний елемент І-НІ (*NAND*)

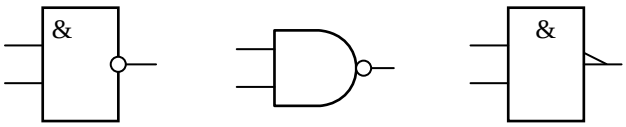
Логічний елемент І-НІ реалізує логічну операцію, яку інколи називають штрих Шеффера. Словесний опис функціонування такого елемента є наступним: на виході елемента І-НІ формується одиниця тоді і тільки тоді, коли на його вхід подано *хоча б один нуль*. Для запису операції І-НІ використовують наступне позначення:

$$y = \overline{x_2 \cdot x_1}\,.$$

Таблиця істинності логічного елемента І-НІ має вигляд:

x_2	x_1	$y = \overline{x_2 \cdot x_1}$
0	0	1
0	1	1
1	0	1
1	1	0

Для позначення І-НІ на схемах згідно стандартів ГОСТ, ANSI та IEC використовуються наступні символи:



ГОСТ ANSI IEC

Логічний елемент АБО-НІ (*NOR*)

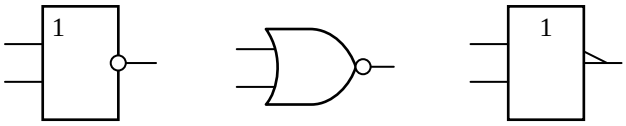
Логічний елемент АБО-НІ реалізує логічну операцію, яку інколи називають стрілка Пірса. Словесний опис функціонування такого елемента є наступним: на виході елемента АБО-НІ формується одиниця тоді і тільки тоді, коли на його *всі його входи* подано *нулі*. Для запису операції АБО-НІ використовують наступне позначення:

$$y = \overline{x_2 \vee x_1}\,.$$

Таблиця істинності логічного елемента АБО-НІ має вигляд:

x_2	x_1	$y = \overline{x_2 \vee x_1}$
0	0	1
0	1	0
1	0	0
1	1	0

Для позначення АБО-НІ на схемах згідно стандартів ГОСТ, ANSI та IEC використовуються наступні символи:



ГОСТ ANSI IEC

Питання та завдання для самоперевірки і контролю засвоєння знань

1. У скількох точках має бути визначена двійкова функція, якщо відомо, що вона залежить від *n* логічних змінних?
2. Скільки існує способів, щоб задати логічну функцію *n* змінних (скільки всього можна створити логічних функцій *n* змінних)?
3. Що стверджує принцип композиції (або суперпозиції)? Чому він є важливим?
4. Які функції називаються виродженими, неповністю визначеними та повністю невизначеними? Наведіть приклади.
5. Що таке функціонально повний базис (набір)? Наведіть приклади функціонально повних базисів.
6. Наведіть таблиці істинності основних логічних елементів.
7. Поясніть як на основі елементарних радіокомпонентів (реле, транзисторів) можна реалізувати логічні елементи?
8. Для логічних елементів І-НІ та АБО-НІ не наведено електричних схем, які б могли їх реалізовувати. За аналогією до розглянутих схем

логічних елементів І, АБО та НІ запропонуйте схемотехнічні реалізації логічних елементів І-НІ та АБО-НІ.

9. Запропонуйте удосконалені реалізації схем логічних елементів І та АБО, на основі комплементарних МДН-транзисторів.
10. Запропонуйте схемотехнічні реалізації логічних елементів І, АБО, І-НІ, АБО-НІ так, щоб вони мали більш ніж два входи.

4. АКСІОМИ, ТЕОРЕМИ ТА ТОТОЖНОСТІ АЛГЕБРИ ЛОГІКИ

4.1. Аксіоми алгебри логіки

Змінні будемо позначати латинськими літерами *x*, *y*, *z*… Ці змінні можуть приймати тільки два значення «0» («хибність») и «1» («істина»).

В алгебрі логіки визначені: одне відношення: відношення *еквівалентності* «=» та 3 операції: *диз'юнкція* – операція АБО, яка позначається знаком «∨», *кон'юнкція* – операція І, позначається символом «∧», амперсандом «&» або точкою «·», яку можна пропускати, а також *заперечення* – операція НІ, інверсія, позначається символом «¬», рискою над змінними або над елементами «0» і «1» [1, 7].

Наприклад: $x = y$, $x \vee y$, $x \cdot z = x \wedge z = xz$, $\bar{x}$, $\bar{y}$, $\bar{1}$, $\bar{0}$.

Відношення еквівалентності задовольняє 3 властивостям:

1) $x = x$ – рефлексивність;
2) якщо $x = y$, то $y = x$ – симетричність;
3) якщо $x = y$ та $y = z$, то $x = z$ – транзитивність.

Із відношення еквівалентності, випливає принцип підстановки: якщо $x = y$, то в будь-якій формулі, що містить x, замість x можна підставити y і в результаті буде отримана еквівалентна формула.

Алгебра логіки базується на наступних аксіомах:

$$\begin{cases} x = 0, \text{ якщо } x \neq 1 \\ x = 1, \text{ якщо } x \neq 0 \end{cases} \tag{4.1}$$

Таким чином в алгебрі логіки розглядаються лише двійкові змінні:

$$\begin{cases} 1 \vee 1 = 1 \\ 0 \cdot 0 = 0 \end{cases} \tag{4.2}$$

$$\begin{cases} 1 \cdot 1 = 1 \\ 0 \vee 0 = 0 \end{cases} \tag{4.3}$$

$$\begin{cases} 0 \vee 1 = 1 \vee 0 = 1 \\ 0 \cdot 1 = 1 \cdot 0 = 0 \end{cases} \tag{5.4}$$

Аксіоми (4.2)-(4.4) визначають операції *диз'юнкції* та *кон'юнкції*.

$$\begin{cases} \overline{0} = 1 \\ \overline{1} = 0 \end{cases} \qquad (4.5)$$

Аксіома (5.5) визначає *операцію заперечення*.

Принцип двоїстості

Із наведених аксіом, записаних парами, можна сформувати принцип двоїстості: якщо в аксіомах (4.2)-(4.5) виконати взаємну заміну операцій диз'юнкції та кон'юнкції, із одночасною заміною значень «0» та «1», то з одної аксіоми даної пари буде отримана інша.

4.2. Теореми і тотожності алгебри логіки

За допомогою аксіом алгебри логіки можна довести цілий ряд теорем, законів і тотожностей. Найчастіше вони доводяться методом перебору всіх значень змінних: якщо теорема істинна, то з урахуванням аксіом, рівняння, яке формує твердження теореми, повинно виконуватись при підставлянні будь-яких значень змінних в обидві частини даного рівняння. Вивівши основні теореми та закони, з їх допомогою можна доводити і виводити інші більш складні тотожності алгебри логіки.

1) Ідемпотентні закони:

$$\begin{cases} x \vee x = x \\ x \cdot x = x \end{cases} \qquad (4.6)$$

Доведення:

При $x = 0 \quad \Rightarrow \quad 0 \vee 0 = 0$ (аксіома (5.3));
При $x = 1 \quad \Rightarrow \quad 1 \vee 1 = 1$ (аксіома (5.2));
При $x = 0 \quad \Rightarrow \quad 0 \cdot 0 = 0$ (аксіома (5.2));
При $x = 1 \quad \Rightarrow \quad 1 \cdot 1 = 1$ (аксіома (5.3)); ■

2) Комутативні закони:

$$\begin{cases} x \vee y = y \vee x \\ x \cdot y = y \cdot x \end{cases} \qquad (4.7)$$

3) Асоціативні закони:

$$\begin{cases} (x \vee y) \vee z = x \vee (y \vee z) \\ (x \cdot y) \cdot z = x \cdot (y \cdot z) \end{cases} \qquad (4.8)$$

4) Дистрибутивні закони:

$$\begin{cases} x \cdot (y \vee z) = x \cdot y \vee x \cdot z \\ x \vee y \cdot z = (x \vee y) \cdot (x \vee z) \end{cases} \qquad (4.9)$$

5) Закони заперечення:

$$\begin{cases} x \vee \bar{x} = 1 \\ x \cdot \bar{x} = 0 \end{cases}$$
$$\begin{cases} 0 \vee x = x \\ 1 \cdot x = x \end{cases} \qquad (4.10)$$
$$\begin{cases} 1 \vee x = 1 \\ 0 \cdot x = 0 \end{cases}$$

6) Закони двоїстості (закони де Моргана):

$$\begin{cases} \overline{x \vee y} = \bar{x} \cdot \bar{y} \\ \overline{x \cdot y} = \bar{x} \vee \bar{y} \end{cases} \qquad (4.11)$$

7) Закони подвійного заперечення:

$$\overline{(\bar{x})} = \bar{\bar{x}} = x \qquad (4.12)$$

8) Закони поглинання (абсорбції):

$$\begin{cases} x \vee x \cdot y = x \\ x(x \vee y) = x \end{cases} \qquad (4.13)$$

9) Операції склеювання:

$$\begin{cases} x \cdot y \vee x \cdot \bar{y} = x \\ (x \vee y) \cdot (x \vee \bar{y}) = x \end{cases} \qquad (4.14)$$

10) Закони узагальненого склеювання:

$$\begin{cases} x \cdot y \vee \bar{x} \cdot z \vee y \cdot z = x \cdot y \vee \bar{x} \cdot z \\ (x \vee y) \cdot (\bar{x} \vee z) \cdot (y \vee z) = (x \vee y) \cdot (\bar{x} \vee z) \end{cases}$$
$$\begin{cases} x \vee \bar{x} \cdot y = x \vee y \\ x \cdot (\bar{x} \vee y) = x \cdot y \end{cases} \qquad (4.15)$$

Запис парами виконано для того, щоб показати, що ці пари пов’язані принципом двоїстості – тобто із одного співвідношення даної пари можна отримати друге, якщо взаємно замінити операції диз’юнкції та кон’юнкції, а також значення «0» та «1», якщо вони наявні в формулі. Теорема (4.12) – закон подвійного заперечення – самодвоїста, тобто вона не змінюється за принципом двоїстості (відсутні елементи «0» та «1», операції диз’юнкції та кон’юнкції).

Алгебра логіки тісно пов'язана з теорією множин. Аналогами операцій диз'юнкції, кон'юнкції та заперечення в теорії множин є об'єднання, перетин та доповнення:

Операції алгебри логіки	**Операції теорії множин**
Диз'юнкція $\vee$	Об'єднання $\cup$
Кон'юнкція $\wedge$	Перетин $\cap$
Заперечення $\neg$	Доповнення $\neg$

4.3. Властивості операції «сума по модулю 2» (Виключне АБО)

Окрім основних операцій алгебри логіки – кон'юнкції, диз'юнкції та заперечення буде доцільно використовувати додаткові операції І-НІ, АБО-НІ та Виключне АБО. Найбільш важливі співвідношення для перших двох операцій задаються законом двоїстості де Моргана (4.11). Операцію «Виключне АБО» розглянемо більш детально [1].

Операція «Виключне АБО» (сума по модулю 2) визначається наступними співвідношеннями:

$$\begin{aligned} x \oplus y &= \bar{x} \cdot y \vee x \cdot \bar{y}, \\ x \oplus y &= (\bar{x} \vee \bar{y}) \cdot (x \vee y). \end{aligned} \qquad (4.16)$$

Нескладно переконатися, що для суми по модулю 2 також виконується наступне співвідношення:

$$x \oplus y = \bar{x} \oplus \bar{y} \qquad (4.17)$$

Основні властивості операції сума по модулю 2.

1) Комутативність:

$$x \oplus y = y \oplus x$$

2) Асоціативність:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z$$

3) Дистрибутивність відносно операції кон'юнкції:

$$x \cdot (y \oplus z) = x \cdot y \oplus x \cdot z$$

Зауваження. Відносно операції диз'юнкції операція суми по модулю 2 *не є дистрибутивною*.

Для суми по модулю 2 справедливі тотожності:

1) $\begin{cases} x \oplus 0 = x \\ x \oplus 1 = \bar{x} \end{cases} \qquad \begin{cases} x \oplus x = 0 \\ x \oplus \bar{x} = 1 \end{cases}$

2) $\overline{x \oplus y} = \bar{x} \cdot \bar{y} \vee x \cdot y = (\bar{x} \vee y) \cdot (x \vee \bar{y}) = \bar{x} \oplus y = x \oplus \bar{y}$

4.4. Пріоритетність операцій

У алгебрі логіки, а отже і при записі логічних виразів прийнято наступну пріоритетність операцій: якщо в логічний вираз входять операції диз'юнкції та кон'юнкції, то кон'юнкція має вищий пріоритет виконання (виконується першою), а диз'юнкція – нижчий пріоритет [1]. Однак порядок можна змінювати якщо використовувати дужки. Наприклад:

1) $(x \cdot y) \vee (\overline{x} \cdot z) = x \cdot y \vee \overline{x} \cdot z$ – тут дужки можна пропустити, оскільки порядок дій не зміниться;
2) $(x \vee y) \cdot (\overline{x} \vee z) \neq x \vee y \cdot \overline{x} \vee z$ – тут дужки вилучати не можна, оскільки при цьому зміниться порядок дій.

Якщо додатково в розгляд ввести операцію суми по модулю 2, то можна показати, що для того, щоб зберегти правильність результату виконання обчислень, її пріоритет має бути вищим за пріоритет диз'юнкції і нижчим за пріоритет кон'юнкції, тобто:

Пріоритет *кон'юнкції*	>	Пріоритет *суми по модулю* 2	>	Пріоритет *диз'юнкції*

Операція заперечення є одномісною, тому *застосовується до аргумента в першу чергу*.

Домовленість про позначення

Надалі для запису кон'юнкцій, диз'юнкцій, заперечень та сум по модулю 2 багатьох змінних використовуватимемо наступні скорочені позначення:

$$\prod_{i=1}^{n} x_i = x_1 \cdot x_2 \cdot x_3 \cdot \ldots \cdot x_n,$$

$$\bigvee_{i=1}^{n} x_i = x_1 \vee x_2 \vee x_3 \vee \ldots \vee x_n,$$

$$\bigoplus_{i=1}^{n} x_i = x_1 \oplus x_2 \oplus x_3 \oplus \ldots \oplus x_n,$$

$$\overline{x \vee y \cdot z} \oplus k = \overline{(x \vee y \cdot z)} \oplus k = \neg(x \vee y \cdot z) \oplus k\,.$$

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Які логічні функції визначаються аксіомами алгебри логіки?
2. Наведіть основні закони та теореми алгебри логіки.
3. Якими способами зазвичай доводяться теореми, закони та тотожності алгебри логіки?
4. Що стверджує принцип двоїстості?
5. Що стверджують закони двоїстості де Моргана?
6. Наведіть основні властивості операції «сума по модулю 2». Чи є дана операція дистрибутивною по відношенню до диз'юнкції?
7. Яка пріоритетність виконання операцій прийнята у алгебрі логіки?
8. Доведіть твердження (4.17).
9. Користуючись основними теоремами і законами (4.1)-(4.12) доведіть закони поглинання (4.13) та операції склеювання (4.14).
10. Користуючись аксіомами, теоремами і законами алгебри логіки спростіть вирази:
    а) $x \oplus 0 \vee y \cdot 1 \cdot x \vee x \oplus \overline{x}$
    б) $x \cdot y \cdot z \vee \overline{x} \cdot 1 \vee 0 \oplus (x \cdot y \cdot \overline{z})$
    в) $(\overline{x} \vee y)(x \vee y) \vee y \cdot (x \vee y)$
    г) $x \vee x \cdot 1 \cdot \overline{y} \vee 1 \oplus x$
    д) $\overline{x} \cdot y \oplus (x \vee \overline{y})$

5. ТЕОРЕМИ РОЗКЛАДУ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Теореми розкладу [1, 3, 7] дозволяють записувати перемикальні функції в формі залежностей з явно виділеними змінними (вхідними сигналами). Як буде видно далі, розклад функцій відіграє важливу роль під час синтезу деяких класів цифрових пристроїв, наприклад, мультиплексорів.

5.1. Теорема розкладу Шеннона

Теорема Шеннона: будь-яку функцію $f(\nu)$ можна розкласти за змінною x_p у наступному вигляді [1, 3]:

$$f(x_n,\ldots,x_p,\ldots,x_1) = \bar{x}_p \cdot f(x_n,\ldots,0,\ldots,x_1) \vee x_p \cdot f(x_n,\ldots,1,\ldots,x_1) \tag{5.1}$$

Доведення:

1) Нехай $x_p = 0$, тоді:

$$f(x_n,\ldots,x_p,\ldots,x_1) = \overline{0} \cdot f(x_n,\ldots,0,\ldots,x_1) \vee 0 \cdot f(x_n,\ldots,1,\ldots,x_1) =$$
$$= 1 \cdot f(x_n,\ldots,0,\ldots,x_1) \vee 0 = f(x_n,\ldots,0,\ldots,x_1) \equiv f(x_n,\ldots,x_p,\ldots,x_1),$$

тобто при $x_p = 0$ отримуємо явну тотожність, а теорема справедлива незалежно від значень інших змінних.

2) Нехай $x_p = 1$, тоді:

$$f(x_n,\ldots,x_p,\ldots,x_1) = \overline{1} \cdot f(x_n,\ldots,0,\ldots,x_1) \vee 1 \cdot f(x_n,\ldots,1,\ldots,x_1) =$$
$$= 0 \vee 1 \cdot f(x_n,\ldots,1,\ldots,x_1) = f(x_n,\ldots,1,\ldots,x_1) \equiv f(x_n,\ldots,x_p,\ldots,x_1),$$

тобто при $x_p = 1$ отримуємо явну тотожність, а теорема справедлива незалежно від значень інших змінних.

Таким чином теорема справедлива для будь-яких значень змінних. ■

Зауваження. Використовуючи принцип суперпозиції (логічні функції можуть виступати аргументами інших, більш складних функцій), можна записати:

$$\begin{cases} \overline{f_1(\nu)} \cdot f_2(\nu, f_1(\nu)) = \overline{f_1(\nu)} \cdot f_2(\nu, 0) \\ f_1(\nu) \cdot f_2(\nu, f_1(\nu)) = f_1(\nu) \cdot f_2(\nu, 1) \end{cases} \tag{5.2}$$

Співвідношення (5.2) може бути використане для спрощення деяких логічних функцій.

Приклади використання розкладу Шеннона

1) Теорема Шеннона може використовуватись для спрощення логічних функцій, зокрема тих, що містять операцію сума по модулю 2:

$$\overline{x_2x_1 \oplus (x_3 \vee \bar{x}_1) \oplus x_3x_1} \oplus (x_2 \vee \bar{x}_1) =$$

Виконаємо розклад по x_1:

$$= \bar{x}_1[\overline{x_2 0 \oplus (x_3 \vee \bar{0}) \oplus x_3 0} \oplus (x_2 \vee \bar{0})] \vee x_1[\overline{x_2 1 \oplus (x_3 \vee \bar{1}) \oplus x_3 1} \oplus (x_2 \vee \bar{1})] =$$

$$= \bar{x}_1[\overline{0 \oplus 1 \oplus 0} \oplus 1] \vee x_1[\overline{x_2 \oplus \underbrace{x_3 \oplus x_3}_{0}} \oplus x_2] = \bar{x}_1 \cdot 1 \vee x_1 \cdot 1 = \bar{x}_1 \vee x_1 = 1$$

2) Користуючись виразом (5.2) спростимо функцію:

$$\overline{\bar{x}_1x_2 \oplus (x_1 \vee x_3)} \cdot [\bar{x}_2\bar{x}_3 \oplus \overline{\bar{x}_1x_2 \oplus (x_1 \vee x_3)} \vee x_2\bar{x}_3]$$

Позначимо $f = \bar{x}_1x_2 \oplus (x_1 \vee x_3)$, тоді:

$$\overline{f} \cdot [\bar{x}_2\bar{x}_3 \oplus \overline{f} \vee x_2\bar{x}_3] = \overline{f} \cdot [\underbrace{\bar{x}_2\bar{x}_3 \oplus \bar{0}}_{\bar{x}_2\bar{x}_3} \vee x_2\bar{x}_3] = \overline{f} \cdot [(x_2 \vee x_3) \vee x_2\bar{x}_3] =$$

$$= \overline{f} \cdot [x_2(\underbrace{1 \vee \bar{x}_3}_{1}) \vee x_3] = \overline{f} \cdot (x_2 \vee x_3)$$

5.2. Мультиплексні функції. Мультплексор

Змінні, що є аргументами перемикальних функцій, при реалізації деяких пристроїв поділять на два типи [1]:

- y_q – інформаційні;
- x_p – керуючі;

Функції $f(y_1, y_0, x_1)$, $f(y_3, y_2, y_1, y_0, x_2, x_1)$ та в загальному вигляді $f(y_{2^n-1}, \ldots, y_1, y_0, x_n, \ldots, x_2, x_1)$, які за змінними x_p мають розклад:

$f(y_1, y_0, x_1) = y_0\bar{x}_1 \vee y_1x_1$

$f(y_3, y_2, y_1, y_0, x_2, x_1) = y_0\bar{x}_2\bar{x}_1 \vee y_1\bar{x}_2x_1 \vee y_2x_2\bar{x}_1 \vee y_3x_2x_1$

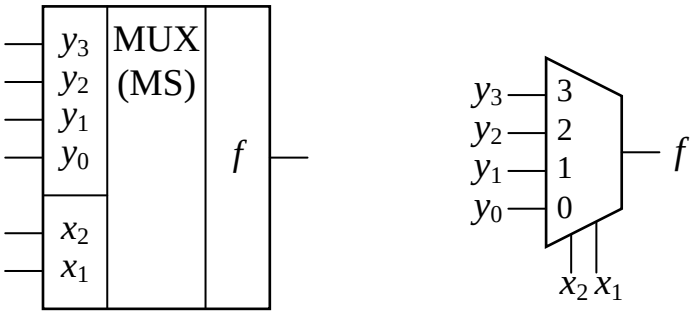
$f(y_{2^n-1}, \ldots, y_1, y_0, x_n, \ldots, x_2, x_1) = y_0\bar{x}_n \ldots \bar{x}_2\bar{x}_1 \vee y_1\bar{x}_n \ldots \bar{x}_2x_1 \vee \ldots \vee y_{2^n-1}x_n \ldots x_2x_1$

називаються *мультиплексними функціями*.

Такі функції є комутаторами сигналів і їх також називають *мультиплекторами*. Вони характеризуються тим, що для кожної комбінації керуючих сигналів, функція приймає значення одного з інформаційних сигналів, номер якого дорівнює двійковій комбінації

(числу) керуючих сигналів [1, 3, 4, 6]. Наприклад, для функції $f(y_3, y_2, y_1, y_0, x_2, x_1)$, розглянутої вище, при комбінації керуючих сигналів $(x_2, x_1) = (0, 0)$, функція f прийматиме значення y_0, аналогічно, якщо $(x_2, x_1) = (0, 1)$, то $f = y_1$, якщо $(x_2, x_1) = (1, 0)$, то $f = y_2$, а якщо $(x_2, x_1) = (1, 1)$, то $f = y_3$.

Умовне графічне позначення мультиплексорів показано на рис. 5.1.



ГОСТ ANSI

Рисунок 5.1 – Приклади умовного графічного позначення мультиплексора з двома керуючими входами

Інверсні мультиплексні функції

Функції мають вигляд:

$$\overline{f(y_1, y_0, x_1)} = \bar{y}_0\bar{x}_1 \vee \bar{y}_1 x_1$$

$$\overline{f(y_3, y_2, y_1, y_0, x_2, x_1)} = \bar{y}_0\bar{x}_2\bar{x}_1 \vee \bar{y}_1\bar{x}_2 x_1 \vee \bar{y}_2 x_2\bar{x}_1 \vee \bar{y}_3 x_2 x_1$$

$$\overline{f(y_{2^n-1}, \ldots, y_1, y_0, x_n, \ldots, x_2, x_1)} = \bar{y}_0\bar{x}_n \ldots \bar{x}_2\bar{x}_1 \vee \bar{y}_1\bar{x}_n \ldots \bar{x}_2 x_1 \vee \ldots \vee \bar{y}_{2^n-1} x_n \ldots x_2 x_1$$

З'ясування призначення деяких логічних функцій

Розклад Шеннона може застосовуватись для з'ясування практичного призначення деяких перемикальних функцій. Розглянемо функцію $f(y_2, y_1, x_2, x_1) = y_1x_1 \vee y_2x_2 \vee y_1y_2$, розклавши її за сигналами керування x_1 та x_2:

$$\begin{aligned}
f(y_2, y_1, x_2, x_1) &= \bar{x}_1(y_1 0 \vee y_2x_2 \vee y_1y_2) \vee x_1(y_1 1 \vee y_2x_2 \vee y_1y_2) = \\
&= \bar{x}_1(y_2x_2 \vee y_1y_2) \vee x_1(y_1 \vee y_2x_2) = \\
&= \bar{x}_2[\bar{x}_1(y_2 0 \vee y_1y_2) \vee x_1(y_1 \vee y_2 0)] \vee x_2[\bar{x}_1(y_2 1 \vee y_1y_2) \vee x_1(y_1 \vee y_2 1)] = \\
&= \bar{x}_2(\bar{x}_1y_1y_2 \vee x_1y_1) \vee x_2[\bar{x}_1y_2 \vee x_1(y_1 \vee y_2)] = \\
&= y_1y_2\bar{x}_2\bar{x}_1 \vee y_1\bar{x}_2x_1 \vee y_2x_2\bar{x}_1 \vee (y_1 \vee y_2)x_2x_1
\end{aligned}$$

Наведена вище функція описує так званий *функціональний комутатор* або *функціональний мультиплексор*, який дозволяє не тільки комутувати вхідні сигнали y_2 та y_1, але і виконувати деякі операції над ними (зокрема, при $(x_2, x_1) = (0, 0) \Rightarrow f = y_2 y_1$, а при $(x_2, x_1) = (1, 1) \Rightarrow f = y_2 \vee y_1$).

Прикладом схеми функціонального комутатора є 4-розрядний функціональний мультиплексор 1561КП4 (CD4019B), який виконує наступні 4 дії:

$$D_j^o = 0 \cdot \bar{x}_2 \bar{x}_1 \vee D_j^{i1} \cdot \bar{x}_2 x_1 \vee D_j^{i2} \cdot x_2 \bar{x}_1 \vee \overline{D_j^{i1} \oplus D_j^{i2}} \cdot x_2 x_1,$$

де x_2 та x_1 – керуючі сигнали; D_j^{i1} та D_j^{i2} – вхідні інформаційні сигнали; D_j^o – вихідні сигнали; $j = 0, 1, 2, 3$.

5.3. Розклад Рида

Розглянемо розклад Шеннона та зробимо позначення [1]:

$$f(x_n,\ldots,x_p,\ldots,x_1) = \bar{x}_p \cdot \underbrace{f(x_n,\ldots,0,\ldots,x_1)}_{g_0} \vee x_p \cdot \underbrace{f(x_n,\ldots,1,\ldots,x_1)}_{g_1} = \\ = \underbrace{\bar{x}_p \cdot g_0}_{x} \vee \underbrace{x_p \cdot g_1}_{y} \qquad (5.3)$$

Неважко впевнитись, що $xy = 0$, оскільки:

$$(\bar{x}_p \cdot g_0)(x_p \cdot g_1) = 0. \qquad (5.4)$$

Водночас, якщо відомо, що виконується наведена вище умова, то операцію $x \vee y$ можна замінити на $x \oplus y$.

Також у цьому можна впевнитись, якщо:

$$x \oplus y \vee \underbrace{x \cdot y}_{0} = \bar{x} \cdot y \vee \underbrace{x \cdot \bar{y} \vee x \cdot y}_{\substack{x \\ \text{відповідно до} \\ \text{операції} \\ \text{склеювання}}} = \underbrace{\bar{x} \cdot y \vee x}_{\substack{\text{закон} \\ \text{узагальненого} \\ \text{склеювання}}} = x \vee y$$

Враховуючи зроблені позначення у розкладі (5.3) та вираз (5.4) отримаємо:

$$f(x_n,\ldots,x_p,\ldots,x_1) = \bar{x}_p \cdot g_0 \vee x_p \cdot g_1 = \underbrace{(1 \oplus x_p)}_{\bar{x}_p} \cdot g_0 \oplus x_p \cdot g_1 = g_0 \oplus x_p g_0 \oplus x_p g_1 = \\ = g_0 \oplus (g_0 \oplus g_1) \cdot x_p$$

Отриманий вираз називають розкладом Рида. Даний розклад показує, що перемикальні функції можуть бути представлені у вигляді поліномів:

$$f(x_n,...,x_p,...,x_1) = f(x_n,...,0,...,x_1) \oplus [f(x_n,...,0,...,x_1) \oplus f(x_n,...,1,...,x_1)] \cdot x_p$$

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Що стверджує теорема Шеннона? Як її довести?
2. Доведіть справедливість виразів (5.2).
3. Які функції називаються мультиплексними? Який цифровий пристрій можна реалізувати на основі цих функцій? Поясніть для чого цей пристрій може застосовуватись.
4. Для чого застосовується розклад Рида?
5. Користуючись теоремою Шеннона виконайте розклад функції за змінною x_3:

$$f(\nu) = x_3x_1 \vee x_4x_2 \vee x_4x_3.$$

6. Користуючись теоремою Шеннона спростіть вираз:

$$\overline{x \oplus z} \oplus (x \vee z).$$

7. Виконайте розклад Рида функції:

$$f(\nu) = x_2x_1 \vee \bar{x}_2 \oplus x_1.$$

6. ПЕРВИННІ ТЕРМИ. МІНТЕРМИ І МАКСТЕРМИ

Нормальні (або канонічні) форми двійкових функцій отримують на основі *термів* [1].

В загальному випадку, термами називають елементарні логічні конструкції, на основі яких формуються логічні функції. Використання термів відіграє важливу роль при аналітичному описі функціонування перемикальних схем.

Для того, щоб формалізувати методики отримання складних логічних функцій використовують поняття «*первинний терм*».

6.1. Первинні (елементарні) терми

Первинними термами називаються логічні змінні *x* та їх інверсії $\bar{x}$. При побудові логічних функцій зручніше користуватися позначеннями термів, що об'єднують в собі і змінні, і їх інверсії (такі терми також відносять до первинних):

$$x_p^{e_p} = \bar{e}_p \bar{x}_p \vee e_p x_p = \overline{e_p \oplus x_p},$$

де, e_p = 0 / 1.

Первинний терм $x_p^{e_p}$ володіє наступною властивістю:

$$x_p^{e_p} = \begin{cases} \bar{x}_p, & \text{якщо } e_p = 0, \\ x_p, & \text{якщо } e_p = 1 \end{cases}$$

Звідси видно, що два первинні терми $x_p^{e_p}$ та $x_p^{e'_p}$ рівні один одному, якщо $e_p = e'_p$. Якщо $e_p \neq e'_p$, то $x_p^{e_p} = \overline{x_p^{e'_p}}$. Із цих співвідношень випливає:

1) $x_p^1 = \bar{x}_p^0 = x_p, \qquad x_p^0 = \bar{x}_p^1 = \bar{x}_p$

2) $\overline{x_p^{e_p}} = x_p^{\bar{e}_p} = \bar{x}_p^{e_p}$

3) $x_p^{e_p} \cdot x_p^{\bar{e}_p} \equiv 0, \qquad x_p^{e_p} \vee x_p^{\bar{e}_p} \equiv 1$

4) $x_p^{e_p} = \begin{cases} 0, & \text{якщо } x_p = \bar{e}_p \\ 1, & \text{якщо } x_p = e_p \end{cases}$

Використовуючи первинні терми $x_p^{e_p}$, можна отримати ряд канонічних форм логічних функцій. Розглянемо деякі з них.

6.2. Мінтерми

Мінтермом (*конституентою одиниці*) називають функцію, яка приймає *одиничне* значення на одному з усіх можливих наборів аргументів та нульове значення при всіх інших наборах аргументів [1, 3].

Математичним виразом для мінтерма (або мінімального терма) є кон'юнкція n первинних термів:

$$K_i(\nu) = x_n^{e_n} \cdot \ldots \cdot x_1^{e_1} = \prod_{p=1}^{n} x_p^{e_p} = C_i^1, \qquad (6.1)$$

де $\nu = \{x_n,\ldots,x_1\}$, $e_p = 0 / 1$, $i_{10} = (e_n \ldots e_1)_2$, C_i^1 – позначення конституенти одиниці. Мінтерми є невиродженими функціями, тобто функціями всіх змінних.

Оскільки, існує 2^n n-розрядних двійкових чисел $i = 0, 2,\ldots,2^n - 1$, то відповідно, існує 2^n різних мінтерма n змінних. Наприклад, мінтерму 3-х змінних $x_3^{e_3} x_2^{e_2} x_1^{e_1}$ відповідає $2^3 = 8$ різних функцій:

$$x_3^{e_3} x_2^{e_2} x_1^{e_1} = \begin{cases} K_0(\nu) = \bar{x}_3\bar{x}_2\bar{x}_1, & \text{при} \quad 0 \quad 0 \quad 0 \\ K_1(\nu) = \bar{x}_3\bar{x}_2 x_1, & \text{при} \quad 0 \quad 0 \quad 1 \\ K_2(\nu) = \bar{x}_3 x_2\bar{x}_1, & \text{при} \quad 0 \quad 1 \quad 0 \\ K_3(\nu) = \bar{x}_3 x_2 x_1, & \text{при} \quad 0 \quad 1 \quad 1 \\ K_4(\nu) = x_3\bar{x}_2\bar{x}_1, & \text{при} \quad 1 \quad 0 \quad 0 \\ K_5(\nu) = x_3\bar{x}_2 x_1, & \text{при} \quad 1 \quad 0 \quad 1 \\ K_6(\nu) = x_3 x_2\bar{x}_1, & \text{при} \quad 1 \quad 1 \quad 0 \\ K_7(\nu) = x_3 x_2 x_1, & \text{при} \quad 1 \quad 1 \quad 1 \end{cases} \qquad \begin{matrix} e_3 & e_2 & e_1 \end{matrix}$$

На основі наведених вище функцій, що відповідають мінтерму 3-х змінних можна встановити загальні властивості мінтермів. Зокрема:

1) $K_i(\nu) = \begin{cases} 1, & \text{якщо } \nu = \nu_i \\ 0, & \text{якщо } \nu \neq \nu_i \end{cases}$

2) $K_i(\nu) \cdot K_j(\nu) = 0$, якщо $i \neq j$, в будь-якій точці ν області визначення, оскільки в цій точці рівним одиниці може виявитись лише один з мінтермів (або ні одного).

3) $\bigvee_{i=0}^{2^n-1} K_i(\nu) \equiv 1$, оскільки хоча б один з мінтермів в точці ν області визначення дорівнює 1.

6.3. Макстерми

Макстермом (*конституентою нуля*) називають функцію, яка приймає *нульове* значення на одному з усіх можливих наборів аргументів та одиничне значення на всіх інших наборах аргументів [1, 3].

Математичним виразом для макстерма (або максимального терма) є диз'юнкція n первинних термів:

$$M_i(\nu) = x_n^{\bar{e}_n} \vee \ldots \vee x_1^{\bar{e}_1} = \bigvee_{p=1}^{n} x_p^{\bar{e}_p} = C_i^0, \qquad (6.2)$$

де $\nu = \{x_n,\ldots,x_1\}$, $e_p = 0 / 1$, $i_{10} = (e_n \ldots e_1)_2$, C_i^0 – позначення конституенти нуля. Макстерми, як і мінтерми є невиродженими функціями, тобто функціями всіх змінних.

Як і у випадку з мінтермом, для n змінних існує 2^n різних макстерма. Зокрема, макстерму 3-х змінних $x_3^{\bar{e}_3} \vee x_2^{\bar{e}_2} \vee x_1^{\bar{e}_1}$ відповідає $2^3 = 8$ різних функцій:

$$x_3^{\bar{e}_3} \vee x_2^{\bar{e}_2} \vee x_1^{\bar{e}_1} = \begin{cases} M_0(\nu) = x_3 \vee x_2 \vee x_1, & \text{при} \quad e_3\ e_2\ e_1 = 0\ 0\ 0 \\ M_1(\nu) = x_3 \vee x_2 \vee \bar{x}_1, & \text{при} \quad 0\ 0\ 1 \\ M_2(\nu) = x_3 \vee \bar{x}_2 \vee x_1, & \text{при} \quad 0\ 1\ 0 \\ M_3(\nu) = x_3 \vee \bar{x}_2 \vee \bar{x}_1, & \text{при} \quad 0\ 1\ 1 \\ M_4(\nu) = \bar{x}_3 \vee x_2 \vee x_1, & \text{при} \quad 1\ 0\ 0 \\ M_5(\nu) = \bar{x}_3 \vee x_2 \vee \bar{x}_1, & \text{при} \quad 1\ 0\ 1 \\ M_6(\nu) = \bar{x}_3 \vee \bar{x}_2 \vee x_1, & \text{при} \quad 1\ 1\ 0 \\ M_7(\nu) = \bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1, & \text{при} \quad 1\ 1\ 1 \end{cases}$$

Загальні властивості макстермів є наступними:

1) $M_i(\nu) = \begin{cases} 0, & \text{якщо } \nu = \nu_i \\ 1, & \text{якщо } \nu \neq \nu_i \end{cases}$

2) $M_i(\nu) \vee M_j(\nu) = 1$, якщо $i \neq j$, в будь-якій точці ν області визначення, оскільки в цій точці рівним нулю може виявитись лише один з макстермів (або ні одного).

3) $\prod_{i=0}^{2^n-1} M_i(\nu) \equiv 0$, оскільки хоча б один з макстермів в точці ν області визначення дорівнює 0.

Зв’язок між мінтермами та макстермами:

$$M_i(\nu) = \overline{K_i(\nu)} = \overline{\prod_{p=1}^{n} x_p^{e_p}} = \bigvee_{p=1}^{n} \overline{x_p^{e_p}} = \bigvee_{p=1}^{n} x_p^{\bar{e}_p}. \qquad (6.3)$$

Приклад

Записати всі мінтерми та макстерми функції двох змінних та визначити їх значення.

Загальна форма запису мінтермів 2-х змінних:

$K_i(\nu) = x_2^{e_2} x_1^{e_1}$

$$K_0(\nu) = x_2^0 x_1^0 = \bar{x}_2 \bar{x}_1 = 1,$$
$$K_1(\nu) = x_2^0 x_1^1 = \bar{x}_2 x_1 = 1,$$
$$K_2(\nu) = x_2^1 x_1^0 = x_2 \bar{x}_1 = 1,$$
$$K_3(\nu) = x_2^1 x_1^1 = x_2 x_1 = 1,$$

де $\nu = (x_2, x_1)$, $i_{10} = (e_2 e_1)_2$

Загальна форма запису макстермів 2-х змінних:

$M_i(\nu) = x_2^{\bar{e}_2} \vee x_1^{\bar{e}_1}$

$$M_0(\nu) = x_2^{\bar{0}} \vee x_1^{\bar{0}} = x_2 \vee x_1 = 0,$$
$$M_1(\nu) = x_2^{\bar{0}} \vee x_1^{\bar{1}} = x_2 \vee \bar{x}_1 = 0,$$
$$M_2(\nu) = x_2^{\bar{1}} \vee x_1^{\bar{0}} = \bar{x}_2 \vee x_1 = 0,$$
$$M_3(\nu) = x_2^{\bar{1}} \vee x_1^{\bar{1}} = \bar{x}_2 \vee \bar{x}_1 = 0,$$

де $\nu = (x_2, x_1)$, $i_{10} = (e_2 e_1)_2$

Значення всіх мінтермів та макстермів в області визначення функції:

			Мінтерми				Макстерми			
i	x_2	x_1	K_3	K_2	K_1	K_0	M_3	M_2	M_1	M_0
0	0	0	0	0	0	1	1	1	1	0
1	0	1	0	0	1	0	1	1	0	1
2	1	0	0	1	0	0	1	0	1	1
3	1	1	1	0	0	0	0	1	1	1

6.4. Дешифратор

Пристрої, що реалізують всі 2^n мінтерма, називаються *повними дешифраторами* [1, 3, 6]. Пристрої, що реалізують 2^n макстерма є *повними дешифраторами з інверсними виходами*. Дані пристрої використовуються

для ввімкнення-вимкнення інших пристроїв, оскільки в кожен момент часу тільки один вихідний сигнал дешифратора дорівнює 1 (або 0).

Позначення дешираторів на схемах показано нижче (рис.6.1).

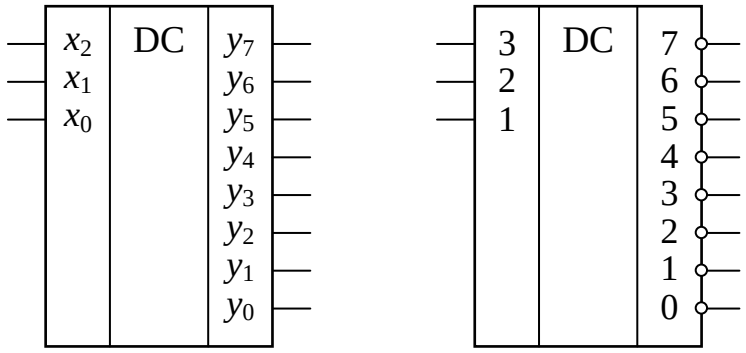


Рисунок 6.1 – Позначення дешифраторів згідно ГОСТ. Варіант наведений справа є позначенням дешифратора з інверсними виходами

Дешифратор називається *неповним*, якщо він реалізує не всі 2^n мінтерма (макстерма). Якщо дешифратор реалізує тільки один мінтерм (макстерм), то його також називають *детектором стану*. Детектори стану використовуються для виявлення однієї визначеної комбінації значень вхідних сигналів.

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Дайте визначення первинного терма. Наведіть властивості первинних термів.
2. Дайте формальне визначення мінімального терма (мінтерма).
3. Дайте формальне визначення максимального терма (макстерма).
4. Запишіть всі можливі мінтерми та макстерми чотирьох змінних.
5. Знайдіть аналітичний вираз логічної функції, якщо відомо, що вона приймає *одиничне* значення в одній єдиній точці:
   а) 1011
   б) 11001
   в) 10100011
6. Знайдіть аналітичний вираз логічної функції, якщо відомо, що вона приймає *нульове* значення в одній єдиній точці:
   а) 0101
   б) 10110
   в) 10010110
7. Який пристрій називається дешифратором? Що таке неповний дешифратор та детектор стану?

7. НОРМАЛЬНІ КАНОНІЧНІ ФОРМИ ДВІЙКОВИХ ФУНКЦІЙ

Форми представлення перемикальних функцій, отриманні завдяки суперпозиції їх мінтермів та макстермів називаються *досконалою диз'юнктивною* та *досконалою кон'юнктивною нормальними формами* (скорочено ДДНФ та ДКНФ) [1, 3, 7].

7.1. Досконала диз'юнктивна нормальна форма (ДДНФ)

Досконала диз'юнктивна нормальна форма (ДДНФ) є диз'юнкцією мінтермів, які відповідають таким наборам аргументів, на яких функція, що розглядається приймає одиничне значення.

Представлення будь-якої функції n змінних у формі ДДНФ можна отримати шляхом застосування розкладу Шеннона n раз до цієї функції, тобто завдяки її розкладу за всіма змінними. Проілюструємо це на прикладі функції 2-х змінних $f(\nu) = f(x_2, x_1)$.

Спочатку розкладемо дану функцію по аргументу x_2:

$$f(x_2, x_1) = \bar{x}_2 f(0, x_1) \vee x_2 f(1, x_1)$$

Далі розкладемо кожну з функцій $f(0, x_1)$ та $f(1, x_1)$ за x_1:

$$\begin{aligned} f(x_2, x_1) &= \bar{x}_2[\bar{x}_1 f(0,0) \vee x_1 f(0,1)] \vee x_2[\bar{x}_1 f(1,0) \vee x_1 f(1,1)] = \\ &= \bar{x}_2\bar{x}_1 f(0,0) \vee \bar{x}_2 x_1 f(0,1) \vee x_2\bar{x}_1 f(1,0) \vee x_2 x_1 f(1,1) = \\ &= x_2^0 x_1^0 f(0,0) \vee x_2^0 x_1^1 f(0,1) \vee x_2^1 x_1^0 f(1,0) \vee x_2^1 x_1^1 f(1,1) = \\ &= \bigvee_{i=0}^{3} f(\nu_i) \cdot K_i(\nu) \end{aligned}$$

Отриманий розклад і є нічим іншим, як ДДНФ будь-якої функції 2-х змінних $f(x_2, x_1)$. Термін «досконала» тут означає, що всі члени розкладу мають однакову розмірність, а термін «нормальна форма» – що у виразах, які задають функцію, послідовно виконується не більше, ніж дві базові операції алгебри логіки без урахування операції заперечення.

У випадку n змінних ДДНФ приймає вигляд:

$$f(\nu) = \bigvee_{i=0}^{2^n-1} f(\nu_i) \cdot K_i(\nu), \qquad (7.1)$$

де $\nu = (x_n, \ldots, x_p, \ldots, x_1)$; $K_i(\nu)$ – мінтерми.

Зауважимо, що розклад виду (7.1) еквівалентний виразу для мультиплексної функції.

Розклад (7.1) є диз'юнкцією 2^n елементів виду:

$$K_i(\nu)\cdot f(\nu_i)$$

Деякі елементи розкладу (7.1) можуть дорівнювати нулю, оскільки якщо $f(\nu_i) = 0$, то $K_i(\nu)\cdot f(\nu_i) = 0$, а якщо $f(\nu_i) = 1$, то $K_i(\nu)\cdot f(\nu_i) = K_i(\nu)$. Отже ДДНФ функції $f(\nu)$ можна представити як:

$$f(\nu) = \bigvee_i K_i(\nu)_1, \qquad (7.2)$$

де i – номери точок та $K_i(\nu)_1$ – мінтерми, що їм відповідають, в яких функція $f(\nu)$ дорівнює 1. Таким чином, ДДНФ функції n змінних є диз'юнкцією деякого числа $k \le 2^n$ мінтермів.

Зауваження. ДДНФ повністю невизначеної функції n змінних має вигляд:

$$\hbar = \bigvee_{i=0}^{2^n-1} c_i \cdot K_i(\nu),$$

де c_i – невизначене значення функції, c_i = {0 або 1}.

7.2. Досконала кон'юнктивна нормальна форма (ДКНФ)

Досконалу кон'юнктивну нормальну форму (ДКНФ) функції n змінних можна знайти завдяки запису ДДНФ інверсної функції $\overline{f(\nu)}$. Нагадаємо, що інверсна функція $\overline{f(\nu)}$ рівна 0 (1) у всіх точках n-мірного простору аргументів, в яких пряма функція $f(\nu)$ дорівнює 1 (0). Отже, користуючись (7.1) можна записати:

$$\overline{f(\nu)} = \bigvee_{i=0}^{2^n-1} \overline{f(\nu_i)}\cdot K_i(\nu)$$

На основі закону двоїстості вираз для функції $f(\nu)$ (інверсної по відношенню до функції $\overline{f(\nu)}$) матиме вигляд:

$$f(\nu) = \overline{\bigvee_{i=0}^{2^n-1} \overline{f(\nu_i)}\cdot K_i(\nu)} = \prod_{i=0}^{2^n-1} \overline{\overline{f(\nu_i)}\cdot K_i(\nu)} = \prod_{i=0}^{2^n-1} [f(\nu_i) \vee \overline{K_i(\nu)}],$$

враховуючи, що інверсними до мінтермів $K_i(\nu)$ функціями є макстерми $M_i(\nu)$, тобто $\overline{K_i(\nu)} = M_i(\nu)$, то:

$$f(\nu) = \prod_{i=0}^{2^n-1} [f(\nu_i) \vee M_i(\nu)]. \qquad (7.3)$$

Дана форма запису функції $f(\nu)$ називається ДКНФ. Оскільки в певних точках області визначення функція $f(\nu_i)$ може перетворюватись на 0, то в цих точках $f(\nu_i) \vee M_i(\nu) = M_i(\nu)$, ті ж точки, в яких $f(\nu_i) = 1$ взагалі

можна взагалі виключити із запису (7.3), оскільки $1 \vee M_i(\nu) = 1$. Таким чином, ДКНФ можна представити більш компактним записом:

$$f(\nu) = \prod_i M_i(\nu)_0 , \qquad (7.4)$$

де i – номери точок та $M_i(\nu)_0$ – макстерми, що їм відповідають, в яких функція $f(\nu)$ перетворюється на 0. Таким чином, ДКНФ функції n змінних є кон'юнкцією деякого числа $k \le 2^n$ макстермів.

Нагадаємо, що раніше було введено визначення *функціонально-повного базису* – набору двійкових функцій, за допомогою якого можна представляти будь-які інші двійкові функції шляхом композиції (суперпозиції) даного набору. Вище було встановлено, що будь-яку перемикальну (двійкову) функцію можна представити у ДДНФ або ДКНФ, при чому для реалізації вказаних досконалих нормальних форм достатньо мати 3 базові логічні операції – І, АБО та НІ. Таким чином, існування ДДНФ та ДКНФ виступає доведенням раніше наведеного факту, що набір функцій І, АБО та НІ є функціонально-повним базисом.

7.3. Досконалі нормальні форми в базисах І-НІ та АБО-НІ

Крім того, раніше стверджувалось, що набори І-НІ та АБО-НІ також володіють функціональною повнотою. Тепер цей факт можна довести математично.

Для цього застосуємо до ДДНФ функції n змінних (7.1) закони подвійного заперечення та двоїстості:

$$f(\nu) = \overline{\overline{\bigvee_{i=0}^{2^n-1} f(\nu_i) \cdot K_i(\nu)}} = \overline{\prod_{i=0}^{2^n-1} \overline{f(\nu_i) \cdot K_i(\nu)}} = \overline{\prod_{i=0}^{2^n-1} \overline{K_i(\nu)_1}} . \qquad (7.5)$$

Враховуючи, що $K_i(\nu)$ – це добуток первинних термів, то вираз для функції (7.5) реалізується повністю через операцій кон'юнкції (І) та заперечення (НІ). Дане представлення функції називається *досконалою нормальною формою* (*ДНФ*) *в базисі І-НІ*.

Аналогічні перетворення виконаємо над ДКНФ функції n змінних (7.3), також застосувавши закони подвійного заперечення та двоїстості:

$$f(\nu) = \overline{\overline{\prod_{i=0}^{2^n-1} [f(\nu_i) \vee M_i(\nu)]}} = \overline{\bigvee_{i=0}^{2^n-1} \overline{f(\nu_i) \vee M_i(\nu)}} = \overline{\bigvee_{i=0}^{2^n-1} \overline{M_i(\nu)_0}} . \qquad (7.6)$$

Отримане представлення функції називається *досконалою нормальною формою* (*ДНФ*) *в базисі АБО-НІ*, оскільки вона потребує використання операції диз'юнкції (АБО) та заперечення (НІ).

Приклади.

1) Функція 3-х змінних задана таблицею істинності. Представити дану функцію у ДДНФ та ДКНФ.

i	x_3	x_2	x_1	$f(x_3, x_2, x_1)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Із таблиці істинності бачимо, що $\nu = (x_3, x_2, x_1)$ функція $f(\nu_1) = f(\nu_3) = f(\nu_4) = f(\nu_6) = 1$. Тоді за формулою (7.2) можемо записати ДДНФ:

$$\begin{aligned} f(\nu) &= K_1(\nu) \vee K_3(\nu) \vee K_4(\nu) \vee K_6(\nu) = \\ &= x_3^0 x_2^0 x_1^1 \vee x_3^0 x_2^1 x_1^1 \vee x_3^1 x_2^0 x_1^0 \vee x_3^1 x_2^1 x_1^0 = \\ &= \bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 x_2 \bar{x}_1 \end{aligned}$$

Із таблиці також видно, що функція $f(\nu_0) = f(\nu_2) = f(\nu_5) = f(\nu_7) = 0$. Тоді за формулою (7.4) можемо записати ДКНФ:

$$\begin{aligned} f(\nu) &= M_0(\nu) \cdot M_2(\nu) \cdot M_5(\nu) \cdot M_7(\nu) = \\ &= (x_3^{\bar{0}} \vee x_2^{\bar{0}} \vee x_1^{\bar{0}}) \cdot (x_3^{\bar{0}} \vee x_2^{\bar{1}} \vee x_1^{\bar{0}}) \cdot (x_3^{\bar{1}} \vee x_2^{\bar{0}} \vee x_1^{\bar{1}}) \cdot (x_3^{\bar{1}} \vee x_2^{\bar{1}} \vee x_1^{\bar{1}}) = \\ &= (x_3 \vee x_2 \vee x_1) \cdot (x_3 \vee \bar{x}_2 \vee x_1) \cdot (\bar{x}_3 \vee x_2 \vee \bar{x}_1) \cdot (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1) \end{aligned}$$

2) Для перемикальної функції заданої у попередньому прикладі знайти ДНФ в базисах І-НІ та АБО-НІ:

Для отримання досконалої нормальної форми функції в базисі І-НІ скористаємось ДДНФ даної функції та рівнянням (7.5):

$$\begin{aligned} f(\nu) &= \overline{\overline{\bar{x}_3 \bar{x}_2 x_1 \vee \bar{x}_3 x_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 \vee x_3 x_2 \bar{x}_1}} = \\ &= \overline{\overline{\bar{x}_3 \bar{x}_2 x_1} \cdot \overline{\bar{x}_3 x_2 x_1} \cdot \overline{x_3 \bar{x}_2 \bar{x}_1} \cdot \overline{x_3 x_2 \bar{x}_1}} \end{aligned}$$

Для отримання досконалої нормальної форми функції в базисі АБО-НІ скористаємось ДКНФ даної функції та рівнянням (7.6):

$$\begin{aligned} f(\nu) &= \overline{\overline{(x_3 \vee x_2 \vee x_1) \cdot (x_3 \vee \bar{x}_2 \vee x_1) \cdot (\bar{x}_3 \vee x_2 \vee \bar{x}_1) \cdot (\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1)}} = \\ &= \overline{\overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee \bar{x}_2 \vee x_1} \vee \overline{\bar{x}_3 \vee x_2 \vee \bar{x}_1} \vee \overline{\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1}} \end{aligned}$$

7.4. Розклад Рида-Маллера

Якщо відомо, що $x \cdot y = 0$, то справедлива тотожність $x \vee y = x \oplus y$. Застосуємо даний факт до мінтермів: оскільки $K_i(\nu) \cdot K_j(\nu) = 0$, то буде справедливим відношення $K_i(\nu) \vee K_j(\nu) = K_i(\nu) \oplus K_j(\nu)$. Таким чином, представлення функції $f(\nu)$ у ДДНФ можна записати сумою по модулю 2 мінтермів:

$$f(\nu) = \bigvee_{i=0}^{2^n-1} f(\nu_i) \cdot K_i(\nu) = \bigoplus_{i=0}^{2^n-1} f(\nu_i) \cdot K_i(\nu). \qquad (7.7)$$

Знайдений таким шляхом розклад називається розкладом Рида-Маллера та дозволяє записати функцію $f(\nu)$ у вигляді полінома. Аналогічний результат випливає із розглянутого раніше розкладу Рида, який було отримано із розкладу Шеннона.

Розглянемо приклад:

1) Нехай задано функцію 3-х змінних у ДДНФ:

$$f(\nu) = K_0(\nu) \vee K_3(\nu) \vee K_7(\nu) = \bar{x}_3\bar{x}_2\bar{x}_1 \vee \bar{x}_3x_2x_1 \vee x_3x_2x_1$$

Тоді:

$$f(\nu) = \bar{x}_3\bar{x}_2\bar{x}_1 \oplus \bar{x}_3x_2x_1 \oplus x_3x_2x_1 =$$

$$= \underbrace{(1 \oplus x_3)(1 \oplus x_2)}_{1 \oplus x_2 \oplus x_3 \oplus x_2x_3}(1 \oplus x_1) \oplus (1 \oplus x_3)x_2x_1 \oplus x_3x_2x_1 =$$

$$= 1 \oplus x_2 \oplus x_3 \oplus x_2x_3 \oplus x_1 \oplus x_1x_2 \oplus x_1x_3 \oplus x_1x_2x_3 \oplus x_2x_1 \oplus$$

$$\oplus x_3x_2x_1 \oplus x_3x_2x_1 = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1x_2x_3$$

2) Нехай задано функцію 3-х змінних у ДДНФ:

$$f(\nu) = K_1(\nu) \vee K_3(\nu) \vee K_7(\nu) = \bar{x}_3\bar{x}_2x_1 \vee \bar{x}_3x_2x_1 \vee x_3x_2x_1$$

Тоді:

$$f(\nu) = \bar{x}_3\bar{x}_2x_1 \oplus \bar{x}_3x_2x_1 \oplus x_3x_2x_1 =$$

$$= \underbrace{(1 \oplus x_3)(1 \oplus x_2)}_{1 \oplus x_2 \oplus x_3 \oplus x_2x_3}x_1 \oplus (1 \oplus x_3)x_2x_1 \oplus x_3x_2x_1 =$$

$$= (1 \oplus x_2 \oplus x_3 \oplus x_3x_2)x_1 \oplus x_2x_1 \oplus \underbrace{x_3x_2x_1 \oplus x_3x_2x_1}_{0} =$$

$$= x_1 \oplus x_2x_1 \oplus x_3x_1 \oplus x_3x_2x_1 \oplus x_2x_1 = x_1 \oplus x_3x_1 \oplus x_3x_2x_1$$

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Дайте визначення та запишіть узагальнений вираз досконалої диз'юнктивної нормальної форми (ДДНФ).
2. Дайте визначення та запишіть узагальнений вираз досконалої кон'юнктивної нормальної форми (ДКНФ).
3. З яких міркувань випливає, що будь-яка двійкова функція може бути представлена у ДДНФ чи ДКНФ?
4. Що означають терміни «досконала форма» і «нормальна форма» функції у визначеннях ДДНФ та ДКНФ?
5. Як можна узагальнити правила запису досконалих нормальних форм у базисах І-НІ та АБО-НІ?
6. Якщо функція задана таблицею істинності, як знайти її аналітичний вираз у формі ДДНФ? А як це ж зробити у формі ДКНФ?
7. В якому вигляді розклад Рида-Малера дозволяє представляти логічні функції?
8. Функція 3-х змінних $f(\nu)$ задана таблицею істинності:

x_3	x_2	x_1	$f(\nu)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

   Запишіть ДДНФ та ДКНФ даної функції.
9. Функція 4-х змінних $f(\nu)$ приймає одиничні значення у точках 0101, 0111, 1000, 1100 та 1110. Знайдіть аналітичний вираз, що реалізує дану функцію.
10. Функція 4-х змінних $f(\nu)$ приймає нульові значення у точках 0010, 0011, 0110, 1100 та 1100. Знайдіть аналітичний вираз, що реалізує дану функцію.

8. СИСТЕМИ ЛОГІЧНИХ РІВНЯНЬ

Відповідно до фізичного змісту задач, які доводиться вирішувати в цифровій електроніці, часто доводиться працювати зі співвідношеннями виду [1]:

$$f(\nu, y_m, \ldots, y_1) = g(\nu, y_m, \ldots, y_1), \qquad (8.1)$$

де $\nu = (x_n, \ldots, x_1)$, $y_r = \varphi_r(\nu)$, $r = 1, 2, \ldots, m$. Подібні співвідношення є не тотожностями, а логічними рівняннями, які перетворюються в тотожності лише при певних значеннях функції $y_r = \varphi_r(\nu)$.

В таких випадках співвідношення (8.1) можна розглядати як рівняння з m невідомими y_r. На відміну від звичайних алгебраїчних рівнянь, що вивчаються у звичайній алгебрі, в алгебрі логіки кількість рівнянь та невідомих можуть не співпадати, при цьому такі рівняння можуть мати цілком конкретні і однозначні рішення. Зауважимо, що в задачах цифрової електроніки кількість рівнянь перевищує кількість невідомих.

8.1. Системи логічних рівнянь з одним невідомим

Нехай задана система логічних рівнянь з одним невідомим:

$$f_j(\nu, y) = g_j(\nu, y), \qquad (8.2)$$

де $\nu = (x_n, \ldots, x_1)$, $j = 1, 2, \ldots, k$. Необхідно її розв'язати відносно y, тобто знайти такі значення $y = \varphi(\nu)$, які б перетворювали у тотожності всі рівняння системи:

$$f_j[\nu, \varphi(\nu)] \equiv g_j[\nu, \varphi(\nu)].$$

Для того, щоб вирішувати системи рівнянь, необхідно навчитися виконувати над ними дії та логічні операції, які б не порушували логічних зв'язків, які ці системи рівнянь виражають.

Так, якщо $x = y$, то очевидно, що $x * z = y * z$, де $*$ – будь-яка двомісна операція алгебри логіки ($\wedge$, $\vee$, $\oplus$ та ін.). Однак, зворотне твердження в загальному випадку не буде істинним, тобто із того, що $x * z = y * z$ не обов'язково випливає, що $x = y$. Наприклад, із рівності $x \vee z = y \vee z$ не слідує, що $x = y$ (якщо $x = 1$, $y = 0$, $z = 1$, то $1 \vee 1 = 0 \vee 1 = 1$). Таким чином подібні операції перетворення виразів у загальному випадку застосовувати до системи рівнянь виду (8.2) неможна.

Але тепер візьмемо у якості операції $*$ суму по модулю 2 «$\oplus$».

Із рівності $x = y$ слідує, що $x \oplus z = y \oplus z$. Запишемо дану рівність двічі відносно змінної z:

$$(x \oplus z) \oplus z = (y \oplus z) \oplus z \;\Rightarrow\; x \oplus \underbrace{(z \oplus z)}_{0} = y \oplus \underbrace{(z \oplus z)}_{0} \;\Rightarrow$$

$$\Rightarrow\; x \oplus 0 = y \oplus 0 \;\Rightarrow\; x = y$$

Тобто із твердження $x \oplus z = y \oplus z$ також випливає і те, що $x = y$. Отже:

$$x = y \;\Leftrightarrow\; x \oplus z = y \oplus z \qquad (8.3)$$

Таким чином, логічні зв'язки, які виражають рівняння, не порушуються при виконанні їх перетворень за допомогою операції $\oplus$. Очевидно, що в якості змінної z у рівнянні (8.3) може виступати і будь-яка зі змінних x або y.

Отже, користуючись наслідком (8.3), виконаємо еквівалентні перетворення рівняння (8.2):

$$f_j(\nu, y) \oplus g_j(\nu, y) = \underbrace{g_j(\nu, y) \oplus g_j(\nu, y)}_{0} \;\Rightarrow$$
$$\Rightarrow\; f_j(\nu, y) \oplus g_j(\nu, y) = 0 \qquad (8.4)$$

На основі аксіом алгебри логіки (8.2) і (8.3), які стверджують:

$$\begin{cases} 1 \vee 1 = 1 \\ 0 \cdot 0 = 0 \end{cases} \quad \begin{cases} 1 \cdot 1 = 1 \\ 0 \vee 0 = 0 \end{cases},$$

легко впевнитись, що якщо $x = 0$ та $y = 0$, то $x \vee y = 0$ і навпаки: якщо $x \vee y = 0$, то $x = 0$ та $y = 0$ одночасно, а це означає, що операцію диз'юнкції «$\vee$» можна використовувати для перетворень рівнянь, права частина яких дорівнює 0. Таким чином, за допомогою диз'юнкції всі рівняння системи (9.4) можна об'єднати в одне:

$$\bigvee_{j=1}^{k} [f_j(\nu, y) \oplus g_j(\nu, y)] = 0 \qquad (8.5)$$

Таким чином, рішення системи рівнянь (8.4) відносно змінної y зводиться до рішення одного рівняння (8.5) відносно тої ж змінної y.

Тепер запишемо розклад Шеннона рівняння (8.5) за змінною y:

$$\bar{y} \cdot \psi_1 \vee y \cdot \psi_2 = 0, \qquad (8.6)$$

де $\psi_1 = \bigvee_{j=1}^{k} [f_j(\nu,0) \oplus g_j(\nu,0)];$

$\psi_2 = \bigvee_{j=1}^{k} [f_j(\nu,1) \oplus g_j(\nu,1)].$

Рішенням рівняння (8.6) може бути лише деяка функція $\varphi(\psi_2, \psi_1)$ від змінних ψ_2 та ψ_1. Складемо таблицю всіх можливих значень змінних ψ_2 та ψ_1 та відповідних їм значень змінної y (див. табл. 8.1).

Таблиця 8.1 – можливих значень змінних ψ_2 та ψ_1 та відповідних їм значень змінної y

№	ψ_2	ψ_1	$\bar{y} \cdot \psi_1 \vee y \cdot \psi_2 = 0$	y
1	0	0	$\bar{y} \cdot 0 \vee y \cdot 0 = 0 \Rightarrow 0 \vee 0 = 0$	$y = \hbar(v)$[†]
2	0	1	$\bar{y} \cdot 1 \vee y \cdot 0 = 0 \Rightarrow \bar{y} = 0$	$y = 1$
3	1	0	$\bar{y} \cdot 0 \vee y \cdot 1 = 0 \Rightarrow y = 0$	$y = 0$
4	1	1	$\bar{y} \cdot 1 \vee y \cdot 1 = 0 \Rightarrow \bar{y} \vee y = 0 \Rightarrow 1 \neq 0$	$y \in \varnothing$[‡]

[†] $\hbar(v)$ – повністю невизначена функція, яка може бути рівною 0 або 1, рішенням може бути будь-яка функція, оскільки y не впливає на ліву частину (8.6) рішення;

[‡] рівняння розв'язків немає, оскільки тотожність не виконуватиметься незалежно від значення y.

Із таблиці 8.1 випливає, рішення існує лише в перших трьох випадках, тобто коли значення ψ_2 та ψ_1 одночасно не дорівнюють 1:

$$\psi_2 \cdot \psi_1 = 0. \qquad (8.7)$$

Таким чином, співвідношення (9.7) є необхідною умовою існування рішення.

Оскільки рішення, наведені в таблиці 1, задовольняють рівняння (8.6), права частина якого дорівнює 0, то їх можна об'єднати за допомогою операції диз'юнкції «$\vee$» (як це вже робилося вище для рівняння (8.5)). Об'єднане рівняння доцільно представити у вигляді мультиплексної функції з керуючими змінними ψ_2 та ψ_1 та інформаційною змінною y (рішення), при цьому рішення, записане в рядку 4 представимо як $c \cdot \psi_2\psi_1$, де c може дорівнювати 0 або 1, оскільки за умовою (8.7) $\psi_2\psi_1 = 0$:

$$y = \phi(\psi_2, \psi_1) = \hbar \cdot \overline{\psi}_2\overline{\psi}_1 \vee \underbrace{1 \cdot \overline{\psi}_2\psi_1}_{\overline{\psi}_2\psi_1} \vee \underbrace{0 \cdot \psi_2\overline{\psi}_1}_{0} \vee \underbrace{c \cdot \psi_2\psi_1}_{\substack{0 \\ \text{із умови (9.7)}}}$$

Приймемо $c = 1$, тоді:

$$y = \varphi(\psi_2, \psi_1) = \hbar \cdot \overline{\psi}_2\overline{\psi}_1 \vee \psi_1\underbrace{(\overline{\psi}_2 \vee \psi_2)}_{1} = \underbrace{\hbar \cdot \overline{\psi}_2\overline{\psi}_1 \vee \psi_1}_{\substack{\text{закон} \\ \text{узагальненого} \\ \text{склеювання}}} = \psi_1 \vee \overline{\psi}_2\hbar$$

Отже рішення системи рівнянь (8.6) в загальному вигляді буде:

$$\begin{cases} y(\psi_2, \psi_1) = \psi_1 \vee \overline{\psi}_2\hbar \\ \psi_1\psi_2 = 0 \end{cases} \qquad (8.8)$$

Для перевірки підставимо рішення (8.8) в рівняння (8.6):

$$\overline{\psi_1 \vee \overline{\psi}_2 \hbar} \cdot \psi_1 \vee (\psi_1 \vee \overline{\psi}_2 \hbar) \cdot \psi_2 =$$
$$= \underbrace{(\psi_1 \overline{\psi}_1)\psi_2 \hbar}_{0} \vee \psi_2 \psi_1 \vee \underbrace{(\psi_2 \overline{\psi}_2)\hbar}_{0} = \psi_2 \psi_1 = 0$$

Із виразу (8.8) випливають наступні висновки:

1) Якщо $\psi_2(v) = 1$, то рішенням є $y = \psi_1(v)$ – рішення є єдиним і є повністю визначеною функцією;
2) Якщо $\psi_2(v) = 0$ і $\psi_1(v) \equiv 0$, то $y = \hbar(v)$ – повністю невизначена функція (рівняння є тотожністю і не залежить від y);
3) В загальному випадку рішенням є $y = \psi_1(v) \vee \hbar(v)$, де $\hbar(v)$ дорівнює 0 або 1, в точках в яких $\psi_2(v) = 0$, а в інших точках $y = \psi_1(v)$, тобто рішенням є не повністю визначена функція, якій відповідає цілий клас повністю визначених функцій.

Записуючи значення ψ_1 і ψ_2 через $f_j(\nu, y)$ та $g_j(\nu, y)$ рішення системи виглядатиме наступним чином:

$$y = \underbrace{\bigvee_{j=1}^{k} [f_j(\nu,0) \oplus g_j(\nu,0)]}_{\psi_1} \vee \hbar(\nu) \cdot \overline{\underbrace{\bigvee_{j=1}^{k} [f_j(\nu,1) \oplus g_j(\nu,1)]}_{\psi_2}} \qquad (8.9)$$

Приклади

1) Довести тотожність (операція узагальненого склеювання):

$$\underbrace{xy \vee \bar{x}z \vee yz}_{f} = \underbrace{xy \vee \bar{x}z}_{g}$$

Вирішимо рівняння відносно x.

$$x = \psi_1 \vee \overline{\psi}_2 \hbar = (0y \vee \overline{0}z \vee yz) \oplus (0y \vee \overline{0}z) \vee \hbar \cdot \overline{(1y \vee \overline{1}z \vee yz) \oplus (1y \vee \overline{1}z)} =$$
$$= \underbrace{(z \vee yz)}_{z(1 \vee y)} \oplus z \vee \hbar \cdot \overline{\underbrace{(y \vee yz)}_{y(1 \vee z)} \oplus y} = \underbrace{z \oplus z}_{\psi_1} \vee \hbar \cdot \overline{\underbrace{y \oplus y}_{\overline{\psi}_2}} = 0 \vee \hbar \cdot \overline{0}$$

Виконаємо перевірку існування рішення:

$$\begin{cases} \psi_1 = z \oplus z = 0 \\ \psi_2 = y \oplus y = 0 \end{cases} \Rightarrow \quad \psi_1 \psi_2 = 0$$

Отже рішення існує і воно є наступним: $0 \vee \hbar \cdot \overline{0} = 0 \vee \hbar \cdot 1 = \hbar$, що означає, що рівність виконується незалежно від значення x, тобто вираз тотожний відносно x.

2) Знайти рішення рівняння відносно y:

$$x \cdot \bar{y} \vee \bar{x} \cdot y = y$$

Рішення:

$$y = \psi_1 \vee \bar{\psi}_2 \hbar = (x \cdot \bar{0} \vee \bar{x} \cdot 0) \oplus 0 \vee \hbar \cdot \overline{(x \cdot \bar{1} \vee \bar{x} \cdot 1) \oplus 1} =$$
$$= x \oplus 0 \vee \hbar \cdot \overline{\bar{x} \oplus 1} = x \vee \hbar \cdot \bar{x}$$

Перевіряємо існування рішення:

$$\begin{cases} \psi_1 = x \\ \psi_2 = x \end{cases} \quad \Rightarrow \quad x \cdot x = x \not\equiv 0,$$

Отже рівняння немає рішень відносно y.

3) Знайти рішення рівняння відносно y:

$$x \cdot \bar{y} \vee \bar{x} \cdot y = x$$

Рішення:

$$y = \psi_1 \vee \bar{\psi}_2 \hbar = (x \cdot \bar{0} \vee \bar{x} \cdot 0) \oplus x \vee \hbar \cdot \overline{(x \cdot \bar{1} \vee \bar{x} \cdot 1) \oplus x} =$$
$$= x \oplus x \vee \hbar \cdot \overline{\bar{x} \oplus x} = 0 \vee \hbar \cdot \bar{1} = 0$$

Перевіряємо існування рішення:

$$\begin{cases} \psi_1 = 0 \\ \psi_2 = 1 \end{cases} \quad \Rightarrow \quad \psi_1 \psi_2 = 0 \cdot 1 = 0,$$

Існує єдиний розв'язок рівняння $y = 0$.

4) Довести теорему:

Якщо $\underbrace{x \cdot y}_{f_1} = \underbrace{x \cdot z}_{g_1}$ та $\underbrace{x \vee y}_{f_2} = \underbrace{x \vee z}_{g_2}$, то $y = z$.

Доведення: вирішимо систему з двох рівнянь відносно y:

$$\psi_1 = (x \cdot 0) \oplus (x \cdot z) \vee (x \vee 0) \oplus (x \vee z) = 0 \oplus xz \vee \underbrace{x \oplus (x \vee z)}_{\bar{x}(x \vee z) \vee x\overline{(x \vee z)}} = \underbrace{xz \vee \bar{x}z}_{z(x \vee \bar{x})} \vee \underbrace{x\bar{x}\bar{z}}_{0} = z$$

$$\psi_2 = (x \cdot 1) \oplus (x \cdot z) \vee (x \vee 1) \oplus (x \vee z) = \underbrace{x \oplus xz}_{x(1 \oplus z)} \vee \underbrace{1 \oplus (x \vee z)}_{\bar{1}(x \vee z) \vee 1\overline{(x \vee z)}} = x\bar{z} \vee \bar{x}\bar{z} = \bar{z}$$

Перевіряємо існування рішення:

$$\begin{cases} \psi_1 = z \\ \psi_2 = \bar{z} \end{cases} \quad \Rightarrow \quad \psi_1 \psi_2 = z \cdot \bar{z} = 0$$

Рішення існує і є єдиним: $y = \psi_1 \vee \bar{\psi}_2 \hbar = z \vee \bar{\bar{z}} \hbar = z \vee z\hbar = z(1 \vee \hbar) = z$

Що і треба було довести.

5) Розв’язати відносно T:

$$Q^+ = Q \oplus T$$

Рішення:

$$\psi_1 = Q^+ \oplus (Q \oplus 0) = Q^+ \oplus Q$$

$$\psi_2 = Q^+ \oplus (Q \oplus 1) = Q^+ \oplus \overline{Q} = \overline{Q}^+ \overline{Q} \vee Q^+ Q = \overline{Q^+ \oplus Q}$$

Перевіряємо існування рішення:

$$\begin{cases} \psi_1 = Q^+ \oplus Q \\ \psi_2 = \overline{Q^+ \oplus Q} \end{cases} \quad \Rightarrow \quad (Q^+ \oplus Q)(\overline{Q^+ \oplus Q}) = 0$$

Рішення існує і є єдиним:

$$T = \psi_1 \vee \overline{\psi}_2 \hbar = Q^+ \oplus Q \vee \overline{\overline{Q^+ \oplus Q}} \hbar = Q^+ \oplus Q(1 \vee \hbar) = Q^+ \oplus Q\,.$$

Перевірка: $Q^+ = Q \oplus \underbrace{Q^+ \oplus Q}_{T} = \underbrace{(Q \oplus Q)}_{0} \oplus Q^+ = 0 \oplus Q^+ = Q^+$

8.2. Арифметичне представлення логічних рівнянь

Якщо символи «0» та «1» вважати відповідними числами, то всі логічні операції можна замінити на арифметичні операції або алгебраїчні формули, на основі очевидних співвідношень:

Логічні операції та змінні	Арифметичні операції та змінні
$y = x$	$y = x$
$y = \neg x$	$y = 1 - x$
$y = x_1 \wedge x_2$	$y = x_1 \cdot x_2$

На основі наведених співвідношень можна отримати операції диз’юнкції та суми по модулю 2:

$$x_1 \vee x_2 = \overline{\overline{x}_1 \cdot \overline{x}_2} = (1 - (1 - x_1) \cdot (1 - x_2))) = 1 - 1 + x_1 + x_2 - x_1 x_2 = x_1 + x_2 - x_1 x_2$$

$$x_1 \oplus x_2 = \overline{x}_1 x_2 \vee x_1 \overline{x}_2 == x_1 + x_2 - 2x_1 x_2$$

За допомогою даних співвідношень логічне рівняння (8.6) можна перетворити на алгебраїчне:

$$\overline{y} \cdot \psi_1 \vee y \cdot \psi_2 = 0 \quad \Rightarrow \quad (1 - y)\psi_1 + y\psi_2 - (1 - y)\psi_1 \cdot y\psi_2 = 0$$

Якщо врахувати, що $\psi_2 \psi_1 = 0$, то дане рівняння зводиться до вигляду:

$$(1 - y)\psi_1 + y\psi_2 = 0 \quad \Rightarrow \quad \psi_1 - y(\psi_1 - \psi_2) = 0$$

Рішенням якого буде:

$$y = \frac{\psi_1}{\psi_1 - \psi_2} \qquad (*)$$

Легко бачити, що при $\psi_1 = \psi_2$ рішення не існує.

Однак, переходячи таким чином до алгебраїчного виду, у рішенні (*) втрачено невизначену функцію $\hbar$. Для того, щоб дана функція була присутня і в алгебраїчному рішенні, приведемо до цього вигляду формулу (9.8), яка справедлива для випадку $\psi_2\psi_1 = 0$:

$$y = \psi_1 \vee \hbar\overline{\psi}_2 \quad \Rightarrow \quad \psi_1 + \hbar \cdot (1-\psi_2) - \psi_1 \cdot \hbar \cdot (1-\psi_2) = 0 \quad \Rightarrow$$
$$\Rightarrow \quad \psi_1 + \hbar - \psi_2\hbar - \psi_1\hbar - \underbrace{\psi_2\psi_1\hbar}_{0} = \psi_1 + \hbar(1-\psi_1-\psi_2) \qquad (**)$$

8.3. Системи логічних рівнянь з більш ніж одним невідомим

Розглянемо спочатку систему рівнянь з двома невідомими:

$$f_j(\nu, y, z) = g_j(\nu, y, z), \qquad (8.10)$$

де y та z – невідомі; $\nu = (x_n, \ldots, x_1)$, $j = 1, 2, \ldots, k$.

Рішення даної системи рівнянь зводиться до послідовного її рішення відносно змінних y та z. При цьому необхідно знайти такі значення $y = \varphi_1(\nu)$ та $z = \varphi_2(\nu)$, які б перетворювали у тотожності всі рівняння системи:

$$f_j[\nu, \varphi_1(\nu), \varphi_2(\nu)] \equiv g_j[\nu, \varphi_1(\nu), \varphi_2(\nu)].$$

Користуючись формулою (8.8) запишемо рішення відносно змінної y:

$$\begin{cases} y(\psi_2, \psi_1) = \psi_1 \vee \overline{\psi}_2\hbar \\ \psi_1\psi_2 = 0 \end{cases},$$

$$\begin{cases} 1) \quad y(\psi_1, \psi_2) = \psi_1(\nu, z) \vee \hbar_1(\nu) \cdot \overline{\psi_2(\nu, z)} \\ 2) \quad \psi_1(\nu, z) \cdot \psi_2(\nu, z) = \psi(\nu, z) \end{cases} \qquad (8.11)$$

$$\text{де} \quad \psi_1(\nu, z) = \bigvee_{j=1}^{k} [f_j(\nu, 0, z) \oplus g_j(\nu, 0, z)];$$
$$\psi_2(\nu, z) = \bigvee_{j=1}^{k} [f_j(\nu, 1, z) \oplus g_j(\nu, 1, z)].$$

1) Якщо функція $\psi(\nu, z) = 0$, то це означає, що рішення системи (8.10) відносно y існує незалежно від значень z. Тому можна прийняти $z = \hbar_2(\nu)$ і підставивши це значення в (8.11), отримаємо:

$$y(\nu) = \psi_1(\nu, \hbar_2) \vee \hbar_1 \cdot \overline{\psi_2(\nu, \hbar_2)} = \varphi_1(\nu)$$

2) Якщо функція $\psi(\nu, z) \neq 0$ при будь-яких z, то можна спробувати з'ясувати, чи існують такі окремі рішення z, при яких функція $\psi(\nu, z)$ перетворюється на 0. Від повідь на це питання можна отримати шляхом рішення рівняння відносно z:

$$\psi(\nu, z) = \psi_1(\nu, z) \cdot \psi_2(\nu, z) = 0$$

Рішенням цього рівняння буде:

$$z(\nu) = \psi(\nu,0) \oplus 0 \vee \hbar_2 \cdot \overline{\psi(\nu,1)} = \varphi_2(\nu),$$

яке існує тільки в тому випадку, якщо виконується умова:

$$\psi(\nu) = \psi(\nu,0) \cdot \psi(\nu,1) = 0,$$

де $\psi(\nu, 0) = \psi_1(\nu, 0) \cdot \psi_2(\nu, 0)$;

$\psi(\nu, 1) = \psi_1(\nu, 1) \cdot \psi_2(\nu, 1)$;

$\hbar_2 = \hbar_2(\nu)$ – незалежна від $\hbar_1 = \hbar_1(\nu)$ повністю невизначена функція.

Якщо дана умова виконується, то рішення системи логічних рівнянь (8.10) відносно y визначається шляхом підстановки у функцію (8.1) системи (8.10) знайденого значення $z = \varphi_2(\nu)$:

$$y(\nu) = \psi_1(\nu, \varphi_2(\nu)) \vee \hbar_1(\nu) \cdot \overline{\psi_2(\nu, \varphi_2(\nu))} = \varphi_1(\nu)$$

В результаті отримують $y = \varphi_1(\nu)$ та $z = \varphi_2(\nu)$, які є не залежними від невідомих y та z.

Таким чином, послідовно вирішуючи рівняння відносно кожної зі змінних знаходять рішення всієї системи. В загальному випадку, залежно від порядку рішення системи логічних рівнянь відносно невідомих отримують різні форми функцій. Однак, всі ці форми є еквівалентними, тобто різним формам отриманих неповністю визначених функцій відповідає один і той самий клас повністю визначених функцій.

Формування та вирішення логічних рівнянь є потужним математичним апаратом для аналізу та синтезу логічних схем та широко застосовується на практиці під час проектування цифрових пристроїв.

Приклади

1) Вирішити систему рівнянь відносно y та z:

$$\overline{x \oplus y \oplus z} = x \vee y \vee z$$

Рішення: спочатку рішення шукатимемо відносно y:

$$y = \psi_1 \vee \hbar\overline{\psi}_2,$$

$$\psi_1 = \overline{x \oplus 0 \oplus z} \oplus (x \vee 0 \vee z) = \overline{x \oplus z} \oplus (x \vee z) = \underbrace{\overline{\overline{x \oplus z}} \cdot (x \vee z)}_{(x \oplus z)\cdot x \vee (x \oplus z)\cdot z} \vee \underbrace{\overline{x \oplus z} \cdot \overline{(x \vee z)}}_{\overline{x \oplus z}\cdot \bar{x}\cdot \bar{z}} =$$

$$= \underbrace{x\bar{x}z}_{0} \vee xx\bar{z} \vee z\bar{x}z \vee \underbrace{z\bar{x}\bar{z}}_{0} \vee \bar{x}\bar{z}\bar{x}\bar{z} \vee \underbrace{xz\bar{x}\bar{z}}_{0} = x\bar{z} \vee \bar{x}z \vee \bar{x}\bar{z} = x\bar{z} \vee \bar{x}(z \vee \bar{z}) =$$

$$= x\bar{z} \vee \bar{x} = \bar{x} \vee \bar{z}$$

$$\psi_2 = \overline{x \oplus 1 \oplus z} \oplus (x \vee 1 \vee z) = \overline{(x \oplus z) \oplus 1} \oplus 1 = \overline{\overline{x \oplus z}} \oplus 1 = \overline{x \oplus z}$$

Перевіримо існування рішення:

$$\psi_1\psi_2 = (\bar{x} \vee \bar{z})\overline{x \oplus z} = (\bar{x} \vee \bar{z})\bar{x}\bar{z} \vee (\bar{x} \vee \bar{z})xz = \bar{x}\bar{x}\bar{z} \vee \bar{z}\bar{x}\bar{z} \vee \bar{x}xz \vee \bar{z}xz =$$

$$= \bar{x}\bar{z} \vee \bar{x}\bar{z} = \bar{x}\bar{z} \neq 0$$

Як бачимо в загальному випадку рішення відносно *y* не існує, отже необхідно визначити, чи існують такі *z*, при яких система рівнянь має рішення. Для цього розв'яжемо рівняння $\psi_1\psi_2 = 0$ відносно *z*:

$$\psi_1\psi_2 = 0 \quad \Rightarrow \quad (\bar{x} \vee \bar{z})\overline{x \oplus z} = 0 \quad \Rightarrow \quad \bar{x}\bar{z} = 0$$

Рішення рівняння $\bar{x}\bar{z} = 0$ відносно *z*:

$$\psi_1' = \bar{x}\overline{0} = \bar{x}$$

$$\psi_2' = \bar{x}\overline{1} = 0$$

Перевірка існування рішення:

$$\psi_1'\psi_2' = \bar{x} \cdot 0 = 0.$$

Рішення відносно *z* існує і рівне: $z = \psi_1' \vee \hbar_2\overline{\psi_2'} = \bar{x} \vee \hbar_2 \cdot \overline{0} = \bar{x} \vee \hbar_2$.

Підставимо рішення відносно *z* у рішення відносно *y*:

Рішення відносно *y*:

$$y = \psi_1 \vee \hbar_1\overline{\psi}_2 \quad \Rightarrow \quad y = (\bar{x} \vee \bar{z}) \vee \hbar_1\overline{\overline{x \oplus z}} \quad \Rightarrow \quad y = (\bar{x} \vee \bar{z}) \vee \hbar_1(x \oplus z)$$

Підставляємо *z*:

$$y = (\bar{x} \vee \overline{\bar{x} \vee \hbar_2}) \vee \hbar_1(x \oplus (\bar{x} \vee \hbar_2)) = \underbrace{\bar{x} \vee x\bar{\hbar}_2}_{\text{склеювання}} \vee \hbar_1(\bar{x}(\bar{x} \vee \hbar_2) \vee x \cdot \overline{\bar{x} \vee \hbar_2}) =$$

$$= \bar{x} \vee \bar{\hbar}_2 \vee \hbar_1(\underbrace{\bar{x}\bar{x}}_{\bar{x}} \vee \bar{x}\hbar_2) \vee x \cdot x \cdot \bar{\hbar}_2) = \bar{x} \vee \bar{\hbar}_2 \vee \hbar_1(\underbrace{\bar{x} \vee x\bar{\hbar}_2}_{\text{склеювання}})$$

$$= (\bar{x} \vee \bar{\hbar}_2) \vee \hbar_1(\bar{x} \vee \bar{\hbar}_2) = (\bar{x} \vee \bar{\hbar}_2)(1 \vee \hbar_1) = \bar{x} \vee \bar{\hbar}_2$$

Остаточне рішення: $y = \bar{x} \vee \bar{\hbar}_2$ та $z = \bar{x} \vee \hbar_2$.

Перевірка:

$$\overline{x \oplus (\bar{x} \vee \bar{\hbar}_2) \oplus (\bar{x} \vee \hbar_2)} = x \vee \bar{x} \vee \bar{\hbar}_2 \vee \bar{x} \vee \hbar_2 \quad \Rightarrow$$

$$\Rightarrow \quad \overline{x \oplus [\underbrace{\overline{(\bar{x} \vee \bar{\hbar}_2)}(\bar{x} \vee \hbar_2)}_{x\hbar_2\bar{x} \vee x\hbar_2\hbar_2} \vee \underbrace{(\bar{x} \vee \bar{\hbar}_2)\overline{(\bar{x} \vee \hbar_2)}}_{\bar{x}x\bar{\hbar}_2 \vee \bar{\hbar}_2 x\bar{\hbar}_2}]} = x \vee \underbrace{\bar{x} \vee \bar{x}}_{\bar{x}} \vee \bar{\hbar}_2 \vee \hbar_2 \quad \Rightarrow$$

$$\Rightarrow \quad \overline{x \oplus (\underbrace{x\hbar_2 \vee x\bar{\hbar}_2}_{x(\hbar_2 \vee \bar{\hbar}_2)})} = 1 \quad \Rightarrow \quad \overline{x \oplus x} = 1 \quad \Rightarrow \quad 1 = 1$$

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Коротко сформулюйте алгоритм аналітичного рішення логічних рівнянь з одним невідомим.
2. Як встановити чи існує рішення логічного рівняння?
3. Чи може існувати рішення системи логічних рівнянь, якщо кількість невідомих перевищує кількість рівнянь?
4. В чому полягає основна відмінність між алгоритмами рішення логічного рівняння з одним невідомим від рішення системи логічних рівнянь?
5. Вирішіть рівняння:

   $$Q^+ = \bar{Q} \cdot J \vee Q \cdot \bar{K}$$

   відносно змінних J і K.

9. МІНІМІЗАЦІЯ ЛОГІЧНИХ ФУНКЦІЙ

Канонічна задача мінімізації логічних функцій полягає у пошуку диз'юнктивної форми, яка містить мінімальну кількість аргументів. Такі форми функцій прийнято називати *мінімальними диз'юнктивними нормальними формами* (*МДНФ*). Початковою формою функції при вирішенні канонічної задачі мінімізації є її досконала диз'юнктивна нормальна форма (ДДНФ) [1–3, 6, 7].

9.1. Метод Квайна

Метод Квайна оснований на використанні двох основних співвідношень:

1. Співвідношення склеювання:

$$Ax \vee A\bar{x} = A(\underbrace{x \vee \bar{x}}_{1}) = A, \qquad (9.1)$$

де A – будь-який елементарний добуток (кон'юнкція).

2. Співвідношення поглинання:

$$A \cdot B \vee A = A(\underbrace{B \vee 1}_{1}) = A, \qquad (9.2)$$

де B – інший відмінний від A елементарний добуток.

Із співвідношення поглинання випливає, що довільний елементарний добуток поглинається будь-якою його частиною.

Суть метода полягає в послідовному виконанні всіх можливих склеювань, а потім поглинань, що дає скорочену ДНФ. Сам метод Квайна має два етапи:

1) Отримання скороченої форми функції.
2) Виключення з даної скороченої форми надлишкових імплікант.

Розглянемо метод більш детально.

Етап 1

На першому етапі використовується поняття *суміжних мітнетрмів* та *імплікант*, а основною операцією скорочення є операція склеювання.

Мінтреми $K_i(\nu)$ та $K_j(\nu)$ називаються суміжними (або сусідніми), якщо вони відрізняються лише одним первинним термом $x_p^1 = x_p$ або $x_p^0 = \bar{x}_p$.

Кожен мінтерм n змінних із загальної кількості 2^n мінтермів має n суміжних мінтермів. Наприклад, у функції 2-х змінних:

$$K_0 = \bar{x}_2\bar{x}_1, \quad K_1 = x_2\bar{x}_1, \quad K_2 = \bar{x}_2 x_1, \quad K_3 = x_2 x_1,$$

мінтерм $K_1 = x_2\bar{x}_1$ має два суміжні мінтерми $K_0 = \bar{x}_2\bar{x}_1$ та $K_3 = x_2x_1$.

Відповідно до співвідношення (9.1) два суміжні мінтерми можна склеїти відносно аргументу, який у них є відмінним. Це призводить до того, що диз'юнкція даних мінтермів перетворюється на кон'юнкцію термів з числом аргументів на *один меншим*, ніж в первинних мінтермах. Наприклад:

$$x_4\bar{x}_3x_2x_1 \vee x_4\bar{x}_3x_2\bar{x}_1 = x_4\bar{x}_3x_2(x_1 \vee \bar{x}_1) = x_4\bar{x}_3x_2.$$

Кон'юнкції термів, які отримують в результаті склеювання двох суміжних мінтермів називають *імплікантами* або *контермами* та позначають як $K_{i,j}(\nu)$.

Імпліканти, які отримані з первинної функції за допомогою склеювання та мають однакову кількість аргументів знову можуть виявитись суміжними. Це дозволяє застосувати до них операцію склеювання ще раз. При цьому, очевидно, всі отримувані імпліканти є повністю еквівалентними мінтермам, які були використані для їх отримання. В даному випадку говорять, що імпліканти *покривають* всі мінтерми, з яких вони були отримані.

Приклад:

Дано логічну функцію 4-х змінних, яка рівна 1 в точках 0100, 0101, 1100, 1101.

ДДНФ заданої логічної функції матиме вигляд:

$$f(\nu) = \underbrace{\bar{x}_4x_3\bar{x}_2\bar{x}_1}_{K_4(\nu)} \vee \underbrace{\bar{x}_4x_3\bar{x}_2x_1}_{K_5(\nu)} \vee \underbrace{x_4x_3\bar{x}_2\bar{x}_1}_{K_{12}(\nu)} \vee \underbrace{x_4x_3\bar{x}_2x_1}_{K_{13}(\nu)}$$

Суміжні мінтерми та результати їх склеювання:

$$\bar{x}_4x_3\bar{x}_2\bar{x}_1 \vee \bar{x}_4x_3\bar{x}_2x_1 = \bar{x}_4x_3\bar{x}_2 \quad \Rightarrow \quad K_4(\nu) \vee K_5(\nu) = K_{4,5}(\nu)$$

$$\bar{x}_4x_3\bar{x}_2\bar{x}_1 \vee x_4x_3\bar{x}_2\bar{x}_1 = x_3\bar{x}_2\bar{x}_1 \quad \Rightarrow \quad K_4(\nu) \vee K_{12}(\nu) = K_{4,12}(\nu)$$

$$\bar{x}_4x_3\bar{x}_2x_1 \vee x_4x_3\bar{x}_2x_1 = x_3\bar{x}_2x_1 \quad \Rightarrow \quad K_5(\nu) \vee K_{13}(\nu) = K_{5,13}(\nu)$$

$$x_4x_3\bar{x}_2\bar{x}_1 \vee x_4x_3\bar{x}_2x_1 = x_4x_3\bar{x}_2 \quad \Rightarrow \quad K_{12}(\nu) \vee K_{13}(\nu) = K_{12,13}(\nu)$$

Отже функцію $f(\nu)$ можна представити як $K_{4,5}(\nu) \vee K_{12,13}(\nu)$ або $K_{4,12}(\nu) \vee K_{5,13}(\nu)$, оскільки обидва варіанти покривають всі початкові мінтерми ДДНФ даної функції:

$$f(\nu) = \underbrace{\bar{x}_4x_3\bar{x}_2}_{K_{4,5}(\nu)} \vee \underbrace{x_4x_3\bar{x}_2}_{K_{12,13}(\nu)},$$

$$f(\nu) = \underbrace{x_3\bar{x}_2\bar{x}_1}_{K_{4,12}(\nu)} \vee \underbrace{x_3\bar{x}_2x_1}_{K_{5,13}(\nu)}.$$

Однак, видно, що в отриманих варіантах функцій знову наявні суміжні мінтерми, які знову можна склеїти:

$$\bar{x}_4 x_3 \bar{x}_2 \vee x_4 x_3 \bar{x}_2 = x_3 \bar{x}_2 \qquad \Rightarrow \qquad K_{4,5}(\nu) \vee K_{12,13}(\nu) = K_{4,5,12,13}(\nu)$$

$$x_3 \bar{x}_2 \bar{x}_1 \vee x_3 \bar{x}_2 x_1 = x_3 \bar{x}_2 \qquad \Rightarrow \qquad K_{4,12}(\nu) \vee K_{5,13}(\nu) = K_{4,12,5,13}(\nu)$$

Процес багатоступеневого склеювання проводиться доти, доки не будуть отримані імпліканти (контерми), які не є взаємно-суміжними. Такі імпліканти називають *простими імплікантими*.

Контерми (імпліканти) є виродженими функціями, якщо кількість термів, що входять до них менше числа змінних функції.

Прості імпліканти, разом із мінтермами, які входили в початкову ДДНФ та з самого початку не мали до себе суміжних, утворюють так звану *скорочену диз'юнктивну нормальну форму* (*СкДНФ*).

Отже, загальне правило мінімізації функцій на першому етапі метода Квайна є таким: одним контермом n змінних $K_{i,j}(\nu)$, що не залежить від m змінних ($m \le n$) можна замінити диз'юнкцію 2^m мінтермів, якщо кожен з них має по m сусідніх серед усіх $2^m - 1$ мінтермів. Таким чином в результаті склеювання суміжних мінтермів функція $f(\nu)$ набуде вигляду:

$$f(\nu) = \bigvee K_{i,j}(\nu).$$

Етап 2

Може виявитись так, що отримана на першому етапі скорочена диз'юнктивна нормальна форма міститиме імпліканти, які одночасно покривають одній й ті самі мінтерми первинної функції. Тоді, відповідно до співвідношення поглинання (9.2) деякі із зазначених імплікант можна не враховувати у остаточній мінімальній функції. Таким чином, другий етап метода Квайна полягає у вилученні зі скороченої диз'юнктивної нормальної форми, отриманої на першому етапі, надлишкових імплікант.

Розглянемо приклад.

Задано функцію 3-х змінних:

$$f(\nu) = \underbrace{x_3 x_2 x_1}_{K_7(\nu)} \vee \underbrace{\bar{x}_3 x_2 x_1}_{K_3(\nu)} \vee \underbrace{x_3 \bar{x}_2 x_1}_{K_5(\nu)} \vee \underbrace{x_3 \bar{x}_2 \bar{x}_1}_{K_4(\nu)}, \qquad (9.3)$$

Виконаємо склеювання. Для того, щоб виконати операцію склеювання необхідно попарно використати мінтерми $K_5(\nu)$ та $K_7(\nu)$ двічі:

$$K_7(\nu) \vee K_3(\nu) = K_{7,3}(\nu) = x_3 x_2 x_1 \vee \bar{x}_3 x_2 x_1 = x_2 x_1$$

$$K_7(\nu) \vee K_5(\nu) = K_{7,5}(\nu) = x_3 x_2 x_1 \vee x_3 \bar{x}_2 x_1 = x_3 x_1$$

$$K_5(\nu) \vee K_4(\nu) = K_{5,4}(\nu) = x_3 \bar{x}_2 x_1 \vee x_3 \bar{x}_2 \bar{x}_1 = x_3 \bar{x}_2$$

Отже, функція $f(\nu)$ матиме наступний вигляд:

$$f(\nu) = x_2 x_1 \vee x_3 x_1 \vee x_3 \bar{x}_2 .$$

Кожна із отриманих імплікант є простою, так як не може бути склеєною з іншими (у функції немає суміжних мінтермів), а отже отримана функція представлена у скороченій диз'юнктивній нормальній формі (СкДНФ).

Разом із тим для покриття мінтермів $K_3(\nu)$, $K_4(\nu)$, $K_5(\nu)$ та $K_7(\nu)$ початкової функції не обов'язково використовувати всі три отримані імпліканти:

- Імпліканта $K_{7,3}(\nu)$ покриває 7 і 3 мінтерми.
- Імпліканта $K_{5,4}(\nu)$ покриває 5 і 4 мінтерми.
- Імпліканта $K_{7,5}(\nu)$ покриває 7 і 5 мінтерми, проте не впливає на задання функції, оскільки ці мінтерми вже покриті іншими імплікантами, тому $K_{7,5}(\nu)$ є надлишковою.

Таким чином функція $f(\nu)$ може бути задана як:

$$f(\nu) = x_2 x_1 \vee x_3 \bar{x}_2 .$$

Отримане представлення називається *тупиковою диз'юнктивною нормальною формою* (*ТДНФ*). Мінімальною диз'юнктивною нормальною формою (МДНФ) є така тупикова ДНФ, яка містить мінімальну кількість змінних.

Для формалізації процедури вилучення надлишкових імплікант також може застосовуватись спеціальна імплікантна матриця Квайна. Рядки цієї матриці відмічаються простими імплікантами (елементами СкДНФ), а стовпці – мінтермами початкової функції. У клітинках матриці на перетині стовпців та рядків, які відповідають мінтермам та простим імплікантам, що входять дані мінтерми (а отже ними поглинаються) робляться позначки. Тоді мінімальна ДНФ по імплікантній матриці може бути знайдена наступним чином:

1) Здійснюється пошук стовпців у імплікантній матриці, що містять тільки одну позначку. Імпліканти, що відповідають знайденим позначкам називаються базовими і складають так зване ядро бульової функції. Ядро обов'язково входить до МДНФ.
2) Розглядаються різні варіанти вибору сукупності простих імплікант, що покривають позначками інші стовпці матриці та обираються варіанти з мінімальною кількістю змінних в такій сукупності імплікант.

Для прикладу складемо імплікантну матрицю для розглянутої вище функції (див. табл. 9.1).

Таблиця 9.1 – Імплікантна матриця для функції (9.3)

Прості імпліканти	Мінтерми			
	$x_3x_2x_1$	$\bar{x}_3x_2x_1$	$x_3\bar{x}_2x_1$	$x_3\bar{x}_2\bar{x}_1$
x_2x_1	*	*		
x_3x_1	*		*	
$x_3\bar{x}_2$			*	*

Як видно з імплікантної матриці ядром функції є імпліканти x_2x_1 та $x_3\bar{x}_2$. Імпліканта x_3x_1 є надлишковою, оскільки ядро покриває всі стовпці імплікантної матриці. Тому функція має одну тупикову та мінімальну ДНФ:

$$f(\nu) = x_2x_1 \vee x_3\bar{x}_2.$$

Отримання *мінімальної кон'юнктивної нормальної форми* (*МКНФ*) зводиться до отримання мінімальної диз'юнктивної нормальної форми інверсної функції (МДНФ) $\overline{f(\nu)}$. При цьому інверсії контермів перетворяться на диз'юнктивні терми – *дизтерми*:

$$M_{i,j}(\nu) = \overline{K_{i,j}(\nu)}.$$

Зауважимо, що в результаті мінімізації можуть бути отримані декілька різних ДНФ, але всі вони будуть рівноцінними.

9.2. Мінімізація логічних функцій методом Квайна-Мак-Класкі

Даний метод є формалізацією метода Квайна. Функція для мінімізації у ньому так само має задаватись досконалою диз'юнктивною нормальною формою (ДДНФ), причому у вигляді двійкових чисел, значення яких відповідають *номерам* відповідних мінтермів ДДНФ. Окрім номера кожному мінтерму присвоюється *індекс*, який визначає кількість одиниць у двійковому числі, що його представляє (кількість змінних без інверсій). Наприклад:

Мінтерм $K_1 = \bar{x}_3\bar{x}_2x_1$ має номер 001 та індекс 1.

Мінтерм $K_5 = x_3\bar{x}_2x_1$ має номер 101 та індекс 2.

Алгоритм Квайна-Мак-Класкі формулюється наступним чином: для того, щоб два мінтерми, були суміжними і могли бути склеєні, необхідно і достатньо, щоб відповідні їм *номери* мали *індекси*, які відрізняються *на*

одиницю, а значення цих номерів відрізнялись на *степінь числа* 2, причому номер з більшим індексом був більшим номера з меншим індексом.

Розглянемо метод Квайна-Мак-Класкі детальніше на прикладі мінімізації логічної функції:

$$f(\nu) = \bar{x}_4\bar{x}_3\bar{x}_2x_1 \vee \bar{x}_4x_3\bar{x}_2x_1 \vee x_4x_3\bar{x}_2\bar{x}_1 \vee \bar{x}_4x_3x_2x_1 \vee x_4\bar{x}_3x_2x_1 \vee \bar{x}_4\bar{x}_3x_2x_1 \quad (9.4)$$

На першому етапі мінімізації необхідно визначити номери та індекси для кожного мінтерма, записуючи логічну функцію у наступному вигляді:

$$f(\nu) = \underset{\substack{1\\(1)}}{0001} \vee \underset{\substack{5\\(2)}}{0101} \vee \underset{\substack{12\\(2)}}{1100} \vee \underset{\substack{7\\(3)}}{0111} \vee \underset{\substack{11\\(3)}}{1011} \vee \underset{\substack{3\\(2)}}{0011},$$

← номери мінтермів
← індекси

де двійкові значення, що відокремлені диз'юнкціями, відповідають точкам, в яких функція приймає одиничні значення, знизу в дужках наведені індекси точок (кількості одиниць).

Далі необхідно згрупувати мінтерми, розташовуючи їх в порядку зростання індексів. Це доцільно представити у вигляді таблиці склеювання показаної нижче (табл. 9.2).

Таблиця 9.2 – Таблиця склеювання для функції (9.4)

Індекси	Номери	Результат склеювання		
(1)	$0001_{(1)}$	$00\text{-}1_{(1,3)}$ $0\text{-}01_{(1,5)}$	$0\text{--}1_{(1,3,5,7)}$ * $0\text{--}1_{(1,5,3,7)}$ *	–
(2)	$0011_{(3)}$ $0101_{(5)}$ $1100_{(12)}$ *	$0\text{-}11_{(3,7)}$ * $\text{-}011_{(3,11)}$ $01\text{-}1_{(5,7)}$		
(3)	$0111_{(7)}$ $1011_{(11)}$			

Після складання матриці, на першому етапі необхідно провести склеювання, користуючись наведеним вище формулюванням алгоритму. Пари мінтермів, що мають склеюватись показані стрілками. При склеюванні змінні, що не співпадають та відповідні їм розряди відмічаються прочерками. Наприклад, склеювання мінтермів заданих номерами $0001_{(1)}$ та $0011_{(3)}$ дає номер $00\text{-}1_{(1,3)}$, де в дужках записано номери відповідних мінтермів та контермів у десятковій системі числення. Результати склеювання записуються в другий стовпчик матриці, який також розділяється за рядками з індексами, що відрізняються на одиницю. Після склеювання першого стовпчика матриці переходять до другого, вписуючи результати склеювання вже у третій стовпчик. При об'єднанні мінтермів другого та наступних стовпчиків, *можна склеювати* лише ті

мінтерми (номери), які *містять прочерки в однакових позиціях*. Склеювання здійснюється доти, доки утворення нового стовпчика стане неможливим. Мін- та контерми, які не склеюються позначені зірочками.

На другому етапі, як і у методі Квайна переходять до побудови імплікантної матриці, записуючи у неї в якості простих імплікант набори, що утворилися в останньому стовпчику матриці склеювання. Крім того, у якості простих імплікант також записують набори і з інших стовпчиків, які не приймали участі у склеюванні (позначені зірочками). Якщо імпліканта, що міститься в *i*-му рядку складає деяку частину мінтерма (конституенти одиниці) *j*-го стовпця, то на перетині *i*-го рядка та *j*-го стовпця робиться позначка. Для отримання мінімальної ДНФ із імплікантної матриці необхідно вибрати мінімальну кількість рядків, щоб для кожного стовпчика, серед обраних рядків знайшовся хоча б один, що містить позначку.

Імплікантна матриця для прикладу, що розглядається матиме вигляд показаний у табл. 9.3.

Таблиця 9.3 – Імплікантна матриця для функції (9.4)

Прості імпліканти	Мінтерми					
	$\bar{x}_4\bar{x}_3\bar{x}_2x_1$	$\bar{x}_4x_3\bar{x}_2x_1$	$x_4x_3\bar{x}_2\bar{x}_1$	$\bar{x}_4x_3x_2x_1$	$x_4\bar{x}_3x_2x_1$	$\bar{x}_4\bar{x}_3x_2x_1$
$x_4x_3\bar{x}_2\bar{x}_1$			*			
$\bar{x}_3x_2x_1$					*	*
$\bar{x}_4x_1$	*	*		*		*

Отже отримана після мінімізації функція може бути записана в наступному вигляді:

$$f(\nu) = \bar{x}_4x_1 \vee \bar{x}_3x_2x_1 \vee x_4x_3\bar{x}_2\bar{x}_1.$$

9.3. Мінімізація логічних функцій методом карт Карно-Вейча

Одним з графічних способів задання двійкових функцій є карти Карно. Їх різновид – діаграми Вейча будуються як розгортки кубів (будь-яка двійкова функція у просторі може бути представлена *n*-мірним кубом) на площині. Далі під картами Карно та діаграмами Вейча будемо розуміти одне й те саме.

Карта Карно функції n змінних $f(\nu)$ є прямокутною таблицею, яка розбита на 2^n клітинок. Число рядків такої таблиці може не співпадати з кількістю стовпців. Кожна клітинка карти відповідає певному набору ν_i та містить відповідне значення функції $f(\nu_i)$. При цьому, очевидно, кожній

клітинці також відповідатиме певний мінтерм K_i, де i є десятковим номером клітинки, а отже і точки n-мірного куба.

Обов'язковою умовою побудови карти Карно для мінімізації функцій є те, що кожна клітинка повинна бути оточена *сусідніми клітинками*. Дві *клітинки називаються сусідніми*, якщо на карті вони межують одна з одною (тобто мають спільну границю), а їм відповідають *сусідні мінтерми*. При цьому вважається, що сусідніми є клітинки не тільки в середині карти, але й по краях карти Карно, якщо вони знаходяться в одному й тому ж рядку чи стовпчику.

Так, карта Карно для функції 3-х змінних може мати вигляд показаний на рис. 9.1.

	x_3		$\bar{x}_3$	
x_1	$x_3\bar{x}_2x_1$ $f(101)$ **5**	$x_3x_2x_1$ $f(111)$ **7**	$\bar{x}_3x_2x_1$ $f(011)$ **3**	$\bar{x}_3\bar{x}_2x_1$ $f(001)$ **1**
$\bar{x}_1$	$x_3\bar{x}_2\bar{x}_1$ $f(100)$ **4**	$x_3x_2\bar{x}_1$ $f(110)$ **6**	$\bar{x}_3x_2\bar{x}_1$ $f(010)$ **2**	$\bar{x}_3\bar{x}_2\bar{x}_1$ $f(000)$ **0**
	$\bar{x}_2$	x_2		$\bar{x}_2$

Рисунок 9.1 – Загальний вигляд карти Карно (діаграми Вейча) для функції 3-х змінних

Де зверху в клітинках записані мінтерми K_i, посередині значення функції f на відповідних наборах аргументів ν_i, знизу жирним наведено номери клітинок i.

Клітинки карти, які містять одиничне значення функції $f(\nu_i) = 1$ називаються 1-*клітинками*, а нульове $f(\nu_i) = 0$ – відповідно 0-*клітинками*.

Приклад.

Функція 3-х змінних задана таблицею істинності. Створити і заповнити для неї карту Карно.

i	x_3	x_2	x_1	$f(\nu_i)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

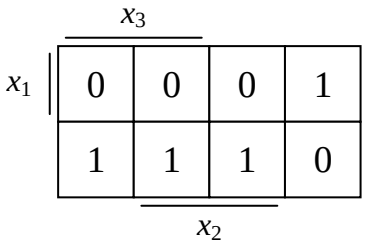
ДДНФ:

$$f(\nu) = \bar{x}_3\bar{x}_2x_1 \vee \bar{x}_3x_2\bar{x}_1 \vee x_3\bar{x}_2\bar{x}_1 \vee x_3x_2\bar{x}_1$$

ДКНФ:

$$f(\nu) = (x_3 \vee x_2 \vee x_1)(x_3 \vee \bar{x}_2 \vee \bar{x}_1)(\bar{x}_3 \vee x_2 \vee \bar{x}_1)(\bar{x}_3 \vee \bar{x}_2 \vee \bar{x}_1)$$

Карта Карно:

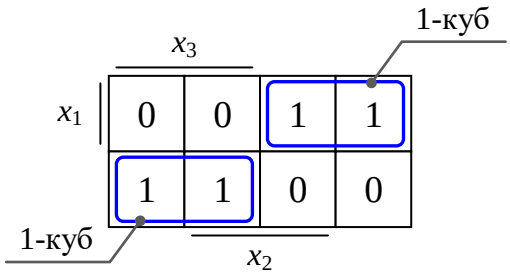


Мінімізація двійкових функцій за допомогою карт Карно виконується шляхом використання основної їх властиві: у карті Карно 1-клітинки будь-якого контерма $K_{i,j}(\nu)$ утворюють область, що має прямокутну форму. В дану область може входити 2^m елементів, а її саму називають m-кубом, $m = 0, 1, \ldots, n$. Щоб записати контерм, що відповідає заданому m-кубу необхідно за допомогою кон'юнкції об'єднати всі первинні терми, які *не змінюються* в області цього m-куба.

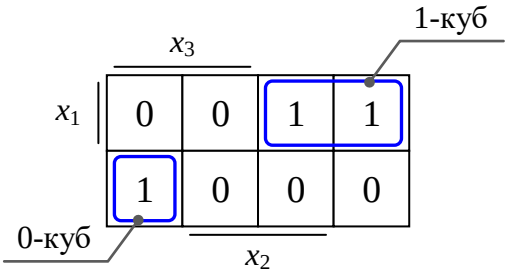
Таким чином, пошук мінімальної диз'юнктивної нормальної форми (МДНФ) функції зводиться до відшукування на карті Карно *мінімальної кількості* m-кубів *максимального розміру* та запису диз'юнкції контермів, що відповідають даним кубам. При визначенні m-кубів їх області можуть перетинатись, накладаючись одна на одну. Крім того, в m-куби можна об'єднувати крайні поля, що розташовані на протилежних сторонах карти в горизонтальному та/або вертикальному напрямках.

Приклади.

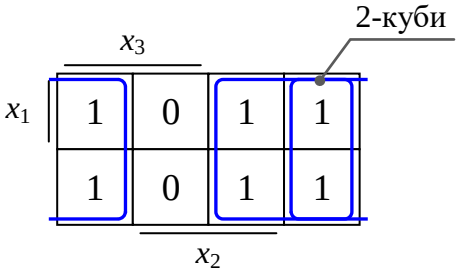
Записати МДНФ функцій за заданих картами Карно.



МДНФ: $f(x_3, x_2, x_1) = x_3\bar{x}_1 \vee \bar{x}_3x_1$



МДНФ: $f(x_3, x_2, x_1) = x_3\bar{x}_2\bar{x}_1 \vee \bar{x}_3 x_1$



МДНФ: $f(x_3, x_2, x_1) = \bar{x}_2 \vee \bar{x}_3$

Карти Карно для логічної функції 4-х змінних наведена на рис. 9.2.

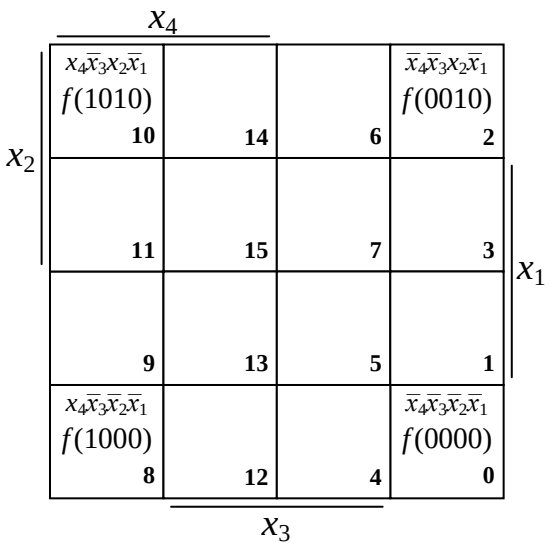


Рисунок 9.2 – Загальний вигляд карти Карно (діаграми Вейча) для функції 4-х змінних із вказаними для кожної клітинки номерами мінтермів, що їм відповідають

Карти Карно застосовуються здебільшого для мінімізації функцій вручну. На практиці зрідка доводиться мати справу з функціями, залежними від більше, ніж чотирьох змінних. Проте все ж наведемо приклади карт Карно для функцій більшої кількості змінних.

Карти Карно для більш ніж чотирьох змінних отримують із карт Карно чотирьох змінних [1]. Зокрема, карта Карно для *п'яти* змінних має вигляд, показаний на рисунку 9.3.

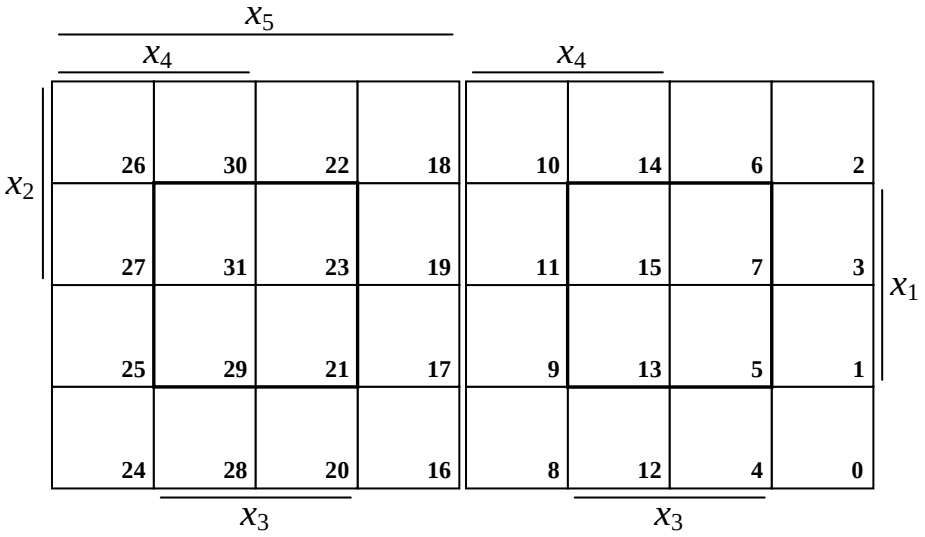
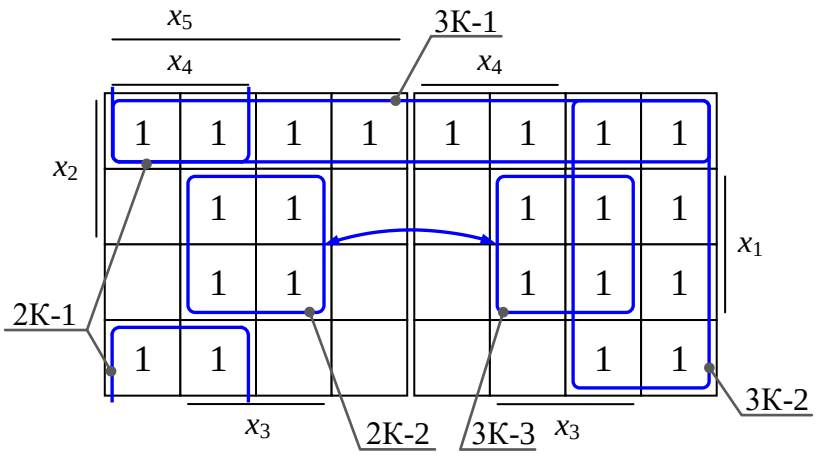


Рисунок 9.3 – Карта Карно для функції п'яти змінних

Області обведені жирним також містять сусідні клітинки (мінтерми), що слід враховувати при пошуку максимальних *m*-кубів. Детальніше розглянемо це на прикладі.

Приклад.

Мінімізувати функцію п'яти змінних за заданою картою Карно.



МДНФ: $f(\nu) = \overbrace{x_2\bar{x}_1}^{\text{3К-1}} \vee \overbrace{\bar{x}_5\bar{x}_4}^{\text{3К-2}} \vee \overbrace{x_5x_4\bar{x}_1}^{\text{2К-1}} \vee \overbrace{\underbrace{x_5x_3x_1}_{x_3x_1}}^{\text{2К-2}} \vee \overbrace{\bar{x}_5x_3x_1}^{\text{2К-3}}$

Карта Карно для функції *шести* змінних наведена на рис. 9.4.

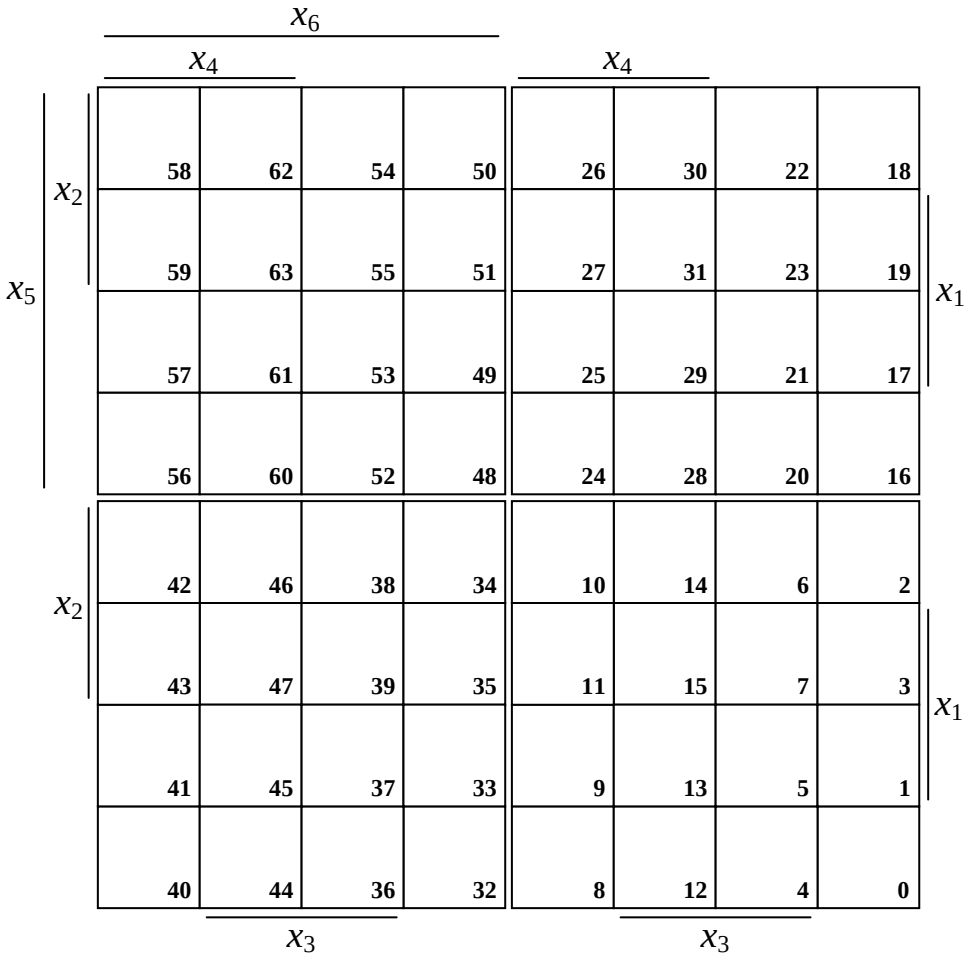
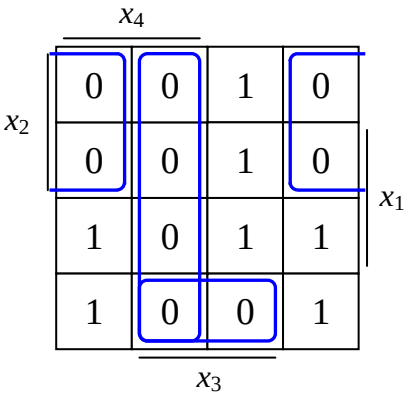


Рисунок 9.4 – Карта Карно для функції шести змінних

Отримання мінімальної кон'юнктивної нормальної форми (МКНФ)

Для отримання МКНФ функції за допомогою карт Карно (діаграм Вейча) необхідно віднайти на карті мінімальну кількість максимальних за розміром *m*-кубів, які покривають 0-*клітинки* (вони фактично відповідають інверсній функції) та записати кон'юнкції *дизтермів*, що відповідають даним кубам. Щоб записати дизтерм, що відповідає заданому *m*-кубу необхідно *проінвертувати* та за допомогою диз'юнкції об'єднати всі первинні терми, які *не змінюються* в області цього *m*-куба.

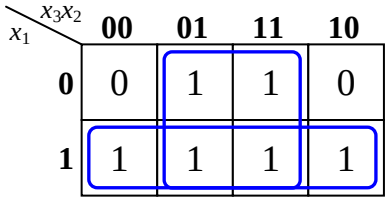
Приклад.



МКНФ:

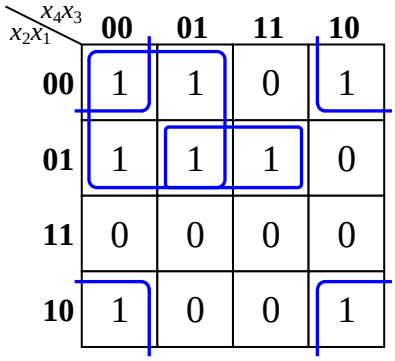
$$f(\nu) = (x_3 \vee \bar{x}_2)(\bar{x}_4 \vee \bar{x}_3)(\bar{x}_3 \vee x_2 \vee x_1).$$

Карти Карно також можуть бути представленні в дещо іншому вигляді (див. рис. 9.5): замість рисок, які вказують на поля клітинок, що відповідають змінним, тут застосовуються двійкові комбінації, які одразу визначають точки функції.



МДНФ, МКНФ:

$$f(\nu) = x_1 \vee x_2.$$



МДНФ:

$$f(\nu) = \bar{x}_3\bar{x}_1 \vee \bar{x}_4\bar{x}_2 \vee x_3\bar{x}_2x_1.$$

МКНФ:

$$f(\nu) = (\bar{x}_2 \vee \bar{x}_1)(\bar{x}_3 \vee \bar{x}_2)(\bar{x}_4 \vee \bar{x}_3 \vee x_1)(\bar{x}_4 \vee x_3 \vee \bar{x}_1).$$

Рисунок 9.5 – Альтернативне представлення карт Карно

Слід зауважити, що при цьому порядок запису двійкових комбінацій для сусідніх комірок відповідає *не звичайній двійковій послідовності*, а *коду Грея*. Це не дивно, оскільки сусідні клітинки карт Карно відповідають сусіднім мінтермам, а отже характеризуються тим, що точки функції, для яких записані сусідні мінтерми будуть відрізнятися *лише одним бітом*, що власне і забезпечує код Грея.

9.4. Мінімізація неповністю визначених функцій

Нагадаємо, що якщо в деяких точках області визначення значення функції не задані, то така функція називається *неповністю визначеною*. ДДНФ такої функції можна представити в наступному вигляді:

$$f(\nu) = \bigvee_{i_1} K_{i_1}(\nu) \vee \Phi \cdot \left[\bigvee_{i_\Phi} K_{i_\Phi}(\nu) \right], \qquad (9.5)$$

де $\nu = (x_n, \ldots, x_1)$, i_1 – номери точок, в яких функція приймає одиничне значення $f(\nu_{i_1}) = 1$, i_Φ – номери точок, в яких функція приймає невизначене значення $f(\nu_{i_\Phi}) = \Phi$, $\Phi = \{0 \text{ або } 1\}$.

Якщо значення функції $f(\nu)$ не задано в m точках, то її можна довизначити 2^m різними способами. Основна задача мінімізації функції в даному випадку полягає у пошуку оптимального варіанта її довизначення, який дозволяє отримати мінімальну із усіх можливих ДНФ або КНФ серед всіх 2^m варіантів її довизначення [1, 3, 7]. При цьому, як і у випадку мінімізації повністю визначеної функції, може бути отримано декілька рівноцінних МДНФ та МКНФ. Розглянемо ідею мінімізації неповністю визначених функцій на прикладі.

Приклад.

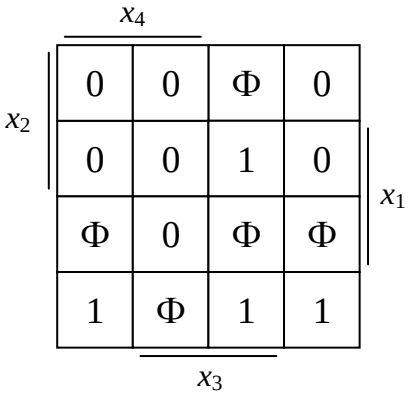
Нехай таблицею істинності задано неповністю визначену функцію $f(\nu)$ 4-х змінних:

i	x_4	x_3	x_2	x_1	$f(\nu)$
0	0	0	0	0	1
1	0	0	0	1	Ф
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	Ф
6	0	1	1	0	Ф
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	Ф
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	Ф
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

ДДНФ даної функції буде наступною:

$$f(\nu) = K_0 \vee K_4 \vee K_7 \vee K_8 \vee \Phi[K_1 \vee K_5 \vee K_6 \vee K_9 \vee K_{12}].$$

Карта Карно даної функції матиме вигляд:



Якщо деяким Ф присвоїти значення «1», то можна отримати два варіанти мінімізації функції, показані нижче на рис. 9.6.

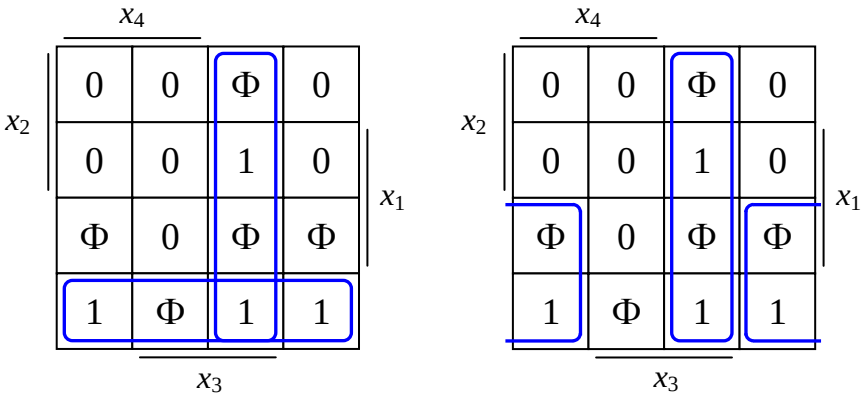


Рисунок 9.6 – Карти Карно та *m*-куби для довизначених одиницями функцій

Тоді МДНФ, що відповідають даним варіантам матимуть вигляд:

$$f(\nu) = \bar{x}_4 x_3 \vee \bar{x}_2 \bar{x}_1,$$

$$f(\nu) = \bar{x}_4 x_3 \vee \bar{x}_3 \bar{x}_2.$$

Хоча отримані вирази для МДНФ відрізняються, вони забезпечують виконання алгоритму функціонування, заданого таблицею істинності.

Аналогічним чином можна знайти МКНФ даної неповністю визначеної функції, виконуючи оптимальне довизначення вже інверсної функції $\overline{f(\nu)}$. Нагадаємо, що для пошуку МКНФ за допомогою карт Карно необхідно об'єднувати в *m*-куби 0-клітинки, адже саме вони відповідають інверсній функції та записати кон'юнкцію дизтермів, що відповідають даним кубам. У випадку неповністю визначеної функції значення Ф доповнюються «0» так, щоб отримати оптимальне покриття *m*-кубами. У

прикладі, що розглядається таких мінімальних покриттів можливо тільки одне, воно показане на рис. 9.7.

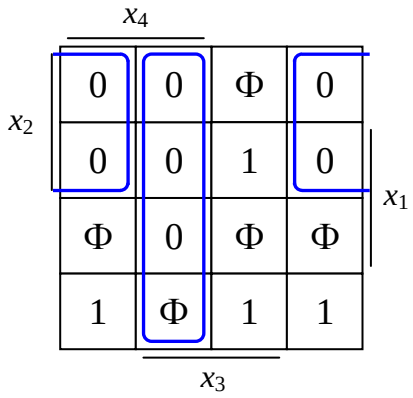


Рисунок 9.7 – Карта Карно та *m*-куби для довизначених нулями функції

Тому МКНФ матиме вигляд:

$$f(\nu) = (\bar{x}_4 \vee \bar{x}_3)(x_3 \vee \bar{x}_2).$$

Зауважимо, що в результаті мінімізації неповністю визначених функцій завжди отримують *повністю* визначені функції.

9.5. Спільна мінімізація декількох функцій

Доволі часто виникає потреба у вирішенні задачі одночасної мінімізації декількох логічних функцій [1]. Дана задача, зокрема, має місце при синтезі логічних пристроїв з декількома виходами.

В загальному випадку означена задача полягає в пошуку спільного покриття всіх функцій мінімальним числом *m*-кубів максимального розміру. Особливість тут полягає в тому, що незалежна мінімізація кожної функції, як правило, не дає найкращого результату в сенсі мінімізації сумарного числа первинних термів. Зробимо пояснення сказаного на прикладі.

Приклад.

Нехай МДНФ двох функцій $f_1(\nu)$ та $f_2(\nu)$ мають вигляд:

$$f_1(\nu) = x_4\bar{x}_3 \vee x_4x_2,$$

$$f_2(\nu) = \bar{x}_4x_3 \vee x_3x_2.$$

Карти Карно, що відповідають цим МДНФ мають вигляд, як показано на рисунку 9.8.

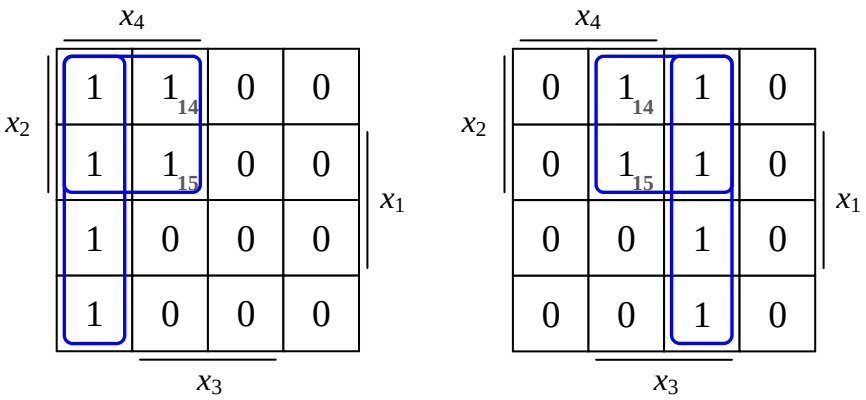


Рисунок 9.8 – Карти Карно, що відповідають заданим МДНФ функцій

Аналізуючи дві ці карти Карно можна бачити, що при подачі на вхід функції комбінацій $1110_2 = 14_{10} : x_4x_3x_2\bar{x}_1$ та $1111_2 = 15_{10} : x_4x_3x_2x_1$ вони приймають однакове значення, а це означає, що їх можна об'єднати по спільним клітинкам. При цьому імплікантою $x_4x_3x_2$, що відповідає даним клітинкам можна замінити в МДНФ останні коннтерми, тобто:

$$f_1(\nu) = x_4\bar{x}_3 \vee x_4x_3x_2\,,$$

$$f_2(\nu) = \bar{x}_4x_3 \vee x_4x_3x_2\,.$$

На перший погляд може здатися, що після проведених маніпуляцій функції ускладнились, однак якщо врахувати, що другий контерм можна реалізувати один раз, а використати двічі, то стає зрозуміло, що схема, яка реалізує обидві функції спростилася. Для ілюстрації реалізуємо обидва варіанти функцій у вигляді логічних схем (див. рис. 9.9).

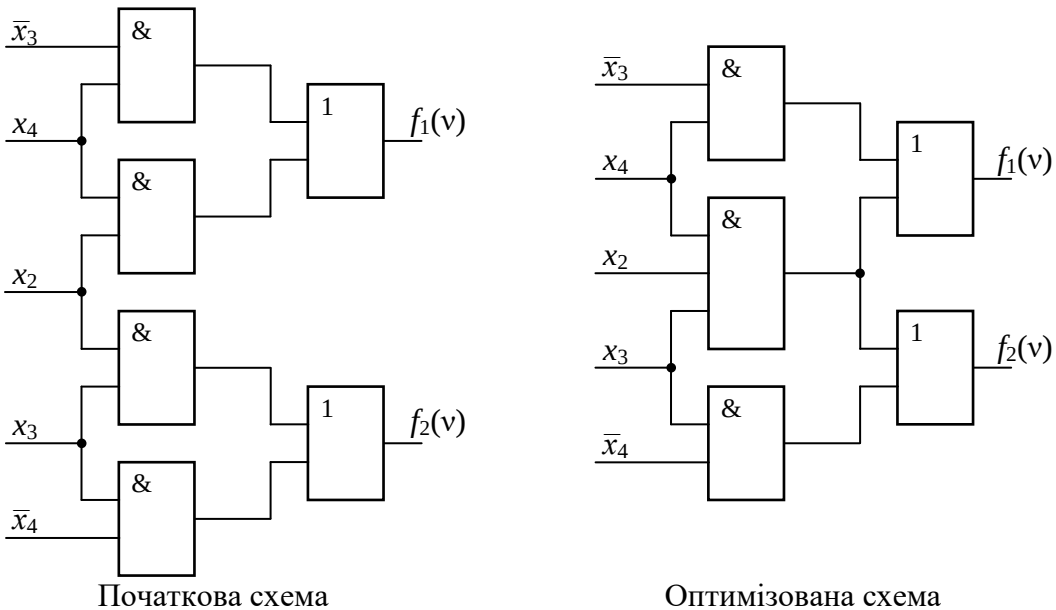


Рисунок 9.9 – Схеми, що реалізують початкову та оптимізовану МДНФ

Якщо порівняти схеми з рис. 9.4, то можна помітити, що оптимізована схема має на один логічний елемент менше, а на її вхідні логічні елементи подаються 7 первинних термів, в той час, як у початковій схемі – 8.

Таким чином, у випадку необхідності одночасної мінімізації декількох функцій одних і тих самих змінних, може бути доцільно виконувати оптимізацію одночасно по всім функціям, а не окремо по кожній з них.

9.6. Факторизовані форми функції

Максимальне число логічних операцій, що виконуються для реалізації функції $f(\nu)$ називається *порядком перемикальної функції*.

Наприклад, МДНФ функція виду:

$$f(\nu) = x_4x_3x_2 \vee x_4x_3x_1 \vee x_4x_2x_1, \qquad (9.6)$$

має другий порядок, оскільки в ній спочатку виконується кон'юнкції для формування контермів, а потім – операції диз'юнкції – для їх об'єднання.

Максимальна кількість послідовно з'єднаних логічних елементів, необхідних для реалізації даної функції $f(\nu)$, називається *порядком комбінаційної схеми*. Наприклад, наведеній вище МДНФ функції 2-го порядку (10.6) відповідає комбінаційна схема 2-го порядку (рис. 9.10).

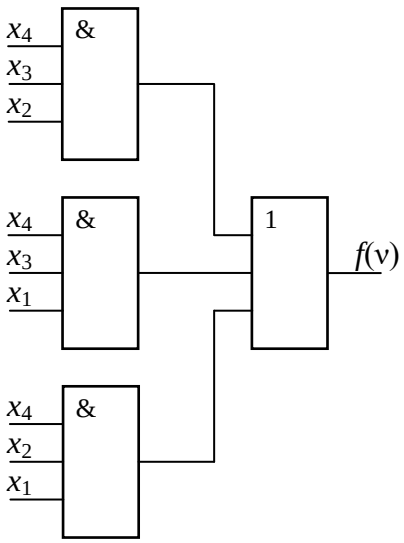


Рисунок 9.10 – Схема, що відповідає МДНФ функції (9.6)

Таким чином, порядки комбінаційних схем та функцій, що їх реалізують збігаються. Користуючись законом дистрибутивності, винесемо за дужки однакові первинні терми в МДНФ [1]:

$$f(\nu) = x_4 \cdot [x_3 \cdot (x_2 \vee x_1) \vee x_2x_1]. \qquad (9.7)$$

Тепер функція $f(\nu)$, як і схема, що її реалізує (рис. 9.11) матиме 4-й порядок.

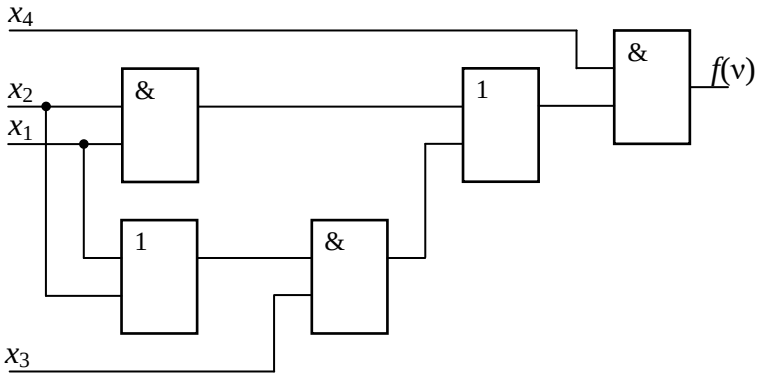


Рисунок 9.11 – Схема факторизованої форми функції (9.7)

Як видно, в наведеній вище схемі використовуються логічні елементи меншої розрядності. Загальна кількість входів логічних елементів у ній становить 10, в той час, як у початковій схемі входів було 12. Отримане представлення функції вже *не є нормальною формою*, натомість його називають *факторизованою або дужковою формою* представлення функцій. Факторизована (дужкова) форма представлення функцій зазвичай використовується для зниження вартості реальної схеми, а також у випадках, коли використання логічних елементів з великою кількістю входів є таким, що не може бути практично реалізоване.

В той же час отримана функція володітиме меншою швидкодією, оскільки кожен реальний логічний елемент має кінцеву швидкодію, що характеризується часом затримки розповсюдження сигналу τ. Очевидно, що у схеми більшого порядку загальний час розповсюдження сигналу буде вищим.

9.7. Синтез комбінаційних схем на мажоритарних елементах

Мажоритарними елементами називаються логічні елементи, що мають непарну кількість входів x_p з однаковим пріоритетом [1, 4]:

$$f(\nu) = \begin{cases} 1, & \text{якщо } \sum x_p \geq k, \\ 0, & \text{якщо } \sum x_p < k, \end{cases}$$

де $\nu = (x_n, \ldots, x_1)$; $p = 1, 2, \ldots, n$; $k = \frac{n+1}{2}$ – пороговий рівень. Мажоритарні елементи зазвичай використовуються для високонадійної передачі даних за рахунок принципу резервування.

Найбільш розповсюдженими є 3-входові мажоритарні елементи. Одним із їх прикладів є елемент, що реалізує наступну функцію:

$$f(\nu) = x_3x_2 \vee x_3x_1 \vee x_2x_1.$$

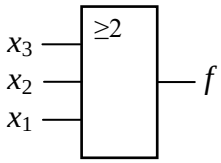
Функція $f(\nu)$, дорівнює тому значенню, які приймають два із трьох її вхідних сигналів (аргументів) x_p, $p = 1, 2, 3$. Таблиця істинності 3-входового мажоритарного елемента показана нижче.

x_3	x_2	x_1	f	a_2	a_1
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

В таблиці істинності наведено, як вихідне значення мажоритарного елемента f при різних значеннях входу (x_3, x_2, x_1), так і сигнали a_2 та a_1, що кодують номер лінії, на якій імовірно виникла помилка (комбінація $a_2a_1 = 00$ означає відсутність помилки). Не складно впевнитись, що функції, які реалізують сигнали a_2 та a_1 мають вигляд:

$$\begin{cases} a_2 = x_3 \oplus x_2, \\ a_1 = x_3 \oplus x_1. \end{cases}$$

Мажоритарний елемент позначається як показано нижче.



На основі мажоритарних елементів також можуть будуватися довільні логічні функції. Розглянемо функцію n змінних $f(\nu) = f(\nu', x_p, x_q)$, де $\nu = (x_p, \ldots, x_q)$, а ν' – це множина змінних, з якої виключені x_p та x_q. Тоді за теоремою розкладу Шеннона:

$$f(\nu', x_p, x_q) = \bar{x}_p \cdot f(\nu', 0, x_q) \vee x_p \cdot f(\nu', 1, x_q) = a_0 \vee b_0$$

де

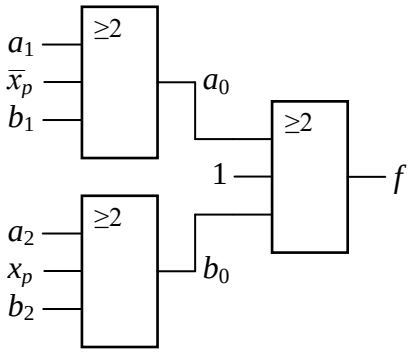
$$a_0 = \bar{x}_p \cdot f(\nu', 0, x_q) = \bar{x}_p \cdot [\underbrace{\bar{x}_q f(\nu', 0, 0)}_{a_1} \vee \underbrace{x_q f(\nu', 0, 1)}_{b_1}] = \bar{x}_p (a_1 \vee b_1),$$

$$b_0 = x_p \cdot f(\nu', 1, x_q) = x_p \cdot [\underbrace{\bar{x}_q f(\nu', 1, 0)}_{a_2} \vee \underbrace{x_q f(\nu', 1, 1)}_{b_2}] = x_p (a_2 \vee b_2).$$

Враховуючи, що $a_r b_r = 0$, для $r = 0, 1, 2$ і підставляючи на входи мажоритарних елементів $x_3 = a_r$, $x_2 = \{x_p$, або $\bar{x}_p$, або $1\}$ та $x_1 = b_r$ можна записати наступні співвідношення:

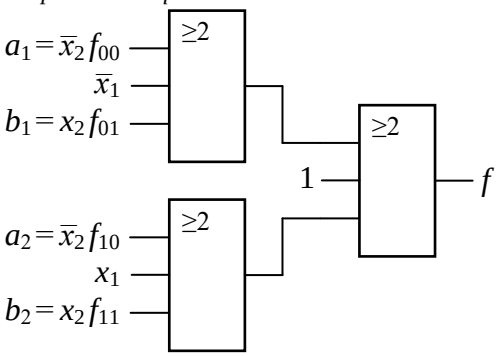
$$a_0 = a_1\bar{x}_p \vee a_1 b_1 \vee \bar{x}_p b_1 = a_1 b_1 \vee \bar{x}_p(a_1 \vee b_1),$$
$$b_0 = a_2 x_p \vee a_2 b_2 \vee x_p b_2 = a_2 b_2 \vee x_p(a_2 \vee b_2),$$
$$f = a_0 \cdot 1 \vee a_0 b_0 \vee 1 \cdot b_0 = a_0 b_0 \vee (a_0 \vee b_0).$$

Із чого слідує, що наведені вище співвідношення можуть бути реалізовані на трьохвходових мажоритарних елементах:



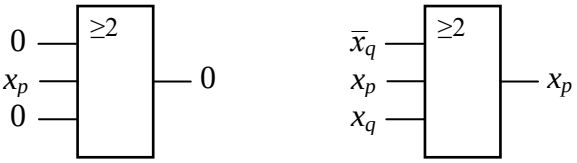
Якщо продовжити розклад функції n змінних $f(\nu)$ за іншими змінними, то на останньому етапі отримаємо значення $a_i = f(\nu_i) = \{0$ або $1\}$. Відповідна їм комбінаційна схема буде містити n каскадів з $2^n - 1$ трьохвходовими мажоритарними елементами.

Наведена вище схема, зокрема, реалізує функцію двох змінних, оскільки, якщо позначити $x_p = x_1$, $x_q = x_2$, то:



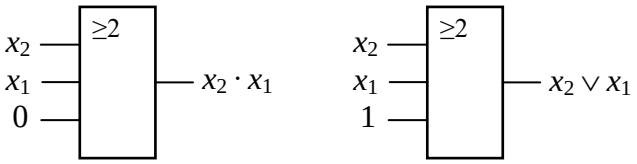
де $f_{ij} = f(i, j)$ – значення функції у точці $(i, j) = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$.

При побудові функцій n змінних за наведеним принципом може виявитись, що на два з трьох входів мажоритарних елементів подаватимуться значення рівні 0 (наприклад, якщо б $f_{00} = f_{01} = 0$), або значення x_q та $\bar{x}_q$ (якщо б мало місце $f_{00} = f_{01} = 1$). При цьому в першому випадку мажоритарний елемент завжди видаватиме 0, а в другому – x_p, що дозволяє спростити схему, відкинувши цей мажоритарний елемент.

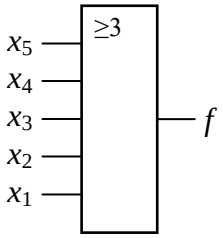


Оскільки розклад Шеннона можна застосувати до будь-яких двійкових функцій $f(\nu)$, то і будь-яку функцію можна реалізувати на мажоритарних елементах. При цьому, в ході послідовного розкладу $f(\nu)$ буде отримано її факторизоване (дужкове) представлення. Водночас, основна складність синтезу схем на базі мажоритарних елементів полягає у мінімізації їх стандартної структури.

Для трьохвходових мажоритарних елементів нескладно переконатися у справедливості наступних тотожностях:



Зауважимо також, що застосовуються мажоритарні елементи з більшою кількістю входів, зокрема з п’ятьма. Позначення такого елементу наведено нижче.



Наразі мажоритарні елементи представляють інтерес здебільшого для організації надійних каналів передачі двійкових даних.

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Охарактеризуйте етапи мінімізації логічних функцій методом Квайна. На яких законах алгебри логіки основані ці два етапи?
2. Що називається імплікантою?
3. Для чого застосовується імплікантна матриця (таблиця)?
4. В чому основна відмінність методу Квайна-Мак-Класкі від метода Квайна?
5. Якою основною властивістю володіють клітинки карти Карно (діаграми Вейча), що дозволяє використовувати дані діаграми для мінімізації логічних функцій?
6. Як саме і які саме клітинки слід обводити в Карті Карно (діаграмі Вейча), щоб отримати мінімальні диз’юнктивну та мінімальну кон’юнктивну нормальні форми?

7. Наведіть карти Карно (діаграми Вейча) для логічних функцій трьох та чотирьох змінних. Яким чином отримують карти Карно для функцій більш ніж чотирьох змінних?
8. В чому полягають особливості мінімізації неповністю визначених логічних функцій?
9. Якою є основна особливість мінімізації одночасно декількох логічних функцій?
10. В чому полягають переваги та недоліки факторизованого представлення логічних функцій?
11. Що таке мажоритарний елемент? Яке їх призначення і для чого мажоритарні елементи найчастіше використовуються сьогодні?
12. Користуючись методом Квайна-Мак-Класкі знайти МДНФ функції чотирьох змінних, яка приймає одиничні значення у точках:

$$f(\nu) = \{0, 1, 2, 3, 4, 6, 8, 9, 11, 14\},$$

номери точок задані в десятковій системі числення.

13. Користуючись методом Квайна-Мак-Класкі знайти МДНФ функції чотирьох змінних:

$$f(\nu) = \bar{x}_4\bar{x}_3\bar{x}_2x_1 \vee \bar{x}_4x_3\bar{x}_2x_1 \vee \bar{x}_4\bar{x}_3x_2x_1 \vee x_4\bar{x}_3\bar{x}_2x_1 \vee x_4\bar{x}_3x_2x_1 \vee \bar{x}_4x_3x_2x_1 \vee x_4x_3x_2x_1$$

14. Функція 3-х змінних задана картою Карно. Запишіть МДНФ та МКНФ даної функції.

x_1 \ $\overline{x_3}$				
	1	1	0	1
	0	0	0	1

$\overline{x_2}$

15. Неповністю визначена функція 4-х змінних задана картою Карно. Запишіть МДНФ та МКНФ цієї функції:

x_2 \ $\overline{x_4}$				
	0	Ф	Ф	0
	0	0	1	0
x_1	Ф	0	Ф	Ф
	1	0	1	1

$\overline{x_3}$

16. Функція приймає одиничні значення у точках 0001, 0101, 0111, 1011 та 0011. Виконайте мінімізацію даної функції за допомогою карти Карно (діаграми Вейча).
17. Неповністю визначена функція задана таблицею істинності:

x_3	x_2	x_1	f
0	0	0	Ф
0	0	1	0
0	1	0	Ф
0	1	1	1
1	0	0	Ф
1	0	1	Ф
1	1	0	1
1	1	1	0

10. СХЕМИ ДЕЯКИХ НАЙБІЛЬШ ПОШИРЕНИХ КОМБІНАЦІЙНИХ ПРИСТРОЇВ

10.1. Демультиплексор

Вище було розглянуто комбінаційний логічний пристрій, який називається мультиплексором. Нагадаємо, що він має 2^n інформаційних та n керуючих входів, а також один вихід. На в*и*хід мультиплексора передається інформаційний сигнал із того входу, номер якого відповідає двійковій комбінації, що подана на керуючі входи даного мультиплексора.

Якщо є мультиплексор, то, очевидно, повинен існувати і пристрій, який функціонує протилежним чином. Такий пристрій називається *демультиплексором*. На відміну від мультиплексора він має єдиний інформаційний вхід, сигнал з якого передається на один з 2^n в*и*ходів, що обирається двійковою комбінацією на керуючих входах [1, 3, 6].

Як приклад розглянемо таблицю істинності чотирьохрозрядного демультиплексора. Вона наведена нижче.

входи			**виходи**			
x_1	x_0	y	f_3	f_2	f_1	f_0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0

Із наведеної таблиці істинності видно, що кожен із вихідних сигналів може бути визначений одним мінтермом:

$$f_0 = \bar{x}_1\bar{x}_0y = (\bar{x}_1\bar{x}_0) \cdot y,$$
$$f_1 = \bar{x}_1x_0y = (\bar{x}_1x_0) \cdot y,$$
$$f_2 = x_1\bar{x}_0y = (x_1\bar{x}_0) \cdot y,$$
$$f_3 = x_1x_0y = (x_1x_0) \cdot y.$$

Записані в дужках вирази еквівалентні виходам повного дешифратора, таким чином демультиплексори також можна реалізовувати за допомогою дешифраторів та логічних елементів І (див. рис. 10.1). Тим

не менш, при такому підході, забезпечуватиметься менша швидкодія схеми, ніж при безпосередній реалізації на основі мінтермів.

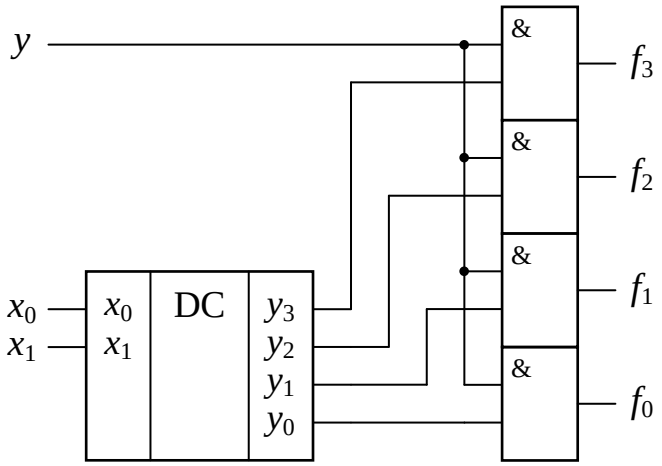


Рисунок 10.1 – Схема чотирьох розрядного демультиплексора

Умовне графічне позначення демультиплексорів, що використовується на схемах згідно стандартів ГОСТ та ANSI, наведено нижче на рис. 10.2.

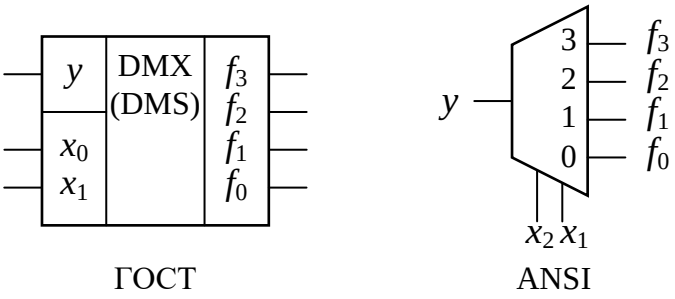


Рисунок 10.2 – Приклади умовних позначені демультиплексорів

Зауважимо, що аналогові мультиплексори є одночасно і демультиплексорами, оскільки сигнал в них може проходити в обидві сторони: як із входу на вихід, так і з виходу на вхід. Прикладами аналогових мультиплексорів/демультиплексорів є мікросхеми 561КП1, 74HC4051, 74HC4067, MAX14778.

10.2. Шифратори

10.2.1. Простий шифратор

Шифратор – це логічний пристрій, що виконує операцію протилежну до дешифратора: він перетворює позиційний унітарний 2^n-розрядний код на відповідне n-розрядне двійкове число [1, 3, 6].

Реалізацію звичайного шифратора розглянемо на прикладі його восьмивходової версії (вхідний унітарний код має 8 розрядів). Наведемо таблицю істинності такого шифратора.

x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Якщо припустити, що на вхід шифратора буде подаватись завжди вірний унітарний код, який міститиме лише одну одиницю, то можна помітити, що виходи шифратора приймають одиничні значення тоді, коли одиниці з'являються *хоча б* на одному певному вході, а саме:

$y_2 = 1$, коли або x_4, або x_5, або x_6, або x_7 дорівнюють одиниці;

$y_1 = 1$, коли або x_2, або x_3, або x_6, або x_7 дорівнюють одиниці;

$y_0 = 1$, коли або x_1, або x_3, або x_5, або x_7 дорівнюють одиниці.

Тоді функції, що реалізують шифратора матимуть вигляд:

$$y_2 = x_7 \vee x_6 \vee x_5 \vee x_4;$$

$$y_1 = x_7 \vee x_6 \vee x_3 \vee x_2;$$

$$y_0 = x_7 \vee x_5 \vee x_3 \vee x_1.$$

Схема, що реалізує наведені рівняння показана на рис. 10.3. Зауважимо, що вхід x_0 в схемі не використовується, але показаний для наочності.

Якщо виникає потреба у виявленні відсутності хоча б одного активного рівня на вході шифратора (на всі входи подані 0), то для цього може застосовуватись наступна проста схема на основі диз'юнкції:

$$AI = \bigvee_{i=0}^{2^n-1} x_i.$$

Вихідний сигнал *AI* (Active Input) приймає значення одиниці тоді, коли хоча б на один із входів подано одиницю.

В той же час, якщо на створений шифратор буде подано не унітарний код, на його виході формуватиметься некоректний сигнал. В цьому випадку застосовуються *пріоритетні шифратори*.

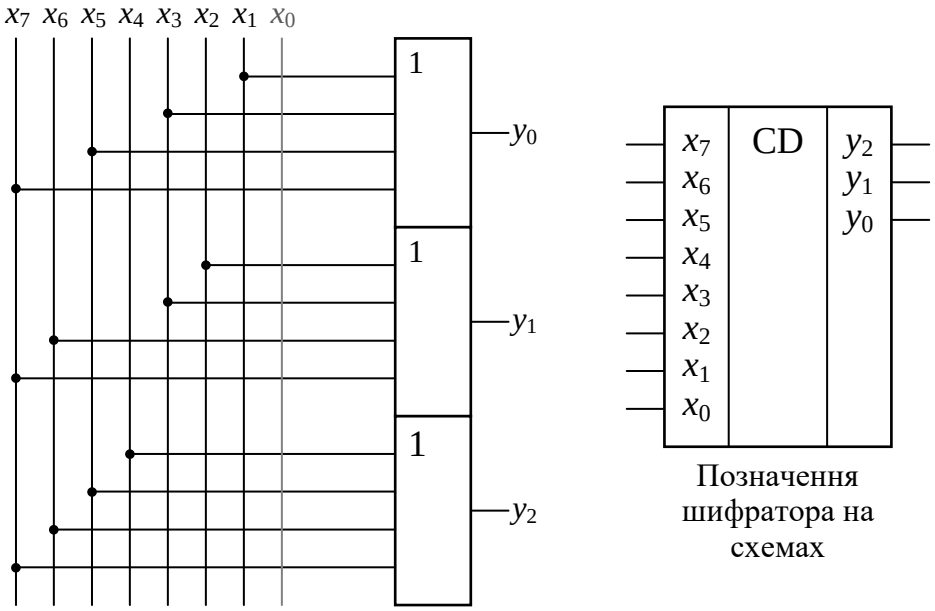


Рисунок 10.3 – Схема непріоритетного 8-входового шифратора та його позначення на схемах

10.2.2. Пріоритетний шифратор

Якщо на шифратор поступають сигнали від різних незалежних джерел, то умова наявності на його входах лише унітарного коду може вже не виконуватись. В такому випадку кожному вхідному сигналу присвоюється свій пріоритет. Дане присвоювання зазвичай виконується за зростанням номеру входу, рідше – за спаданням. При цьому шифратор видаватиме двійкове число i, якщо його вхід $x_i = 1$, а на всі інші входи x_j з більшим пріоритетом подані 0. Такий шифратор називається *пріоритетним*. Приклад таблиці істинності восьмивходового шифратора, в якому пріоритет зростає зі збільшенням номеру входу, показано нижче.

x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	Ф	0	0	1
0	0	0	0	0	1	Ф	Ф	0	1	0
0	0	0	0	1	Ф	Ф	Ф	0	1	1
0	0	0	1	Ф	Ф	Ф	Ф	1	0	0
0	0	1	Ф	Ф	Ф	Ф	Ф	1	0	1
0	1	Ф	Ф	Ф	Ф	Ф	Ф	1	1	0
1	Ф	Ф	Ф	Ф	Ф	Ф	Ф	1	1	1

Безпосередній запис значень функцій виходу по рядках таблиці істинності дає:

$$y_2 = x_7 \vee \overline{x}_7 \cdot x_6 \vee \overline{x}_7\overline{x}_6 \cdot x_5 \vee \overline{x}_7\overline{x}_6\overline{x}_5 \cdot x_4;$$
$$y_1 = x_7 \vee \overline{x}_7 \cdot x_6 \vee \overline{x}_7\overline{x}_6\overline{x}_5\overline{x}_4 \cdot x_3 \vee \overline{x}_7\overline{x}_6\overline{x}_5\overline{x}_4\overline{x}_3 \cdot x_2;$$
$$y_0 = x_7 \vee \overline{x}_7\overline{x}_6 \cdot x_5 \vee \overline{x}_7\overline{x}_6\overline{x}_5\overline{x}_4 \cdot x_3 \vee \overline{x}_7\overline{x}_6\overline{x}_5\overline{x}_4\overline{x}_3\overline{x}_2 \cdot x_1.$$

Згідно теореми розкладу Шеннона (5.1) та закону узагальненого склеювання (4.15) правомірно стверджувати, що:

$$x_p \vee f(x_n,\ldots,x_p,\ldots,x_1) = \overline{x}_p[0 \vee f(x_n,\ldots,0,\ldots,x_1)] \vee x_p[\underbrace{1 \vee f(x_n,\ldots,1,\ldots,x_1)}_{1}] =$$

$$= \underbrace{\overline{x}_p \cdot f(x_n,\ldots,0,\ldots,x_1) \vee x_p}_{\text{узагальнене склеювання}} = f(x_n,\ldots,0,\ldots,x_1) \vee x_p$$

тоді:

$$y_2 = x_7 \vee \overline{x}_7(x_6 \vee \overline{x}_6(x_5 \vee \overline{x}_5 \cdot x_4))) =$$
$$x_7 \vee \overline{0}(x_6 \vee \overline{0}(x_5 \vee \overline{0} \cdot x_4))),$$
$$y_1 = x_7 \vee \overline{x}_7(x_6 \vee \overline{x}_6(\overline{x}_5\overline{x}_4 \cdot x_3 \vee \overline{x}_5\overline{x}_4\overline{x}_3 \cdot x_2)) =$$
$$x_7 \vee \overline{0}(x_6 \vee \overline{0}(\overline{x}_5\overline{x}_4 \cdot x_3 \vee \overline{x}_5\overline{x}_4\overline{0} \cdot x_2)),$$
$$y_0 = x_7 \vee \overline{x}_7\overline{x}_6(x_5 \vee \overline{x}_5\overline{x}_4(x_3 \vee \overline{x}_3\overline{x}_2 \cdot x_1)) =$$
$$x_7 \vee \overline{0}\,\overline{x}_6(x_5 \vee \overline{0}\,\overline{x}_4(x_3 \vee \overline{0}\,\overline{x}_2 \cdot x_1))$$

та врешті:

$$y_2 = x_7 \vee x_6 \vee x_5 \vee x_4,$$
$$y_1 = x_7 \vee x_6 \vee \overline{x}_5\overline{x}_4 \cdot x_3 \vee \overline{x}_5\overline{x}_4 \cdot x_2,$$
$$y_0 = x_7 \vee \overline{x}_6 \cdot x_5 \vee \overline{x}_6\overline{x}_4 \cdot x_3 \vee \overline{x}_6\overline{x}_4\overline{x}_2 \cdot x_1.$$

Умовне позначення пріоритетних шифраторів на схемах таке ж, як і у непріоритетних за виключенням підпису «PRCD» (рис. 10.4)

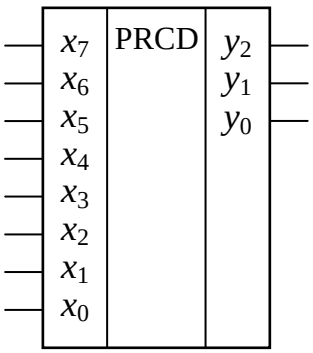


Рисунок 10.4 – Позначення 8-входового пріоритетного шифратора

Для створення пріоритетного шифратора також можна використати ще один підхід. Зокрема можна створити спеціальну схему, яка буде виконувати перетворення будь-якого двійкового коду на унітарний код, в якому позиція одиниці відповідатиме найбільш старшому ненульовому

біту двійкового коду. Функції, що реалізовуватимуть такий пристрій матимуть вигляд:

$$f_m = x_m;$$
$$f_{m-1} = \overline{x}_m \cdot x_{m-1};$$
$$f_{m-2} = \overline{x}_m \overline{x}_{m-1} \cdot x_{m-2} = \overline{x_m \vee x_{m-1}} \cdot x_{m-2};$$
$$\dots$$
$$f_1 = \overline{x}_m \overline{x}_{m-1} \cdot \ldots \cdot \overline{x}_2 x_1 = \overline{\overline{x}_m \vee \overline{x}_{m-1} \vee \ldots \vee \overline{x}_2} \cdot x_1;$$
$$f_0 = \overline{x}_m \overline{x}_{m-1} \cdot \ldots \cdot \overline{x}_2 \overline{x}_1 x_0 = \overline{\overline{x}_m \vee \overline{x}_{m-1} \vee \ldots \vee \overline{x}_2 \vee \overline{x}_1} \cdot x_0.$$

Дані функції отримані безпосередньо по лівій частині таблиці істинності пріоритетного шифратора з міркувань, що в правій частині (частині виходів) в кожному рядку мав би бути записаний унітарний позиційний код. За допомогою пристрою, що реалізує наведені вище функції, можна отримати пріоритетний шифратор на базі звичайного шифратора, подавши на всі входи останнього відповідні вихідні сигнали f_i (рис. 10.5).

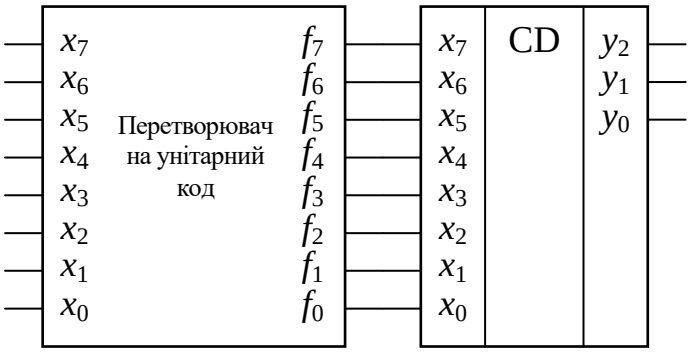


Рисунок 10.5 – Реалізація пріоритетного шифратора за допомогою перетворювача двійкового коду на унітарний код та звичайного непріоритетного шифратора

Отриманий складений пріоритетний шифратор матиме неоптимальну структуру, проте змінюючи порядок підключення виходів пристрою перетворення до входів звичайного шифратора можна легко змінювати пріоритетність.

Мікросхеми, що реалізують пріоритетні шифратори: 555ИВ1, МС14532В, 74148. Пріоритетні шифратори широко використовуються для створення контролери клавіатури, переривань, арбітри шин тощо. На основі пріоритетних шифраторів також можна створити АЦП (аналого-цифровий перетворювач) прямого перетворення (рис. 10.6).

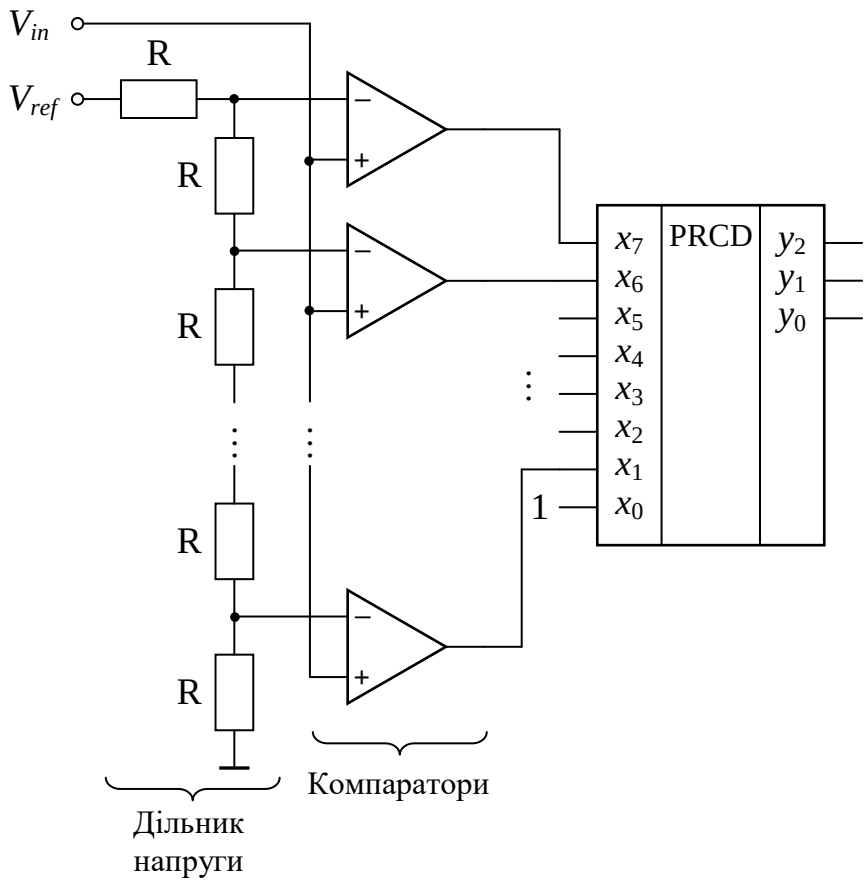


Рисунок 10.6 – АЦП прямого перетворення (Flash ADC) на основі пріоритетного шифратора

10.3. Схеми комбінаційного зсуву

Схеми зсуву використовуються для зміщення розрядів двійкового числа на задану кількість позицій у заданому напрямку – вліво чи вправо. Операції зсуву, як було визначено у розділі «Двійкова арифметика», еквівалентні множенню (зсув вліво) та діленню (зсув вправо) даного числа на степінь двійки (основи системи числення). Існує декілька видів зсуву, що використовуються у цифрових пристроях [3], зокрема:

- *Логічний зсув* – передбачає зсув числа вліво чи вправо так, що порожні розряди заповнюються нулями.
  Приклади: $11010 \underset{L}{\leftarrow} 2 = 01000$, $11010 \underset{L}{\rightarrow} 1 = 01101$.
- *Арифметичний зсув* – при зсуві вправо найбільш значущий біт заповнюється значенням знакового розряду, що необхідно для збереження знаку числа, яке зсувається; при зсуві вліво функціонують аналогічно логічному зсуву: найменш значущий біт заповнюється нулями.

Приклади: $11010 \underset{A}{\leftarrow} 1 = 10100$, $11010 \underset{A}{\rightarrow} 2 = 11110$, $01010 \underset{A}{\rightarrow} 2 = 00010$.

- *Циклічний зсув* – передбачає зсув числа по колу так, що пусті розряди заповнюються бітами, які висуваються з протилежної сторони.

  Приклади: $11010 \underset{C}{\leftarrow} 1 = 10101$, $11010 \underset{C}{\rightarrow} 2 = 10110$.

Схема комбінаційного зсуву (Barrel Shifter) можуть бути побудовані на основі мультиплексорів. Приклади схем логічного зсуву вліво та арифметичного зсуву вправо наведено на рис. 10.7.

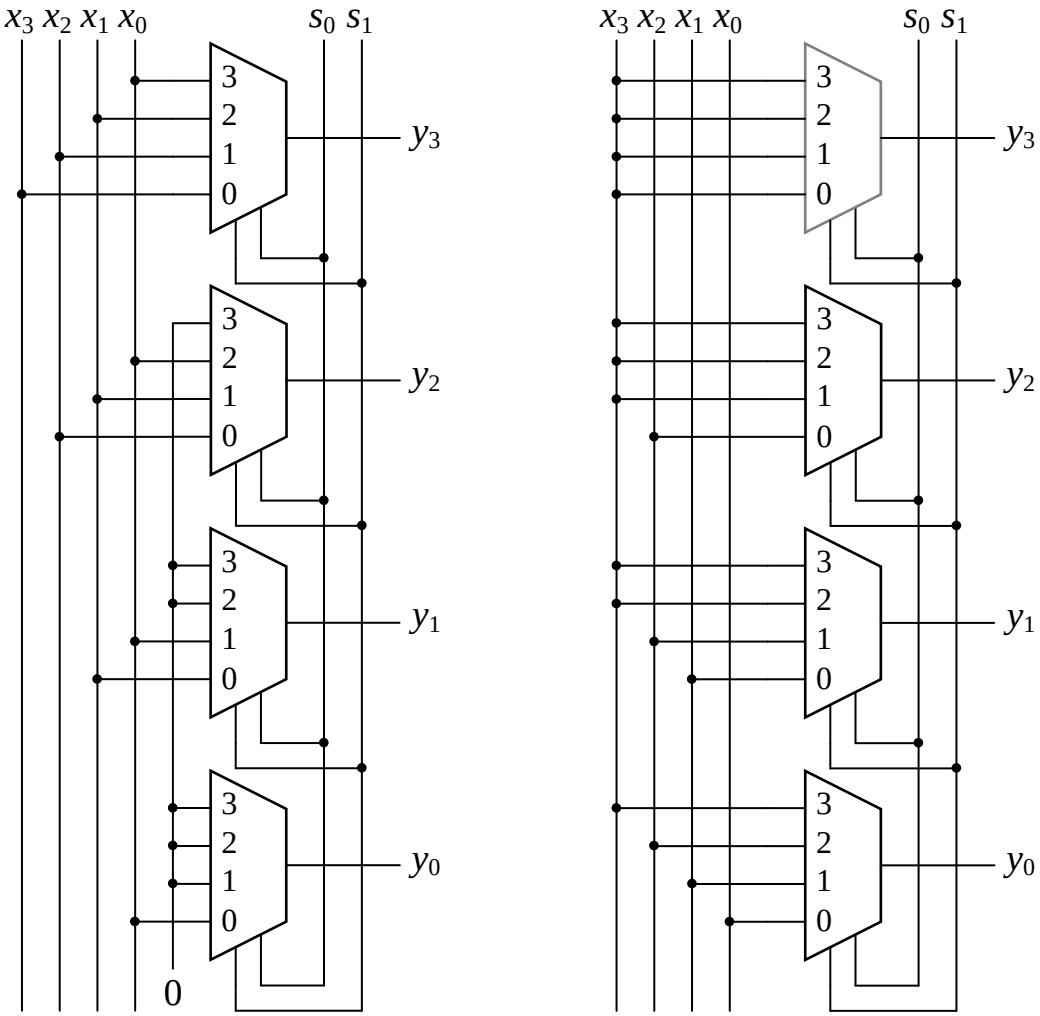


Схема логічного зсуву вліво

Схема арифметичного зсуву вправо

Рисунок 10.7 – Приклади комбінаційних 4-розрядних схем логічного та арифметичного зсуву

Схеми зсуву також можна будувати виключно на двохвходових мультиплексорах. Приклад універсальної схеми зсуву наведено на рис. 10.8 [5].

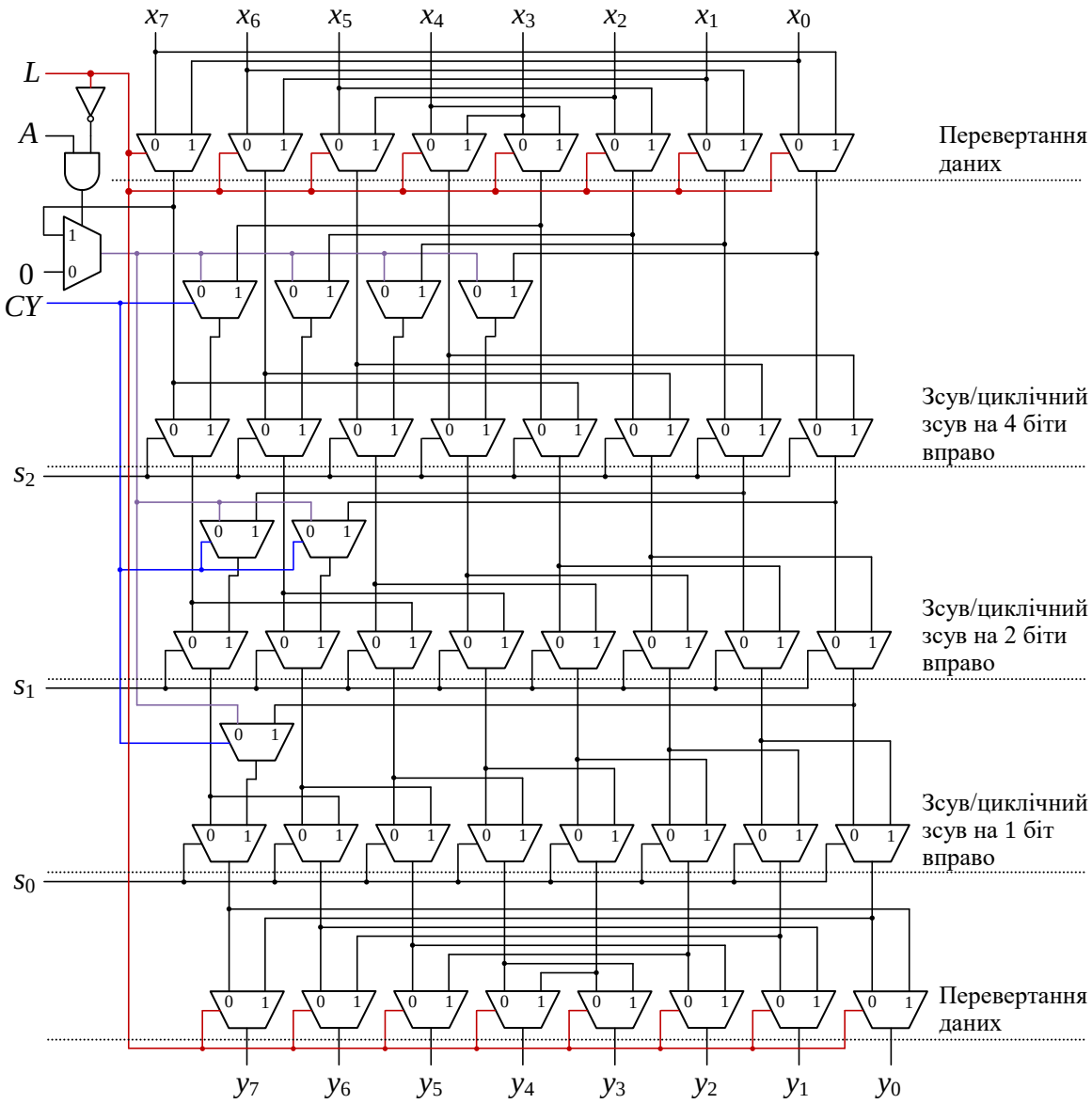


Рисунок 10.8 – Універсальна схема комбінаційного зсуву на двохвходових мультиплексорах

Схема з рисунка 10.8 може виконувати всі види операцій зсуву 8-бітних чисел. У ній високі рівні сигналів означають:

- *A* – виконується арифметичний зсув (тільки вправо);
- *L* – зсув виконується вліво;
- *CY* – зсув виконується циклічно.

Сигнали $s_{2:0}$ визначають наскільки має бути зсунуте вхідне 8-бітне число $x_{7:0}$. Результат задається числом $y_{7:0}$.

10.4. Комбінаційні суматори

Паралельним двійковим комбінаційним суматором називається комбінаційна схема, яка виконує обчислення суми двійкових n-розрядних чисел при одночасній їх подачі [1, 3, 4, 6].

10.4.1. Однорозрядний суматор

Операція сумування двійкових чисел вже була розглянута раніше в підрозділі «Двійкова арифметика». Вона, зокрема, передбачає складання відповідних цифр $a_i + b_i$ доданків, причому якщо значення сум виявлятиметься більшим, ніж допустиме значення для цифри $a_i + b_i > p - 1$ в даній системі числення (p – основа системи числення), то має відбуватись перенесення *одиниці* у наступний розряд. Оскільки, пошук суми виконується порозрядно, то для реалізації n-розрядного суматора потрібно спочатку реалізувати схему, що правильно виконує додавання двох довільних розрядів числа, а потім на її основі побудувати повний n-розрядний суматор. Схема такого однорозрядного суматора повинна містити *три* входи: a та b для розрядів чисел, що додаються та вхід c_i для врахування перенесення з попереднього розряду, а також *два* виходи: s для виведення суми і вихід c_o для перенесення у наступний розряд. Складемо таблицю істинності однорозрядного суматора. Вона матиме вигляд показаний нижче.

i	a	b	c_i	s	c_o
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

На основі отриманої таблиці істинності знайдемо ДДНФ функцій, що реалізують функції виходу s та c_o:

$$s = \bar{a}\bar{b}c_i \vee \bar{a}b\bar{c}_i \vee a\bar{b}\bar{c}_i \vee abc_i = (\bar{a}\bar{b} \vee ab)\cdot c_i \vee (\bar{a}b \vee a\bar{b})\cdot \bar{c}_i =$$

$$= \underbrace{(\bar{a}\bar{b} \vee ab)}_{\overline{\overline{\bar{a}\bar{b}} \vee ab} = \overline{\bar{a}\bar{b}}\cdot\overline{ab}} \cdot c_i \vee (\bar{a}b \vee a\bar{b})\cdot \bar{c}_i = \underbrace{\overline{(\bar{a} \vee \bar{b})\cdot(a \vee b)}}_{\overline{\bar{a}a \vee \bar{a}b \vee a\bar{b} \vee \bar{b}b}} \cdot c_i \vee (\bar{a}b \vee a\bar{b})\cdot \bar{c}_i$$

$$= \overline{(\bar{a}b \vee a\bar{b})}\cdot c_i \vee (\bar{a}b \vee a\bar{b})\cdot \bar{c}_i = \overline{(a \oplus b)}\cdot c_i \vee (a \oplus b)\cdot \bar{c}_i = a \oplus b \oplus c_i;$$

$$c_o = \bar{a}bc_i \vee a\bar{b}c_i \vee abc_i = (\bar{a}b \vee a\bar{b}) \cdot c_i \vee (\bar{c}_i \vee c_i) \cdot ab =$$
$$= (a \oplus b)c_i \vee ab.$$

Схема, що реалізує наведені вирази називається повним суматором та має вигляд, показаний нижче на рис. 10.9.

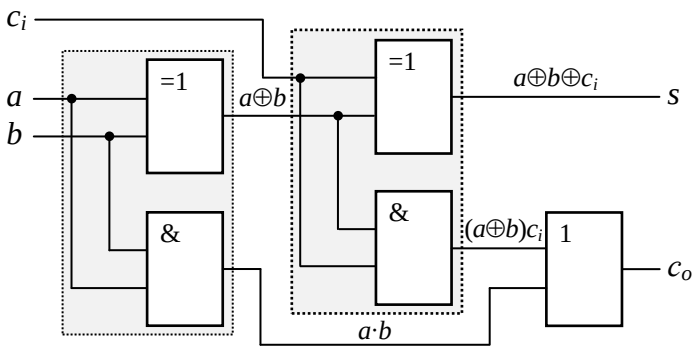


Рисунок 10.9 – Схема повного однобітного суматора

Вузли, що обведені в даній схемі пунктирною лінією називаються *напівсуматорами*. Напівсуматор виконує додавання бітів $a \oplus b$ без урахування перенесення з попереднього розряду, але сам цей сигнал ($a \cdot b$) генерує. Повний суматор на схемах позначається як:

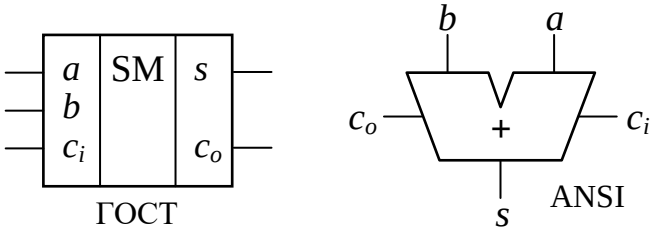
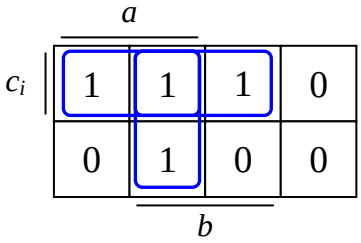


Рисунок 10.10 – Умовне позначення однорозрядного суматрра

Варто зауважити, що існує інша схема однорозрядного суматора, в якій перенесення формується більш швидко, проте вона матиме більшу кількість логічних елементів. Дану схему можна отримати, якщо оптимізувати функцію формування перенесення c_o. Для цього складемо карту Карно.



Тоді МДНФ для сигналу c_o матиме вигляд:

$$c_o = ab \vee ac_i \vee bc_i.$$

В базисі І-НІ:

$$c_o = \overline{\overline{ab} \cdot \overline{ac_i} \cdot \overline{bc_i}}.$$

Тоді, оптимізована по швидкодії схема однорозрядного суматора матиме вигляд, наведений на рис. 10.11.

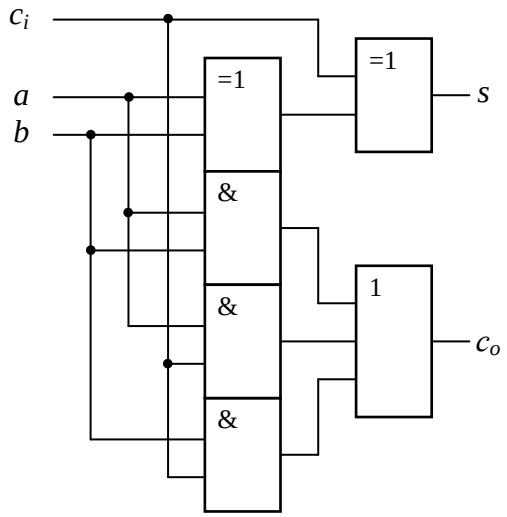


Рисунок 10.11 – Схема оптимізованого за швидкодією однобітного повного суматора

10.4.2. Суматор з послідовним перенесенням

На основі схеми повного однобітового суматора з урахуванням перенесень можна будувати схеми багаторозрядних суматорів. Приклад 4-розрядного суматора отриманого з чотирьох окремих однорозрядних суматорів наведена нижче (рис. 10.12).

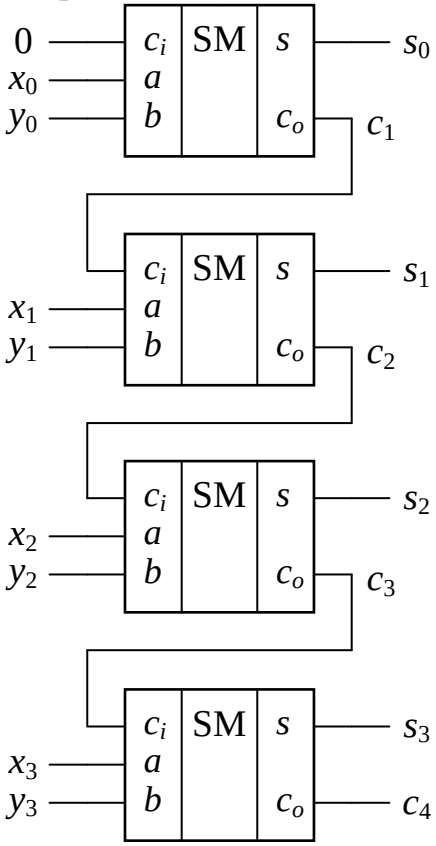


Рисунок 10.12 – Схема 4-х бітного суматора з послідовним перенесенням

Швидкодія наведеного вище суматора визначається затримкою розповсюдження сигналу перенесення: як видно зі схеми, для отримання кожної наступної суми розрядів s_i необхідно дочекатися сигналу перенесення з попереднього розряду c_{i-1}. Таким чином, коректна сума в останньому n-му розряді буде отримана тільки після того, як сформуються всі $n-1$ перенесень. При цьому сигнали перенесень розповсюджуються послідовно від розряду до розряду, саме тому розглянутий вище суматор називається *суматором з послідовним перенесенням*.

10.4.3. Суматор з паралельним перенесенням

Зі сказаного вище слідує, що для підвищення швидкодії багаторозрядних суматорів необхідно зменшити час розповсюдження сигналу перенесення від c_0 до c_n. Розглянемо як це можна зробити. Якщо проаналізувати схему формування перенесення повного суматора $c_o = (a \oplus b)c_i \vee ab$, то можна зробити наступні висновки: сигнал перенесення поточного розряду формується функцією $g = ab$, тому його також називають *функцією генерування перенесення* (Carry Generation), а сигнал $p = a \oplus b$ виконує функцію дозволу проходження перенесення попереднього розряду c_i, у зв'язку з чим його називають *функцією розповсюдження перенесення* (Carry Propagation). Важливою характеристикою цих двох функцій є те, що вони формуються виключно вхідними сигналами a та b бітів, які необхідно додати у поточному розряді та не залежать від будь-яких сигналів попередніх розрядів. Ввівши зазначені позначення сигнал перенесення тепер можна переписати як:

$$c_{i+1} = p_i c_i \vee g_i,$$

де c_{i+1} – перенесення у наступний розряд, $p_i = a \oplus b$ та $g_i = ab$ – функції розповсюдження та формування перенесення поточного розряду відповідно.

Користуючись знайденим виразом та рекурсивно підставляючи значення перенесень молодших розрядів у перенесення старших розрядів, отримаємо наступні вирази:

$$\begin{aligned}
c_1 &= p_0 c_0 \vee g_0,\\
c_2 &= p_1(p_0 c_0 \vee g_0) \vee g_1 =\\
&= p_1 p_0 c_0 \vee p_1 g_0 \vee g_1,\\
c_3 &= p_2(p_1(p_0 c_0 \vee g_0) \vee g_1) \vee g_2 =\\
&= p_2 p_1 p_0 c_0 \vee p_2 p_1 g_0 \vee p_2 g_1 \vee g_2\\
&\dots\\
c_n &= p_{n-1}(p_{n-2}\dots(p_1(p_0 c_0 \vee g_0) \vee g_1) \vee \dots \vee g_{n-2}) \vee g_{n-1} =\\
&= p_{n-1}p_{n-2}\dots p_1 p_0 c_0 \vee p_{n-1}p_{n-2}\dots p_1 g_0 \vee \dots \vee p_{n-1}g_{n-2} \vee g_{n-1}.
\end{aligned}$$

Як вже було встановлено, значення функцій p_i та g_i формуються незалежно для кожного розряду, тому перенесення можуть бути оцінені для всіх розрядів одночасно. Оскільки, формування перенесень, а отже і сум у розрядах відбувається одночасно (або інакше кажучи паралельно), суматор побудований за таким принципом називається *суматором з паралельним перенесенням*. Таким чином, суматори з паралельним перенесенням мають вищу швидкодію, проте є більш складними у реалізації. Схема суматора з паралельним перенесенням наведена нижче на рисунку 10.13.

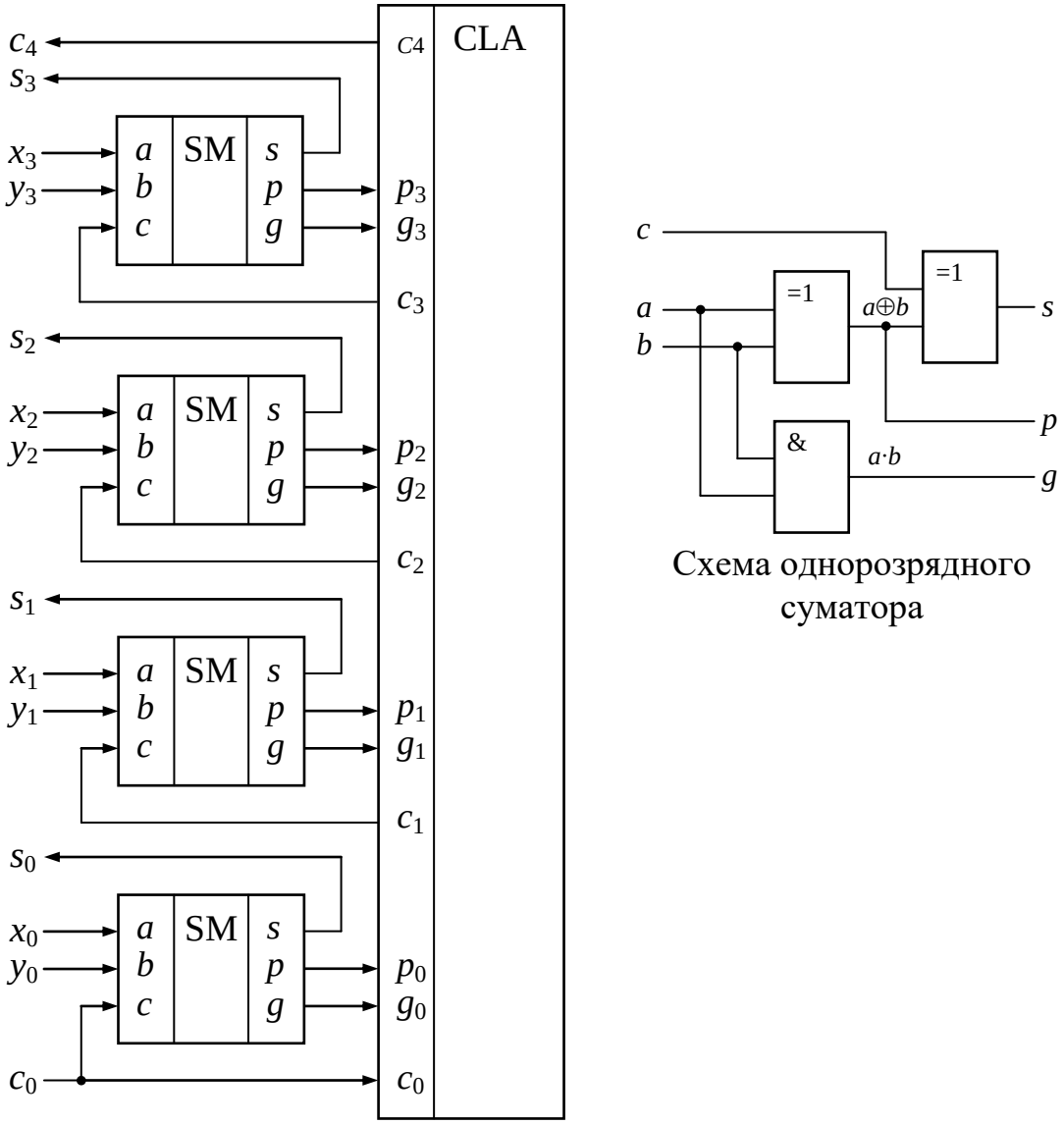


Схема однорозрядного суматора

Рисунок 10.13 – Схема суматора з паралельним перенесенням

Блок формування перенесення реалізує наведені вище формули та позначається у схемі як CLA (Carry-LookAhead).

Суматори з паралельним перенесенням також часто називають *суматорами з прискореним перенесенням*.

10.5. Схема віднімання

Як вже було встановлено раніше у підрозділі «Двійкова арифметика», операція віднімання може бути реалізована за допомогою переведення від'ємника у доповнювальний код та додавання його до зменшуваного. Нагадаємо, що перехід у доповнювальний код передбачає інвертування двійкового числа та додання до нього одиниці. З огляду на це, операцію віднімання можна представити наступним чином:

$$A - B = A + [B]_{Д} = A + (\overline{B} + 1),$$

де $[B]_{Д}$ – число B у доповнювальному коді; $\overline{B}$ – число B, кожен біт якого інвертовано:

$$\overline{B} = \{\overline{b}_{n-1}\overline{b}_{n-2}\ldots\overline{b}_1\overline{b}_0\}.$$

Як видно із отриманого виразу, для віднімання числа B від числа A, потрібно до A додати, інверсний код числа $\overline{B}$ та одиницю. Додавання одиниці при цьому можна здійснити, подавши її на вхід перенесення нульового розряду. Це дозволяє синтезувати наступну схему пристрою віднімання (рис. 10.14).

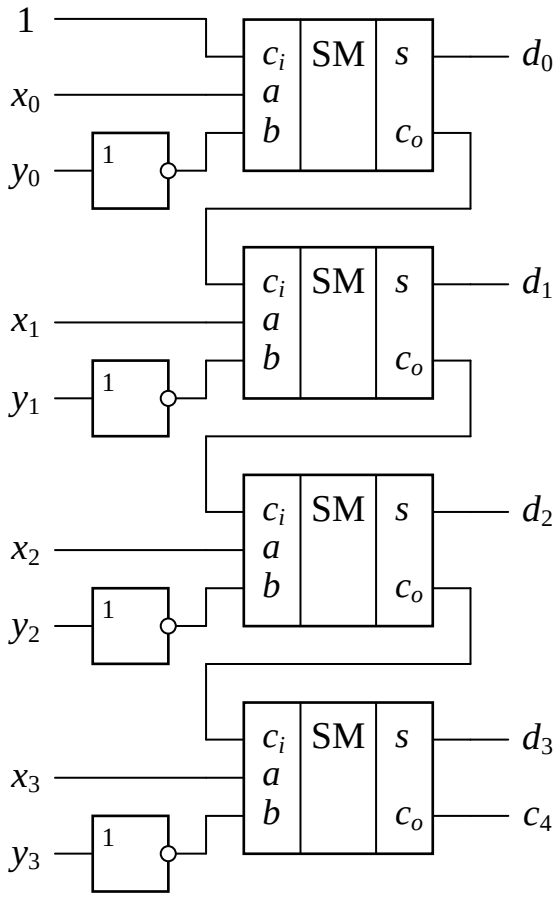


Рисунок 10.14 – Схема пристрою двійкового віднімання, побудована на основі 4-х розрядного суматора з послідовним перенесенням

Відзначимо, що якщо необхідно виконувати віднімання виключно додатних чисел, то наведену схему можна ще дещо спростити. Ідея тут полягає у тому, що у коректному від'ємному числі (числі у доповнювальному коді) обов'язково найстарший розряд дорівнює «1», тому на вхід *b* старшого однорозрядного суматора можна одразу завести «1», економлячи на одному інверторі.

10.6. Універсальна схема додавання та віднімання

Щоб отримати універсальну схему додавання та віднімання, в якій необхідна операція може обиратись, доцільно скористатись властивістю двомісної операції *сума по модулю* 2. Зокрема, дана операція дозволяє інвертувати сигнал, що поданий на один її вхід при подачі на другий вхід *одиниці*, а також пропускати сигнал без змін, при подачі на другий вхід *нуля*. Таким чином, універсальна схема додавання та віднімання матиме наступний вигляд, наведений на рис. 10.15.

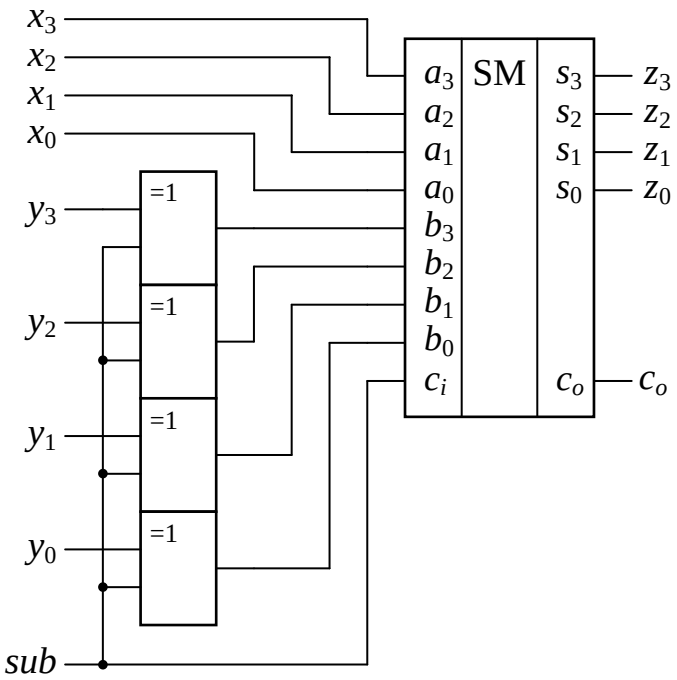


Рисунок 10.15 – Універсальна схема додавання та віднімання

10.7. Схема логічного множення

Процедура множення розглядалася в розділі «Двійкова арифметика». Вона виконується аналогічно до звичайного десяткового множення, з тою особливістю, що в якості базової порозрядної операції множення тут виступає *кон'юнкція*. При цьому кожен біт множника перемножується з усім множеним (виконується побітова кон'юнкція), в результаті чого

утворюються частинні добутки. Отримані частинні добутки зсуваються та додаються, даючи остаточний результат. При множенні двох N-розрядних чисел формується $2N$-розрядний результат [3].

Схематично процедуру множення можна узагальнити наступним чином (для спрощення наведено чотирьох розрядні числа):

				a_3	a_2	a_1	a_0	множене
			×	b_3	b_2	b_1	b_0	множник
				a_3b_0	a_2b_0	a_1b_0	a_0b_0	
+			a_3b_1	a_2b_1	a_1b_1	a_0b_1		частинні добутки
		a_3b_2	a_2b_2	a_1b_2	a_0b_2			
	a_3b_3	a_2b_3	a_1b_3	a_0b_3				
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	результат

Користуючись вже розробленим повним однобітним суматором, наведений вище вираз можна представити комбінаційною схемою, показаною на рис. 10.16.

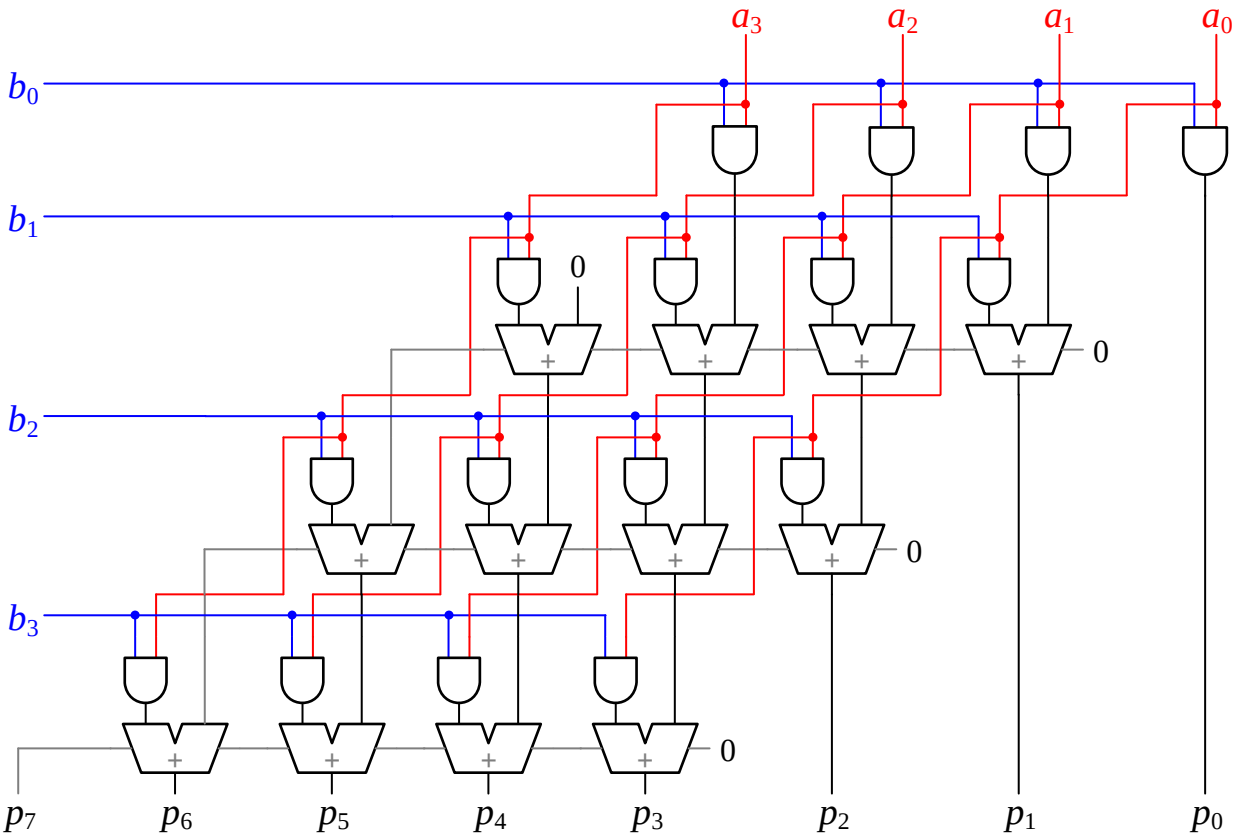


Рисунок 10.16 – Схема множення 4-х розрядних двійкових чисел

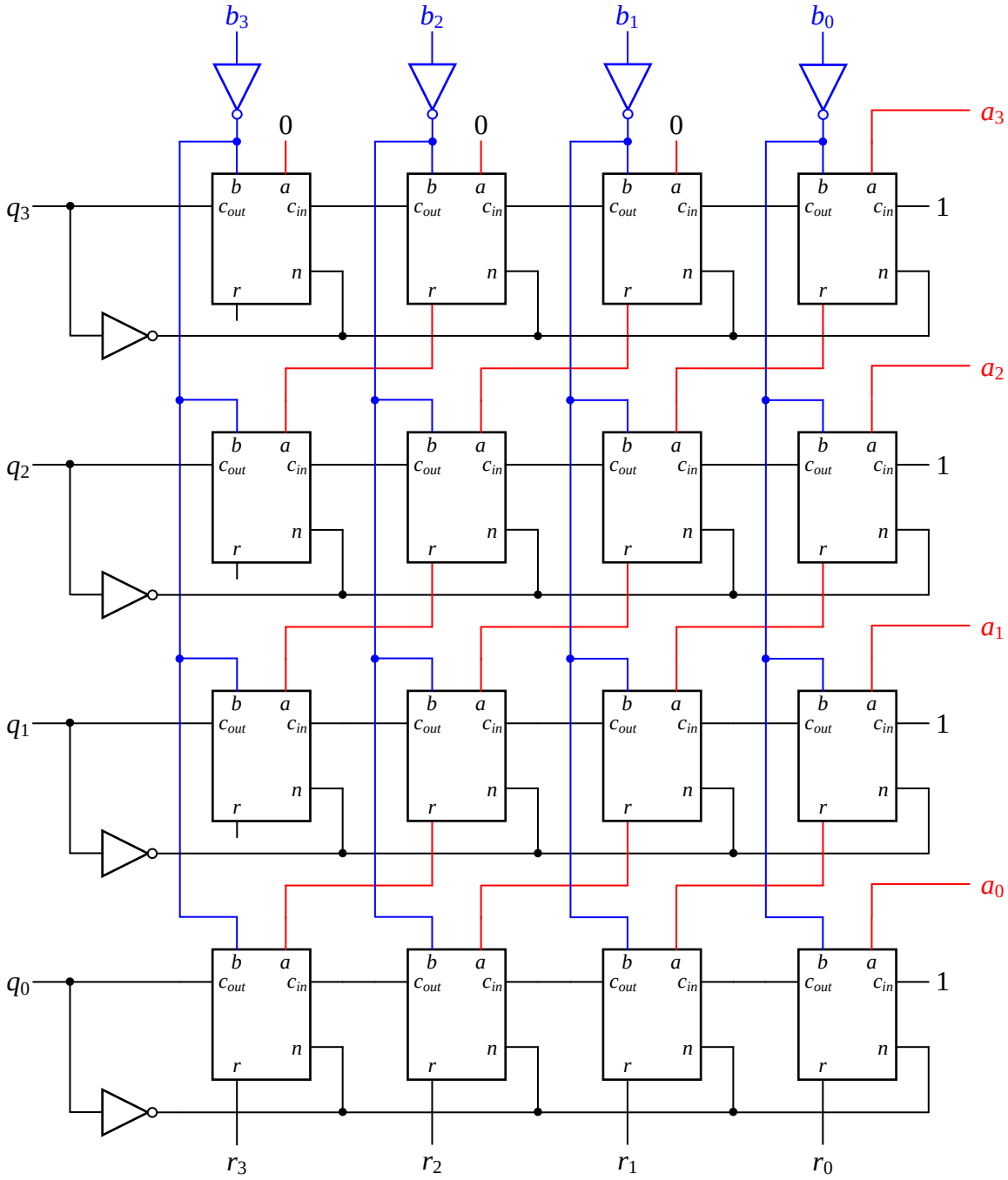
10.8. Схема логічного ділення

Альтернативно до розглянутої в розділі «Двійкова арифметика» процедури ділення двійкових додатних M-розрядних чисел може використовуватись також і наступний алгоритм [3]:

```
r′ = 0
i = M
поки  i ≥ 0
      r = r′ · 2 + a_i                 // r = { r′ << 1, a_i }
      d = r – b                        // d = r + [b]_д, [b]_д – доповнювальний код b
      якщо  n = (d < 0),  то
            q_i = 0,   r′ = r          // r < b
      інакше
            q_i = 1,   r′ = d          // r ≥ b
      кінець якщо
кінець поки
r = r′
```

Наведений алгоритм виконує ділення числа a на число b. В алгоритмі змінні r позначають частинні остачі (reminder), змінна d – різницю, n – знак різниці, i – номер поточного біта, a_i означає i-ий біт числа a. Змінна q позначає частку (quotient). Результат виконання алгоритму задовольняє умові: $a / b = q + r / b$, тобто результатом є як ціла частина частки q, так і остача від ділення r.

На основі наведеного алгоритму можна побудувати схему комбінаційного ділення (матрицю ділення). Для спрощення матриця ділення представлена для 4-розрядних чисел. Дана схема показана нижче на рисунку 10.17.



Структура базового блоку

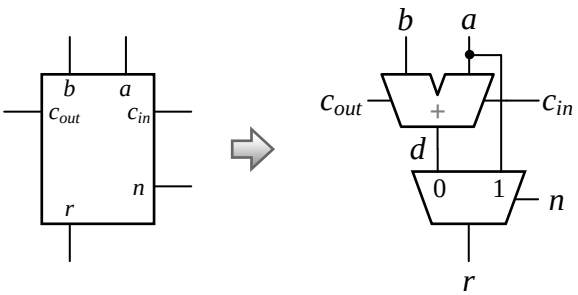


Рисунок 10.17 – Схема матриці ділення 4-х розрядних двійкових чисел

Питання та завдання для самоперевірки і контролю засвоєння знань

1. Поясніть на чому оснований принцип синтезу демультиплексорів? Чи може демультиплексор бути одночасно мультиплексором, якщо так, то в якому випадку?
2. В чому полягають особливості синтезу звичайних шифраторів?
3. Що таке пріоритетний шифратор? Наведіть приклади використання даного пристрою.
4. Яким чином можна отримати пріоритетний шифратора зі звичайного непріоритетного?
5. На рисунку 10.5 наведено приклад АЦП прямого перетворення, що побудований на основі пріоритетного шифратора. Який пристрій серед вже розглянутих можна використати, щоб за аналогією побудувати ЦАП (цифрово-аналоговий перетворювач)?
6. Які схеми комбінаційного логічного зсуву існують? Поясніть в чому їх відмінності.
7. На рисунку 10.6 показано комбінаційні схеми звичайного логічного зсуву вліво та арифметичного зсуву вправо. За аналогією запропонуйте свої схеми циклічного зсуву.
8. В чому полягають відмінності між повним суматором та напівсуматором?
9. Які типи багаторозрядних суматорів існують? В полягає основна їх відмінність?
10. Як із схеми суматора отримати схему віднімання? Яке представлення від’ємника при цьому використовується?
11. Поясніть яким чином користуючись суматором можна отримати схеми множення та ділення?
12. Як можна реалізувати схеми порівняння двійкових чисел?

Перелік скорочень

АЦП	Аналого-цифровий перетворювач
ДДНФ	Досконала диз’юнктивна нормальна форма
ДКНФ	Досконала кон’юнктивна нормальна форма
ДНФ	Диз’юнктивна нормальна форма
ЕОМ	Електронно-обчислювальна машина (комп’ютер)
КНФ	Кон’юнктивна нормальна форма
МДНФ	Мінімальна диз’юнктивна нормальна форма
МКНФ	Мінімальна кон’юнктивна нормальна форма
ПЛІС	Програмована логічна інтегральна схема
СкДНФ	Скорочена диз’юнктивна нормальна форма
ТДНФ	Тупикова диз’юнктивна нормальна форма
ЦАП	Цифрово-аналоговий перетворювач
CLA	Carry-lookahead – паралельне перенесення при додаванні
HDL	Hardware description language
Inf	Infinity («нескінченність»)
NaN	Not a number («не число»)

Предметний покажчик

А

Алгебра логіки....45
  аксіоми....56
  теореми....57

Б

Булева алгебра....45

Д

Двійкова арифметика
  віднімання....24, 126
  ділення....32, 129
  додавання....23, 121
  множення....29, 128
Демультиплексор....112
Дешифратор....70
Диз'юнкція....46
Дизтерм....92
Діаграма Вейча....94
Досконала нормальна форма (ДНФ)...72
  в базисі АБО-НІ....74
  в базисі І-НІ....74
  диз'юнктивна....72
  кон'юнктивна....73

З

Закон
  асоціативності....57, 59
  двоїстості де Моргана....58
  дистрибутивності....57, 59
  заперечення....58
  комутативності....57, 59
  поглинання....58, 88
  подвійного заперечення....58
  склеювання....58, 88
Заперечення....46

І

Імпліканта....88
Імплікація....46

К

Карта Карно....94
Код
  Грея....12
  доповнювальний....24, 126
  прямий....28
  унітарний....10
Кон'юнкція....46
Контерм....89

Л

Логічна функція....45
  вироджена....46
  інверсна....47
  неповністю визначена....47
  повністю невизначена....47
Логічний елемент....48
  АБО....49
  АБО-НІ....54
  Виключне-АБО....52
  І....48
  І-НІ....53
  НІ....51
Логічні рівняння....78
  з більш ніж одним невідомим....84
  з одним невідомим....78

М

Мажоритарні елементи....106
Макстерм....69
Мінімізація логічних функцій....88
  картами Карно-Вейча....94
  методом Квайна....88
  методом Квайна-Мак-Класкі....92
  неповністю визначених....101

спільна декількох функцій 103
Мінтерм.....68
Мультиплексор.....63

П

Первинний терм 67
Перемикальна функція 45
Перехід від однієї системи числення до іншої
дробові числа 16, 19
цілі числа.....13
Принцип двоїстості.....57
Принцип суперпозиції (композиції)....47
Пріоритетність операцій 60

Р

Розклад
Рида.....65
Рида-Маллера 76
Шеннона.....62

С

Система числення 7
двійкова 21
економічність.....20
непозиційна.....8
оптимальна.....20
позиційна.....10
неоднорідна 10
однорідна.....11
унарна.....8
Суматор.....121
багаторозрядний.....123
з паралельним перенесенням 125
з послідовним перенесенням 124
однорозрядний.....122
повний 122
Схема
віднімання.....126
ділення.....129
додавання-віднімання.....127
множення 127
Схема зсуву 118
арифметичного 118
логічного 118
на двохвходових мультплексорах 119
циклічного 119

Ф

Факторизована форма функції 106
Функціонально повний базис 47

Ч

Числа
з плаваючою точкою.....38
з фіксованою точкою 34

Ш

Шифратор пріоритетний.....115
Шифратор простий 113

ЛІТЕРАТУРА

Основна література

1. Пухальский Г.И. Цифровые устройства: Учебное пособие для втузов / Пухальский Г.И., Новосельцева Т.Я. – СПб.: Политехника, 1996. – 885 с.
2. Прикладная теория цифровых автоматов / К.Г. Самофалов и др. – К.: Вища школа, 1987. – 375 с.
3. Харрис Д.М. Цифровая схемотехника и архитектура компьютера / Д.М. Харрис, С.Л. Харрис. – Нью-Йорк: Elsevier, 2013. – 1621 с.
4. Угрюмов Е.П. Цифровая схемотехника: Учеб. пособие для вузов. – 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2004.
5. Pillmeier M.R. Design alternatives for barrel shifters / M.R. Pillmeier, M.J. Schulte and E.G. Walters III // Proceedings of the SPIE. – Vol. 4791, 2002. – P.436– 447.

Додаткова література

6. Потемкин И.С. Функциональные узлы цыфровой автоматики / И.С. Потемкин. – М. Энергоатомиздат, 1988. – 320 с.
7. Глушков В.М. Синтез цифровых автоматов / В.М. Глушков. – М. Государственное издательство физико-математической литературы, 1962. – 476 с.

ANSI/IEEE-754-2008 Standard for Binary Floating-Point Arithmetic
ISO/IEC/IEC 60559::2011 – Information technology – Microprocessor Systems – Floating-Point Arithmetic