МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАСИЛЯ СТУСА

О. О. Баркалов, Л. О. Титаренко, А. В. Баєв, Т. В. Нескородєва

ДИСКРЕТНИЙ АНАЛІЗ ПРИ ПРОЄКТУВАННІ І РЕАЛІЗАЦІЇ ЦИФРОВИХ АВТОМАТІВ

Навчальний посібник

Вінниця
2022

УДК 519.1:519.713-028(075.8)
Д 48

Рекомендовано до друку
Вченою радою ДонНУ імені Василя Стуса
(протокол № 16 від 31 травня 2021 р.)

Автори: *О. О. Баркалов,* д-р техн. наук, професор кафедри інформаційних технологій;
А. В. Баєв, канд. фіз.-мат. наук, доцент кафедри прикладної математики, декан факультету інформаційних і прикладних технологій;
Л. О. Титаренко, д-р техн. наук, професор Харківського національного університету радіоелектроніки;
Т. В. Нескородєва, д-р техн. наук, доцент, завідувач кафедри інформаційних технологій.

Рецензенти: *Є. Є. Федоров,* д-р техн. наук, доцент, професор кафедри робототехніки та спеціалізованих комп'ютерних систем Черкаського державного технологічного університету;
О. В. Багацький, канд. техн. наук, старший науковий співробітник відділу 200, ІК НАНУ.

О. О. Баркалов, Л. О. Титаренко, А. В. Баєв, Т. В. Нескородєва

Д 48 Дискретний аналіз при проєктуванні і реалізації цифрових автоматів: навч. посіб. для здобувачів спеціальностей 122 «Комп'ютерні науки», 125 «Кібербезпека», 113 «Прикладна математика» / О. О. Баркалов, Л. О. Титаренко, А. В. Баєв, Т. В. Нескородєва. Вінниця: ДонНУ імені Василя Стуса, 2022. 320 с.

Навчальний посібник призначений для здобувачів вищої освіти у галузі інформаційних технологій. Він є базовим і формує у студентів основи знань, необхідних для розуміння принципів організації апаратури (hardware) сучасної обчислювальної техніки. Таке розуміння необхідне як для синтезу та аналізу цифрових систем, так і для написання ефективних програм (software), що дають змогу враховувати всі особливості апаратури, яка виконує ці програми. В посібнику розглядаються такі основні питання: системи числення; основи булевої алгебри; мінімізація булевих функцій; синтез комбінаційних схем; синтез комбінаційних схем на базі програмованих логічних пристроїв; синтез абстрактних автоматів; синтез структурних автоматів; синтез операційних автоматів.

УДК 519.1:519.713-028(075.8)

ЗМІСТ

ВСТУП

До кінця XX ст. світ був свідком трьох інформаційних революцій: винахід писемності (Месопотамія, 5–6 тисяч років тому), винахід рукописної книги (Китай, близько 1300 р. до нашої ери) і винахід в Європі друкарського верстата (Йоганн Гутенберг, між 1450 і 1455 рр.). Під інформаційною революцією розуміється сукупність якісних змін у всіх сферах життєдіяльності суспільства, вироблених в результаті впровадження нових засобів передачі інформації. Сьогодні світ переживає четверту інформаційну революцію, пов'язану з широким розповсюдженням ЕОМ і мережі Інтернет. Однією з теоретичних основ цієї революції стала інформатика (Computer Science), під якою розуміється наука про способи отримання, накопичення, зберігання, перетворення, передачі і використання інформації. Інформатика охоплює дисципліни, пов'язані з обробкою інформації в ЕОМ і обчислювальних мережах. Одним з основних понять інформатики є цифровий автомат, під яким розуміється перетворювач дискретної інформації. Цифрові автомати прийнято розділяти на автомати без пам'яті (комбінаційні схеми) і автомати з пам'яттю (послідовні схеми). Найпростішим цифровим автоматом без пам'яті є вентиль, який реалізує одну з функцій булевої алгебри. З'єднання вентилів у вигляді мережі без зворотних зв'язків породжує складні комбінаційні схеми (суматор, дешифратор тощо). Найпростішим автоматом з пам'яттю є тригер, який має два стійкі стани. Спеціаліст в галузі інформатики повинен мати фундаментальні знання, пов'язані з прикладними аспектами теорії цифрових автоматів.

Як відомо, ЕОМ обробляє двійкову інформацію, тобто деякі сукупності нулів і одиниць. Обробка інформації в ЕОМ проводиться на основі принципу мікропрограмного управління, сформульованого в 1951 р. професором Кембриджа М. Уілксом. Відповідно до цього принципу будь-

яка операція представляється у вигляді послідовності мікрокоманд, які складаються з мікрооперацій, що виконують елементарну обробку інформації. Цей принцип був розвинений академіком В. М. Глушковим, що сформулював концепцію операційного пристрою, який є композицією операційного (ОА) і керуючого (КА) автоматів.

Операційний автомат приймає і зберігає виконавчі дані, обробляє їх і формує проміжні та кінцеві результати операцій. Керуючий автомат аналізує код операції, що виконується, і формує розподілену в часі послідовність керуючих сигналів (мікрооперацій). Для визначення набору мікрооперацій (мікрокоманд), який формується в кожен момент часу, КА аналізує логічні умови, які несуть інформацію про перебіг операції, що виконується. Отже, логічні умови є зворотним зв'язком від керованого об'єкта до керуючого об'єкта. Така взаємодія ґрунтується на загальних законах кібернетики – науки про управління. Термін «кібернетика» використовувався ще Платоном на позначення управління в людському суспільстві. У теорію інформації цей термін був введений Н. Вінером в його книзі «Кібернетика».

Методи синтезу операційних і керуючих автоматів відрізняються значною мірою. Відмінність методів синтезу породжується тим, що число станів ОА і КА значно різниться. Число станів КА зазвичай не перевищує кілька тисяч, що дає змогу використовувати для їхнього синтезу підходи, засновані на канонічному методі структурного синтезу. Операційні автомати можуть мати практично нескінченне число станів. Якщо пам'ять ОА складається з восьми регістрів по 32 розрядам, то загальне число станів ОА $M_{OA} = 2^{8*32} = 2^{256} \approx 10^{85}$.

Ця книга написана на основі лекцій з курсу «Прикладна теорія цифрових автоматів» (ПТЦА) і орієнтована, насамперед, на студентів інформаційних спеціальностей. Курс «ПТЦА» є базовим і формує у студентів основи знань, необхідних для розуміння принципів організації апаратури (hardware) сучасної обчислювальної техніки. Таке розуміння

необхідне як для синтезу та аналізу цифрових систем, так і для написання ефективних програм (software), що дають змогу враховувати всі особливості апаратури, яка виконує ці програми.

У книзі розглядаються такі основні питання:

1. Системи числення. Розглянуто загальні принципи подання чисел. Особливий акцент зроблений на двійкову систему числення. Розглянуто методи переведення чисел з однієї системи числення в іншу.

2. Основи булевої алгебри. Булева алгебра лежить в основі методів синтезу та оптимізації комбінаційних схем. Оскільки наша книга орієнтована на студентів технічних спеціальностей, то акцент робиться на прикладних аспектах булевої алгебри. Розглянуто аксіоми і закони булевої алгебри, а також методи представлення булевих функцій.

3. Мінімізація булевих функцій. Для зменшення витрат апаратури в комбінаційних схемах необхідно спрощувати їхнє подання. З цією метою використовуються методи мінімізації булевих функцій. Розглянуто як методи, що дають змогу мінімізувати функції вручну (карти Карно, метод Квайна–Мак-Класскі), так і евристичні методи, що використовуються на практиці в системах автоматизованого проєктування.

4. Синтез комбінаційних схем. Розглядаються методи синтезу одновихідних і багатовихідних комбінаційних схем. Методи синтезу орієнтовані на базис елементів І-НЕ і АБО-НЕ з урахуванням обмежень на параметри вентилів (число входів і здатність навантаження). Розглянуто методи синтезу стандартних комбінаційних блоків ЕОМ (дешифратори і мультиплексори), а також методи синтезу комбінаційних схем на їхній основі.

5. Синтез комбінаційних схем на базі програмованих логічних пристроїв. Наразі прогрес в області напівпровідникової технології перетворив мікроелектроніку в наноелектроніку. Це зумовило появу широкого класу програмованих логічних пристроїв (ПЛУ). Методи синтезу схем на базі ПЛУ значно відрізняються від методів синтезу схем

на вентилях. У книзі розглянуто еволюцію елементного базису ЕОМ і методи синтезу схем з використанням основних представників класу ПЛУ. Підрозділ 5.4 написаний спільно з доцентом кафедри комп'ютерної інженерії ДонНТУ А. А. Красичковим.

6. Синтез абстрактних автоматів. Абстрактний автомат є найбільш загальною моделлю послідовних схем, що задає її поведінку. У роботі розглядаються методи завдання автоматів Мілі і Мура, а також методи мінімізації числа станів. Крім того, розглянуто множину практичних послідовних схем, що задаються у вигляді абстрактних автоматів: від генератора одиничних імпульсів до схеми регулювання пішохідним переходом.

7. Синтез структурних автоматів. Структурний автомат є послідовною схемою, що синтезується за описом, заданим абстрактним автоматом. У книзі розглядаються різні тригери і методи синтезу структурних автоматів Мілі і Мура за графами відповідних абстрактних автоматів. Методи синтезу керуючих автоматів багато в чому подібні до методів синтезу структурних автоматів. З огляду на це методи синтезу керуючих автоматів в нашій книзі не розглядаються.

8. Синтез операційних автоматів. Розглядається базис для синтезу схем операційних автоматів – шини, регістри і комбінаційні схеми. Вводиться поняття типових мікрооперацій і функціональних мікропрограм. Розглядаються формальні методи синтезу операційних автоматів з різними структурними схемами за функціональними мікропрограмами.

Питання прикладної теорії цифрових автоматів надзвичайно важливі для підготовки фахівців в області розробки апаратного і програмного забезпечення засобів обчислювальної техніки. Водночас останні підручники в цій області, доступні широкому колу читачів, вийшли понад двадцять років тому. Така невідповідність між важливістю

проблеми і відсутністю доступної літератури і стала спонукальним стимулом для написання цієї книги.

Автори висловлюють подяку аспірантці Університету Зеленогурського (Польща) Е. Хебді та аспіранту ДонНТУ К. Єфименку за допомогу в підготовці цієї книги до видання.

Книга передусім орієнтована на студентів, чим і пояснюється манера викладу матеріалу, де математична строгість принесена в жертву наочності. Книга не претендує на оригінальність матеріалу, який є добре відомим. Основна мета книги – сприяти поліпшенню якості підготовки фахівців у галузі інформатики.

1 СИСТЕМИ ЧИСЛЕННЯ

1.1 Загальна характеристика систем числення

Система числення – це сукупність прийомів і правил запису чисел цифровими знаками. Найбільш відома десяткова система, де для подання чисел використовують цифри від 0 до 9. Числа виникли в далекій давнині як засіб підрахунку предметів навколишнього світу. Існує безліч способів запису чисел, однак для застосування на практиці система числення має бути наділена такими властивостями:

– можливість подання будь-якого числа в заданому форматі;

– однозначність подання чисел;

– простота виконання операцій над числами.

Системи числення поділяються на непозиційні та позиційні, головна відмінність між якими полягає в способі визначення значення символу (цифри) в числі.

У непозиційних системах числення значення символу не залежить від його положення в числі. Нехай A_D – запис числа в системі числення D, $D_i \in D$ – символи системи, що утворюють базу $D = \{D_1, \ldots, D_I\}$. Тоді число A_D може бути представлене у вигляді

$$A_D = D_1 + D_2 + \ldots + D_K = \sum_{i=1}^{K} D_i . \qquad (1.1)$$

Як випливає з (1.1), для запису числа використані K символів $D_i \in D$. Очевидно, що в таких системах кількісний еквівалент будь-якої цифри постійний і залежить тільки від її накреслення (графічного образу).

Вираз (1.1) характерний для системи числення, що виникла в Давньому Єгипті в XX ст. до н. е. Кожен символ зображувався ієрогліфом (від грецького «священна різьба»). Відтак такі системи були названі ієрогліфічними. Найбільш відомою ієрогліфічною системою числення є

римська, в якій є знаки I, V, X, L, C, D, M. Вони позначають десяткові числа 1, 5, 10, 50, 100, 500 і 1000 відповідно.

У римській системі перші числа натурального ряду від 1 до 10 записуються у такий спосіб: I, II, III, IV, V, VI, VII, VIII, IX, X.

На відміну від (1.1), в римській системі використані як принцип складання, так і віднімання. Зокрема, символ I, поставлений ліворуч від старшого знаку (V, X), зменшує його значення на одиницю. Водночас, символ I, поставлений праворуч від старшого знаку (V, X), збільшує його значення на одиницю. Цей самий принцип діє для будь-якої пари знаків. Наприклад, число 1954 представляється MCMLIV, де CM = 900, а IV = 4.

Другим представником класу непозиційних систем є алфавітна система. Найбільш відома грецька система, що виникла близько 500 р. до н. е. в Мілеті. Тут для подання чисел використовувалися всі букви алфавіту і деякі старі, що вийшли з ужитку, букви.

Непозиційні системи мають низку недоліків:

– відсутність нуля;

– нескінченне число символів;

– виняткова складність арифметичних операцій.

У позиційних системах числення значення кожної цифри визначається як її зображенням, так і місцем (позицією) в числі. Алфавіт позиційних систем має обмежене (кінцеве) число символів, що представляють цифри. Найбільш відомою є десяткова система, що має алфавіт (базу) {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} і підставу 10. Наприклад, число 222 включає 3 однакові цифри, але їм відповідають різні значення. Першому символу відповідає число $200 = 2 \cdot 10^2$, наступному – число $20 = 2 \cdot 10^1$ і, нарешті, останньому – число $2 = 2 \cdot 10^0$. Цю залежність можна виразити формулою:

$$222_{10} = 2 \cdot 10^2 + 2 \cdot 10^1 + 2 \cdot 10^0. \qquad (1.2)$$

Індекс 10 в (1.2) означає, що число записане в десятковій системі числення.

У загальному випадку будь-яке число в позиційній системі записується у вигляді:

$$A = a_n a_{n-1} a_{n-2} \ldots a_1 a_0 . \quad (1.3)$$

При цьому окрема позиція в зображенні числа називається розрядом, а номер позиції – номером розряду. У загальному випадку число A_{10} можна представити у такий спосіб:

$$A_{10} = a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_1 \cdot 10^1 + a_0 \cdot 10^0 . \quad (1.4)$$

Цей поліном можна представити коротко у вигляді:

$$A_{10} = \sum_{i=0}^{n} a_i \cdot 10^i . \quad (1.5)$$

Існують системи числення, заснування яких відмінні від 10. Здебільшого люди використовували для рахунку пальці. З огляду на це історично існували п’ятіркова система (одна рука), десяткова (пальці двох рук), дванадцяткова (фаланги чотирьох пальців руки), двадцяткова (пальці рук і ніг). Першою відомою позиційною системою була 60-кова, що виникла в стародавньому Вавилоні близько 2500 р. до н. е. Десяткова система була створена в Індії близько VII ст., при цьому нуль з’явився тільки в IX ст. Індійські цифри були запозичені арабськими купцями, а від них цифри потрапили в Європу.

Нехай для подання будь-якого розряду числа використовуються цифри $a_i \in \{0, \ldots, p-1\}$, де p – основа системи числення. Тоді число, записане у вигляді (1.3), відповідає виразу:

$$A_p = a_n \cdot p^n + a_{n-1} \cdot p^{n-1} + \ldots + a_1 \cdot p^1 + a_0 \cdot p^0 = \sum_{i=0}^{n} a_i p^i . \quad (1.6)$$

Очевидно, що за $p = 10$ вирази (1.5) і (1.6) адекватні. Вираз (1.6) відповідає однорідній системі числення, в якій підстава однакова для всіх розрядів. У загальному випадку кожному розряду a_i може відповідати своя основа p_i. Це характерно для неоднорідних систем числення, де кількість допустимих символів може бути неоднаковою для різних розрядів.

Прикладом неоднорідної системи числення є система числення часу, для якої молодший розряд $p_0 = 1$ (секунди); другий (хвилини) $p_1 = 60$ секунд; третій (годин) $p_2 = 60$ хвилин; четвертий (доби) $p_3 = 24$ години; п'ятий (роки) $p_4 = 365$ діб. Наприклад, час в 3 роки, 15 діб, 13 годин, 39 хвилин, 28 секунд можна визначити у такий спосіб:

$$A = 3 \cdot 365 \cdot 24 \cdot 60 \cdot 60 + 15 \cdot 24 \cdot 60 \cdot 60 + 13 \cdot 60 \cdot 60 + 39 \cdot 60 + 28.$$

Далі будемо розглядати тільки однорідні позиційні системи числення і домовимося називати їх просто «системи».

При записі правильних дробів позиції нумеруються в такому порядку: 1, 2, ... При цьому ваги розрядів будуть негативними. Наприклад, дріб

$$A_p = a_{-1}a_{-2}\ldots a_{-m}$$

представляється у вигляді виразу

$$A_p = a_{-1}p^{-1} + a_{-2}p^{-2} + \ldots + a_{-m}p^{-m}. \qquad (1.7)$$

Змішані дроби також представляються у вигляді поліномів:

$$A_p = a_n p^n + \ldots + a_0 p^0 + a_{-1}p^{-1} + \ldots + a_{-m}p^{-m}, \qquad (1.8)$$

де p – основа системи числення. Вираз (1.8) відповідає лінійному запису числа A:

$$A_p = a_n a_{n-1}\ldots a_1 a_0 a_{-1}a_{-2}\ldots a_{-m},$$

і його значення визначається за формулою

$$A_p = \sum_{i=-m}^{n} a_i \cdot p^i. \qquad (1.9)$$

У табл. 1.1 наведені представлення перших 16 чисел в різних системах числення.

Вираз (1.9) можна використовувати для знаходження десяткових значень чисел, представлених в будь-якій системі числення. При цьому

ступені підстави визначаються за правилами десяткової системи. Наприклад, число 324_5 можна представити у вигляді (1.9) і знайти, що:

$$3\cdot 5^2 + 2\cdot 5^1 + 4\cdot 5^0 = 3\cdot 25 + 2\cdot 5 + 4\cdot 1 = 89_{10}.$$

Таблиця 1.1

Подання чисел в деяких системах числення

p = 10	p = 2	p = 5	p = 8	p = 16
0	0	0	0	0
1	1	1	1	1
2	10	2	2	2
3	11	3	3	3
4	100	4	4	4
5	101	10	5	5
6	110	11	6	6
7	111	12	7	7
8	1000	13	10	8
9	1001	14	11	9
10	1010	20	12	A
11	1011	21	13	B
12	1100	22	14	C
13	1101	23	15	D
14	1110	24	16	E
15	1111	30	17	F

Арифметичні операції в системі з основою p виконуються за правилами цієї системи. Наприклад, правила для складання $(a+b)$ і віднімання $(a-b)$ чисел в п’ятірковій системі наведені на рис. 1.1.

На підставі таблиці додавання (рис. 1.1а) можна побудувати таблицю для виконання операції множення чисел $a\times b$ (рис. 1.2).

$b\downarrow$	$a\rightarrow$	0	1	2	3	4
0		0	1	2	3	4
1		1	2	3	4	10
2		2	3	4	10	11
3		3	4	10	11	12
4		4	10	11	12	13

а)

$b\downarrow$	$a\rightarrow$	0	1	2	3	4
0		0	1	2	3	4
1		-1	0	1	2	3
2		-2	-1	0	1	2
3		-3	-2	-1	0	1
4		-4	-3	-2	-1	0

б)

Рис. 1.1 – Виконання операцій додавання (а) і віднімання (б) в системі $p=5$

$b\downarrow$	$a\rightarrow$	0	1	2	3	4
0		0	0	0	0	0
1		0	1	2	3	4
2		0	2	4	11	13
3		0	3	11	14	22
4		0	4	13	22	31

Рис. 1.2 – Виконання операції множення в системі числення з $p=5$

Добуток $a\times b$ отриманий на основі багаторазового складання, використовуючи правила, наведені на рис. 1.1а. Наприклад, результат множення 4×3 отримано у такий спосіб: 4 + 4 = 13 + 4 = 10 + (3 + 4) = 10 + 12 = 22.

Правила виконання основних операцій можуть бути отримані для будь-якої системи числення. Для цього необхідно побудувати відповідні таблиці. Правила подання на рис. 1.1 і рис. 1.2 можна використовувати для додавання і множення багаторозрядних п’ятіркових чисел. Наприклад, знайдемо результат операцій $(a+b)$, $(a-b)$ і $a\times b$ для чисел 23_5 і 14_5 (рис. 1.3):

а) $\begin{array}{r} 23 \\ 14 \\ \hline 42 \end{array}$; б) $\begin{array}{r} 23 \\ -14 \\ \hline 04 \end{array}$; в) $\begin{array}{r} 23 \\ 14 \\ \hline 202 \\ 23 \\ \hline 423 \end{array}$.

Рис. 1.3 – Виконання складання (а), віднімання (б) і множення (в) для $a = 23_5$ *і* $b = 14_5$

Для перевірки правильності виконання цих операцій можна виконати перевірку в десятковій арифметиці, використовуючи формулу (1.9) і правила десяткової арифметики:

$$\begin{aligned} 23_5 &= 2 \cdot 5^2 + 3 = 13_{10}; \\ 14_5 &= 1 \cdot 5^1 + 2 = 9_{10}; \\ 42_5 &= 4 \cdot 5^1 + 2 = 22_{10} = 13_{10} + 9_{10}; \\ 4_5 &= 4_{10} = 13_{10} - 9_{10}; \\ 432_5 &= 4 \cdot 5^2 + 3 \cdot 5^1 + 2 = 117_{10} = 13_{10} \cdot 9_{10}. \end{aligned} \qquad (1.10)$$

Розрахунки (1.10) підтверджують правильність результатів, наведених на рис. 1.3.

У повсякденному житті люди використовують в основному десяткову арифметику. При цьому 60-кова використовується для представлення часу і кутів. Знаходить застосування і 12-кова система. Наприклад, дюжинами рахують посуд і олівці, а для числа 12×12 існує спеціальна назва – «грос». Однак в ЕОМ всі обчислення проводяться на підставі двійкової системи, до розгляду якої ми переходимо.

1.2 Двійкова система числення

У двійковій системі числення кожен розряд може приймати тільки одне з двох значень – «0» або «1». Ця одиниця інформації називається біт (від англійського «bit» – binary digit). Інформація в пам'яті ЕОМ зберігається у вигляді груп бітів, які називаються словами. Число біт в слові різних ЕОМ різні. Наприклад, в деяких мікроконтролерах слово складається з чотирьох

біт, в разі ЕОМ Cray-1 слово включає 64 біти, а в графічних процесорах – 128 біт. Група з 8 біт називається байт (byte). Команди ЕОМ і слова інформації кратні байту, тобто можуть бути довжиною 2, 4 і 8 байт. Двійкові слова можуть представляти таку інформацію.

1. Команди ЕОМ. Команда може бути представлена у вигляді одного слова. Зв'язок між бітовим рядком і операцією, що виконує ЕОМ, визначається розробником комп'ютера. Зауважимо, що команди можуть представлятися і декількома словами, збереженими в пам'яті ЕОМ.

2. Числа. Слова інформації можуть представляти числа, задані в деякому форматі. Наприклад, у форматі BCD 001001102 відповідає десятковому числу 26. Якщо також послідовність біт визначає ціле двійкове число, то воно відповідає числу 38_{10}. Тут можна провести аналогію зі значенням слів в різних мовах. Наприклад, для англійців слово «gift» означає «подарунок», а для німця – «отрута».

3. Символи. В ЕОМ часто використовують операції, пов'язані з введенням або набором текстів. Зауважимо, що і програми ЕОМ вводяться у вигляді тексту. Для подання букв, цифр і спеціальних символів (+, –, !, ? тощо) використовуються двійкові коди, оскільки ЕОМ може обробляти тільки їх. На практиці переважно використовується семирозрядний код ASCII (American Standard Code for Information Interchange).

Цей код включає 7 біт, тобто дає змогу представити $2^7 = 128$ символів. З цих 128 символів 96 використовуються для друку (включно з малими та великими символами). Решта 32 символи використовуються для символів, що не друкуються, але мають деяке спеціальне призначення, наприклад, пробіл або повернення каретки друкарської машинки.

Наприклад, букві «А» відповідає код 1000001. Цей код вводиться в ЕОМ при натисканні на клавішу «А» у верхньому регістрі. Для друку літери «А» на принтер надсилається код 1000001. Аналогічно, десяткові цифри від 0 до 9 представляються такими ж цифрами від 0 до 9. Отже, при натисканні на клавішу «2» на клавіатурі відбувається введення бітового

рядка 00110010 в пам’ять ЕОМ. Очевидно, що при друці символів необхідно їхні коди перетворити у відповідні графічні зображення.

Для зображення символів з точками і штрихами (á, ö тощо) використовується 8-бітовий код ISO 8859-1 (Latine code). Для роботи з тисячами японських і китайських ієрогліфів використовується Unicode, що має 16 біт. Перші 256 символів цього коду відповідають ASCII. Зауважимо, що Unicode є стандартом мови JAVA.

4. Графічні елементи. Різні малюнки представляються в пам’яті ЕОМ у вигляді сукупності двійкових кодів. Існують різноманітні методи представлення графічної інформації. Наприклад, рисунок може бути параметризований і зберігатися в пам’яті як набір команд, які можуть бути використані для його відновлення. У цьому разі рисунок представляється в термінах ліній, дуг і поліномів, а також їхньому розташуванні у рисунку. Щоб надрукувати або показати рисунок на дисплеї, він має відновитися на основі своїх параметрів.

Однак більш складні рисунки подаються у вигляді бітового шаблона (bit-map). Будь-який рисунок може бути перетворений у квадратний масив елементарних зображень, які називаються пікселями («pixel» від англ. «Picture element»). Піксель є аналогом біта, але використовується для представлення найпростішої інформації, з якої складається рисунок. Однак, на відміну від біта, піксель може мати атрибути, такі як, наприклад, колір.

Щоб зрозуміти складність представлення графічної інформації, розглянемо рисунок формату А4 (210×297 мм). Для чіткості зображення необхідно, щоб в 1 мм було близько 12 пікселів. Відтак, один квадратний міліметр містить $12 \times 12 = 144$ пікселів, а вся картинка містить $210 \times 297 \times 144 = 8\,951\,040$ пікселів. Це число визначає обсяг пам’яті 1 Мбайт для зберігання монохромної (чорно-білої) картинки. Якщо ж рисунок має колір, а кожен колір має 256 відтінків, то для зберігання такого рисунка потрібно 8 Мбайт. Це пояснює високу ціну графічних комп’ютерів. Зауважимо, що існують методи стиснення інформації, що представляє рисунок.

Ціле число в двійковій системі представляється у вигляді:

$$A_2 = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_1 2^1 + a_0 2^0. \qquad (1.11)$$

Правильний дріб має вигляд:

$$A_2 = a_{-1} 2^{-1} + a_{-2} 2^{-2} + \ldots + a_{-m} 2^{-m}. \qquad (1.12)$$

Змішаний дріб представляється такою загальною формулою:

$$A_2 = \sum_{i=-m}^{n} a_i 2^i. \qquad (1.13)$$

Вибір двійкової системи для ЕОМ був зроблений, виходячи з аналізу таких факторів:

1. Наявність фізичних елементів, здатних зобразити символи системи. Елементи, які мають два стійкі стани, є найбільш простими. До них зараховують транзистори, реле, конденсатори. Цей критерій був одним з найбільш важливих для вибору двійковій системи.

2. Економічність системи, яка визначається кількістю обладнання, необхідного для подання багаторозрядних чисел і виконання операцій над ними. За цим критерієм найприйнятнішою є трійкова система, а двійкова посідає друге місце.

3. Трудомісткість виконання арифметичних операцій, що визначається складністю алгоритмів обробки інформації. При цьому, чим меншою є підстава системи, тим меншою буде і трудомісткість.

4. Швидкодія обчислювальних пристроїв виявляється тим вище, чим меншою є підстава системи числення. Відомо, що множення в двійковій системі виконується на 26 % швидше, ніж в трійковій, і в 2,7 раза швидше, ніж в десятковій.

5. Існування формальних методів синтезу та аналізу цифрових пристроїв, що важливо для розробки схем з деякими оптимальними характеристиками. У разі двійкової системи таким апаратом є алгебра логіки.

6. Зручність роботи людини з ЕОМ є високою тільки при використанні десяткової системи. Однак дані і результат досить просто переводяться з двійкової системи в десяткову систему і навпаки.

У табл. 1.1–1.3 наведені результати виконання операцій додавання, віднімання і множення відповідно для однорозрядних двійкових чисел a і b: $(a+b)$, $(a-b)$, $a \times b$.

Таблиця 1.2

Двійкове складання

$b\downarrow$	$a\rightarrow$	0	1
0		0	1
1		1	10

Таблиця 1.3

Двійкове віднімання

$b\downarrow$	$a\rightarrow$	0	1
0		0	1
1		-1	0

Таблиця 1.4

Двійкове множення

$b\downarrow$	$a\rightarrow$	0	1
0		0	0
1		0	1

Як видно з табл. 1.2, при додаванні однорозрядних чисел можливий результат, який має два розряди. Отже, з обсягом розрядної сітки ця одиниця використовується як перенесення між сусідніми розрядами при додаванні багаторозрядних чисел (див. підрозділ 1.4). З табл. 1.3 випливає, що при відніманні позитивних чисел результат може бути негативним. Ця ситуація відповідає позиченню одиниці зі старшого розряду числа (див. підрозділ 1.4). Таблиця для виконання операції ділення не приведена. Очевидно, що для $a:b$ результат дорівнює a за $b=1$, і результат буде нескінченним за $b=0$.

Фахівцям в області інформатики необхідно знати хоча б перші 10 ступенів числа 2. У табл. 1.5 показані ці числа для перших 16 ступенів числа 2. Крім того, дані терміни, що використовуються для назви цих чисел як масиву інформації в пам’яті ЕОМ.

Таблиця 1.5

Цілі ступені числа 2

p	2p	термін	p	2p	термін
1	2	1 біт	9	512	64 байти
2	4	напівбайт	10	1024	1 кбіт
3	8	байт	11	2048	2 Кбіти
4	16	2 байти	12	4096	4 Кбіти
5	32	4 байти	13	8192	1 кбайт
6	64	8 байтів	14	16384	2 кбайти
7	128	16 байтів	15	32768	4 кбайти
8	256	32 байти	16	65536	8 Кбайт

1.3 Переведення чисел з однієї позиційної системи в іншу

Це завдання постійно виникає при обробці цифрової інформації в ЕОМ. При цьому практичне значення має завдання переведення з двійкової системи в десяткову систему і навпаки. У загальному вигляді задача переведення має такий вигляд:

нехай число A представлено в системі числення з основою p_1, що містить коефіцієнти $0 \le a_i \le p_1 - 1$, де $-m \le i \le n$:

$$A_{p_1} = \sum_{i=-m}^{n} a_i p_1^i . \tag{1.14}$$

Необхідно знайти представлення цього числа в системі числення з основою p_2, що містить коефіцієнти $0 \le b_i \le p_2 - 1$, де $-s \le i \le k$:

$$A_{p_2} = \sum_{i=-s}^{k} b_i p_2^i . \tag{1.15}$$

Для знаходження коефіцієнтів b_i необхідно виконати низку дій, заснованих на правилах арифметики в системі p_1. Методи переведення цілих чисел засновані на поділі, дрібних – на множенні числа A_{p_1} на число p_2, представлене в системі p_1.

Переведення цілих чисел. Ціле число A_{p_1} в системі p_2 може бути записане у вигляді:

$$A_{p_2} = b_k p_2^k + b_{k-1} p_2^{k-1} + \ldots + b_1 p_2^1 + b_0 .$$

Використовуючи схему Горнера (послідовне винесення p_2 за дужки), можна отримати такий вираз:

$$A_{p_2} = \left(\ldots\left(b_k p_2 + b_{k-1}\right)p_2 + \ldots + b_1\right)p_2 + b_0 . \tag{1.16}$$

Вираз (1.16) можна представити у вигляді:

$$A_{p_2} = A_1 p_2 + b_0 , \tag{1.17}$$

де значення A_1 відоме з (1.16). Розділивши (1.17) на число p_2, знайдемо цілу частину A_1 і залишок b_0, тобто молодшу цифру числа A_{p_2}. Очевидно, залишок A_1 представляється у вигляді:

$$A_1 = (\ldots(b_k p_2 + b_{k-1})p_2 + \ldots + b_2)p_2 + b_1, \qquad (1.18)$$

що можна уявити у такій формі:

$$A_1 = A_2 p_2 + b_1. \qquad (1.19)$$

Отже, при розподілі A_1 на число p_2 отримаємо цілу частину A_2 і залишок b_1. Процес повторюється доти, доки не буде знайдений коефіцієнт $b_k < p_2$. Це і буде старша цифра числа A_{p_2}.

Приклад 1.1. Перевести десяткове число $A = 28$ в двійкову систему числення.

Рішення: маємо $p_2 = 2$. Виконаємо ділення в стовпчик:

$$\begin{array}{r|l} 28 & 2 \\ \underline{28} & 14 \\ 0 = b_0 & \end{array} \qquad \begin{array}{r|l} 14 & 2 \\ \underline{14} & 7 \\ 0 = b_1 & \end{array} \qquad \begin{array}{r|l} 7 & 2 \\ \underline{6} & 3 \\ 1 = b_2 & \end{array} \qquad \begin{array}{r|l} 3 & 2 \\ \underline{2} & 1 = b_4 \\ 1 = b_3 & \end{array}$$

Відповідь: $A_2 = 11100$.

Перевірка: $A_{10} = 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 = 16 + 8 + 4 = 28_{10}$.

Приклад 1.2. Перевести двійкове число $A_2 = 1110011$ в десяткову систему числення. При цьому підстава $p_2 = 10$ зображується в двійковій формі $p_2 = 1010$.

Рішення:

$$\begin{array}{r|l} 1110011 & 1010 \\ \underline{1010} & 1011 \\ 10001 & \\ \underline{1010} & \\ 001111 & \\ \underline{1010} & \\ 0101 = b_0 & \end{array}$$

$$\begin{array}{r|l} 1011 & 1010 \\ \underline{1010} & 1 = b_2 \\ 0001 = b_1 & \end{array}$$

Відповідь: отже, $b_0 = 0101_2 = 5_{10}$, $b_1 = b_2 = 1$. Відтак, $A_{10} = 115$.

Перевірка:

$$A_2 = 1\cdot 2^6 + 1\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^1 = 64 + 32 + 16 + 2 + 1 = 115_{10}.$$

Приклад 1.3. Перевести десяткове число $A_{10} = 4629$ у вісімкову систему числення ($p_2 = 8$).

Рішення:

$4629 \mid \underline{8}$	$578 \mid \underline{8}$	$72 \mid \underline{8}$	$9 \mid \underline{8}$
$\underline{5624} \mid 576$	$\underline{576} \mid 72$	$\underline{72} \mid 9$	$\underline{8} \mid 1 = b_4$
$b_0 = 5$	$b_1 = 2$	$b_2 = 0$	$b_3 = 1$

Відповідь: $A_8 = 11025.$

Перевірка: $A_{10} = 1 \cdot 8^4 + 1 \cdot 8^3 + 2 \cdot 8^1 + 5 = 4096 + 512 + 16 + 5 = 4629_{10}.$

Зауважимо, що для виконання останньої перевірки доцільно використовувати схему Горнера. В цьому разі десяткове число представляється у вигляді

$$A_{10} = (b_0 + 8(b_1 + 8(b_2 + \ldots))). \qquad (1.20)$$

Вираз (1.20) приводить до такої схеми обчислень. Необхідно взяти старший розряд b_k, помножити його на 8 і додати до результату b_{k-1}. Цей процес триває циклічно до додавання до останнього добутку цифри b_0. Перевіримо цей метод на прикладі числа 110258. Виконуємо таку послідовність дій:

$$1 \cdot 8 + 1 = 9;\ 9 \cdot 8 + 0 = 72;\ 72 \cdot 8 + 2 = 578;$$

$$578 \cdot 8 + 5 = 4624 + 5 = 4629_{10}.$$

Приклад 1.4. Перевести десяткове число $A = 53241$ в шістнадцяткову систему ($p_2 = 16$).

Рішення:

$53241 \mid \underline{16}$	$3327 \mid \underline{16}$	$207 \mid \underline{16}$
$\underline{53232} \mid 3327$	$\underline{3312} \mid 207$	$\underline{192} \mid 12 = b_3$
$b_0 = 9$	$b_1 = 15$	$b_2 = 15$

Відповідь: оскільки $15_{10} = F_{16}$, а $12_{10} = C_{16}$, то число A_{10} представляється у вигляді $A_{16} = CFF9$.

Для перевірки правильності результату скористаємося виразом (1.21), замінивши в ньому 8 на 16:

$$C \cdot 16 + F = 12 \cdot 16 + 15 = 207;$$

$$207 \cdot 16 + F = 3312 + 15 = 3327;$$

$$3327 \cdot 16 + 9 = 53232 + 9 = 53241.$$

Отже, $53241_{10} = CFF9_{16}$.

Переведення правильних дробів. Завдання формулюється за аналогією до попереднього завдання. Нехай число A в системі p_1 має вигляд

$$A_{p_1} = a_{-1}p_1^{-1} + a_{-2}p_1^{-2} + \ldots + a_{-m}p_1^{-m}.$$

Це число необхідно представити в системі з основою p_2, використовуючи коефіцієнти $0 \le b_i \le p_2 - 1$. Отже, необхідно знайти представлення

$$A_{p_2} = b_{-1}p_2^{-1} + b_{-2}p_2^{-2} + \ldots + b_{-k}p_2^{-k}. \qquad (1.21)$$

Перепишемо вираз (1.21) за схемою Горнера і отримаємо:

$$A_{p_2} = p_2^{-1}\left(b_{-1} + p_2^{-1}\left(b_{-2} + \ldots + p_2^{-1}b_{-k}\right)\ldots\right). \qquad (1.22)$$

З виразу (1.22) випливає метод переведення. Помножимо A_{p_2} на число p_2 (за правилами системи p_2). При цьому отримаємо результат

$$A \cdot p_2 = b_{-1} + p_2^{-1} \cdot A_1, \qquad (1.23)$$

де $A_1 = \left(b_{-2} + \ldots + p_2^{-1}\left(b_{-3} + \ldots + p_2^{-1}b_{-k}\right)\ldots\right)$. Отже, ціла частина числа (1.23) є коефіцієнтом b_{-1}. Помножимо дріб з (1.23) на число p_2 і отримаємо

$$A_1 \cdot p_2 = b_{-2} + p_2^{-1} \cdot A_2.$$

Отже, отриманий коефіцієнт b_{-2}. Цей процес триває доти, доки не будуть отримані всі коефіцієнти з (1.21).

Труднощі переведення полягають в тому, що результат переведення може бути нескінченним. Відтак, переведення припиняється або за рівності дробової частини залишку виду (1.23) нулю, або за отримання дробу із

заданим числом розрядів. Природно це число визначається розрядом сітки ЕОМ.

Приклад 1.5. Перевести десятковий дріб $A_{10} = 0.6875$ в двійкову систему числення.

Рішення:

$$0{,}6875 \times 2 = 1{,}3750, \text{ тобто } b_{-1} = 1;$$

$$0{,}3750 \times 2 = 0{,}7500, \text{ тобто } b_{-2} = 0;$$

$$0{,}7500 \times 2 = 1{,}5000, \text{ тобто } b_{-3} = 1;$$

$$0{,}5000 \times 2 = 1{,}0000, \text{ тобто } b_{-4} = 1.$$

Оскільки дрібна частина результат четвертого кроку обчислень дорівнює нулю, процес закінчено.

Відповідь: $0{,}6875_{10} = 0{,}1011_2$.

Перевірка:

$$A_{10} = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = 0{,}5 + 0{,}125 + 0{,}00625 = 0{,}6875.$$

Приклад 1.6. Перекласти двійковий дріб $A_2 = 0.1101$ в десяткову систему числення.

Рішення:

$$0{,}1101 \times 1010 = 1000{,}0010, \text{ тобто } b_{-1} = 8;$$

$$0{,}0010 \times 1010 = 0001{,}0100, \text{ тобто } b_{-2} = 1;$$

$$0{,}0100 \times 1010 = 0010{,}1000, \text{ тобто } b_{-3} = 2;$$

$$0{,}1000 \times 1010 = 0101{,}0000, \text{ тобто } b_{-4} = 5.$$

Оскільки дрібна частина результат четвертого кроку множення на число $10_2 = 1010$ дорівнює нулю, процес закінчено.

Відповідь: $0{,}1101_2 = 0{,}8125_{10}$.

Перевірка: перевірку можна виконати, як в прикладі 1.5. Однак можна вчинити в інший спосіб. У десятковій системі число тисяча сто один дорівнює 13, тобто дріб можна знайти як 13 : 16 = 0,8125. Аналогічно,

у прикладі 1.5 перевірку можна провести, поділивши 11 (десятковий еквівалент числа 1011) на 16.

Приклад 1.7. Перевести десятковий дріб $A_{10} = 0{,}12$ у вісімкову систему числення.

Рішення:

$$0{,}12 \times 8 = 0{,}96, \text{ тобто } b_{-1} = 0;$$

$$0{,}96 \times 8 = 7{,}68, \text{ тобто } b_{-2} = 7;$$

$$0{,}68 \times 8 = 5{,}44, \text{ тобто } b_{-3} = 5;$$

$$0{,}44 \times 8 = 3{,}52, \text{ тобто } b_{-4} = 3.$$

Відповідь: за довжини розрядної сітки чотири, маємо відповідність $0{,}12_{10} = 0{,}0753_8$, тобто є похибка в поданні вихідного дробу, яка визначається як $0{,}000041_8$.

Приклад 1.8. Перевести десятковий правильний дріб $A_{10} = 0{,}12$ в шістнадцяткову систему з точністю до чотирьох розрядів.

Рішення:

$$0{,}12 \times 16 = 1{,}92, \text{ тобто } b_{-1} = 1;$$

$$0{,}92 \times 16 = 14{,}72, \text{ тобто } b_{-2} = E;$$

$$0{,}72 \times 16 = 11{,}52, \text{ тобто } b_{-3} = B;$$

$$0{,}52 \times 16 = 8{,}32, \text{ тобто } b_{-4} = 8.$$

Відповідь: за заданої точності $0{,}12_{10} = 0{,}1EB8_{16}$.

Перевірку правильності переведення в двох останніх прикладах ми залишаємо читачеві.

При переведенні змішаних дробів їхня ціла частина переводиться шляхом розподілу, а старша – множення на основу p_2.

Приклад 1.9. Перевести десяткове число $A_{10} = 28{,}6875$ в двійкову систему числення.

Рішення: з прикладу 1.1 можемо отримати цілу частину (11100), з прикладу 1.5 маємо дробову частину (0,1011).

Відповідь: $28{,}6875_{10} = 11100{,}1011_2$.

Використання проміжної системи числення. Цей підхід використовується для переведення із двійкової системи в десяткову систему і навпаки. Як проміжні системи використовуються вісімкова (наразі все рідше) і шістнадцяткова системи. Як видно з табл. 1.1, одна вісімкова цифра замінює три двійкові (тріаду), а одна шістнадцяткова – чотири (тетраду). З цієї причини числа у вісімковій і шістнадцятковій системах в три і чотири рази коротше відповідно, ніж у двійковій.

Сьогодні 16-кова система широко використовується для представлення чисел в комп'ютерах, а також при роздруківці вмісту пам'яті ЕОМ. Здебільшого, роздруківка частини пам'яті ЕОМ (core-dump) використовується при діагностиці несправностей, які неможливо знайти автоматизованими методами. Однак бувають ситуації, коли і для людини двійкова система буває більш інформативною, ніж інші.

Наприклад, нехай якийсь хімічний процес охоплює використання трьох нагрівачів (H1, H2, H3), трьох засувок (V1, V2, V3) і двох насосів (P1, P2, P3). Для управління цими пристроями використовується байт інформації. Якщо деякий розряд цього байту дорівнює нулю, то відповідний йому агрегат вимкнений. Якщо цей розряд дорівнює одиниці, то – ввімкнений. Нехай розряди цього байту відповідають пристроям в такому порядку H1, H2, H3, V1, V2, V3, P1, P2. При цьому байт 10110101 однозначно вказує на стан апаратури. При цьому інші представлення цього числа, а саме 268, 5_{16} або 93_{10}, не дають нічого.

При переведенні числа A_8 в A_2 необхідно відобразити кожну вісімкову цифру у вигляді тріади. Наприклад, $A_8 = 376_8$ відповідає $A_2 = 11111110_2$ (див. табл. 1.1). При зворотному переведенні двійкове число розбивається на тріади, починаючи з правого крайнього розряду. Потім кожна тріада замінюється її вісімковим еквівалентом. Наприклад, $A_2 = 11011010_2$ розбивається на 3 тріади $\underline{11}\ \ \underline{011}\ \ \underline{010}$, чому відповідає число $A_8 = 332_8$.

При переведенні числа A_{16} в A_2 необхідно відобразити кожну 16-кову цифру у вигляді тетради. Наприклад, $A_{16} = A2B_{16}$ відповідає коду $A_2 = 101000101011_2$. При зворотному переведенні двійкове число, що починається з самого правого розряду, розбивається на тетради. Далі кожна тетрада замінюється своїм 16-ковим еквівалентом. Наприклад, число $A_2 = 1010110110100011_2$ розбивається на чотири тетради 1010 1101 1010 0011, які перетворюються в $A_{16} = ADA3_{16}$.

1.4 Різновиди двійкових систем

Двійкова система може використовуватися для представлення десяткових чисел, причому кожна десяткова цифра кодується за допомогою чотирьох двійкових розрядів. Такі системи називаються двійково-десятковими (аналітичний варіант BCD – binary-coded decimal). Такий підхід спрощує завдання переведення чисел з десяткової системи в двійкову систему і назад. Однак, з теорії двійкової арифметики відомо, що для представлення однієї десяткової цифри достатньо близько 3,3 біт. Отже, використання двійково-десяткової арифметики веде до більших розрядних чисел, ніж двійкової арифметики. Це зі свого боку підвищує вимоги до ємності пам’яті, що зберігає числа.

Двійково-десяткова система є прикладом кодування позиційної системи, в якій цифри однієї системи числення кодуються цифрами іншої системи. Розрізняють два підходи до кодування – з природними розрядами і зі штучними вагами розрядів. Прикладом першої системи є BCD-система з вагами розрядів 8-4-2-1, тобто 2^3-2^2-2^1-2^0. Кодування десяткових цифр в системі 8-4-2-1 наведено в табл. 1.6.

Таблиця 1.6

Подання десяткових цифр

Десяткова цифра	Код 8-4-2-1	Код 2-4-2-1	Код 8-4-2-1
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	1010	0111
5	0101	0101	1000
6	0110	1100	1001
7	0111	тисяча сто один	1010
8	1000	1110	1011
9	1001	1111	1100

Системі 8-4-2-1 притаманний суттєвий недолік, пов'язаний з виконанням арифметичних операцій. Наприклад, 5 + 7 = 12, тобто є перенесення, і результат в даному розряді дорівнює 2. При додаванні ж чисел 0101 і 0111 виходить результат: 1100, тобто 1210. При цьому перенесення відсутнє, і його необхідно створити штучно, що ускладнює апаратуру суматора. Перевагою коду 8-4-2-1 є простота подання інформації.

Цього недоліку позбавлена система з вагами 2-4-2-1, яка належить системі зі штучними вагами розрядів. Подання чисел в цій системі наведене в табл. 1.6. Наприклад, число 146810 представляється у вигляді такої послідовності біт: 0001 0100 1100 1110. У цій системі складання 5 + 7 = = 0101 + 1101 = 10010, тобто результат дорівнює 2. Це зумовлено тим, що система 2-4-2-1 є самодоповнюваною. Код називається самодоповнюваним, якщо з рівності $a_{10}+b_{10}=9$ випливає рівність $a_2+b_2=1111$. Наприклад, 5 + 4 = 9, а 0101 + 1010 = 1111. Це спрощує проблему формування переносу між розрядами, але викликає труднощі переведення.

Ще одним прикладом самодоповнюваного коду є так званий «код з надлишком 3», тобто система 8-4-2-1+3 (табл. 1.6). Ця система дає змогу спростити арифметичні операції, а при переведенні з десяткової системи

достатньо додати до коду 8-4-2-1 число 3 = 0011. Очевидно, це складання уповільнює переведення чисел. Крім того, при зворотному переведенні число 3 необхідно відняти з кожної тетради коду BCD. Система 8-4-2-1+3 має непостійні ваги розрядів. Наприклад, цифра $1_{10}=0100_2$, тобто третій розряд має вагу 1. У числі ж $9_{10}=1100_2$ цей розряд має вагу 4. Зауважимо, що табл. 1.6 задає однорідні коди, оскільки в кожному розряді знаходиться тільки 0 або 1.

Відомі виконавчі системи спеціального призначення, створені для прискорення обчислень на комп'ютерах. Однак всі ці системи характеризуються труднощами переведення чисел з класичних позиційних систем і назад. Відтак їх застосовують тільки у разі, якщо не потрібно змінювати систему при введенні-виведенні даних, або якщо переведення здійснюється без залучення складної апаратури.

Прикладом такої системи є врівноважена трійкова система, де $p=3$ і використовуються цифри 1, 0, $\overline{1}$. Число $\overline{1}$ означає -1. У цій системі будь-яке дійсне число (позитивне чи негативне) представляється без знаку. Розглянемо запис деяких чисел в цій системі:

$$10\overline{1}=1\cdot 3^2+0\cdot 3^1-1\cdot 3^0=9-1=8_{10};$$

$$1\overline{1}\overline{1}0,1\overline{1}=1\cdot 3^3-1\cdot 3^2-1\cdot 3^1+1\cdot 3^0+1\cdot 3^{-1}-1\cdot 3^{-2}=27-9-3+\frac{1}{3}-\frac{1}{9}=15,\frac{2}{9}_{10};$$

$$\overline{1}110,\overline{1}1=-27+9-3-\frac{1}{3}+\frac{1}{9}=-15,\frac{2}{9}_{10}.$$

З цих прикладів видно основні переваги системи 1, 0, $\overline{1}$, а саме:

– знак числа визначається його першим значущим розрядом (1 – відповідає +, а $\overline{1}$ – негативне число);

– для переходу до числа з протилежним знаком необхідно поміняти 1 на $\overline{1}$ і навпаки;

– для отримання цілого числа (округлення до найближчого цілого) досить відкинути дробову частину.

Для підсумовування багаторозрядних чисел достатньо знати такі правила:

$$\bar{1}+\bar{1}=(-3)+(-3)=-6=\bar{1}1;1+\bar{1}=0;1+1=3+3=6=1\bar{1}.$$

Приклад 1.10. Знайти результат додавання чисел 7 + 2 і 7 + (–2) в системі 1, 0, $\bar{1}$.

Рішення: число $7_{10}=1\bar{1}1$, число $2_{10}=01\bar{1}$, число $-2_{10}=0\bar{1}1$.

$$\begin{array}{lll} 1\bar{1}1 & & 1\bar{1}1 \\ \underline{01\bar{1}} & ; & \underline{0\bar{1}1} \\ 100 \quad =3^2=9_{10} & & 1\bar{1}\bar{1} \quad =3^2-3^1-1=5_{10} \end{array} .$$

Віднімання в цій системі зводиться до складання з числом, яке протилежно віднімається. Наприклад, $7-2=7+(-2)=1\bar{1}1+0\bar{1}1=1\bar{1}\bar{1}$ (з прикладу 1.10). Для виконання множення достатньо врахувати три правила:

– при множенні на 0 частковий добуток дорівнює нулю;

– при множенні на «мінус 1» знак часткового добутку змінюється на протилежний, що дає негативний множник;

– при множенні на 1 частковий добуток збігається з множником.

Приклад 1.11. Помножити числа 19_{10} і 33_{10}, представлені в трійковій системі числення.

Рішення: $19_{10}=1\bar{1}01=3^3-3^2+1$; $33_{10}=11\bar{1}0=27+9-3$;

$$\begin{array}{r} 1\bar{1}01 \\ 11\bar{1}0 \\ \hline 0000 \\ \bar{1}10\bar{1} \\ 1\bar{1}01 \\ \underline{1\bar{1}01} \\ 10\bar{1}\bar{1}1\bar{1}0 \end{array}$$

Відповідь:

$10\bar{1}\bar{1}1\bar{1}0=3^6-3^4-3^3+3^2-3^1=729-81-27+9-3=627_{10}$.

Перевірка: $33\times19=627$, тобто результат правильний.

Якщо числа представлені в двійковому коді, то при переході від зображення одного числа до зображення сусіднього числа може відбуватися одночасна зміна чисел в розрядах числа. Наприклад, при переході від $7_{10} = 0111$ до $8_{10} = 1000$ змінюються всі чотири значення. Це може призвести до збоїв в роботі апаратури.

Для усунення цього недоліку використовується код Грея (табл. 1.7), розряди якого не мають постійної ваги. Як видно з табл. 1.7, в числі 710 одиниця третього розряду має вагу 7, а в 4_{10} – три або один. Цей код має кілька переваг перед звичайним кодом 8-4-2-1, а саме:

– при послідовному переході від числа до числа змінюється тільки один розряд коду;

– зміна значень кожного розряду відбувається вдвічі швидше, ніж в коді 8-4-2-1 (при послідовному переході між числами).

Ці особливості використовуються, наприклад, при побудові оптичних пристроїв кодування, що перетворюють кут повороту в двійковий код. Такі пристрої дають змогу вимірювати кут (знаходити кутове положення) без механічного зв’язку між валом і вимірювальним приладом. Типовий приклад використання оптичного кодера – автоматична система прогнозу погоди. Зокрема, напрямок вітру в цьому разі визначається кутом повороту флюгера. Цей флюгер знаходиться на валу, пов’язаному з оптичним кодером, що переводить кут повороту валу (напрямок вітру) в двійковий код.

Таблиця 1.7

Код Грея для десяткових чисел

A10	код Грея	A10	код Грея
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

Оптичний кодер включає пластиковий диск, вкритий концентричними доріжками. На доріжках виділяються сектори, що пропускають (логічна 1) і не пропускають (логічний 0) світло. Ці доріжки відповідають розрядам двійкового коду. За кожною доріжкою знаходиться джерело світла. На іншій стороні диска знаходяться фотоелектричні сенсори, по одному навпроти кожного джерела світла. Якщо сенсор освітлений (прозорий сектор доріжки), то фіксується одиниця у відповідному розряді коду сектора повороту. У табл. 1.8 показані можливі кодування при розбитті диска на 8 секторів.

Фізичні елементи системи мають низку недоліків. По-перше, фотоелектричні елементи не можуть бути встановлені абсолютно точно проти джерел світла. По-друге, ці джерела не є точковими, тобто їхнє світло розсіюється і може потрапити на дві доріжки. По-третє, межі секторів не є ідеальними прямими. Під час використання двійкового коду 421 можлива ситуація зміни декількох розрядів коду одночасно. Внаслідок недосконалості обладнання один із розрядів може змінитися раніше за інший. Наприклад, перехід від коду 001 до коду 010 може проходити як послідовність 001, 000, 010. Якщо комбінація 000 буде зафіксована, то це може призвести до збою в роботі апаратури. Під час використання коду Грея такі ситуації виключені, оскільки завжди змінюється тільки один розряд коду (змінюється фотопроникність тільки однієї доріжки).

Таблиця 1.8

Кодування інформації в оптичному коді

Сектор	Кут повороту	Двійковий код 421	Код Грея
0	0–45	000	000
1	45–90	001	001
2	90–135	010	011
3	135–180	011	010
4	180–225	100	110
5	225–270	101	111
6	270–315	110	101
7	315–360	111	100

Недоліком позиційних систем числення з непостійними вагами розрядів (зокрема і коду Грея) є складність обробки інформації. Відтак для роботи з такими кодами необхідне їхнє перекодування при введенні в ЕОМ і виведенні з неї.

Недоліком позиційних систем числення є наявність міжрозрядних переносів (позичок) при виконанні арифметичних операцій над багаторозрядними числами. Це обмежує швидкість виконання операцій або призводить до ускладнення апаратури, оскільки вводяться блоки прискорення операцій. Цих недоліків позбавлені символічні системи числення, в яких цифри є символами, кожен з яких окремо ніяк не характеризує окреме число. Характерним прикладом символьної системи є система залишкових класів (СОК), в якій числа представляються в залишках.

Число в СОК зображується у вигляді залишків від ділення заданого числа A_{10} на ряд взаємно простих чисел $s_1, s_2, \ldots, s_n$. При цьому утворюється число з вагами розрядів, що відповідно дорівнюють $s_1, s_2, \ldots, s_n$. Отож, число A_{10} можна представити у вигляді $a_1, a_2, \ldots, a_n$, де

$$\begin{aligned} a_i &= A_{10} - k_i s_i; \\ k_i &= \left[A_{10} / s_i \right]. \end{aligned} \qquad (1.24)$$

У виразі (1.24) [a] означає цілу частину числа a. Цей вираз покладено в основу табл. 1.9, що представляє перші 16 чисел десяткової системи в СОК з вагами розрядів $s_1 = 2$, $s_2 = 3$ і $s_3 = 5$, тобто СОК <2, 3, 5>.

У СОК операції рахунку, додавання, віднімання і множення виконуються порозрядно. Такі системи застосовуються в спеціалізованих ЕОМ зі строго фіксованим діапазоном вихідних чисел і проміжних результатів. При цьому операції ділення повинні виконуватися вкрай рідко.

Таблиця 1.9

Подання десяткових чисел в СОК <2, 3, 5>

Число	a1	a2	a3	Число	a1	a2	a3
0	0	0	0	8	0	2	3
1	1	1	1	9	1	0	0
2	0	2	2	10	0	1	0
3	1	0	3	11	1	2	1
4	0	1	4	12	0	0	2
5	1	2	0	13	1	1	3
6	0	0	1	14	0	2	4
7	1	1	2	15	1	0	0

Матеріал, розглянутий у цьому розділі, дає змогу навести класифікацію системи числення (рис. 1.4).

У [2] наведені корисні навички поводження з двійковими числами, які повинен вміти фахівець в області інформатики. Це такі прийоми:

1. Число 1000 ... 0 = 2k, де k – число нулів в числі. Необхідно знати напам’ять хоча б перші дванадцять ступенів двійки (табл. 1.5).

2. Число 11 ... 11 = 2k – 1, де k – число одиниць в записі числа.

3. Необхідно знати напам’ять десяткові значення двійкових чисел від 0 до 31. Надалі будемо їх називати «малими числами».

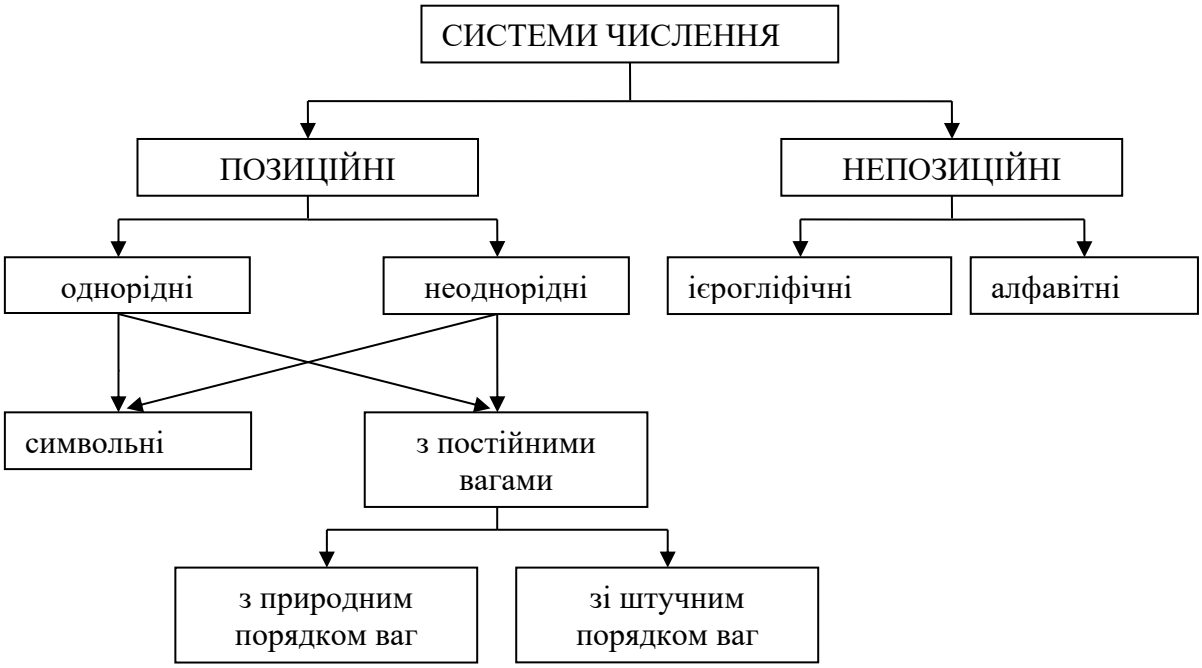


Рис. 1.4 – Класифікація систем числення

Знаючи напам’ять малі числа, легко перевірити правильність операції додавання (віднімання) двійкових чисел, якщо їх можна розбити на групи, не пов’язані між собою переносами (позичками).

Приклад 1.14. Необхідно перевірити результат додавання чисел $A = 1011101011$ і $B = 10101110$, що дорівнює 1110011001.

Рішення: при додаванні немає перенесення з п’ятого розряду, тобто числа можна розбити на дві групи, молодша з яких має 5 розрядів

$$\begin{array}{r|l} 10111 & 01011 \\ 101 & 01110 \\ \hline 11100 & 11001 \end{array} \equiv \begin{array}{r} 23 \\ 5 \\ \hline 28 \end{array} + \begin{array}{r} 11 \\ 14 \\ \hline 25 \end{array} .$$

Відповідь: порівняння лівих і правих частин результату показує правильність складання.

4. Читання двійкових дробів проводиться у такий спосіб. Розряди праворуч від коми читаються як ціле число, яке є чисельником. Знаменник дробу читається як ціле число, що є ступенем двох, показник якого дорівнює номеру молодшого розряду.

Приклад 1.15. Число $A = 0{,}1011 = \frac{11}{16}$ (одинадцять шістнадцятих);

$A = 0{,}00110110 = \frac{54}{256}$ (п’ятдесят чотири двісті п’ятдесят шостих).

Дріб $A = 0{,}00\ldots01$, що включає $n-1$ нулів після коми, дорівнює 2^{-n}. Дріб $A = 0{,}11\ldots1$, що включає n одиниць після коми, дорівнює $1-2^{-n}$.

5. Двійкові дроби можуть бути періодичними. Наприклад, дріб виду $1/(2^k-1)$ включає групу з $k-1$ нулів і однієї одиниці. Аналогічно, дріб виду $1/(2^k+1)$ включає групу з k одиниць.

Приклад 1.16.

$$\frac{1}{3} = \frac{1}{2^1+1} = 0{,}01\ldots = 0{,}(01); \frac{1}{7} = \frac{1}{2^3-1} = 0{,}(001)\ldots = 0{,}001001\ldots;$$

$$\frac{1}{17} = \frac{1}{2^4+1} = 0{,}(00001111).$$

Список літератури до розділу 1

1. Ковриженко Г. А. Системы счисления и двоичная арифметика. Киев: Рад. шк.,1984. 79 с.

2. Прикладная теория цифровых автоматов / К. Г. Самойлов, А. М. Романкевич, В. Н. Валуйский, Ю. С. Каневский, М. М. Пиневич. Киев: Вища шк., 1987. 375 с.

3. Савельев А. Я. Прикладная теория цифровых автоматов. Москва: Высш. шк., 1987. 272 с.

4. Clements A. The Principles of Computer Hardware. Oxford: Oxford University Press, 2000. 718 p.

2 БУЛЕВА АЛГЕБРА І ЇЇ ЗАСТОСУВАННЯ

1.2 Базові булеві функції

Як уже зазначалося, інформація всередині ЕОМ представляється у вигляді сукупності біт, тобто в двійковій формі. Для синтезу схем, що обробляють двійкову інформацію, використовується спеціальний математичний апарат, що називається булевою алгеброю. Елементами булевої алгебри є булеві константи, булеві змінні і булеві операції. Існують дві булеві константи, які прийнято позначати 0 і 1. Ці константи не розглядаються як числа, і для їхнього позначення можна використовувати будь-які слова (наприклад, є й немає, істина і брехня, true і false тощо). Булеві змінні можуть приймати тільки значення булевих констант, тобто 0 або 1. Отже, змінна x_l називається булевою змінною, якщо і тільки якщо $x_l \in \{0,1\}$.

Булева алгебра охоплює три операції, а саме: заперечення ($\neg$), кон'юнкцію ($\wedge$) і диз'юнкцію ($\vee$). Ці операції представлені в табл. 2.1 для булевих змінних a і b.

Таблиця 2.1

Операції булевої алгебри

a	b	заперечення a	заперечення b	$a \wedge b$	$a \vee b$
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	0	0	1
1	1	0	1	1	1

Часто заперечення називається операцією «НЕ», і вираз $\neg a$ читається як «НЕ a». Для стислості це позначається як $\overline{a}$. Операція $\wedge$ часто називається логічним множенням, а також операцією «І». При цьому «a *і* b» позначається як $a \& b$, $a \cdot b$ або ab. Операція $\vee$ часто називається логічним складанням, а також операцією «АБО». При цьому «*a або b*»

позначається як $a+b$. Очевидно, кон'юнкція довільного числа булевих змінних дорівнює одиниці, якщо і тільки якщо всі ці змінні дорівнюють одиниці. Аналогічно диз'юнкція довільного числа булевих змінних дорівнює нулю, якщо і тільки якщо всі ці змінні дорівнюють нулю. Отже, кон'юнкцію (диз'юнкцію) іноді називають операцією вибору мінімального (максимального) значення змінних.

Булеву формулу можна визначити у такий спосіб:

– кожна булева змінна і константа є формулою;

– якщо A і B – формули, то формулами є вирази $\neg A$, $\neg B$, $A \wedge B$, $A \vee B$;

– інших формул немає.

Булеві формули визначають особливий клас функцій, а саме булевих. Для задання порядку виконання операцій у формулі використовуються дужки. Якщо дужок немає, то першою виконується операція $\neg$, потім $\wedge$, і останньою – операція $\vee$. Зауважимо, що булева функція приймає тільки одне з двох фіксованих значень. Нехай функція $f(x)$ залежить від L булевих змінних від x_1 до x_L. З цих змінних можна утворити послідовність двійкових векторів, відповідних десятковим числам від 0 до $2^L - 1$. Отже, існує

$$n_L = 2^L \tag{2.1}$$

різних наборів L булевих змінних. На кожному з цих наборів функція може приймати тільки одне з двох значень. Отже, існує

$$n(L) = 2^{n_L} \tag{2.2}$$

різних булевих функцій L булевих змінних. Це число швидко зростає зі зростанням L. Наприклад, за $L=0$ маємо $n(0)=2$, за $L=1$ маємо $n(1)=2$, $L=2$ дає $n(2)=16$, а за $L=5$ маємо $n_L=32$, і існує 4.294967296 різних булевих функцій. У табл. 2.2 наведені всі шістнадцять булевих функцій для двох змінних, тобто для $L=2$.

Таблиця 2.2

Булеві функції двох змінних

a	b	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Розглянемо функції з табл. 2.2, що мають практичне значення для синтезу схем блоків ЕОМ:

1. Функції f_1 і f_{16} є відповідно константами 0 і 1. Ці функції не залежать від значень змінних a і b, тобто ці змінні є несуттєвими для функцій f_1 і f_{16}.

2. Функція $f_2 = a \wedge b = a \& b = a \cdot b = ab$, тобто це кон'юнкція a і b. Цю функцію можна записати і для довільного числа змінних $x_l \in X = \{x_1, \ldots, x_L\}$ в такому вигляді:

$$f_\wedge = \bigwedge_{l=1}^{L} x_l .$$

3. Функція $f_4 = a$ і функція $f_6 = b$ повторюють значення відповідних змінних. При цьому функція f_4 має несуттєву змінну b, а функція $f_6 - a$.

4. Функція f_7, що дорівнює одиниці при розбіжності значень a і b. Ця функція називається «нерівнозначністю», «виключає або» або «сума по модулю 2». Для позначення цієї операції використовується знак $\oplus$, тобто $f_7 = a \oplus b$. За довільного числа змінних ця функція дорівнює одиниці, якщо число одиниць в наборі вхідних змінних непарне. Іноді ця функція називається функцією перевірки на парність.

5. Функція $f_8 = a \vee b = a + b$, тобто це диз'юнкція a і b. Цю функцію можна записати у вигляді

$$y_\vee = \bigvee_{l=1}^{L} x_l$$

для довільного числа змінних.

6. Функція $f_9 = \overline{f_7}$ називається «запереченням диз'юнкції» або «АБО-НЕ». Цю функцію можна записати, використовуючи знаки $\vee$ і $\neg$ у вигляді $f_9 = \neg(a \vee b)$, або знаки + і $\neg$, тобто $f_9 = \neg(a + b)$. Крім того, (переважно) використовується запис $f_9 = \overline{a + b}$ або $f_9 = \overline{a \vee b}$. Іноді називається стрілкою Пірса і позначається як $f_9 = a \downarrow b$.

7. Функція $f_{10} = \overline{f_7}$ називається «рівнозначність» або «тотожність». Ця функція дорівнює одиниці за рівності всіх змінних вхідного набору.

8. Функції $f_{10} = \bar{b}$ і $f_{13} = \bar{a}$. Зауважимо, що для функції $f_{11}(f_{13})$ змінна $a(b)$ є несуттєвою.

9. Функція $f_{15} = \overline{f_2}$ називається «запереченням кон'юнкції» або функцією «І-НЕ». Ця функція записується в одній з таких форм: $\neg(a \wedge b)$, $\neg(a \,\&\, b)$, $\neg(a \cdot b)$, $\neg(ab)$, $\overline{a \wedge b}$, $\overline{a \,\&\, b}$, $\overline{ab}$. Іноді ця функція називається штрихом Шеффера і записується $f_9 = a'b$.

В англомовній нотації заперечення називається функцією NOT, кон'юнкція – AND, диз'юнкція – OR, заперечення кон'юнкції – NAND, заперечення диз'юнкції – NOR і нерівнозначності – EXOR.

10. Функція $f_{13} = \bar{a}$ і $f_{11} = \bar{b}$. Ці функції заперечують значення відповідних змінних.

Як видно з табл. 2.2, функції двох змінних охоплюють константи (0, 1), чотири функції однієї змінної (a, b, $\bar{a}$, $\bar{b}$) і десять функцій двох змінних. Аналогічно можна побудувати таблиці, що задають функції довільного числа булевих змінних. Однак в цьому немає необхідності, оскільки значення функцій, що залежать від будь-якого числа змінних, можна визначити, знаючи спосіб обчислення функцій з табл. 2.2.

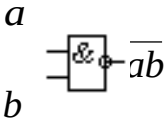
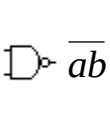
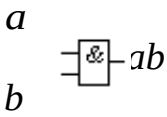
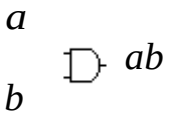
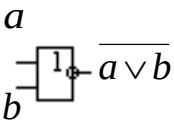
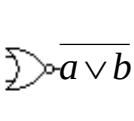
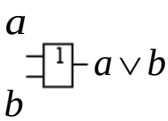
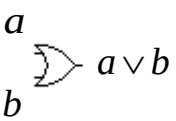
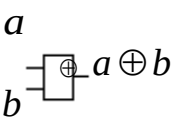
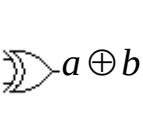
Наприклад, знайдемо значення функції $y_1 = x_1x_2x_3 \vee \overline{x_4x_5} \oplus x_3x_4$ на наборі $x_1 = x_2 = x_3 = 1$, $x_4 = x_5 = 0$. Використаємо для цього функції двох

змінних. Введемо такі позначення: $y_2 = x_1 x_2$, $y_3 = y_2 x_3$, $y_4 = \overline{x_4 x_5}$, $y_5 = y_3 \vee y_4$; $y_6 = x_3 x_4$, тоді $y_1 = y_5 \oplus y_6$. Використовуючи табл. 2.2, знайдемо $y_2 = 1,\ y_3 = 1,\ y_4 = 1,\ y_5 = 1,\ y_6 = 0,\ y_1 = 1$.

Схеми ЕОМ будуються з логічних елементів, які називаються вентилями. При цьому практичне значення мають лише вентилі, що відповідають повноваженням NOT, NAND і NOR. Це зумовлене тим, що вентилі будуються на основі транзисторів, які мають внутрішню інверсію. Отже, вентиль, відповідний функції NAND (NOR), має менше транзисторів і змінює свій стан в два рази швидше, ніж вентиль AND (OR). Вентилі, що відповідають повноваженням AND і OR, мають теоретичне значення, як і вентиль, відповідний функції EXOR. Позначення основних вентилів для європейського та американського стандартів показані в табл. 2.3.

Таблиця 2.3

Позначення основних вентилів

Функція	Європа	США	Функція	Європа	США
NOT			NAND		
AND			NOR		
OR			EXOR		

Далі в цій книзі ми будемо використовувати європейські позначення вентилів. Зазначимо, що вентилі можуть мати довільну кількість входів, здебільшого вона обмежена на практиці числом 8. Природно, вентиль NOT має тільки один вхід. При цьому вентиль NOT будемо називати «інвертор», вентиль AND – «кон'юнктор», вентиль OR – «диз'юнктор».

2.2 Аксіоми і закони булевої алгебри

Як і звичайна алгебра, булева алгебра містить низку фундаментальних правил, які приймаються без доказів і називаються аксіомами. Аксіоми є основою для доведення теорем, які є законами булевої алгебри. Аксіоми і закони використовуються для спрощення булевих функцій, що дає змогу побудувати схеми з визначеними характеристиками, зазвичай з мінімальним числом елементів.

Розглянемо аксіоми булевої алгебри, які іноді називають постулатами.

1. Булева змінна x завжди дорівнює або нулю, або одиниці:

$$x = 0, \quad якщо\ x = 1, \qquad (A_1)$$

$$x = 1, \quad якщо\ x \neq 0, \qquad (A_2)$$

тут A_i означає аксіоми, які пронумеровані для подальшого використання.

2. Інверсне значення змінної x протилежне її прямим значенням:

$$x = 0, \quad якщо\ \overline{x} = 1, \qquad (A_3)$$

$$x = 1, \quad якщо\ \overline{x} = 0. \qquad (A_4)$$

3. Логічне множення підпорядковується таким правилам:

$$0 \cdot 0 = 0, \qquad (A_5)$$

$$1 \cdot 1 = 1, \qquad (A_6)$$

$$0 \cdot 1 = 1 \cdot 0 = 0. \qquad (A_7)$$

4. Логічне складання підпорядковується таким правилам:

$$0 + 0 = 0, \qquad (A_8)$$

$$1 + 1 = 1, \qquad (A_9)$$

$$0 + 1 = 1 + 0 = 1. \qquad (A_{10})$$

Отримана система аксіом, що прямо випливає з визначення функцій НЕ, І та АБО, є основою для доведення теорем булевої алгебри. Оскільки нас цікавлять прикладні аспекти булевої алгебри, то ми розглянемо тільки

теореми для функцій НЕ, І та АБО, які утворюють логічний базис. Розглянемо основні теореми T_i і їхні докази.

1. Ідентичність.

$$x + 0 = x, \qquad (T_1)$$

$$x \cdot 1 = x. \qquad (T_2)$$

Доказ теореми T_1 випливає з аксіом A_8 і A_{10}, в яких перший доданок замінено на x. Згідно з теоремою T_1 з булевого виразу можна видалити частину, що дорівнює нулю і пов'язана з іншим виразом операцією диз'юнкція. Теорема T_2 доводиться на основі аксіом A_6 і A_7, і згідно з нею з булевого виразу можна видалити частину, що дорівнює одиниці і пов'язана з іншим виразом операцією кон'юнкція.

2. Наявність нульових елементів.

$$x + 1 = 1, \qquad (T_3)$$

$$x \cdot 0 = 0. \qquad (T_4)$$

Теорема T_3 випливає з аксіом A_9 і A_{10}, вказуючи на те, що якщо якась частина булевого виразу, пов'язана з іншим виразом операцією диз'юнкція, дорівнює одиниці, то і весь вираз дорівнює одиниці. Теорема T_4 випливає з аксіом A_5 і A_6, вказуючи на те, що якщо якась частина булевого виразу, пов'язана з іншим виразом знаком кон'юнкція, дорівнює нулю, то і весь вираз дорівнює нулю.

3. Ідемпотента.

$$x + x = x, \qquad (T_5)$$

$$x \cdot x = x. \qquad (T_6)$$

Теорема T_5 випливає з аксіом A_5 і A_9. Згідно з нею додавання або видалення ідентичних частин булевого виразу, пов'язаних операцією диз'юнкція з іншою частиною виразу, не змінює значення булевого виразу. Теорема T_5 може бути доведена по індукції в такому вигляді:

$$x + x + \ldots + x = n \cdot x = x.$$

Теорема T_6 випливає з аксіом A_5 і A_6. Вона вказує на те, що множення булевого виразу саме на себе не змінює значення булевого виразу. Теорема T_6 може бути розширена до $x \cdot x \cdots x = x^n = x.$

4. Еволюція (подвійне заперечення).

$$\bar{\bar{x}} = x. \qquad (T_7)$$

Теорема T_7 доводиться послідовним застосуванням аксіом A_3 і A_4 або з використанням методу повної індукції. Ідея методу полягає в побудові таблиці істинності для лівої і правої частин рівності, що доводиться. Якщо на всіх наборах вхідних змінних значення лівої і правої частин збігаються, то вони ідентичні і, отже, теорема доведена. Для доказу T_7 побудуємо табл. 2.4.

Таблиця 2.4

x	$\bar{x}$	$\bar{\bar{x}}$
1	0	1
0	1	0

Як видно з табл. 2.4, значення x і $\bar{\bar{x}}$ збігаються для всіх вхідних наборів. Отже, теорема T_7 справедлива.

З T_7 випливає, що подвійне заперечення булевого виразу не змінює його значення. Отже, будь-яке парне число заперечень булевого виразу може бути введено або видалено без зміни його значення.

5. Доповненості.

$$x + \bar{x} = 1, \qquad (T_8)$$

$$x \cdot \bar{x} = 0. \qquad (T_9)$$

Доведення цих теорем наведено в табл. 2.5.

Таблиця 2.5

x	$\bar{x}$	$x + \bar{x}$	$x \cdot \bar{x}$
0	1	1	0
1	0	1	0

Теореми T_8 і T_9 часто використовуються для доведення інших теорем, а також для спрощення булевих виразів.

6. Асоціативність.

$$(a+b)+c=a+(b+c), \qquad (T_{10})$$

$$(ab)c=a(bc). \qquad (T_{11})$$

Теореми T_{10} і T_{11} – повністю аналогічні теореми звичайної алгебри. Згідно з ними значення булевого виразу не залежить від порядку обчислення значень його частин.

7. Комутативність.

$$a+b=b+a, \qquad (T_{12})$$

$$a\cdot b=b\cdot a. \qquad (T_{13})$$

Згідно з теоремами T_{12} і T_{13} перестановка членів булевого виразу не змінює його значення. Ці теореми – аналогічні теореми звичайної алгебри.

8. Дистрибутивність.

$$ab+ac=a(b+c), \qquad (T_{14})$$

$$(a+b)(a+c)=a+bc. \qquad (T_{15})$$

Теорема T_{14} має свого аналога в звичайній алгебрі, а теорема T_{15} потребує доведення. Проведемо доказ шляхом перетворення лівої частини T_{15}. Якщо в результаті перетворень з лівої частини вийде права частина, то теорема доведена.

Доказ:

$$(a+b)(a+c)=a\cdot a+b\cdot a+a\cdot c+b\cdot c[T_{14}]=a+a\cdot b+a\cdot c+b\cdot c[T_6,T_{13}]=$$
$$=a\cdot 1+a\cdot b+a\cdot c+b\cdot c[T_2]=a(1+b+c)+bc[T_{14}]=a\cdot 1+b\cdot c[T_3]=a+b\cdot c[T_2]$$

Отже, $(a+b)(a+c)=a+bc$.

Після кожного перетворення виразів у квадратних дужках наведені теореми, на підставі яких проведені ці перетворення.

9. Поглинання.

$$a + ab = a, \qquad (T_{16})$$

$$a(a + b) = a. \qquad (T_{17})$$

Теореми T_{16} і T_{17} дають змогу зменшити число членів булевого виразу. Доведемо, наприклад, теорему T_{16}:

$$a + ab = a \cdot 1 + ab \cdot [T_2] = a(1 + b) \cdot [T_{14}] = a \cdot 1 \cdot [T_2] = a\,.$$

Для доведення теореми T_{17} зведемо її до T_{16}:

$$a(a + b) = a \cdot a + a \cdot b[T_{14}] = a + a \cdot b = a[T_{16}].$$

10. Склеювання.

$$ab + a\bar{b} = a, \qquad (T_{18})$$

$$(a + b)(a + \bar{b}) = a. \qquad (T_{19})$$

Теореми T_{18} і T_{19} також дають змогу зменшити число членів булевого виразу. Доведемо, наприклад, теорему T_{18}:

$$ab + a\bar{b} = a(b + \bar{b})[T_{14}] = a \cdot [T_8] = a[T_2]\,.$$

Тепер доведемо теорему T_{19}:

$$(a + b)(a + \bar{b}) = aa + a\bar{b} + ba + b\bar{b}\,[T_{14}\,] = a + a\bar{b} + ab + 0\,[T_6, T_{13}, T_9\,] =$$
$$= a + a\,[T_{18}, T_1\,] = a\,[T_5\,].$$

11. Теореми де Моргана.

$$\overline{a + b} = \bar{a} \cdot \bar{b}, \qquad (T_{20})$$

$$\overline{a \cdot b} = \bar{a} \vee \bar{b}. \qquad (T_{21})$$

Теореми де Моргана мають найважливіше значення при синтезі комбінаційних схем. Теорема T_{20} читається як «заперечення диз'юнкції еквівалентно кон'юнкції заперечень». Доведемо теорему T_{20}, для чого скористаємося такою лемою.

Лема. Якщо для булевих змінних A і B одночасно виконуються умови $AB = 0$ і $A \vee B = 1$, то $A = \bar{B}$.

Для доказу леми розглянемо табл. 2.6, з якої випливає, що умови $AB = 0$ і $A \vee B = 1$ виконуються одночасно для наборів $A = 0,\ B = 1$ і $A = 1,\ B = 0$. Отже, $A = \overline{B}$.

Таблиця 2.6

A	B	AB	$A \vee B$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Доказ теореми T_{20}. Позначимо $A = a + b$ і $B = \overline{a} \cdot \overline{b}$. Знайдемо значення виразів AB і $A \vee B$. Якщо ми отримаємо $AB = 0$ і $A \vee B = 1$, то з доведення леми буде випливати, що $A = \overline{B}$.

$AB = (a+b)(\overline{a}\overline{b}) = a\overline{a}\overline{b} + b\overline{a}\overline{b}\ [T_{14}] = 0 + 0\ [T_9] = 0\ [T_8]$.

$A \vee B = a + b + \overline{a}\overline{b} = a(b \vee \overline{b}) + b(a \vee \overline{a})\ [T_8, T_2] + \overline{a}\overline{b} = ab \vee a\overline{b} \vee \overline{a}b\ [T_{14}, T_{13}] \vee$

$\vee\ \overline{a}\overline{b} = ab \vee a\overline{b} \vee \overline{a}b \vee \overline{a}\overline{b}\ [T_5] = (ab \vee a\overline{b}) \vee (\overline{a}b \vee \overline{a}\overline{b})\ [T_{14}] = a \vee \overline{a}\ [T_{18}] = 1\ [T_8]$.

Отже, обидві умови леми справедливі, відтак, маємо $A = \overline{B}$ і $\overline{A} = B$, тобто $\overline{a + b} = \overline{a}\overline{b}$.

Теорема T_{21} доводиться аналогічно аналітичним шляхом, однак доведемо її методом повної індукції (табл. 2.7). З табл. 2.7 випливає, що значення в стовпцях $\overline{ab}$ і $\overline{a} \vee \overline{b}$ збігаються для всіх вхідних наборів, отже, теорема T_{21} справедлива. Теорема T_{21} читається у такий спосіб: «заперечення кон'юнкції еквівалентно диз'юнкції заперечень».

Таблиця 2.7

a	b	ab	$\overline{ab}$	$\overline{a}$	$\overline{b}$	$\overline{a} \vee \overline{b}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Теореми T_{20} і T_{21} справедливі для будь-якого числа вхідних змінних і можуть бути записані в такому вигляді:

$$\overline{\bigvee_{l=1}^{L} x_l} = \bigwedge_{l=1}^{L} \overline{x_l}, \tag{2.2}$$

$$\overline{\bigwedge_{l=1}^{L} x_l} = \bigvee_{l=1}^{L} \overline{x_l}. \tag{2.3}$$

У табл. 2.8 наведені основні теореми булевої алгебри, включно з їхніми україномовними та англомовними назвами.

Число теорем булевої алгебри незмірно більше, але для синтезу логічних схем цілком достатньо знання теорем T_1–T_{21}.

2.3 Методи представлення булевих функцій

Здебільшого на практиці використовуються табличне, кубічне й аналітичне подання булевих функцій. Розглянемо докладніше ці методи.

1. Табличне представлення. Будь-яка булева функція, що залежить від L змінних, може бути представлена таблицею істинності, що містить 2^L рядки і $L+1$ стовпець. Рядки таблиці містять двійкові вхідні набори, які відповідають десятковим цифрам від 0 до 2^L-1. В останньому стовпчику таблиці задається значення функції на кожному наборі.

Приклад 2.1. Задати у вигляді таблиці істинності функцію голосування щодо прийняття результату у важкій атлетиці. Змагання оцінюють головний суддя і два бокові арбітри. Результат зараховується, якщо не менше двох суддів вважають вагу взятою, причому один з них – головний суддя.

Таблиця 2.8

Основні теореми булевої алгебри

№	Теорема	Назва
1	2	3
T_1 T_2	$x+0=x,$ $x\cdot 1=x.$	Ідентичність Identities
T_3 T_4	$x+1=1,$ $x\cdot 0=0$	Наявність нульових елементів Null elements
T_5 T_6	$x+x=x,$ $x\cdot x=x$	Ідемпотентність Idempotency

Продовження табл. 2.8

1	2	3
T_7	$\bar{\bar{x}} = x$	Еволюція Involution
T_8 T_9	$x + \bar{x} = 1,$ $x \cdot \bar{x} = 0$	Доповнюваність Complements
T_{10} T_{11}	$(a+b)+c = a+(b+c),$ $(ab)c = a(bc)$	Асоціативність Associative
T_{12} T_{13}	$a+b = b+a,$ $a \cdot b = b \cdot a$	Комутативність Commutative
T_{14} T_{15}	$ab+ac = a(b+c)$ $(a+b)(a+c) = a+bc$	Дистрибутивність Distributive
$T_{16}\,T_{17}$	$a+ab = a,$ $a(a+b) = a$	Поглинання Absorption
$T_{18}\,T_{19}$	$ab+a\bar{b} = a$ $(a+b)(a+\bar{b}) = a$	Склеювання Expansion
T_{20} T_{21}	$\overline{a+b} = \bar{a} \cdot \bar{b},$ $\overline{a \cdot b} = \bar{a} \vee \bar{b}$	Теореми де Моргана De Morgan's theorems

Нехай y_1 – функція голосування, при цьому $y_1 = 1$, якщо вага узята. Позначимо результат голосування головного судді буквою a, а інших суддів – змінними b і c. Якщо результат зарахований, то змінна відповідного судді встановлюється в одиницю. Функція y_1 задана в табличній формі (табл. 2.9).

Якщо число одиниць в наборі не більше однієї, то вага не зараховується (набори 0–2, 4); якщо число одиниць дорівнює двом, але головний суддя проти, то вага не зараховується (набір 3); якщо число одиниць в наборі не менше двох і головний суддя за, то вага зараховується (набори 5–7).

Приклад 2.2. Задати таблицю істинності мажоритарної функції трьох змінних $y_2 = f(a,b,c)$.

Мажоритарна функція приймає значення, що дорівнює значенню більшості елементів набору. Таблиця істинності функції y_2 приведена в табл. 2.10.

Таблиця 2.9

a	b	c	y_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Таблиця 2.10

a	b	c	y_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Мажоритарні функції широко використовуються для прийняття рішень при резервуванні ресурсів. Наприклад, три комп'ютери управляють атомним реактором. Як управлінське рішення вибирається те, яке обирається більшістю. При цьому фіксується збій в роботі третього комп'ютера, і він підлягає ремонту. Бездефектна робота системи управління відповідає наборам 0 і 7.

Табличне представлення функцій є найбільш наочним і найбільш просто формується. Ця форма є первинною, а потім вона модифікується для введення інформації в ЕОМ або для оптимізації функцій і синтезу схеми. Однак табличне представлення є громіздким і не дає змоги компактно представити булеву функцію.

2. Кубічне уявлення. Кожному набору L булевих змінних може бути поставлена у відповідність точка L-мірного простору в системі L координат. Оскільки змінні є булевими, то і простір називається L-мірним булевим простором, або L-кубом.

На рис. 2.1 показано кубічне уявлення функцій L булевих змінних при $L = 0,1,2,3$. Домовимося, що якщо функція дорівнює одиниці на будь-якому наборі, то відповідна вершина куба обводиться знаком О.

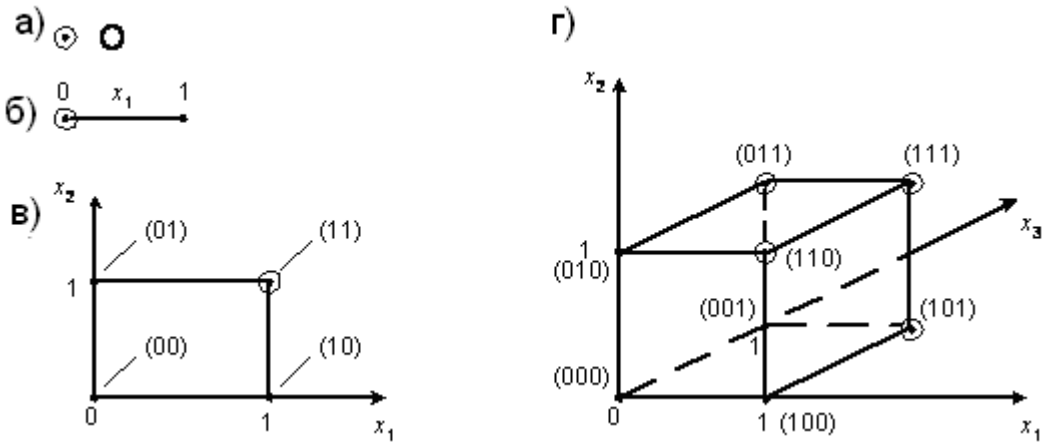


Рис. 2.1 – Кубічне уявлення булевих функцій

Рис. 2.1а відповідає випадку $L=0$, при цьому функції є константами. Зокрема, на рис. 2.1а задана константа 0. Відтак, булевим функціям при числі змінних $L=0$ відповідають 0-куби.

Рис. 2.1б задає функцію $y=\overline{x_1}$. Функції однієї змінної відповідає ребро або 1-куб.

Рис. 2.1в задає функцію $y=x_1x_2$. Функції двох змінних відповідає грань або 2-куб. У дужках біля кожної вершини записано значення вхідного набору, що відповідає цій вершині. Набори записуються в порядку зростання індексу булевої змінної. Для функції двох змінних набори відповідають парам (x_1x_2).

Рис. 2.1г задає мажоритарну функцію (табл. 2.10). Функції трьох змінних відповідає 3-куб, тобто куб, що має три виміри. З ростом числа вимірювань куби починають відрізнятися від звичних геометричних фігур. Як випливає з аналізу рис. 2.1г, 3-куб може бути представлений як два 2-куби для змінних x_1 і x_2. Однойменні вершини цих кубів з'єднані ребрами, протилежним кінцям яких відповідають протилежні значення змінної x_3.

Аналогічно куб функції L змінних може бути представлений як два куби функції $L-1$ змінних. Одному з цих кубів відповідає, наприклад, значення $x_L=0$, другому – $x_L=1$.

Кубічне уявлення є потужним засобом для введення інформації про булеві функції в ЕОМ для подальшої обробки. Наприклад, вершини 3-куба 101 і 111 пов’язані ребром (рис. 2.1г). Інформація про ці ребра представляється як 1∗1, де ∗ означає змінюваний елемент набору. Аналогічно вершини 010, 011, 110, 111 входять в одну грань 3-куба (рис. 2.1г) і інформація про цю межу може бути представлена як ∗1∗. Отже, якщо 2^n вершин є n-кубом, що входить в L-куб, то вони можуть бути представлені одним вектором, у якого n елементів замінені символом ∗. Це уявлення є більш компактним, ніж таблиця істинності.

Ще однією формою завдання булевих функцій є плоскі розгортки кубів, які називаються картами Карно. Карта Карно для функції L булевих змінних містить 2^L осередків, кожен з яких відповідає одному набору вхідних змінних. Карта Карно зберігає важливу властивість куба – набори для сусідніх клітин відрізняються тільки значенням однієї зі змінних. Отже, будь-які 2^n сусідні клітини карти Карно відповідають n-кубу ($n = 0,1,\ldots,L$).

На рис. 2.2 представлені карти Карно для функцій, які були представлені кубами на рис. 2.1.

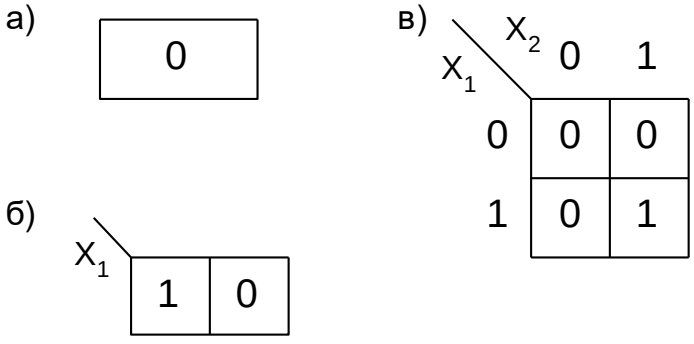


Рис. 2.2 – Подання булевих функцій картами Карно

Для побудови карти Карно для L-змінних скористаємося таким методом:

1. Намалювати 2 карти Карно для ($L-1$)-ої змінної.

2. Розташувати другу карту як дзеркальне відображення першої.

3. У першій карті для всіх клітин заповнити першу позицію нулями.

4. У другій карті для всіх клітин заповнити першу позицію одиницями.

5. З'єднати дві карти в одну.

Скористаємося цим правилом для побудови карти Карно для $L=3$, проілюструвавши процес рис. 2.3.

Після застосування першого і другого етапів маємо дві карти Карно для $L=2$ із симетричною розміткою (рис. 2.3а). Після виконання третього етапу маємо дві карти з різною розміткою (рис. 2.3б), з'єднання яких дає карту Карно для $L=3$. В осередках підсумкової карти записані набори $x_1x_2x_3$, які відповідають цим осередкам (рис. 2.3в).

Аналогічно будуються карти Карно для $L=4$ (показана на рис. 2.4а) і $L=5$ (показана на рис. 2.4б), а також карти Карно для будь-якого параметра L.

Карти Карно є більш компактною формою подання булевих функцій, ніж таблиці істинності. Однак більш важливим їхнім застосуванням є використання карт для ручної мінімізації функцій, про що йдеться далі. Методи ручної мінімізації є основою для мінімізації за допомогою ЕОМ.

Для оптимізації функцій необхідно вміти знаходити сусідні клітини карти Карно, а також виділяти n-куби функції L змінних, де $n=0,1,\ldots,L-1$.

З рис. 2.1б випливає, що за $L=2$ кожна вершина 2-куба має дві сусідні вершини. З рис. 2.1г випливає, що кожна вершина 3-куба має 3 сусідні вершини. Аналогічно кожна вершина L-куба має L сусідніх вершин. Отже, кожна клітина карти Карно для функції L змінних має L сусідніх клітин.

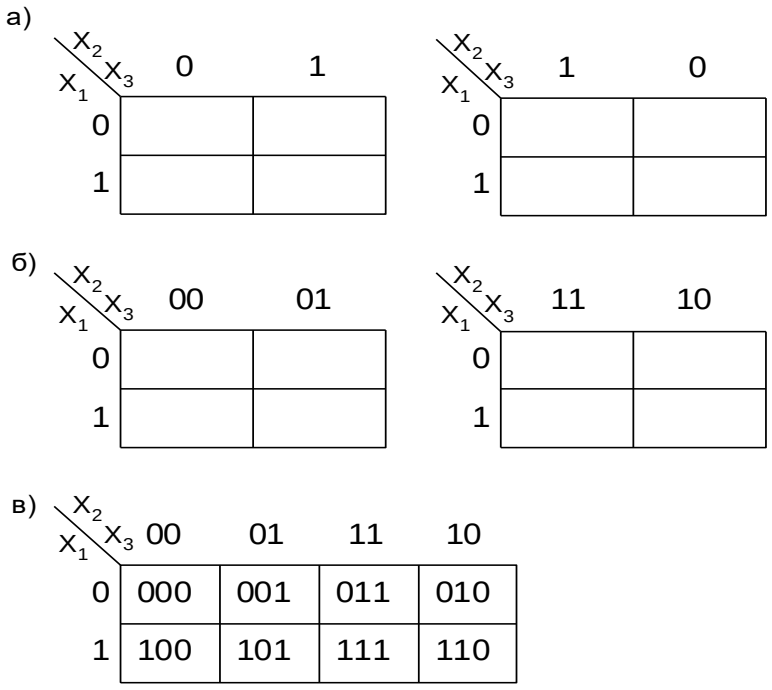


Рис. 2.3 – Формування карти Карно для $L = 3$

Знайдемо, наприклад, клітини, сусідні з клітиною 00010 (рис. 2.4б). Три клітини знаходяться безпосередньо поруч – це 00011, 00110 і 01010. Щодо осі a клітина 00010 є сусідньою з клітиною 00000, а щодо осі d – з клітиною 10010. Отже, всі п’ять сусідів знайдені.

Розглянемо клітини зі знаком «–» (рис. 2.4б). Це клітини 00000, 10000, 00100, 10100. Перша пара клітин відповідає ребру ∗0000 друга – ребру ∗0100. Порівняння цих ребер показує, що вони відрізняються по третій координаті, тобто їм відповідає грань ∗0∗00. Аналогічно клітинам зі знаком «+» (рис. 2.4б) відповідає грань куба ∗1∗00. Порівняння цих граней показує, що клітинам зі знаками «–» і «+» відповідає такий 3-куб: ∗∗∗00. Знаходження різних n-кубів усередині L-куба ($n = 0,1,\ldots,L-1$) вимагає деяких практичних навичок, отриманих шляхом тренування. Зауважимо, що ці навички необхідні для подальшої мінімізації булевих функцій, заснованої на використанні карт Карно. Очевидно, що пошук сусідніх клітин необхідний тільки в тому разі, якщо проводиться мінімізація функції.

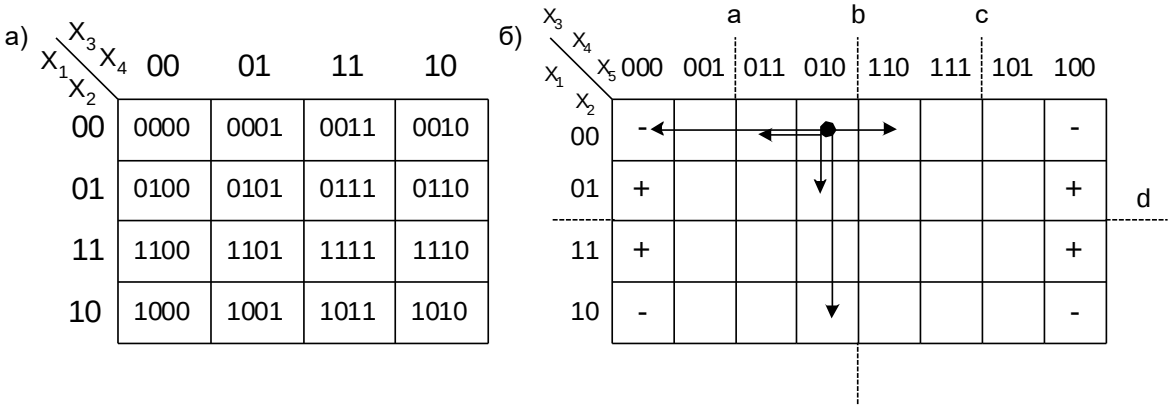


Рис. 2.4 – Карти Карно для $L=4$ *(А) і* $L=5$ *(Б)*

2.4 Аналітичне подання булевих функцій

Табличне і кубічне уявлення функцій є початковими формами, за якими формуються аналітичні вирази, які є основою для синтезу комбінаційної схеми. В рамках цієї книги ми розглянемо дві канонічні форми подання булевих функцій.

1. Диз'юнктивна нормальна форма булевої функції.

Диз'юнктивна нормальна форма (ДНФ) формується як диз'юнкція елементарних кон'юнкцій, які називаються мінтермами. Мінтермом F_h називається булева функція, яка приймає середнє арифметичне значення тільки на вхідному наборі з номером h, де $h=0,\ldots,2^L-1$, тут, як і скрізь, символ L означає число вхідних булевих змінних.

Розглянемо деяку функцію y_1 (табл. 2.11) і визначимо її мінтерм F_1, відповідний набору $x_1=0$, $x_2=0$, $x_3=1$. З визначення мінтермів випливає, що булева функція F_1 дорівнює одиниці, якщо дорівнює одиниці кон'юнкція НЕ (x_1) І НЕ (x_2) І x_3, отже маємо формулу $F_1=\overline{x_1}\,\overline{x_2}x_3$. Щоб перевірити наше припущення, побудуємо таблицю для визначення значення булевої функції F_1 на всіх можливих восьми наборах вхідних змінних (табл. 2.12). Ця таблиця має деяке розширення, порівняно з класичною таблицею істинності.

Таблиця 2.11

x_1	x_2	x_3	y_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Таблиця 2.12

x_1	x_2	x_3	$\bar{x}_1$	$\bar{x}_2$	x_3	F_1
0	0	0	1	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	0	0
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	0	0	0	0
1	1	1	0	0	1	0

Друга частина табл. 2.12 містить додаткові стовпці $\bar{x}_1$, $\bar{x}_2$, x_3, що відповідає набору 001. Як видно з аналізу цієї таблиці, набору 001 відповідає комбінація $\bar{x}_1 \bar{x}_2 x_3$, що дорівнює 111. Відповідно до визначення кон'юнкції, мінтерм F_1 дорівнює одиниці саме для цього випадку.

У загальному вигляді мінтерм може бути представлений у такий спосіб:

$$F_h = \bigwedge_{l=1}^{L} x_l^{E_{lh}}, \tag{2.4}$$

де $E_{lh} \in \{0,1\}$ – значення l-го розряду h-го набору вхідних змінних, $x_l^0 = \bar{x}_l$, $x_l^1 = x_l$ $(l = \overline{1,L})$.

Скориставшись формулою (2.4), визначимо мінтерми F_2, F_5, F_7, відповідні наборам, де функція y_1 (табл. 2.10) дорівнює одиниці. Це такі мінтерми: $F_2 = \bar{x}_1 x_2 \bar{x}_3$, $F_5 = x_1 \bar{x}_2 x_3$, $F_7 = x_1 x_2 x_3$.

Отже, для формування мінтермів F_h необхідно записати кон'юнкцію $x_1 x_2 \ldots x_L$ і поставити знак заперечення над змінними, що приймають нульове значення в h-му наборі.

Очевидно, що в кожен момент часу змінні x_l $(l = \overline{1,L})$ мають якесь стабільне значення, відповідне конкретному набору. Отже, в кожен момент часу тільки один з усіх мінтермів функції може дорівнювати одиниці. З

огляду на цю властивість побудуємо таблицю, що задає залежність функції y_1 від її мінтермів F_1, F_2, F_5, F_7 (табл. 2.13).

Таблиця 2.13

F_1	F_2	F_5	F_7	набір	y_1
0	0	0	0	0,3,4,6	0
1	0	0	0	1	1
0	1	0	0	2	1
0	0	1	0	5	1
0	0	0	1	7	1

Перший рядок таблиці відповідає ситуації $F_1 = F_2 = F_5 = F_7 = 0$. З аналізу табл. 2.11 очевидно, що ця ситуація відповідає одному з наборів 0, 2, 4 або 6. Отже, функція y_1 дорівнює нулю, якщо і тільки якщо виконується умова $F_1 = F_2 = F_5 = F_7 = 0$. Відтак, функція y_1 є диз'юнкцією мінтермів: $y_1 = F_1 \vee F_2 \vee F_5 \vee F_7$.

З попереднього випливає, що ДНФ довільної булевої функції формується у такий спосіб:

1. Виписуються мінтерми, відповідні наборам, на яких функція дорівнює одиниці.

2. Отримані мінтерми функції далі об'єднуються знаком диз'юнкції.

Наприклад, використання цього підходу для функції y_2 (табл. 2.14) приводить до такої ДНФ:

$$y_2 = F_2 \vee F_3 \vee F_6 = \overline{x_1} x_2 \overline{x_3} \vee \overline{x_1} x_2 x_3 \vee x_1 x_2 \overline{x_3}.$$

Таблиця 2.14

x_1	x_2	x_3	y_2
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Аналогічно ДНФ може бути побудована по карті Карно. Наприклад, для функції y_3 (рис. 2.5) маємо ДНФ:

$$y_3 = F_1 \vee F_3 \vee F_4 \vee F_6 = \overline{x_1}\,\overline{x_2}x_3 \vee \overline{x_1}x_2x_3 \vee x_1\overline{x_2}\,\overline{x_3} \vee x_1x_2\overline{x_3}.$$

Очевидно, що таблиці істинності, карти Карно і ДНФ є взаємозамінними формами подання булевих функцій. Ми розглянули, у який спосіб по карті Карно можна отримати ДНФ булевої функції. Тепер розглянемо зворотний процес.

x_1 \ x_2 x_3	00	01	11	10
0	0	1	1	0
1	1	0	0	1

Рис. 2.5 – Карта Карно для функції y_3

Нехай задана така ДНФ функції $y_4 = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2x_3$. Побудуємо її таблицю істинності і карту Карно. З аналізу ДНФ функції y_4 випливає, що $L = 3$, і функція дорівнює одиниці на наборах 1, 3, 7. Отже, в рядках таблиці істинності і клітинах карти Карно для цієї функції, що відповідають наборам 1, 3 і 7, повинні бути записані одиниці, а в інших – нулі (табл. 2.15 і рис. 2.6 відповідно).

Таблиця 2.15

x_1	x_2	x_3	y_4
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

x_1 \ x_2 x_3	00	01	11	10
0	0	1	1	0
1	0	0	1	0

Рис. 2.6 – Карта Карно для функції y_4

Рангом кон'юнкції назвемо число букв, що входять до неї. Якщо всі терми ДНФ функції L змінних мають ранг L, то це досконала ДНФ

(ДДНФ). Отже, ми розглянули методи формування ДДНФ за таблицями істинності і картами Карно.

Термін «мінтерм» означає «мінімальний терм», що зумовлено тим, що мінтерм відповідає 0-кубу, тобто кубу мінімального можливого розміру.

2. Кон'юнктивна нормальна форма булевої функції.

Кон'юнктивна нормальна форма (КНФ) булевої функції формується як кон'юнкція елементарних диз'юнкцій, що називаються макстермами.

Макстермом Φ_h називається булева функція, що дорівнює нулю тільки на наборі з номером h, де $h = 0,\ldots,2^L - 1$, а L – число вхідних змінних.

Розглянемо макстерм Φ_3 функції y_1 (табл. 2.11). Побудуємо таблицю істинності булевої функції Φ_3 (табл. 2.16).

Таблиця 2.16

x_1	x_2	x_3	x_1	$\bar{x}_2$	$\bar{x}_3$	Φ_3
0	0	0	0	1	1	1
0	0	1	0	1	0	1
0	1	0	0	0	1	1
0	1	1	0	0	0	0
1	0	0	1	1	1	1
1	0	1	1	1	0	1
1	1	0	1	0	1	1
1	1	1	1	0	0	1

З цієї таблиці видно, що функція Φ_3 є диз'юнкцією змінних x_1, $\bar{x}_2$ і $\bar{x}_3$:

$$\Phi_3 = x_1 \vee \overline{x_2} \vee \overline{x_3}.$$

Отже, для формування макстерма Φ_h необхідно записати диз'юнкцію вхідних змінних, причому, якщо змінна x_1 дорівнює 1 (0) на h-му наборі, то в макстерм Φ_h входить її інверсне (пряме) значення. Отже, маємо таку формулу:

$$\Phi_h = \bigvee_{l=1}^{L} x_l^{\overline{E}lh}. \qquad (2.5)$$

У формулі (2.5) всі компоненти мають таке саме значення, що і в формулі (2.4).

Очевидно, що в кожен момент часу може дорівнювати одиниці тільки один макстерм. Побудуємо таблицю, що задає залежність функції y_1 від її макстермів Φ_0, Φ_3, Φ_4, Φ_6, де функція дорівнює нулю (табл. 2.17).

Як випливає з аналізу цієї таблиці, функція y_1 дорівнює одиниці, якщо тільки всі її макстерми дорівнюють одиниці, тобто маємо формулу:

$$y_1 = \Phi_0\Phi_3\Phi_4\Phi_6 = (x_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x_2} \vee \overline{x_3})(\overline{x_1} \vee x_2 \vee x_3)(\overline{x_1} \vee \overline{x_2} \vee x_3).$$

Таблиця 2.17

Φ_0	Φ_3	Φ_4	Φ_6	y_1
1	1	1	1	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

Отже, для побудови КНФ довільної булевої функції може бути застосована така процедура:

1. Виписати макстерми, які відповідають вхідним наборам, на яких функція дорівнює нулю.

2. Отримані макстерми об’єднати знаком кон’юнкції.

Наприклад, для функції y_2 (табл. 2.14) маємо формулу:

$$y_2 = \Phi_0\Phi_1\Phi_4\Phi_5\Phi_7 = (x_1 \vee x_2 \vee x_3)(x_1 \vee x_2 \vee \overline{x_3})(\overline{x_1} \vee x_2 \vee x_3)(\overline{x_1} \vee x_2 \vee \overline{x_3}) \cdot$$
$$\cdot (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}).$$

Аналогічно КНФ може бути побудована по карті Карно. Наприклад, для функції y_3 (рис. 2.5) маємо:

$$y_3 = \Phi_0\Phi_2\Phi_5\Phi_7 = (x_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x_2} \vee x_3)(\overline{x_1} \vee x_2 \vee \overline{x_3})(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}).$$

Пояснимо походження терміну «макстерм». Розглянемо макстерм $\Phi_1 = x_1 \vee x_2 \vee \bar{x}_3$ і визначимо вершини 3-куба, які йому відповідають.

Змінна x_1 у формулі Φ_1 відповідає межі 1 **, а змінна x_2 – * 1 * і, нарешті, змінна $\bar{x}_3$ – ** 0. Грань 1 ** охоплює вхідні набори 100, 101, 110, 111, грань * 1 * – 010, 011, 110, 111, грань ** 0 – 000, 010, 100, 110. Отже, макстерм Φ_1 відповідає всім наборам при числі змінних $L=3$ (крім набору 001). Отже, макстерм покриває 2^L-1 набір, тобто максимально можливе число наборів, які не є в сукупності L-кубом.

За аналогією до ДДНФ, КНФ називається досконалою, якщо всі її макстерми мають ранг L, де L – число змінних, від яких залежить вихідна булева функція.

Усі розглянуті форми подання булевих функцій є взаємно однозначними, тобто за будь-якою з цих форм можуть бути отримані всі інші. Зауважимо, що для переходу ДДНФ→ДКНФ і навпаки необхідно перейти до однієї з інших форм представлення. Цей факт відображений на рис. 2.7, де двонаправлені ребра відображають факт еквівалентності (тобто, одна форма подання може бути замінена іншою) різних форм представлення однієї і тієї самої функції.

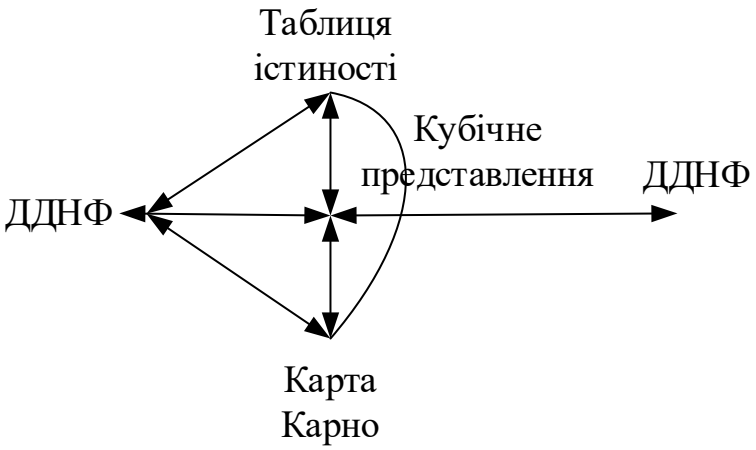


Рис. 2.7 – Взаємозв'язок між формами подання булевої функції

При синтезі схем вузлів ЕОМ доцільно використовувати обмежений набір логічних елементів. Чим менше елементів в наборі, тим простіше налагодити виробництво надійних схем. До цього набору висувається така вимога – будь-яка як завгодно складна логічна схема може бути

побудована з елементів цього набору. Технічне завдання відшукання такого набору логічних елементів зводиться до математичної задачі визначення функціонально повного набору булевих функцій.

Система булевих функцій називається функціонально повною, якщо будь-яка булева функція може бути виражена суперпозицією цих функцій. З матеріалу попередніх розділів зрозуміло, що система функцій {І, АБО, НЕ} буде функціонально повною, оскільки вона дає змогу виразити ДНФ і КНФ довільної булевої функції.

Очевидно, що якщо будь-яку функцію довільної функціональної системи Ψ_1 можна виразити через функції деякої системи Ψ_2, то система Ψ_2 так само є функціонально повною.

Знайдемо деякі функціонально повні системи булевих функцій, що мають практичне значення. Уявімо базис {І, АБО, НЕ} у вигляді $\{\bar{a}, a \wedge b, a \vee b\}$. Тоді справедливим є таке:

1. Система $\{\bar{a}, a \vee b\}$ функціонально повна, оскільки $a \wedge b = \overline{\bar{a} \vee \bar{b}}$. Це означає, що система {НЕ, АБО} дає змогу виразити всі функції, які входять у функціонально повну систему {І, АБО, НЕ}.

2. Система $\{\bar{a}, a \wedge b\}$ функціонально повна, оскільки $a \vee b = \overline{\bar{a} \wedge \bar{b}}$. Це означає, що система {НЕ, І} дає змогу виразити всі функції, які входять у функціонально повну систему {І, АБО, НЕ}.

3. Система $\{\overline{a \wedge b}\}$ функціонально повна. Для доказу розглянемо рівняння: $\bar{a} = \overline{a \cdot a}$, $a \wedge b = \overline{\overline{a \wedge b}}$. Отже, функція Шеффера визначає функціонально повну систему.

4. Система $\{\overline{a \vee b}\}$ функціонально повна, оскільки $\bar{\bar{a}} = a$ і $a \vee b = \overline{\overline{a \vee b}}$. Це означає, що функція Пірса дає змогу отримати будь-яку функцію з функціонально повної системи $\{\bar{a}, a \vee b\}$.

5. Система функцій $\{1, a \wedge b, a \oplus b\}$ є функціонально повною, оскільки $\bar{a} = a \oplus 1$. Тобто ця система дає змогу виразити будь-яку функцію

базису $\{\bar{a}, a \wedge b\}$. З повнотою цього базису пов'язана алгебра Жегалкіна. У цій алгебрі будь-яка булева функція представляється поліномом Жегалкіна, тобто сумою по модулю двох представлень неінвертованих змінних.

У нашій книзі, переважно, використовується базис І-НЕ ($\overline{a \wedge b}$) або АБО-НЕ ($\overline{a \vee b}$), тобто базис, заснований на функціях Шеффера і Пірса. Зауважимо, що перший з цих базисів набув найбільшого використання у процесі проєктування практичних вузлів ЕОМ.

Список літератури до розділу 2

1. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

2. Закревский А. Д., Поттосин Ю. В., Черемисинова Л. Д. Основы логического проектирования. В 3-х кн. Кн. 1. Комбинаторные алгоритмы дискретной математики. Минск: ОИПИ НАН Беларуси, 2004. 226 с.

3. Прикладная теория цифровых автоматов / К. Г. Самофалов, А. М. Романкевич, В. Н. Валуйский, Ю. С. Каневский, М. М. Пиневич. Киев: Вища шк., 1987. 375 с.

4. Савельев А. Я. Прикладная теория цифровых автоматов. Москва: Высш. шк., 1987. 272 с.

3 МІНІМІЗАЦІЯ БУЛЕВИХ ФУНКЦІЙ

3.1 Принципи спрощення ДНФ

Будь-якому аналітичному виразу булевої функції відповідає логічна схема, побудована у вигляді мережі з елементів НЕ, І та АБО. Ця схема є ще однією формою завдання булевої функції.

Наприклад, логічна схема, показана на рис. 3.1, відповідає булевій функції $f_1 = abc \vee ab\bar{c} \vee ade \vee a\bar{d}e = F_1 \vee F_2 \vee F_3 \vee F_4$.

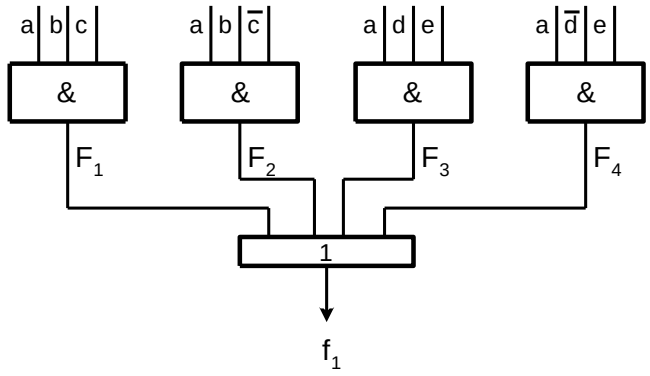


Рис. 3.1 – Реалізація функції f_1 на логічних елементах

У цій схемі кожному терму F_h $(h = \overline{1,4})$ функції відповідає логічний елемент І з числом входів L_h, що дорівнює числу букв (рангу) в термі F_h. Виходи схем І подаються на вхід елемента АБО.

Одна і та сама функція може бути представлена різними аналітичними виразами, яким відповідають різні схеми. Ці схеми можуть мати різну кількість елементів і з'єднань між ними, що визначає їхню різну вартість. Для оцінки складності реалізації j-го аналітичного виразу функції f_i на практиці використовують ціну схеми за Квайном $C(f_{ij})$. Ціна за Квайном $C(f_{ij})$ дорівнює сумарному числу входів всіх елементів логічної схеми, що реалізує функцію f_i.

Наприклад, для схеми на рис. 3.1 маємо $C(f_1) = 16$. Цю оцінку можна визначити і без схеми, маючи тільки аналітичну форму

представлення функції. Якщо аналітичний вираз функції f має H термів $F_1,\ldots,F_H$ і h-ий терм F_h має L_h букв, то отримуємо таку оцінку:

$$C(f_1) = \sum_{h=1}^{H} L_h + H. \qquad (3.1)$$

Аналіз функції f_1 показує, що до термів F_1, F_2 і F_3, F_4 може бути застосований закон склеювання, що дає $f_{1,1} = ab \vee ae$. Згідно з (3.1) логічна схема для функції $f_{1,1}$ повинна мати ціну за Квайном $C(f_{1,1}) = 6$. Це видно і зі схеми на рис. 3.2а.

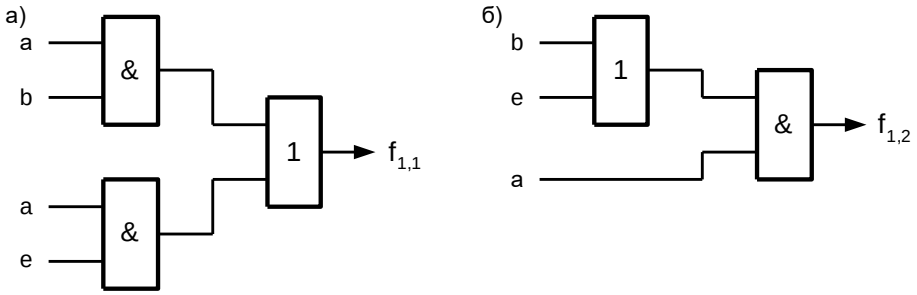


Рис. 3.2 – Реалізація функції f_1 після застосування законів склеювання (а) і асоціативності (б)

Застосування асоціативного закону до булевої функції $f_{1,1}$ приводить до виразу $f_{1,2} = a(b \vee e)$, що породжує схему (рис. 3.2б) з $C(f_{1,2}) = 4$.

Отже, застосування законів булевої алгебри дає змогу зменшити початкову вартість схеми в чотири рази. Очевидно, чим менша ціна за Квайном, тим меншою буде і вартість схеми в грошовому обчисленні. Отже, правильне застосування законів булевої алгебри дає змогу зменшити вартість схем, якщо це можливо. Отже, одна і та сама булева функція може бути представлена різними ДНФ. Ці форми називаються еквівалентними ДНФ. Це приводить до задачі знаходження оптимальної в якомусь сенсі ДНФ серед множини еквівалентних ДНФ. Це завдання називається завданням мінімізації булевої функції (в класі ДНФ) і має складний комбінаторний характер.

Якщо критерієм оптимальності є мінімальна кількість букв (літералів) в кінцевій формулі, то йдеться про знаходження мінімальної ДНФ. Якщо мінімізується число кон'юнкції у формулі, то говорять про пошук найкоротшої ДНФ. Іноді на практиці достатньо знайти тільки так звану ненадлишкову ДНФ, тобто формулу, з якої не можна видалити жодного терма (кон'юнкції) і жодного літерала.

Найпростіші методи спрощення ДНФ мають локальний характер, тобто їхнє застосування дає змогу спростити тільки окремі частини ДНФ. Зокрема, при розгляді пар термів використовуються три операції:

1. Поглинання, яке визначається формулою

$$a \vee ab = a. \tag{3.2}$$

Вираз (3.2) відбиває той факт, що терм ab поглинається змінною a. Формула (3.2) доводиться у такий спосіб: $a \vee ab = a(1 \vee b) = a \cdot 1 = a$. Цю формулу можна поширити на випадок довільних кон'юнкцій F_1 і F_2:

$$F_1 \vee F_1 F_2 = F_1. \tag{3.3}$$

У виразі (3.3) терми F_1 і F_2 можуть складатися з довільного числа змінних. Нехай, наприклад, $F_1 = ab$, а $F_2 = cd$, тоді з (3.3) маємо:

$$ab \vee abcd = ab.$$

2. Склеювання, яке визначається формулою

$$ab \vee \bar{a}b = b. \tag{3.4}$$

При цьому говорять, що терми $F_1 = ab$ і $F_2 = \bar{a}b$ склеюються по булевій змінній a. Формула (3.4) доводиться у такий спосіб: $ab \vee \bar{a}b = b(a \vee \bar{a}) = b \cdot 1 = b$. Ця формула також може бути поширена на довільні терми F_1 і F_2:

$$F_1 F_2 \vee \bar{F}_1 F_2 = F_2. \tag{3.5}$$

Наприклад, нехай $F_1 = ab$, $F_2 = cd$, тоді $abcd \vee \overline{ab}cd = cd$.

3. Видалення літерала, яке визначається формулою

$$a \vee \bar{a}b = a \vee b. \tag{3.6}$$

Формулу (3.6) можна довести у такий спосіб:

$$a \vee \bar{a}b = a(b \vee \bar{b}) \vee \bar{a}b = ab \vee a\bar{b} \vee \bar{a}b = (ab \vee a\bar{b}) \vee (ab \vee \bar{a}b) = a \vee b.$$

Ця формула може бути поширена на випадок будь-яких термів F_1 і F_2:

$$F_1 \vee \bar{F}_1 F_2 = F_1 \vee F_2. \qquad (3.7)$$

Для попереднього прикладу маємо: $ab \vee \overline{ab}cd = ab \vee cd$.

При розгляді трьох термів може застосовуватися операція узагальненого склеювання:

$$F_1F_2 \vee \bar{F}_1F_3 \vee F_2F_3 = F_1F_2 \vee \bar{F}_1F_3. \qquad (3.8)$$

Формула (3.8) може бути доведена у такий спосіб:

$$F_1F_2 \vee \bar{F}_1F_3 \vee F_2F_3 = F_1F_2 \vee \bar{F}_1F_3 \vee F_2F_3(F_1 \vee \bar{F}_1) = F_1F_2 \vee \bar{F}_1F_3 \vee F_1F_2F_3 \vee$$
$$\vee \bar{F}_1F_2F_3 = (F_1F_2 \vee F_1F_2F_3) \vee (\bar{F}_1F_3 \vee \bar{F}_1F_2F_3) = F_1F_2(1 \vee F_3) \vee \bar{F}_1F_3(1 \vee F_2) =$$
$$= F_1F_2 \cdot 1 \vee \bar{F}_1F_3 \cdot 1 = F_1F_2 \vee \bar{F}_1F_3.$$

Для спрощення ДНФ часто застосовують такі два прийоми:

1. Дублювання умов (елементарна кон'юнкція) функції, тобто використання закону $a = a \vee a$.

2. Диз'юнктивне розкладання термів функції щодо відсутніх змінних. Наприклад, терм ab може бути розкладений по змінній c, що дає $abc \vee ab\bar{c}$. Як видно, ця операція є зворотною для операції склеювання. Можна розкласти терм ab по змінним c і d, що приведе до виразу $ab\bar{c}\bar{d} \vee ab\bar{c}d \vee abc\bar{d} \vee abcd$. Це рівносильно множенню ab на вираз $(c \vee \bar{c})(d \vee \bar{d}) = 1 \cdot 1 = 1$.

Обидва прийоми використовуються для подальшого склеювання отриманих термів з деякими з уже існуючих термів ДНФ.

3.2 Метод карт Карно

Природно, що безпосереднє застосування законів булевої алгебри для спрощення функцій є досить складним процесом. Відтак на практиці використовуються методи, засновані на застосуванні карт Карно або деяких формальних методик.

Мета спрощення булевих функцій – отримання схеми з мінімальною ціною за Квайном, що відповідає аналітичному виразу функції з мінімальним числом букв. Назвемо ДНФ з мінімальним числом букв мінімальною ДНФ (МДНФ), а процес побудови МДНФ по ДДНФ – мінімізацією булевої функції.

Розглянемо функцію f_2, задану картою Карно, на рис. 3.3.

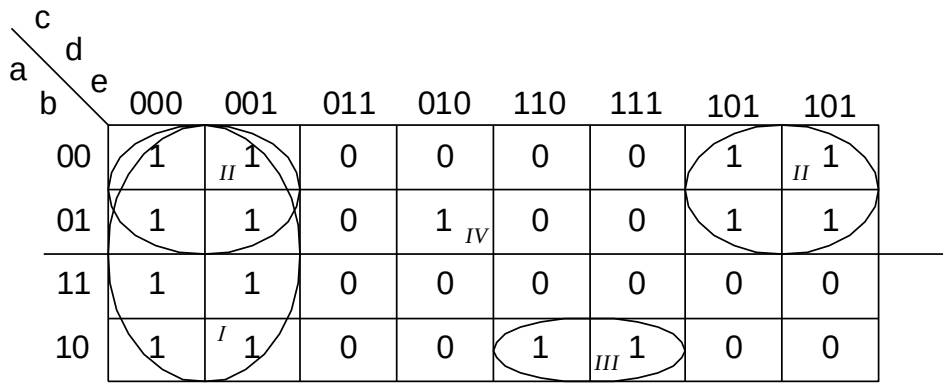


Рис. 3.3 – Карта Карно для функції f_2

Знайдемо підфункцію $\varphi_1 = \bar{a}\bar{b}c\bar{d}e \vee \bar{a}\bar{b}c\bar{d}\bar{e} = F_5 \vee F_4 = \bar{a}\bar{b}c\bar{d}$. Отриманий результат відповідає ребру $0010*$, що з'єднує вершини куба 00101 (терм F_5) і 00100 (терм F_4). Тепер знайдемо підфункцію $\varphi_2 = F_5 \vee F_4 \vee F_{13} \vee F_{12} = \bar{a}\bar{b}c\bar{d} \vee \bar{a}bc\bar{d}e \vee \bar{a}bc\bar{d}\bar{e} = \bar{a}c\bar{d}$. Отриманий результат відповідає межі 0 * 10 *, утвореній вершинами 00101 (F_5), 01101 (F_{13}), 00100 (F_4) і 01100 (F_{12}). Далі знайдемо наступну булеву підфункцію $\varphi_3 = \varphi_2 \vee F_0 \vee F_1 \vee F_8 \vee F_9 = \varphi_2 \vee \bar{a}\bar{b}\bar{c}\bar{d}\bar{e} \vee \bar{a}\bar{b}\bar{c}de \vee \bar{a}b\bar{c}\bar{d}\bar{e} \vee \bar{a}bc\bar{d}e = \varphi_2 \vee \bar{a}\bar{b}\bar{c}\bar{d} \vee$ $\vee \bar{a}b\bar{c}\bar{d} = \bar{a}c\bar{d} \vee \bar{a}\bar{c}\bar{d} = \bar{a}\bar{d}$. Отриманий результат відповідає 3-кубу, який об'єднує вершини 0, 1, 4, 5, 8, 9, 12 і 13.

З розглянутих прикладів можна зробити висновок про те, що будь-яким сусіднім 2^n клітинам довільної карти Карно відповідає n-куб, який породжує підфункцію з $L-n$ буквами. Отримана підфункція є термом МДНФ. Назвемо n-куб булевої функції $(n = 0,1,\ldots,L-1)$ покриттям. Завдання мінімізації булевої функції зводиться до знаходження n-кубів, що покривають в сукупності всі клітини вихідної карти Карно, для яких

булева функція дорівнює одиниці. Природно, що кожне з таких покриттів має включати максимальне число клітин карти Карно.

Процес мінімізації по карті Карно може бути представлений у такий спосіб:

1. Знайти всі пари клітин, що містять одиниці, і відповідні 1-куби. Якщо деякі клітини не можна об'єднати з іншими, то відповідні їм терми з L буквами увійдуть в мінімальну ДНФ.

2. Знайти всі четвірки, одержані як комбінації пар першого етапу, і відповідні 2-куби. Якщо деякі пари не можна об'єднати з іншими, то відповідні їм терми із $L-1$ буквами увійдуть в мінімальну ДНФ.

3. Процес триває доти, доки на черговому етапі ніяка пара з $(n-1)$-куб не об'єднається в n-куб $(n=0,1,\ldots,L-1)$. Кожен з отриманих кубів відповідає покриттю, що входить в МДНФ.

Розглянемо процес мінімізації функції f_2, зазначивши отримані покриття замкнутими лініями з римськими номерами. Результуюча МДНФ матиме вигляд $f_2 = F_I \vee F_{II} \vee F_{III} \vee F_{IV}$, де числа в римській системі числення відповідають номерам покриттів на рис. 3.3.

З розглянутого нами методу отримання булевих функцій $\varphi_1 - \varphi_3$ випливає, що для знаходження будь-якого терма, відповідного покриття необхідно виконати аналіз:

1. Якщо l-а компонента наборів, що входять в покриття, змінюється, то відповідна їй буква не входить в результуючий терм $(l=1,\ldots,L)$.

2. Якщо l-а буква входить в результуючий терм, то за рівності одиниці l-ої компоненти l-а буква входить в терм без інверсії.

Отже, результуючий терм F_c, який відповідає покриттю, визначається формулою

$$F_c = \bigwedge_{l=L}^{L} x_l^{E_{lc}}, \qquad (3.9)$$

де $E_{lc} \in \{0,1,*\}$, 0 (1) – значення l-ої компоненти для всіх наборів покриття, що дорівнює 0 (1), * – значення l-ої компоненти в різних наборах покриття, що змінюється: $x_l^1 = x_l$, $x_l^0 = \bar{x}_l$, $x_l^* = 1$.

Використовуючи формулу (3.9), отримаємо такі підсумкові терми, які входять в ДНФ: $F_I = a^* b^* c^0 d^0 a^* = \bar{c}\bar{d}$; $F_{II} = a^0 b^* c^* d^0 e^* = \bar{a}\bar{d}$; $F_{III} = a^1 b^0 c^1 d^1 e^* = a\bar{b}cd$; $F_{IV} = a^0 b^1 c^0 d^1 e^0 = \bar{a}b\bar{c}d\bar{e}$. Отже, МДНФ функції f_2 має вигляд: $f_2 = \bar{c}\bar{d} \vee \bar{a}\bar{d} \vee a\bar{b}cd \vee \bar{a}b\bar{c}d\bar{e}$. Ціна за Квайном $C(f_2) = 17$, а ціна за Квайном без мінімізації, згідно з (3.1), дорівнює $15 \times 5 + 15 = 90$. Відтак, мінімізація функції f_2 зменшує ціну за Квайном у понад 5 разів, порівняно зі схемою, яка реалізується за ДДНФ.

Для ефективної мінімізації булевої функції за картами Карно студентам необхідно набути деяких практичних навичок, після чого мінімізація буде виконуватися автоматично.

Аналогічно формується і мінімальна КНФ (МКНФ). Відмінність від попереднього прикладу полягає в тому, що n-куби формуються з сусідніх клітин карти, що містять нулі. Мінімізація тепер заснована на використанні закону склеювання

$$(A \vee b) \cdot (A \vee \bar{b}) = A,$$

де A – довільна булева функція. В цьому разі результуючий терм Φ_c визначається виразом

$$\Phi_c = \bigvee_{l=1}^{L} x_l^{E_{cl}}, \qquad (3.10)$$

де $E_{cl} \in \{0,1,*\}$, $x_l^0 = x_l$, $x_l^1 = \bar{x}_l$, $x_l^* = 0$.

Розглянемо карту Карно (рис. 3.4) і знайдемо мінімальну КНФ функції f_3. Очевидно, що в разі КНФ ми повинні шукати покриття, що складаються з клітин карти, в яких записані нулі. Принцип пошуку таких покриттів не відрізняється від принципу, розглянутого раніше для ДНФ.

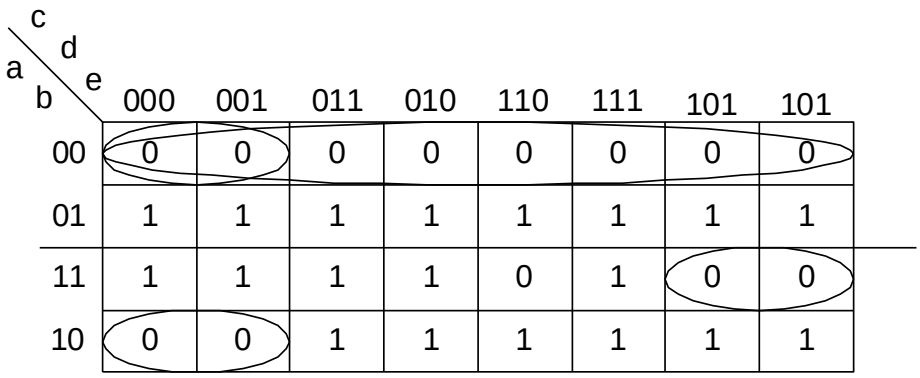


Рис. 3.4 – Карта Карно для функції f_3

Після знаходження покриттів I–IV очевидно, що результуюча МКНФ $f_3 = \Phi_I \cdot \Phi_{II} \cdot \Phi_{III} \cdot \Phi_{IV}$. Використовуючи формулу (3.10), отримаємо: $\Phi_I = a^0 \vee b^0 \vee c^* \vee d^* \vee e^* = a \vee b$; $\Phi_{II} = a^* \vee b^0 \vee c^0 \vee d^0 \vee e^* = b \vee c \vee d$; $\Phi_{III} = a^1 \vee b^1 \vee c^1 \vee d^0 \vee e^* = \bar{a} \vee \bar{b} \vee \bar{c} \vee d$; $\Phi_{IV} = a^1 \vee b^1 \vee c^1 \vee d^1 \vee e^0 = \bar{a} \vee \vee \bar{b} \vee \bar{c} \vee \bar{d} \vee e$. Отже, $f_3 = (a \vee b)(b \vee c \vee d)(\bar{a} \vee \bar{b} \vee \bar{c} \vee d)(\bar{a} \vee \bar{b} \vee \bar{c} \vee \bar{d} \vee e)$.

Розглянуті раніше функції належать до визначених функцій булевої алгебри, значення яких визначені на всіх можливих наборах змінних. Однак через фізичні особливості роботи пристрою частина наборів може бути заборонена. Фізично це означає, що подібне поєднання параметрів є неможливим. Якщо функція $f = f(x_1, \ldots, x_L)$ визначена не на всіх 2^L можливих наборах вхідних змінних, то вона називається неповністю визначеною або частковою булевою функцією. Наявність заборонених наборів призводить до особливостей в отриманні МДНФ і МКНФ часткових булевих функцій.

Розглянемо деякі приклади, що ілюструють появу невизначених (заборонених) наборів.

Приклад 3.1. У приміщенні необхідно підтримувати рівень температури в межах $15^\circ C \le t^\circ \le 25^\circ C$. Якщо $t^\circ < 15^\circ C$, то необхідно ввімкнути обігрівач для нагріву приміщення. Якщо $t^\circ > 15^\circ C$, то вмикається кондиціонер. Необхідно поставити систему булевих функцій для управління цим процесом.

Система стабілізації температури може бути представлена схемою на рис. 3.5.

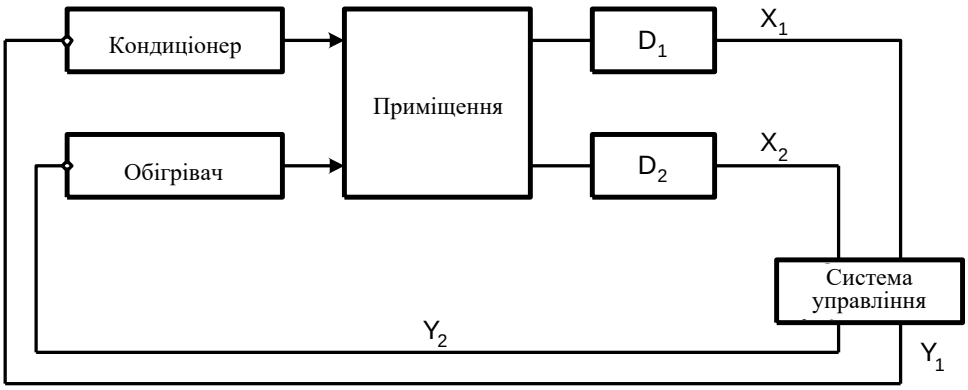


Рис. 3.5 – Структурна схема системи стабілізації рівня температури

Ця система включає, крім кондиціонера й обігрівача, два датчики D_1 і D_2, а також систему управління. Домовимося, що $x_1 = 1$, якщо $t^\circ > 25^\circ C$, і $x_1 = 0$, якщо $t^\circ \le 25^\circ C$; $x_2 = 1$, якщо $t^\circ < 15^\circ C$, і $x_2 = 0$, якщо $t^\circ \ge 15^\circ C$. Система управління аналізує показники датчиків і формує два керуючі сигнали: $y_1 = 1$ – ввімкнути кондиціонер, $y_1 = 0$ – вимкнути кондиціонер, $y_2 = 1$ – ввімкнути обігрівач, $y_2 = 0$ – вимкнути обігрівач. Фізичний сенс сигналів x_1, x_2, y_1 і y_2 зображений на рис. 3.6.

Закон функціонування системи управління задається таблицею істинності (табл. 3.1). Тут набір $x_1 = x_2 = 1$ вказує на ситуацію, коли температура одночасно більше 250 °C і менше 150 °C. Очевидно, що одночасно це неможливо, і така ситуація вказує на несправність одного з датчиків D_1, D_2 (або обох відразу). З фізичного боку ця ситуація неможлива, тому набір 11 є забороненим. Отже, значення функцій на цьому наборі є невизначеними, що позначається знаком * у шпальтах y_1 і y_2.

Приклад 3.2. Задати таблицю істинності функції y_3, де $y_3 = 1$, якщо і тільки якщо $A \ge B$. При цьому виконується умова $A, B \in \{0,1,2\}$.

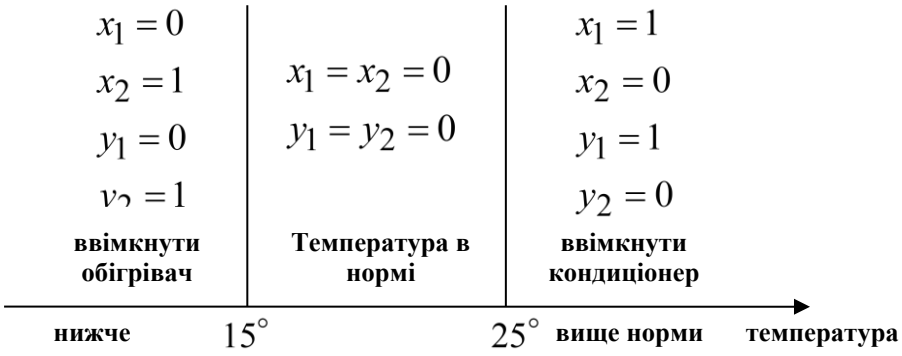


Рис. 3.6 – Фізичний сенс сигналів датчиків і системи управління

Таблиця 3.1

x_1	x_2	y_1	y_2
0	0	0	0
0	1	0	1
1	0	1	0
1	1	*	*

Очевидно, для представлення десяткових чисел A і B необхідно по два двійкові розряди a_1a_2 і b_1b_2 відповідно. Нехай 00 відповідає нулю, 01 – одиниці, 10 – двійці. Тоді таблиця істинності (табл. 3.2) буде містити заборонені набори при $a_1a_2 = 11$ і / або $b_1b_2 = 11$.

Таблиця 3.2

a_1	a_2	A	b_1	b_2	B	y
0	0	0	0	0	0	1
0	0	0	0	1	1	0
0	0	0	1	0	2	0
0	0	0	1	1	?	*
0	1	1	0	0	0	1
0	1	1	0	1	1	1
0	1	1	1	0	2	0
0	1	1	1	1	?	*
1	0	2	0	0	0	1
1	0	2	0	1	1	1
1	0	2	1	0	2	1
1	0	2	1	1	?	*
1	1	?	0	0	0	*
1	1	?	0	1	1	*
1	1	?	1	0	2	*
1	1	?	1	1	?	*

У табл. 3.2 стовпці A і B використані як коментарі, а також для спрощення порівняння A і B.

На рис. 3.7 приведена карта Карно, відповідна табл. 3.2.

a_1a_2 \ b_1b_2	00	01	11	10
00	1	0	*	0
01	1	1	*	0
11	*	*	*	*
10	1	1	*	1

Рис. 3.7 – Карта Карно функції y_3

При синтезі логічної схеми знаки $*$ повинні бути замінені нулями або одиницями. Очевидно, кожен знак $*$ може бути замінений на 0 або 1. Отже, якщо функція y має n невизначених наборів, то їй відповідає 2^n повністю визначених функцій. При цьому кожна заміна призводить до різної мінімальної ДНФ або КНФ.

Розглянемо низку прикладів. Нехай всі знаки $*$ в карті Карно (рис. 3.7) замінені нулями, що буде відповідати функції y_3^1 (рис. 3.8а). З цієї карти Карно маємо МДНФ функції y_3^1 з ціною за Квайном $C(y_3^1)=16$:

$$y_3^1=\bar{a}_1\bar{b}_1\bar{b}_2\vee\bar{a}_1a_2\bar{b}_1\vee a_1\bar{a}_2\bar{b}_1\vee a_1\bar{a}_2\bar{b}_2.$$

Тепер замінимо всі знаки $*$ одиницями (рис. 3.8б) і отримаємо МДНФ

$$y_3^2=a_1\vee b_1b_2\vee a_2\bar{b}_1\vee\bar{b}_1b_2$$

з $C(y_3^2)=10$. Зауважимо, що функція y_3^2 включає терм $F_{II}=b_1b_2$, який відповідає області невизначених наборів в карті Карно (рис. 3.7).

З порівняння МДНФ y_3^1 і y_3^2 випливає, що заміна $*\to 1$ і $*\to 0$ повинна виконуватися у такий спосіб, щоб в результуючій формулі покриття охоплювали максимально можливе число сусідніх клітин, і щоб МДНФ не містила термів, відповідних покриттям, які містять тільки знаки $*$.

Оптимальний варіант заміни наведено в карті Карно на рис. 3.8в, звідки маємо остаточну формулу

$$y_3^3 = a_1 \vee a_2\bar{b_1} \vee \bar{b_1}b_2$$

з ціною за Квайном $C(y_3^3) = 7$. Як видно з порівняння функцій $y_3^1 - y_3^3$, правильна заміна $* \to 1$ і $* \to 0$ значно зменшує вартість схеми.

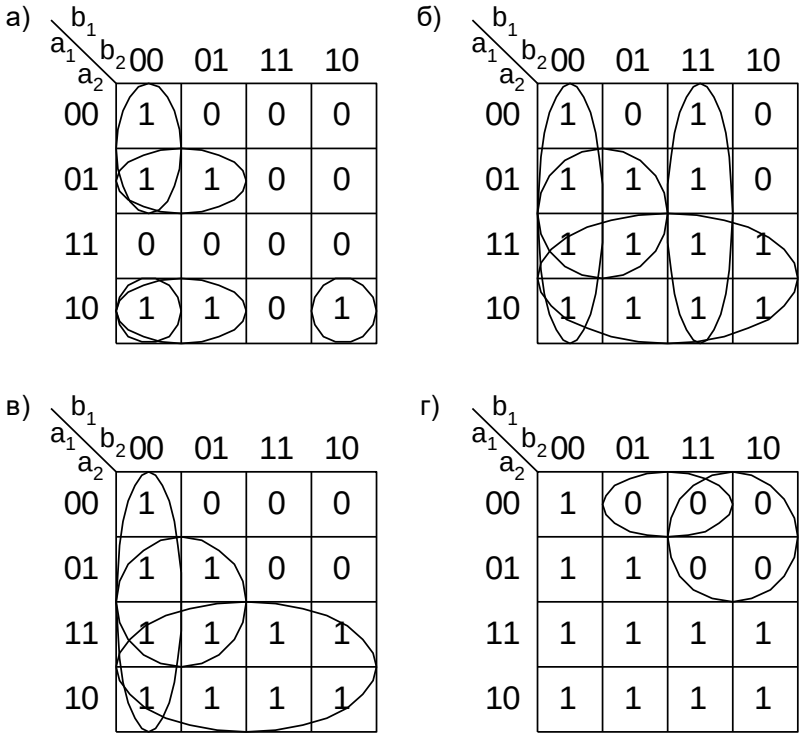


Рис. 3.8 – Повністю визначені функції, відповідні частковій функції y_3

При формуванні МКНФ необхідно керуватися такими самими правилами. Карта Карно (рис. 3.8г) дає змогу отримати МКНФ з ціною за Квайном $C(y_3^4) = 7$: $y_3^4 = (a_1 \vee \bar{b_2}) \cdot (a_1 \vee a_2 \vee \bar{b_2})$.

Оптимальна заміна вимагає практичних навичок, які, проте, досить швидко набуваються внаслідок відповідних тренувань.

Зауважимо, що при використанні формули (3.1) маємо $C(y_3^3) = 7$, $C(y_3^2) = 11$. Ця формула не враховує той факт, що якщо якийсь терм

містить тільки одну букву, то для його реалізації не потрібен вентиль. Уточнимо формулу (3.1) і представимо її у вигляді

$$C(y) = \sum_{h=1}^{H} (L_h - i_h) + H, \qquad (3.11)$$

де i_h – булева змінна, що дорівнює одиниці, якщо і тільки якщо h-ий терм ДНФ (КНФ) містить одну букву.

Тепер, згідно з (3.11) для функції y_3^3, наприклад, маємо таку вартість за Квайном: $C(y_3^3) = (1-1) + (2-0) + (2-0) + 3 = 7$.

3.3 Метод Квайна–Мак-Класкі

Метод Квайна–Мак-Класкі є формалізованим методом мінімізації булевих функцій, що використовують практично той самий алгоритм, який ми розглядали при використанні карт Карно. Проілюструємо цей метод на прикладі мінімізації функції φ (рис. 3.9).

Ідея методу Квайна–Мак-Класкі полягає в послідовному перебуванні 1-кубів по 0-кубів, 2-кубів по 1-кубів і так далі. Цей процес виконується максимум за L кроків, де L – число вхідних змінних функції. На останньому етапі шукається мінімальне покриття, тобто покриття функції кубами, що породжує мінімальну ДНФ.

x_1 \ x_3 x_4 x_2	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	1	0	0
10	1	0	1	0

Рис. 3.9 – Карта Карно функції φ

1. На першому етапі будується таблиця розмірності $H_0 \times H_0$, де H_0 – число одиниць в карті Карно (або в таблиці істинності функції). Рядки і стовпці таблиці визначаються кон'юнкцією рангу L, відповідними 0-кубам, для яких функція дорівнює одиниці. На перетині рядка i і

стовпця j записується 1-куб, що покриває терми з рядка i і стовпця j, якщо такий 1-куб існує. Якщо деякий терм не входить ні в один з отриманих 1-кубів, то він характеризується знаком «+», що вказує на входження даного терма в мінімальну ДНФ. Назвемо цю таблицю таблицею 0-кубів. Для функції φ таблиця 0-кубів приведена в табл. 3.3.

Таблиця 3.3

Таблиця 0-кубів функції φ

F_h	$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	$\bar{x}_1x_2x_3\bar{x}_4$	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$x_1\bar{x}_2x_3x_4$	$x_1x_2\bar{x}_3\bar{x}_4$	$x_1x_2\bar{x}_3x_4$	
$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$	-	$\bar{x}_1\bar{x}_2\bar{x}_4$	$\bar{x}_1\bar{x}_3\bar{x}_4$		$\bar{x}_2\bar{x}_3\bar{x}_4$				∨
$\bar{x}_1\bar{x}_2x_3\bar{x}_4$	-	-	-	$\bar{x}_1x_3\bar{x}_4$					∨
$\bar{x}_1x_2\bar{x}_3\bar{x}_4$	-	-	-	$\bar{x}_1x_2\bar{x}_4$			$x_2\bar{x}_3\bar{x}_4$		∨
$\bar{x}_1x_2x_3\bar{x}_4$	-	-	-	-					∨
$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	-	-	-	-	-		$x_1\bar{x}_3\bar{x}_4$		∨
$x_1\bar{x}_2x_3x_4$	-	-	-	-	-	-			+
$x_1x_2\bar{x}_3\bar{x}_4$	-	-	-	-	-	-	-	$x_1x_2\bar{x}_3$	∨
$x_1x_2\bar{x}_3x_4$	-	-	-	-	-	-	-	-	∨

З табл. 3.3 випливає, що терм $F_{11} = x_1\bar{x}_2x_3x_4$ входить в мінімальну ДНФ, а решту термів утворюють $H_1 = 8$ 0-кубів. Оскільки табл. 3.3 симетрична щодо її головної діагоналі внаслідок комутативного закону, то заповнюється тільки її права частина.

2. Будується таблиця 1-кубов розмірності $H_1 \times H_1$, рядки і стовпці якої характеризуються однокубами, отриманими на попередньому етапі. З цієї таблиці знаходяться 2-куби, для яких буде тривати процес об'єднання, і 1-куби, що не склеюються з іншими 1-кубами. Очевидно, однокуби, позначені символом «+», входять в мінімальну ДНФ булевої функції. Таблиця 1-кубів функції φ (табл. 3.4) має розмірність 8×8. З аналізу табл. 3.4 випливає, що терм $x_1x_2\bar{x}_3$ входить в мінімальну ДНФ, а решту термів функції утворюють $H_2 = 2$ куби розмірності 2.

3. Будується таблиця 2-кубів розмірності $H_2 \times H_2$, яка потрібна для знаходження 3-кубів, а також 2-кубів, що входять в мінімальну ДНФ. Для функції φ ця таблиця (табл. 3.5) має розмірність 2×2.

Як випливає з табл. 3.5, 2-куби функція φ не об'єднує, тому процес побудови кубів закінчений ($H_3 = 0$).

Таблиця 3.4

Таблиця 1-кубів функції φ

1-куби	$\bar{x}_1\bar{x}_2\bar{x}_4$	$\bar{x}_1\bar{x}_3\bar{x}_4$	$\bar{x}_1\bar{x}_3\bar{x}_4$	$\bar{x}_1x_3\bar{x}_4$	$\bar{x}_1x_2\bar{x}_4$	$x_2\bar{x}_3\bar{x}_4$	$x_1\bar{x}_3\bar{x}_4$	$x_1x_2\bar{x}_3$	
$\bar{x}_1\bar{x}_2\bar{x}_4$	—				$\bar{x}_1\bar{x}_4$				∨
$\bar{x}_1\bar{x}_3\bar{x}_4$	—	—		$\bar{x}_1\bar{x}_4$			$\bar{x}_3\bar{x}_4$		∨
$\bar{x}_1\bar{x}_3\bar{x}_4$	—	—	—			$\bar{x}_3\bar{x}_4$			∨
$\bar{x}_1x_3\bar{x}_4$	—	—	—	—					∨
$\bar{x}_1x_2\bar{x}_4$	—	—	—	—	—				∨
$x_2\bar{x}_3\bar{x}_4$	—	—	—	—	—	—			∨
$x_1\bar{x}_3\bar{x}_4$	—	—	—	—	—	—	—		∨
$x_1x_2\bar{x}_3$	—	—	—	—	—	—	—	—	+

Таблиця 3.5

Таблиця 2-кубів функції φ

2-куби	$\bar{x}_1\bar{x}_4$	$\bar{x}_3\bar{x}_4$	
$\bar{x}_1\bar{x}_4$	—		+
$\bar{x}_3\bar{x}_4$	—	—	+

4. Знаходиться покриття вихідної функції отриманими кубами, для чого може бути використаний, наприклад, метод Петрика. Цей метод заснований на побудові таблиці покриття, рядки якої характеризуються отриманими кубами, позначеними знаком «+», а стовпці – вихідними 0-кубами. На перетині i-го рядка і j-го стовпця таблиці покриття ставиться знак ∨, якщо вершина зі шпальти j входить в куб з рядка i. Потім для кожного стовпця знаходиться диз'юнкція, що покриває його рядки, і отримані диз'юнкції для всіх стовпців логічно множаться. Застосування

законів булевої алгебри дає змогу отримати ДНФ результуючої функції покриття. Кожен терм цієї ДНФ відповідає покриттю всіх стовпців таблиці покриття. З них вибирається терм, який відповідає ДНФ вихідної функції з мінімальним числом букв.

Для функції φ таблиця покриття (табл. 3.6) має $H = 8$ стовпців $(a - h)$ і чотири рядки (A, B, C, D).

Таблиця 3.6

Таблиця покриття функції φ

υ-куб → / терми ↓	a $\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$	b $\bar{x}_1\bar{x}_2x_3\bar{x}_4$	c $\bar{x}_1x_2\bar{x}_3\bar{x}_4$	d $\bar{x}_1x_2x_3\bar{x}_4$	e $x_1\bar{x}_2\bar{x}_3\bar{x}_4$	f $x_1\bar{x}_2x_3x_4$	g $x_1x_2\bar{x}_3\bar{x}_4$	h $x_1x_2\bar{x}_3x_4$	
$x_1\bar{x}_2x_3x_4$						∨			A
$x_1x_2\bar{x}_3$							∨	∨	B
$\bar{x}_1\bar{x}_4$	∨	∨	∨	∨					C
$\bar{x}_3\bar{x}_4$	∨		∨		∨		∨		D

З цієї таблиці будуються формули для покриття стовпців:

$$a = C \vee D; b = C;\ c = C \vee D;\ d = C;\ e = D;\ f = A;\ g = B \vee D;\ h = B.$$

Далі формується формула функції покриття, що має такий вигляд:

$$\Psi = (C \vee D)C(C \vee D)CDA(B \vee D)B = (C \vee D)CDA(B \vee D)B =$$
$$= (CDA \vee CDA)(B \vee BD) = CDAB.$$

Отже, для функції φ існує єдиний варіант мінімальної ДНФ, а саме:

$$\varphi = x_1\bar{x}_2x_3x_4 \vee x_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_4 \vee \bar{x}_3\bar{x}_4.$$

Зауважимо, що такий самий варіант виходить і по карті Карно (рис. 3.9).

Вид функції Ψ можна спростити, використовуючи такі міркування. Якщо j-ий стовпець має тільки одну мітку $\vee$, то терм з i-го рядка, де знаходиться ця мітка, повинен обов'язково входити в мінімальну ДНФ. В іншому разі 0-куб з j-го стовпця не буде відображений в мінімальній ДНФ, і формула буде некоректною. Оскільки i-ий терм обов'язково увійде в мінімальну ДНФ, то з таблиці покриття можна виключити всі стовпці, що мають мітку $\vee$ на перетині з i-им рядком. Ця процедура дає змогу

зменшити число формул покриття стовпців і, отже, спростити процедуру покриття Ψ.

У такий спосіб з табл. 3.6 виключається рядок A, оскільки як тільки вона покриває терм f, рядок $B - h$ (що дає змогу видалити і стовпець g), рядок $C - b$ і d (що дає змогу видалити і стовпці a, c), рядок $D - e$ і g (що дає змогу видалити стовпці a, c). Отже, $\Psi = ABCD$, тобто ніяких обчислень для неї не потрібно.

Якщо застосувати розглянутий метод Квайна–Мак-Класкі до функції φ_1 (рис. 3.10), то отримаємо терми

$$A = \bar{x}_3\bar{x}_4,\ B = \bar{x}_1\bar{x}_4,\ C = \bar{x}_1x_3,\ D = x_1x_2\bar{x}_3,\ E = \bar{x}_2x_3x_4.$$

x_1x_2 \ x_3x_4	00	01	11	10
00	1	0	1	1
01	1	0	1	1
11	1	1	0	0
10	1	0	1	0

Рис. 3.10 – Карта Карно функції φ_1

Побудуємо таблицю покриття для функції φ_1 (табл. 3.7), в якій для спрощення замість змінних написані відповідні їм набори. Після аналізу цієї таблиці можна побачити, що терми A, C, D, E повинні входити в остаточну функцію, що виключає з таблиці покриття стовпці a, d, g, i (покриваються A), b, c, e, f (покриваються C), i, j (покриваються D) і c, h (покриваються E). Отже, з табл. 3.7 видаляються всі стовпці, визначаючи функцію $\varphi_1 = ACDE = \bar{x}_3\bar{x}_4 \vee \bar{x}_1x_3 \vee x_1x_2\bar{x}_3 \vee \bar{x}_2x_3x_4$.

Як видно з цього прикладу, терм B є надмірним і не входить в остаточний результат. Проблема пошуку надлишкових термів дуже важлива. Як показано далі, для її вирішення використовуються складні формальні методи, засновані на евристиках.

Таблиця 3.7

Таблиця покриття функції φ_1

	0000	0010	0011	0100	0110	0111	1000	1011	1100	тисяча сто один	
** 00	∨			∨			∨		∨		A
0 ** 0	∨	∨		∨	∨						B
0 * 1 *		∨	∨		∨	∨					C
110 *									∨	∨	D
* 011			∨					∨			E
	a •	*b* •	*c* •	*d* •	*e* •	*f* •	*g* •	*h*	*i* •	*j*	

Якщо початкова функція є частковою, то для її мінімізації використовується такий підхід:

1. Усі невизначеності (позначені знаком ∗) в карті Карно або таблиці істинності замінюються одиницями.

2. Знаходяться терми, відповідні рядкам таблиці покриття. Терми, відповідні кубам, які охоплюють тільки невизначені набори, виключаються з таблиці покриття.

3. Знаходиться мінімальна ДНФ вихідної функції.

Розглянемо як приклад булеву функцію φ_2 (рис. 3.11а), яка включає невизначеності.

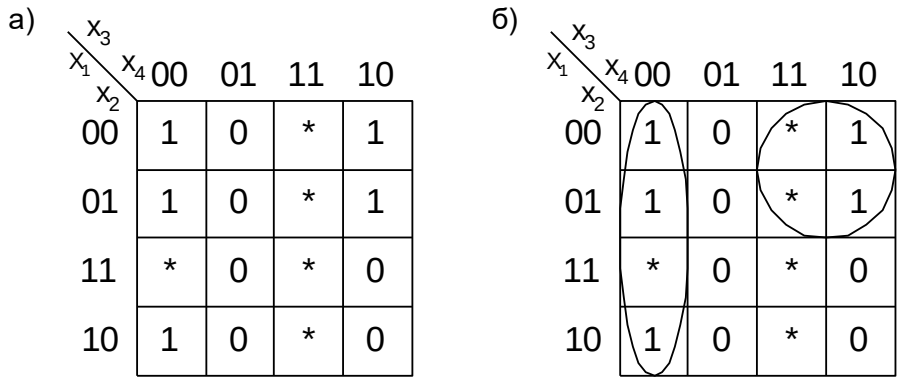


Рис. 3.11 – Функція φ_2 (а) та оптимальне рішення (б)

Замінивши всі знаки∗ одиницями, можна отримати такі терми $A = \bar{x}_3\bar{x}_4, B = \bar{x}_1\bar{x}_4, C = \bar{x}_1x_3, D = x_3x_4,$ чому відповідає таблиця покриття (табл. 3.8). Зауважимо, що терм x_3x_4 не входить в цю таблицю, оскільки він покриває тільки клітини з невизначеностями. З цієї ж причини в таблицю не входять стовпці 1011 і 1111, оскільки вони покриваються тільки термом D.

Таблиця 3.8

Таблиця покриття функції φ_2

	0000	0010	0011	0100	0111	0110	1011	1100	
** 00	∨			∨			∨	∨	A
0 ** 0	∨	∨		∨		∨			B
0 * 1 *		∨	∨		∨	∨			C
	a •	*b* •	*c* •	*d* •	*e* •	*f* •	*g* •	*h*	

З аналізу табл. 3.8 випливає, що терм A повинен бути включений в ДНФ, що виключає стовпці a, d, g, h. Аналогічно терм C також потрібно включити в мінімальну ДНФ, що виключає стовпці b, c, e, f. Отже, маємо формулу $\varphi_2 = AC = \bar{x}_3\bar{x}_4 \vee \bar{x}_1x_3,$ що відповідає рішенню, знайденому на карті Карно (рис. 3.11б).

Метод Квайна–Мак-Класкі застосуємо також для формування мінімальної КНФ. В цьому разі вихідна таблиця будується для 0-кубів, відповідних наборам, на яких функція дорівнює нулю. Щоб звести задачу до вихідної, достатньо у всіх таблицях замість мінтермів розглядати відповідні їм куби.

Наприклад, отримаємо мінімальну КНФ функції φ_3 (рис. 3.12).

x_1x_2 \ x_3x_4	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	1	1
10	0	1	0	1

Рис. 3.12 – Карта Карно функції φ_3

1. Побудуємо таблицю 0-кубів (табл. 3.9).

З аналізу табл. 3.9 випливає, що терм $\Phi_{II} = \bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4$ входить в мінімальну КНФ, а решта термів утворюють $H_1 = 8$ 1-кубів.

Таблиця 3.9

Таблиця 0-кубів функції φ_3

0-куби	0000	0010	0100	0110	1000	1011	1100	тисяча сто один	
0000	—	00 * 0	0 * 00	—	* 000				∨
0010	—	—		0 * 10					∨
0100	—	—	—	01 * 0			* 100		∨
0110	—	—	—	—					∨
1000	—	—	—	—	—		1 * 00		∨
1011	—	—	—	—	—	—			+
1100	—	—	—	—	—	—	—	110 *	∨
тисяча сто один	—	—	—	—	—	—	—	—	∨

2. Побудуємо таблицю однокубів функції φ_3 (табл. 3.10).

Таблиця 3.10

Таблиця 1-кубів функції φ_3

1-куби	00 * 0	0 * 00	* 000	0 * 10	01 * 0	* 100	1 * 00	110 *	
0000	—				0 ** 0				∨
0 * 00	—	—		0 ** 0			** 00		∨
* 000	—	—	—			** 00			∨
0 * 10	—	—	—	—					∨
01 * 0	—	—	—	—	—				∨
* 100	—	—	—	—	—	—			∨
1 * 00	—	—	—	—	—	—	—		∨
110 *	—	—	—	—	—	—	—	—	+

З табл. 3.10 випливає, що терм $\bar{x}_1 \vee \bar{x}_2 \vee x_3$ входить в мінімальну КНФ, а решта термів утворюють $H_2 = 2$ 2-куби.

3. Побудуємо таблицю 2-кубів функції φ_3 (табл. 3.11). З аналізу цієї таблиці випливає, що всі 2-куби входять в мінімальну КНФ, і процес побудови кубів завершено. Зауважимо, що на практиці подібні таблиці

можуть містити сотні і тисячі рядків. Все це ускладнює завдання мінімізації булевих функцій практичної розмірності. З огляду на це на практиці використовуються кубічні уявлення, але тільки у вигляді рядків, які складаються з нулів і одиниць.

Таблиця 3.11

Таблиця 2-кубів функції φ_3

2-куби	0 ** 0	** 00	
0 ** 0	—		+
** 00	—	—	+

4. Побудуємо таблицю покриття функції φ_3 (табл. 3.12).

Таблиця 3.12

Таблиця покриття функції φ_3

	0000	0010	0100	0110	1000	1011	1100	тисяча сто один	
1011						∨			A
110 *							∨	∨	B
0 ** 0	∨	∨	∨	∨					C
** 00	∨		∨		∨		∨		D
	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	

Аналіз табл. 3.12 показує, що терм A входить в МКНФ, оскільки тільки він включає стовпець f; терм B входить в МКНФ, оскільки тільки він включає стовпець h; терм C входить в МКНФ, оскільки тільки він включає стовпець d; терм D входить в МКНФ, оскільки тільки він включає стовпець e. Отже, немає необхідності в побудові функції покриття, і мінімальна КНФ функції φ_3 має вигляд:

$$\varphi_3 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4) \vee (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \vee (x_1 \vee x_4) \vee (x_3 \vee x_4).$$

Якщо порівнювати МДНФ φ і МКНФ φ_3, то видно, що вони утворюються однаковими кубами. При цьому в разі ДНФ куби відповідають кон'юнктивним термам, а в разі КНФ – диз'юнктивним термам. Однаковість кубів, що утворюють мінімальні форми φ і φ_3, можна було очікувати, порівнявши карти Карно на рис. 3.9 і рис. 3.12.

Мінімальні форми шукалися для одних і тих самих 0-кубів з використанням одного і того самого методу, що і привело до однакових результатів.

Пошук МКНФ для часткової функції проводиться аналогічно пошуку МДНФ. Природно, що при цьому враховується той факт, що необхідно аналізувати нулі функції. Метод має вигляд:

1. Усі невизначеності замінюються нулями і знаходяться всі куби вищого порядку.

2. Якщо пошук кубів завершений, то куби, що покривають лише невизначені набори, виключаються з подальшого розгляду.

Зазначимо, що на практиці часто доводиться мінімізувати системи булевих функцій, однак ці методи досить трудомісткі і виходять за рамки цієї книги.

3.4 Евристичні методи мінімізації

Розглянутий метод Квайна–Мак-Класкі належить до точних методів мінімізації булевих функцій, що дає змогу знайти мінімальну ДНФ. Однак, як було вже зазначено, блоки ЕОМ задаються функціями, які мають десятки і сотні аргументів і тисячі термів. Такі функції, звичайно, можуть бути мінімізовані з використанням методу Квайна–Мак-Класкі, однак час мінімізації буде неприйнятно великим (дні і тижні). Зауважимо, що така мінімізація може виконуватися тільки ЕОМ, а за тиждень роботи цілком може статися збій. Це означає, що процес мінімізації треба буде починати спочатку. З огляду на це для вирішення практичних задач великої розмірності використовуються евристичні методи, що дають змогу знайти ненадлишкову ДНФ за прийнятний час. При цьому важливим є те, що евристичні методи не вимагають стільки пам’яті ЕОМ, як точні методи.

Терм F_i називається імплікантом функції f, якщо $f=1$ за $F_i=1$. Якщо імплікант F_i не є частиною будь-якого іншого імпліканта, то він

називається первинним імплікантом функції. Проблема пошуку первинних імплікантів для великого числа аргументів дуже складна. Розглянемо карту Карно (рис. 3.13), що задає деяку функцію φ_4.

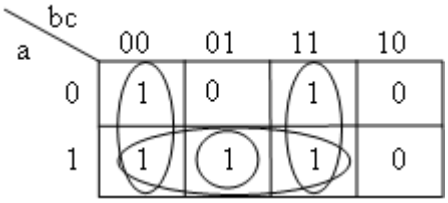


Рис. 3.13 – Карта Карно для функції φ_4

Ця карта містить чотири первинні імпліканти, відповідні обведеним областям. Це імпліканти $\bar{b}\bar{c}$, $a\bar{b}$, ac і bc, яким відповідають куби $*00$, $10*$, $1*1$ і $*11$. При цьому карта на рис. 3.13 містить п'ять імплікантів, відповідних кубам 000, 100, 101, 011 і 111. Первинний імплікант називається істотним, якщо тільки він покриває деякий мінтерм функції. Істотні імпліканти не можна видалити з кінцевої ДНФ, оскільки при цьому буде синтезовано деякий інший пристрій, а не той, який було задано початковій ДДНФ. Для нашого прикладу первинний імплікант $\bar{b}\bar{c}$ є істотним, тому що тільки він покриває набір 000. Аналогічно первинний імплікант bc є істотним, тому що тільки він покриває набір 011. Решта імплікантів не є суттєвими.

В евристичних методах мінімізація починається з первинного покриття, яке охоплює всі мінтерми (набори, відповідні одиницям в карті Карно) і несуттєві набори (позначені знаком *). Для отримання рішення використовуються чотири основні операції:

1. Розширення (Expand). При цьому кожний непервинний імплікант замінюється своїм первинним імплікантом. Потім з функції видаляються всі імпліканти, що покриваються первинними імплікантами.

2. Винятки (Reduce). Ця операція використовується для видалення з покриття надлишкових первинних імплікантів. При цьому таке видалення неможливе, якщо імплікант є істотним. Для перевірки можливості

видалення імпліканта виконується перевірка, яка повинна показати, чи є імплікант суттєвим.

3. Переформатування (Reshape). Ця операція розглядає пари імплікантів, і якщо один із імплікантів покривається іншим, то імплікант, що покривається, видаляється.

4. Ненадлишковість (Irredundant). З покриття функції видаляються несуттєві імпліканти зі збереженням покриття всіх вихідних мінтермів.

Відома низка практичних методів, які використовують ці операції. Зокрема, програма PRESTO використовує тільки операцію Expand. При цьому отримують рішення швидко, але воно може бути далеким від оптимального результату. Програма MINI використовує перші три з розглянутих вище операцій. При цьому операції Reduce і Reshape використовуються для поліпшення початкового покриття, яке виходить після процедури Expand. Однак покриття може бути надмірним. Наприклад, для функції φ_4 може бути отримана ДНФ виду: $\varphi_4 = \bar{b}\,\bar{c} \vee bc \vee a\bar{b} \vee ac$. Зрозуміло, що одна з умов ($a\bar{b}$ або ac) може бути видалена. Подібні стратегії використані в програмах POP і PRESTO-L, що засновані на методі PRESTO. Всі чотири операції використовуються в програмі ESPRESSO, яка розроблена в Каліфорнійському Університеті (США). Зараз ця програма входить практично в усі промислові пакети, які використовуються для синтезу цифрових пристроїв.

Розглянемо основні ідеї, що використовуються в методі ESPRESSO, на прикладі функції φ_5 (рис. 3.14).

ab \ cd	00	01	11	10
00	1			1
01	1	1	1	**1**
11		1		
10	1	1	1	1

Рис. 3.14 – Карта Карно функції φ_5

Отже, функція φ_5 має 11 мінтермів, які можуть бути покриті такими шістьма первинними імплікантами: $F_1 = \bar{a}\bar{d}$ (куб $0{**}0$), $F_2 = \bar{b}\bar{d}$ ($*0*0$), $F_3 = \bar{a}b$ ($01{**}$), $F_4 = a\bar{b}$ ($10{**}$), $F_5 = a\bar{c}d$ ($1*01$), $F_6 = b\bar{c}d$ ($*101$). При цьому імпліканти F_3 і F_4 є суттєвими. Виключивши їх з розгляду, отримаємо таблицю покриття (табл. 3.13).

Таблиця 3.13

Таблиця покриття функції φ_5

	0000	0010	тисяча сто один	F_i
0 ** 0	∨	∨		F_1
* 0 * 0	∨	∨		F_2
1 * 01			∨	F_5
* 101			∨	F_6

З табл. 3.13 можна отримати чотири мінімальні покриття, що визначається рівнянням: $\varphi_5 = (F_1 \vee F_2) \cdot (F_5 \vee F_6) \vee F_3 \vee F_4$.

Запишемо імпліканти функції φ_5 в такому порядку: 0000 0010, 0100, 0110, 1000, 1010, 0101, 0111, 1001, 1011 1101. Цей набір розглядається як мінімальне покриття, яке буде поліпшуватися. Застосуємо операцію Expand до імплікантів в тому порядку, як вони записані. Якщо який-небудь мінтерм покривається отриманим після Expand результатом, то він відкидається. Наприклад, імплікант 0000 може бути розширений до 0 ** 0, що дає змогу виключити з розгляду мінтермів 0010, 0100 і 0110. При цьому напрямок розширення, тобто, за якими розрядами мінтермів відбувається порівняння, задається в програмі. Наприклад, нехай розглянуті всі імпліканти, крім 1101. Нехай при цьому отримано покриття $\alpha = \{F_1, F_2, F_3, F_4\}$. Залежно від напрямку розширення з 1101 можна отримати або F_5 (1 * 01), або F_6(* 101). Якщо вибір напрямку дає F_5, то отримуємо покриття $\{F_L, \ldots, F_5\}$, що має 5 елементів.

Застосуємо до цього покриття операцію Reduce. Нехай вона застосовується в порядку номерів i в F_i. Первинний імплікант F_1 може бути зведений до порожнього імпліканта, оскільки всі його мінтерми покриті іншими імплікантами покриття α. Первинний імплікант F_2 може бути зведений до $F_2^{'} = 00*0$, оскільки частина його мінтермів покривається імплікантами F_4. Аналогічно імплікант F_5 може бути зведений до $F_5^{'} = 1101$, оскільки частина його одиниць покривається. При цьому приходимо до покриття $\beta = \{F_2^{'}, F_3, F_4, F_5^{'}\}$, що має чотири елементи.

Застосувавши операцію Reshape до пари $\left\langle F_2^{'}, F_4 \right\rangle$, отримаємо пару кубів $\left\langle F_2, F_4^{'} \right\rangle$, де $F_4^{'} = 10*1$. Тепер маємо покриття $\gamma = \{F_2, F_3, F_4^{'}, F_5^{'}\}$. Наступне застосування операції Expand веде до покриття $\delta = \{F_2, F_3, F_4, F_5\}$, яке складається з чотирьох первинних імплікантів і є мінімальним. Очевидно, що покриттю δ відповідає ДНФ $\varphi_5 = \bar{b}\bar{d} \vee \bar{a}b \vee a\bar{b} \vee a\bar{c}d$. Зауважимо, що застосування операцій Expand, Reduce та Reshape не гарантує отримання мінімального або ненадлишкових покриттів. Однак застосування операції Irredundant гарантує отримання ненадлишкових покриттів, оскільки вона видаляє з покриття або F_1, або F_2. Загалом результат мінімізації значною мірою визначається системою правил, які використовуються в евристичних алгоритмах.

3.5 Основні алгоритми евристичної мінімізації

Метою операції Expand є збільшення розміру кожного імпліканта (збільшення числа позицій, позначені знаком *) деякого покриття α. Це дає змогу видалити з покриття імпліканти меншого розміру, що покриваються отриманими первинними імплікантами.

Нехай F^1, F^0, F^* – множина мінтермів, для яких функція f дорівнює відповідно 0, 1 або приймає невизначене значення. Використовуємо два двійкові розряди для представлення кожної змінної: 01 для 1, 10 для 0 і 11 для *. Операція Expand виконується через перехід від 1 до невизначеності *, тобто від 01 до 11. За такого переходу необхідно перевірити, щоб отриманий куб покривав тільки мінтерми з множини F^1 і F^0. Це виконується одним з двох способів:

– знаходження перетину отриманого куба з кубами з множини F^0. Цей підхід використовується в програмах MINI і ESPRESSO;

– знаходження того, що новий куб покриває тільки мінтерми з множин F^1 і F^*. При цьому множина F^0 не використовується. Такий підхід використаний в програмі PRESTO.

Результат мінімізації (як і її час) залежить як від порядку вибору розгортання імплікантів, так і позицій, в яких 1 замінюються *. У програмах MINI і ESPRESSO використовують такий підхід для вирішення цього завдання. Кожному набору з L вхідних змінних відповідає набір, який має $2L$ розрядів. Формується матриця F^1 і знаходяться ваги її стовпців як суми одиниць і нулів в цих стовпцях. Потім знаходиться порозрядний добуток кожного куба на вектор імпліканта, і результат підсумовується. Імпліканти упорядковано відповідно до зменшення ваги. Природно, що первинні імпліканти, отримані на попередніх кроках, не беруть участі в подальшій процедурі.

Приклад 3.3. Застосувати процедуру Expand до функції y_1 (рис. 3.15).

a \ bc	00	01	11	10
0	1	1	0	1
1	1	0	0	*

Рис. 3.15 – Карта Карно для функції y_1

З цієї карти можна отримати три матриці кубів для множини F^1, F^0 і F^*, які представлені на рис. 3.16. Ця матрична форма використовується для подальшої мінімізації функції.

$$F^1 = \begin{vmatrix} 10 & 10 & 10 \\ 01 & 10 & 10 \\ 10 & 01 & 10 \\ 10 & 10 & 01 \end{vmatrix};\ F^* = \begin{vmatrix} 01 & 01 & 10 \end{vmatrix};\ F^0 = \begin{vmatrix} 01 & 11 & 01 \\ 11 & 01 & 01 \end{vmatrix}.$$

Рис. 3.16 – Матричне представлення функції y_1

Підсумовуючи всі одиниці в колонках матриці F^1, побудуємо вектор $\langle 3,1,3,1,3,1 \rangle$. Помноживши цей вектор на перший рядок матриці F^1, отримаємо вектор $\langle 3,0,3,0,3,0 \rangle$, на другий – $\langle 0,1,3,0,3,0 \rangle$, на третій – $\langle 3,0,0,1,3,0 \rangle$, на четвертий – $\langle 3,0,3,0,0,1 \rangle$. Підсумовуючи компоненти цих векторів, отримаємо ваги імплікантів, що утворюють вектор $\langle 9,7,7,7 \rangle$. Отже, першим для Expand вибирається імплікант $\bar{a}\bar{b}\bar{c}$, що має вагу 9.

Нехай 1 змінюються на * зліва направо (для спрощення процедури). Змінивши цифру 0 в першій колонці імпліканта $\bar{a}\bar{b}\bar{c}$ на 1, отримаємо вектор $\langle 11 \quad 10 \quad 10 \rangle$, який не перетинається з векторами з матриці F^0. Змінивши 0 на 1 в четвертій колонці, отримаємо вектор $\langle 11 \quad 11 \quad 10 \rangle$, який також не перетинається з F^0. Змінивши 0 на 1 в колонці 6, отримаємо вектор $\langle 11 \quad 11 \quad 11 \rangle$, який перетинається з векторами з F^0. Отже, отримуємо розширений імплікант $\langle 11 \quad 11 \quad 10 \rangle$, який покриває перший, другий і третій рядки матриці F^1, яка тепер представляється у вигляді

$$F^1 = \begin{vmatrix} 11 & 11 & 10 \\ 10 & 10 & 01 \end{vmatrix}.$$

Тепер обробляється останній імплікант. При зміні другої колонки отримуємо вектор $\langle 11 \quad 10 \quad 01 \rangle$, який перетинається з F^0. Після зміни колонки 5 отримуємо вектор $\langle 10 \quad 10 \quad 11 \rangle$, який не перетинається з F^0. Отже, маємо такий результат, в якому остання компонента змінена з 01 на 11:

$$F^1 = \begin{vmatrix} 11 & 11 & 10 \\ 10 & 10 & 11 \end{vmatrix}.$$

Цей результат відповідає мінімальній ДНФ $y_5 = \bar{c} \vee \bar{a}\bar{b}$, яка легко може бути отримана з рис. 3.15.

Метою операції Reduce є зменшення розмірності кожного імпліканта в початковому покритті для подальшого застосування операції Expand. Після застосування операції Reduce отримують скорочені імпліканти. Редукований імплікант називається значущим (valid) імплікантом, якщо він покриває вихідну функцію разом з рештою імплікантів. При цьому операція Reduce не применшує число імплікантів в покритті. Отже, покриття може залишатися надмірним. У нашому попередньому прикладі зменшення розмірності імпліканта $\bar{c}$ приводить до збільшення числа імплікантів в покритті. Наприклад, якщо перетворити куб ** 0 в куб 0 * 0, то в покриття доведеться ввести куб 100. Куб 00 * може бути скорочений до 001 без введення додаткових кубів. Це приводить до ДНФ $y_5 = \bar{c} \vee \bar{a}\bar{b}c$.

Операція Irredundant використовується для отримання покриття, яке відрізняється від початкового максимально можливим числом віддалених надлишкових імплікантів. Ця операція була вперше застосована в алгоритмі ESPRESSO і виконується після операції Expand, коли всі імпліканти в покритті є первинними. Таке покриття F розподіляється на 3 класи:

– клас E^R щодо істотних імплікантів, які покривають ті функції мінтермів, які не мають покриття іншими первинними імплікантами з F;

– клас R^t повністю надлишкових імплікантів, до яких належать імпліканти, які покриваються імплікантами класу E;

– клас R^P частково надлишкових імплікантів, до цього класу належать імпліканти з множини $F/(E' \cup R')$.

Для визначення класу E^R необхідно перевірити, чи не покривається імплікант $F_i \in F$ імплікантами з $F \cup F^* - \{\alpha\}$. Далі для визначення елементів класу R^t необхідно перевірити, чи покривається мінтерм $F_i \in F$ імплікантами з множини $E^R \cup F^*$. Це дає змогу знайти множину R^P як різницю між F і $E^R \cup R^t$. Основним завданням операції Irredundant є знаходження деякої підмножини з множини R^P, яка разом з E^R покриває F^1.

Приклад 3.4. Визначити множини E^R, R^t, R^P функції $y_6 = f(a,b,c)$, заданої картою Карно (рис. 3.17).

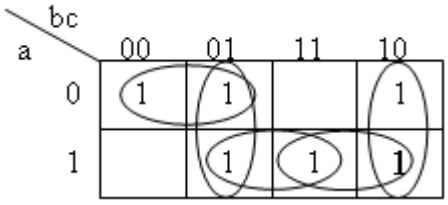


Рис. 3.17 – Карта Карно для функції y_6

Області, обведені на карті, відповідають покриттю F, матрична форма якого наведена на рис. 3.18.

$$F = \begin{array}{c|ccc} F_1 & 10 & 10 & 11 \\ F_2 & 11 & 10 & 01 \\ F_3 & 01 & 11 & 01 \\ F_4 & 01 & 01 & 11 \\ F_5 & 11 & 01 & 10 \end{array}$$

Рис. 3.18 – Матрична форма покриття F функції y_6

З карти Карно на рис. 3.17 можна знайти, що $E^R = \{F_1, F_5\}$, $R^t = \varnothing$, $R^R = \{F_2, F_3, F_4\}$. Тепер необхідно вибрати з множини R^R мінімальну за розміром підмножину, яка разом з E^R покриває функцію y_6. З рис. 3.17 очевидно, що це покриття $F = \{F_1, F_3, F_5\}$. Однак для його формального пошуку необхідно застосувати досить складний алгоритм.

Процедура Essential дає змогу вибрати істотні імпліканти, що необхідно для алгоритмів точної та евристичної мінімізації. Ці імпліканти виключаються з подальшого розгляду і додаються в кінці процесу мінімізації до отриманого програмою покриття. Наприклад, перевіримо істотність для імпліканта F_1 (рис. 3.18). Відповідь буде позитивною, оскільки видалення імпліканта F_1 залишає мінтерм 000 непокритим.

Усі ці операції використовуються в програмі ESPRESSO, яка дає змогу досить швидко знайти ненадлишкове покриття. Як оцінка використовується процедура Cost (F), що дає змогу знайти число термів в покритті F. Алгоритм виконується доти, доки ціна кожного наступного покриття буде меншою, ніж попереднього. Якщо поліпшення не досягнуто, то робиться остання спроба поліпшення рішення, що виконується процедурою last.gasp. У цій процедурі використовуються більш складні евристичні правила, ніж в основному циклі. З огляду на це час процедури last.gasp значно перевищує час ітерації в основному циклі. У спрощеній формі алгоритм ESPRESSO представлений на рис. 3.19.

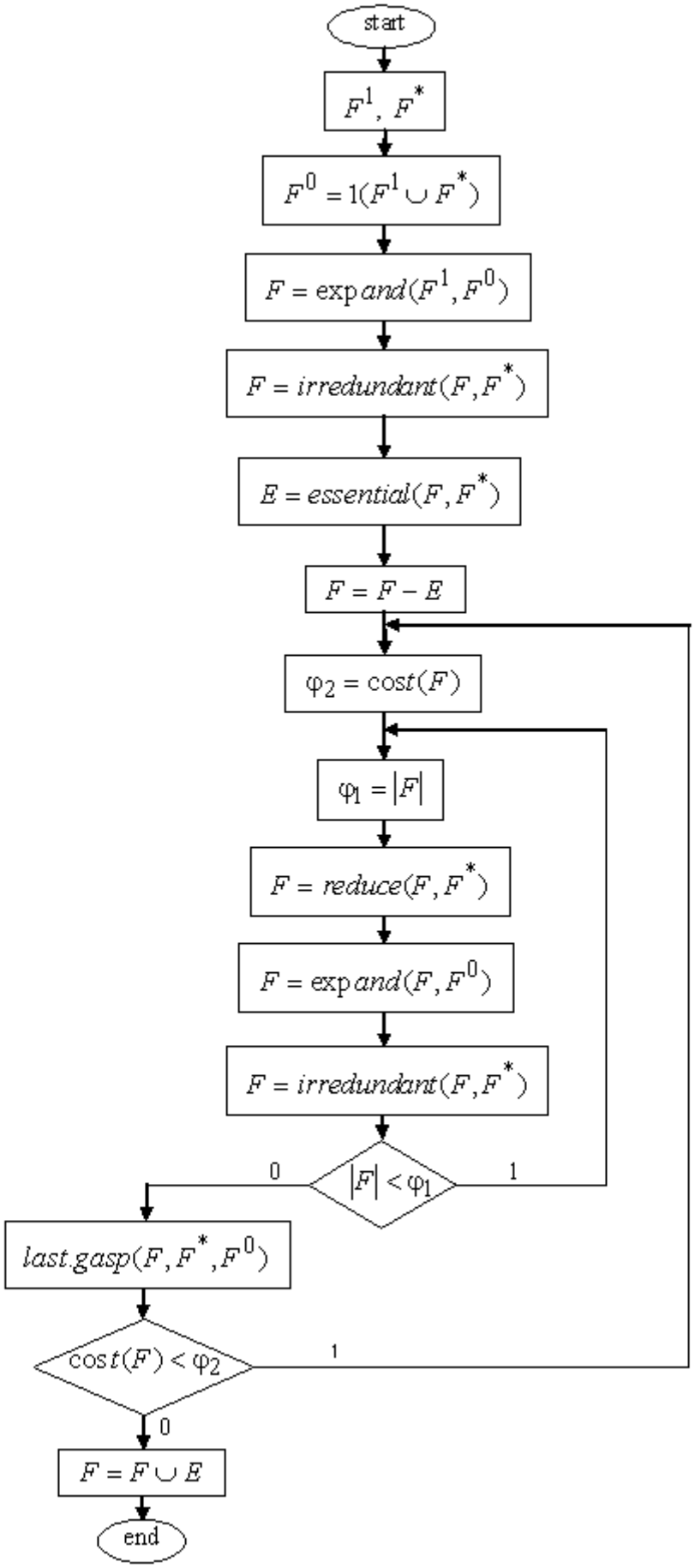


Рис. 3.19 – Спрощена блок-схема алгоритму EESPRESSO

Список літератури до розділу 3

1. Прикладная теория цифровых автоматов / К. Г. Самойлов, А. М. Романкевич, В. Н. Валуйский, Ю. С. Каневский, М. М. Пиневич. Киев: Вища шк., 1987. 375 с.

2. Савельев А. Я. Прикладная теория цифровых автоматов. Москва: Высш. шк., 1987. 272 с.

3. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

4. Закревский А. Д., Поттосин Ю. В., Черемисинова Л. Д. Основы логического проектирования. В 3-х кн. Кн. 1. Комбинаторные алгоритмы дискретной математики. Минск: ОИПИ НАН Беларуси, 2004. 226 с.

4 СИНТЕЗ КОМБІНАЦІЙНИХ СХЕМ

4.1 Синтез схем в базисах І-НЕ і АБО-НЕ

Під комбінаційною схемою розуміється деяка мережа логічних елементів, в якій немає зворотного зв'язку. Комбінаційна схема (КС) задається системою булевих функцій $Y = F(X)$, в якій $Y = \{y_1, \ldots, y_N\}$, $X = \{x_1, \ldots, x_L\}$, $y_n = f_n(x_1, \ldots, x_L)\,(n = 1, \ldots, N)$.

В обчислювальній техніці для синтезу КС використовується кінцева множина логічних елементів, що називається елементним базисом. Це поняття пов'язане з поняттям функціонального базису, під яким розуміється кінцеве число елементарних булевих функцій, достатніх для аналітичного подання довільної булевої функції.

Оскільки будь-яка функція може бути представлена у вигляді ДНФ або КНФ, то очевидно, що для подання довільних булевих функцій достатньо тільки елементарних функцій І, АБО, НЕ. Базис {І, АБО, НЕ} отримав назву булевого базису. Отже, для реалізації будь-якої КС достатньо елементів І, АБО і НЕ. Будь-яку ДНФ або КНФ можна виразити за допомогою операцій І-НЕ або АБО-НЕ, використовуючи закони де Моргана:

$$\overline{A \vee B} = \overline{A} \cdot \overline{B}, \tag{4.1}$$

$$\overline{A \cdot B} = \overline{A} \vee \overline{B}. \tag{4.2}$$

Відтак, базис І-НЕ (АБО-НЕ) є достатнім для реалізації довільної функції, що і було показано раніше.

На практиці переважно використовуються базиси І-НЕ і АБО-НЕ. Розглянемо методи реалізації комбінаційних схем в цих базисах.

1. Реалізація елементарної кон'юнкції в базисі І-НЕ.

Реалізація схеми ґрунтується на законі подвійної інверсії $\overline{\overline{A}} = A$. Наприклад, функція $F_3 = abc$ перетворюється до виду $F_3 = \overline{\overline{abc}}$, чому

відповідає комбінаційна схема (рис. 4.1). Ця схема має два рівні вентилів, причому вентиль другого рівня реалізує функцію інверсії, для чого використовується очевидне співвідношення

$$\overline{AA} = \overline{A}. \qquad (4.3)$$

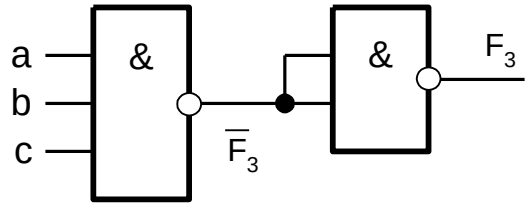


Рис. 4.1 – Реалізація елементарної кон'юнкції в базисі І-НЕ

Зазначимо, що для реалізації інвертора може бути використано і правило

$$\overline{A \cdot 1} = \overline{A}. \qquad (4.4)$$

2. Реалізація елементарної диз'юнкції в базисі І-НЕ.

Реалізація схеми заснована на одночасному використанні закону подвійного заперечення і одного із законів де Моргана. Наприклад, функція $\Phi_З = a \vee b \vee c$ перетворюється у такий спосіб: $\Phi_З = \overline{\overline{a \vee b \vee c}} = \overline{\overline{a} \cdot \overline{b} \cdot \overline{c}}$. Якщо джерело вхідних змінних формує як прямі, так і інверсні значення змінних a, b, c, то схема для $\Phi_З$ матиме тільки один рівень (рис. 4.2а). Якщо джерело вхідних сигналів формує тільки прямі значення змінних a, b, c, то схема має два рівні (рис. 4.2б).

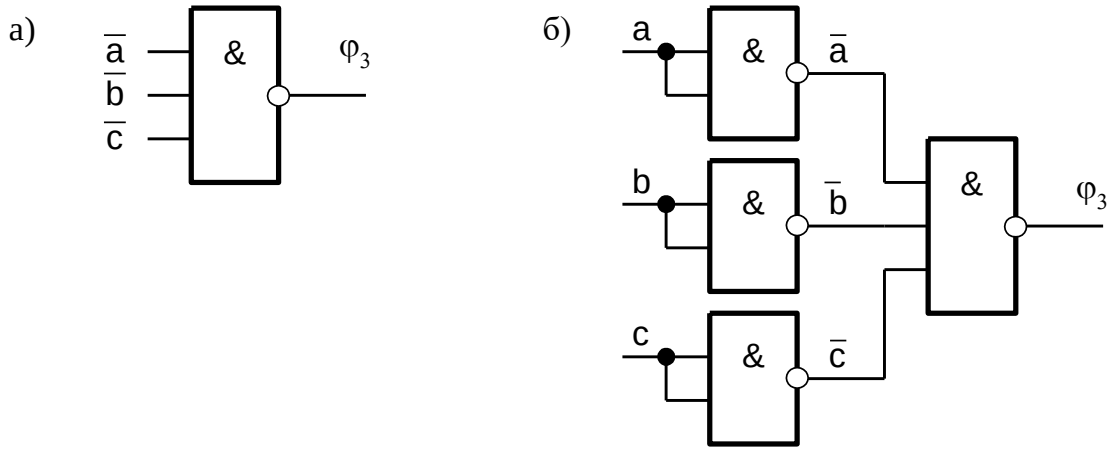


Рис. 4.2 – Реалізація елементарної диз'юнкції в базисі І-НЕ

Зауважимо, що друга ситуація є більш поширеною на практиці, де джерелами інформації є виходи регістрів.

3. Реалізація ДНФ в базисі І-НЕ.

Реалізація схеми заснована на одночасному застосуванні закону подвійного заперечення і закону де Моргана (4.1). Наприклад, для функції $y_1 = ab \vee cd = F_1 \vee F_2$ маємо: $y_1 = \overline{\overline{F_1 \vee F_2}} = \overline{\overline{F_1} \cdot \overline{F_2}} = \overline{\overline{ab} \cdot \overline{cd}}$, що відповідає схемі на рис. 4.3.

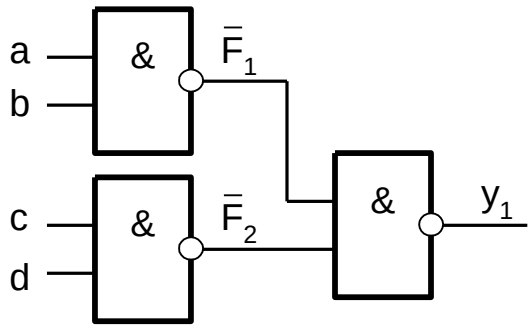


Рис. 4.3 – Реалізація ДНФ в базисі І-НЕ

4. Реалізація КНФ в базисі І-НЕ.

Оскільки термами КНФ є елементарні диз'юнкції, то кожна з них повинна бути перетворена за правилами, розглянутими в пункті 2. При цьому до КНФ загалом необхідно застосувати закон подвійної інверсії. Наприклад, для булевої функції $y_2 = (a \vee b)(c \vee d)$, що залежить від чотирьох змінних, маємо $y_2 = \Phi_1 \cdot \Phi_2 = \overline{\overline{\Phi_1} \cdot \overline{\Phi_2}} = \overline{\overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d}}$, що відповідає схемі на рис. 4.4.

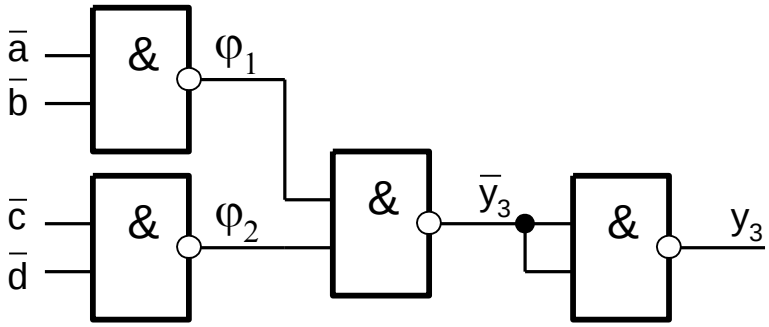


Рис. 4.4 – Реалізація КНФ в базисі І-НЕ

Кожен логічний елемент має деякий кінцевий час $t_{ЛЕ}$, який проходить від зміни сигналів на входах до зміни сигналів на його виході. Назвемо $t_{ЛЕ}$ часом перемикання логічного елемента. Комбінаційна схема загалом також має час перемикання $t_{кс}$, який вираховується як сума часів перемикання логічних елементів, що знаходяться на найдовшому ланцюжку від входу до виходу. Наприклад, для функції y_1 маємо $t_{кс} = 2$, а для y_2 – $t_{кс} = 3$. Отже, для даного конкретного випадку функції y_1 схема в базисі І-НЕ, побудована по ДНФ, в 1,5 рази швидше, ніж схема, побудована по КНФ.

При синтезі комбінаційних схем розробник може користуватися такими критеріями:

– мінімізація числа елементів схеми і, отже, її вартості в грошовому еквіваленті;

– мінімізація часу перемикання КС.

У разі використання другого критерію реалізація КНФ в базисі І-НЕ не використовується. Однак може використовуватися реалізація схеми по КНФ зворотної функції. Наприклад, для функції y_3 (рис. 4.5) може бути отримана КНФ для оберненої функції $\overline{y_3}$, яка використовує мінтерми, відповідні одиницям карти Карно.

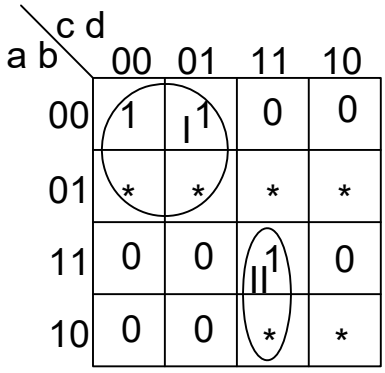


Рис. 4.5 – Карта Карно для функції y_3

З цієї карти Карно маємо формулу $\overline{y_3} = (a \vee c)(\overline{a} \vee \overline{c} \vee \overline{d})$. Очевидно, що для отримання прямої функції достатньо взяти інверсію функції $\overline{y_3}$, а

для реалізації термів в базисі І-НЕ необхідно використовувати подвійну інверсію і закон де Моргана: $y_3 = \overline{\overline{a \vee c} \cdot \overline{a \vee c \vee d}} = \overline{\overline{a} \cdot \overline{c} \cdot \overline{acd}}$, що приводить до схеми (рис. 4.6), яка має два рівні.

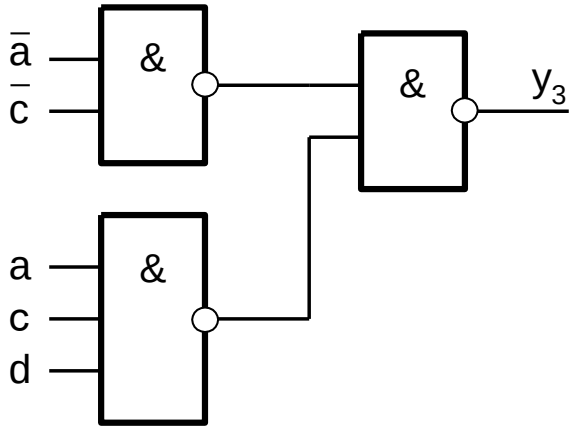


Рис. 4.6 – Реалізація КНФ зворотної функції в базисі І-НЕ

Ця схема має такий самий час перемикання, як і логічна схема, що реалізується за ДНФ.

5. Реалізація елементарної кон'юнкції в базисі АБО-НЕ.

Для синтезу схеми необхідно використовувати закон подвійної інверсії і закон де Моргана. Наприклад, для кон'юнктивного терму $F_3 = abc$ маємо $F_3 = \overline{\overline{abc}} = \overline{\overline{a} \vee \overline{b} \vee \overline{c}}$, що породжує схему (рис. 4.7).

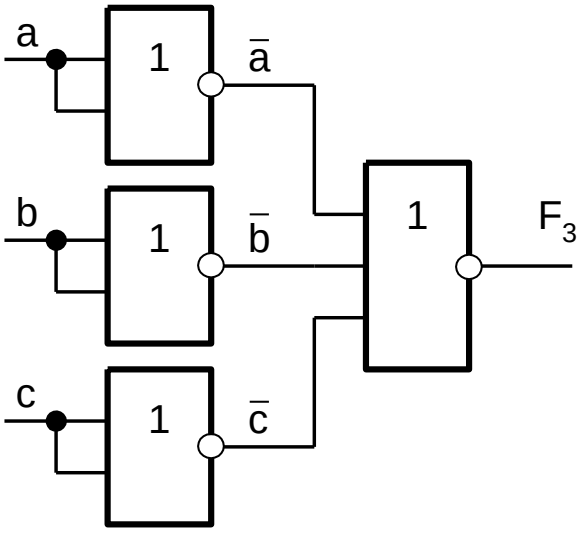


Рис. 4.7 – Реалізація елементарної кон'юнкції в базисі АБО-НЕ

У цій схемі для реалізації інверторів використано правило $\overline{a \vee a} = \bar{a}$. Очевидно, що в цій формулі може бути використаний і логічний нуль.

6. Реалізація елементарної диз'юнкції в базисі АБО-НЕ.

Для реалізації терму необхідно використовувати правило подвійної диз'юнкції. Наприклад, для виразу $\varphi_3 = a \vee b \vee c$ маємо $\varphi_3 = \overline{\overline{a \vee b \vee c}}$, що породжує схему (рис. 4.8).

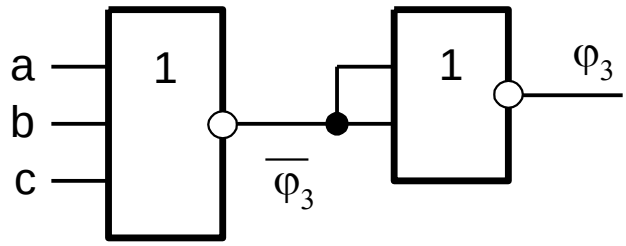


Рис. 4.8 – Реалізація елементарної диз'юнкції в базисі АБО-НЕ

7. Реалізація ДНФ в базисі АБО-НЕ.

Застосований підхід аналогічний методу реалізації КНФ в базисі І-НЕ: над кожним термом і над формулою загалом береться подвійна інверсія, а потім внутрішня інверсія для кожного терму перетворюється за законом де Моргана. Наприклад, для якоїсь булевої функції $y_1 = ab \vee cd$ маємо: $y_1 = \overline{\overline{ab} \vee \overline{cd}} = \overline{\overline{a} \vee \overline{b} \vee \overline{c} \vee \overline{d}}$, що породжує схему (рис. 4.9).

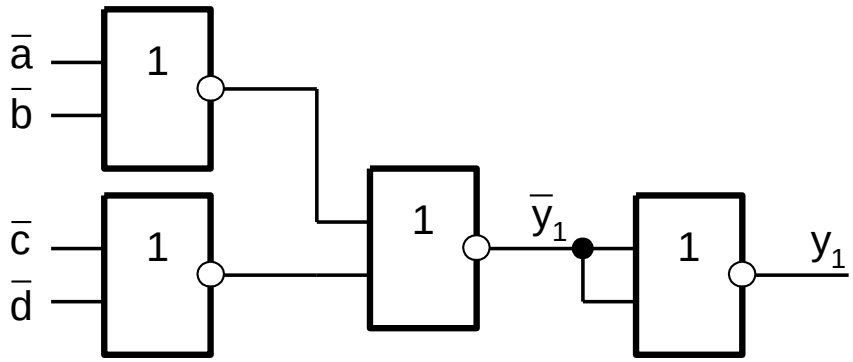


Рис. 4.9 – Реалізація ДНФ в базисі АБО-НЕ

Очевидно, що така схема буде повільною і доцільніше використовувати ДНФ зворотної функції. Наприклад, з карти Карно (рис. 4.5) маємо ДНФ

$\overline{y_3} = \bar{a}\bar{c} \vee \bar{a}c \vee a\bar{d} = \overline{\overline{a}\overline{c}} \vee \overline{\overline{a}c} \vee \overline{a\overline{d}}$. Отже, маємо $y_3 = \overline{\overline{\overline{a} \vee c} \vee \overline{a \vee \overline{c}} \vee \overline{\overline{a} \vee d}}$, що породжує схему (рис. 4.10). Як видно з формули, схема повинна включати чотири елементи, що підтверджується і рис. 4.10.

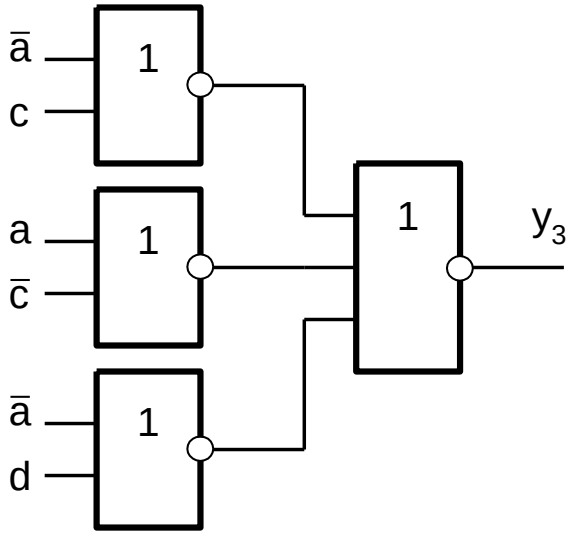


Рис. 4.10 – Реалізація ДНФ зворотної функції в базисі АБО-НЕ

8. Реалізація КНФ в базисі АБО-НЕ.

Реалізація заснована на застосуванні законів подвійного заперечення і де Моргана. Наприклад, для деякої функції $y_2 = (a \vee b)(c \vee d) = \Phi_1 \cdot \Phi_2$ маємо: $y_2 = \overline{\overline{\Phi_1 \cdot \Phi_2}} = \overline{\overline{\Phi_1} \vee \overline{\Phi_2}} = \overline{\overline{a \vee b} \vee \overline{c \vee d}}$, що породжує схему (рис. 4.11).

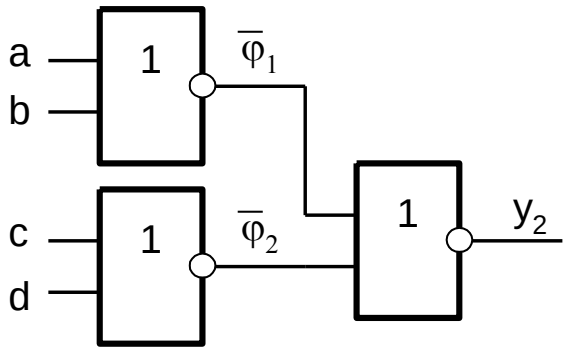


Рис. 4.11 – Реалізація КНФ в базисі АБО-НЕ

Результати, наведені в цьому підпункті, зведені в табл. 4.1.

4.2 Синтез схем з урахуванням обмежень базису

Елементи, що випускаються серійно, мають два важливі технологічні обмеження, що впливають на число елементів і швидкодію комбінаційних схем, що одержуються на їхній основі. Це такі обмеження:

1. Число входів S логічного елемента не перевищує восьми.

2. Здатність навантаження n_e не перевищує десяти.

Нагадаємо, що здатність навантаження логічного елемента визначає число входів, яке може бути пов'язане з виходом даного елемента. Це зумовлене тим, що логічний елемент характеризується кінцевим значенням струму, який може бути отриманий з його виходу. Водночас підключення до нього елемента також вимагає деякого значення струму для правильної роботи. Розглянемо вплив параметра S, прийнявши, що нормальна форма функції містить H термів і h-ий терм містить L_h букв.

Таблиця 4.1

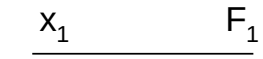
№	Функція	Базис	Реалізація
1	$F = ab$	І-НЕ	$F = \overline{\overline{ab}}$
		АБО НЕ	$F = \overline{\overline{a} \vee \overline{b}}$
2	$\Phi = a \vee b$	І-НЕ	$\Phi = \overline{\overline{a} \cdot \overline{b}}$
		АБО НЕ	$\Phi = \overline{\overline{a \vee b}}$
3	$y = ab \vee cd$	І-НЕ	$y = \overline{\overline{ab} \cdot \overline{cd}}$
		АБО НЕ	$y = \overline{\overline{a} \vee \overline{b} \vee \overline{c} \vee \overline{d}}$
4	$y = (a \vee b)(c \vee d)$	І-НЕ	$y = \overline{\overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{d}}$
		АБО НЕ	$y = \overline{\overline{a \vee b} \vee \overline{c \vee d}}$

1. Реалізація елементарної кон'юнкції в базисі І-НЕ.

Існує три області, які визначаються поєднанням параметрів L_h і S: $L_h = 1;\ 1 < L_h \le S;\ L_h > S$. Розглянемо особливості реалізації термів в цих областях, прийнявши для визначеності $S = 4$.

а) $L_h = 1$.

Розглянемо терм $F_1 = x_1$. Цей терм відповідає як диз'юнкції, так і кон'юнкції. Очевидно, що для реалізації схеми, що відповідає терму F_1, жодних логічних елементів не потрібно. У цьому разі схема є просто провідником (рис. 4.12а). У практичних випадках може виникнути необхідність у дублюванні змінної.

а) 

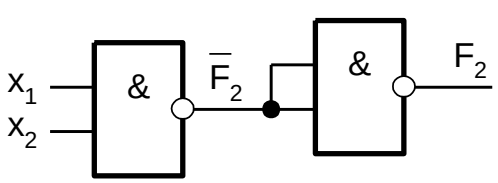
б) 

Рис. 4.12 – Реалізація кон'юнкції F_h за $L_h = 1$(А) і $L_h = 2$(Б)

б) $L_h = 2 \le S$.

Реалізація схеми ґрунтується на використанні законів подвійної інверсії ($\overline{\overline{A}} = A$) та ідемпотентності ($\overline{A \cdot A} = \overline{A}$). Наприклад, для терма $F_2 = x_1 x_2$ маємо $F_2 = \overline{\overline{F_2}} = \overline{\overline{x_1 x_2}} = \overline{\overline{F_2} \cdot \overline{F_2}}$, що відповідає схемі на рис. 4.12б.

в) $L_h = 3 > S$.

Якщо $L_h > S$, то для реалізації схеми використовується метод розбиття терма на підтерми, що охоплює такі етапи:

1. Розбити терм F_h на S підтермів $F_h^1, ..., F_h^S$ з приблизно однаковою кількістю букв.

2. Якщо підтерм F_h^i ($i = \overline{1, S}$) містить одну букву, то перетворення підтерма F_h^i закінчено.

3. Якщо підтерм $F_h^i (i = \overline{1, S})$ містить не більше S букв, то до нього застосовується операція подвійної інверсії, і на цьому перетворення підтерма F_h^i закінчується.

4. Якщо підтерм $F_h^i(i=\overline{1,S})$ містить більше S букв, то до підтерма застосовуються операції подвійної інверсії, і знаходиться розбиття підтерма на S підтермів $F_{hi}^1,...,F_{hi}^S$ з приблизно однаковою кількістю букв.

5. Пункти 2–4 виконуються циклічно доти, доки всі отримані підтерми не міститимуть не більше S букв.

6. Перетворені підтерми збираються в єдиний терм F_h.

Розбиття на приблизно однакову кількість літер необхідне для отримання схеми з максимально можливою швидкодією, як буде показано далі. Поки що можна зазначити, що рівномірне розбиття термів приводить до схем з мінімально можливим числом логічних рівнів. При цьому мінімізується загальна затримка поширення сигналу в схемі.

Розглянемо застосування цього методу до терма F3 = x1x2x3, при цьому будемо позначати підтерми буквами латинського алфавіту для більшої ясності викладу.

1. Терм $F_3 = x_1x_2x_3$ розбивається на $S=2$ частини: $a = x_1x_2$ і $b = x_3$.

2. Підтерм a містить дві літери, перетворення полягає в подвійній інверсії $a = \overline{\overline{x_1x_2}}$.

3. Підтерм b містить одну букву і не вимагає перетворення.

4. Після виконання п. 6 методу маємо: $F_3 = \overline{\overline{ab}} = \overline{\overline{x_1x_2x_3}}$, що відповідає схемі на рис. 4.13.

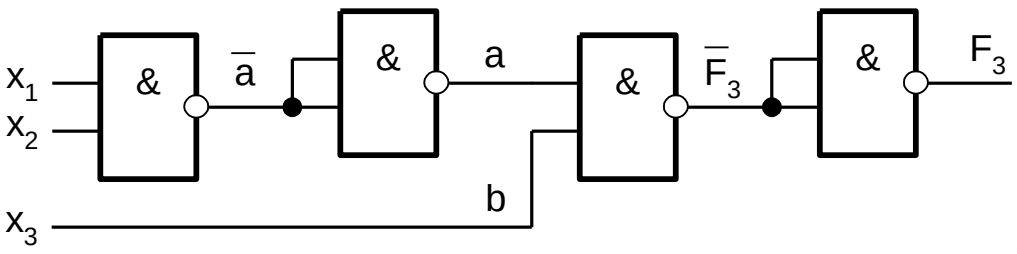


Рис. 4.13 – Реалізація кон'юнкції F_3

Розглянемо більш складний приклад, коли необхідно реалізувати терм $F_7 = x_1x_2x_3x_4x_5x_6x_7$ на елементах І-НЕ з двома входами (2 І-НЕ).

1. Розіб'ємо F_7 на дві частини $F_7 = ab$, де $a = x_1x_2x_3x_4$, $b = x_5x_6x_7$.

2. Перетворимо частину a до виду $a = \overline{\overline{x_1x_2x_3x_4}}$ і розіб'ємо на дві частини $c = x_1x_2$ і $d = x_3x_4$: $a = \overline{\overline{c \cdot d}}$.

3. Розіб'ємо $b = \overline{\overline{x_5x_6x_7}}$ на дві частини $e = x_5x_6$ і $f = x_7$: $b = \overline{\overline{e \cdot f}}$.

4. Перетворимо c: $c = \overline{\overline{x_1 \cdot x_2}}$ – перетворення c закінчено.

5. Перетворимо d: $d = \overline{\overline{x_3 \cdot x_4}}$ – перетворення d закінчено. Тепер маємо: $a = \overline{\overline{x_1x_2} \cdot \overline{x_3x_4}}$.

6. Перетворимо e: $e = \overline{\overline{x_5 \cdot x_6}}$ – перетворення e закінчено. Оскільки терм f не вимагає перетворень, то маємо $b = \overline{\overline{x_5x_6} \cdot x_7}$.

7. Тепер формуємо терм $F_7 = \overline{\overline{ab}}$ і отримаємо вираз $F_7 = \overline{\overline{x_1x_2} \cdot \overline{x_3x_4} \cdot \overline{x_5x_6} \cdot x_7}$, що відповідає схемі (рис. 4.14).

Процес розбиття терму F_7 на підтерми і синтез схеми показані графом на рис. 4.15, де знаком * позначений момент закінчення перетворення підтерма і початку побудови схеми.

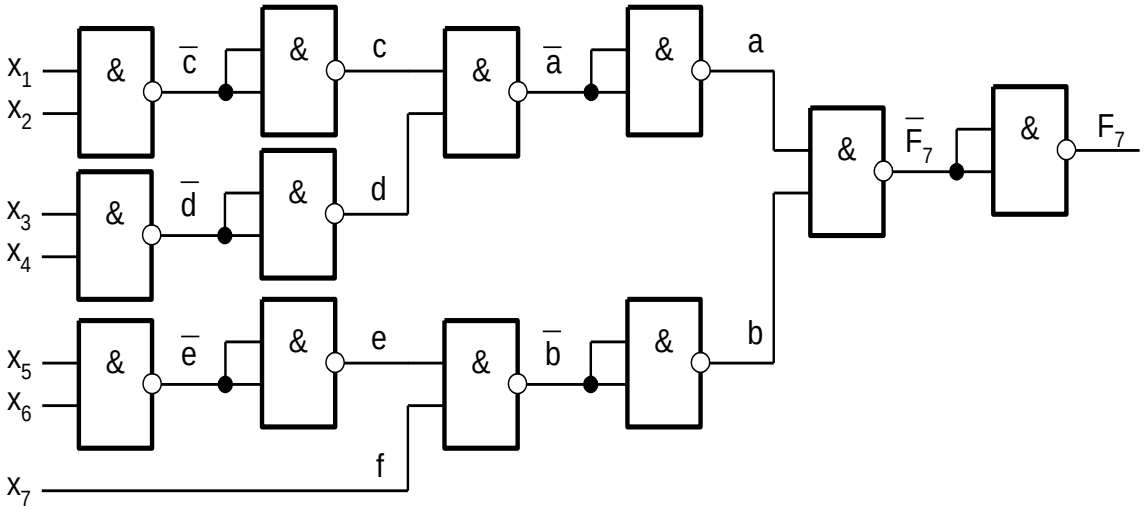


Рис. 4.14 – Реалізація кон'юнкції F_7 на елементах 2І-НЕ

Застосування розглянутого методу до терму з L_h буквами при використанні елементів з S входами завжди приводить до схем, які мають $n(L_h,S)_\wedge$ логічних елементів, де

$$n(L_h,S)_\wedge = 2\cdot\left\lceil\frac{L_h-1}{S-1}\right\rceil. \qquad (4.5)$$

Тут $\lceil a\rceil$ означає найближче ціле число, більше, ніж число a, якщо a – дробове число, або дорівнює a, якщо a – ціле число.

Застосування (4.5) за $L_h=7$, $S=2$ дає $n(7,2)=12$, що відповідає схемі на рис. 4.14. Зауважимо, що число вентилів $n(L_h,S)$ не залежить від способу розбиття терма F_h на підтерми. Модифікуємо розглянутий метод у такий спосіб, щоб один із підтермів завжди складався з однієї літери. Тоді перетворення терму F_7 породжує формулу

$F_7 = x_1\cdot x_2\cdot x_3\cdot x_4\cdot x_5(x_6x_7)$, що приводить до схеми на рис. 4.16.

Назвемо схемами $C1(C2)$ реалізацію терму на рис. 4.14, 4.16 і проаналізуємо їх. Обидві схеми $C1$ і $C2$ мають однакове число вентилів $n(L_h,S)=14$.

Зазначимо, що 14 дорівнює числу інверсій, які об’єднують більше за одну букву формули. На найдовшому шляху від вхідних змінних до вихідних в схемі $C1$ знаходиться $n(C1)=6$ вентилів, а $n(C2)=12$ вентилів.

Відтак, час перемикання схеми $C1$ $t(C1)=n(C1)$ $t_{ле}=6t_{ле}$, а $t(C2)=12t_{ле}$. Отже, за однакових апаратурних витрат схема $C1$ в два рази швидше, ніж схема $C2$. Оскільки при проєктуванні цифрових пристроїв прагнуть до отримання схем з максимально можливою швидкодією, то схема $C1$ є найкращою з усіх можливих реалізацій за $S=2$.

Оскільки спосіб розбиття терма на підтерми приводить до схем з різною швидкодією, то в розглянутому методі використовується розбиття на підтерми з приблизно однаковою кількістю букв. Як видно з порівняння

схем $C1$ і $C2$, такий підхід дає змогу оптимізувати швидкодію комбінаційної схеми.

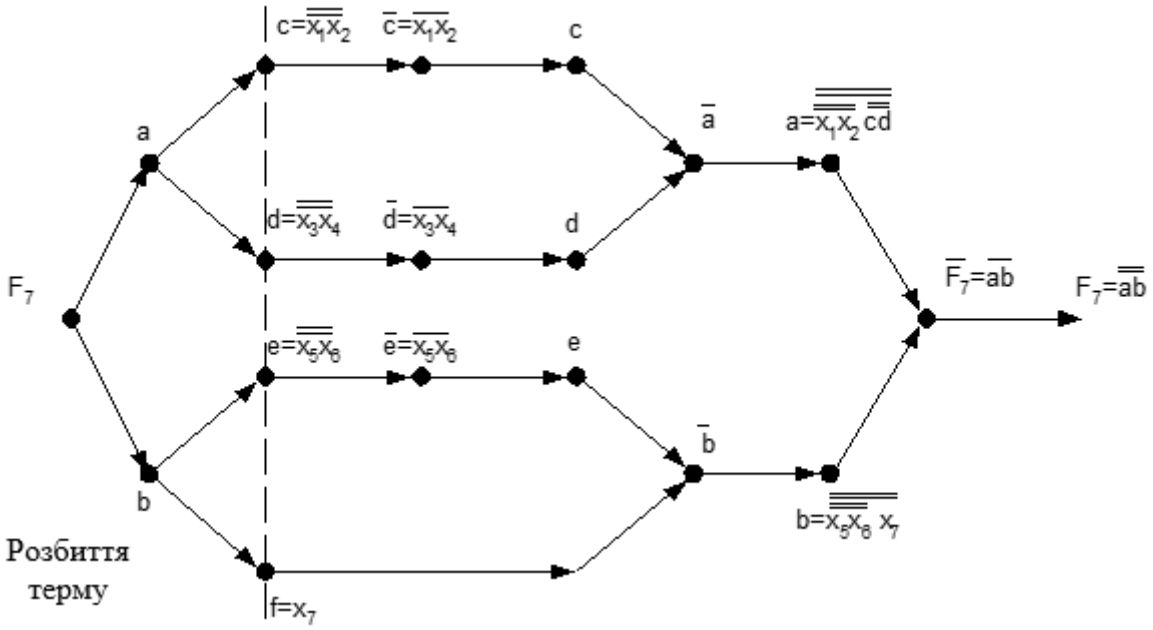


Рис. 4.15 – Графічне зображення процесу перетворення терму F_7 і синтезу схеми на елементах 2І-НЕ

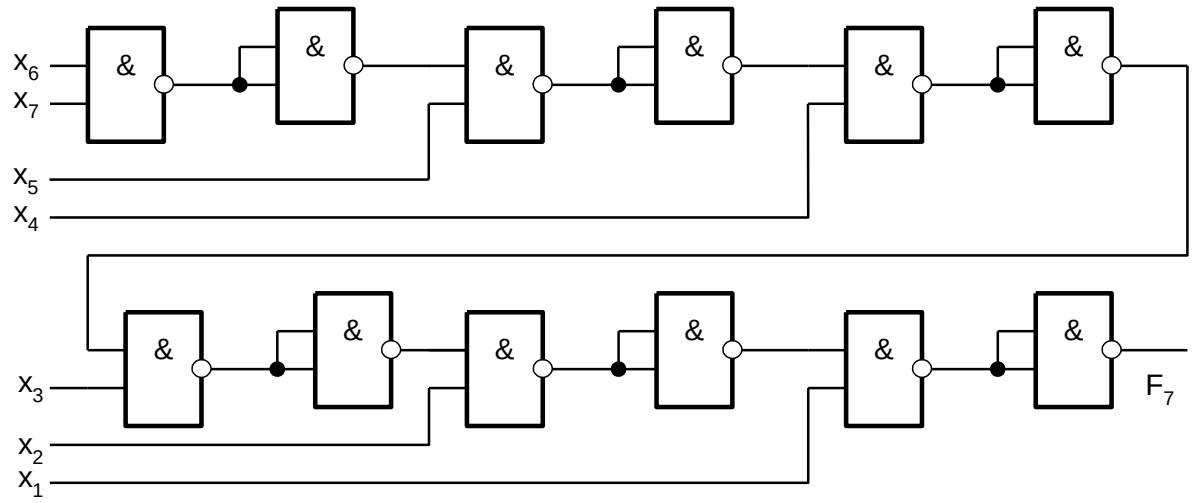


Рис. 4.16 – Нераціональна реалізація терму F_7 в базисі 2І-НЕ

2. Реалізація ДНФ в базисі І-НЕ.

У цьому разі до ДНФ застосовується подвійне заперечення і закон де Моргана. Наприклад, для якоїсь булевої функції дванадцяти змінних $y_1 = x_1x_2x_3 \vee x_4x_5x_6x_7 \vee x_8x_9x_{10} \vee x_{11}x_{12} = F_1 \vee F_2 \vee F_3 \vee F_4$ маємо формулу:

$y_1 = \overline{\overline{F_1 \vee F_2 \vee F_3 \vee F_4}} = \overline{\overline{F_1} \cdot \overline{F_2} \cdot \overline{F_3} \cdot \overline{F_4}}$. Отже, логічний вираз є кон'юнкцією, отож, реалізація комбінаційної схеми зводиться до послідовного перетворення кожного терму F_h і формули загалом за розглянутим раніше методом. Оскільки кожен з елементів формули входить в неї з інверсією, то на першому етапі перетворення відпадає необхідність в подвійній інверсії.

Розглянемо приклад застосування розглянутого методу. Наприклад, для перетворення формули для y_1 потрібно шість кроків. Розглянемо процес перетворення ДНФ y_1 без докладних коментарів ($S = 2$):

а) $\overline{y_1} = \overline{ab}$, де $a = \overline{\overline{F_1} \cdot \overline{F_2}}$, $b = \overline{\overline{F_3} \cdot \overline{F_4}}$;

б) $\overline{F_1} = \overline{cd}$, де $c = \overline{x_1 x_2}$, $d = x_3$, тобто $\overline{F_1} = \overline{\overline{x_1 x_2} x_3}$;

в) $\overline{F_2} = \overline{ef}$, де $e = \overline{x_4 x_5}$, $f = \overline{x_6 x_7}$, Тобто $\overline{F_2} = \overline{x_4 x_5} \cdot \overline{x_6 x_7}$;

г) $\overline{F_3} = \overline{gh}$, де $g = \overline{x_8 x_9}$, $h = x_{10}$, Тобто $\overline{F_3} = \overline{\overline{x_8 x_9} x_{10}}$;

д) $\overline{F_4} = \overline{x_{11} x_{12}}$;

е) отже, $y_1 = \overline{\overline{x_1 x_2} x_3 \cdot \overline{x_4 x_5} \cdot \overline{x_6 x_7} \cdot \overline{x_8 x_9} x_{10} \cdot \overline{x_{11} x_{12}}}$.

Отриманий вираз для y_1 відповідає схемі на рис. 4.17.

Визначимо формулу для підрахунку числа елементів в комбінаційній схемі. Оскільки кожен терм реалізується в інверсній формі, то для його реалізації необхідно $N(L_h, S)_\wedge - 1$ вентиль. Для функції, що має H термів, необхідно $N(L_h, S)_\wedge - 1$ вентиль. Отже, для реалізації ДНФ функції y з H термами $F_1, \ldots, F_H$, кожен з яких має L_h букв, де $L_h \le L$, потрібно $N_{ДНФ\wedge}$ вентилів:

$$N_{ДНФ\wedge} = \sum_{h=1}^{H} (N(L_k, S)_\wedge - 1) + N(H, S)_\wedge - 1. \qquad (4.7)$$

Використовуючи (4.6) і (4.7), отримаємо шукану формулу.

$$N_{ДНФ\wedge} = 2\sum_{h=1}^{H}\left\lceil\frac{L_h - 1}{S-1}\right\rceil + \left\lceil\frac{H-1}{S-1}\right\rceil - H - 1. \qquad (4.8)$$

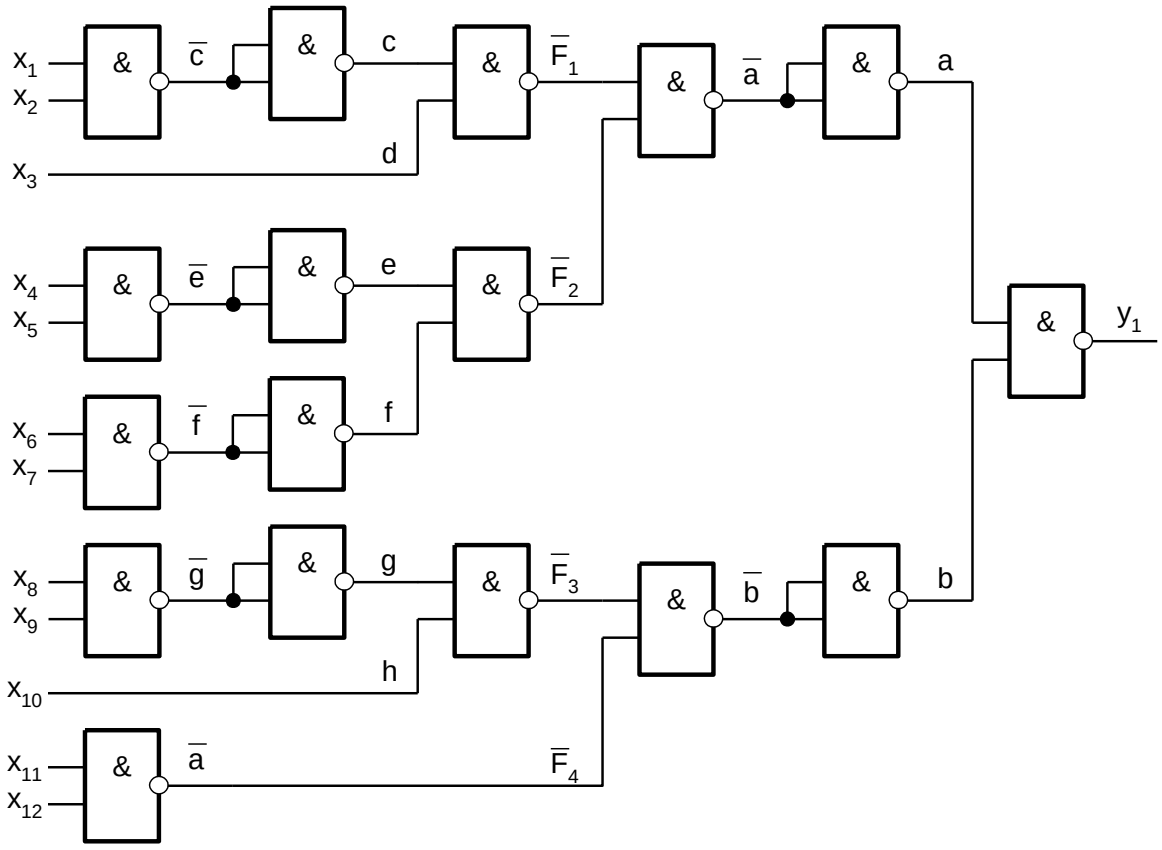


Рис. 4.17 – Реалізація функції y_1 в базисі 2І-НЕ

Наприклад, для функції y_1 маємо:

$$N_{ДНФ\wedge} = 2\left(2\cdot\left\lceil\frac{3-1}{2-1}\right\rceil + \left\lceil\frac{4-1}{2-1}\right\rceil + \left\lceil\frac{2-1}{2-1}\right\rceil\right) + 2\cdot\left\lceil\frac{4-1}{2-1}\right\rceil - 4 - 1 = 17,$$

що відповідає числу вентилів у схемі $С3$ (рис. 4.17). Для схеми $С3$ маємо $n(С3) = 6$, тобто $t(С3) = 6t_{ле}$. Зазначимо, що це число збігається з максимальним числом заперечень, загальних більш ніж для однієї літери перетвореної формули для y_1. Формула (4.8) може бути використана для попередньої оцінки числа вентилів у схемі. Це дає змогу визначити складність схеми без її реалізації.

3. Реалізація елементарної диз’юнкції в базисі І-НЕ.

Реалізація схеми заснована на використанні законів подвійної інверсії і де Моргана, а також на застосуванні розглянутого методу перетворення кон’юнкції.

Наприклад, для терму $\varphi_4 = x_1 \vee x_2 \vee x_3 \vee x_4$ за $S = 2$ маємо $\varphi_4 = \overline{\overline{x_1 \vee x_2 \vee x_3 \vee x_4}} = \overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}} = \overline{\overline{x_1} \cdot \overline{x_2}} \cdot \overline{\overline{x_3} \cdot \overline{x_4}}$, що приводить до схеми на рис. 4.18.

Якщо в схемі немає джерел, які формують як прямі, так і інверсні значення сигналів, то для реалізації схеми необхідно L_h вентилів, які формують заперечення вхідних сигналів. Отже, для реалізації диз’юнкції необхідна така кількість вентилів із заданими обмеженнями:

$$N(L_h, S)_{\vee} = 2 \cdot \left\lceil \frac{L_h - 1}{S - 1} \right\rceil - 1 + L_h. \qquad (4.9)$$

Знак $\vee$ у формулі (4.9) підкреслює, що в основі формули знаходиться диз’юнкція. Наприклад, для φ_4 маємо $N(L_h, S)_{\vee} = 9$ логічних елементів, що відповідає схемі на рис. 4.18.

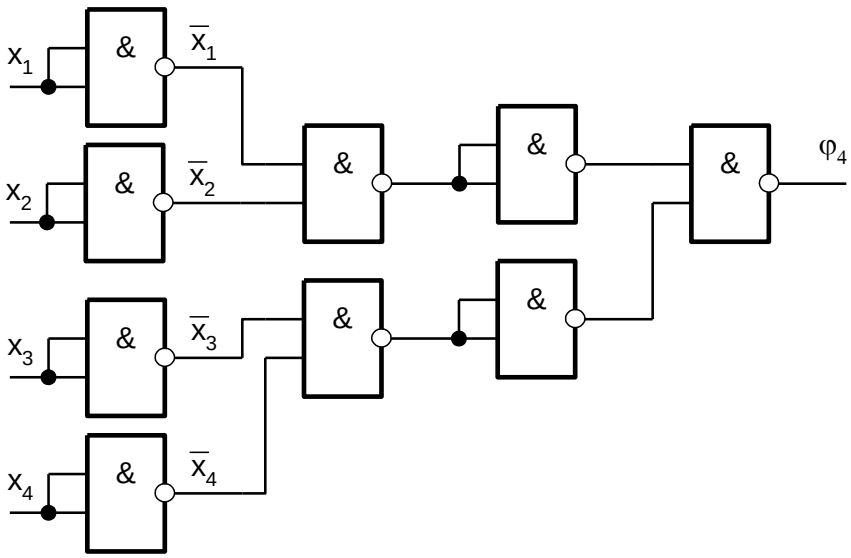


Рис. 4.18 – Реалізація диз’юнкції φ_4 в базисі 2І-НЕ

4. Реалізація КНФ в базисі І-НЕ.

Реалізація схеми заснована на застосуванні законів подвійної інверсії до кожного терму Φ_h і КНФ загалом, а також на застосуванні закону де Моргана до кожного терма функції. Наприклад, для булевої функції $y_2 = (x_1 \vee x_2)(x_3 \vee x_4 \vee x_5)(x_6 \vee x_7) = \Phi_1\Phi_2\Phi_3$ можна отримати формулу: $y_2 = \overline{\overline{x_1 \vee x_2} \cdot \overline{x_3 \vee x_4 \vee x_5} \cdot \overline{x_6 \vee x_7}} = \overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} \cdot \overline{x_5} \cdot \overline{x_6} \cdot \overline{x_7}}$. Перебіг подальшого перетворення зводиться до перетворення кожного терма і функції загалом. Для функції y_2 маємо $y_2 = \overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} \cdot \overline{x_5} \cdot \overline{x_6} \cdot \overline{x_7}}$, що приводить до схеми (рис. 4.19), де інвертори, що використовуються для формування сигналів $\overline{x_1},...,\overline{x_7}$, не показані. Принцип побудови цієї комбінаційної схеми очевидний. Схема починається від внутрішніх кон'юнкцій, а потім відбувається підйом по формулі, як по деякому дереву.

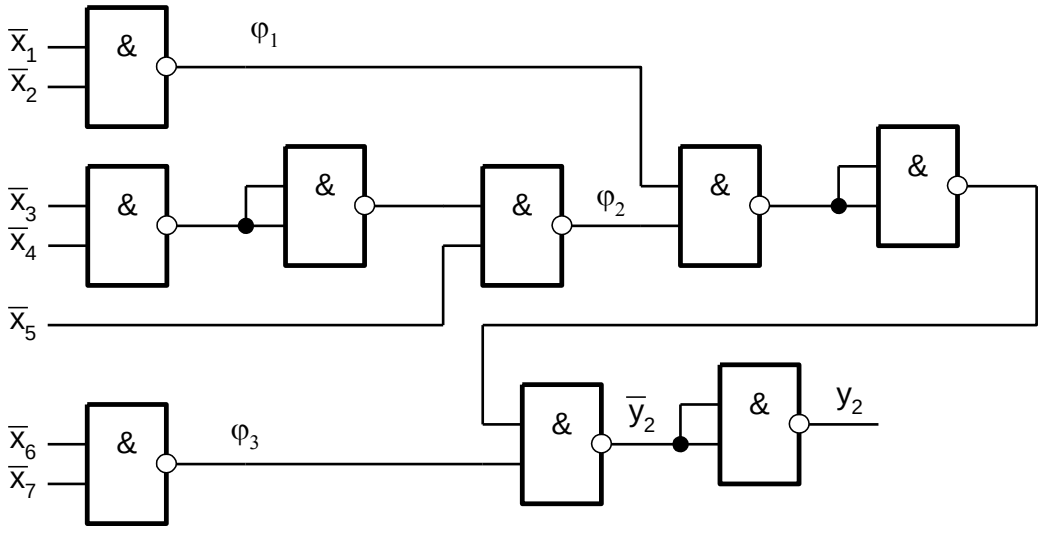


Рис. 4.19 – Реалізація КНФ y_2 в базисі 2І-НЕ

Для реалізації кожного терма Φ_h потрібно $N(L_h,S)_\vee - L_h$ елементів, для реалізації кон'юнкції $\Phi_1 \ldots \Phi_H$ необхідно $N(H,S)_\vee$ логічних елементів, для реалізації інверсних значень входів необхідно L логічних елементів. Отже, для реалізації КНФ необхідно

$$N_{КНФ\wedge} = 2\sum_{h=L}^{H}\left\lceil\frac{L_h - 1}{S-1}\right\rceil - H + 2\left\lceil\frac{H-1}{S-1}\right\rceil + L \qquad (4.10)$$

логічних елементів І-НЕ з S входами. Наприклад, для КНФ y_2 за $S=2$ маємо $N_{КНФ\vee}=16$, що відповідає рис. 4.19 з урахуванням $L=7$ інверторів для вхідних сигналів.

Число рівнів схеми може бути зменшено за рахунок використання КНФ зворотної функції. Нагадаємо, що зворотна функція вже несе в собі внутрішню інверсію. Число вентилів $N_{\overline{КНФ\wedge}}$ в цьому разі визначається виразом:

$$N_{\overline{КНФ}\wedge} = N_{\overline{КНФ}\wedge} - 1. \qquad (4.11)$$

5. Реалізація схем в базисі АБО-НЕ.

Методи синтезу схем у базисі АБО-НЕ аналогічні методам синтезу в базисі І-НЕ. При цьому диз'юнкція Φ_h реалізується також як кон'юнкція F_h в базисі І-НЕ, КНФ як ДНФ, F_h як Φ_h, ДНФ як КНФ. Розглянемо, наприклад, реалізацію комбінаційної схеми для функції, заданої КНФ $y_3 = (x_1 \vee x_2 \vee x_3)\cdot(x_4 \vee x_5)\cdot(x_6 \vee x_7 \vee x_8 \vee x_9) = \Phi_1\Phi_2\Phi_3$, у базисі 2АБО-НЕ:

а) $y_3 = \overline{\overline{\Phi_1\Phi_2\Phi_3}} = \overline{\overline{\Phi_1} \vee \overline{\Phi_2} \vee \overline{\Phi_3}} = \overline{\overline{\overline{\Phi_1} \vee \overline{\Phi_2} \vee \overline{\Phi_3}}}$;

б) $\overline{\Phi_1} = \overline{x_1 \vee x_2 \vee x_3} = \overline{\overline{x_1 \vee x_2 \vee x_3}}$;

в) $\overline{\Phi_2} = \overline{x_4 \vee x_5}$;

г) $\overline{\Phi_3} = \overline{x_6 \vee x_7 \vee x_8 \vee x_9} = \overline{\overline{x_6} \vee \overline{x_7} \vee \overline{x_8} \vee \overline{x_9}}$;

д) $y_3 = \overline{\overline{x_1 \vee x_2 \vee x_3} \vee \overline{x_4 \vee x_5} \vee \overline{x_6} \vee \overline{x_7} \vee \overline{x_8} \vee \overline{x_9}}$.

Як підсумок проведених перетворень будується схема (рис. 4.20).

У табл. 4.2 наведені формули, що дають змогу визначити число елементів у комбінаційних схемах, реалізованих в різних базисах.

Розглянемо вплив фактору навантажувальної спроможності на кінцевий вигляд комбінаційної схеми. Якщо число вентилів, пов'язаних з

виходом логічного елемента, перевищує n_e, то використовуються два основні способи підвищення здатності навантаження:

1. Дублювання виходу вентиля.

2. Дублювання вентиля.

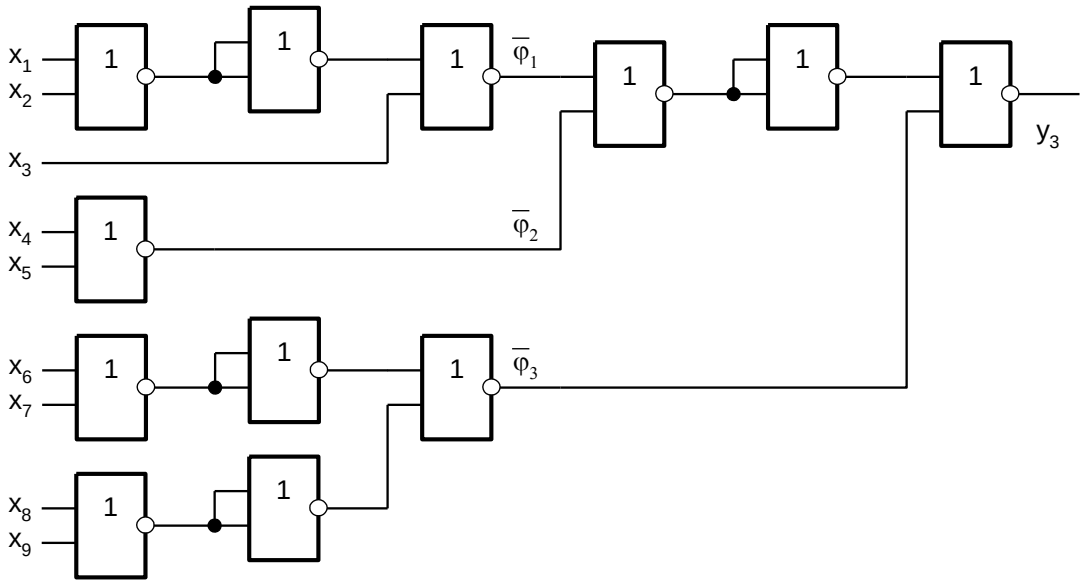


Рис. 4.20 – Реалізація КНФ y_3 в базисі 2АБО-НЕ

Наприклад, вентиль І-НЕ має $n_e = 3$, а терм $F_2 = \overline{ab}$ входить в ДНФ п’яти функцій, тобто до виходу вентиля підключені входи п’яти вентилів ($I_e = 5$). Метод дублювання виходу вентиля породжує схему, зображену на рис. 4.21а, а метод дублювання вентиля – схему, зображену на рис. 4.21б).

Перший з цих методів зменшує швидкодію комбінаційної схеми за рахунок збільшення числа рівнів. Другий метод дає змогу зберегти швидкодію, але підвищує вимоги до характеристик вхідних сигналів a і b, здатність навантаження яких також скінченна.

Розглянуті в цьому підпункті методи синтезу комбінаційних схем можна назвати методами «розкладання, інвертування і з’єднання», що відбиває процес синтезу КС з урахуванням обмежень елементного базису.

Методи синтезу КС в базисах І-НЕ і АБО-НЕ аналогічні, тому в подальшому будемо розглядати тільки базис І-НЕ.

Таблиця 4.2

Визначення числа елементів комбінаційних схем

№	Функція	Базис	Число елементів у схемі
1	F_h	І-НЕ	$N(L_h,S)=2\left\lceil\frac{L_h-1}{S-1}\right\rceil$
2	Φ_h	АБО НЕ	
3	F_h	АБО НЕ	$N(L_h,S)-1+L_h$
4	Φ_h	І-НЕ	
5	ДНФ y	І-НЕ	$\sum_{h=1}^{H} N(L_h,S)+N(H,S)-H-1$
6	КНФ y	АБО НЕ	
7	ДНФ y	АБО НЕ	$\sum_{h=1}^{H} N(L_h,S)-H+N(H,S)+L=N(y,S)$
8	КНФ y	І-НЕ	
9	ДНФ $\overline{y}$	АБО НЕ	$N(y,S)-1$
10	КНФ $\overline{y}$	І-НЕ	

4.3 Синтез типових комбінаційних схем

В обчислювальній техніці використовуються деякі типові рішення для реалізації систем булевих функцій, яким відповідають типові КС. До таких схем передусім належать дешифратори, мультиплексори, суматори, компаратори. Ці схеми використовуються при синтезі операційних автоматів, тому розглянемо їх докладніше, орієнтуючись на базис І-НЕ.

1. Дешифратор.

Дешифратором (DC) називається КС, що перетворює вхідний S-розрядний двійковий код в унітарний t-розрядний код, де $t=2^S$.

Таблиця істинності тривхідного DC (табл. 4.3) містить входи x_1, x_2, x_3 і виходи $y_0,\ldots,y_7$. З табл. 4.3 випливає, що кожна вихідна функція y_i ($i=\overline{0,7}$) є мінтермом, тобто $y_i=F_i$. Переважно, дешифратори мають спеціальний вхід, який називається входом вибірки кристала (CS), що керується сигналом y_{CS}. Зазвичай виходи DC і вхід CS є інверсними.

Якщо $y_{CS}=1$, то на всіх виходах DC встановлюються одиничні рівні. Якщо $y_{CS}=0$, то на виході, відповідному вхідній комбінації, встановлюється рівень $y_i=0$, а на всіх інших виходах – поодинокі рівні. Отже, таблиця істинності для реального DC матиме вигляд, наведений у табл. 4.4. З цієї таблиці формується така система функцій:

$$\overline{y_0}=\overline{y_{CS}\cdot\overline{x_1}\cdot\overline{x_2}\cdot\overline{x_3}};$$

$$\overline{y_1}=\overline{y_{CS}\cdot\overline{x_1}\cdot\overline{x_2}\cdot x_3};$$

$$\ldots$$

$$\overline{y_7}=\overline{y_{CS}\cdot x_1\cdot x_2\cdot x_3};$$

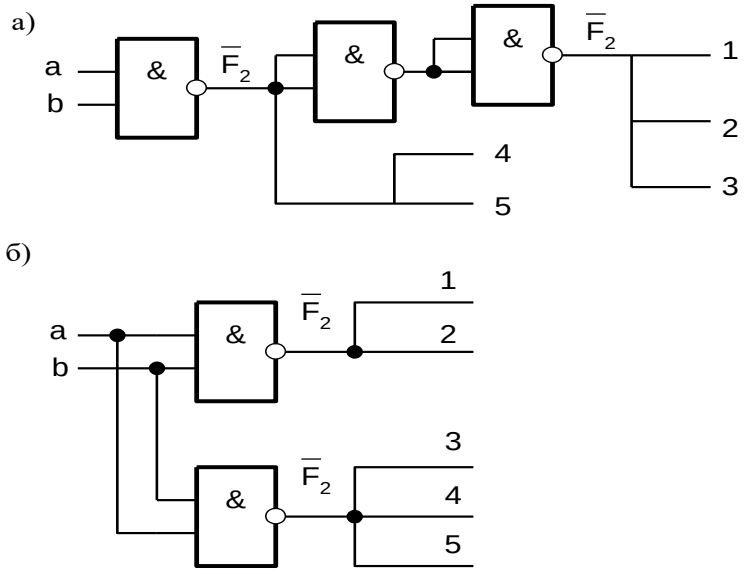


Рис. 4.21 – Реалізація схем з дублюванням виходу вентиля (а) і дублюванням вентиля (б)

Отримана система функцій є основою для синтезу схеми DC (рис. 4.22а), умовне позначення якого наведено на рис. 4.22б.

На рис. 4.22а всі вхідні провідники пронумеровані і зведені в загальну шину. Такий підхід дає змогу спрощувати схеми, оскільки замість найменувань змінних пишуться номери відповідних провідників, що робить зображення схеми менш громіздким.

Таблиця 4.3

x_1	x_2	$x3$	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Таблиця 4.4

$\overline{y_{CS}}$	x_1	x_2	x_3	$\overline{y_0}$	$\overline{y_1}$	$\overline{y_2}$	$\overline{y_3}$	$\overline{y_4}$	$\overline{y_5}$	$\overline{y_6}$	$\overline{y_7}$
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0
1	*	*	*	1	1	1	1	1	1	1	1

2. Мультиплексори.

Мультиплексором називається пристрій, що передає інформацію від декількох джерел до одного приймача. Мультиплексор (MX) має S керуючих входів, $L = 2^S$ інформаційних входів, вхід вибірки $\overline{y_{CS}}$ і один (або два парофазні) вихід. Таблиця істинності MX з $S = 2$ приведена в табл. 4.5, в якій прийнято, що Z_1, Z_2 – керуючі входи, y – вихід MX, $x_1,\ldots,x_4$ – інформаційні входи. Якщо $\overline{y_{CS}} = 0$, то інформація, що передається з входу, визначається комбінацією сигналів.

З табл. 4.5 маємо таке рівняння для вихідної функції:

$$y = y_{CS} \cdot \overline{Z_1} \cdot \overline{Z_2} \cdot x_1 \vee y_{CS} \cdot \overline{Z_1} \cdot Z_2 \cdot x_2 \vee y_{CS} \cdot Z_1 \cdot \overline{Z_2} \cdot x_3 \vee y_{CS} \cdot Z_1 \cdot Z_2 \cdot x_4 .$$

Це рівняння породжує після деяких перетворень схему мультиплексора (рис. 4.23а), що має умовне позначення (рис. 4.23б).

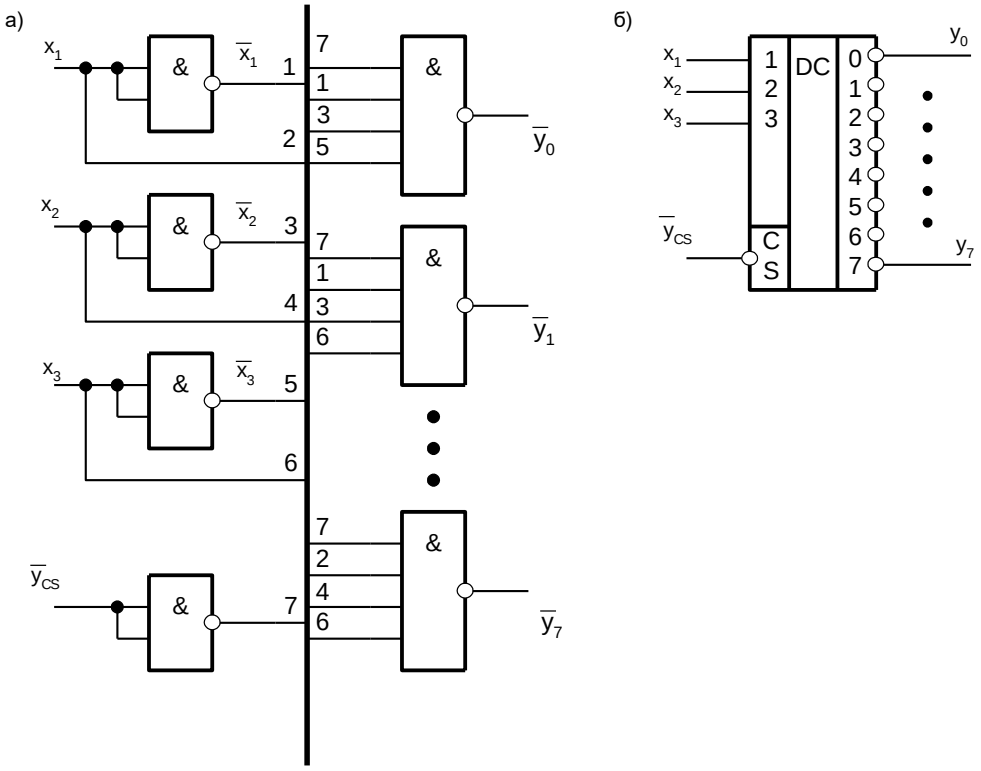


Рис. 4.22 – Схема (а) і умовне позначення (б) тривхідного дешифратора з входом вибірки

Таблиця 4.5

$\overline{y_{CS}}$	Z_1	Z_2	y
0	0	0	x_1
0	0	1	x_2
0	1	0	x_3
0	1	1	x_4
1	*	*	0

3. Суматори.

Суматором називається схема, яка визначає результат складання двох n-розрядних двійкових чисел $S = A + B$, що має $n+1$ розряд. Старший розряд суми називається переносом. Основою багаторозрядних суматорів є однорозрядні повні суматори (рис. 4.24а), поєднання яких (рис. 4.24б) дає багаторозрядний суматор. Тут p – вихід переносу, c – вхід переносу, C_{in} – вхід переносу з попередньої секції суматора, C_{out} – вихідний перенос. Схема на рис. 4.24б називається суматором з

послідовним переносом. Це найбільш повільний з усіх можливих паралельних суматорів. Для збільшення швидкодії існують різні методи прискореного перенесення, які виходять за рамки цієї книги.

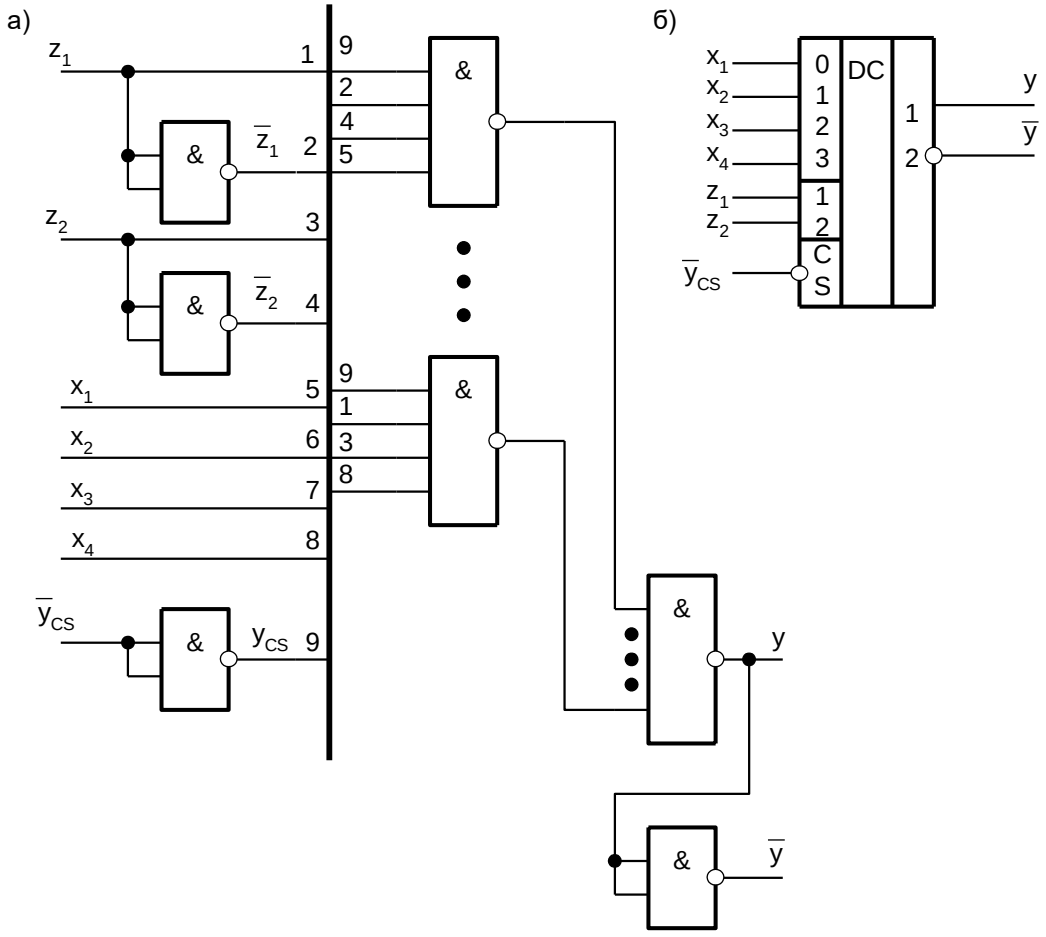


Рис. 4.23 – Схема (а) і умовне позначення (б) чотиривхідного мультиплексора

Розглянемо таблицю істинності для однорозрядного суматора (табл. 4.6) і побудуємо карти Карно для функцій S і p (рис. 4.25).

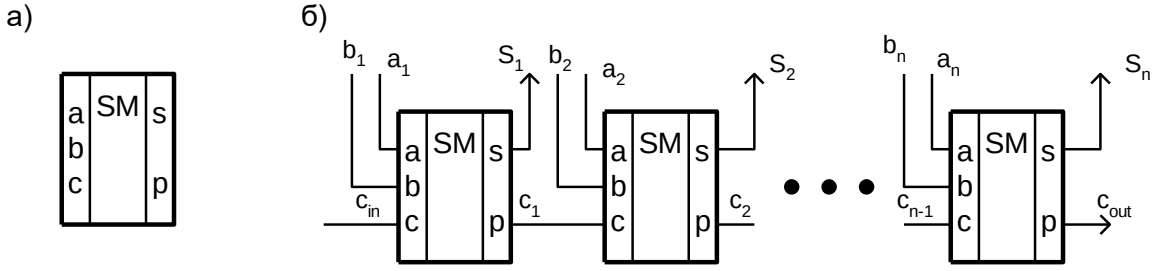


Рис. 4.24 – Умовне позначення однорозрядного суматора (а) й організація багаторозрядного суматора (б)

Таблиця 4.6

Таблиця істинності

a	*b*	*c*	*S*	*P*
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

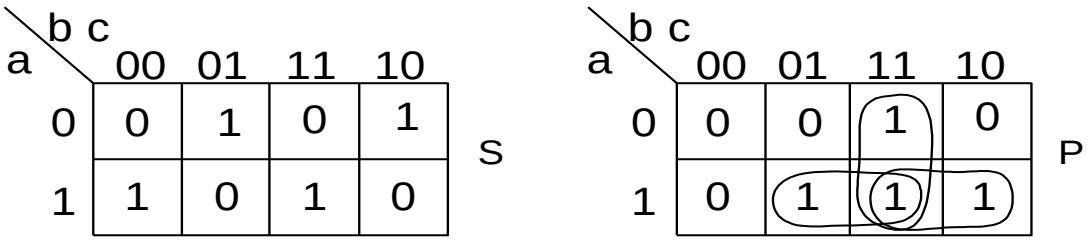


Рис. 4.25 – Карти Карно для однорозрядного суматора

Як видно з рис. 4.25, функція S НЕ мінімізується, тобто

$$S = \bar{a}\bar{b}c \vee \bar{a}b\bar{c} \vee a\bar{b}\bar{c} \vee abc. \qquad (4.12)$$

Функція p може бути мінімізована, що дає формулу

$$P = ac \vee ab \vee bc. \qquad (4.13)$$

Застосування законів подвійної інверсії і де Моргана до виразів (4.12)–(4.13) приводить до канонічних рівнянь схеми однорозрядного суматора: $S = \overline{\overline{\bar{a}\cdot\bar{b}\cdot c}\cdot\overline{\bar{a}\cdot b\cdot\bar{c}}\cdot\overline{a\cdot\bar{b}\cdot\bar{c}}\cdot\overline{a\cdot b\cdot c}}$, $p = \overline{\overline{ac}\cdot\overline{ab}\cdot\overline{bc}}$, за якими далі будується канонічна схема (рис. 4.26).

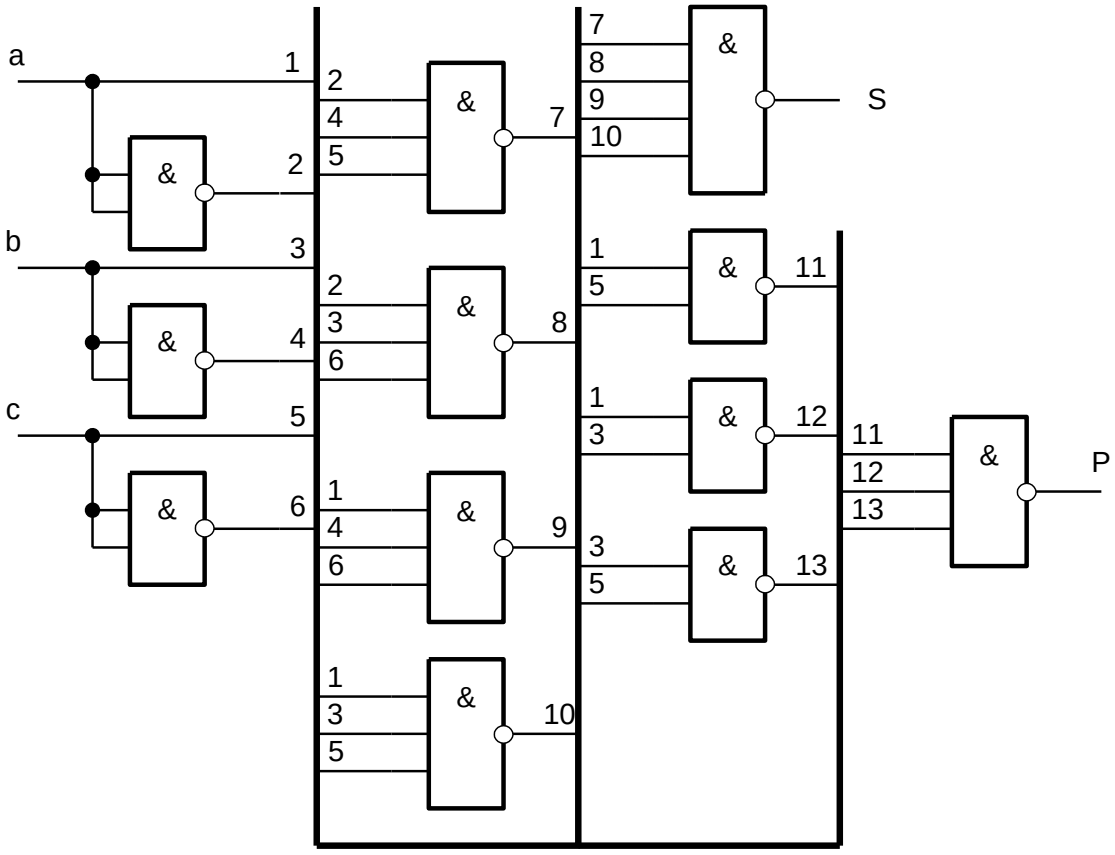


Рис. 4.26 – Канонічна схема однорозрядного повного суматора

Схема має три рівні, тобто її швидкодія $t_{SM} = 3t_{ле}$, і є найбільш швидкодіючою з можливих варіантів. Ця схема має низку недоліків, важливих в умовах масового виробництва:

1. Схема використовує як прямі, так і інверсні значення вхідних змінних, що зумовлює появу великої кількості провідників. Переважно при побудові практичних схем використовують прямі виходи регістрів

2. У схемі використовуються елементи з різним числом входів, що призводить до деяких труднощів при виготовленні шаблонів.

Оскільки на практиці в реальних схемах багаторозрядних суматорів використовуються різні методи прискореного перенесення, то число рівнів у схемі однорозрядного суматора не має істотного впливу на час виконання складання. Розглянемо метод синтезу суматора, який використовується на практиці.

З аналізу табл. 4.6 випливає, що $S=1$, якщо число одиниць у вхідному наборі непарне, тобто $S=a\oplus b\oplus c$. З табл. 4.6 маємо ДДНФ функції $p=\bar{a}bc\vee a\bar{b}c\vee ab\bar{c}\vee abc=ab\vee c(a\bar{b}\vee\bar{a}b)=ab\vee c(a\oplus b)$.

Покажемо, що $a\bar{b}=a\overline{ab}=a(\bar{a}\vee\bar{b})=a\bar{a}\vee a\bar{b}=0\vee a\bar{b}=a\bar{b}$. Аналогічно, маємо $\bar{a}b=b\overline{ab}$. Тоді для функції p можна виконати такі перетворення:

$$p=ab\vee c(a\overline{ab}\vee b\overline{ab})=\overline{\overline{ab}\vee\overline{c(a\overline{ab}\vee b\overline{ab})}}=\overline{\overline{ab}\cdot\overline{c\overline{a\overline{ab}}\cdot\overline{b\overline{ab}}}}=\overline{F_1}\cdot\overline{F_2},$$

з чого випливає, що $a\bar{b}\vee\bar{a}b=\overline{a\overline{ab}\cdot b\overline{ab}}$. Цей вираз показує спосіб реалізації функції $a\oplus b$, при цьому схема для функції S є послідовним з'єднанням схем для функцій $S_1=a\oplus b$ і $S=S_1\oplus C$. Схема однорозрядного суматора, заснована на розглянутих перетвореннях виразів (4.12)–(4.13), наведена на рис. 4.27.

Схема на рис. 4.27 не має недоліків, властивих канонічній схемі, крім того, ціна за Квайном канонічної схеми дорівнює 28, а для практичної схеми – 20, в канонічній схемі 12 вентилів, а в практичній – 10. Природно, практична схема є більш повільною: $t_S=6t_{ле}$ і $t_p=5t_{ле}$.

Цей приклад синтезу показує, що для досягнення заданих на практиці критеріїв часто потрібні глибокі перетворення систем булевих функцій, що описують закон функціонування комбінаційної схеми.

4. Компаратори.

Компаратором називається комбінаційна схема, що виконує операції порівняння двох двійкових чисел. Очевидно, що порівняння можуть бути досить різноманітними (наприклад, одне число більше, ніж інше число). Природно, що такі комбіновані перевірки можна отримати з простих перевірок. З огляду на це стандартний компаратор реалізує тільки три види відносин: $A>B$, $A<B$ і $A=B$. Таблиця істинності дворозрядного компаратора (табл. 4.7) містить чотири входи $a_1a_2(A)$, $b_1b_2(B)$ і три виходи $y_{12}(A>B)$, $y_2(A<B)$ і $y_3(A=B)$.

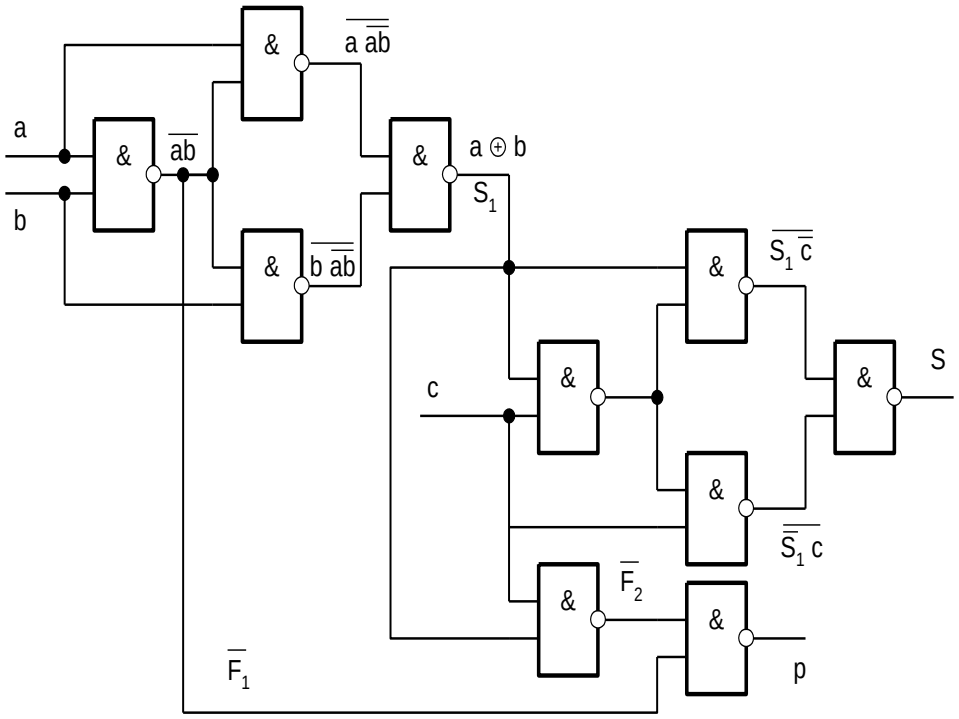


Рис. 4.27 – Схема однорозрядного суматора, що використовується на практиці

Таблиця 4.7

Таблиця істинності

a_1	a_2	b_1	b_2	y_1	y_2	y_3	a_1	a_2	b_1	b_2	y_1	y_2	y_3
0	0	0	0	0	0	1	1	0	0	0	1	0	0
0	0	0	1	0	1	0	1	0	0	1	1	0	0
0	0	1	0	0	1	0	1	0	1	0	0	0	1
0	0	1	1	0	1	0	1	0	1	1	0	1	0
0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	1	0	1	0	0	1	1	1	0	1	1	0	0
0	1	1	0	0	1	0	1	1	1	0	1	0	0
0	1	1	1	0	1	0	1	1	1	1	0	0	1

Функції y_1 і y_2 мають досить складний вид і характеризуються нерегулярністю. Для спрощення схеми на практиці використовують такі підходи:

1. Компаратор реалізує тільки дві функції, наприклад y_1 і y_3, а функція $y_2 = \overline{y_1} \cdot \overline{y_3}$.

2. Порівняння $A > B$ проводиться послідовно, починаючи зі старших розрядів. Якщо, наприклад, $a_1 > b_1$, то формується ознака y_1, якщо $a_1 = b_1$, то формується сигнал, що робить можливим порівняння розрядів a_2 і b_4.

Очевидно, що відношення $a_i = b_i$ задається функцією $f_i = a_i \cdot b_i \vee \overline{a_i} \cdot \overline{b_i}$, а відношення $a_i > b_i$ задається функцією $\varphi_i = a_i \cdot \overline{b_i}$. Очевидно, що $A = B$, якщо кон'юнкція всіх функцій f_i дорівнює одиниці. Порівняння для розрядів a_2 і b_2 відбувається, якщо $\varphi_1 = 0$ і $f_1 = 1$, в цьому разі необхідно сформувати функцію дозволу $r_1 = \overline{\varphi_1} f_1$. Для вирішення порівняння розрядів a_3, b_3 необхідно сформувати функцію $r_2 = \overline{\varphi_2} f_2$. У загальному випадку для вирішення порівняння розрядів a_{i+1}, b_{i+1} необхідно сформувати функцію $r_i = \overline{\varphi_i} f_i$. Функція $y_1 = \varphi_1 \vee \varphi_2 \vee ... \vee \varphi_I$, де I – розрядність числа. Виходячи з цих міркувань, маємо таку систему рівнянь (4.14). На рис. 4.28 приведена схема компаратора, побудована за системою (4.14) за $I = 3$.

$$
\begin{aligned}
& f_i = a_i b_i \vee \overline{a}_i \overline{b}_i = \overline{\overline{a_i b_i} \cdot \overline{\overline{a}_i \cdot \overline{b}_i}}, (i = 1, \ldots, I); \\
& \varphi_1 = a_1 \overline{b}_1 = \overline{\overline{a_1 \overline{b}_1}}; \\
& r_i = \overline{\overline{\varphi}_1 f_1 r_{i-1}}, (i = 1, \ldots, I-1); \\
& \varphi_i = r_{i-1} a_i \overline{b}_i = \overline{\overline{r_{i-1} a_i \overline{b}_i}}, (i = 2, \ldots, I); \\
& y_1 = \overline{\overline{f_1 \vee f_2 \vee \ldots \vee f_I}} = \overline{\overline{f}_1 \cdot \overline{f}_2 \cdot \ldots \cdot \overline{f}_I}; \\
& y_2 = \overline{\overline{y}_1 \cdot \overline{y}_3}; \\
& y_3 = \overline{\overline{f_1 \cdot f_2 \cdot \ldots \cdot f_I}}.
\end{aligned}
\qquad (4.14)
$$

Цей приклад показує, що для досягнення регулярності структури схеми, тобто повторюваності її окремих частин, синтез проводиться не за таблицею істинності системи булевих функцій, а на основі аналізу принципів роботи пристрою.

5. Зсувник.

Зсувом називається пристрій, що здійснює зрушення інформації на певне число розрядів праворуч або ліворуч. Зсувник широко використовується при організації послідовної обробки паралельної інформації. При операції зсуву праворуч молодші розряди записуються на місце старших, при зсуві ліворуч – навпаки.

Таблиця істинності зсувника за $I = 4$ (число розрядів) може бути представлена у спосіб, наведений у табл. 4.8. У цій таблиці сигнал $y_1 = 1$ ініціює зрушення на один розряд праворуч, $y_2 = 1$ – на один розряд ліворуч, $y_3 = 1$ означає відсутність зсуву.

Таблиця 4.8

y_1	y_2	y_3	a_1	a_2	a_3	a_4	зсув
1	0	0	0	a_1	a_2	a_3	праворуч
0	1	0	a_2	a_3	a_4	0	ліворуч
0	0	1	a_1	a_2	a_3	a_4	немає

Очевидно, що в кожен момент часу тільки один з сигналів y_j ($j = \overline{1,3}$) може дорівнювати одиниці.

З табл. 4.8 можна отримати систему рівнянь для зсувника. У нашому випадку ця система має вигляд:

$$\begin{aligned} a'_1 &= y_2 a_2 \vee y_3 a_1, \\ a'_2 &= y_1 a_1 \vee y_2 a_3 \vee y_3 a_2, \\ a'_3 &= y_1 a_2 \vee y_2 a_4 \vee y_3 a_3, \\ a'_4 &= y_1 a_3 \vee y_3 a_4. \end{aligned} \qquad (4.15)$$

Тут виходи із зсувом позначені для визначеності як $a'_i (i = \overline{1,4})$. Зауважимо, що в системі (4.15) рівняння для крайніх розрядів різняться, а для інших розрядів збігаються за структурою, відрізняючись тільки індексами, що говорить про регулярність структури схеми. Застосувавши

до рівнянь (4.15) закони подвійної інверсії і закони де Моргана, отримаємо схему із зсувом (рис. 4.29).

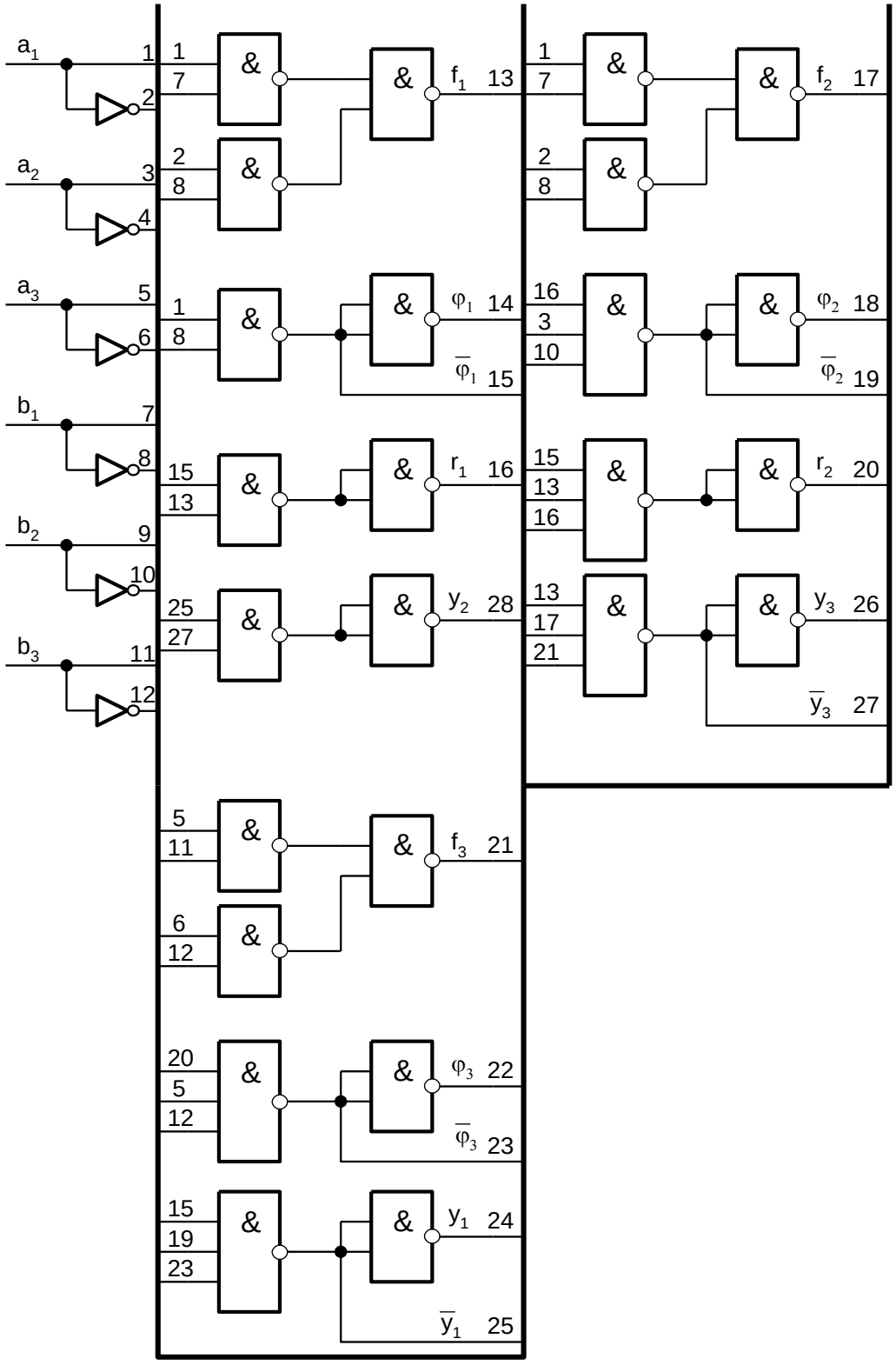


Рис. 4.28 – Схема компаратора за $I = 3$

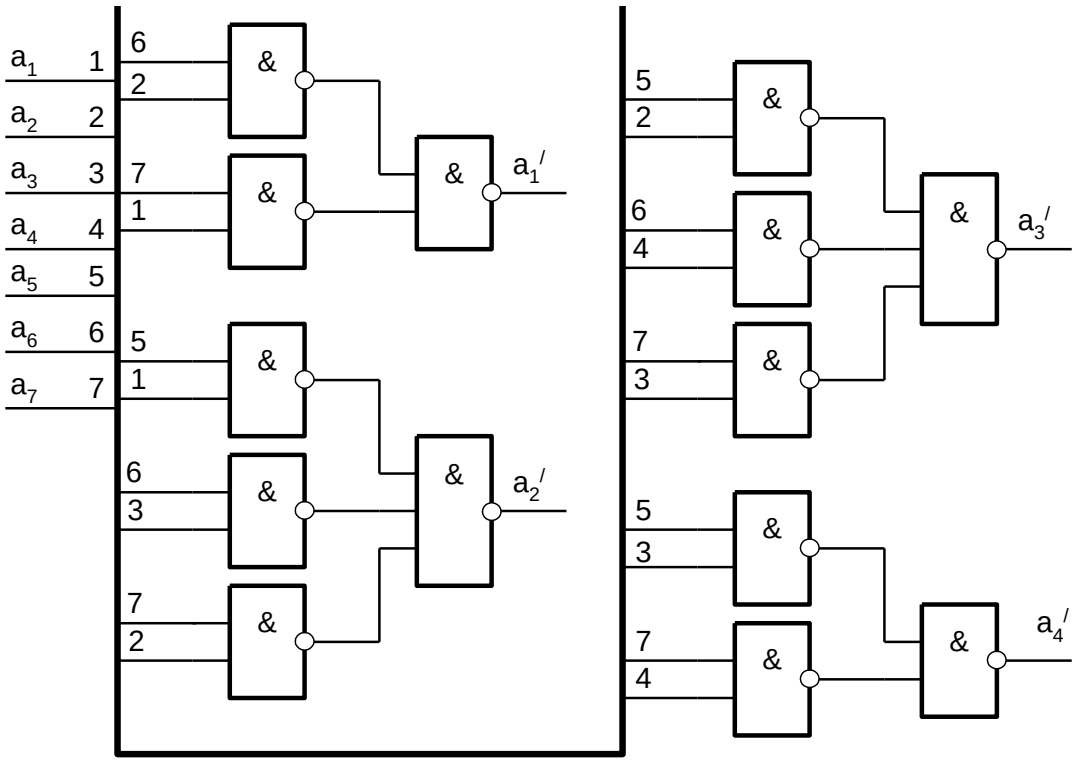


Рис. 4.29 – Схема зсувника паралельного коду

Деякі з розглянутих в цьому підпункті типових елементів є ефективним засобом реалізації комбінаційних схем. До цих елементів належать дешифратори і мультиплексори. Розглянемо методи синтезу КС на цих елементах.

4.4 Синтез схем на дешифраторах і мультиплексорах

З таблиці істинності дешифратора (табл. 4.3) випливає, що він реалізує всі 2^L мінтерми функції L булевих змінних. Природно, що диз'юнкція визначених термів є ДНФ деякої функції. Розглянемо методи синтезу КС для дешифраторів з інверсними виходами і входом вибірки (рис. 4.22).

Приклад 4.2. Синтезувати схему компаратора, що формує ознаки $A > B$ і $A = B$, якщо $A, B \in \{0,1,2,3\}$.

Нехай $y_1 = 1$, якщо $A > B$ і $y_2 = 1$, якщо $A = B$, а числа A і B задані наборами змінних $a_1 a_2$ і $b_1 b_2$ відповідно. Функції y_1 і y_2 задані картою

Карно (рис. 4.30), в клітинах якої записані y_1 і y_2, якщо відповідна функція дорівнює одиниці, і нуль – в іншому разі.

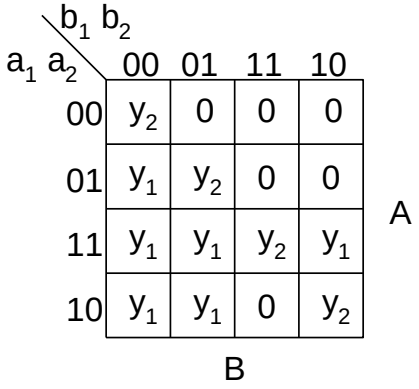


Рис. 4.30 – Карта Карно для компаратора

З карти Карно (рис. 4.30) маємо два рівняння:

$$y_1 = F_4 \vee F_8 \vee F_9 \vee F_{12} \vee F_{13} \vee F_{14} = \overline{\overline{F_4} \cdot \overline{F_8} \cdot \overline{F_9} \cdot \overline{F_{12}} \cdot \overline{F_{13}} \cdot \overline{F_{14}}};$$

$$y_2 = F_0 \vee F_5 \vee F_7 \vee F_{15} = \overline{F_0} \cdot \overline{F_5} \cdot \overline{F_7} \cdot \overline{F_{15}}.$$

Схема компаратора реалізується на стандартному дешифраторі з двома входами і чотирма виходами (рис. 4.31), на вхід $\overline{CS}$ якого подається сигнал ініціалізації операції порівняння $\overline{y_{CP}}$.

За виконання умови

$$L > S, \tag{4.16}$$

де L – число змінних в реалізованій системі функцій $Y = y(x)$, S – число входів дешифратора, схема може бути реалізована на одному DC. Для реалізації KC за виконання (4.16) використовується каскадне з'єднання дешифраторів, яке проводиться за такою методикою:

1. На першому рівні схеми знаходиться $n_1 = \left\lceil \frac{H}{q} \right\rceil$ дешифраторів, де $H = 2^L$, $q = 2^s$. На їхні входи подаються булеві змінні $x_k, \ldots, x_L$, де $k = L - S + 1$.

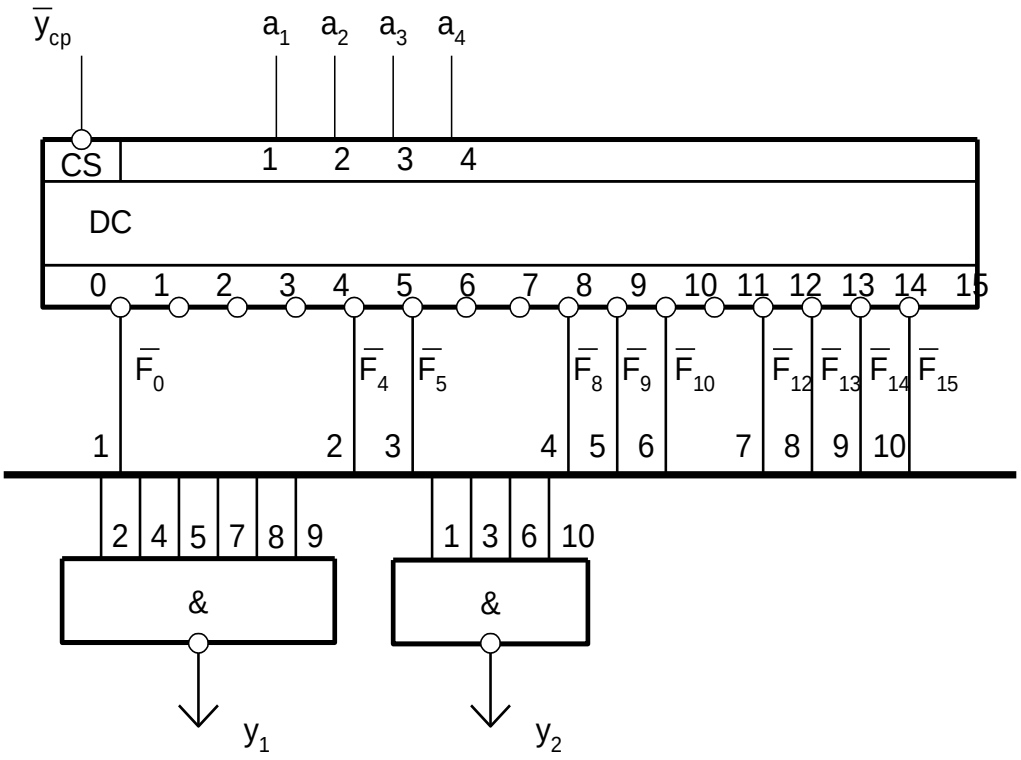


Рис. 4.31 – Реалізація схеми компаратора на дешифраторі і схемах І-НЕ

2. На другому рівні схеми знаходиться $n_2 = \left\lceil \frac{n_1}{q} \right\rceil$ дешифраторів, на входи яких подаються змінні $x_j, \ldots, x_{k-l}$, де $j = L - 2S + 1$.

3. Процедура триває доти, доки на i-му рівні буде тільки один дешифратор ($n_i = 1$).

4. Вихід DC рівня $l-1$, відповідний набору змінних, що дорівнює k, подається на вхід вибірки DC рівня l з номером $k-1$.

Розглянемо застосування цієї процедури до функції y_1 (табл. 4.9) при використанні дешифратора з числом входів $S = 4$.

На першому рівні схеми знаходиться $n_1 = \left\lceil \frac{32}{4} \right\rceil = 8$ дешифраторів, на входи яких подаються змінні x_4, x_5. Виходи цих дешифраторів відповідають термам системи функцій, в даному випадку – функції y_1. Очевидно, що перший DC цього рівня відповідає набору $\langle x_1 x_2 x_3 \rangle = \langle 000 \rangle$, другий – $\langle 001 \rangle$ і так далі.

На другому рівні схеми знаходиться $n_2 = \left\lceil \frac{n_1}{4} \right\rceil = 2$ дешифратори, на входи яких подаються змінні x_2, x_3. Перший дешифратор цього рівня відповідає $x_1 = 0$, другий – $x_1 = 1$.

Таблиця 4.9

x_1	x_2	x_3	x_4	x_5	y_1	x_1	x_2	x_3	x_4	x_5	y_1
0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	1	1
0	0	0	1	0	0	1	0	0	1	0	0
0	0	0	1	1	1	1	0	0	1	1	1
0	0	1	0	0	0	1	0	1	0	0	1
0	0	1	0	1	1	1	0	1	0	1	0
0	0	1	1	0	0	1	0	1	1	0	1
0	0	1	1	1	0	1	0	1	1	1	1
0	1	0	0	0	1	1	1	0	0	0	1
0	1	0	0	1	0	1	1	0	0	1	0
0	1	0	1	0	1	1	1	0	1	0	1
0	1	0	1	1	1	1	1	0	1	1	1
0	1	1	0	0	0	1	1	1	0	0	0
0	1	1	0	1	1	1	1	1	0	1	1
0	1	1	1	0	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1

На третьому рівні схеми знаходиться один дешифратор, на вхід якого подається змінна x_1. Як видно зі схеми (рис. 4.32), дешифратор третього рівня вироджується у вентиль І-НЕ. Однак це справедливо тільки для нашого конкретного прикладу. За $L = 6$ на третьому рівні буде знаходитися звичайний дешифратор.

Число дешифраторів у схемі визначається за формулою

$$n_{DC} = \sum_{i=1}^{I} (n_i - a_i), \qquad (4.17)$$

де a_i – число дешифраторів i-го рівня, всі виходи яких відповідають термам, на яких функція дорівнює нулю.

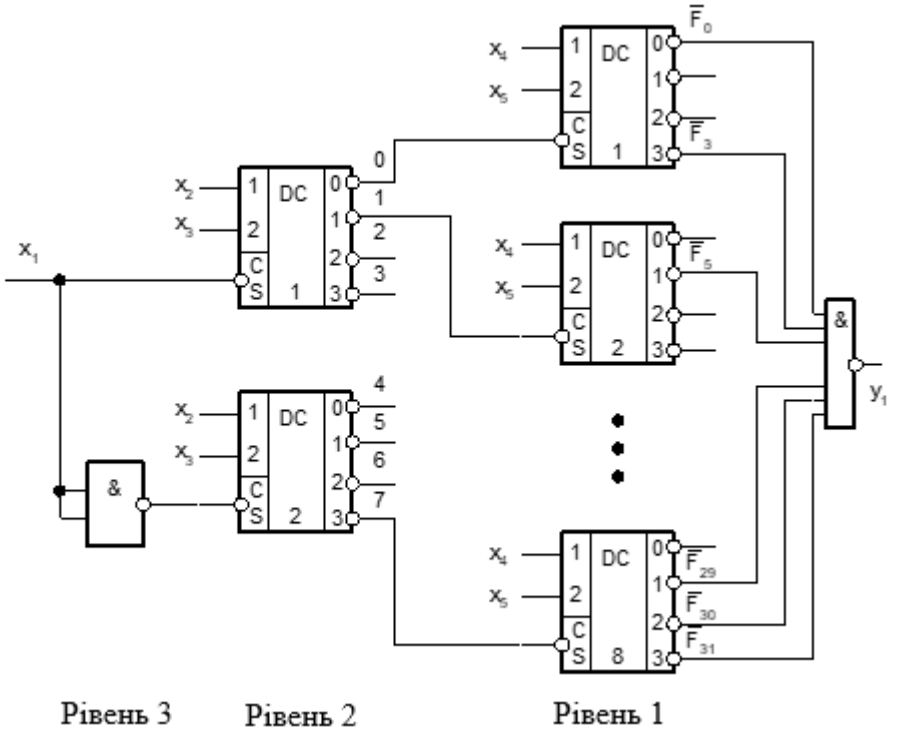


Рис. 4.32 – Реалізація функції y_1 на дешифраторі

Число рівнів I визначається виразом:

$$I = \left\lceil \frac{L}{S} \right\rceil. \tag{4.18}$$

Наприклад для булевої функції y з формули (4.18) маємо число $I = 3$, що відповідає схемі (рис. 4.31).

Природно, що якщо число H_0 мінтермів функції буде перевищувати число $S_\wedge$ входів кінцевого елемента І-НЕ, то для реалізації функції потрібно $n_\wedge = \left\lceil \frac{H_0 - 1}{S_\wedge - 1} \right\rceil$ логічних елементів, що збільшує загальний час перемикання схеми.

Розглянемо методи реалізації функцій на мультиплексорах. Нехай функція y_2 задана таблицею істинності (табл. 4.10). Очевидно, ця функція може бути реалізована тривіальним способом на мультиплексорі MX з чотирма керуючими входами (рис. 4.33). За тривіальної реалізації функції на керуючі входи MX подаються всі вхідні змінні $x_1, \ldots, x_L$, а на h-ий

інформаційний вхід подається значення функції на h-му вхідному наборі ($h = 1, \ldots, H = 2^L$).

Таблиця 4.10

x_1	x_2	x_3	x_4	y_2	x_1	x_2	x_3	x_4	y_2
0	0	0	0	0	1	0	0	0	0
0	0	0	1	1	1	0	0	1	0
0	0	1	0	1	1	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0

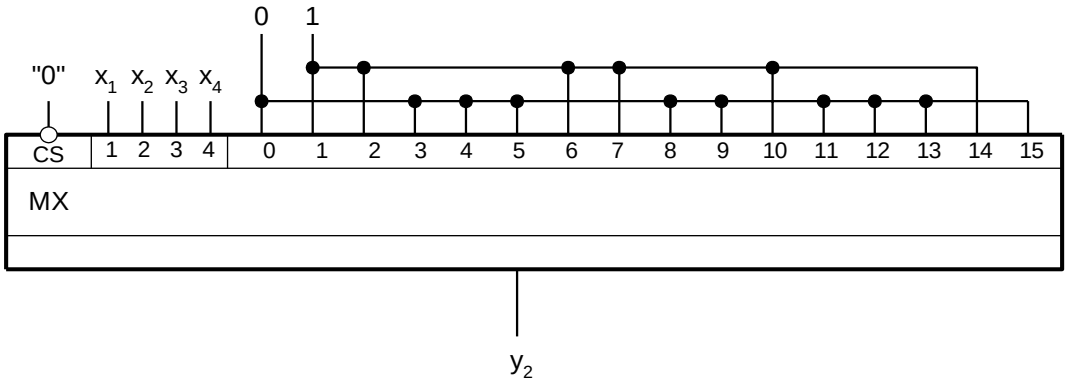


Рис. 4.3 – Тривіальна реалізація функції y_2 на мультиплексорі

Число керуючих входів мультиплексора можна зменшити до $L-1$. Порівняємо значення змінної x_4 і функції y_2 для різних наборів $\langle x_1, x_2, x_3 \rangle$. Для набору 000 – $y_2 = x_4$, 001 – $\overline{x_4}$, 010 – 0, 011 – 1, 100 – 0, 101 – $\overline{x_4}$, 110 – 0, 111 – $\overline{x_4}$. Відтак, на інформаційні входи можна подавати елементи множини $\{0,1, x_L, \bar{x}_L\}$, а на керуючі – $\{x_1, \ldots, x_{L-1}\}$. Реалізація схеми для функції y_2, заснована на цьому принципі, наведена на рис. 4.34. У цій схемі інформаційні входи мультиплексора відповідають векторам 000*, 001*, ..., 111*.

У будь-якій таблиці істинності можна виділити дві підтаблиці. Одна з них підтаблиця T_1, відповідна $x_1 = 0$, і друга підтаблиця T_2, відповідна

$x_1 = 1$. У табл. 4.10 підтаблиця T_1 утворюється рядками з 1-го по 8-ий, а рядки 9–16 утворюють підтаблицю T_2. З огляду на цей факт число інформаційних входів мультиплексора може бути зменшено до 2^{L-2}, а число керуючих входів – до $L-4$. На рис. 4.35 приведена реалізація функції y_2, заснована на цьому підході.

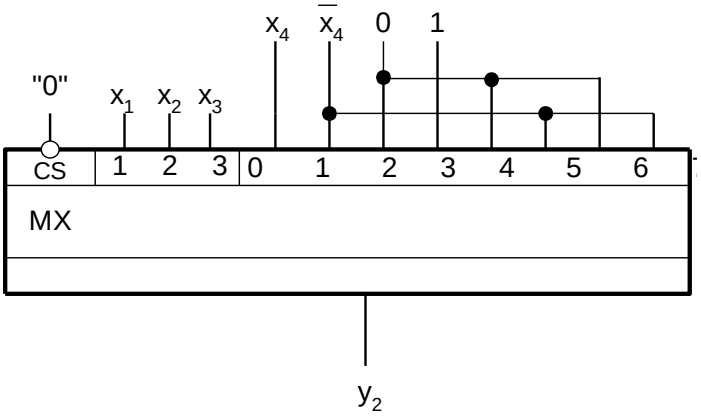


Рис. 4.34 – Реалізація функції y_2 на мультиплексорі за $S = L-1$

У цій схемі на виході першого *MX* формуються значення функції для рядків 1–8, позначені як y_2^1, на виході другого *MX* – для рядків 9–16, позначені як y_2^2. Для формування остаточного значення функції y_2 використовується вентиль АБО.

Очевидно, що для управління цими мультиплексорами був використаний одновхідний *DC* з входом x_4 і виходами x_4, $\overline{x_4}$. Аналогічно можна використовувати *DC* на будь-яке число входів. У цьому разі на входи мультиплексорів подаються змінні з множини $\{0,1,\bar{x}_L, x_L\}$ (інформаційні входи) і з множини $\{x_i,\ldots,x_{L-1}\}$ (керуючі входи), де $i = L-S$, а на входи *DC* подаються змінні з множини $\{x_1,\ldots,x_{i-1}\}$.

Наприклад, при реалізації функції y_1 (табл. 4.9) на мультиплексорах з $S = 2$ (число керуючих входів) і дешифраторах з двома входами схема буде мати вигляд, як на рис. 4.36.

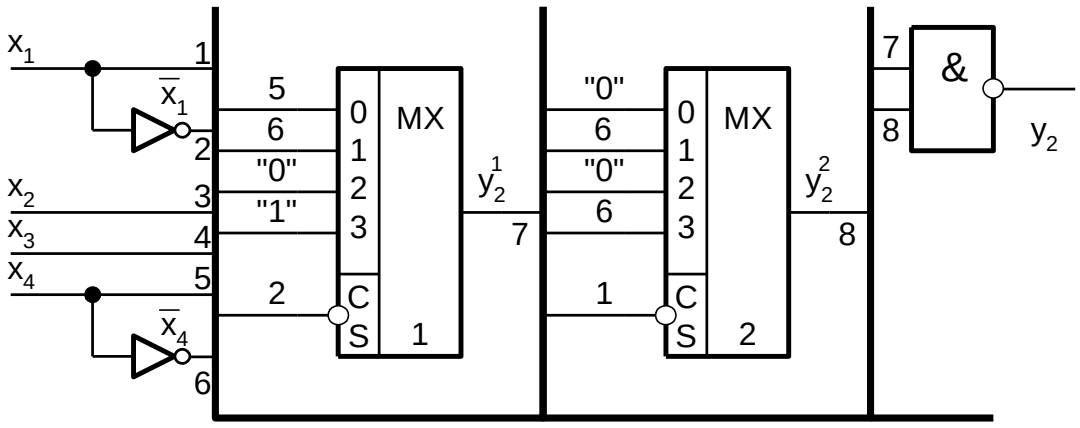


Рис. 4.35 – Реалізація функції y_2 на мультиплексорах за $S = L - 2$

Кожен мультиплексор $MX_1 - MX_4$ схеми на рис. 4.36 відповідає одній четвертій вихідної таблиці істинності, тобто підтаблиці $T_1 - T_4$. Підтаблиця T_1 відповідає набору $x_1 x_2 = 00$, $T_2 - 01$, $T_3 - 10$ і $T_4 - 11$. Багаторівнева схема, подібна до схеми на рис. 4.36, може бути побудована тільки на мультиплексорах.

При побудові логічної схеми на мультиплексорах за $L > S + 1$ використовується така методика.

1. На першому рівні логічної схеми знаходиться $n_1 = \left\lceil 2^{L-1} / 2^S \right\rceil$ мультиплексорів, при цьому мультиплексор M_i відповідає i-ій підтаблиці вихідної таблиці істинності T_i, де $i = 2^{L-1-S} = n_1$. На інформаційні входи мультиплексорів першого рівня подаються елементи множини $\{0, 1, \bar{x}_L, x_L\}$, на керуючі – $\{x_{L-S}, \ldots, x_{L-1}\}$.

2. На другому рівні схеми знаходиться $n_2 = \left\lceil n_1 / 2^S \right\rceil$ мультиплексорів, на інформаційні входи яких подаються виходи мультиплексорів першого рівня, а на керуючі входи подаються елементи множини $\{x_{L-2S}, \ldots, x_{L-S-1}\}$.

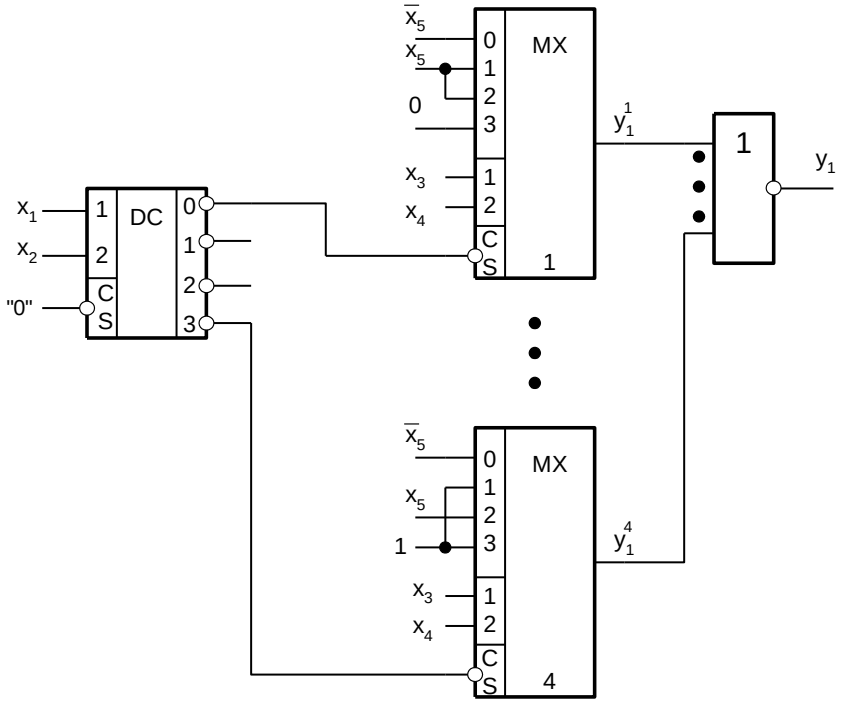


Рис. 4.36 – Реалізація функції y_1 на дешифраторах і мультиплексорах

3. На j-му рівні знаходиться $n_j = \left\lceil n_{j-1} / 2^s \right\rceil$ мультиплексорів, на інформаційні входи яких подаються виходи мультиплексорів $(j-1)$-го рівня. Процедура завершується, коли на рівні J знаходиться лише один мультиплексор. Число рівнів схеми визначається відношенням

$$J = \left\lceil (L-1) / S \right\rceil. \tag{4.19}$$

Застосування цієї методики до булевої функції y_1 приводить до комбінаційної схеми (рис. 4.37).

Для цієї схеми $J = \lceil 4/2 \rceil = 2$, $n_1 = \left| 16/4 \right| = 4$ і $n_2 = \left| 4/4 \right| = 1$. Зауважимо, що отримана схема на мультиплексорах (рис. 4.37) є більш швидкодіючою, ніж схема на мультиплексорах і дешифраторах (рис. 4.36).

Число мультиплексорів у схемі визначається виразом

$$n_{MX} = \sum_{j=1}^{J} (n_j - a_j), \tag{4.20}$$

де a_j – число мультиплексорів j-го рівня, на всі інформаційні входи яких подаються нулі. Згідно з (4.19) і (4.20) маємо, що схема на

мультиплексорах для функції y_1 повинна мати $J = 2$ рівні і $n_{MX} = 4+1=5$, що відповідає схемі на рис. 4.37.

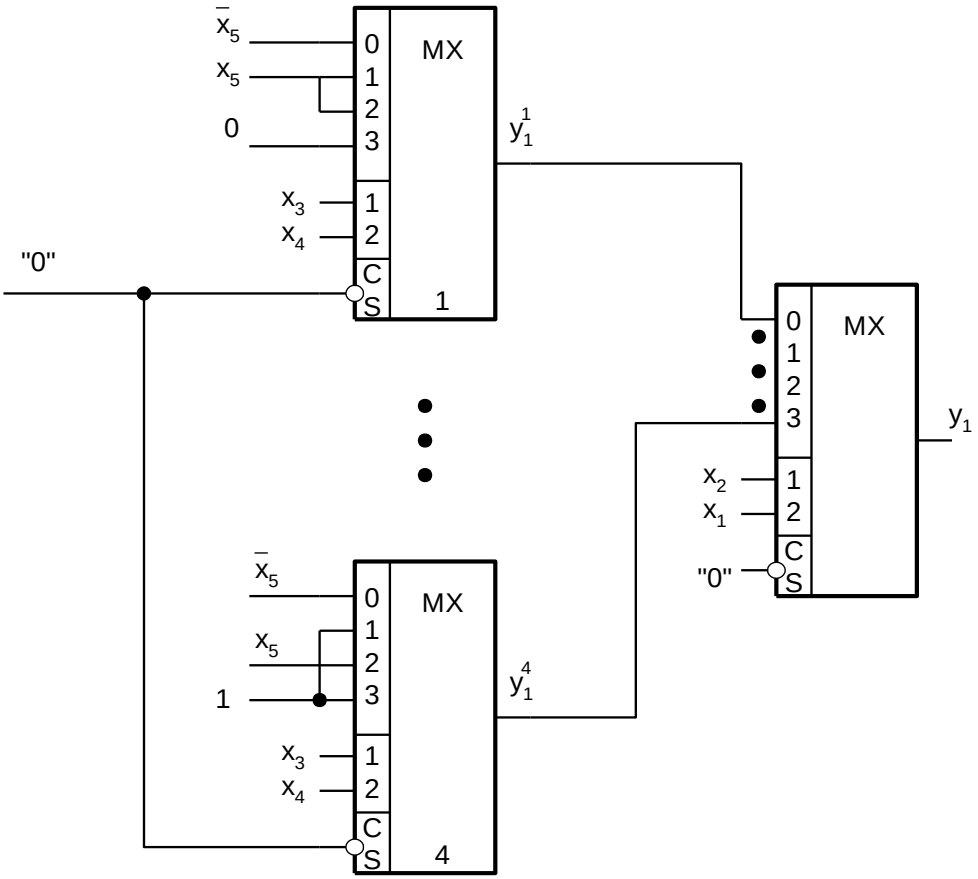


Рис. 4.37 – Реалізація функції y_1 на мультиплексорах

Порівняння формул (4.18) і (4.19) показує, що число рівнів у схемі на мультиплексорах може бути на одиницю меншим, ніж у схемі на дешифраторах за однакового числа входів *DC* і керуючих входів мультиплексора. Крім того, за цих умов число елементів першого рівня для схеми на мультиплексорах в два рази менше, ніж для схеми на дешифраторах, що випливає і з аналізу схем на рис. 4.32 і на рис. 4.37. При цьому загальне число елементів у схемі на мультиплексорах менше, а швидкодія цієї схеми вище, ніж у схемі на дешифраторах. Вибір методу синтезу в даному разі залежить від того, які мікросхеми знаходяться в розпорядженні розробника комбінаційної схеми.

Список літератури до розділу 4

1. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

2. DeMicheli G. Synthesis and Optimization of Digital Circuits. New York: McGraw Hill, 1994. 579 p.

3. Миллер Р. Теория переключательных схем. Т. 1. Комбинаторные схемы. Москва: Мир, 1970. 416 с.

5 СИНТЕЗ КОМБІНАЦІЙНИХ СХЕМ НА ПРОГРАМОВАНИЙ ЛОГІЧНИЙ ПРИСТРІЙ

5.1 Розвиток елементного базису цифрових пристроїв

Методи синтезу цифрових пристроїв значною мірою залежать від елементного базису, що використовується. При цьому під «цифровим пристроєм» будемо розуміти пристрій, який обробляє двійкову інформацію.

Історично першими пристроями такого типу були релейно-контактні схеми. Ці схеми з'явилися в першій чверті XX ст. і широко використовувалися, наприклад, в телефонії. Основними елементами таких схем були реле – перемикачі, призначені для комутації електричних ланцюгів (стрибкоподібної зміни вихідних величин) за заданих змін вхідних величин (не обов'язково електричних). Основні три частини реле – це електромагніт, якір і перемикач. Електромагніт є котушкою з намотаним на неї електричним дротом. Котушка містить сердечник з магнітного матеріалу. Якір – це пластина з магнітного матеріалу, що керує контактами через штовхач. Під час проходження через реле електричного струму виникає магнітне поле, яке притягує якір до сердечника. При цьому якір через штовхач зміщує і тим самим перемикає контакти. Перемикачі можуть бути трьох типів: ті, що замикають, ті, що розмикають, і ті, що перемикають. На рис. 5.1а наведене схематичне зображення реле. Змінна x_1 управляє контактом реле. Якщо $x_1 = 1$, то контакт замикається, і на вихід y_1 передається сигнал 1 з вхідного полюса реле. Отже, схема на рис. 5.1а реалізує повторювач, тобто $y_1 = x_1$.

Булева функція y_2 дорівнює одиниці, якщо $x_1 = x_2 = 1$, тобто $y_2 = x_1 x_2$ (рис. 5.1б). Функція y_3 дорівнює одиниці, якщо $x_1 = 1$ або $x_2 = 1$,

тобто $y_3 = x_1 \vee x_2$ (рис. 5.1в). Функція $y_4 = x_1(x_2 \vee x_3)$, що очевидно з попередніх рисунків (рис. 5.1г).

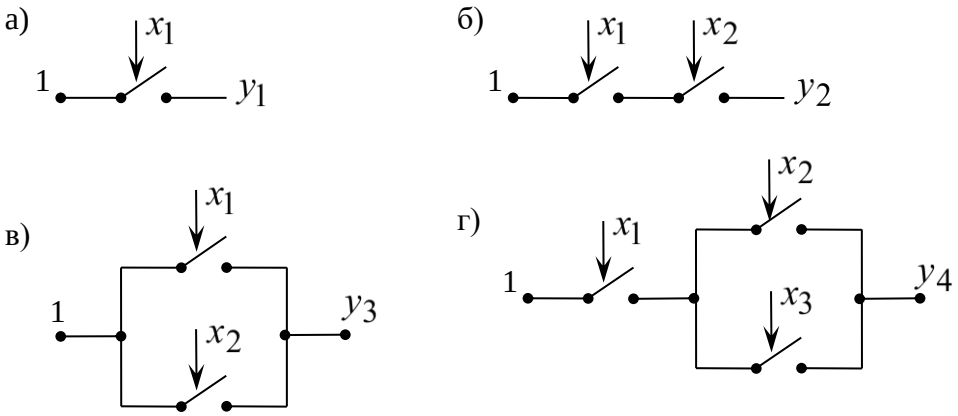


Рис. 5.1 – Реалізація булевих функцій у вигляді релейних схем

Зв’язок між релейними схемами і алгеброю Буля вперше помітив американський вчений Клод Шеннон (1916–2001 рр.). Перша його робота на цю тему називалася «Символічний аналіз реле і комутаторів» і була опублікована в 1938 р. Ця робота стала основою для розробки теорії синтезу і мінімізації комбінаційних схем. Першою вітчизняною роботою в цій області була стаття Віктора Івановича Шестакова (1907–1987 рр.), опублікована в 1941 р. Його роботи в цій галузі перетворили синтез релейних схем з мистецтва в науку.

Першими широко розповсюдженими цифровими системами були обчислювальні машини, які з’явилися в кінці 40-х рр. XX ст. Першим творцем автоматичної обчислювальної машини вважається німецький вчений К. Цузе, який в 1938 р. побудував модель машини Z1. Як елементний базис він використовував реле. У 1941 р. К. Цузе розробив релейну обчислювальну машину Z3 з програмним управлінням. У 1939 р. американський професор Д. Атанасов застосував для обчислювальної машини електронні лампи. З цього і почалася епоха електронних обчислювальних машин (ЕОМ).

Елементною базою першого покоління ЕОМ (1948–1958 рр.) були електронні лампи – діоди і тріоди. У цих машинах були реалізовані основні концепції Джона фон Неймана, що стосуються роботи ЕОМ зі збереженою в пам'яті програмою.

Друге покоління ЕОМ (1959–1967 рр.) використовувало для побудови своїх схем напівпровідникові прилади – діоди і транзистори. Перший транзистор був створений в 1948 р., а перша ЕОМ з їхнім використанням – в 1956 р. При цьому розрізнялися резисторно-транзисторна логіка (РТЛ) і діод-транзисторна логіка (ДТЛ). Наприклад, ДТЛ – це технологія побудови цифрових схем на основі біполярних транзисторів, діодів і резисторів. У ДТЛ логіка реалізовувалася за допомогою діодних ланцюгів, а транзистор використовувався для посилення сигналу. На рис. 5.2 зображений елемент 2І-НЕ (в спрощеній формі).

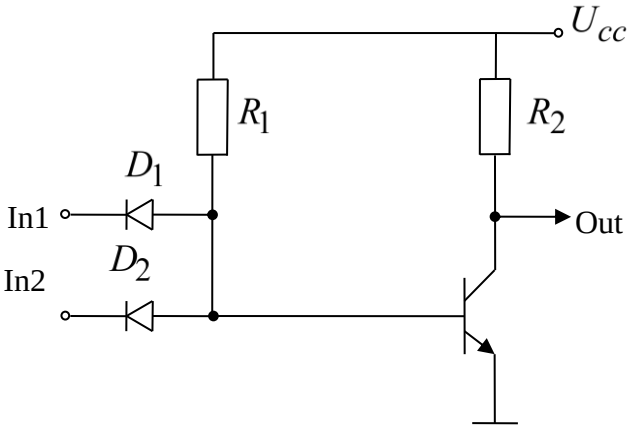


Рис. 5.2 – Логічний елемент 2І-НЕ системи ДТЛ

Цей елемент працює у такий спосіб. Якщо, наприклад, вхід $In1$ містить рівень логічного нуля, то діод D_1 відкритий, і струм від U_{cc} тече до входу $In1$ через резистор R_1. При цьому на вході Out формується рівень логічної одиниці ($\approx U_{cc}$). Якщо $In1 = In2 = 1$, то обидва діоди закриті, струм від U_{cc} тече на вихід через резистор R_2. При цьому $Out \approx 0{,}2B$, що відповідає рівню логічного нуля. Подібні елементи були

розташовані на одній платі, представляючи собою логічний модуль, який можна було видалити або замінити.

Розвиток напівпровідникової технології зумовив створення інтегральних схем. Перша ІС була розроблена незалежно Джеком Кілбі (1923–2005 рр.), який працював у фірмі Texas Instrument, і Робертом Нойсом (1927–1990 рр.), одним із засновників відомої фірми Fairchild Semiconductor. Основна ідея була об'єднати в одному напівпровідниковому кристалі транзистори, резистори, конденсатори та інші елементи. При цьому Кілбі використовував германій, а Нойс – кремній. У 1959 р. обидва вони отримали патенти на ІС, а вже в 1961 р. фірма Fairchild Semiconductor Corporation випустила у вільний продаж першу ІС на основі спільної з Texas Instrument ліцензії. Окремі фірми стали випускати сімейства інтегральних схем, що мають різні функціональні можливості. В інтегральних схемах тих років здебільшого застосовувалася технологія транзисторно-транзисторної логіки (ТТЛ). Прикладом мікросхем цього класу є сімейство К155, аналогом яких є мікросхеми серії SN155 фірми Signetics. Для реалізації логіки в К155 використовувалися багатоелементні транзистори, а для посилення сигналу використовувався повторювач, що складався з трьох транзисторів. Відтак, для реалізації схеми 2 І-НЕ потрібно чотири транзистори. Ці мікросхеми були в керамічному корпусі, мали 14 контактів, живилися від напруги +5V. В одному корпусі було, наприклад, 4 елементи 2І-НЕ, один елемент 8І-НЕ. У міру розвитку технології з'явилися мікросхеми, що реалізують дешифратори, суматори, мультиплексори, крім того, випускалися схеми на основі тригерів і власне тригери (про це ми поговоримо нижче). Прикладом мікросхем цього класу є сімейство К155, аналогом яких є мікросхеми серії SN155 фірми Signetics. Для реалізації логіки в К155 використовувалися багатоелементні транзистори, а для посилення сигналу використовувався повторювач, що складався з трьох транзисторів.

Інтегральні схеми стали основою ЕОМ третього покоління (1970–1979 рр.). Розвиток напівпровідникової технології привів до збільшення ступеня інтеграції, а це дало змогу фірмі Intel в 1970 р. створити перший мікропроцесор Intel 4004. Цей мікропроцесор був чотирирозрядним, мав можливість введення-виведення і обробки чотирирозрядних слів. Його швидкодія становила 8 000 операцій за секунду. Цей процесор був розрахований на застосування в програмованих калькуляторах з пам'яттю в 4 кбайт. Одночасно розвивався напрям програмованих логічних пристроїв (ПЛУ).

У ПЛУ логіка роботи задається користувачем, для чого всередині кристала встановлюються міжз'єднання. Часто ПЛУ називаються програмованими логічними інтегральними схемами (ПЛІС). Сучасні ПЛІС належать до класу НВІС (надвеликих інтегральних схем), в яких число транзисторів може перевищувати мільярд, а число зовнішніх контактів – тисячу. Першими представниками цього класу були програмовані постійні запам'ятовувальні пристрої (ППЗУ), які були розроблені фірмою Harris Semiconductor в 1970 р. У ті ж роки з'явилися перші програмовані логічні матриці (ПЛМ), які створила фірма Signetics. Подальший розвиток цієї ідеї втілився в мікросхемах програмованої матричної логіки (ПМЛ), які були розроблені фірмою Monolithic Memories в 1978 р. Прогрес в області напівпровідникової технології привів до появи комплементарних транзисторів, заснованих на тріаді метал-оксид-напівпровідник (МОП). Відтак, з'явилася КМОП-технологія, що дала змогу перейти від інтегральних схем середнього рівня інтеграції (СІС) і великих інтегральних схем до надвеликих БІС. При цьому виділилися два напрями НВІС, що використовуються для реалізації цифрових пристроїв і систем:

1. Complex programmable logic devices (CPLD). Саме ці мікросхеми у вітчизняній літературі і називаються ПЛІС. Ці ПЛУ складаються з макрокомірок на основі ПМЛ (переважно) або ПЛМ (сімейство CoolRunner фірми Xilinx). Макрокомірки з'єднуються між собою і з виходами мікросхеми за допомогою програмованої матриці міжз'єднань.

2. Field-programmable gate arrays (FPGA), які у вітчизняній літературі називаються програмованими матрицями вентилів (ПМВ). Вони складаються з величезного числа макрокомірок типу ППЗУ, що з'єднуються програмованою матрицею міжз'єднань.

Розглянемо особливості ППЗУ, ПЛМ, ПМЛ і методи синтезу комбінаційних схем на їхній основі.

Кожен з представників цих класів ПЛУ має дворівневу структуру, що складається з матриць І і АБО (рис. 5.3).

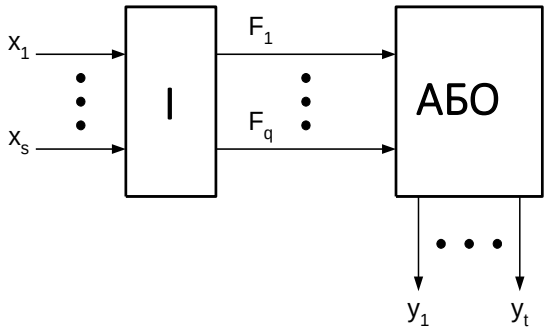


Рис. 5.3 – Узагальнена структура ППЗУ, ПЛМ і ПМЛ

Кожна з матриць є системою ортогональних шин, на перетині яких знаходяться елементи з однобічною провідністю. Матриця І формує q термів $F_1,\ldots,F_q$ від S вхідних змінних (або їхніх інверсій) $x_1,\ldots,x_S$. Матриця АБО формує t вихідних функцій $y_1,\ldots,y_t$ як диз'юнкції термів $F_1,\ldots,F_q$. Програмування матриць полягає у встановленні зв'язків між вертикальними і горизонтальними шинами матриць. Детальний розгляд цього базису виходить за рамки даного підручника, ці питання висвітлені в рекомендованій літературі.

Різниця між ППЗУ, ПЛМ і ПМЛ:

1. У ППЗУ матриця І є жорсткою (не програмується) і є повним дешифратором: $q = 2^S$. Це дає змогу реалізувати таблицю істинності системи функцій (або досконалу ДНФ).

2. У ПЛМ обидві матриці є програмованими, що дає змогу реалізувати довільні ДНФ систем булевих функцій.

3. У ПМЛ матриця І є програмованою, що дає змогу реалізувати не тільки мінтерми, але і терми довільних ДНФ. Матриця АБО є фіксованою. Це дає змогу збільшити параметри S, t, q, порівняно з ПЛМ, в яких значна площа кристала витрачається на елементи програмування зв'язків.

5.2 Синтез схем на базі ППЗУ і ПЛМ

Розглянемо методи реалізації систем функцій на ППЗУ (S,t), де S – число входів, t – число виходів. Умовне позначення ППЗУ (рис. 5.4) включає абревіатуру PROM (Programmable Read Only Memory – програмована пам'ять, яка припускає лише читання інформації). Тут CS – вхід вибірки кристала ППЗУ. Якщо $\overline{y_{CS}} = 0$, то на виходах $y_1,\ldots,y_t$ формується інформація з осередку ППЗУ, адреса якої визначається вхідним набором $< x_1,\ldots,x_S >$.

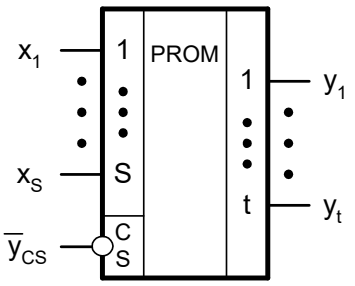


Рис. 5.4 – Умовне позначення ПЗУ

Якщо $\overline{y_{CS}} = 1$, то на всіх виходах ППЗУ встановлюється нульовий рівень або третій стан, про що йдеться в розділі, що пояснює організацію шин в операційних автоматах.

Метод синтезу системи функцій $y_1,\ldots,y_N$, що залежать від L вхідних змінних і визначених на H термах ($H \le 2^L$), залежить від поєднання параметрів системи (L,N,H) і ППЗУ (S,t). При цьому можливі такі ситуації:

1. $S \geq L$, $t \geq N$. В цьому разі система Y тривіально реалізується на одному ППЗУ (S,t). Наприклад, система $y_1,\ldots,y_4$ (табл. 5.1) тривіально реалізується на ППЗУ (4,4) (рис. 5.5).

Таблиця 5.1

x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0
0	0	0	1	0	1	1	0	1	0	0	1	0	0	0	1
0	0	1	0	1	0	0	1	1	0	1	0	1	0	1	0
0	0	1	1	0	1	0	0	1	0	1	1	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1
0	1	0	1	0	0	0	0	1	1	0	1	0	0	1	0
0	1	1	0	1	0	0	0	1	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0

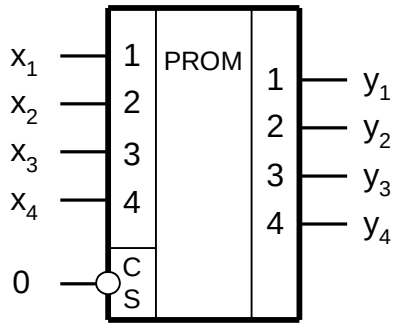


Рис. 5.5 – Тривіальна реалізація системи булевих функцій на ППЗУ

Програмування ППЗУ зводиться до занесення в його осередок слів інформації, відповідних вихідних функцій. Наприклад, за адресою 0000 заноситься слово 1010 (перший рядок табл. 5.1), за адресою 0001 заноситься слово 0110 (другий рядок) і так далі.

2. $S \geq L$, $t < N$. В цьому разі для реалізації системи Y потрібно $n_1 = \lceil N/t \rceil$ мікросхем ППЗУ (рис. 5.6), на кожній з яких реалізується частина вихідних функцій. Параметр i визначається як $i = t(n_1 - 1) + 1$. Такий підхід називається «розширенням ППЗУ по виходах». Зауважимо, що кожен елемент схеми реалізує всі терми системи функцій Y.

Для системи, зображеній у табл. 5.1, і ППЗУ (4,2) $n_1 = \lceil 4/2 \rceil = 2$, і комбінаційна схема складаються з двох мікросхем ППЗУ (4,2) (рис. 5.7).

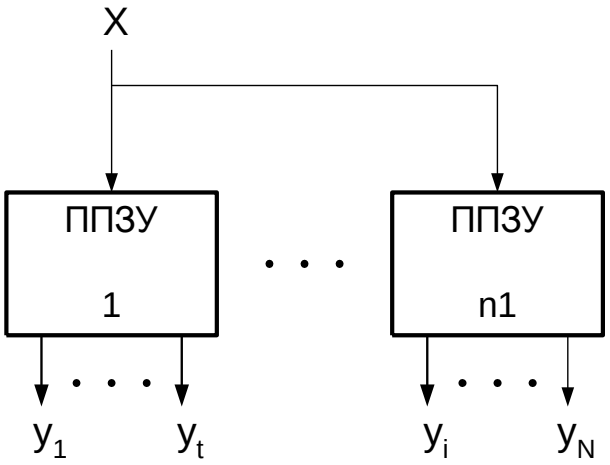


Рис. 5.6 – Розширення ППЗУ по виходах

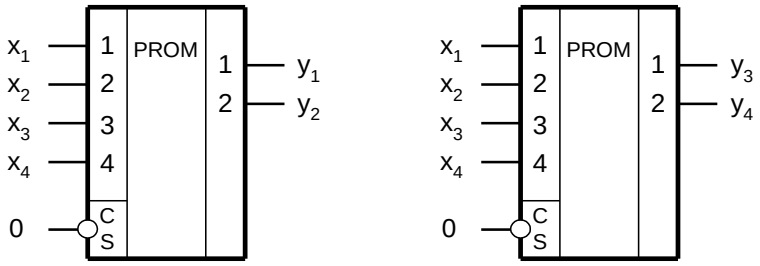


Рис. 5.7 – Реалізація системи Y з розширенням ППЗУ по виходах

3. $S < L$, $t \geq N$. В цьому разі для реалізації системи Y потрібно $n_2 = \lceil H/q \rceil$ мікросхем ППЗУ, на входи яких подаються молодші розряди адреси. Це відповідає розбиттю вихідної таблиці істинності на n_2 підтаблиць, кожна з яких визначається однією з комбінацій старших розрядів адреси. Наприклад, для набору $\langle x_1,\ldots,,x_L \rangle$ старші $i = L - S$ розрядів $x_1,\ldots,,x_i$ визначають одну з $n_2 = 2^{L-S}$ підтаблиць, а молодші розряди $\langle x_{i+1},\ldots,,x_L \rangle$ визначають слово всередині конкретної підтаблиці. Для вибору підтаблиць необхідний дешифратор DC (рис. 5.8).

На виходах j-ої мікросхеми ППЗУ формуються деякі часткові значення $y_1^j,...,y_N^j$ функцій $y_1,...,y_N$ з j-ої підтаблиці вихідної таблиці істинності ($j = 1,\ldots,n_2$). Для формування значення функції y_n необхідно реалізувати диз'юнкцію значень цієї функції для всіх підтаблиць

$y_n = \bigvee_{j=1}^{n_2} y_n^j$ $(=1,\ldots,N)$. Такий підхід називається розширенням ППЗУ за термами.

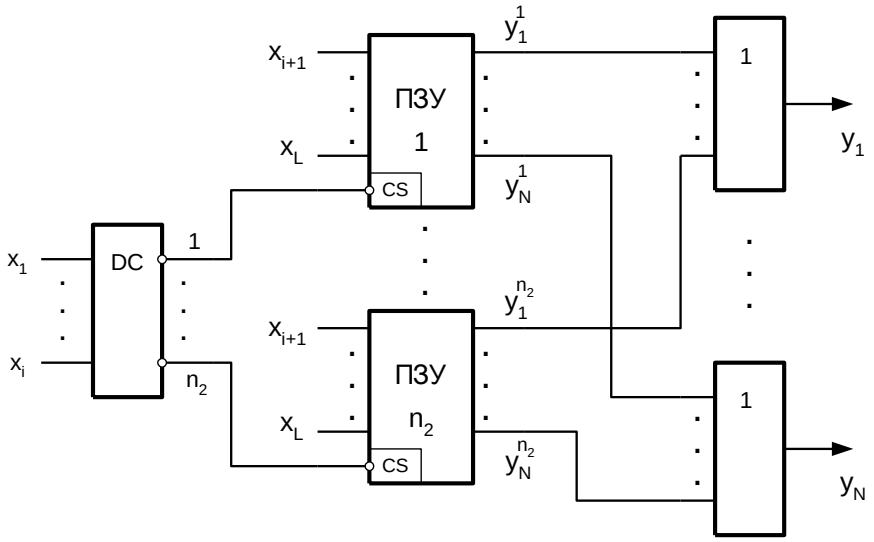


Рис. 5.8 – Реалізація схеми з розширенням ППЗУ по термам

При реалізації системи функцій Y, заданої табл. 5.1, на ППЗУ (2,4) необхідно $n_2 = 4$ мікросхеми ППЗУ (рис. 5.9).

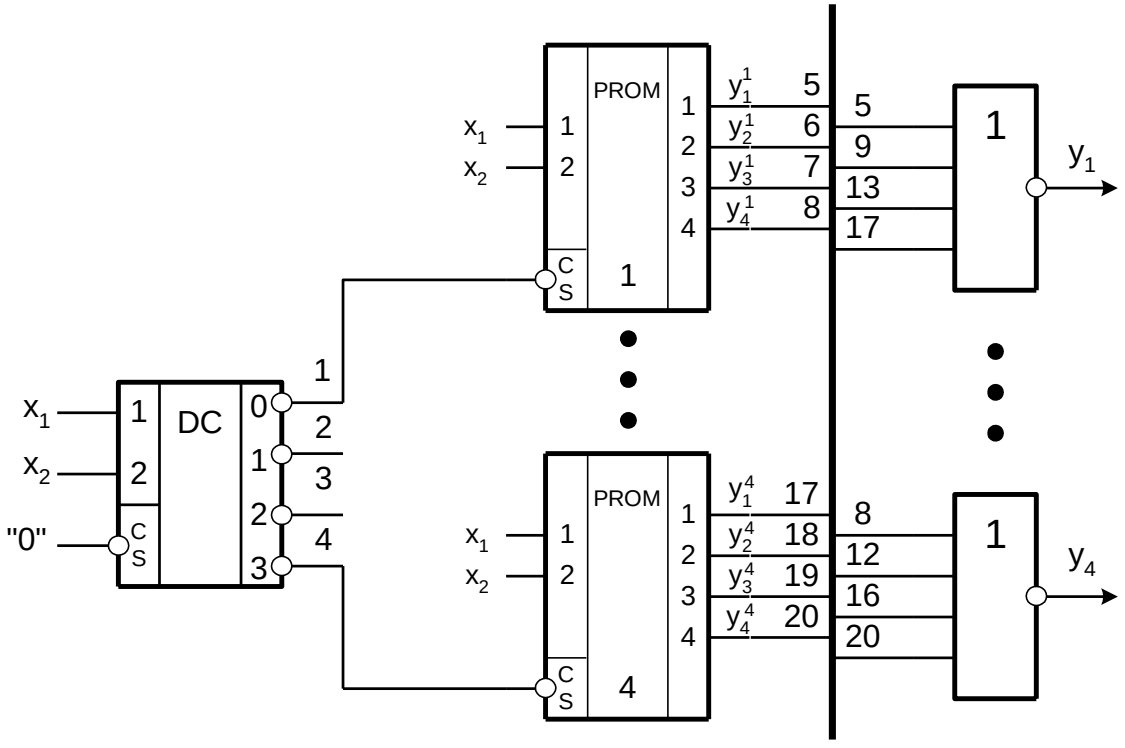


Рис. 5.9 – Реалізація системи Y з розширенням ППЗУ за термами

4. $S < L$, $t < N$. Цей випадок об'єднує два розглянуті раніше. Для реалізації схеми необхідно розширити ППЗУ за термами в n_2 разів і для кожного рівня розширення за термами виконати розширення за виходами в n_1 разів. Отже, для реалізації системи Y потрібно $n_3 = n_1 n_2$ корпусів ППЗУ.

При реалізації системи Y (табл. 5.1) на ППЗУ (2,2) $n_2 = \lceil 16/4 \rceil = 4$, $n_1 = \lceil 4/2 \rceil = 2$, отже, $n_3 = 8$. Схема на рис. 5.10 складається з чотирьох шарів (рівнів) ППЗУ, для вибірки яких використовується дешифратор DC, і кожен шар складається з двох ППЗУ.

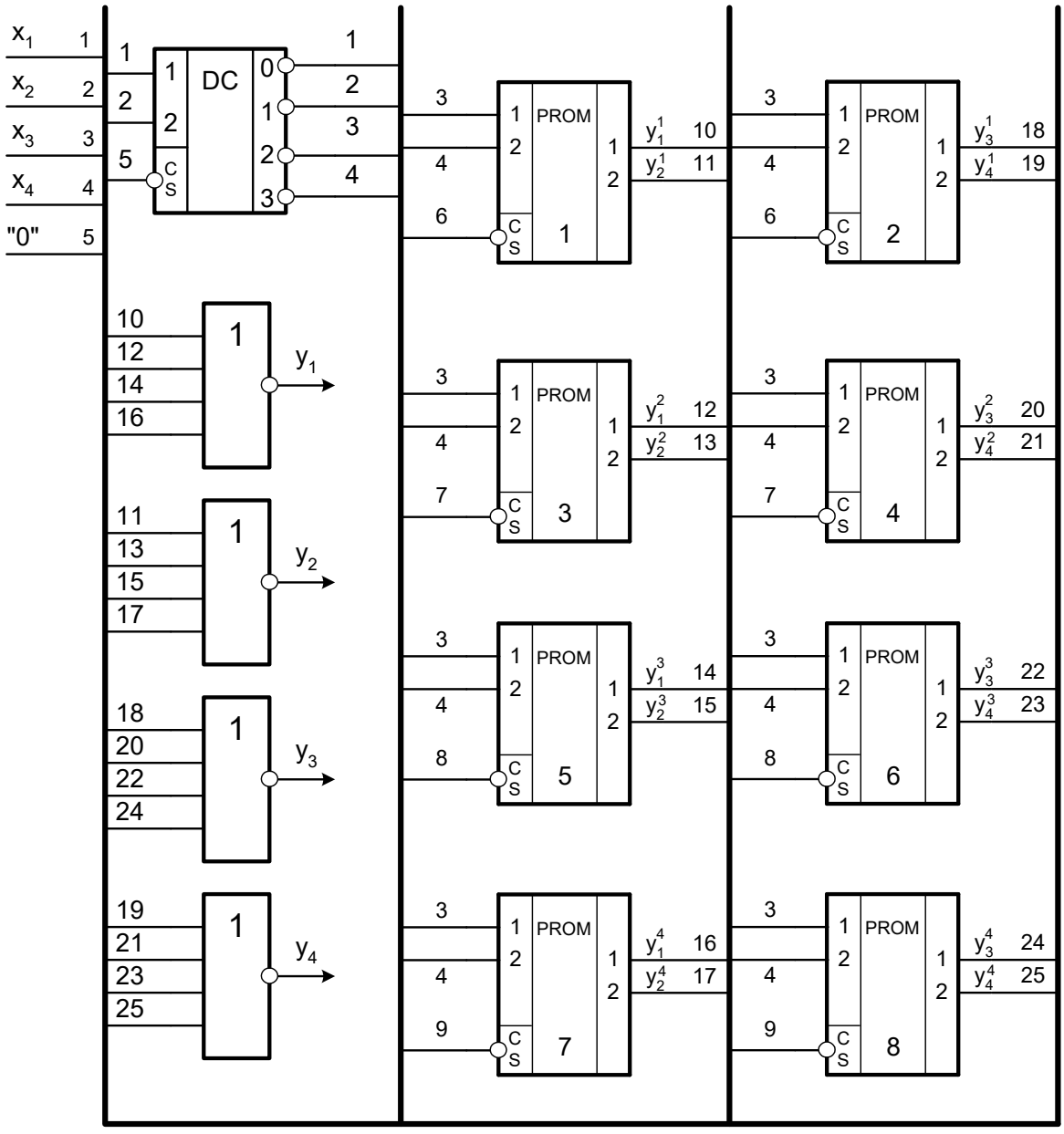


Рис. 5.10 – Реалізація системи Y з розширенням ППЗУ за термами і виходами

У цій схемі мікросхемі ППЗУ 5, наприклад, відповідає підтаблиця T_5 вихідної таблиці істинності, наведена в табл. 5.2. Тут вхід 8 відповідає входу вибірки ППЗУ 5 в схемі (рис. 5.10). Входу 8 відповідає набір $x_1x_2 = 10$, необхідний для вибору третього шару ППЗУ схеми. Підтаблиця T_5 виділена в початковій таблиці істинності (табл. 5.1).

Таблиця 5.2

8	x_3	x_4	y_1	y_2
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	0	1
1	*	*	0	0

Усі розглянуті ситуації можуть бути представлені графічно (рис. 5.11), якщо ППЗУ уявити як таблицю розмірності qt, а систему функцій – як таблицю розмірності HN.

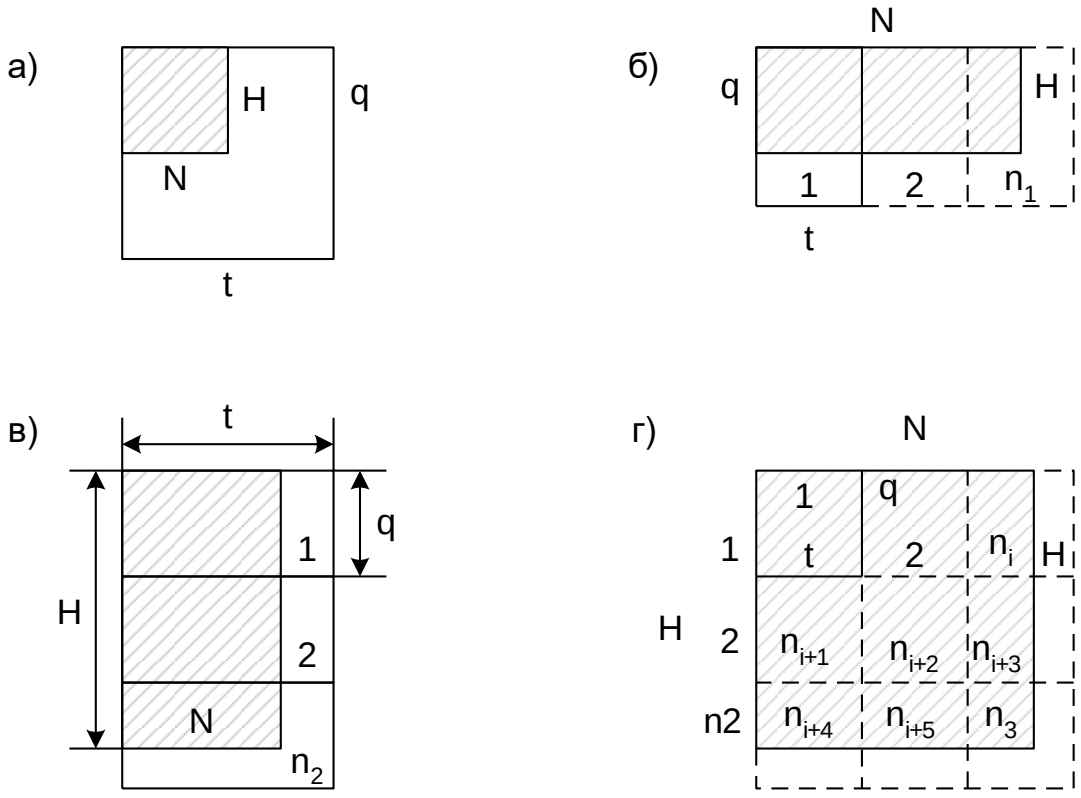


Рис. 5.11 – Графічне представлення методів реалізації систем булевих функцій на ППЗУ

На рис. 5.11 таблиця системи функцій заштрихована. Рис. 5.11а відповідає тривіальній реалізації системи функцій, рис. 5.11б – розширенню за виходами, рис. 5.11в – розширенню за термами, рис. 5.11г – спільному розширенню за виходами і термами.

Методи синтезу схем на ПЛМ мають свої особливості. Позначимо ПЛМ, що має s входів, t виходів і q термів символом $ПМЛ(s,t,q)$. Нехай система функцій Y має параметри L, N, H і позначається як $Y(L,N,H)$. Розглянемо, наприклад, таку систему функцій:

$$\begin{aligned} y_1 &= \bar{x}_1\bar{x}_2x_3 \vee x_1x_2 \vee \bar{x}_3x_4 = F_1 \vee F_2 \vee F_3; \\ y_2 &= \bar{x}_1\bar{x}_2 \vee x_1x_2 \vee x_3x_4 = F_4 \vee F_2 \vee F_5; \\ y_3 &= \bar{x}_1\bar{x}_2\bar{x}_3 \vee x_3x_4 \vee x_3\bar{x}_4 = F_6 \vee F_5 \vee F_7. \end{aligned} \qquad (5.1)$$

Метод синтезу залежить від поєднання параметрів L, N, H і s, t, q. У загальному випадку можливі 8 ситуацій. Розглянемо деякі з них більш детально.

1. $s \geq L$, $t \geq N$, $q \geq H$. В цьому випадку система Y тривіально реалізується на одній ПЛМ. При цьому система (5.1) представляється у вигляді такої таблиці програмування (табл. 5.3).

Таблиця 5.3

Вивчення функцій (5.1) за тривіальної реалізації на ПЛМ

Входи				Терми	Виходи		
x_1	x_2	x_3	x_4		y_1	y_2	y_3
0	0	1	-	F_1	1	•	•
1	1	-	-	F_2	1	1	•
-	-	0	1	F_3	1	•	•
0	0	-	-	F_4	•	1	•
-	-	1	1	F_5	•	1	1
0	0	0	-	F_6	•	•	1
1	-	-	0	F_7	•	•	1

Знак «0» («1») в кожному рядку означає, що відповідна змінна входить у терм F_h у вигляді інверсії (без інверсії). Знак «–» означає, що змінна є несуттєвою. Знак «1» («•») В колонці y_n означає, що функція залежить (не залежить) від даного терму.

2. $s \geq L$, $t < N$, $q \geq H$. При цьому необхідно виконати розширення ПЛМ за входами в n_1 разів, як це виконувалося для ППЗУ. При цьому схема буде збігатися зі схемою на рис. 5.6. Нехай, наприклад, для реалізації системи (5.1) використовуються *ПЛМ(4,2,8)*. Тоді реалізація цієї системи показана на рис. 5.12.

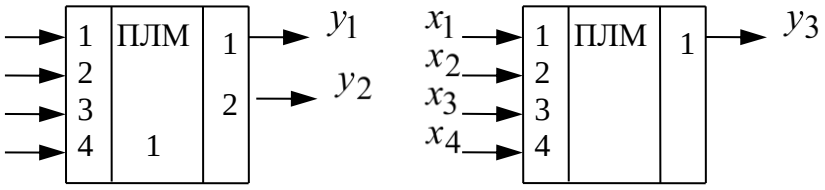


Рис. 5.12 – Реалізація системи (5.1) на ПЛМ(4,2,8)

3. $s \geq L$, $t \geq N$, $q < H$. При цьому необхідно виконати розширення ПЛМ за термами. На відміну від схеми на рис. 5.9, дешифратор не потрібний, що приводить до схеми на рис. 5.13.

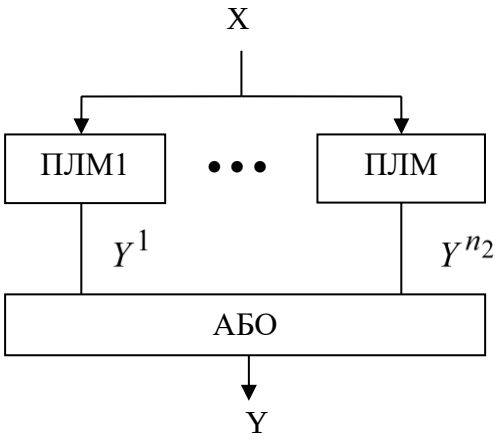


Рис. 5.13 – Розширення ПЛМ за термами

Дешифратор DC відсутній, оскільки одні й ті самі змінні надходять на входи всіх ПЛМ. Як і в разі ППЗУ, параметр $n_2 = \left\lceil \frac{H}{q} \right\rceil$. Нехай для

реалізації системи (5.1) використовуються *ПЛМ(4,3,4)*, тоді $n_2 = 2$. Розіб'ємо табл. 5.3 на дві частини, в першій з яких будуть терми $F_1 - F_4$, а в другій – інші терми. Це приводить до схеми, зображеній на (рис. 5.14).

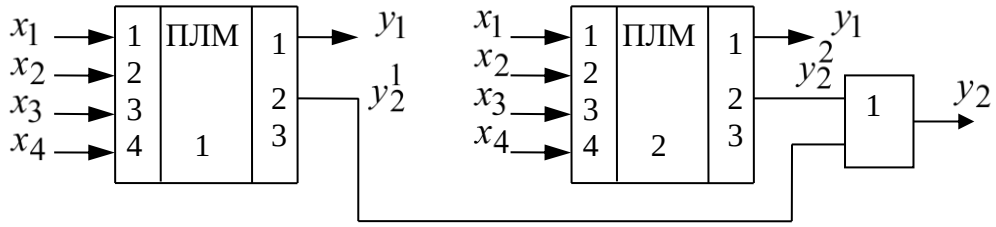


Рис. 5.14 – Реалізація системи (5.1) на ПЛМ(4,3,4)

Як видно зі схеми (рис. 5.14), частина термів функції y_2 реалізується на $ПЛМ_1$, а частина – на $ПЛМ_2$. Цим термам відповідають підфункції y_2^1 і y_2^2 (що пояснює позначення Y^1 на рис. 5.13), які потім з'єднуються схемою АБО. Очевидно, для реалізації системи (5.1) достатньо *ПЛМ(4,2,4)*, оскільки система Y розбилася на дві підсистеми $Y^1 = \{y_1, y_2^1\}$ і $Y^2 = \{y_2^2, y_3\}$. Кожна з цих підсистем має два елементи, тому достатньо ПЛМ з $t = 2$.

4. $s \geq L$, $t < N$, $q < H$. В цьому випадку для реалізації системи Y необхідно застосувати як розширення ПЛМ за виходами, так і за термами. Число ПЛМ в кінцевій схемі залежить від того, як розбиті функції $y_n \in Y$ і терми цих функцій за різними підсистемами. Як уже зазначалося, при розбитті термів системи (5.1) на підсистеми $F^1 = \{F_1, F_2, F_3, F_4\}$ і $F^2 = \{F_5, F_6, F_7\}$ достатньо 2 *ПЛМ(4,2,4)*, хоча $N = 3 > t = 2$.

Розглянемо реалізацію системи (5.1) на мікросхемах *ПЛМ(4,1,4)*. Очевидно, $n_1 = 3$, $n_2 = 2$, і для реалізації цієї схеми необхідно в гіршому випадку $n_1 \cdot n_2 = 6$ *ПЛМ(4,1,4)*. Однак для розбиття $F^1 = \{F_1, \ldots, F_4\}$ і

$F^2 = \{F_5, F_6, F_7\}$ маємо схему (рис. 5.15), що складається тільки з 4-х *ПЛМ(4,1,4)*.

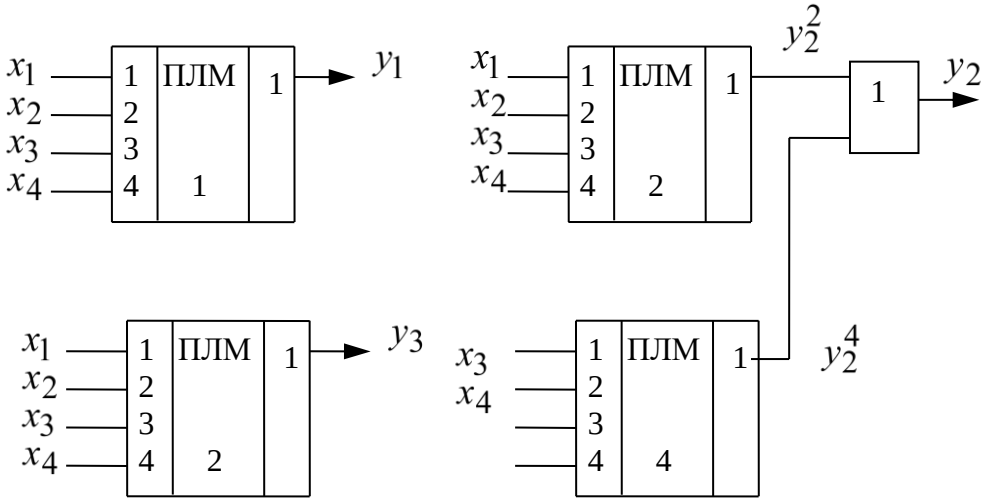


Рис. 5.15 – Реалізація системи (5.1) на ПЛМ(4,1,4)

Очевидно, що кожна з ПЛМ цієї схеми містить деяку частину вихідної системи функцій. При цьому може використовуватися тільки частина наявних ресурсів мікросхеми (тобто її входи, виходи і терми). Практично ніколи не вдається використовувати всі наявні ресурси БІС. Вміст $ПЛМ_1 - ПЛМ_4$ цієї схеми представлено в табл. 5.4–5.7 відповідно.

Таблиця 5.4

Програмування $ПЛМ_1$

Входи				Терми	Виходи
x_1	x_2	x_3	x_4		y_1
0	0	1	-	F_1	1
1	1	-	-	F_2	1
-	-	0	1	F_3	1
0	0	-	-	F_4	•

Таблиця 5.5

Програмування $ПЛМ_2$

Входи		Терми	Виходи
x_1	x_2		y_2^2
1	1	F_2	1
0	0	F_4	1

Таблиця 5.6

Програмування $ПЛМ_3$

Входи				Терми	Виходи
x_1	x_2	x_3	x_4		y_3
-	-	1	1	F_5	1
0	0	0	-	F_6	1
1	-	-	0	F_7	1

Таблиця 5.7

Програмування $ПЛМ_4$

Входи		Терми	Виходи
x_3	x_4		y_2^4
1	1	F_5	1

Як видно зі схеми (рис. 5.15), в мікросхемах $ПЛМ_2$ і $ПЛМ_4$ використовуються тільки по 2 входи. При цьому кількість термів, що використовуються в різних ПЛМ схеми, змінюється від одного до чотирьох.

Якщо $L > s$, то для синтезу комбінаційної схеми необхідно використовувати складні комбінаторні алгоритми. Проєктування схеми

зводиться до знаходження розбиття Π_F множини термів F (де $|F| = H$) на мінімальне число блоків U. Нехай $X(E_u)$ – множина аргументів, що входять в терм з множини $E_u \subseteq \Pi_F$, де $E_u \subseteq Y$. Тоді розбиття Π_F має задовольняти умову:

$$\begin{aligned} &|X(E_u)| \le S; \\ &|Y(E_u)| \le t; \\ &|E_u| \le q; \\ &U \to \min. \end{aligned} \qquad (5.2)$$

Існують модифікації завдання (5.2) і методів його вирішення, але вони виходять за рамки нашої книги.

5.3 Синтез схем на базі мікросхем ПМЛ

У найпростішому випадку структурна схема ПМЛ представляється у вигляді t програмованих матриць І, жорстко пов'язаних з одним входом вихідної матриці АБО (рис. 5.16).

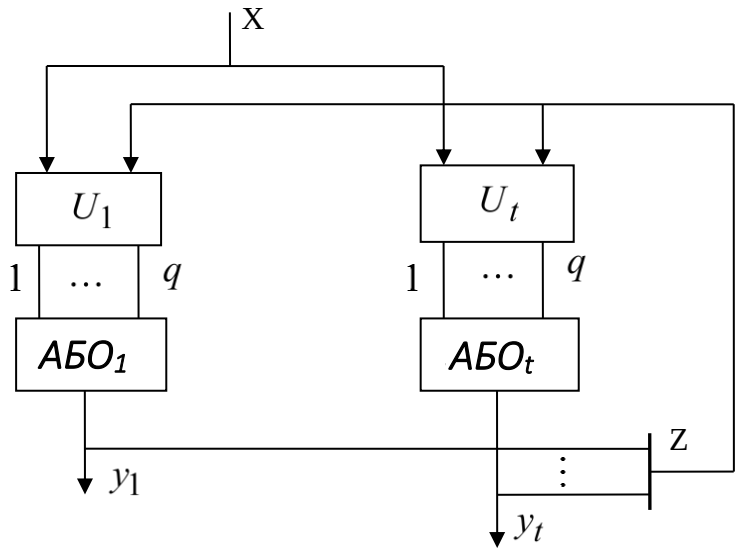


Рис. 5.16 – Структурна схема ПМЛ

На входи кожної схеми І надходять s вхідних сигналів X і t сигналів зворотного зв'язку Z, утворені з виходів $y_1, \ldots, y_t$. Отже, кожна схема І має $s+t$ входів (t з цих входів є внутрішніми). Наявність

зворотного зв'язку Z дає змогу реалізувати дужкові формули. Домовимося ПМЛ зі структурою, показаною на рис. 5.16, позначати символом $ПМЛ(s,t,q)$. Однак будемо пам'ятати, що насправді мікросхема містить tq різних термів.

Наявність ланцюгів зворотного зв'язку дає змогу реалізувати на одній ПМЛ як однорівневі, так і багаторівневі комбінаційні схеми. Нехай $H(y)$ – число термів у ДНФ функції $y_n \in Y$. Тоді функція y_n може бути реалізована на одній парі матриць $\langle I, АБО \rangle$ ПМЛ, якщо виконується умова

$$H(y_n) \le q. \tag{5.3}$$

Якщо для всіх функцій $y_n \in Y$ виконується умова (5.3), то система Y реалізується у вигляді однорівневої схеми за виконання умови

$$L \le S. \tag{5.4}$$

Якщо при цьому виконується умова

$$N \le t, \tag{5.5}$$

то для реалізації системи Y достатньо однієї $ПМЛ(s,t,q)$. Якщо умова (5.5) порушується, то для реалізації системи Y виконується розширення ПЛМ за виходами. Для реалізації схеми в цьому випадку потрібно

$$n_1 = \left\lceil \frac{N}{t} \right\rceil \tag{5.6}$$

мікросхем $ПМЛ(s,t,q)$.

Наприклад, система функцій (5.1) тривіально реалізується на одній мікросхемі $ПМЛ(4,3,3)$, оскільки $H(y_1) = H(y_2) = H(y_3) = 3$ (рис. 5.17).

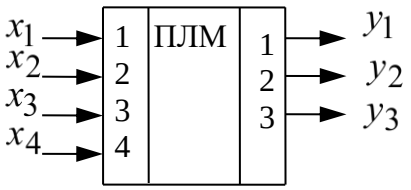


Рис. 5.17 – Тривіальна реалізація системи (5.1) на $ПМЛ(4,3,3)$

Для програмування матриць U цієї схеми необхідно побудувати таблиці програмування (табл. 5.8). На відміну від табл. 5.3, що використовується для програмування ПЛМ при реалізації системи (5.1), в ПМЛ програмується 9 термів. При цьому терм F_2 дублюється в матрицях U_1 і U_2, а терм F_5 – в матрицях U_2 і U_3. Очевидно, що така реалізація має максимально можливу швидкодію, оскільки між входами функції і її виходами знаходиться тільки один рівень матриць $\langle I, АБО \rangle$.

Таблиця 5.8

Вивчення функцій (5.1) за тривіальної реалізації на ПМЛ

Входи				Терми	Виходи			Матриця АБО
x_1	x_2	x_3	x_4		y_1	y_2	y_3	
0	0	1	-	F_1	1			1
1	1	-	-	F_2	1			
-	-	0	1	F_3	1			
1	1	-	-	F_2		1		2
0	0	-	-	F_4		1		
-	-	1	1	F_5		1		
-	-	1	1	F_5			1	3
0	0	0	-	F_6			1	
1	-	-	0	F_7			1	

Якщо система (5.1) реалізується на $ПМЛ(4,2,3)$, то для реалізації схеми, згідно з (5.6), достатньо 2 мікросхеми.

Якщо умова (5.3) порушується, то метод реалізації комбінаційної схеми залежить від наявності можливості об’єднання виходів ПМЛ за допомогою монтажного АБО. Це можливо у процесі використання технології відкритого колектору або відкритого стоку (open-drain). Якщо така можливість є, то на кожному виході формується деяка підфункція

функції $y_n \in Y$, а потім всі виходи об'єднуються за схемою монтажного АБО. При цьому значення функції інвертується. Щоб отримати пряме значення функції, необхідно або поставити інвертори на вихід схеми, або реалізувати зворотну функцію. Розглянемо більш докладно останній випадок на прикладі функції f, заданої картою Карно (рис. 5.18). Очевидно, що зворотна функція $\bar{f} = 1$, якщо функція $f = 0$.

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	0	1	1	1
10	0	1	0	1

Рис. 5.18 – Карта Карно для функції f

З карти Карно (рис. 5.18) маємо ДНФ для функцій f і $\bar{f}$:

$$f = \bar{a}c\bar{d} \vee a\bar{c}d \vee abc \vee ab\bar{c}; \tag{5.7}$$

$$f = \bar{a}d \vee \bar{a}c \vee a\bar{c}\bar{d} \vee \bar{b}cd. \tag{5.8}$$

Обидві функції мінімізовані, причому $H(f) = H(\bar{f}) = 4$. Нехай для реалізації функції використовуються $ПМЛ(4,2,3)$, тоді умова (5.3) порушується як для прямої функції f, так і для зворотної функції $\bar{f}$. Нехай виходи ПМЛ можуть бути об'єднані монтажним АБО. Тоді реалізацію схеми доцільно проводити для функцій $\bar{f}$, що приводить до схеми на рис. 5.19. Тут знак «×» означає монтажне АБО.

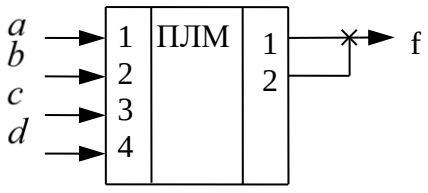


Рис. 5.19 – Реалізація функції f на ПМЛ з монтажним АБО

Метод синтезу досить простий. Необхідно визначити, скільки матриць І (вхід ПМЛ) потрібно для реалізації термів функції. Цей параметр визначається за формулою

$$n_4 = \left\lceil \frac{H(y_n)}{q} \right\rceil. \qquad (5.8)$$

Далі терми функції розподіляються по n_4 матрицями U, а виходи відповідних матриць АБО об'єднуються. У розглянутому прикладі $n_4 = 2$, а таблиця програмування приведена в табл. 5.9.

Таблиця 5.9

Вивчення функцій (5.8) при реалізації на $ПМЛ(4,2,3)$ **з монтажним АБО**

Входи				Терми	Виходи		Матриця АБО
a	b	c	d		f^1	f^2	
0	-	-	1	F_1	1	•	1
0	-	1	-	F_2	1	•	
1	-	0	0	F_3	1	•	
-	0	1	1	F_4	•	1	2

Якщо можливість монтажного АБО відсутня, то необхідно використовувати зворотні зв'язки Z. При цьому схема буде дворівневою, якщо виконується умова

$$H(y_n) \le tq - n_4 + 1. \qquad (5.9)$$

Оскільки $H(f) = 4$, а $q = 3$, то для розглянутого прикладу умова (5.3) порушується, тобто потрібна багаторівнева реалізація функції f. Оскільки $tq = 6$, то умова (5.9) виконується, і для реалізації функції f достатньо однієї мікросхеми $ПМЛ(4,2,3)$, що показано на схемі (рис. 5.20). Тепер в таблиці програмування необхідно відобразити терми Z_1 і Z_2, відповідні входам зворотного зв'язку (ОС) матриць U_1 і U_2 (табл. 5.10).

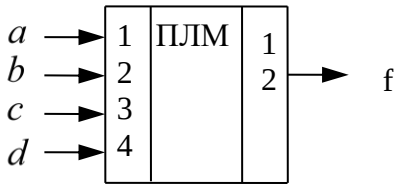


Рис. 5.20 – Реалізація функції (5.70) на $ПМЛ(4,2,3)$

Таблиця 5.10

Вивчення функцій (5.7) при реалізації на $ПМЛ(4,2,3)$

Входи				ОС		Терми	Виходи		Матриця АБО
a	b	c	d	z_1	z_2		f^1	f	
0	-	0	0	-	-	F_1	1	•	1
1	-	0	1	-	-	F_2	1	•	
1	1	1	-	-	-	F_3	1	•	
-	-	-	-	1	-	Z_1	•	1	2
1	1	0	-	-	-	F_4	•	1	

Як видно з табл. 5.10, функція f може бути представлена як $f = z_1 \vee F_4$, де $z_1 = f^1 = F_1 \vee F_2 \vee F_3$. Якщо один рівень $\langle I, АБО \rangle$ спрацьовує за 5 наносекунд, то значення функції f в даному випадку формується за 10 наносекунд.

Розглянемо булеву функцію f_1 (рис. 5.21), для реалізації якої необхідно використовувати мікросхеми $ПМЛ(4,2,3)$.

З цієї карти Карно можна отримати таку мінімальну ДНФ функції f_1:

$$\begin{aligned} f_1 &= \bar{a}\bar{b}\bar{c}\bar{d} \vee ab\bar{c}\bar{d} \vee \bar{a}b\bar{c}d \vee a\bar{b}\bar{c}d \vee \bar{a}\bar{b}cd \vee bc\bar{d} = \\ &= F_1 \vee F_2 \vee F_3 \vee F_4 \vee F_5 \vee F_6 \end{aligned}. \qquad (5.10)$$

ab \ cd	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	0	1
10	0	1	0	0

Рис. 5.21 – Карта Карно для функції f_1

Для *ПМЛ*(4,2,3) маємо $tq = 6$, при цьому $n_4 = 2$, і умова (5.9) порушується. Отже, для реалізації схеми знадобиться 2 мікросхеми *ПМЛ*(4,2,3). Уявімо функцію (5.10) у вигляді трьох функцій:

$$\begin{aligned} z_1 &= F_1 \vee F_2 \vee F_3; \\ f_1^1 &= z_1 \vee F_4 \vee F_5; \\ f_1 &= f_1^1 \vee F_6. \end{aligned} \qquad (5.11)$$

На рис. 5.22 показана реалізація комбінаційної схеми, яка відповідає системі (5.11).

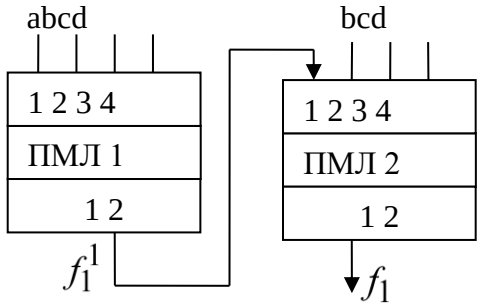


Рис. 5.22 – Реалізація функції (5.22) на ПМЛ(4,2,3)

Така реалізація можлива тільки тому, що в терм F_6 входить тільки три букви. Отож, один з входів $ПМЛ_2$ може бути використаний для реалізації підфункції f_1^1. В даній схемі між входом і виходом знаходиться три рівні ⟨*І, АБО*⟩.

При реалізації функції (5.10) на *ПМЛ*(5,2,3) можна отримати і дворівневу схему. При цьому логічна функція f_1 представляється у вигляді такої системи:

$$f_1^1 = F_1 \vee F_2 \vee F_3;$$
$$f_1^2 = F_4 \vee F_5 \vee F_6; \qquad (5.12)$$
$$f_1 = f_1^1 \vee f_1^2.$$

Реалізація схеми для системи (5.12) показана на рис. 5.23.

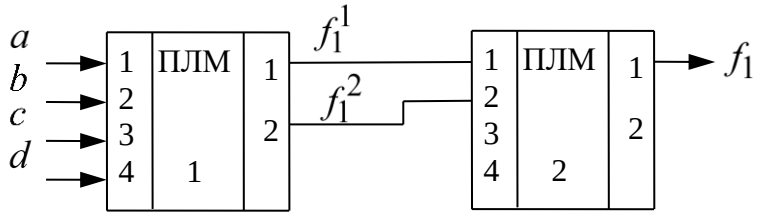


Рис. 5.23 – Дворівнева реалізація функції f_1

З наведених прикладів зрозуміло, що ПМЛ дають більше можливостей оптимізації характеристик схеми, ніж ПЛМ. Крім того, при реалізації систем булевих функцій на ПМЛ необхідно виконати роздільну мінімізацію. Це означає, що кожна функція $y_n \in Y$ мінімізується окремо. При реалізації системи Y на ПМЛ необхідно виконати спільну мінімізацію функцій системи.

Розвиток напівпровідникової технології привів до появи мікросхем, що складаються з множини макрокомірок типу ПМЛ. Як вже зазначалося раніше в § 5.1, такі мікросхеми отримали назву CPLD. Макрокомірки, що використовуються в мікросхемах CPLD, мають більш складну структуру, порівняно з ПМЛ, показаною на рис. 5.16. До складу цих макрокомірок входять мультиплексори, програмовані запам’ятовувальні елементи (тригери), а також є можливість зміни полярності вхідного сигналу. Остання властивість корисна при реалізації зворотних функцій, що ми розглядали під час обговорення монтажного АБО. У колі зворотного зв’язку CPLD зазвичай наявні як комбінаційні, так і реєстрові виходи

макрокомірок. Різні макрокомірки можуть мати різну кількість термів. Крім того, існує можливість перерозподілу термів між макрокомірками в межах деякого блоку. Методи синтезу пристроїв на CPLD характеризуються великою складністю. Для отримання оптимального за витратами обладнання результату (схеми) необхідно враховувати всі особливості конкретної мікросхеми.

При цьому постійно з’являються нові рішення, як структурні, так і схемотехнічні. Це не дає змоги отримувати оптимальні за апаратурою схеми у процесі використання тільки деяких загальних моделей ПМЛ. Отже, для фахівця в галузі інформатики є широке поле досліджень в цій області.

5.4 Синтез комбінаційних схем на БІС з архітектурою FPGA

На відміну від структур ПЛУ, розглянутих в розділах 5.2–5.3, мікросхеми програмованих користувачами вентильних матриць FPGA (Field Programmable Gate Arrays) в основному складаються з великого числа конфігурованих логічних блоків (КЛБ), розташованих по рядках і стовпцях у вигляді матриці, трасувань ресурсів (ТР), що забезпечують їхні міжз’єднання з блоками введення-виведення (БВВ). Узагальнена структура FPGA приведена на рис. 5.24.

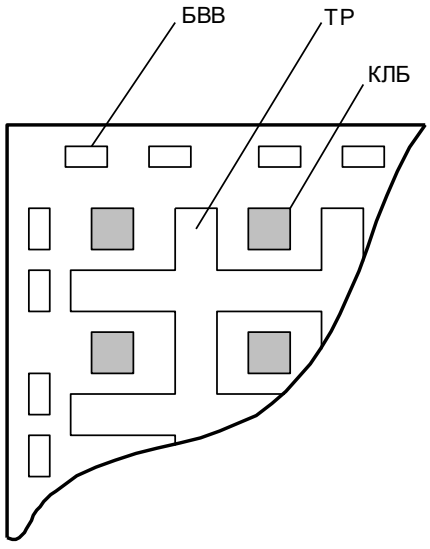


Рис. 5.24 – Узагальнена структура мікросхем FPGA

Усі блоки конфігуруються або реконфігуруються самими розробниками цифрових схем на FPGA. При конфігурації блоки налаштовуються у певний спосіб для реалізації заданої системи булевих функцій (БФ). На відміну від розглянутих раніше ПЛУ, мікросхеми FPGA містять БВВ, які мають в своєму складі тригери. Це дає змогу синтезувати пристрої з елементами пам'яті (керуючі автомати, порти введення-виведення, багатофункціональні регістри тощо). Трасувальні ресурси є ієрархічною структурою провідників, з'єднаних в шину за допомогою програмованих ключів. Множина КЛБ, з'єднаних за їхньої допомоги, реалізують КС досить великої міри складності. Всі КЛБ ідентичні і є програмованими логічними елементами, що реалізують довільну булеву функцію від S змінних, де S – число входів блока. Існує кілька типів реалізації КЛБ в різних мікросхемах FPGA, однак найчастіше використовуються логічні блоки типу LUT (Look-Up Table) на основі регістра і мультиплексора, які реалізують БФ способом, описаним у розділі 4. Структура LUT-блока для випадку $S = 2$ приведена на рис. 5.25. ОЗУ програмується розробником і служить для зберігання значень БФ під час роботи схеми.

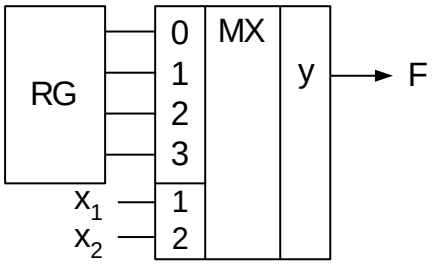


Рис. 5.25 – Узагальнена структура LUT-блока

До складу КЛБ будь-якого типу входить також тригерна пам'ять D -типу, яка може бути запрограмована на фіксацію виходу LUT. У різних моделях FPGA КЛБ зазвичай містить два блоки LUT, два тригери і два програмовані мультиплексори, що дають змогу з'єднувати виходи тригерів з входами LUT, а також комбінувати з'єднання виходів LUT з виходами і входами тригерів. Це дає змогу синтезувати на FPGA схеми з внутрішніми

зворотними зв’язками, а також реалізувати деякі функції на двох блоках LUT усередині одного КЛБ, що допомагає розвантажити лінії ТР і приводить до підвищення швидкодії синтезованої схеми.

Залежно від конкретної моделі мікросхеми FPGA число входів змінюється від 2 до 5 ($S = \overline{2,5}$). Число змінних в БФ проєктованих систем зазвичай значно більше п’яти і коливається в широких межах. Отже, для реалізації системи булевих функцій від N змінних необхідно кожну функцію представити у вигляді сукупності підфункцій від M змінних, де $M \le S$. Тобто виникає задача функціональної декомпозиції в базис із меншим числом входів. Для вирішення цього завдання можуть бути застосовані такі методи:

1. Декомпозиція за допомогою карт Карно.
2. Класичне розкладання Шеннона.
3. Аналітична декомпозиція за допомогою розкладань Шеннона.

1. Декомпозиція за допомогою карт Карно.

У тому разі, якщо аргументами функції є не більше п’яти змінних, можливість розкладання можна легко оцінити за допомогою карт Карно. Для цього на карту Карно з вихідною функцією по черзі накладаються шаблонні карти, заповнені особливим способом. Необхідно за допомогою шаблонів покрити карту у такий спосіб, щоб задана БФ була однозначно визначена перетином нулів і одиниць двох або більше шаблонів. На рис. 5.26 наведені шаблони для розкладання БФ від трьох змінних на функції $f_1 = F(a)$ і $f_2 = F(b,c)$, на рис. 5.27, відповідно, на функції $f_1 = F(b)$ і $f_2 = F(a,c)$, а на рис. 5.28 – $f_1 = F(c)$ і $f_2 = F(a,b)$. Обведені області відповідають одиницям. При накладенні шаблонів використовуються тільки ті області, які покривають одиниці на мапі вихідної функції.

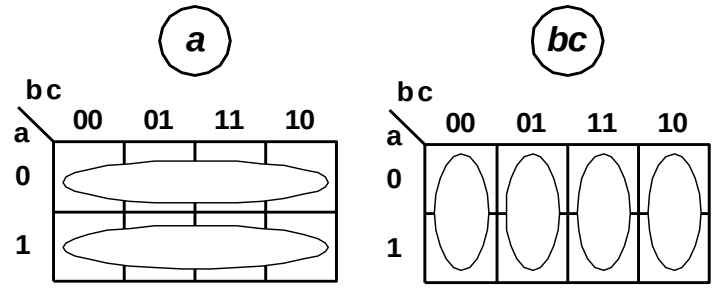


Рис. 5.26 – Шаблони для розкладання по a, bc

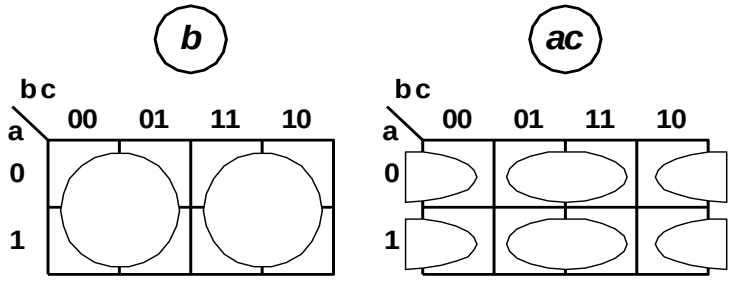


Рис. 5.27 – Шаблони для розкладання по b, ac

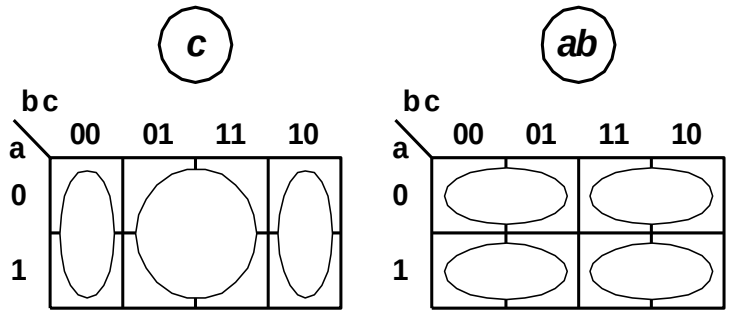


Рис. 5.28 – Шаблони для розкладання по c, ab

Шаблони можна застосовувати не тільки попарно, як показано на рисунках, а й в будь-якому поєднанні. Наприклад, якщо функція не розклалася по жодній парі шаблонів, можна спробувати застосувати такі комбінації, як: $ab + bc$, $ab + ac$, $ac + bc$. Застосування більше двох шаблонів не має сенсу для БФ трьох змінних, оскільки це призведе до її подання у вигляді функції трьох аргументів. Доведено, що з усіх 256-ти БФ трьох змінних тільки 152 функції представляються у вигляді функцій від однієї і двох змінних і 232 – від двох і двох. Решта 24 БФ, що не піддаються такому розкладанню, можна розбити на дві групи термів ДНФ

або КНФ, тобто на дві функції, що розкладаються, і застосувати шаблонне розбиття для кожної з них. Складніше йде справа з розбивкою БФ від чотирьох і більше змінних. Як показали дослідження, з 65 536 функцій, тільки 1 208 можуть бути представлені у вигляді двох функцій, кожна від незалежної пари змінних, і 51 568 функцій можуть бути представлені у вигляді двох функцій, кожна від трьох змінних, що допускають частковий перетин по одній змінній.

Розглянемо приклад декомпозиції БФ за допомогою карт Карно. Нехай БФ $F = F(a,b,c)$ задана ДНФ: $F = \bar{a}c \vee a\bar{b}\bar{c} \vee bc$. Побудуємо для неї карту Карно (рис. 5.29.):

a \ bc	00	01	11	10
0	0	1	1	0
1	1	0	1	0

Рис. 5.29 – Карта Карно функції $F(a,b,c)$

З цього рисунку видно, що розкладання можна здійснити за допомогою шаблонів, представлених на рис. 5.28, що і відображено на рис. 5.30.

a \ bc	00	01	11	10
0	0	1	1	0
1	1	0	1	0

Рис. 5.30 – Розкладання функції $F(a,b,c)$

Отже, $f_1(ab) = a\bar{b},\ f_2(c) = c$. Функція $f_3 = F$ залежить від аргументів f_1, f_2 і знаходиться по таблиці істинності (табл. 5.11). З цієї таблиці випливає, що результуюча функція є сумою по модулю двох отриманих підфункцій. Отже, можна записати, що $F = f_3 = f_1 \oplus f_2 = a\bar{b} \oplus c$.

Таблиця 5.11

Таблиця істинності функції

f_1 f_2	f_3
0 0	0
0 1	1
1 0	1
1 + 1	0

2. Класичне розкладання Шеннона.

За допомогою розкладання Шеннона будь-яку БФ $Y = Y(x_1, x_2, \ldots, x_N)$ можна представити у вигляді:

$$\begin{aligned} Y &= x_1 \,\&\, Y_1^{'}(x_2, x_3, ..., x_N) \vee \overline{x_1} \,\&\, Y_1^{''}(x_2, x_3, ..., x_N) = \\ &= x_2 \,\&\, Y_2^{'}(x_1, x_3, ..., x_N) \vee \overline{x_2} \,\&\, Y_2^{''}(x_1, x_3, ..., x_N) = \\ &= ... = x_N \,\&\, Y_N^{'}(x_1, x_2, ..., x_{N-1}) \vee \overline{x_N} \,\&\, Y_N^{''}(x_1, x_2, ..., x_{N-1}). \end{aligned} \qquad (5.13)$$

Кожна з підфункцій $Y_1^{'} - Y_N^{''}$ таким самим способом піддається розкладанню Шеннона доти, доки число аргументів її кінцевих підфункцій не дорівнюватиме S. Кожна з кінцевих підфункцій реалізується на одному КЛБ, а інші частини формули об'єднуються в групи у такий спосіб, щоб кожна група містила не більше S входів і реалізувалася на одному КЛБ.

Розглянемо приклад застосування цього методу.

Нехай БФ Y задана ДНФ $Y = \bar{a}bde \vee bcd\bar{e} \vee \bar{c}\,\bar{d}$ і $S = 3$. Виконано перше розкладання: $Y = \bar{a} \,\&\, (bde \vee bcd\bar{e} \vee \bar{c}\,\bar{d}) \vee a \,\&\, (bcd\bar{e} \vee \bar{c}\,\bar{d})$.

На цьому етапі отримані підфункції чотирьох аргументів, тому необхідно виконати ще одне розкладання, що призводить до:

$Y = \bar{a} \,\&\, [b \,\&\, (de \vee cd\bar{e} \vee \bar{c}\,\bar{d}) \vee \bar{b} \,\&\, (\bar{c}\,\bar{d})] \vee a \,\&\, [b \,\&\, (cd\bar{e} \vee \bar{c}\,\bar{d}) \vee \bar{b} \,\&\, (\bar{c}\,\bar{d})]$.

Далі за отриманою формулою будується функціональна схема в булевому базисі, і виконується угрупування вентилів за КЛБ (рис. 5.31).

Цей метод має низку істотних недоліків:

1. Метод не орієнтований на оптимізацію витрат обладнання, оскільки кожне наступне розкладання Шеннона призводить до подвоєння

числа підфункцій одних і тих самих аргументів. Число кінцевих підфункцій P визначається як:

$$P = 2^{N-S}, \tag{5.14}$$

де N – число аргументів вихідної БФ, S – кількість входів КЛБ.

2. Кінцеве угрупування може бути виконане безліччю способів, тобто виникає завдання пошуку оптимального рішення, яке в даному випадку важко формалізувати, оскільки для різних функцій і за різних значень S оптимальне рішення є окремим випадком і не залежить від способу угрупування.

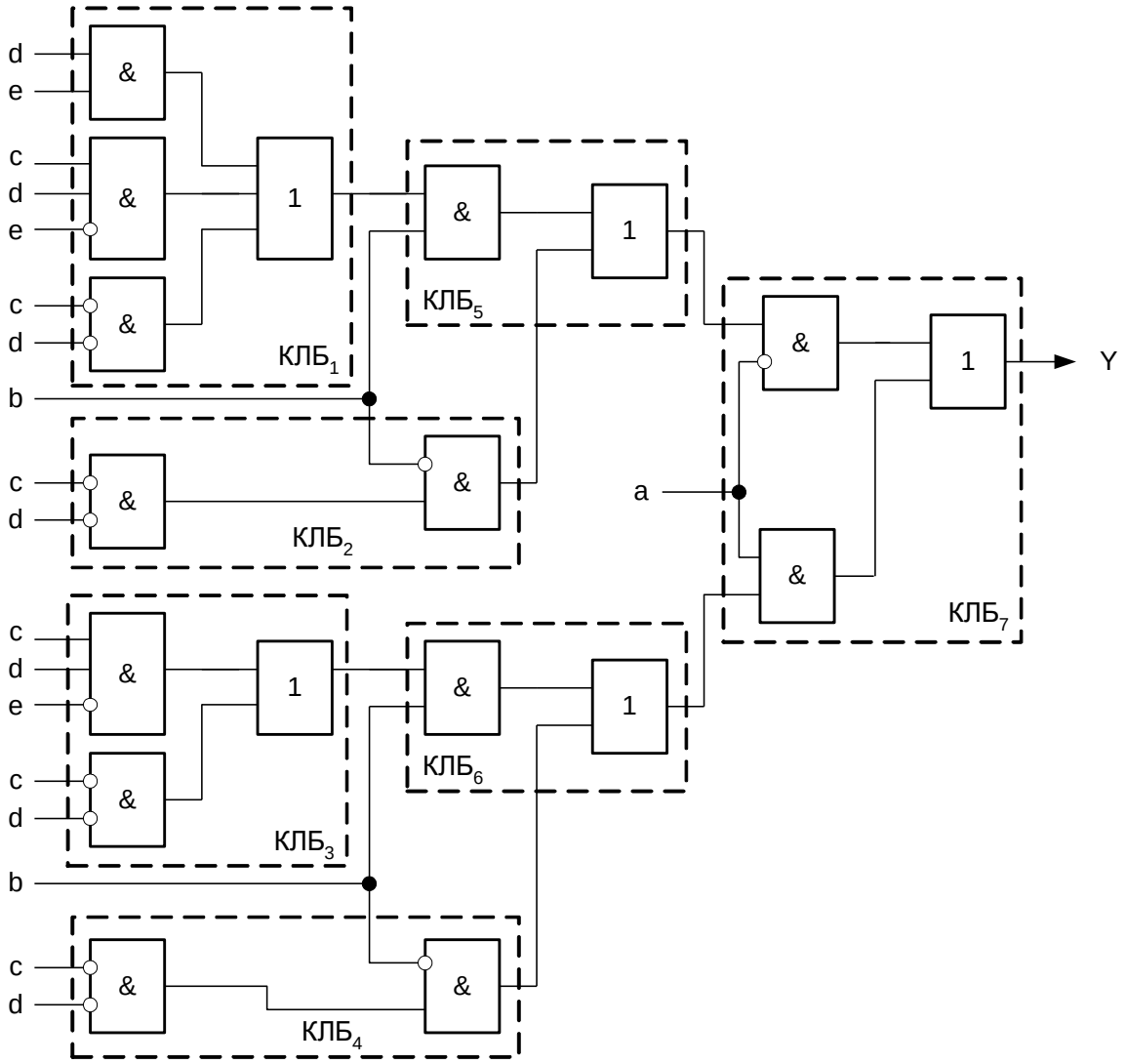


Рис. 5.31 – Реалізація функції Y з використанням розкладання Шеннона

3. Оскільки аргументи, за якими виконується черговий крок розкладання Шеннона, можна вибирати довільним способом, то для знаходження оптимальної реалізації БФ на КЛБ необхідно виконати M варіантів реалізації:

$$M = C_N^S = \frac{N!}{S!(N-S)!}. \quad (5.15)$$

Використовуючи вирази (5.14)–(5.15), можна побачити, що, наприклад, за $N = 10$ і $S = 4$ число кінцевих підфункцій P становить 64, а для вибору оптимального рішення необхідно виконати 210 декомпозицій цим методом.

Цей метод приводить до реалізації будь-якої функції на КЛБ, але число підфункцій, з яких складається реалізована функція, обчислюється цілими ступенями двійки і залежить від числа змінних розкладання. Для реалізації функції на КЛБ це найнераціональніший метод з погляду витрат апаратури.

3. Аналітична декомпозиція за допомогою розкладань Шеннона

Оскільки КЛБ може реалізувати будь-яку БФ від S змінних, то необхідно лише забезпечити подання заданої функції у вигляді мінімального числа підфункцій з числом змінних $M \le S$. Функція виражається у вигляді підфункцій таким способом, щоб перетин за аргументами різних підфункцій був мінімальним. В ідеальному випадку аргументи в підфункції не перетинаються, тоді реалізація такої функції в базисі КЛБ найбільш раціональна.

Існує безліч варіантів з'єднання КЛБ для реалізації функції, які можуть складатися з фрагментів структур двох типів, представлених на рис. 5.32 і рис. 5.33 відповідно.

Швидкодія за паралельного з'єднання КЛБ вище, однак, як показали дослідження низки авторів, більшість БФ реалізуються у вигляді послідовного з'єднання КЛБ.

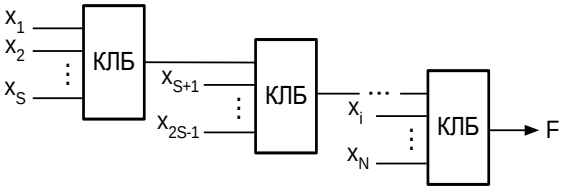


Рис. 5.32 – Послідовне з'єднання КЛБ

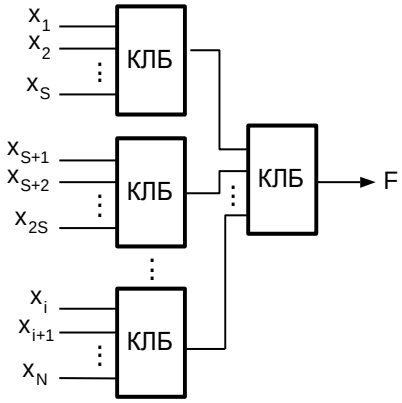


Рис. 5.33 – Паралельне з'єднання КЛБ

Для реалізації БФ на ПЛІС FPGA можна використовувати такі методи, що характеризуються параметром S.

1. Метод реалізації функцій на КЛБ з $S = 2$.

Нехай функція F є БФ від N змінних: $F = F(x_1, x_2, \ldots, x_N)$, і її необхідно реалізувати на КЛБ з $S = 2$. Оскільки заздалегідь невідомо, які аргументи функції можуть бути винесені в підфункції, виникає необхідність перевірки такої можливості для всіх можливих груп аргументів. Найбільш раціонально відразу спробувати розбити функцію F на дві підфункції з незалежними аргументами. Це приводить до реалізації функції у вигляді паралельного з'єднання КЛБ. Таку можливість пропонується перевірити за допомогою розкладання Шеннона і подальшого аналізу отриманих підфункцій. Для перевірки відмежування від функції групи i аргументів $x_1, x_2, \ldots, x_i$ запишемо:

$$F = f_1(x_1, x_2, ..., x_i) \,\&\, f_1'(x_{i+1}, x_{i+2}, ..., x_N) \vee f_2(x_1, x_2, ..., x_i) \,\&\, f_2'(x_{i+1}, x_{i+2}, ..., x_N) \vee$$
$$\vee ... \vee f_K(x_1, x_2, ..., x_i) \,\&\, f_K'(x_{i+1}, x_{i+2}, ..., x_N),$$

де $f_1', f_2', ..., f_K'$ – функції, отримані в результаті підстановки в БФ F значень аргументів $x_1, x_2, ..., x_i$, $K = 2^i$, а функції $f_1, f_2, ..., f_K$ представлені таблицею істинності (табл. 5.12).

Таблиця 5.12

Таблиця істинності функцій $f_1, f_2, ..., f_K$

$x_1\ x_2 \dots x_{i-1}\ x_i$	$f_1\ \ f_2\ \ f_3\ \dots\ f_{K-1}\ f_K$
0 0 ... 0 0	1 0 0 ... 0 0
0 0 ... 0 1	0 1 0 ... 0 0
0 0 ... 1 0	0 0 1 ... 0 0
. . .	. . .
1 1 ... 1 0	0 0 0 ... 1 0
1 + 1 ... 1 + 1	0 0 0 ... 0 1

Очевидно, що отримані функції $f_1, f_2, \dots, f_K$ мають властивість ортогональності:

$$f_1 \vee f_2 \vee f_3 \vee \dots \vee f_{K-1} \vee f_K = 1, \tag{5.16}$$

$$f_1 \ \& \ f_2 \ \& \ f_3 \ \& \dots \& \ f_{K-1} \ \& \ f_K = 0. \tag{5.17}$$

З (5.16)–(5.17) випливає:

$$f_1 \vee f_2 \vee f_3 \vee \dots \vee f_{K-1} = \overline{f_K},$$

.....

$$f_1 \vee f_2 \vee \dots \vee f_i = \overline{f_{i+1} \vee \dots \vee f_{K-1} \vee f_K}.$$

Отже, диз'юнкція будь-якої групи функцій $f_1, f_2, \dots, f_K$ дорівнює інверсії диз'юнкції решти функцій системи. Отже, розкладання $F = F_3[F_1(x_1, x_2, ..., x_i), F_2(x_{i+1}, x_{i+2}, ..., x_N)]$ має місце, якщо множину функцій $f_1', f_2', ..., f_K'$ можна розбити на дві підмножини будь-якої розмірності у такий спосіб, що:

1. У першій підмножині всі елементи рівні, у другій – дорівнюють нулю.

2. У першій підмножині всі елементи рівні, у другій – дорівнюють одиниці.

3. У першій підмножині всі елементи рівні, у другій – всі елементи протилежні елементам першої підмножини.

4. Елементи обох підмножин рівні.

Розбиття виходить після винесення загальних співмножників за дужки і заміни однієї з груп функцій $f_1, f_2, \ldots, f_K$ інверсією іншої групи. В результаті виходить функція від двох аргументів-підфункцій. За успішного розбиття кожна підфункція піддається аналогічному розкладанню доти, доки число аргументів підфункції не досягне двох.

У тому разі, якщо на якомусь етапі декомпозиції функція або підфункція не може бути представлена у вигляді двох підфункцій з незалежними аргументами, виконується розбиття на підфункції із загальними аргументами. Для цього також виконується розкладання Шеннона, однак, при аналізі підфункцій $f_1', f_2', \ldots, f_K'$ необхідно представити їх у вигляді функцій не від $N-I$, а від $N-I+K$ змінних, де K – число загальних аргументів обох підфункцій.

Розглянемо приклад застосування цього методу. Нехай функція F задана у вигляді МДНФ: $F = ab \vee ac \vee cd \vee b\bar{c}\bar{d}$. Спробуємо уявити її у вигляді: $F = f_3[f_1(ab), f_2(cd)]$. Для цього виконаємо розкладання Шеннона за змінними a і b: $F = \bar{a}\bar{b} \& (cd) \vee \bar{a}b \& (\overline{c \oplus d}) \vee a\bar{b} \& c \vee ab \& 1$. Тобто, $f_1' = cd$, $f_2' = \overline{c \oplus d}$, $f_3' = c$, $f_4' = 1$. Ці підфункції не можна розбити на дві підмножини, які відповідають методу декомпозиції, описаному вище. Згідно з методом, можна уявити функцію F також у вигляді: $F = f_3[f_1(ab), f_2(acd)]$, тобто із загальним аргументом a. Для цього складемо таблицю істинності такого вигляду (табл. 5.13). Отже, з функцій $f_1' - f_4'$ двох змінних отримуємо частково певні функції від трьох змінних, однак, їх також не можна розбити на дві підмножини для декомпозиції.

Таблиця 5.14

Розкладання Шеннона функції *F* за аргументами *a, b*

acd	$f_1^{'} = cd$	$f_2^{'} = \overline{c \oplus d}$	$f_3^{'} = c$	$f_4^{'} = 1$
000	0	1	*	*
001	0	0	*	*
010	0	0	*	*
011	1	1	*	*
100	*	*	0	1
101	*	*	0	1
110	*	*	1	1
111	*	*	1	1
ab	00	01	10	11

Вивчення функцій F із загальним аргументом b приведе до такої самої таблиці. Тепер спробуємо уявити функцію у вигляді: $F = f_3[f_1(bc), f_2(ad)]$. Для цього виконаємо розкладання Шеннона за змінними b і c: $F = \bar{b}\bar{c} \,\&\, 0 \vee \bar{b}c \,\&\, (a \vee d) \vee b\bar{c} \,\&\, (a \vee \bar{d}) \vee bc \,\&\, (a \vee d)$. Тобто, $f_1^{'} = 0$, $f_2^{'} = (a \vee d)$, $f_3^{'} = (a \vee \bar{d})$, $f_4^{'} = (a \vee d)$. Ці підфункції теж не можна розбити на дві підмножини, які відповідають методу. Уявимо функцію F у вигляді: $F = f_3[f_1(bc), f_2(acd)]$ (табл. 5.14).

Таблиця 5.14

Розкладання Шеннона функції *F* за аргументами *b, c*

acd	$f_1^{'} = 0$	$f_2^{'} = (a \vee d)$	$f_3^{'} = (a \vee \bar{d})$	$f_3^{'} = (a \vee \bar{d})$
000	0	*	1	*
001	*	*	0	*
010	0	0	*	0
011	*	1	*	1
100	0	*	1	*
101	*	*	1	*
110	0	1	*	1
111	*	1	*	1
bc	00	01	10	11

Видно, що розбиття має місце. Це випливає з того, що $f_1^{'} = 0$, а підфункції $f_2^{'}$, $f_3^{'}$ і $f_4^{'}$ можна склеїти в одну, використовуючи невизначеності: $f_2^{'} = f_3^{'} = f_4^{'} = a \vee (\overline{c \oplus d})$, причому подальша декомпозиція для функції F не потрібна, оскільки отримана підфункція вже поділена за аргументами a і c, d. Реалізація функції представлена на рис. 5.34.

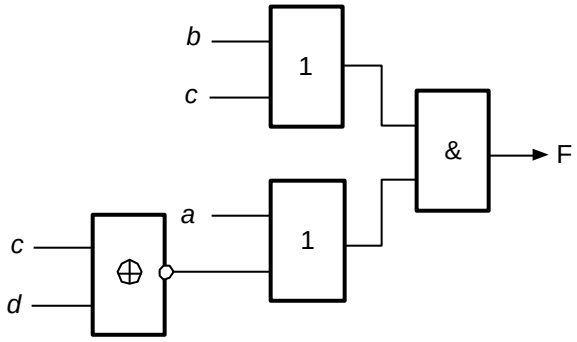


Рис. 5.34 – Реалізація функції F на КЛБ

2. Метод реалізації функцій на КЛБ з $S > 2$.

Розглянемо метод реалізації функції на КЛБ з числом входів $S = \overline{3,5}$ в разі, коли функція F є БФ від 10 змінних: $F = F(x_1, x_2, \ldots, x_{10})$ і $S = 4$. При реалізації у вигляді послідовного з'єднання КЛБ застосовується такий метод.

Для початку, необхідно спробувати розбити функцію на дві підфункції – від трьох і семи аргументів відповідно. Тоді схема для цієї функції буде виглядати у спосіб, наведений на рис. 5.35. Блок із зірочкою позначає поки ще не визначену множину КЛБ. Надалі будемо називати його ∗-блок.

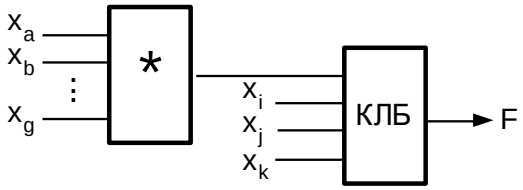


Рис. 5.35 – Початкова стадія декомпозиції функції F

Для перевірки такої можливості виконується розкладання Шеннона за змінними $x_a - x_g$. Щоб це розбиття мало місце, необхідно і достатньо, щоб виконувалася така умова: отримані підфункції розбиваються на дві підмножини довільного розміру, в кожній з яких підфункції рівні. У нашому випадку серед 128 підфункцій від змінних x_i, x_j, x_k має бути два класи однакових підфункцій. Отже, отримуємо проміжну реалізацію функції F у вигляді: $F = f_3[G(x_a, x_b, ..., x_g), x_i, x_j, x_k]$. Далі робимо те саме з підфункцією f_1, а потім і з її підфункцією доти, доки на певному етапі число аргументів не стане меншим або не дорівнювати чотирьом. У разі, якщо на якомусь етапі декомпозиція не матиме місця для всіх можливих комбінацій аргументів, необхідно спробувати виконати декомпозицію із загальними аргументами, як було описано вище. Число таких комбінацій значно вище, ніж при декомпозиції без загальних аргументів. Якщо ж і таке розкладання не матиме місця, то, можливо, ця функція може бути реалізована у вигляді паралельного з'єднання КЛБ, для реалізації якої справедлива така методика.

Насамперед, необхідно спробувати розбити функцію на дві підфункції – від трьох і семи аргументів – у такий спосіб, щоб вона виглядала, як на рис. 5.36.

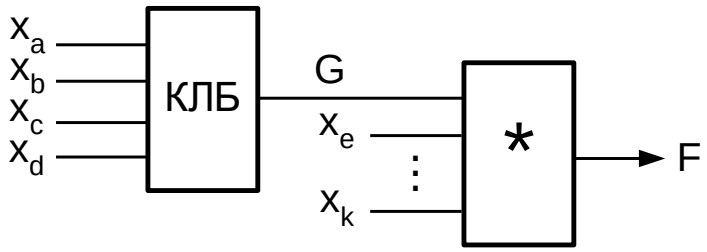


Рис. 5.36 – Початкове розбиття функції F

Для перевірки такої можливості виконується розкладання Шеннона за змінними $x_a - x_d$. Щоб це розбиття мало місце, необхідно і достатньо,

щоб отримані підфункції розбивалися на дві однорідні підмножини довільного розміру. У нашому випадку, серед 16 підфункцій від 6 змінних $x_e, x_f, \ldots, x_k$ має бути два класи однакових підфункцій. Отже, отримуємо: $F = f_3[G(x_a, x_b, x_c, x_d), x_e, x_f, x_g, x_h, x_j, x_k]$. Цей вираз відповідає проміжній реалізації функції F. Далі робимо таке. Функція, що реалізується КЛБ за змінними $x_a - x_d$, є булевою змінною для $*$-блока. Далі застосовується описана методика декомпозиції для підфункції від семи змінних доти, доки всі аргументи не будуть зібрані в групи по 4. У разі, якщо на якомусь етапі декомпозиція не матиме місця для всіх можливих комбінацій аргументів, необхідно спробувати виконати декомпозицію із загальними аргументами, як було вже описано. Якщо ж і таке розкладання не матиме місця, то ця функція може бути реалізована у вигляді послідовно-паралельного з'єднання КЛБ зі значним числом загальних аргументів на різних КЛБ. Для цього задана функція розбивається на групи термів ДДНФ або ДКНФ, кожна з яких реалізується на КЛБ описаними методами. Число таких розбиттів настільки велике, що ймовірність реалізації заданої функції на КЛБ досить велика.

Розглянемо приклад застосування методу на прикладі ДНФ БФ Y:

$$Y = abcd \vee \bar{a}b\bar{d}\,\bar{e}f \vee \bar{a}\,\bar{b}d \vee abce\bar{f} \vee a\bar{b}\,\bar{d}\,\bar{e}f \vee a\bar{c}\,\bar{d}\,\bar{e}f \vee \bar{a}\,\bar{b}e\bar{f} \vee b\bar{c}\,\bar{d}\,\bar{e}f .$$

Нехай необхідно реалізувати цю функцію на КЛБ з $S = 4$. Виконаємо розкладання Шеннона за аргументами d, e, f і зведемо значення підфункцій G_i і F_i в табл. 5.15. Підфункції F_i не можна поділити на дві множини, отже, розкладання не має місця. Виконаємо розкладання Шеннона за аргументами a, b, c і зведемо значення підфункцій G_i і F_i в табл. 5.16.

Таблиця 5.15

Розкладання функції за аргументами *d, e, f*

$d\ e\ f$	$G_i(d,e,f)$	$F_i(a,b,c)$
0 0 0	G_1	0
0 0 1	G_2	$\bar{a}b \vee a\bar{b} \vee a\bar{c} \vee b\bar{c}$
0 1 0	G_3	$abc \vee \bar{a}\bar{b}$
0 1 1	G_4	0
1 0 0	G_5	$abc \vee \bar{a}\bar{b}$
1 0 1	G_6	$abc \vee \bar{a}\bar{b}$
1 1 0	G_7	$abc \vee \bar{a}\bar{b}$
1 1 1	G_8	0

Очевидно, що розбиття має місце, і функцію Y можна записати у вигляді:

$$Y = (d \vee e\bar{f}) \,\&\, (G_1 \vee G_2 \vee G_8) \vee (\bar{d}\,\bar{e}f) \,\&\, (G_3 \vee G_4 \vee G_5 \vee G_6 \vee G_7) =$$
$$= (d \vee e\bar{f}) \,\&\, (G_1 \vee G_2 \vee G_8) \vee (\bar{d}\,\bar{e}f) \,\&\, (\overline{G_1 \vee G_2 \vee G_8}) = \quad .$$
$$= (d \vee e\bar{f}) \,\&\, G \vee \bar{d}\,\bar{e}f \,\&\, \bar{G}, \text{де } G = G_1 \vee G_2 \vee G_8 = \bar{a}\,\bar{b} \vee abc$$

Таблиця 5.16

Розкладання функції за аргументами *a, b, c*

$a\ b\ c$	$G_i(a,b,c)$	$F_i(d,e,f)$
0 0 0	G_1	$d \vee e\bar{f}$
0 0 1	G_2	$d \vee e\bar{f}$
0 1 0	G_3	$\bar{d}e\bar{f}$
0 1 1	G_4	$\bar{d}e\bar{f}$
1 0 0	G_5	$\bar{d}e\bar{f}$
1 0 1	G_6	$\bar{d}e\bar{f}$
1 1 0	G_7	$\bar{d}e\bar{f}$
1 1 1	G_8	$d \vee e\bar{f}$

З’єднання КЛБ для реалізації функції Y наведено на рис. 5.37.

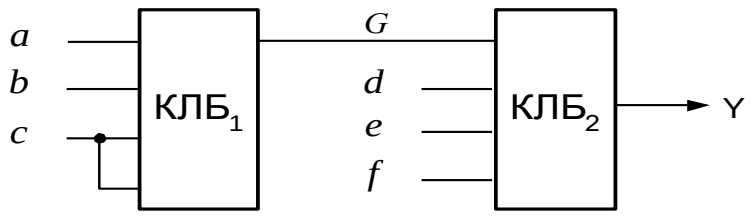


Рис. 5.37 – Реалізація функції Y на КЛБ

Список літератури до розділу 5

1 Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

2 DeMicheli G. Synthesis and Optimization of Digital Circuits. New York: McGraw Hill, 1994. 579 p.

3 Соловьёв В. В. Проектирование цифровых систем на основе программируемых логических интегральных схем. Москва: Горячая линия – Телеком, 2001. 636 с.

4 Соловьёв В. В., Климович А. Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. Москва: Горячая линия – Телеком, 2008. 376 с.

5 Гаврилов М. А. Избранные труды. Теория релейных устройств и конечных автоматов. Москва: Наука, 1983. 280 с.

6 Шестаков В. И. Алгебраический метод синтеза многотактных релейных систем. *Доклады ФН СССР*. 1954. Т. 99. № 6. С. 987–990.

7 Jenkins J. Designing with FPGAs and CPLDs. New Jersey: PTR Prentice Hall, 1994. 273 p.

6 АБСТРАКТНИЙ АВТОМАТ

6.1 Абстрактний автомат як модель послідовних схем

У комбінаційних схемах вихідний сигнал залежить тільки від набору вхідних змінних. Наприклад, якщо в ДНФ функції y_1 входить терм $F_1 = abc$, то в будь-який момент часу $y_1 = 1$ за виконання рівності $a = b = c = 1$. Однак існує клас пристроїв, вихідна інформація в яких залежить від вхідних сигналів, що надходять в усі попередні моменти часу. Такі пристрої називаються послідовними схемами. Характерним прикладом послідовних схем є лічильник. Якщо в деякий момент часу t в лічильнику знаходиться число 0, то при надходженні одиниці на вхід (момент $t+1$) в лічильнику буде 1. У момент часу $t+2$ при надходженні одиниці на вхід на виході буде 2. Якщо ж у момент часу $t+2$ надходить нуль, то лічильник, як і раніше, генерує число 1. Отже, вихід лічильника залежить від деякої передісторії надходження вхідних сигналів в часі. До цього ж класу належать системи управління, що реалізують деякі алгоритми.

Перші елементи теорії послідовних схем з'явилися в роботах Дж. Хаффмана в 1954 р. Основними авторами цієї теорії були Стівен Кліні, Едвард Мур, Джордж Мілі, М. А. Гаврилов, В. М. Глушков. В роботі Хлін «Representation of events in nerve sets and finite automata» (1956 р.) був введений термін «кінцевий автомат» для послідовних схем. Американські вчені Мілі (1955) і Мур (1956) розглянули два типи автоматів, названих пізніше їхніми іменами. Теорія кінцевих автоматів була доведена до рівня практичного застосування в роботах академіка В. М. Глушкова (1923–1982 рр.), що вийшли в 1960–1962 рр.

У роботах Глушкова була доведена теорема про структурну повноту, яка стала основою для переходу від абстрактного уявлення

послідовних пристроїв (абстрактний автомат) до деякої схеми, що реалізує задану поведінку (структурний автомат). У цьому розділі розглядаються основи абстрактної теорії автоматів.

Абстрактний автомат є математичною моделлю цифрового пристрою і задається вектором:

$$S = \langle A, Z, W, \delta, \lambda, a_1 \rangle. \tag{6.1}$$

У вираз (6.1) входять такі компоненти: $A = \{a_1, \ldots, a_M\}$ – множина внутрішніх станів автомата; $Z = \{z_1, \ldots, z_F\}$ – множина вхідних сигналів; δ – функція переходів, що реалізує відображення множини $D_\delta \subseteq A \times Z$ в A ($a_s = \delta(a_m, z_f)$, де $a_m, a_s \in A$, $z_f \in Z$); λ – функція виходів, що реалізує відображення множини $D_\lambda \subseteq A \times Z$ в W ($\omega_g = \lambda(a_m, z_f)$), де $a_m \in A$, $z_f \in Z$, $\omega_g \in W$); $a_1 \in A$ – початковий стан автомата. Множини A, Z, W іноді називаються алфавітами абстрактного автомата.

Домовимося надалі розглядати тільки кінцеві автомати (тобто множини A, Z, W є кінцевими) з явно виділеним початковим станом a_1. Стан автомата є пам'яттю про вхідні сигнали, що надходили на вхід автомата в попередні моменти часу. Це поняття введено з огляду на те, що вихідні сигнали операційних і керуючих автоматів залежать не тільки від вхідних сигналів в момент часу t, але і від вхідних сигналів в попередні моменти часу $0,1,\ldots,t-1$. При переході до абстрактного автомата це приводить до такої формули

$$\omega(t) = \lambda(z(0), z(1), \ldots, z(t)), \tag{6.2}$$

де $\omega(t)$ – вихідний сигнал в момент часу t, $z(i)$ – вхідний сигнал в момент часу $i = 0,1,\ldots,t$. Формули виду (6.2) є надзвичайно громіздкими і не дають змоги компактно задати закон функціонування автомата. Якщо стан $a(t)$ є деякою функцією вхідних сигналів $z(0), \ldots, z, z(t-1)$, то формулу (6.2) можна спростити до формули

$$\omega(t) = \lambda(a(t), z(t)). \tag{6.3}$$

Стан $a(t)$ у формулі (6.3) може бути визначено за допомогою оригінального стану $a_1 = a(0)$ і функції переходів λ:

$$a(t) = \lambda(a(t-1), z(t-1)) = \lambda(\lambda(a(t-2), z(t-2)), z(t-1)) = \\ = \lambda(\lambda \cdots (\lambda(a_0, z(0)), z(1), \cdots), z(t-1)). \tag{6.4}$$

Формула (6.4) вказує на те, що для визначення стану в будь-який момент t досить знати початковий стан (а він унікальний!) і послідовність вхідних сигналів $z(0), z(1), \ldots, z, z(t-1)$.

Абстрактний автомат (рис. 6.1) має один вхідний канал і один вихідний канал.

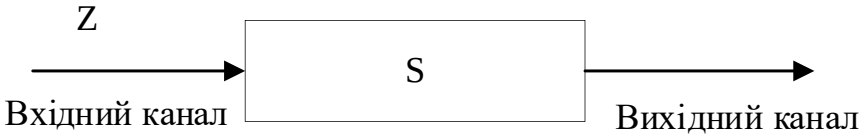


Рис. 6.1 – Абстрактний автомат

У момент часу $t = 0$ автомат знаходиться в початковому стані $a_1 \in A$. Нехай в момент часу t автомат знаходиться в стані $a(t) = a_m$, і на вхід автомата надходить буква вхідного алфавіту $z(t) = z_f$. Автомат формує вихідний сигнал $\omega(t) = \lambda(a_m, z_f) = \omega_g$ і переходить в стан $a(t+1) = \delta(a_m, z_f) = a_s$. Отже, абстрактний автомат перетворює літери вхідного алфавіту в букви вихідного алфавіту. Це означає, що абстрактний автомат є найбільш загальною моделлю пристрою переробки інформації.

Найбільшого поширення на практиці набули три типи автоматів:

1. Автомат Мілі, що задається рівняннями

$$a(t+1) = \delta(a(t), z(t)); \\ \omega(t) = \lambda(a(t), z(t)). \tag{6.5}$$

2. Автомат Мура, що задається рівняннями

$$a(t+1) = \delta(a(t), z(t)); \\ \omega(t) = \lambda(a(t)). \tag{6.6}$$

3. Комбінаційна схема, що задається рівняннями

$$A = \{a_1\};$$
$$\omega(t) = \lambda(z(t)). \tag{6.7}$$

Методи завдання, мінімізації та синтезу комбінаційних схем розглянуті раніше, тому розглянемо докладніше методи, пов'язані з автоматами Мілі і Мура. Обмежимося розглядом повністю визначених автоматів, для яких рівняння (6.5)–(6.6) визначені для всіх пар декартового добутку множин станів і вхідних букв: $A \times Z = \{(a_1, z_1), (a_1, z_2), \ldots, (a_1, z_F), \ldots, (a_M, z_F)\}$.

Існує два основні методи завдання автоматів: табличний і графічний.

1. Табличне завдання автомата Мілі. Автомат Мілі задається таблицями переходів і виходів, які в загальному випадку представлені табл. 6.1 і 6.2 відповідно. Рядки цих таблиць характеризуються вхідними сигналами $z_f \in Z$, а стовпці – станами автомата $a_m \in A$. На перетині рядка z_f і стовпця a_m в таблиці переходів записується стан переходу $a_s = \delta(a_m, z_f)$, а в таблиці виходів – вихідний сигнал $\omega_g = \lambda(a_m, z_f)$.

2.

Таблиця 6.1

Загальний вигляд таблиці переходів автомата Мілі

a_m / z_f	α_1	$\cdots$	α_M
z_1	$f(\alpha_1, z_1)$		$f(\alpha_M, z_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
z_F	$f(\alpha_1, z_F)$		$f(\alpha_M, z_F)$

Таблиця 6.2

Загальний вигляд таблиці виходів автомата Мілі

a_m / z_f	α_1	$\cdots$	α_M
z_1	$\lambda(\alpha_1, z_1)$		$\lambda(\alpha_M, z_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
z_F	$\lambda(\alpha_1, z_F)$		$\lambda(\alpha_M, z_F)$

Таблиці переходів і виходів в сукупності містять всю інформацію про абстрактні автомати, тобто визначають всі компоненти множини (6.1), включно з рівнянням (6.5).

Наприклад, з табл. 6.3 і 6.4 для автомата S_1 маємо $A=\{a_1, a_2, a_3\}$, $Z=\{z_1, z_2\}$, $W=\{\omega_1, \omega_2, \omega_3\}$.

Таблиця 6.3

Таблиця переходів автомата Мілі S_1

a_m / z_f	α_1	α_2	α_3
z_1	α_1	α_3	α_1
z_F	α_3	α_1	α_2

Таблиця 6.4

Таблиця виходів автомата Мілі S_1

a_m / z_f	α_1	α_2	α_3
z_1	ω_1	ω_3	ω_1
z_F	ω_3	ω_1	ω_2

З табл. 6.3 маємо, наприклад, $\delta(a_2, z_1)=a_3$, $\delta(a_1, z_1)=a_2$. З табл. 6.4 маємо, наприклад, $\lambda(a_1, z_1)=\omega_1$, $\lambda(a_3, z_2)=\omega_2$.

Оскільки рядки і стовпці таблиць переходів і виходів позначені однаково, всю інформацію про автомат Мілі можна задати однією суміщеною таблицею переходів і виходів. У цій таблиці (табл. 6.5 для автомата S_1) на перетині рядка z_f і стовпця a_m знаходяться і стан переходу $a_s = \delta(a_m, z_f)$, і вихідний сигнал $\omega_g = \lambda(a_m, z_f)$.

Таблиця 6.5

Об'єднана таблиця переходів і виходів автомата Мілі S_1

a_m / z_f	α_1	α_2	α_3
z_1	α_2 / ω_1	α_3 / ω_3	α_1 / ω_1
z_2	α_3 / ω_3	α_1 / ω_1	α_2 / ω_2

2. Графічне завдання автомата Мілі. Граф автомата є орієнтованим зв'язним графом, вершини якого відповідають станам автомата, а ребра (дуги) – переходам між станами. В автоматі Мілі дуга, відповідна переходу з a_m в a_s під впливом сигналу z_f, характеризується вихідним сигналом $\omega_g = \lambda(a_m, z_f)$ і вхідним сигналом z_f (рис. 6.2). Граф автомата Мілі S_1 (рис. 6.3) має три вершини і шість дуг, що відповідає виразу $M \cdot F$.

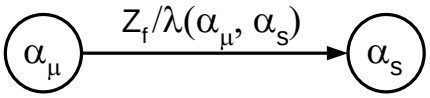


Рис. 6.2 – Фрагмент графа автомата Мілі

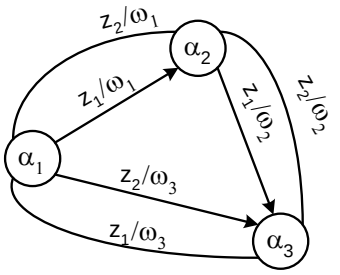


Рис. 6.3 – Граф автомата Мілі S_1

3. Табличне завдання автомата Мура. Як випливає з рівнянь (6.6), вихідний сигнал автомата Мура залежить тільки від його стану. Отож, автомат Мура задається однією зазначеною таблицею переходів (табл. 6.6).

Таблиця 6.6

Загальний вигляд зазначеної таблиці переходів автомата Мура

ω_g	$\lambda(\alpha_1)$		$\lambda(\alpha_M)$
α_m / z_f	α_1	…	α_M
z_1	$\delta(\alpha_1, z_1)$	…	$\delta(\alpha_M, z_1)$
…	…	…	…
z_F	$\delta(\alpha_1, z_F)$	…	$\delta(\alpha_M, z_F)$

Наприклад, з табл. 6.7 випливають всі елементи множини S автомата Мура S_2: $A=\{a_1,a_2,a_3\}$, $M=3$, $Z=\{z_1,z_2\}$, $F=2$, $W=\{w_1,w_2\}$, $G=2$, $\delta(a_1,z_1)=a_2$, $\lambda(a_1)=\lambda(a_3)=w_2$ тощо.

Таблиця 6.7

Зазначена таблиця переходів автомата Мура S_2

ω_g	ω_2	ω_1	ω_2
α_m / z_f	α_1	α_2	α_3
z_1	α_2	α_3	α_1
z_2	α_3	α_2	α_3

4. Графічне завдання автомата Мура. Оскільки вихідні сигнали автомата Мура залежать тільки від вхідних станів, то кожній вершині графа автомата (рис. 6.4) ставиться у відповідність вихідний сигнал.

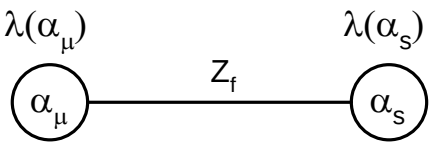


Рис. 6.4 – Фрагмент графа автомата Мура

Дуга графа автомата Мура характеризується тільки вхідним сигналом. Граф автомата Мура S_2 наведено на рис. 6.5.

Як видно з рис. 6.5, деякі дуги утворюють петлі. Стани відповідних петель називаються очікуючими. Автомат буде залишатися в режимі цього стану доти, доки не зміниться значення вхідного сигналу. Наприклад, для стану a_2 маємо $\delta(a_2, z_2) = a_2$, тобто автомат S_2 буде перебувати в стані a_2, поки вхідний сигнал не дорівнюватиме z_1.

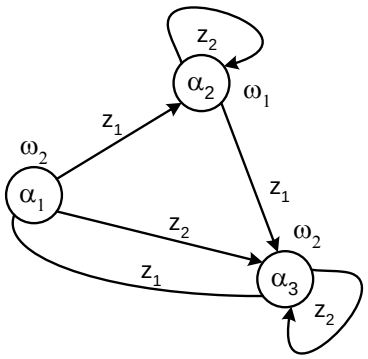


Рис. 6.5 – Граф автомата Мура S_2

6.2 Синтез абстрактних автоматів

Домовимося розуміти під синтезом абстрактних автоматів завдання закону функціонування автомата. Метод синтезу абстрактного автомата охоплює такі етапи:

1. Формування таблиці істинності комбінаційної схеми, яка виконує аналогічну обробку паралельної інформації.

2. Формування вхідного і вихідного алфавіту.

3. Формування графа автомата.

Розглянемо деякі приклади синтезу абстрактних автоматів.

Приклад 6.1. Синтезувати абстрактний автомат Мілі S_3, на вхід якого надходить трирозрядний послідовний код $x_1x_2x_3$, починаючи з розряду x_1. Якщо код містить дві послідовні одиниці, то необхідно сформувати сигнал $y_1 = 1$, в іншому разі формується сигнал $y_1 = 0$.

Побудуємо таблицю істинності для відповідної паралельної комбінаційної схеми (табл. 6.8), тобто схеми, на вхід якої сигнали $x_1x_2x_3$ надходять одночасно.

Таблиця 6.8

Таблиця істинності паралельної КС

x_1	x_2	x_3	y_3	x_1	x_2	x_3	y_1
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	1

Щоб отримати вхідний алфавіт, проаналізуємо стовпці $x_1 - x_3$ таблиці. У кожному стовпці стоїть або 0, або 1. Отже, вхідний алфавіт складається з двох символів: z_1 відповідає $x_l = 0$, z_2 відповідає $x_l = 1$, де $l = 1,2,3$.

Інформація на вхід схеми надходить послідовно, тому, наприклад, змінна x_1 не дає змоги визначити вихідний сигнал. Якщо протягом двох тактів надійшла комбінація 00, то необхідно встановити $y_1 = 0$, а якщо 11 – то $y_1 = 1$. Якщо надходить комбінація 01, то стан виходу невідомий. Отже, вихідний алфавіт автомата S_3 містить 3 літери: w_1, якщо $y_1 = 0$, w_2, якщо $y_1 = 1$, і w_3, якщо вихідний сигнал ще не визначений. Отже, для автомата S_3 маємо $Z = \{z_1, z_2\}$, $W = \{w_1, w_2, w_3\}$.

Розглянемо процес побудови графа автомата S_3. У момент часу $t = 0$ автомат знаходиться у стані a_1 (рис. 6.6а).

У момент часу $t = 0$ на вхід автомата можуть надійти сигнали z_1 або z_2. Якщо $Z(0) = z_1$, то $x_1 = 0$, і це відповідає рядкам 1–4 таблиці істинності. У рядках 1–4 функція y_1 може мати будь-яке значення, отже, перехід (a_1, z_1) повинен характеризуватися невизначеністю (рис. 6.6б). Аналогічно, якщо $Z(0) = z_2$, то $x_1 = 1$, що відповідає рядкам 5–8 табл. 6.8.

Цей перехід також характеризується вихідним символом, що відповідає невизначеності (рис. 6.6б). Отже, маємо такі переходи $a(1) = a_2 = \delta(a_1, z_1)$ або $a(1) = a_3 = \delta(a_1, z_2)$, і виходи автомата $\lambda(a_1, z_1) = \lambda(a_1, z_2) = w(0) = w_3$ (рис. 6.6б).

Якщо в момент часу $t = 1$ автомат знаходиться у стані a_2, то сигнал z_1 відповідає ситуації $x_1 x_2 = 00$, оскільки a_2 відповідає історії $x_1 = 0$. В цьому разі комбінація вхідних сигналів (рядки 1 або 2 таблиці істинності) не може містити двох послідовних одиниць, і необхідно сформувати сигнал w_1.

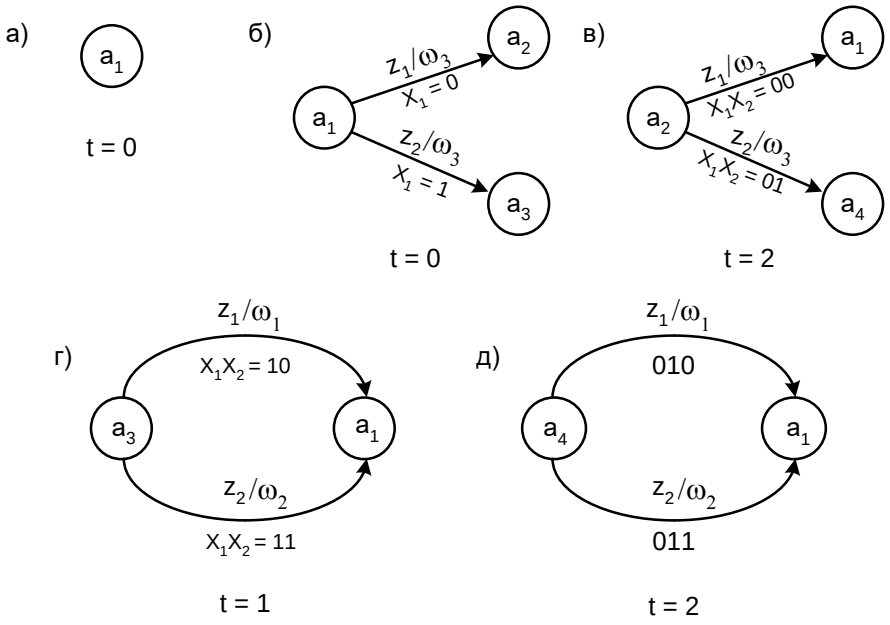


Рис. 6.6 – Процес побудови графа автомата S_3

Розпізнавання вхідної комбінації завершено, і автомат повинен повернутися в стан a_1, щоб мати можливість обробляти наступну вхідну послідовність (рис. 6.6в). Якщо ж на вхід автомата надходить сигнал z_2, то це відповідає ситуації $x_1 x_2 = 01$ (рядки 3 і 4), яка є невизначеною. Автомат переходить у стан a_4, що відповідає набору 01, і формує сигнал w_3, що відображено на рис. 6.6в. Природно, що вхідні букви можуть надходити в таких послідовностях, які визначать іншу зміну станів.

Якщо в момент часу $t=1$ автомат знаходиться в стані a_3 (ситуація $x_1=1$), то по вхідному сигналу z_1 ($x_1x_2=10$) автомат переходить в стан a_1 і формує вихідний сигнал w_1, що відповідає рядкам 5 і 6 вихідної таблиці істинності (рис. 6.6г). Якщо вхідний сигнал $z(1)=z_2$, то це відповідає рядками 7 і 8 табл. 6.8. В цьому разі автомат формує сигнал w_2 і переходить у стан a_1(рис. 6.6г). Якщо автомат знаходиться у стані a_4 ($t=2$), то надходження літери z_1 означає набір 010, а надходження z_2 – 011 (рис. 6.6д).

Тепер можна зібрати повний граф абстрактного автомата S_3, поєднавши окремі підграфи (рис. 6.6), починаючи зі стану a_1. Процес такої збірки не представляє особливих труднощів. Спочатку малюється граф, отриманий для переходів зі стану a_1 (рис. 6.6б). Тепер до станів a_2 і a_3 домальовують відповідні підграфи, представлені на рис. 6.6в і рис. 6.6г. Далі вершина a_4 розглядається як початок підграфа, який показаний на рис. 6.6д. Цей підграф приєднується до вже отриманого графа. Після цього приєднання процес побудови графа автомата закінчується. Граф автомата Мілі S_3 наведено на рис. 6.7, а поєднана таблиця переходів і виходів показана в табл. 6.9.

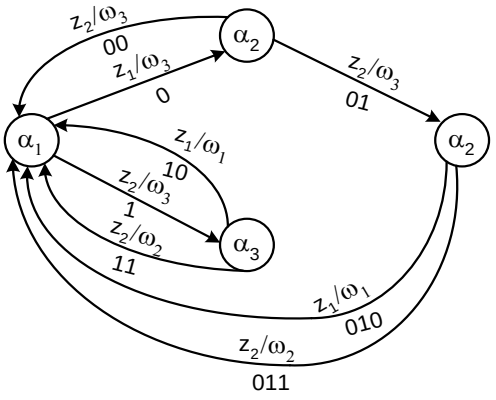


Рис. 6.7 – Граф автомата Мілі S_3

Таблиця 6.9

Поєднана таблиця переходів і виходів автомата Мілі S_3

a_m / z_f	α_1	α_2	α_3	α_4
z_1	α_2 / ω_3	α_1 / ω_1	α_1 / ω_1	α_1 / ω_1
z_2	α_3 / ω_3	α_4 / ω_3	α_1 / ω_2	α_1 / ω_2

З аналізу рис. 6.7. і табл. 6.9 випливає, що множина станів автомата S_3 містить чотири елементи: $A = \{a_1, a_2, a_3, a_4\}$, $M = 4$.

Приклад 6.2. Синтезувати абстрактний автомат Мура S_4, що виконує ті самі функції, що і автомат Мілі S_3.

Очевидно, що таблиця істинності паралельної КС для автомата S_4 збігається з табл. 6.8, а алфавіти Z і W автомата S_4 будуть такими: $Z = \{z_1, z_2\}$, $W = \{w_1, w_2, w_3\}$.

Розглянемо процес побудови графа автомата S_4. У момент часу $t = 0$ автомат знаходиться в початковому стані $a(0) = a_1$. Оскільки вхідні комбінації ще не зрозумілі, стан a_1 характеризується вихідним сигналом w_3. За сигналом z_1 ($x_1 = 0$) автомат переходить у стан a_2, для якого вихід не визначений, за сигналом z_2 – в стан a_3, в якому вихід автомата також не визначений (рис. 6.8а).

Якщо в момент часу $t = 1$ автомат знаходиться у стані a_2, то надходження літери z_1 означає ситуацію 00, за якою автомат переходить у стан a_4, позначений вихідним сигналом w_1 ($y_1 = 0$). Якщо на вхід автомата надходить сигнал z_2, то це відповідає ситуації 01, для якої вихід ще не очевидний. Автомат переходить у стан a_5, що відповідає w_3 (рис. 6.8б). Якщо $a(1) = a_3$, то за сигналом z_1 автомат формує ознаку w_1 у стані a_6, а за сигналом z_2 – ознаку w_2 у стані a_7 (рис. 6.8в).

Якщо $a(2) = a_5$, то сигнал z_1 визначає набір 010, тому автомат переходить у стан a_8 з виходом w_1. Якщо вхідний сигнал дорівнює z_2, то це визначає набір 011, тому автомат переходить у стан a_9 і формує сигнал w_2 (рис. 6.8г).

Підграфи для станів a_4, $a_6 - a_9$ збігаються, оскільки ці стани відповідають певним ситуаціям. У всіх випадках автомат переходить у стан a_1 незалежно від вхідних сигналів (рис. 6.8д–е). Об'єднання фрагментів дає граф автомата S_4 (рис. 6.9).

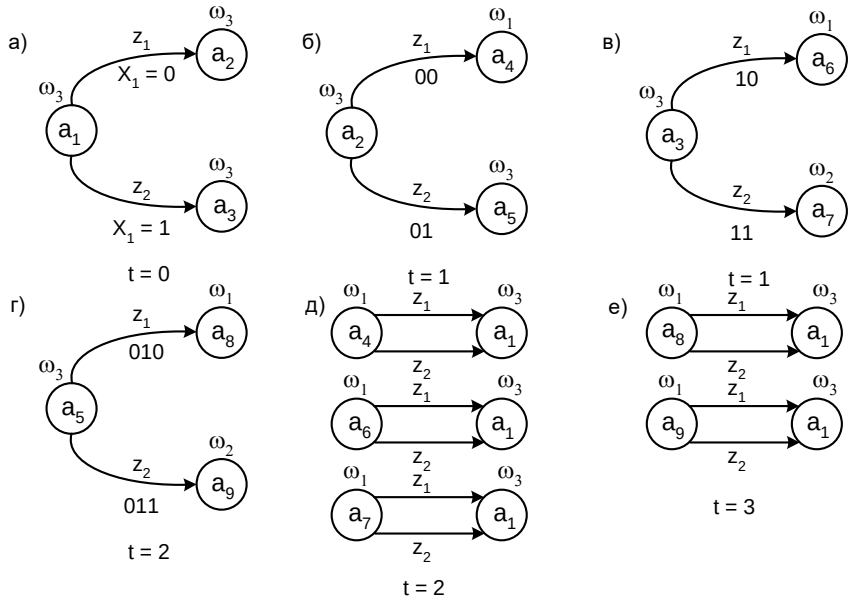


Рис. 6.8 – Процес побудови графа автомата S_4

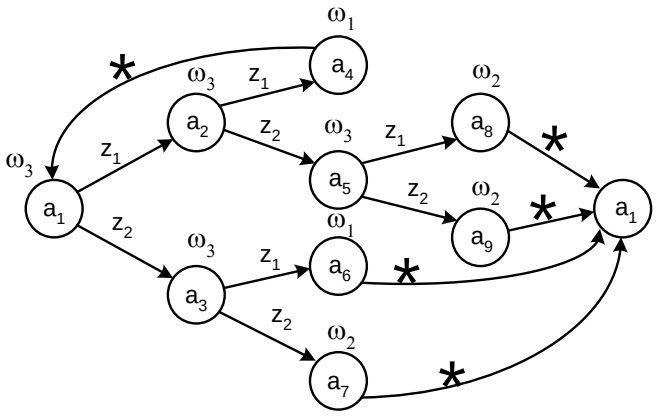


Рис. 6.9 – Граф автомата Мура S_4

Граф на рис. 6.9 містить два стани a_1, що зроблено для більш наочного зображення. При цьому один із символів a_1 відповідає початковому стану, а другий – кінцевому. Знак * над дугою означає, що перехід не залежить від значень вхідних сигналів.

З цього графа (рис. 6.9) будується зазначена таблиця переходів автомата Мура S_4 (табл. 6.10).

Таблиця 6.10

Зазначена таблиця переходів автомата Мура S_4

ω_g	ω_3	ω_3	ω_3	ω_1	ω_3	ω_1	ω_2	ω_1	ω_2
a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9
z_1	α_2	α_4	α_6	α_1	α_8	α_1	α_1	α_1	α_1
z_2	α_3	α_5	α_7	α_1	α_9	α_1	α_1	α_1	α_1

Автомати S_3 і S_4 виконують одну і ту саму функцію, тобто в цьому сенсі вони є еквівалентними. З аналізу графів автоматів S_3 і S_4 можна зробити висновки, що автомат Мілі містить менше станів, ніж еквівалентний автомат Мура, а також, що автомат Мілі формує вихідні сигнали на один такт швидше, ніж еквівалентний автомат Мура. Ці висновки справедливі і в загальному випадку.

З аналізу графа автомата S_3 випливає, що для станів a_3 і a_4 за однакових вхідних сигналів формуються однакові вихідні сигнали, і відбувається перехід у стан a_1, тобто

$$\lambda(a_3, z_1) = \lambda(a_4, z_1) = w_1,$$

$$\lambda(a_3, z_2) = \lambda(a_4, z_2) = w_2,$$

$$\delta(a_3, z_1) = \delta(a_4, z_1) = a_1,$$

$$\delta(a_3, z_2) = \delta(a_4, z_2) = a_2.$$

Отже, стани a_3 і a_4 повністю ідентичні, і один з них можна виключити з графа автомата. Виключимо, наприклад, стан a_4, що приведе

до мінімізації графа (рис. 6.10). Оскільки тепер стан a_3 замінює стан a_4, то всі переходи з $a_s = a_4$ замінюються на переходи з $a_5 = a_3$. Наприклад, перехід з a_2 по z_2 в мінімізованому графі відбувається у стан a_3.

Аналогічно в автоматі Мура S_4 стани a_4, a_8, a_6 ідентичні станам a_7, a_9. Облік цієї обставини приводить до мінімізації графа автомата Мура S_4 (рис. 6.11), в якому група станів замінюється станом з найменшим індексом, що входить в цю групу.

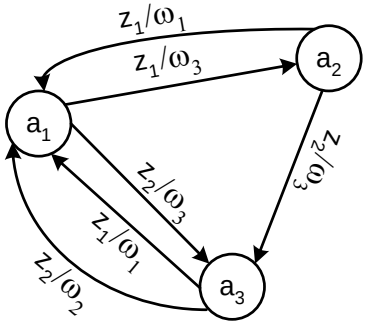


Рис. 6.10 – Мінімізований граф автомата Мілі S_3

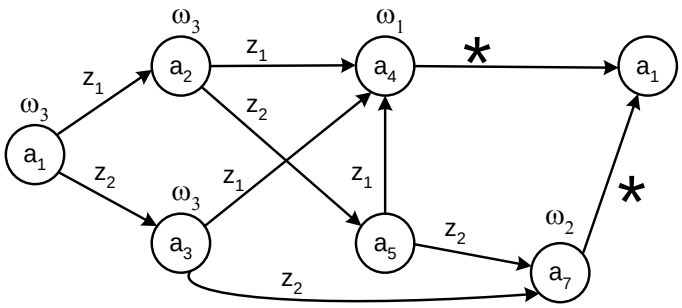


Рис. 6.11 – Мінімізований граф автомата Мура S_4

Аналіз графа на рис. 6.11 показує, що стани a_3 і a_5 також ідентичні, і один з них можна видалити, наприклад, стан a_5. Відтак, мінімальний граф автомата S_4 (рис. 6.12) буде мати тільки $M = 5$ станів.

Чим менше станів має граф автомата, тим простіше схема відповідного цифрового пристрою. Однак мінімізація на основі аналізу графа автомата є трудомісткою. Крім того, людина завжди може

помилитися. Для мінімізації автоматів існують формальні методи, які завжди дають змогу отримувати автомати з мінімальним числом станів.

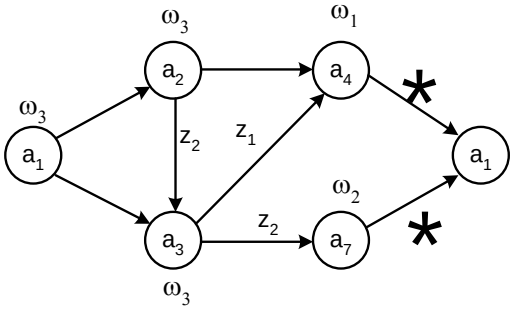


Рис. 6.12 – Мінімальний граф автомата Мура S_4

6.3 Мінімізація абстрактних автоматів

Класичні методи мінімізації абстрактних автоматів Мілі були запропоновані Д. Ауфенкампом та Ф. Хоном і засновані на знаходженні еквівалентних станів автоматів. Стани $a_m, a_s \in A$ називаються еквівалентними, якщо $\lambda(a_m, B) = \lambda(a_s, B)$, де B – деяка довільна послідовність літер вхідного алфавіту, що називається словом. Стани a_m, $a_s \in A$ називаються i-еквівалентними, якщо $\lambda(a_m, B_i) = \lambda(a_s, B_i)$, де B_i – вхідна довжина слова i $(i = 1,2,\ldots)$.

Основна ідея алгоритму мінімізації полягає в розбитті множини станів A на класи еквівалентних станів і заміні всіх станів будь-якого класу одним представником цього класу. Отже, в основі алгоритму мінімізації полягає пошук необхідного розбиття. Цей алгоритм виконується у такий спосіб.

Знайдемо розбиття $\Pi_1, \Pi_2, \ldots, \Pi_k$ множини A на класи 1-, 2-, ..., k-еквівалентних станів. Якщо на $k+1$-му кроці отримуємо $\Pi_k = \Pi_{k+1}$, то розбиття Π знайдено $(\Pi = \Pi_k)$. Число кроків на цьому етапі не перевищує $M-1$, де M – число станів.

У кожному класі розбиття Π вибирають між окремими елементами, які утворюють множину A' станів мінімального автомата $S' = \{A', Z, W, \delta', \lambda', a'_1\}$, еквівалентного вихідного абстрактного автомата

$S = \{A, Z, W, \delta, \lambda, a_1\}$. Отже, мінімізується число станів автомата, що в подальшому дає змогу зменшити апаратурні витрати.

Функції δ' і λ' визначаються на множини $A' \times Z$, для чого в таблицях переходів і виходів викреслюють стовпці, що відповідають станам $a_m \in A$, а всередині таблиці переходів стани $a_m \in A$ замінюють представниками класу еквівалентності, до яких вони належать.

Розглянемо приклад застосування цього алгоритму до автомата Мілі S_5, заданого суміщеною таблицею переходів (табл. 6.11).

Таблиця 6.11

Зазначена таблиця переходів автомата Мура S_4

a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}
z_1	α_2/ω_3	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1	α_1/ω_1
z_2	α_3/ω_3	α_4/ω_3	α_1/ω_2	α_1/ω_2	α_1/ω_2	α_1/ω_2	α_1/ω_2	α_1/ω_2	α_1/ω_2	α_1/ω_2	α_1/ω_2	α_1/ω_2

З табл. 6.11 випливає, що $A = \{a_1, \ldots, a_2\}$, $Z = \{z_1, z_2\}$, $W = \{w_1, w_2\}$. Оскільки $M = 12$, то для мінімізації автомата S_5 буде потрібно не більше одинадцяти кроків.

Побудуємо розбиття $\Pi_1 = \{B_1, \ldots, B_I\}$ множини A на класи l-еквівалентних станів, до яких належать стани a_m, $a_s \in A$ такі, для яких $\lambda(a_m, z_1) = \lambda(a_s, z_1)$ і $\lambda(a_m, z_2) = \lambda(a_s, z_2)$. Для автомата S_5 $\Pi_1 = \{B_1, B_2\}$, де $B_1 = \{a_1, a_2, a_5, a_7, a_8\}$, $B_2 = \{a_3, a_4, a_6, a_9, a_{10}, a_{11}, a_{12}\}$. Для станів класу B_1 $\lambda(a_m, z_1) = w_1$, $\lambda(a_m, z_2) = w_2$, для станів класу B_2 $\lambda(a_m, z_1) = w_2$, $\lambda(a_m, z_2) = w_1$. Отже, розбиття Π_1 будується шляхом порівняння стовпців таблиці на збіг вихідних сигналів.

Побудуємо таблицю розбиття Π_2, замінюючи стани в табл. 6.11 відповідними класами $B_i \in \Pi_1$ (табл. 6.12).

Очевидно, що 1-еквівалентні стани будуть 2-еквівалентними, якщо за вхідними сигналами вони переводяться в однакові класи

одноеквівалентних станів, тобто $\lambda(a_m, z_f) = \lambda(a_s, z_f) = B_i$, де $a_m,\ a_s \in B_i$. З табл. 6.12 знаходимо розбиття $\Pi_2 = \{C_1, \ldots, C_4\}$, де $C_1 = \{a_1, a_2\}$, $C_2 = \{a_5, a_7, a_8\}$, $C_3 = \{a_3, a_4, a_6, a_9, a_{11}\}$, $C_4 = \{a_{10}, a_{12}\}$.

Таблиця 6.12

Розбиття Π_1 множини станів автомата S_5

B_i	B_1					B_2						
a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}
z_1	B_2	B_2	B_2	B_2	B_2	B_1	B_1	B_1	B_1	B_1	B_1	B_1
z_2	B_1	B_1	B_2	B_2	B_2	B_2	B_2	B_2	B_2	B_1	B_2	B_1

Оскільки $\Pi_1 \neq \Pi_2$, то необхідно знайти розбиття Π_3, для чого будується таблиця розбиття Π_2 (табл. 6.13). Очевидно, що 2-еквівалентні стани будуть і 3-еквівалентними, якщо за однаковими вхідним сигналам вони переходять в однакові класи 2-еквівалентних станів. Це правило можна застосувати по індукції для будь-якого k.

З табл. 6.13 маємо $\Pi_3 = \{D_1, \ldots, D_5\}$, де $D_1 = \{a_1, a_2\}$, $D_2 = \{a_5, a_7\}$, $D_3 = \{a_8\}$, $D_4 = \{a_3, a_4, a_6, a_9, a_{11}\}$, $D_5 = \{a_{10}, a_{12}\}$. Оскільки виконується умова $\Pi_2 \neq \Pi_3$, то знайдемо розбиття Π_4 (табл. 6.14).

З табл. 6.14 маємо $\Pi_4 = \{E_1, \ldots, E_5\}$, де класи визначаються як $E_1 = D_1 = \{a_1, a_2\}$, $E_2 = D_2 = \{a_5, a_7\}, E_5 = D_5 = \{a_{10}, a_{12}\}$ $E_3 = D_3 = \{a_8\}$, $E_4 = D_4 = \{a_3, a_4, a_6, a_9, a_{11}\}$, отже, $\Pi_3 = \Pi_4$ і процес розбиття множини станів завершено.

Таблиця 6.13

Розбиття Π_2 множини станів автомата S_5

C_j	C_1		C_2			C_3					C_4	
a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}
z_1	C_4	C_4	C_3	C_3	C_4	C_2	C_2	C_2	C_2	C_2	C_1	C_1
z_2	C_2	C_2	C_3	C_3	C_3	C_3	C_3	C_3	C_3	C_3	C_2	C_2

Таблиця 6.14

Розбиття Π_3 множини станів автомата S_5

D_l	D_1		D_2		D_3	D_4					D_5	
a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}
z_1	D_5	D_5	D_4	D_4	D_5	D_2	D_2	D_2	D_2	D_2	D_1	D_1
z_2	D_2	D_2	D_4	D_4	D_4	D_4	D_4	D_4	D_4	D_4	D_3	D_3

Виберемо з кожного класу $E_i \in \Pi_4$ по одному стану, наприклад, виберемо стани з найменшими індексами. Ці стани утворюють множину станів мінімального автомата Мілі S_5, який охоплює п'ять станів: $A' = \{a_1, a_5, a_8, a_3, a_{10}\}$. Видалимо з вихідної таблиці (табл. 6.11) стовпці, позначені станами $a_2, a_4, a_6, a_7, a_9, a_{11}, a_{12} \in A'$. У решти стовпців замінимо всі стани $a_s \notin A'$ представниками відповідного класу еквівалентності. Наприклад, $\lambda(a_3, z_2) = a_6$ і $a_6 \notin A'$, оскільки $a_6 \in E_4$, то $\lambda(a_3, z_2) = a_3$. Після аналогічних перетворень отримаємо поєднану таблицю переходів і виходів мінімального автомата S_5 (табл. 6.15).

Таблиця 6.15

Таблиця переходів і виходів мінімального автомата Мілі S_5

a_m / z_f	α_1	α_3	α_5	α_8	α_{10}
z_1	α_{10}/ω_3	α_5/ω_2	α_3/ω_1	α_{10}/ω_1	α_1/ω_2
z_2	α_5/ω_3	α_3/ω_1	α_3/ω_2	α_3/ω_2	α_8/ω_1

При мінімізації автоматів Мура вводиться поняття 0-еквівалентності і розбиття Π_0. Стани $a_m, a_s \in A$ називаються 0-еквівалентними, якщо $\lambda(a_m) = \lambda(a_s)$. Подальший процес мінімізації аналогічний процесу мінімізації автомата Мілі. Розглянемо застосування цього методу до автомата Мура S_6 (табл. 6.16).

Таблиця 6.16

Зазначена таблиця автомата Мура S_6

ω_g	ω_3	ω_3	ω_3	ω_1	ω_3	ω_1	ω_2	ω_1	$\omega 2$
a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9
z_1	α_2	α_4	α_6	α_1	α_8	α_1	α_1	α_1	α_1
z_2	α_3	α_5	α_7	α_1	α_9	α_1	α_1	α_1	α_1

З табл. 6.16 маємо розбиття $\Pi_0 = \{B_1, B_2, B_3\}$, де $B_1 = \{a_1, a_2, a_3, a_5\}$, $B_2 = \{a_4, a_6, a_8\}$, $B_3 = \{a_7, a_9\}$. Побудуємо розбиття Π_1, використовуючи табл. 6.17. З цієї таблиці маємо розбиття $\Pi_1 = \{C_1, \ldots, C_5\}$, де $C_1 = \{a_1\}$, $C_2 = \{a_2\}$, $C_3 = \{a_1, a_5\}$, $C_4 = \{a_4, a_6, a_8\}$, $C_5 = \{a_7, a_9\}$.

Оскільки $\Pi_0 \neq \Pi_1$, то необхідно продовжити процес мінімізації. Далі необхідно знайти таке розбиття множини станів вихідного автомата Мура. Отже, знайдемо розбиття Π_2 (табл. 6.18).

Таблиця 6.17

Розбиття Π_0 автомата Мура S_6

B_i	B_1				B_2			B_3	
a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9
z_1	B_1	B_2	B_2	B_2	B_1	B_1	B_1	B_1	B_1
z_2	B_1	B_1	B_3	B_3	B_1	B_1	B_1	B_1	B_1

Таблиця 6.18

Розбиття Π_1 автомата Мура S_6

C_i	C_1	C_2	C_3		C_4			$C5$	
a_m / z_f	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9
z_1	C_2	C_4	C_4	C_4	C_1	C_1	C_1	C_1	C_1
z_2	C_3	C_3	$C5$	$C5$	C_1	C_1	C_1	C_1	C_1

З табл. 6.18 маємо розбиття $\Pi_2 = \{D_1, \ldots, D_5\}$, де $D_1 = C_1 = \{a_1\}$, $D_2 = C_2 = \{a_2\}$, $D_3 = C_3 = \{a_3, a_8\}$, $D_4 = C_4 = \{a_4, a_6, a_8\}$, $D_5 = C_5 = \{a_7, a_9\}$. Отже, $\Pi_2 = \Pi_1$, і класи еквівалентних станів знайдені. Побудуємо множину $A' = \{a_1, a_2, a_3, a_4, a_7\}$ і перетворимо табл. 6.16 у зазначену таблицю мінімального автомата Мура S_6 (табл. 6.19).

Таблиця 6.19

Зазначена таблиця мінімального автомата Мура S_6

ω_g	ω_3	ω_3	ω_3	ω_1	ω_2
a_m / z_f	α_1	α_2	α_3	α_4	α_7
z_1	α_2	α_4	α_4	α_1	α_1
z_2	α_3	α_3	α_7	α_1	α_1

Граф автомата Мура S_6, побудований за табл. 6.19, збігається з графом автомата Мура S_4 (рис. 6.12). Це цілком зрозуміло, тому що вихідна табл. 6.16 була побудована за графом немінімального автомата Мура S_4 (рис. 6.11). Отже, формальний метод мінімізації привів до таких самих результатів, як і метод, заснований на емпіричних міркуваннях.

Більшість цифрових систем можна уявити на основі моделі операційного пристрою (ОП). Ця модель розвинена В. М. Глушковим на базі запропонованого М. Уілксом принципу мікропрограмного управління (1951 р.). Модель операційного пристрою представлена на рис. 6.13.

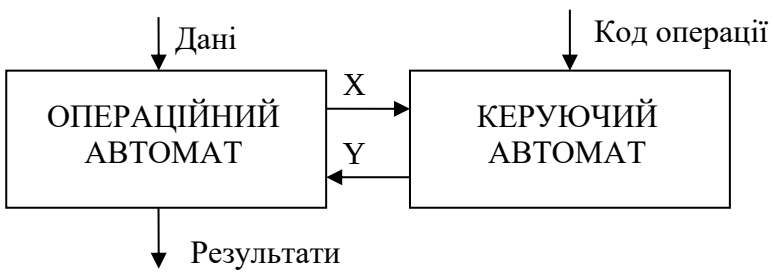


Рис. 6.13 – Структура операційного пристрою

До складу ОП входить операційний автомат (ОА), який приймає, обробляє і зберігає дані і формує проміжні і кінцеві результати виконання операцій. Крім того, ОА формує інформаційні сигнали (логічні умови, прапори) X, що характеризують стан процесу обробки даних. Керуючий автомат (КА) аналізує код операції, яку необхідно виконати ОА, і значення сигналів X. На основі цього аналізу формуються керуючі сигнали (мікрооперації) Y для деякого алгоритму управління ОА. Обидві частини ОП можуть бути представлені у вигляді абстрактних автоматів. Зазначимо, що для практичних задач розмірність цих автоматів буде занадто великою, а іноді і нескінченною. Однак існує велика кількість реальних завдань, які можуть бути представлені абстрактними автоматами невеликої розмірності.

6.4 Практичні приклади абстрактних автоматів

У попередніх розділах розглядалися деякі приклади абстрактних автоматів, які ілюстрували загальні принципи їхнього синтезу. Розглянемо деякі приклади, які застосовуються на практиці.

Приклад 6.3. Генератор одиничних імпульсів з пам'яттю генерації. Такий генератор має вхід S. При появі $S=1$ генератор формує одиничний імпульс на виході P. Нехай цей імпульс триває протягом одного такту автоматного часу. Одночасно на виході L формується сигнал, що дорівнює одиниці за $P=1$. Цей вихід є пам'яттю про те, що одиничний імпульс був сформований. Якщо тепер $S=L$, то імпульс на виході P не формується. Якщо тепер $S=0$ (тобто переходить з 1 в 0), то на обох входах P і L встановлюються нулі, і автомат переходить в початковий стан. Цій поведінці відповідає граф, показаний на рис. 6.14.

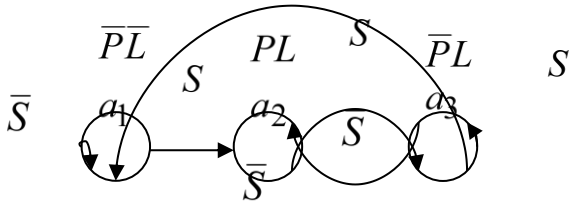


Рис. 6.14 – Граф генератора одиничних імпульсів з пам'яттю генерації

На рис. 6.14 використані символи, які полегшують сприйняття цього прикладу. Якщо ми приймемо, що $z_1 = \overline{S}$, $z_2 = S$, $\omega_1 = \overline{P}\overline{L}$, $\omega_2 = PL$, $\omega_3 = \overline{P}L$, то ми отримаємо звичний для нас граф абстрактного автомата Мура.

Приклад 6.4. Дільник частоти на два. Такий генератор має 2 входи: вхід S, який впливає на виходи P і L так само, як і в генераторі одиничних імпульсів, і вхід R. Якщо $R=1$, то P встановлюється з 1 в 0. Якщо $R = 0$, то $P = 0$, і автомат реагує на вхід S так само, як і для випадку генератора одиничних імпульсів. Така поведінка описується графом автомата Мура на рис. 6.15.

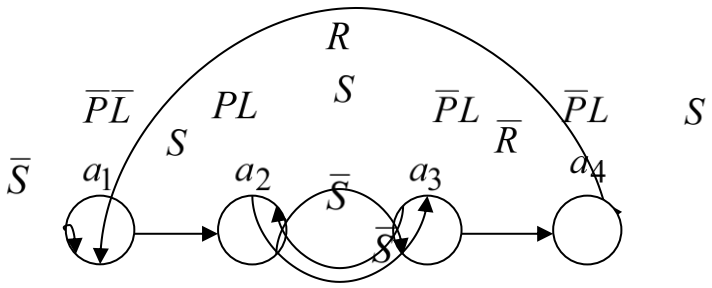


Рис. 6.15 – Граф генератора одиничних імпульсів і дільника частоти на два

Нехай $z_1 = \overline{S}$, $z_2 = S$, $z_3 = \overline{R}$, $z_4 = R$, $\omega_1 = \overline{P}\overline{L}$, $\omega_2 = PL$, $\omega_3 = \overline{P}L$. В цьому випадку автомат на рис. 6.15 представляється зазначеною таблицею переходів (табл. 6.20).

Таблиця 6.20

Генератор одиничних імпульсів і дільник частоти на два

ω_g	ω_1	ω_2	ω_3	ω_3
a_m / z_f	α_1	α_2	α_3	α_4
z_1	α_1	α_3	*	α_4
z_2	α_2	α_3	*	α_1
z_3	*	*	α_2	*
z_4	*	*	α_3	*

Як видно з табл. 6.20, цей автомат є частковим, оскільки не всі пари $\langle a_m, z_f \rangle$ мають певне значення. Як тренування читач може побудувати тимчасову діаграму, яка визначається графом на рис. 6.15.

Приклад 6.5. Організація очікування. Дуже часто на практиці автомат повинен знаходитися у стані очікування, поки не закінчиться деякий період часу. Наприклад, необхідно ввімкнути деяке реле, а через певний період часу вимкнути його. Цього можна домогтися тривіальним способом, ввівши ланцюжок зі станів. Наприклад, якщо треба чекати 40 нсек, а період автоматного часу дорівнює 2 нсек. В цьому випадку для очікування необхідно ввести 20 послідовних станів. Такий підхід веде до різкого збільшення числа станів автомата і, отже, до збільшення витрат апаратури при реалізації відповідної схеми.

На практиці для вирішення цього завдання використовується спеціальний пристрій, який називається таймером. Таймер встановлюється на певний час, для запуску використовується вхід ts, а вихід T_0 показує момент закінчення заданого проміжку часу. Отже, $ts = 1$ – запуск таймера, $T_0 = 1$ – проміжок часу закінчено. Автомат, що керує таймером, повинен мати вихід Ts, який запускає таймер, і вхід t_0, який пов'язаний з виходом T_0 таймера (рис. 6.16).

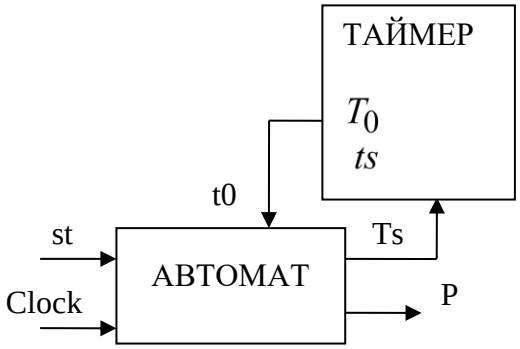


Рис. 6.16 – Організація очікування з таймером

На рис. 6.16 вхід $st=1$ потрібен для запуску процесу ($P=1$), зміна станів відбувається за сигналом Clock. Одночасно із запуском процесу відбувається запуск таймера ($Ts=1$). Якщо період часу не закінчився ($T_0=0$), то стан автомата не змінюється, за $T_0=1$ процес припиняється ($P=0$). Ця поведінка задана графом (рис. 6.17).

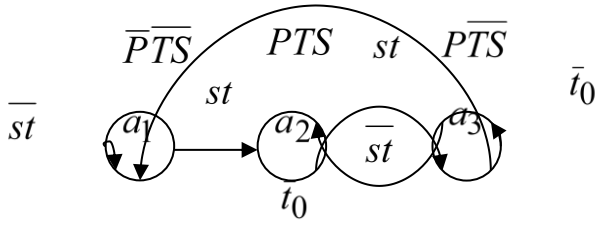


Рис. 6.17 – Граф автомата з таймером

У стані a_1 процес і таймер є неактивними, тобто $P=T_s=0$, доки $st=0$. За $st=1$ автомат переходить у стан a_2, запускаючи процес і таймер ($P=T_s=1$). За наступного імпульсу Clock автомат переходить у стан a_3. При цьому процес залишається активним ($P=1$), а таймер продовжує відміряти час, за сигналом запуску знімається ($T_s=0$). Автомат залишається у стані a_3, доки час процесу не закінчується, що відповідає $t_0=1$. За $t_0=0$ автомат чекає тайм-ауту, а за $t_0=1$ – переходить у стан a_1 і чекає наступного запуску.

Приклад 6.6. Управління аналогово-цифровим перетворювачем. Аналогово-цифрові перетворювачі (АЦП) використовуються для перекладу даних з аналогової форми (безперервний сигнал, наприклад,

напруга) в цифрову форму (двійковий код, який відповідає цьому рівню напруги). Здебільшого, АЦП є невід'ємною частиною будь-якої керуючої системи. Переважно АЦП управляється мікропроцесором, але часто схема управління АЦП може бути виконана як спеціалізована схема, яка використовує CPLD або FPGA.

Будь-який АЦП має вхід аналогових даних (AD) і вихід цифрових даних (DD). Для початку перетворення використовується сигнал SC, а кінець перетворень ініціюється вихідним сигналом $ES = 1$. Очевидно, доки $ES = 0$, перетворення не завершено.

Система з АЦП і автоматом управління показана на рис. 6.18. Тут автомат має входи st (для запуску) і ES (для аналізу станів АЦП).

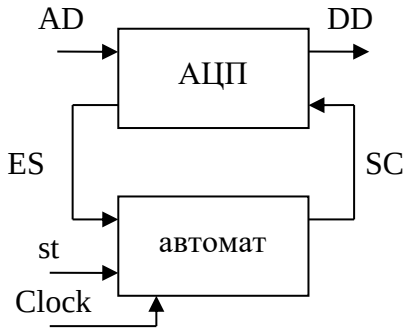


Рис. 6.18 – Система управління АЦП

Граф автомата Мура, керуючий АЦП, показаний на рис. 6.19. У початковому стані a_1 автомат чекає сигнал запуску ($st = 1$), який означає необхідність запуску АЦП. При цьому автомат переходить у стан a_2 і формує сигнал $SC = 1$. Далі автомат переходить у стан a_3, в якому знімається сигнал запуску АЦП, і чекає переходу сигналу $ES = 1$, який переводить автомат в початковий стан.

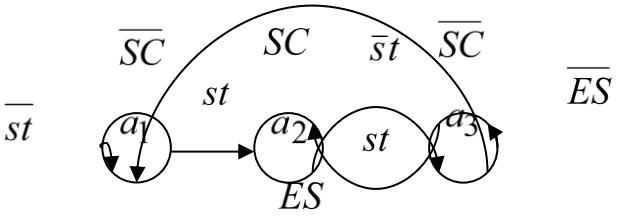


Рис. 6.19 – Граф автомата управління таймером

Ця система може бути частиною більш складної системи, яка займається збором даних.

Приклад 6.7. Система збору даних. Розглянемо систему, яка приймає дані, представлені у вигляді змінного струму, і переводить їх в постійний аналоговий сигнал (назвемо цей пристрій реєстратором). Далі ці дані перетворюються АЦП і запам'ятовуються в пристрої пам'яті (назвемо цей пристрій пам'яттю). Така система може управлятися віддаленим мікропроцесором і збирати для нього дані. Для запису інформації в пам'ять використовується лічильник, який обнуляється за сигналом RC і до вмісту якого додається одиниця за сигналом CC. Якщо лічильник (CT) перемикається (наприклад, він може записати числа від 0000 до 1111, а приходить сімнадцятий поспіль сигнал CC), то на виході формується сигнал переповнення OVF. Структура системи збору інформації, що керується автоматом, показана на рис. 6.20.

Ця система функціонує у такий спосіб. За сигналом *st* починається черговий цикл роботи. Автомат запускає реєстратор ($SH=1$), таймер ($TS=1$) і обнуляє лічильник ($RC=1$). Після закінчення реєстрації ($t_0=1$) на виході реєстратора перебувають аналогові дані D_A. Автомат формує сигнал $SC=1$ і запускає АЦП. Після закінчення перетворення ($ES=1$) дані D_D у цифровій формі перебувають на вході пам'яті. Автомат формує сигнал запису даних ($WR=1$), і дані D_D заносяться в пам'ять за адресою, що знаходиться на виході CT. Коли запис закінчено ($ER=1$), автомат збільшує вміст CT ($CC=1$). Якщо приходить сигнал $reset=1$, то автомат переходить у стан a_1, і цикл завершується. В іншому разі ($reset=0$) цикл повторюється доти, доки пам'ять повністю заповнюється, чому відповідає $OVF=1$. Якщо $OVF=1$, то автомат формує сигнал $Done=1$ і переходить у стан a_1. Подібна поведінка задається графом автомата Мілі (рис. 6.21).

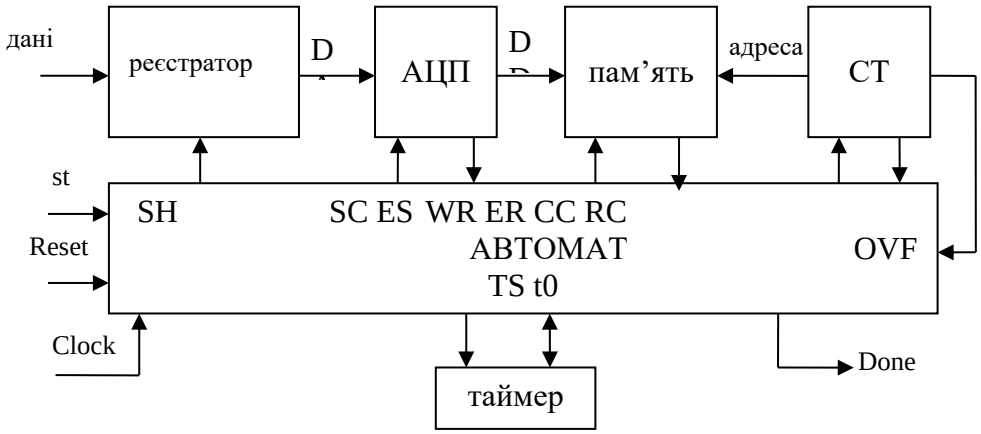


Рис. 6.20 – Система збору даних

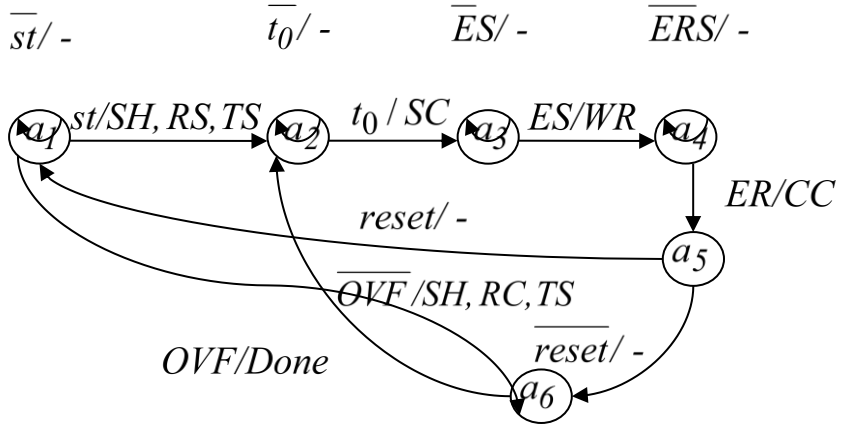


Рис. 6.21 – Граф автомата Мілі, що керує системою збору даних

Визначимо алфавіти відповідного абстрактного автомата. Нехай $z_1 = \overline{st}$, $z_2 = st$, $z_3 = \overline{t_0}$, $z_4 = t_0$, $z_5 = \overline{ES}$, $z_6 = ES$, $z_7 = \overline{ER}$, $z_8 = ER$, $z_9 = \overline{\mathrm{Re}\, set}$, $z_{10} = \mathrm{Re}\, set$, $z_{11} = \overline{OVF}$, $z_{12} = OVF$, тоді $Z = \{z_1, \ldots, z_{12}\}$ і $F = 12$. Нехай ω_1 відповідає відсутності вихідного сигналу, $\omega_2 = 1$, якщо $SH = RC = TS = 1$, $\omega_3 = SC$, $\omega_4 = WR$, $\omega_5 = CC$, $\omega_6 = Done$, тоді $W = \{\omega_1, \ldots, \omega_6\}$ і $G = 6$. В цьому разі автомат управління представляється такою поєднаною таблицею (табл. 6.21).

Таблиця 6.21

Таблиця автомата, що керує системою збору даних

z_f \ a_m	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_1/ω_1	-	-	-	-	-
z_2	-	-	-	-	-	-
z_3	-	a_2/ω_1	-	-	-	-
z_4	-	a_3/ω_3	-	-	-	-
z_5	-	-	a_3/ω_1	-	-	-
z_6	-	-	a_4/ω_4	-	-	-
z_7	-	-	-	a_4/ω_1	-	-
z_8	-	-	-	a_5/ω_5	-	-
z_9	-	-	-	-	a_6/ω_1	-
z_{10}	-	-	-	-	a_1/ω_1	-
z_{11}	-	-	-	-	-	a_2/ω_2
z_{12}	-	-	-	-	-	a_1/ω_1

Приклад 6.8. Електронні годинники. Практично в кожному будинку є подібний пристрій, що характеризується різноманітністю функцій. Розглянемо приклад подібних годинників, що керуються автоматом Мілі.

Необхідно побудувати граф автомата Мілі, що керує установкою та індикацією часу в електронному годиннику. Вхідний алфавіт містить дві літери: $Z = \{z_1, z_2\}$, де z_1 – сигнал необхідної установки часу (хвилини, години, дні, місяці), z_2 – сигнал, що ініціює додавання одиниці до встановленого часу. Алгоритм роботи цього автомата можна описати у такий спосіб:

– під час підключення живлення автомат переходить у початковий стан і висвічує поточний час (стан a_1);

– при надходженні літери z_1 (натискання кнопки на клавіатурі годинника) автомат переходить у стан a_2, що відповідає налаштуванню хвилин;

– при надходженні літери z_2 у стані a_2 автомат залишається в цьому стані і додає одиницю до лічильника хвилин;

– при надходженні літери z_1 у стані a_2 автомат переходить у стан a_3 (налаштування годин);

– при надходженні літери z_2 у стані a_3 автомат залишається в цьому стані і додає одиницю до лічильника годин;

– при надходженні літери z_1 у стані a_3 автомат переходить у стан відображення часу a_1;

– при надходженні літери z_2 у стані a_1 автомат переходить у стан a_4 і відображає дату;

– при надходженні літери z_2 у стані a_4 автомат повертається у стан a_1;

– при надходженні сигналу z_1 у стані a_4 автомат переходить у стан a_5, що відповідає налаштуванню числа;

– при надходженні літери z_2 у стані a_5 автомат додає одиницю до лічильника днів;

– при надходженні літери z_1 у стані a_5 автомат переходить у стан a_6, що відповідає налаштуванню місяця;

– при надходженні літери z_2 у стані a_6 автомат додає одиницю до лічильника місяців і залишається у стані a_6;

– при надходженні літери z_1 у стані a_6 автомат переходить у стан a_1.

Отже, вихідний алфавіт автомата охоплює такі літери:

ω_1 – додаток одиниці до лічильника хвилин;

ω_2 – додаток одиниці до лічильника годин;

ω_3 – додаток одиниці до лічильника днів;

ω_4 – додаток одиниці до лічильника місяців;

ω_5 – відображення часу (години та хвилини);

ω_6 – відображення дати (дні і місяці).

Тепер отримані всі алфавіти автомата, а саме $Z = \{z_1, z_2\}$, $W = \{\omega_1, \ldots, \omega_6\}$ і $A = \{a_1, \ldots, a_6\}$. Граф автомата управління електронними годинниками показаний на рис. 6.22.

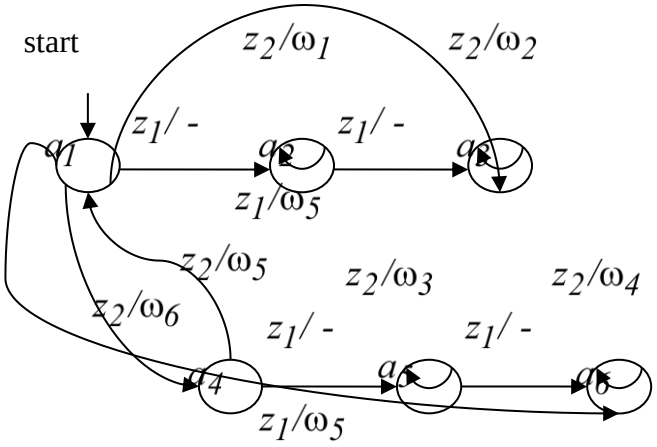


Рис. 6.22 – Граф автомата Мілі пристроїв управління електронними годинниками

Приклад 6.9. Регулювання пішохідного переходу. Існують різноманітні системи управління світлофорами. Розглянемо одне з можливих рішень цього завдання.

Нехай пішохідний перехід регулюється натисканням кнопки «запитів на перехід» пішоходом. Це натискання обробляється тільки в початковому стані автомата. При дозволу переходу зелене світло блимає, і подається звуковий сигнал. При цьому можливі такі вхідні сигнали автомата, що визначають його алфавіт входів:

z_1 – натискання кнопки «запит на перехід»;

z_2 – початок жовтого кольору для транспорту або червоного – для пішоходів;

z_3 – початок зеленого кольору для транспорту або для пішоходів;

z_4 – початок блимання зеленого кольору для пішоходів;

z_5 – кінець одного інтервалу блимання.

Вихідний алфавіт автомата визначається такими поєднаннями кольорів світлофора:

ω_1 – зелений для транспорту і червоний для пішоходів;

ω_2 – жовтий для транспорту і червоний для пішоходів;

ω_3 – червоний для транспорту і зелений для пішоходів;

ω_4 – червоний для транспорту і зелений для пішоходів вимкнений;

ω_5 – жовтий і червоний для транспорту і червоний для пішоходів.

Зауважимо, що в кожному стані $a_m \in A$ автомат реагує тільки на деякі сигнали, а інші вхідні букви ігноруються. Це означає, що автомат є частково визначеним. Задаємо цей автомат у вигляді автомата Мура (рис. 6.23). З цього графа можна визначити всі алфавіти автомата: $Z = \{z_1, \ldots, z_5\}$, $W = \{\omega_1, \ldots, \omega_5\}$ і $A = \{a_1, \ldots, a_7\}$. Функції переходів і виходів автомата управління пішохідним перехрестям очевидні з графа автомата і зазначеної таблиці переходів (табл. 6.22). Зазначимо, що задання закону поведінки автомата можна починати безпосередньо з побудови таблиці, але процес побудови графа більш очевидний.

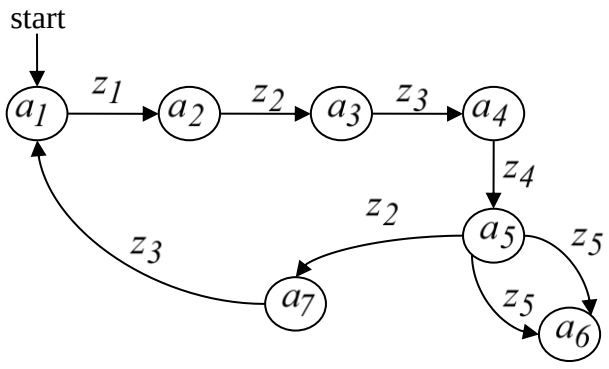


Рис. 6.23 – Граф автомата Мура для регулювання пішохідного переходу

Таблиця 6.22

Зазначена таблиця переходів автомата для регулювання пішохідного переходу

ω_g	ω_1	ω_1	ω_2	ω_3	ω_4	ω_3	ω_5
a_m / z_f	a_1	a_2	a_3	a_4	a_5	a_6	a_7
z_1	a_2	-	-	-	-	-	-
z_2	-	a_3	-	-	a_7	-	-
z_3	-	-	a_4	-	-	-	a_1
z_4	-	-	-	a_5	-	a_5	-
z_5	-	-	-	-	a_6	-	-

З цієї таблиці випливає, наприклад, $\delta(a_2, z_2) = a_3$ або $\lambda(a_6) = \omega_3$. Ці самі функції легко визначаються і за графом (рис. 6.23).

Автори сподіваються, що наведених прикладів достатньо, щоб читачі переконалися, яке важливе практичне значення мають абстрактні автомати.

Список літератури до розділу 6

1. Глушков В. М. Синтез цифровых автоматов. Москва: Физматгиз, 1962. 476 с.

2. Карнов Ю. Г. Теория автоматов. Санкт-Петербург: Питер, 2003. 208 с.

3. Баранов С. И. Синтез микропрограммных автоматов. Ленинград: Энергия, 1979. 232 с.

4. Ковриженко Г. А. Системы счисления и двоичная арифметика. Киев: Радянська школа,1984. 79 с.

5. Прикладная теория цифровых автоматов / К. Г. Самофалов, А. М. Романкевич, В. Н. Валуйский, Ю. С. Каневский, М. М. Пиневич. Киев: Вища шк., 1987. 375 с.

6. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

7. Baranov S. Logic Syntethesis for Control Automata. Boston: Kluwer Academic Publisher, 1994. 312 p.

8. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. Москва: Вильямс, 2002. 528 с.

9. Minns P., Elliot I. FSM-based Digital Design using Verilog HDL. New York: John Wileys Sons, Ltd, 2008. 391 p.

10. Mealy G. Method for Synthesis Sequential Circuits. *Bell Sestems Techn. Journal.* V. 34, 1955. P. 1045–1079.

11. Moore E. Gedanker – experiments on sequential machines. *C. Shannon and J. McCarthy editors.* Automata Studies, Princeton Hall University Press, 1956. P. 129–153.

7 СТРУКТУРНИЙ АВТОМАТ

7.1 Модель структурного автомата

Структурний автомат – це пристрій, що реалізує закон поведінки абстрактного автомата і є мережею з логічних елементів. На відміну від абстрактного автомата, що має один вхідний і один вихідний канали, структурний автомат має:

1. L вхідних каналів, на які надходять елементи множини $X = \{x_1, \ldots, x_L\}$, де $L = \lceil \log_2 F \rceil$, $F = |Z|$, $Z = \{z_1, \ldots, z_F\}$ – вхідний алфавіт абстрактного автомата.

2. N вихідних каналів, з яких знімаються елементи множини $Y = \{y_1, \ldots, y_N\}$, де $N = \lceil \log_2 G \rceil$, $G = |W|$, $W = \{w_1, \ldots, w_G\}$ – вихідний алфавіт абстрактного автомата.

Для синтезу схеми структурного автомата необхідна наявність структурно повної системи елементарних автоматів, з яких і синтезується логічна схема. Існує загальний конструктивний прийом, який називається канонічним шляхом структурного синтезу, що дає змогу синтезувати схему структурного автомата за таблицями або за графом абстрактного автомата. Цей метод був запропонований відомим математиком і кібернетиком академіком В. М. Глушковим. Метод передбачає наявність структурно повної системи, що включає елементарні автомати двох типів: автомати з пам’яттю, що мають понад один стан, і автомати без пам’яті, що мають один стан. Перші автомати називаються елементами пам’яті, другі – логічними елементами (І, АБО, НЕ, І-НЕ, АБО-НЕ).

Зупинимося докладніше на елементах пам’яті, які є автоматами Мура з нетривіальною пам’яттю, що мають повноту системи переходів і системи виходів. Розглянемо як приклад автомат Мура S_7 (табл. 7.1).

Таблиця 7.1

Зазначена таблиця переходів автомата Мура S_7

ω_g	ω_2	ω_3	ω_1
α_m / z_f	α_1	α_2	α_3
z_1	α_1	α_2	α_3
z_2	α_2	α_3	α_1
z_3	α_3	α_1	α_2

Автомат S_7 має повноту системи переходів, оскільки з будь-якого стану $a_m \in A = \{a_1, a_2, a_3\}$ можливий перехід в будь-який інший стан. Автомат S_7 має повноту системи виходів, оскільки кожному його стану $a_m \in A$ відповідає унікальний вихідний сигнал $w_g \in W = \{w_1, w_2, w_3\}$. Це дає змогу ототожнити стани і виходи. Автомати з повнотою системи переходів і виходів називаються повними автоматами. Використання повних автоматів як елементів пам'яті структурних автоматів дає змогу зменшити число елементів пам'яті до $R = \lceil \log_k M \rceil$, де M – число станів елемента пам'яті. Зауважимо, що на практиці використовуються двійкові елементи пам'яті, які будуть розглянуті пізніше. При цьому для визначення числа елементів пам'яті у схемі автомата необхідно використовувати двійкові логарифми, тобто $R = \lceil \log_2 M \rceil$.

Канонічний метод структурного синтезу заснований на доведеній В. М. Глушковим теоремі про структурну повноту, яка формулюється у такий спосіб:

«Будь-яка система елементарних автоматів, яка містить повний автомат Мура з нетривіальною пам'яттю і будь-яку функціонально повну систему логічних елементів, є структурно повною. Існує загальний конструктивний прийом, що називається канонічним методом

структурного синтезу, який дає змогу звести синтез структурного автомата до синтезу комбінаційної схеми».

Результатом застосування цього методу є система рівнянь, що задає залежність вихідних сигналів і сигналів, що подаються на входи запам'ятовуючих елементів, від сигналів, що надходять на входи автомата ззовні, і сигналів, що знімаються з виходів запам'ятовуючих елементів. Ці рівняння називаються канонічними.

Схема структурного автомата (рис. 7.1) складається з комбінаційної схеми КС та елементів пам'яті $П_1,\ldots,П_R$, які утворюють пам'ять автомата.

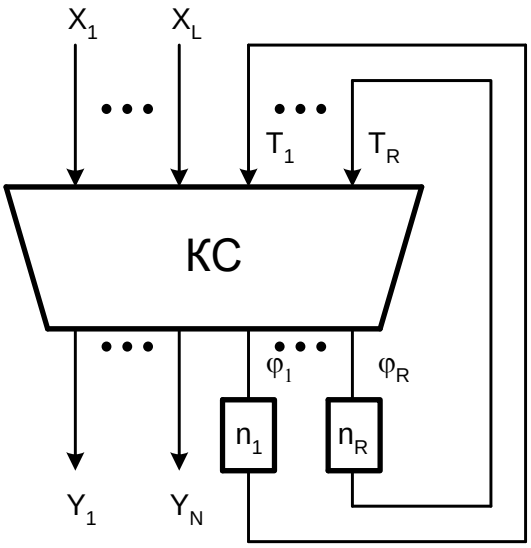


Рис. 7.1 – Схема структурного автомата

На входи КС надходять вхідні сигнали $x_1,\ldots,x_L$ і сигнали $T_1,\ldots,T_R$, що формуються на виходах елементів пам'яті і утворюють множину внутрішніх змінних T структурного автомата. На виходах КС формуються вихідні сигнали $y_1,\ldots,y_N$ і функції збудження елементів пам'яті $\varphi_1,\ldots,\varphi_R$, що утворюють множину функцій збудження пам'яті автомата Ф. Комбінаційна схема зображується у вигляді трапеції, а елементи пам'яті – у вигляді прямокутників, що відрізняє схеми без пам'яті від схем з пам'яттю.

Комбінаційна схема КС задається системою рівнянь:

$$\begin{aligned} &y_1 = y_1(x_1,\ldots,x_L,T_1,\ldots,T_R);\\ &\vdots\\ &y_N = y_N(x_1,\ldots,x_L,T_1,\ldots,T_R);\\ &\varphi_1 = \varphi_1(x_1,\ldots,x_L,T_1,\ldots,T_R);\\ &\vdots\\ &\varphi_R = \varphi_R(x_1,\ldots,x_L,T_1,\ldots,T_R). \end{aligned} \tag{7.1}$$

Система (7.1) може бути представлена у векторній формі у такий спосіб:

$$\begin{aligned} Y &= Y(X,T),\\ \Phi &= \Phi(X,T). \end{aligned} \tag{7.2}$$

Оскільки в системі (7.2) вихідні сигнали залежать від вхідних сигналів і станів, то рівняння (7.2) задають автомат Мілі. Автомат Мура задається системою рівнянь

$$\begin{aligned} Y &= Y(T),\\ \Phi &= \Phi(X,T). \end{aligned} \tag{7.3}$$

Перш ніж розглянути приклади синтезу структурних автоматів, необхідно зупинитися на елементах пам'яті, які використовуються при синтезі реальних цифрових систем. Ці елементи мають два стійкі стани і називаються тригерами. Другим важливим питанням є забезпечення стійкої роботи автомата і завдання автоматного часу. Розглянемо докладніше ці питання.

7.2 Тригерні схеми

Розглянемо схему (рис. 7.2), що складається з двох елементів АБО-НЕ і встановимо закон її функціонування. Аналіз закону функціонування виконаємо на основі системи логічних рівнянь для цієї схеми. Безпосередньо зі схеми можна сформувати таку систему, що складається з двох рівнянь:

$$\begin{aligned} b_1 &= \overline{\varphi_1 \vee b_2},\\ b_2 &= \overline{\varphi_2 \vee b_1}. \end{aligned} \tag{7.4}$$

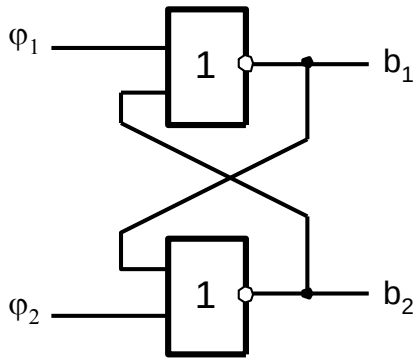


Рис. 7.2 – Запам'ятовуючий елемент

Нехай $\varphi_1 = 1,\ \varphi_2 = 0$, тоді з (7.4) маємо: $b_1 = 0, b_2 = \overline{0 \vee 0} = 1$. Отже, незалежно від значень b_1 і b_2 при вхідному наборі $\varphi_1 = 1,\ \varphi_2 = 0$ на виході b_1 встановлюється нуль, на виході b_2 – одиниця. Нехай $\varphi_1 = 0$, $\varphi_2 = 1$, тоді з (7.4) маємо: $b_2 = 0, b_1 = \overline{0 \vee 0} = 1$. Отже, незалежно від попередніх значень виходів b_1 і b_2 при вхідному наборі $\varphi_1 = 0,\ \varphi_2 = 1$ на виході b_1 встановлюється одиниця, на виході b_2 – нуль. Нехай $\varphi_1 = \varphi_2 = 0$, при цьому виконуються рівності $b_1 = \bar{b}_2, b_2 = \bar{b}_1$, тобто елемент зберігає колишні значення виходів. Якщо $\varphi_1 = \varphi_2 = 1$, то $b_1 = b_2 = 0$. Проведений аналіз можна представити у вигляді тимчасової діаграми (рис. 7.3).

Нехай у вихідному стані $(t = 0)$ маємо $b_1 = 1,\ b_2 = \varphi_1 = \varphi_2 = 0$. У момент часу $t = 1$ отримаємо $\varphi_1 = 1,\ \varphi_2 = 0$. Елемент АБО-НЕ з виходом b_1 починає переключатися з одиниці в нуль, перемикання відбувається за час $\boldsymbol{t_{ле}}$. Потім починає перемикатися елемент АБО-НЕ з виходом b_2. У момент часу $t = 2$ $\varphi_1 = \varphi_2$, і стан виходів залишається незмінним. У момент часу $t = 3$ $\varphi_1 = 0,\ \varphi_2 = 1$. Елемент АБО-НЕ з виходом b_2 починає переключатися з одиниці в нуль, а через час $\boldsymbol{t_{ле}}$ починає перемикатися другий елемент. У момент часу $t = 4$, $\varphi_1 = \varphi_2 = 0$, і стан виходів стабільний. У момент часу $t = 5$ $\varphi_1 = \varphi_2 = 1$, і обидва елементи переключаються в нуль за час $\boldsymbol{t_{ле}}$ і так далі.

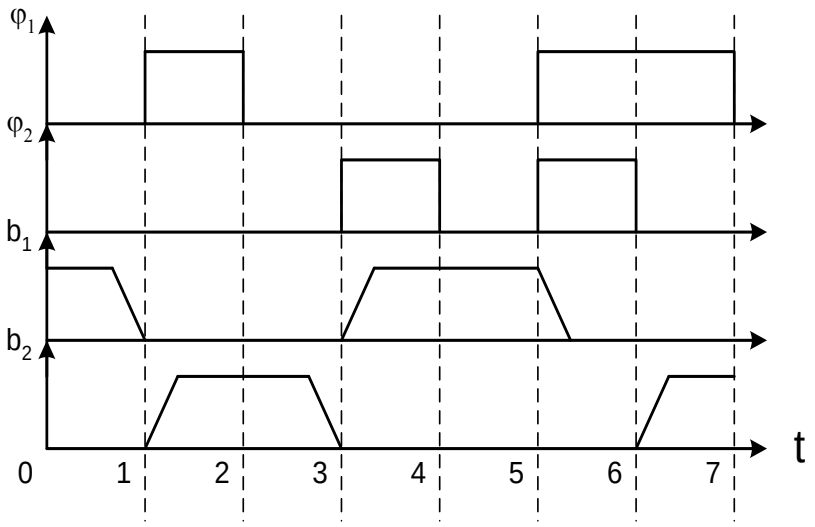


Рис. 7.3 – Тимчасова діаграма роботи запам'ятовуючого елемента

Оскільки тригер повинен мати два стійкі стани, комбінація $\varphi_1 = \varphi_2 = 1$ є забороненою. Для всіх інших комбінацій схема на рис. 7.2 має властивості тригера. Нехай b_1 відповідає прямому виходу тригера, а b_2 – інверсному, тобто станам 1 і 0 відповідно. Тоді вхід φ_1 є входом установки тригера в нуль або входом скидання, а вхід φ_2 – входом установки в одиницю або входом установки. Запам'ятовуючий елемент на рис. 7.2 називається RS-тригером з прямим управлінням, вхід φ_1 є входом R (від reset – скидання), а φ_2 – вхід S (від set – установка). Схема та умовне позначення RS-тригера наведені на рис. 7.4.

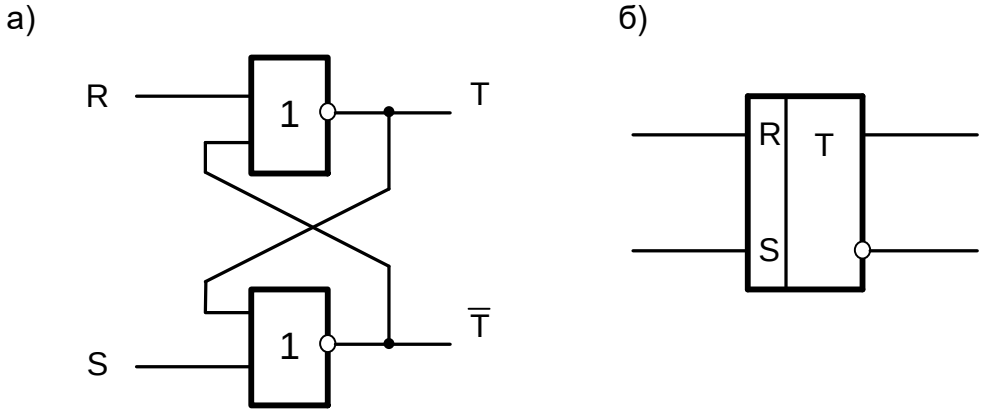


Рис. 7.4 – Схема (а) й умовне позначення (б) RS-тригера

Термін «пряме управління» означає, що робочим сигналом на входах тригера є одиниця. Таблиця переходів RS-тригера приведена в табл. 7.2, яка побудована на основі раніше проведеного аналізу за рівняннями (7.4).

Таблиця 7.2

Таблиця переходів RS-тригера

R	S	T	T	коментар
0 0	0 0	0 1	0 1	зберігання
0 0	1 1	0 1	1 1	установка
1 1	0 0	0 1	0 0	скидання
1 1	1 1	0 1	* *	заборона

Ця таблиця відображає процес функціонування тригера в часі, тому ліва частина таблиці відображає стан схеми в момент часу t, а права – в момент $t+1$ $(t = 0,1,\ldots)$. Отже, система (7.4) може бути записана у такий спосіб:

$$\begin{aligned} T^{t+1} &= \overline{R^t \vee \overline{T}^t}, \\ \overline{T}^{t+1} &= \overline{S^t \vee T^t}. \end{aligned} \qquad (7.5)$$

Аналіз рівнянь (7.1) показує, що для синтезу структурного автомата необхідно знати, якими повинні бути функції збудження пам’яті в момент часуt для правильного перемикання пам’яті автомата. Тобто нас цікавить не система (7.5), а деяка система

$$\begin{aligned} R^t &= f(T^t, T^{t+1}), \\ S^t &= \varphi(T^t, T^{t+1}). \end{aligned} \qquad (7.6)$$

Для формування системи (7.6) необхідно проаналізувати всі переходи виду $\langle T^t, T^{t+1} \rangle$ з табл. 7.2. Виконання подібного аналізу не становить труднощів.

Переходу $\left\langle T^t, T^{t+1} \right\rangle = \langle 0,0 \rangle$ відповідають перший і п'ятий рядки таблиці, при цьому $R^t S^t = 00$ або $R^t S^t = 10$, отже, значення R^t може бути довільним. Відтак, переходу $\langle 0,0 \rangle$ відповідає вхідний набір $R^t S^t = *0$. Переходу $\left\langle T^t, T^{t+1} \right\rangle = \langle 0,1 \rangle$ відповідає третій рядок таблиці, тобто вхідний набір $R^t S^t = 01$. Переходу $\left\langle T^t, T^{t+1} \right\rangle = \langle 1,0 \rangle$ відповідає шостий рядок таблиці, тобто вхідний набір $R^t S^t = 10$. Переходу $\left\langle T^t, T^{t+1} \right\rangle = \langle 1,1 \rangle$ відповідають другий і четвертий рядки таблиці, тобто вхідний набір $R^t S^t = 0*$.

На основі проведеного аналізу побудуємо таблицю входів RS-тригера (табл. 7.3).

Таблиця 7.3

Таблиця входів RS-тригера

T^t	T^{t+1}	R^t	S^t
0	0	*	0
0	1	0	1
1	0	1	0
1	1	0	*

У лівій частині цієї таблиці записані переходи між станами, а в правій – функції збудження тригера. З урахуванням невизначеності можна отримати систему (7.6), яка має такий вигляд:

$$\begin{aligned} R^t &= \overline{T^{t+1}}, \\ S^t &= T^{t+1}. \end{aligned} \qquad (7.7)$$

Перемикання пам'яті автомата складається з перемикань окремих тригерів і має починатися у строго певні моменти часу. Для завдання цих моментів використовується спеціальний генератор тактових імпульсів ГТІ, який є годинами автомата.

Основна функція ГТІ полягає у формуванні розподіленої в часі послідовності імпульсів (рис. 7.5).

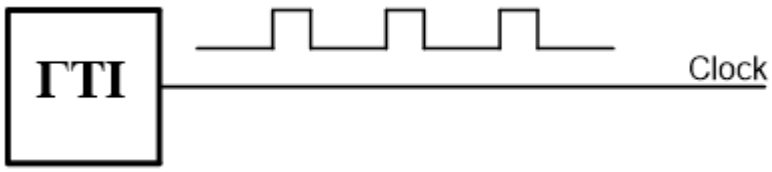


Рис. 7.5 – Діаграма роботи генератора тактових сигналів

Вихід ГТІ зазвичай позначається символом Clock, а сигнали, що ним формуються, називаються імпульсами синхронізації.

Для перемикання в моменти часу, коли $Clock = 1$, тригер повинен мати спеціальний вхід синхронізації C. Якщо $C = 0$, то тригер не реагує на зміну вхідних сигналів і залишається в поточному стані. Такі тригери називаються синхронними. Таблиця переходів синхронного RS-тригера приведена в табл. 7.4.

Якщо $C = 0$, то $T^{t+1} = T^t$, тобто тригер зберігає попередній стан. Якщо $C = 1$, то синхронний тригер функціонує як і асинхронний тригер (табл. 7.2).

Таблиця 7.4

Таблиця переходів синхронного RS-тригера з прямим управлінням

C^t	R^t	S^t	T^t	T^{t+1}	коментар
0	*	*	*	T^t	зберігання
1	0	0	*	T^t	
1	0	1	*	1	Set
1	1	0	*	0	Reset
1	1	1	*	*	заборона

З аналізу табл. 7.4 випливає, що за $C = 0$ на R і S входи тригера повинні подаватися нулі, що досягається за рахунок введення двох вентилів типу І (рис. 7.6а). Якщо $C = 0$, то $R_a = C \cdot R = 0$ і $S_a = C \cdot S = 0$,

якщо $C=1$, то $R_a = R$, $S_a = S$, що відповідає закону (табл. 7.4). Умовне позначення RS-тригера з синхровходом наведено на рис. 7.6б.

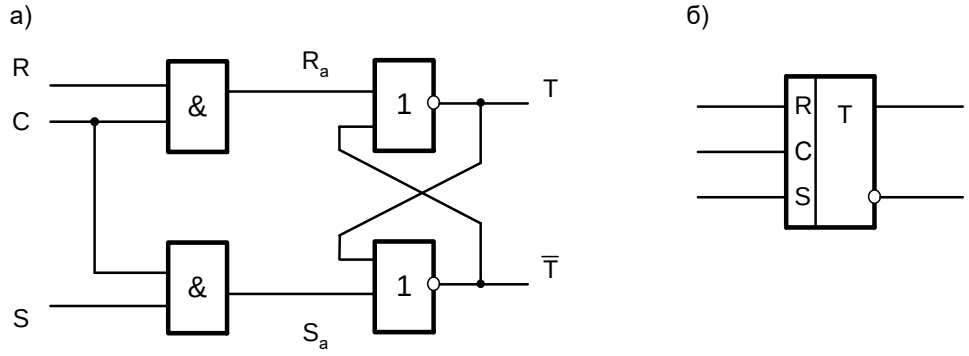


Рис. 7.6 – Функціональна схема (а) і умовне позначення (б) синхронного RS-тригера

Аналіз схеми структурного автомата показує, що в ній є зворотний зв'язок між пам'яттю і комбінаційною схемою (рис. 7.1). Це може призвести до збоїв у роботі автомата через розкид часу спрацьовування тригерів. Наприклад, автомат повинен перейти зі стану a_2 з кодом 010 у стан a_3 з кодом 101 за сигналом z_1 (рис. 7.7а).

Цей перехід складається з переходів кожного елемента пам'яті $П_1 - П_3$: $T_1^t \to T_1^{t+1}: 0 \to 1; T_2^t \to T_2^{t+1}: 1 \to 0; T_3^t \to T_3^{t+1}: 0 \to 1$, де T_r – вихід r-го елемента пам'яті. Якщо тригер $П_3$ перемикається швидше, ніж $П_1$ і $П_2$, то автомат деякий час буде перебувати у стані a_4 з кодом 011 (рис. 7.7б).

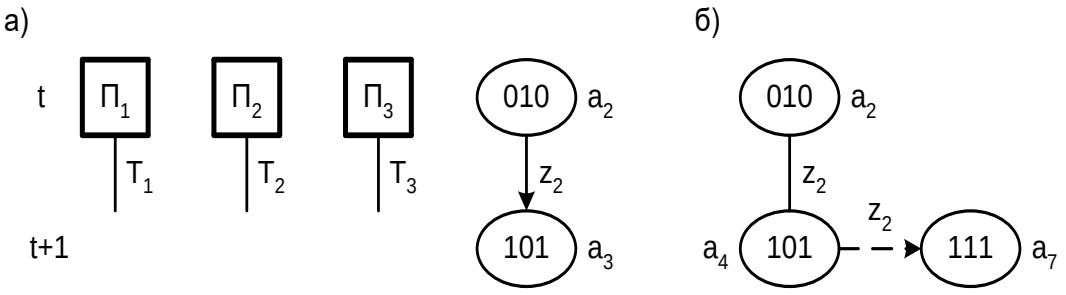


Рис. 7.7 – Збій в роботі автомата

Цей код ланцюгом зворотного зв'язку надходить на вхід комбінаційної схеми, і це може призвести до зміни функцій збудження пам'яті таким способом, що автомат перейде, наприклад, у стан a_5 з кодом 111 (рис. 7.7б) і залишиться в цьому стані. Така ситуація називається збоєм і призводить до неправильного функціонування автомата.

Розглянуте явище розкиду часу перемикання елементів пам'яті називається гонками, які і є причиною збою. Найбільш ефективним методом боротьби з гонками є двотактна синхронізація, за якої число елементів пам'яті фактично подвоюється. У цьому разі кожен елемент пам'яті дублюється, що призводить до дворівневої схеми пам'яті. Перший рівень пам'яті синхронізується сигналом C_1, другий – сигналом C_2 (рис. 7.8).

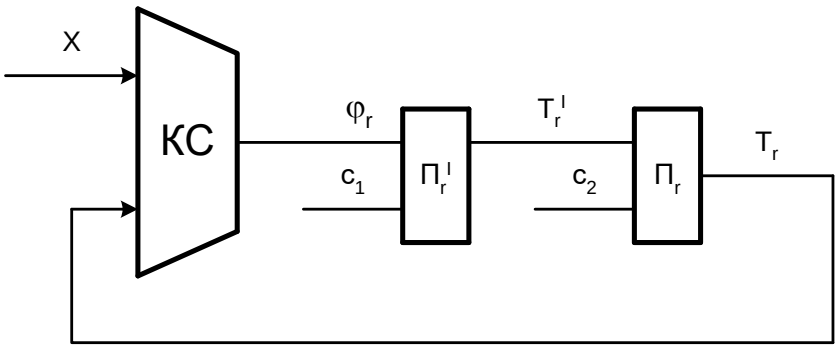


Рис. 7.8 – Двотактна синхронізація пам'яті структурного автомата

Розглянемо тимчасову діаграму роботи цієї схеми (рис. 7.9). У момент часу $t=0$ обидва елементи пам'яті $\Pi_r^{'}$ і Π_r знаходяться в однаковому стані. Нехай до моменту $t=1$ сформувалися правильні значення функцій збудження φ_r. У момент $t=1$ подається сигнал C_1, що викликає переключення тригера $\Pi_r^{'}$. Оскільки $C_2=0$, то тригер Π_r знаходиться у стабільному стані, інформація T_r з його виходу надходить до КС і підтримує стабільне значення функцій збудження пам'яті. Тривалість сигналу C_1 повинна бути достатньою для перемикання тригера першого рівня пам'яті.

У момент часу $t=2$ тригер $T_r^{'}$ містить новий стан і подається сигнал $C_2=1$, що приводить до перезапису інформації з $\Pi_r^{'}$ в Π_r .

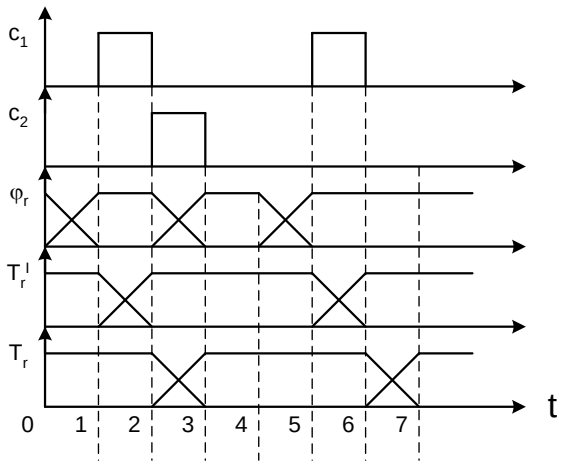


Рис. 7.9 – Тимчасова діаграма роботи схеми з двотактною синхронізацією

За час перемикання тригера Π_r можлива зміна функцій φ_r , але це не впливає на кінцевий стан тригера Π_r, оскільки $C_1=1$, і стан тригера $\Pi_r^{'}$ є стабільним. До моменту часу $t=3$ перемикання тригера Π_r завершується, і обидва тригери містять однакову інформацію. У момент часу $t=4$ змінюються значення логічних умов $x_l \in X$, і починається зміна функцій збудження пам'яті. До моменту часу $t=5$ сформовані правильні значення функцій збудження пам'яті, тобто ситуація ідентична моменту часу $t=1$. Функціонування схеми триває циклічно.

Якщо обидва елементи $\Pi_r^{'}$ і Π_r реалізовані в одному корпусі, то така схема називається двотактним тригером. При цьому нераціонально мати два входи синхронізації, тому використовується залежність $C_2=\overline{C_1}$. Функціональна схема й умовне позначення двотактного RS-тригера наведені на рис. 7.10.

Після ввімкнення живлення схеми стан тригера може бути довільним. Оскільки в момент часу $t=0$ (коли схема починає функціонувати) автомат повинен бути встановлений в певний стан, то

кожен тригер має два асинхронні входи установки: S_a (установка в одиницю) і R_a (установка в нуль). Наприклад, на рис. 7.11 показана функціональна схема і умовне позначення синхронного RS-тригера з асинхронними установчими входами.

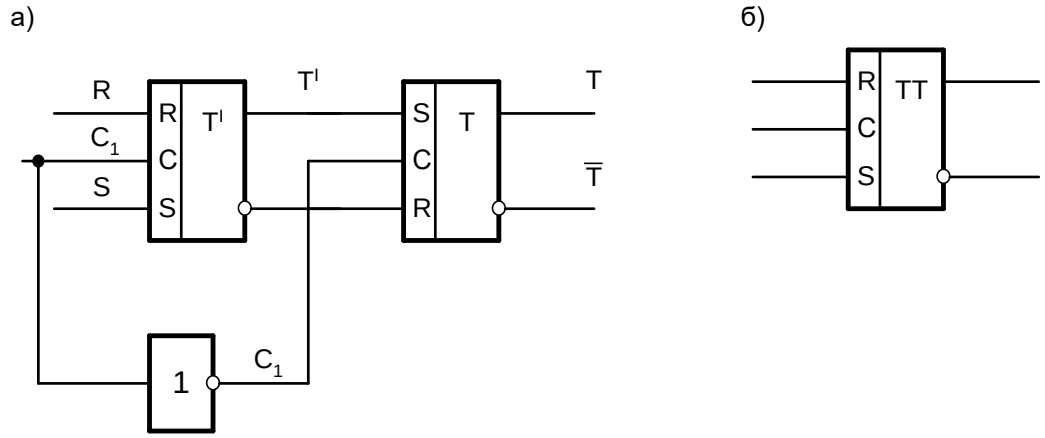


Рис. 7.10 – Функціональна схема (а) й умовне позначення (б) двотактного RS-тригера

Якщо, наприклад, $S_a = 1$, $R_a = 0$, то $\overline{T} = 0$ і $T = 1$. За $R_a = S_a = 0$ схема реагує на зміну входів R і S за $C = 1$.

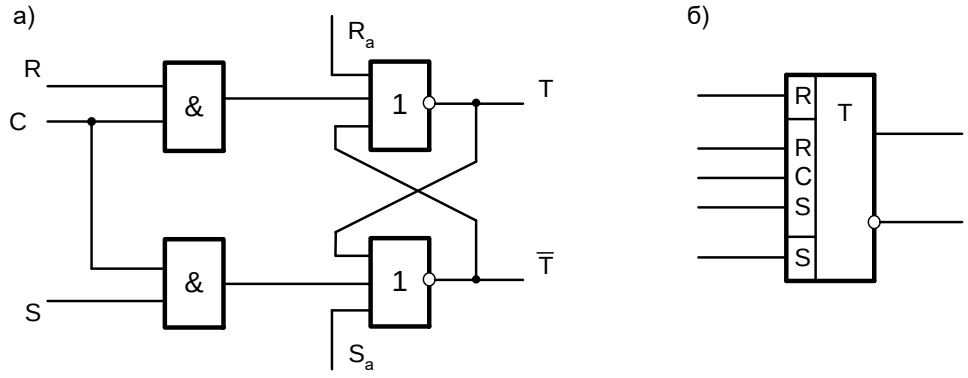


Рис. 7.11 – Функціональна схема (а) й умовне позначення (б) синхронного RS-тригера з установчими входами

Отже, для реалізації пам’яті структурних автоматів використовуються двотактні тригери з асинхронними установчими входами. Зазвичай автомат вмикає генератор одиничних імпульсів ГОІ, який при ввімкненні живлення генерує одиничний імпульс Start, що

подається на установчі входи тригерів. Наприклад, для установки схеми в початковий стан 101 вихід ГОІ повинен бути підключений до асинхронного входу S тригера T_1, входу R – T_2 і входу S – T_3 (рис. 7.12).

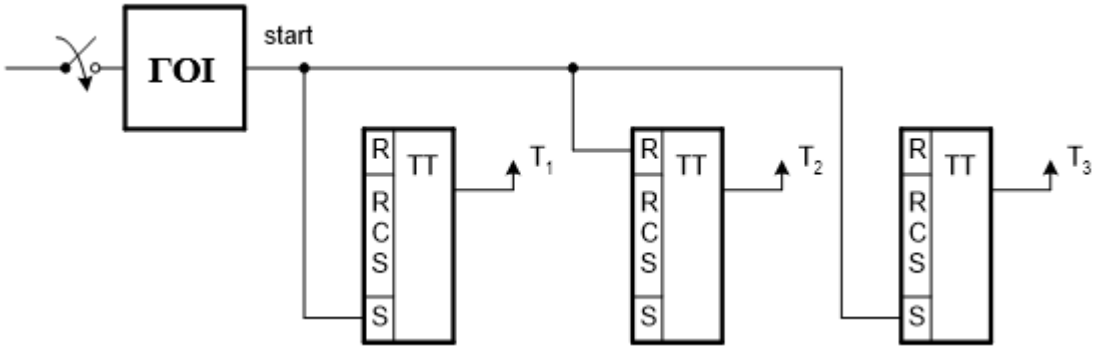


Рис. 7.12 – Початкова установка пам'яті автомата

Сьогодні для реалізації пам'яті структурних автоматів використовуються такі тригери.

1. *D-тригер.* Цей тригер називається елементом затримки і функціонує у такий спосіб (табл. 7.5).

Таблиця 7.5

Таблиця переходів синхронного D-тригера

C^t	D^t	T^t	T^{t+1}
0	*	*	T^t
1	0	*	0
1	1	*	1

Стан T^{t+1} залежить тільки від вхідного сигналу D^t, якщо $C^t = 1$. Для цього тригера немає заборонених вхідних наборів. Таблиця входів D-тригера приведена в табл. 7.6. З цієї таблиці випливає, що функція збудження D^t повністю визначається станом переходу тригера, тобто справедливою є рівність $D^t = T^{t+1}$.

Наявність одного входу для функції збудження дає змогу зменшити число виходів комбінаційної схеми, порівняно з RS-тригером. Отож, D-тригер (рис. 7.13) переважно використовується для організації пам'яті

структурних автоматів. Ця ж особливість привела до того, що тільки D-тригери використовуються в пристроях управління.

Таблиця 7.6

Таблиця D-тригера

T^t	T^{t+1}	D^t
0	0	0
0	1	1
1	0	0
1	1	1

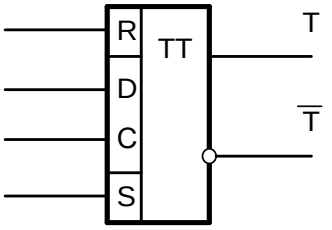


Рис. 7.13 – Умовне позначення двотактного D-тригера з установчими входами

2. *JK-тригер.* Цей елемент пам'яті подібний до RS-тригера, однак у нього немає заборонених комбінацій функцій збудження. Ситуація $J = K = 1$ викликає зміну стану тригера на протилежний і називається перемиканням (табл. 7.7).

В іншому разі вхід J є аналогом входу S, вхід K – аналогом входу R. Побудуємо таблицю входів JK-тригера. Переходу $\langle T^t, T^{t+1} \rangle = 00$ відповідають вхідні комбінації $J^t K^t = 00$ і 01, тобто маємо $JK = 0*$. Переходу $\langle T^t, T^{t+1} \rangle = 11$ відповідають вхідні комбінації 00 і 10, тобто маємо $JK = *0$. Переходу $\langle T^t, T^{t+1} \rangle = 01$ відповідають вхідні комбінації 10 і 11, тобто $JK = 1*$. Переходу $\langle T^t, T^{t+1} \rangle = 10$ відповідають вхідні комбінації 01 і 11, тобто $JK = *1$. Отже, таблиця входів JK-тригера має вигляд, наведений у табл. 7.8.

Через можливість перемикання за $J = K = 1$, JK-тригер повинен бути дворівневим, оскільки вихід $T^{t+1} = \overline{T}^t$, що може привести до явища генерації, якщо у схемі не використовується двотактна синхронізація. Умовне позначення двотактного JK-тригера повністю аналогічне рис. 7.6б, якщо входи R і S замінити входами J і K відповідно.

Таблиця 7.7

Таблиця переходів синхронного JK-тригера

C^t	J^t	K^t	T^t	T^{t+1}	коментар
0	*	*	*	T^t	зберігання
1	0	0	0	0	
1	0	0	1	1	
1	0	1	0	0	скидання
1	0	1	1	0	
1	1	0	0	1	установка
1	1	0	1	1	
1	1	1	0	1	перемикання
1	1	1	1	0	

Таблиця 7.8

Таблиця входів JK-тригера

T^t	T^{t+1}	J^t	K^t
0	0	0	*
0	1	1	*
1	0	*	1
1	1	*	0

3. *Т-тригер.* Цей елемент пам’яті має один вхід (табл. 7.9). Якщо $T^t = 0$, то тригер зберігає старий стан, за $T^t = 1$ тригер переходить у протилежний стан. Щоб уникнути плутанини, вихід Т-тригера в табл. 7.9 позначений буквою Q.

Таблиця 7.9

Таблиця переходів Т-тригера

T^t	Q^t	Q^{t+1}	коментар
0	0	0	зберігання
0	1	1	
1	0	1	перемикання
1	1	0	

Через наявність режиму перемикання Т-тригер завжди є двотактним. Таблиця входів Т-тригера (табл. 7.10) дає змогу отримати залежність

$$T^t = Q^t \oplus Q^{t+1}.$$

Таблиця 7.10

Таблиця входів Т-тригера

Q^t	Q^{t+1}	T^t
0	0	0
0	1	1
1	0	1
1	1	0

Умовне позначення синхронного двотактного Т-тригера з установчими входами повністю збігається з рис. 7.13, якщо замінити вхід D на вхід T.

4. *RS-тригер з інверсним управлінням.* Цей тригер повністю аналогічний RS-тригеру з прямим управлінням, але робочими сигналами тут є нулі (табл. 7.11).

Таблиця 7.11

Таблиця переходів $\overline{R}\overline{S}$-тригер

$\overline{R}$	$\overline{S}$	T^t	T^{t+1}	коментар
0	0	0	*	заборона
0	0	1	*	
0	1	0	0	скидання
0	1	1	0	
1	0	0	1	установка
1	0	1	1	
1	1	0	0	зберігання
1	1	1	1	

Щоб наголосити на тому, що вхідні сигнали є інверсними, схема називається $\overline{R}\overline{S}$-тригер. Оскільки таблиці переходів RS- і $\overline{R}\overline{S}$-тригерів збігаються, якщо поміняти для вхідних сигналів нулі на одиниці і навпаки, то це буде справедливо і для таблиці входів (табл. 7.12).

Таблиця 7.12

Таблиця входів $\overline{R}\overline{S}$ -тригера

T^t	T^{t+1}	$\overline{R}^t$	$\overline{S}^t$
0	0	*	1
0	1	1	0
1	0	0	1
1	1	1	*

На рис. 7.14 показано умовне позначення двотактного $\overline{R}\overline{S}$ -тригера з прямими установчими входами.

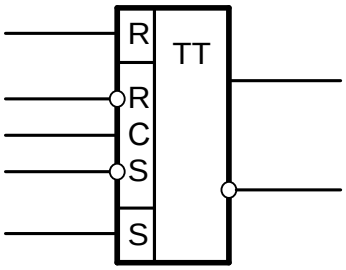


Рис. 7.14 – Умовне позначення двотактного RS-тригера з інверсним керуванням і прямими установчими входами

Зауважимо, що промисловість випускає тригери як у вигляді окремих мікросхем, так і у вигляді регістрів. Регістр – це сукупність тригерів, призначена для прийому, зберігання і видачі слова інформації. Якщо регістр реалізований як одна мікросхема, то всі його тригери мають спільний вхід скидання R і загальний вхід синхронізації C. На рис. 7.15 показаний зв'язок регістра з іншими елементами автомата.

7.3 Канонічний синтез структурного автомата Мілі

Методи синтезу автоматів залежать від методу уявлення закону його поведінки. Розглянемо застосування канонічного методу структурного синтезу для двох основних форм завдання автомата Мілі.

Табличний метод синтезу. Нехай автомат Мілі S_8 заданий таблицею переходів (табл. 7.13) і таблицею виходів (табл. 7.14).

Таблиця 7.13

Таблиця переходів автомата S_8

α_m / z_f	α_1	α_2	α_3	α_4	α_5
z_1	α_1	α_2	α_3	α_4	α_5
z_2	α_2	α_3	α_4	α_5	α_1
z_3	α_3	α_4	α_5	α_1	α_2

Таблиця 7.14

Таблиця виходів автомата S_8

α_m / z_f	α_1	α_2	α_3	α_4	α_5
z_1	ω_1	ω_4	ω_3	ω_2	ω_3
z_2	ω_2	ω_1	ω_4	ω_3	ω_2
z_3	$\omega 3$	ω_2	ω_1	ω_4	ω_1

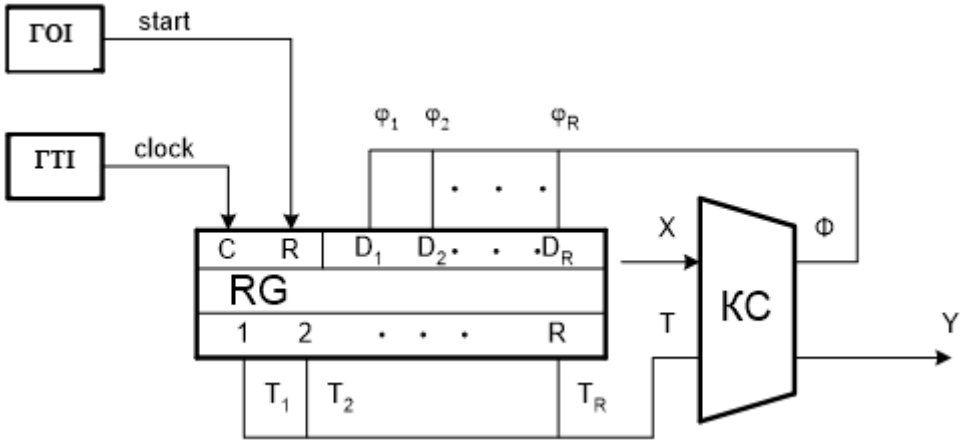


Рис. 7.15 – Використання регістра як пам'яті

Розглянемо задачу синтезу схеми автомата S_8 в базисі І-НЕ з використанням синхронного двотактного JK-тригера.

1. Формування структурних алфавітів. З аналізу табл. 7.13 і 7.14 випливає, що абстрактний автомат Мілі S_8 має такі характеристики: $Z=\{z_1,z_2,z_3\}$, $F=3$, $W=\{\omega_1,\omega_2,\omega_3,\omega_4\}$, $G=4$ і $A=\{a_1,\ldots,a_5\}$, $M=5$. Отже, для кодування F букв вхідного алфавіту достатньо $L=\lceil \log_2 F \rceil = 2$ букви вхідного структурного алфавіту, тобто $X=\{x_1,x_2\}$. Для кодування G букв вихідного алфавіту достатньо $N=\lceil \log_2 G \rceil = 2$ букви вихідного структурного алфавіту, тобто $Y=\{y_1,y_2\}$. Для кодування M букв алфавіту станів достатньо $R=\lceil \log_2 M \rceil = 3$ літери алфавіту внутрішніх

змінних, тобто $T=\{T_1,T_2,T_3\}$. Закодуємо ці літери у такий спосіб, як показано в табл. 7.15.

Зауважимо, що в цьому прикладі кодування букв виконується довільно, єдина умова – початковий стан a_1 кодується нульовим кодом. У загальному випадку результат кодування значно впливає на кількість логічних елементів у схемі автомата. Отож, існує велика кількість методів кодування букв вхідного, вихідного і внутрішнього алфавітів автомата.

Таблиця 7.15

Кодування букв абстрактних алфавітів автомата Мілі S_8

z_f	x_1	x_2	ω_g	y_1	y_2	a_m	T_1	T_2	T_3
z_1	0	0	ω_1	0	0	a_1	0	0	0
z_2	0	1	ω_2	0	1	a_2	0	0	1
z_3	1	0	ω_3	1	0	a_3	0	1	0
			ω_4	1	1	a_4	0	1	1
						a_5	1	0	0

2. Формування структурної таблиці виходів. З цією метою необхідно в таблиці виходів замінити літери z_f їхніми кодами: $K(z_f)$, ω_g – $K(\omega_g)$ і a_m – $K(a_m)$. У нашому випадку за таблицею абстрактного автомата Мілі (табл. 7.14) формується табл. 7.16, яка задає поведінку структурного автомата Мілі S_8.

Таблиця 7.16

Структурна таблиця виходів автомата Мілі S_8

$T_1T_2T_3$ / x_1x_2	000	001	010	011	100
00	00	11	10	01	10
01	01	00	11	10	01
10	10	01	00	11	00

Як видно з табл. 7.16, змінні z_f замінені змінними x_1x_2, змінні ω_g замінені змінними y_1y_2, змінні стану a_m замінені змінними $T_1T_2T_3$. Всі змінні відповідають поточному моменту часу t ($t = 0,1,2,\ldots$).

3. Формування системи вихідних функцій. Очевидно, структурна таблиця виходів автомата аналогічна N таблицям істинності для функцій $y_1,\ldots,y_N$, входами яких є змінні $x_1,\ldots,x_L$, $T_1,\ldots,T_{12}$. З табл. 7.16 маємо таку систему функцій:

$$\begin{aligned} y_1 &= F_{16} \vee F_1 \vee F_2 \vee F_{10} \vee F_{11} \vee F_{19} \vee F_4, \\ y_2 &= F_8 \vee F_1 \vee F_{17} \vee F_{10} \vee F_3 \vee F_{19} \vee F_{12}. \end{aligned} \qquad (7.8)$$

Номери термів у системі (7.8) відповідають наборам $\langle x_1x_2T_1T_2T_3\rangle$, що визначають клітини табл. 7.8. Наприклад, терм F_{16} відповідає набору 10000 і $F_{16} = x_1\bar{x}_2\bar{T}_1\bar{T}_2\bar{T}_3$.

4. Формування структурної таблиці переходів. З цією метою необхідно в таблиці переходів замінити літери абстрактних алфавітів їхніми кодами. У нашому випадку це приводить до табл. 7.17.

Таблиця 7.17

Структурна таблиця переходів автомата Мілі S_8

$T_1T_2T_3$ / x_1x_2	000	001	010	011	100
00	000	001	010	011	100
01	001	010	011	100	000
10	010	011	100	000	001

Зауважимо, що вхідний набір відповідає моменту часу t, а змінні $T_1 - T_3$ в клітинах таблиці відповідають моменту часу $t+1$ ($t = 0,1,2,\ldots$). Отже, структурна таблиця переходів задає залежність $T_r^{t+1} = T_r(T^t, X^t)$ $(r = 1,\ldots,R)$. Однак для управління пам'яттю автомата необхідно мати залежність функцій збудження від стану і вхідного сигналу в момент часу t:

$$\Phi_r^t = \Phi_r(T^t, X^t) \quad (r = 1,..., R). \tag{7.9}$$

Для формування системи (7.9) необхідно побудувати таблицю функцій збудження пам'яті, для чого використовується таблиця входів JK-тригера (табл. 7.8).

Наприклад, в рядку табл. 7.17 з номером 0 (набір 00000) відбувається перехід зі стану з кодом 000 у стан з кодом 000, тобто для всіх тригерів автомата відбувається перехід $\langle 0,0\rangle = \left\langle T_r^t, T_r^{t+1}\right\rangle$. Як випливає з табл. 7.8, для всіх тригерів маємо рівняння $J_r^t = 0, K_r^t = *$. Оскільки за структурного синтезу не розглядається задача оптимізації схеми автомата, то всі символи $*$ замінюються нулями. Далі, набору 10010 (номер 18) відповідає перехід зі стану з кодом 010 у стан з кодом 100. Це відповідає таким переходам: $\left\langle T_1^t, T_1^{t+1}\right\rangle = \langle 0,1\rangle$, $\left\langle T_2^t, T_2^{t+1}\right\rangle = \langle 1,0\rangle$, $\left\langle T_3^t, T_3^{t+1}\right\rangle = \langle 0,0\rangle$. З табл. 7.8 маємо значення функцій збудження $J_1^t = 1$, $K_1^t = 0$, $J_2^t = 0$, $K_2^t = 1$, $J_3^t = K_3^t = 0$. З використанням подібних перетворень і побудована таблиця функцій збудження пам'яті автомата S_8 (табл. 7.18).

Таблиця 7.18

Таблиця функцій збудження пам'яті автомата S_8

$T_1T_2T_3$ / x_1x_2	000	001	010	011	100
00	000000	000000	000000	000000	000000
01	000010	001001	000010	100101	010000
10	001000	001000	100100	000101	010010

У кожній клітині цієї таблиці міститься шість змінних, що відповідають значенням функцій $J_1, K_1, J_2, K_2, J_3, K_3$.

5. *Формування системи функцій збудження пам'яті автомата.* Система (7.9) будується аналогічно до системи (7.8), але з використанням таблиці функцій збудження пам'яті. У нашому випадку маємо:

$$
\begin{aligned}
&J_1 = F_{18} \vee F_{11};\\
&K_1 = F_{12} \vee F_{20};\\
&J_2 = F_{16} \vee F_9 \vee F_{17};\\
&K_2 = F_{18} \vee F_{11} \vee F_{19};\\
&J_3 = F_8 \vee F_{10} \vee F_{20};\\
&K_3 = F_9 \vee F_{11} \vee F_{19}.
\end{aligned}
\tag{7.10}
$$

6. Перетворення систем булевих функцій з урахуванням обмежень елементного базису. В даному випадку використовується базис І-НЕ без обмежень на число входів, тому перетворення систем (7.8) і (7.10) зводиться до використання законів подвійної інверсії і правил де Моргана. В результаті будується система функцій (7.11), в якій, наприклад:

$$
\begin{aligned}
&y_1 = \overline{\overline{F}_{16}\overline{F}_1\overline{F}_2\overline{F}_{10}\overline{F}_{11}\overline{F}_{19}\overline{F}_4} = \overline{\overline{x_1\bar{x}_2\overline{T}_1\overline{T}_2\overline{T}_3} \cdot \overline{\bar{x}_1\bar{x}_2\overline{T}_1\overline{T}_2T_3} \cdots x_1\bar{x}_2T_1\overline{T}_2\overline{T}_3};\\
&\vdots\\
&J_3 = \overline{\overline{F}_8 \vee \overline{F}_{10} \vee \overline{F}_{20}} = \overline{\bar{x}_1x_2\overline{T}_1\overline{T}_2\overline{T}_3} \cdot \overline{\bar{x}_1x_2\overline{T}_1T_2\overline{T}_3} \cdot \overline{x_1\bar{x}_2T_1\overline{T}_2\overline{T}_3};\\
&K_3 = F_9 \vee F_{11} \vee F_{19} = \overline{\bar{x}_1x_2\overline{T}_1\overline{T}_2T_3} \cdot \overline{\bar{x}_1x_2\overline{T}_1T_2T_3} \cdot \overline{x_1\bar{x}_2\overline{T}_1T_2T_3}.
\end{aligned}
\tag{7.11}
$$

7. Синтез логічної схеми автомата. Очевидно, комбінаційна схема буде складатися з двох рівнів елементів І-НЕ. На першому рівні реалізуються схеми для змінних $\overline{F}_h$ ($h = 0,\ldots,4,8,\ldots,12,16,\ldots,20$), а на другому рівні реалізуються схеми для функцій $y_n \in Y$ і $\varphi_r \in \Phi$ відповідно до системи (7.11). Пам’ять автомата містить три JK-тригери, що керуються сигналами *Clock* і *Start* (рис. 7.16).

Отже, застосування канонічного методу дало змогу за таблицями, що задає закон функціонування абстрактного автомата Мілі S_8, побудувати схему з аналогічним законом функціонування.

Розглянемо задачу синтезу схеми автомата S_9 в базисі І-НЕ з використанням синхронного Т-тригера. Природно, що можна перейти від графа до таблиць переходів і входів, звівши цим завдання до вже розглянутої схеми. Однак синтез можна провести і за графом.

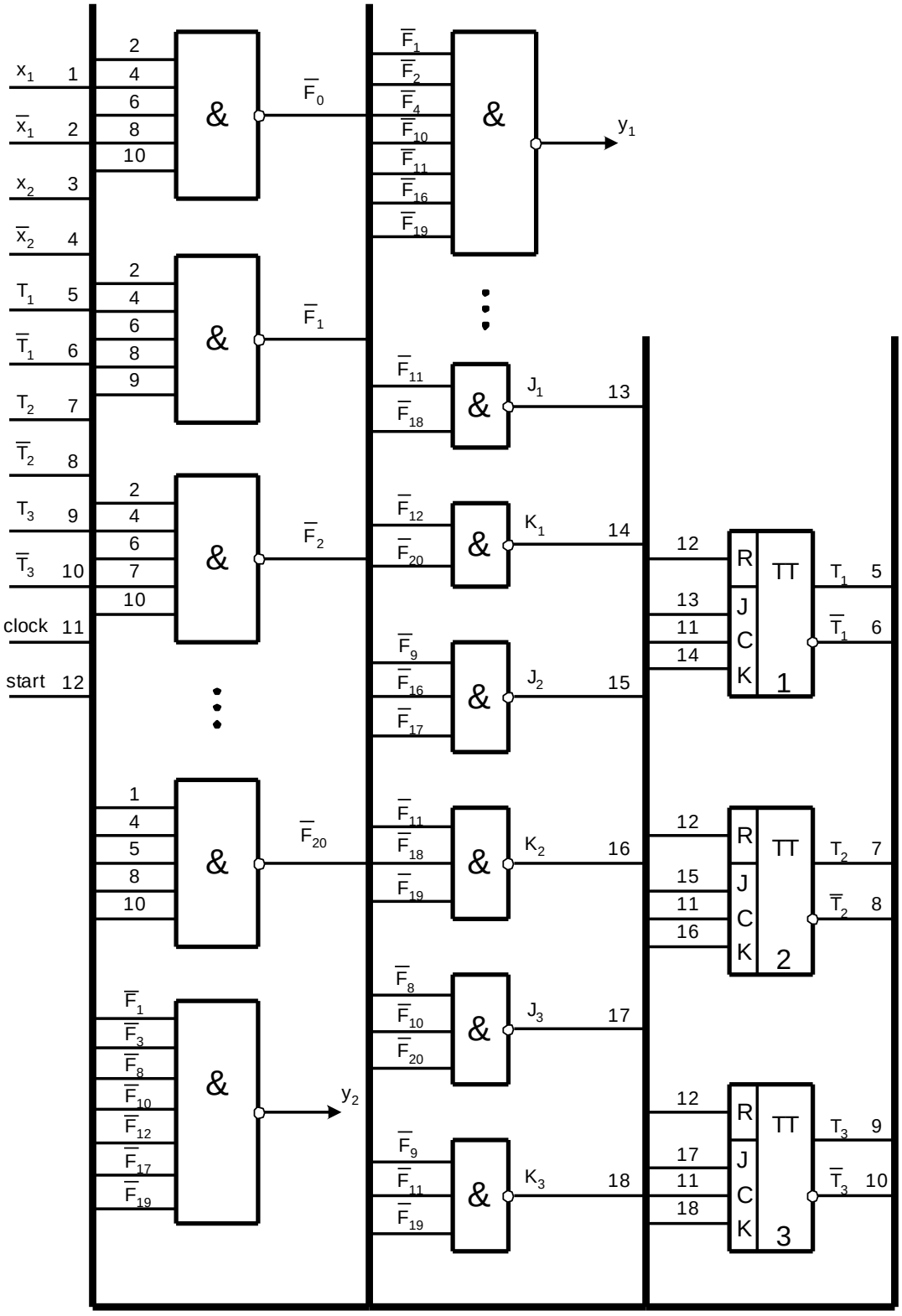


Рис. 7.16 – Функціональна схема структурного автомата Мілі S_8

Графічний метод синтезу. Нехай абстрактний автомат Мілі S_9 заданий графом (рис. 7.17).

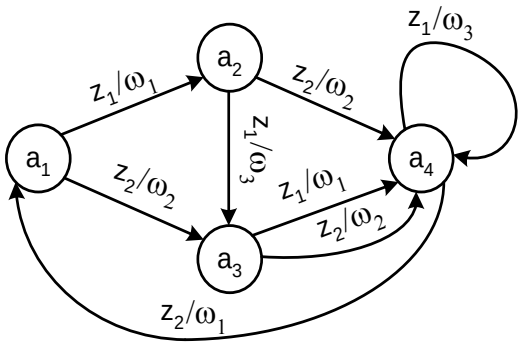


Рис. 7.17 – Граф абстрактного автомата Мілі S_9

1. Формування структурних алфавітів. З рис. 7.17 випливає, що в автомата S_9 $Z=\{z_1,z_2\}$, $F=2, W=\{\omega_1,\omega_2,\omega_3\}$, $G=3$ і $A=\{a_1,\ldots,a_4\}$, $M=4$. Для переходу від букв $z_f \in Z$ до кодів $K(z_f)$ досить $L=\lceil \log_2 F \rceil = 1$ букви вхідного структурного алфавіту, тобто $X=\{x_1\}$. Для переходу від букв $\omega_g \in W$ до кодів $K(\omega_g)$ досить $N=\lceil \log_2 G \rceil = 2$ букви вихідного структурного алфавіту $Y=\{y_1,y_2\}$.

Для кодування станів $a_m \in A$ кодами $K(a_m)$ досить $r=\lceil \log_2 M \rceil = 2$ внутрішні змінні, тобто $T=\{T_1,T_2\}$. Оскільки як елемент пам'яті використовується Т-тригер, то нехай $T=\{Q_1,Q_2\}$.

Таблиця кодування букв $z_f \in Z$, $\omega_g \in W$ і $a_m \in A$ (табл. 7.19) дає один з варіантів для автомата S_9.

Таблиця 7.19

Кодування букв абстрактних алфавітів автомата Мілі S_9

z_f	x_1	ω_g	y_1	y_2	a_m	T_1	T_2
z_1	0	ω_1	0	0	a_1	0	0
z_2	1	ω_2	0	1	a_2	0	1
		ω_3	1	0	a_3	1	0
					a_4	1	1

2. Формування структурного графа автомата. З цією метою необхідно в початковому графі замінити літери абстрактного алфавіту відповідними кодами. У нашому випадку за графом (рис. 7.17) і таблицею кодування (табл. 7.19) формується граф на рис. 7.18.

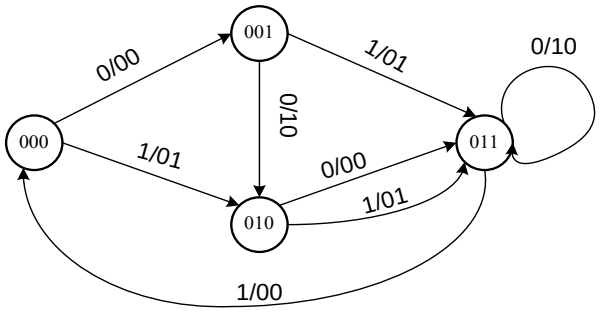


Рис. 7.18 – Структурний граф автомата S_9

3. Відмітка дуг графа функціями збудження пам'яті. Розглянемо перехід $\langle a_1, a_2 \rangle$, що відповідає переключенню пам'яті $\left\langle Q_1^t, Q_1^{t+1} \right\rangle = \langle 0,0 \rangle$, $\left\langle Q_2^t, Q_2^{t+1} \right\rangle = \langle 0,1 \rangle$. З таблиці входів Т-тригера маємо $T_r^t = Q_r^t \oplus Q_r^{t+1}$, отже, $T_1 = 0$, $T_2 = 1$. Аналогічно знайдемо функції збудження T_1, T_2 для всіх переходів автомата. Додамо ці функції як третю складову для кожної дуги і отримаємо повний структурний граф автомата. З цього графа можна отримати диз'юнктивні нормальні форми для функцій збудження пам'яті автомата і для його вихідних функцій. Однак такий підхід ускладнює процес синтезу. Для спрощення процесу синтезу автомата замінимо коди буквами, як показано на рис. 7.19.

4. Формування системи функцій автомата. Якщо перехід з вершини $\overline{Q_1}\overline{Q_2}$ відбувається за сигналом $\bar{x}_1$, то терм $F_0 = \overline{Q_1}\overline{Q_2}\bar{x}_1$ входить в усі функції, записані на відповідній дузі. Для автомата S_9 терм F_0 входить тільки в ДНФ функції T_2, що відповідає переходу з a_1 в a_2 по z_1. Застосувавши ці міркування, отримаємо систему:

$$y_1 = F_2 \vee F_6 = \overline{Q}_1 Q_2 \bar{x}_L \vee Q_1 Q_2 \bar{x}_L;$$
$$y_2 = F_1 \vee F_3 \vee F_5;$$
$$T_1 = F_1 \vee F_2 \vee F_3 \vee F_7;$$
$$T_2 = F_0 \vee F_2 \vee F_4 \vee F_5 \vee F_7.$$

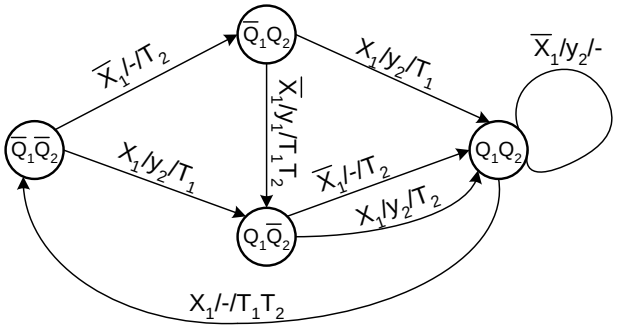


Рис. 7.19 – Повний структурний граф автомата S_9

5. Формування перетвореної системи функцій автомата. Оскільки схема автомата реалізується в базисі І-НЕ, то для її реалізації необхідно застосувати закони подвійної інверсії і де Моргана. Після подібних перетворень маємо таку систему:

$$\begin{aligned} y_1 &= \overline{\overline{F}_2 \overline{F}_6} = \overline{\overline{\overline{Q}_1 Q_2 \bar{x}_L} \cdot \overline{Q_1 Q_2 \bar{x}_L}}; \\ y_2 &= \overline{\overline{F}_1 \overline{F}_3 \overline{F}_5}; \\ T_1 &= \overline{\overline{F}_1 \overline{F}_2 \overline{F}_3 \overline{F}_7}; \\ T_2 &= \overline{\overline{F}_0 \overline{F}_2 \overline{F}_4 \overline{F}_5 \overline{F}_7}. \end{aligned} \qquad (7.12)$$

У разі обмежень елементного базису необхідно було б перетворити кожне рівняння системи (7.12) за правилами, розглянутим раніше для синтезу комбінаційних схем. Наприклад, за числа входів елемента «І» $S = 2$, формула для функції T_1 має вигляд:

$$T_1 = \overline{\overline{F}_1 \overline{F}_2 \overline{F}_3 \overline{F}_7} = \overline{\overline{\overline{Q}_1 \overline{Q}_2 x_L} \cdot \overline{\overline{Q}_1 Q_2 x_L} \cdot \overline{\overline{Q}_1 Q_2 x_L} \cdot \overline{Q_1 Q_2 x_L}}.$$

6. Синтез логічної схеми автомата в заданому базисі. Схема автомата S_9 (рис. 7.20) містить два рівні схем І-НЕ, перший з яких

реалізує терми $\overline{F}_h$ ($h=0,\ldots,7$), а другий – функції (рис. 7.20). Пам'ять автомата S_9 складається з двох Т-тригерів.

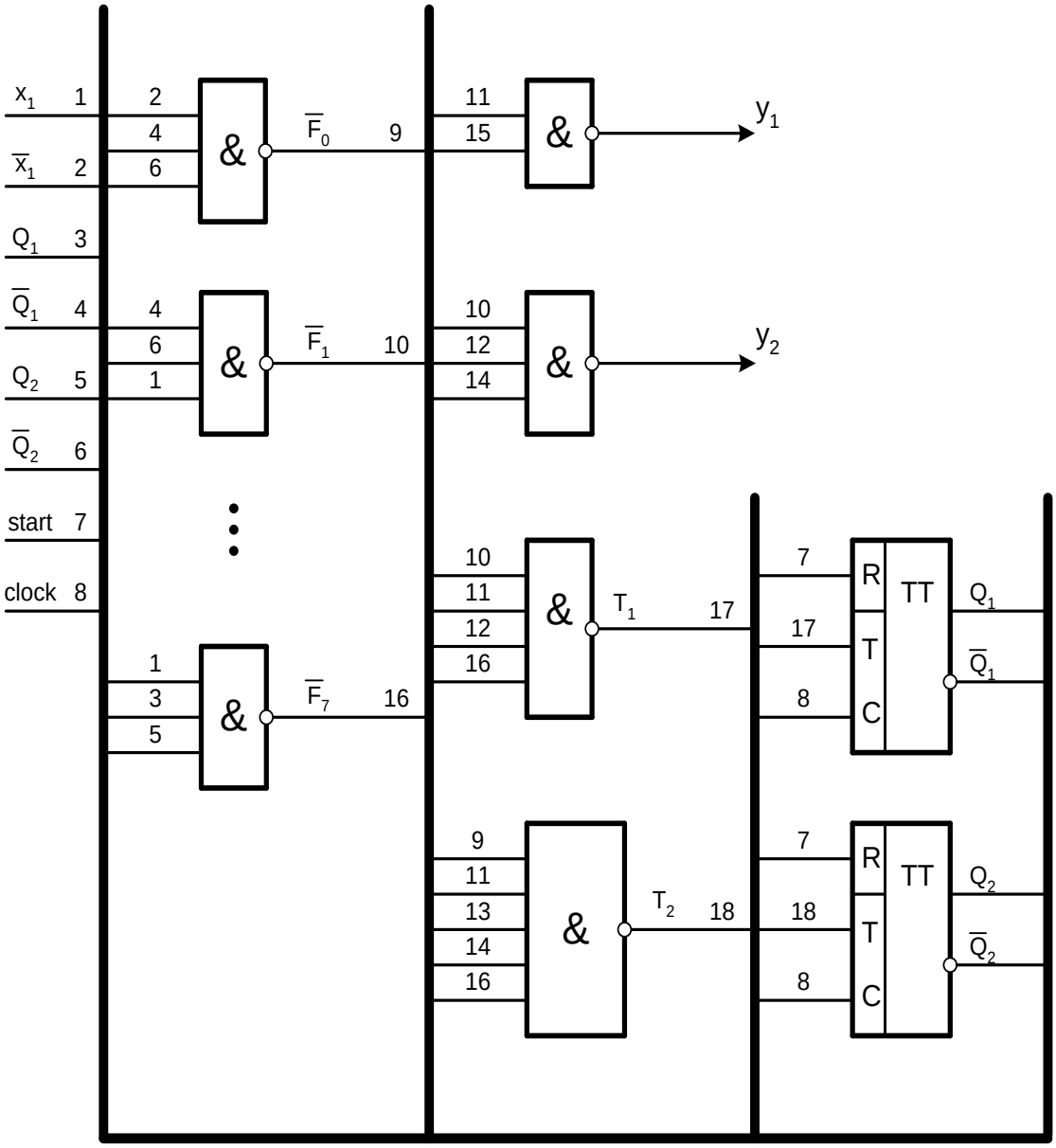


Рис. 7.20 – Логічна схема структурного автомата Мілі S_9

7.4 Канонічний синтез структурного автомата Мура

Для автомата Мура також можливі два підходи до синтезу схеми автомата. Це табличний і графічний методи.

Табличний метод синтезу. Нехай абстрактний автомат Мура S_{10} заданий зазначеною таблицею переходів (табл. 3.39). Розглянемо як приклад

завдання синтезу логічної схеми автомата S_{10} в базисі І-НЕ з використанням синхронного двотактного RS-тригера з прямим управлінням.

Таблиця 7.20

Зазначена таблиця переходів автомата Мура S_{10}

ω_g	ω_2	ω_4	ω_1	ω_3	ω_3	ω_2
a_m / z_f	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_2	a_3	a_4	a_5	a_6	a_1
z_2	a_3	a_4	a_5	a_6	a_1	a_2
z_3	a_1	a_2	a_3	a_4	a_5	a_6

1. Формування структурних алфавітів. З табл. 7.20 випливає, що абстрактний автомат Мура S_{10} має такі характеристики: $Z=\{z_1,z_2,z_3\}$, $F=3$, $W=\{\omega_1,\ldots,\omega_4\}$, $G=4$ і $A=\{a_1,\ldots,a_6\}$, $M=6$.

Отже, для кодування F букв вхідного алфавіту досить $L=\lceil\log_2 F\rceil=2$ літери структурного алфавіту і $X=\{x_1,x_2\}$. Для кодування G букв вихідного алфавіту досить $N=\lceil\log_2 G\rceil=2$ літери структурного алфавіту і $Y=\{y_1,y_2\}$. Для кодування M станів абстрактного автомата досить $R=\lceil\log_2 M\rceil=3$ внутрішні змінні і $T=\{T_1,T_2,T_3\}$. Закодуємо ці літери $z_f\in Z$, $\omega_g\in W$ і $a_m\in A$, як показано в табл. 7.21.

Таблиця 7.21

Кодування букв абстрактних алфавітів автомата Мура S_{10}

z_f	x_1	x_2	ω_g	y_1	y_2	a_m	T_1	T_2	T_3
z_1	0	0	ω_1	0	0	a_1	0	0	0
z2	0	1	ω_2	0	1	a_2	0	0	1
z3	1	0	ω_3	1	0	a_3	0	1	0
			ω_4	1	1	a_4	0	1	1
						a_5	1	0	0
						a_5	1	0	1

2. Формування структурної зазначеної таблиці переходів. Для виконання цього етапу необхідно в табл. 3.39 замінити літери всіх абстрактних алфавітів їхніми кодами з табл. 3.40. У нашому випадку це приводить до формування табл. 7.22.

3. Формування структурної зазначеної таблиці функцій збудження пам'яті автомата. Таблиця переходів дає залежність $T_r^{t+1} = T_r(X^t, T^t)\,(r = 1,...,R)$, а для синтезу схеми автомата необхідна залежність $\varphi_r^t = \varphi_r(T^t, X^t)$. Для формування цієї залежності необхідно використати таблицю входів RS-тригера (табл. 7.3) і побудувати таблицю функцій збудження пам'яті. Наприклад, абстрактний автомат S_{10} переходить зі стану a_2 у стан a_3 під дією сигналу z_1. У структурному автоматі це відповідає переключенню пам'яті з $K(a_2) = 001$ в $K(a_3) = 010$ за $x_1 = x_2 = 0$. Цей перехід пам'яті складається з перемикань її тригерів: $\left\langle T_1^t, T_1^{t+1} \right\rangle = \langle 0,0 \rangle$, тобто $R_1^t = *, S_1^t = 0$; $\left\langle T_2^t, T_2^{t+1} \right\rangle = \langle 0,1 \rangle$, тобто $R_2^t = 0, S_2^t = 1$; $\left\langle T_3^t, T_3^{t+1} \right\rangle = \langle 1,0 \rangle$, тобто $R_3^t = 1, S_3^t = 0$. Отже, для переходу $001 \to 010$ необхідно сформувати набір функцій збудження ∗00110, де змінні записані в порядку $R_1, S_1, R_2, S_2, R_3, S_3$.

Таблиця 7.22

Структурна зазначена таблиця переходів автомата Мура S_{10}

y_1y_2	01	11	00	10	10	01
$T_1T_2T_3$ / x_1x_2	000	001	010	011	100	101
00	001	010	011	100	101	000
01	010	011	100	101	000	001
10	000	001	010	011	100	101

Після аналогічного аналізу всіх перемикань пам'яті в табл. 7.22 формується структурна зазначена таблиця функцій збудження пам'яті

автомата Мура S_{10} (табл. 7.23). Оскільки питання оптимізації за канонічного синтезу не розглядаються, то всі невизначеності $*$ замінені нулями.

Таблиця 7.23

Структурна зазначена таблиця функцій збудження автомата Мура S_{10}

y_1y_2	**01**	**11**	**00**	**10**	**10**	**01**
$T_1T_2T_3$ / x_1x_2	000	001	010	011	100	101
00	0000001	000110	000001	011010	000001	100010
01	000100	000100	011000	011000	100000	100000
10	000000	000000	000000	000000	000000	000000

4. Формування системи функцій автомата. З табл. 7.23 маємо такі рівняння:

$$
\begin{aligned}
&y_1 = A_2 \vee A_4 \vee A_5 = \overline{T}_1T_2\overline{T}_3 \vee \overline{T}_1T_2T_3 \vee T_1\overline{T}_2\overline{T}_3;\\
&y_2 = A_1 \vee A_2 \vee A_6;\\
&R_1 = F_5 \vee F_{12} \vee F_{13} = \bar{x}_1\bar{x}_2T_1\overline{T}_2T_3 \vee \bar{x}_1x_2T_1\overline{T}_2\overline{T}_3 \vee \bar{x}_1x_2T_1\overline{T}_2T_3;\\
&S_1 = F_3 \vee F_{10} \vee F_{11};\\
&R_2 = F_3 \vee F_{10} \vee F_{11};\\
&S_2 = F_1 \vee F_8 \vee F_9;\\
&R_3 = F_1 \vee F_3 \vee F_5;\\
&S_3 = F_0 \vee F_2 \vee F_4.
\end{aligned}
\qquad (7.13)
$$

У системі (7.13) використані такі позначення: A_m – кон’юнкція внутрішніх змінних, що відповідає коду внутрішнього стану $a_m \in A$, наприклад, $K(a_3) = 010$ і $A_3 = \overline{T}_1T_2\overline{T}_3$. Функція y_1 (перша позиція першого рядка табл. 7.23) формується у станах a_2, a_4, $a_5 \in A$, звідки і випливає перше рівняння системи (7.13).

5. Перетворення системи функцій автомата. Як і раніше, для перетворення досить застосувати до кожного рівняння системи (7.13)

закони подвійної інверсії і закони де Моргана, після чого маємо, наприклад:

$$y_1 = \overline{\overline{A}_2 \overline{A}_4 \overline{A}_5};$$

$$R_1 = \overline{\overline{F}_5 \overline{F}_{12} \overline{F}_{13}}.$$

6. Синтез логічної схеми автомата. Схема буде складатися з двох рівнів елементів І-НЕ. На першому рівні формуються терми системи (7.13) – $\overline{A}_1, \overline{A}_2, \overline{A}_4, \overline{A}_5, \overline{A}_6, \overline{F}_0, ..., \overline{F}_5, \overline{F}_8, ..., \overline{F}_{13}$. На другому рівні формуються вихідні сигнали і функції збудження пам'яті автомата S_{10} (рис. 7.21).

Графічний метод синтезу. Нехай автомат Мура S_{11} заданий графом (рис. 7.22). Розглянемо задачу синтезу логічної схеми автомата S_{11} в базисі АБО-НЕ з використанням синхронного двотактного D-тригера.

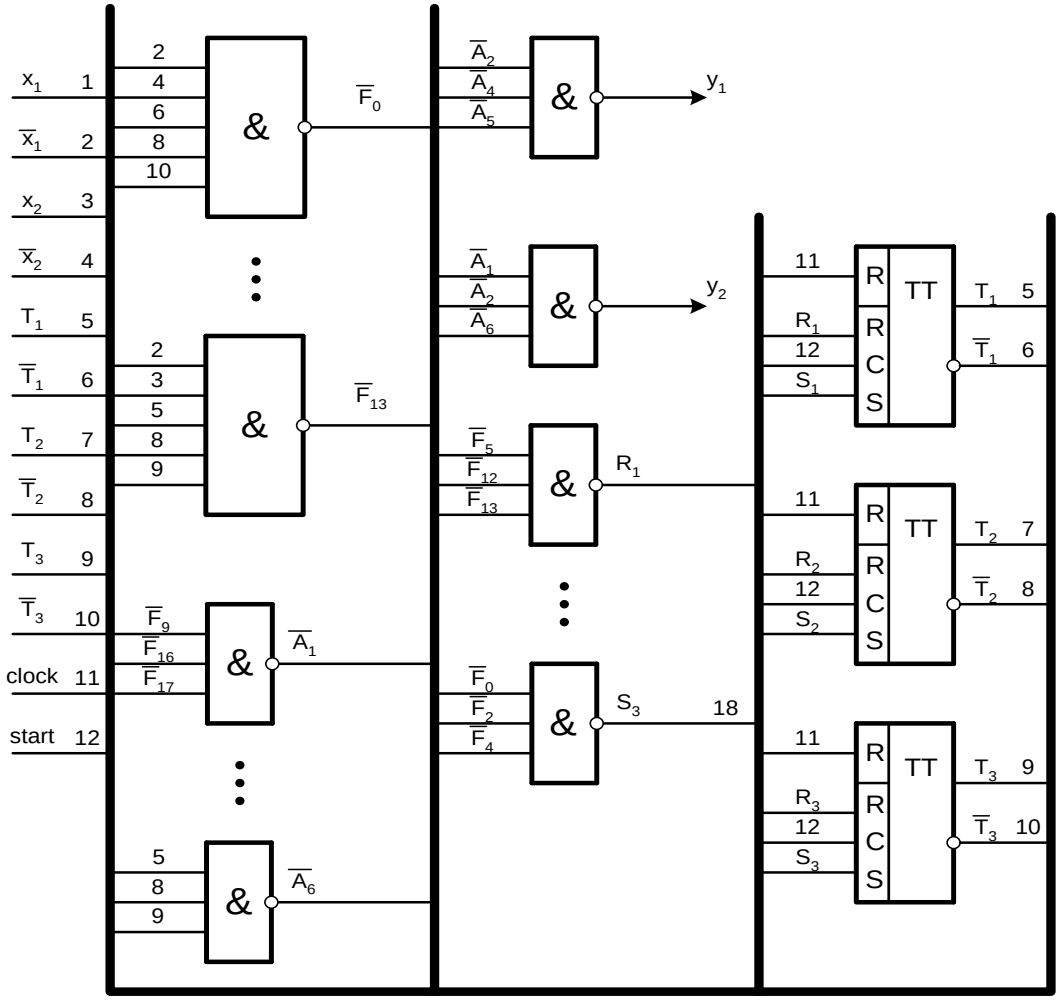


Рис. 7.21 – Логічна схема структурного автомата Мура S_{10}

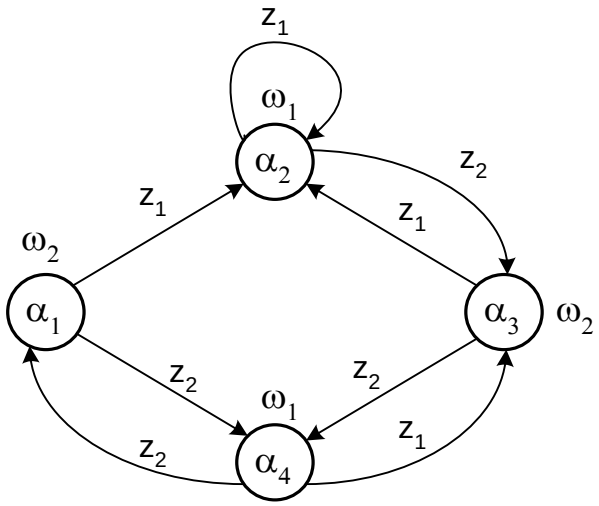


Рис. 7.22 – Граф абстрактного автомата Мура S_{11}

1. Формування структурних алфавітів. З графа автомата маємо $Z=\{z_1,z_2\}$, $F=2$, отже, $L=1$ і $X=\{x_1\}$; далі маємо множину $W=\{\omega_1,\omega_2\}$, $G=2$, отже, $R=2$ і $T=\{T_1,T_2\}$. Закодуємо літери $z_f \in Z$, $\omega_g \in W$ і $a_m \in A$, як показано в табл. 7.24.

Таблиця 7.24

Кодування букв абстрактних алфавітів автомата Мура S_{11}

z_f	x_1	ω_g	y_1	a_m	T_1	T_2	a_m	T_1	T_2
z_1	0	ω_1	0	a_1	0	0	a_3	1	0
z_2	1	ω_2	1	a_2	0	1	a_4	1	1

2. Формування структурного графа автомата. Замінивши літери z_f, ω_g і a_m їхніми кодами $K(z_f)$, $K(\omega_g)$ і $K(a_m)$, отримаємо з графа абстрактного автомата граф структурного автомата (рис. 7.23).

3. Відмітка дуг графа функціями збудження пам'яті автомата. Як випливає з таблиці входів D-тригера (табл. 3.6), функції збудження в момент часу t збігаються зі станами в момент часу $t+1$ ($t=0,1,\ldots$). Отже, і набір функцій збудження пам'яті автомата, що складається з D-тригерів, збігається з кодом стану переходу. Отже, в разі D-тригера досить

перенести коди станів переходу на відповідні дуги. Для нашого прикладу цей підхід породжує повний структурний граф автомата S_{11} (рис. 7.24).

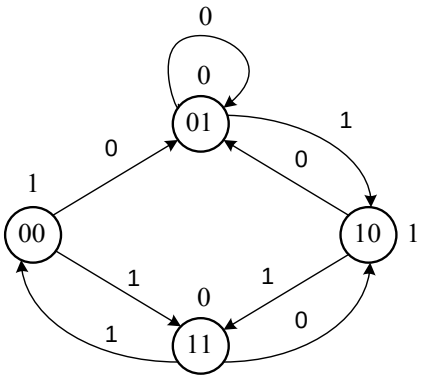


Рис. 7.23 – Граф структурного автомата Мура S_{11}

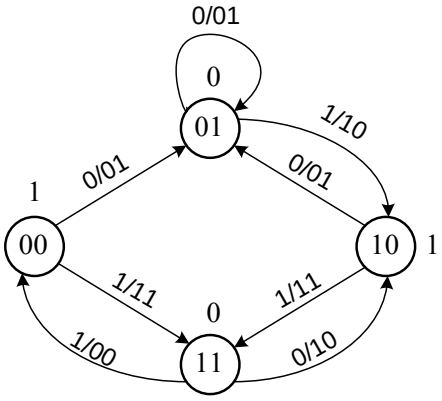


Рис. 7.24 – Повний структурний граф автомата Мура S_{11}

4. Формування системи функцій автомата. Оскільки базисом для синтезу комбінаційної схеми автомата є елементи АБО-НЕ, то рівняння повинні бути представлені в КНФ. Наприклад, переходу зі стану a_1 з кодом $K(a_1)=00$ за сигналом z_1 ($x_1=0$) відповідає макстерм $\Phi_0=\overline{T}_1\vee\overline{T}_2\vee\overline{x}_1$. Оскільки на переході з a_1 в a_2 формуються функції $D_1=0$, $D_2=1$, то терм Φ_0 входить в КНФ функції D_1. Якщо автомат знаходиться у стані a_2 з $K(a_2)=01$, то $y_1=0$, отже, терм $\overline{T}_1\vee T_2=A_2$ входить в КНФ функції y_1. Після аналізу графа (рис. 7.24) отримуємо таку систему функцій:

$$y_1 = A_2 \cdot A_4;$$
$$D_1 = \Phi_0 \cdot \Phi_7 \cdot \Phi_4 \cdot \Phi_2; \qquad (7.14)$$
$$D_2 = \Phi_3 \cdot \Phi_6 \cdot \Phi_7.$$

5. Формування перетвореної системи функцій автомата. Для перетворення системи (7.14) достатньо до кожного її рівняння застосувати закони подвійної інверсії і де Моргана. В результаті маємо систему:

$$y_1 = \overline{\overline{A}_2 \vee \overline{A}_4} = \overline{\overline{\overline{T}_1 \vee \overline{T}_2} \vee \overline{\overline{T}_1 \vee \overline{T}_2}};$$
$$D_1 = \overline{\overline{\Phi}_0 \vee \overline{\Phi}_7 \vee \overline{\Phi}_4 \vee \overline{\Phi}_2} = \overline{\overline{T_1 \vee T_2 \vee x_0} \vee \overline{\overline{T}_1 \vee \overline{T}_2 \vee \overline{x}_0} \vee \cdots}; \qquad (7.15)$$
$$D_2 = \overline{\overline{\Phi}_3 \vee \overline{\Phi}_6 \vee \overline{\Phi}_7}.$$

6. Синтез логічної схеми автомата в заданому базисі. Схема автомата S_{11} приведена на рис. 7.25.

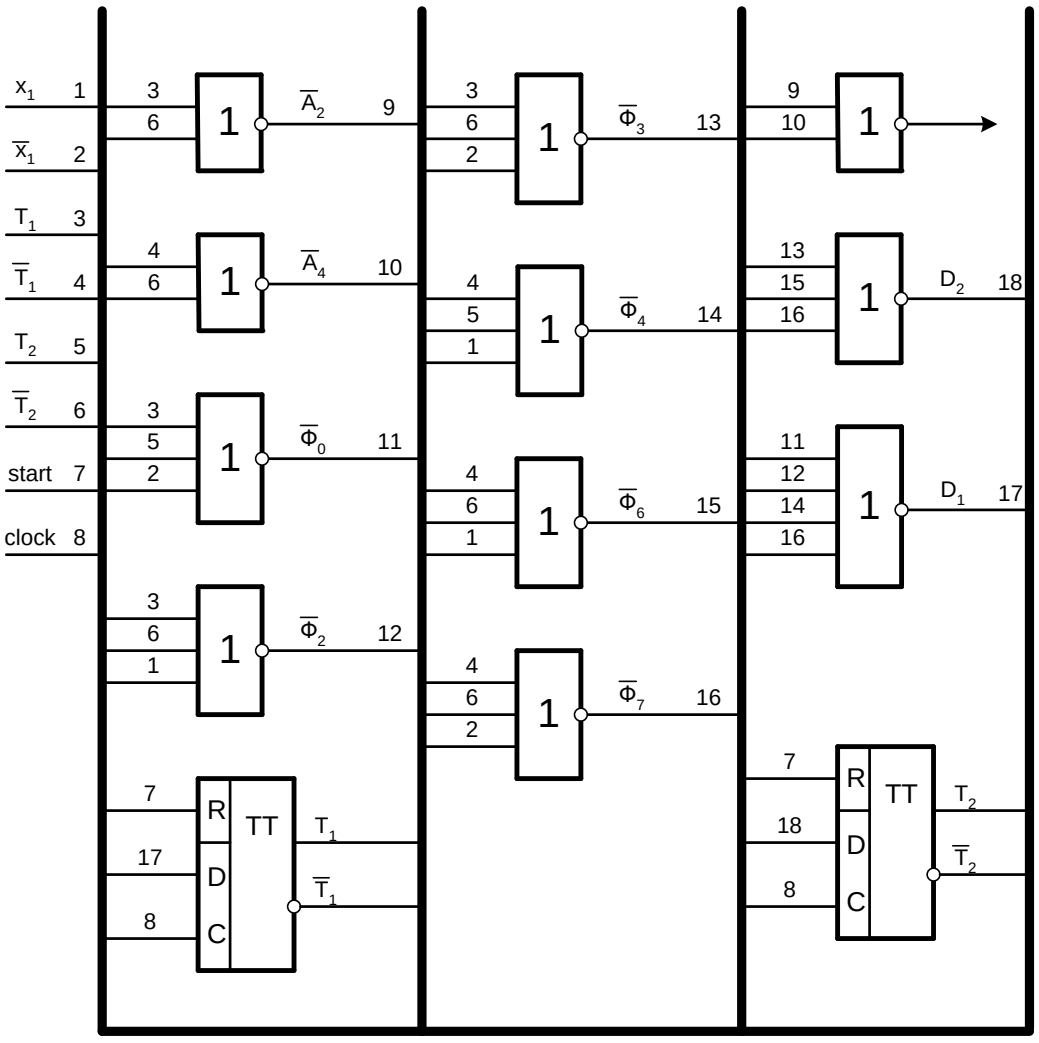


Рис. 7.25 – Логічна схема структурного автомата Мура S_{11}

Ця схема містить два рівні елементів АБО-НЕ, що реалізують систему (7.15), і два D-тригери, що утворюють пам’ять автомата.

Канонічний метод структурного синтезу є формальним інструментом, що дає змогу синтезувати функціональні схеми структурних автоматів за описом їхньої поведінки. Процес синтезу можна представити у спосіб, наведений на рис. 7.26.

Етапи 1–3 відповідають абстрактному синтезу, етапи 4–8 – структурному синтезу.

На жаль, цей метод практично не придатний для синтезу цифрових пристроїв реальної розмірності, оскільки неможливо поставити як абстрактний автомат пристрій з мільярдами станів і мільйонами вхідних і вихідних сигналів. Саме такі розмірності і мають реальні цифрові автомати. Отож, при проєктуванні реальних операційних і керуючих автоматів застосовуються інші методи. Зауважимо, що ідеї канонічного методу структурного синтезу найбільш широко застосовуються при синтезі керуючих автоматів.

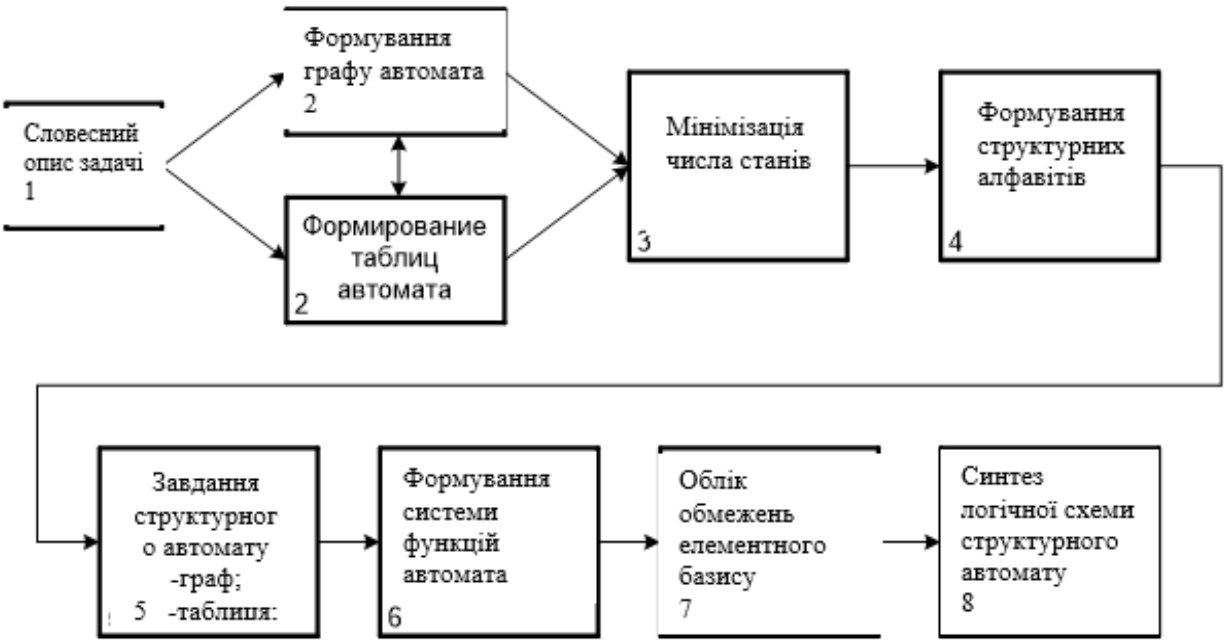


Рис. 7.26 – Процес синтезу схеми цифрового автомата

Список літератури до розділу 7

1. Глушков В. М. Синтез цифровых автоматов. Москва: Физматгиз, 1962. 476 с.

2. Баранов С. И. Синтез микропрограммных автоматов. Ленинград: Энергия, 1979. 232 с.

3. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

8 ОСНОВИ ТЕОРІЇ ОПЕРАЦІЙНИХ АВТОМАТІВ

8.1 Структура операційного автомата

Як і будь-який структурний автомат, операційний автомат (ОА) представляється у вигляді композиції комбінаційної схеми та пам'яті. Як показано на рис. 8.1, операційний автомат є частиною операційного пристрою.

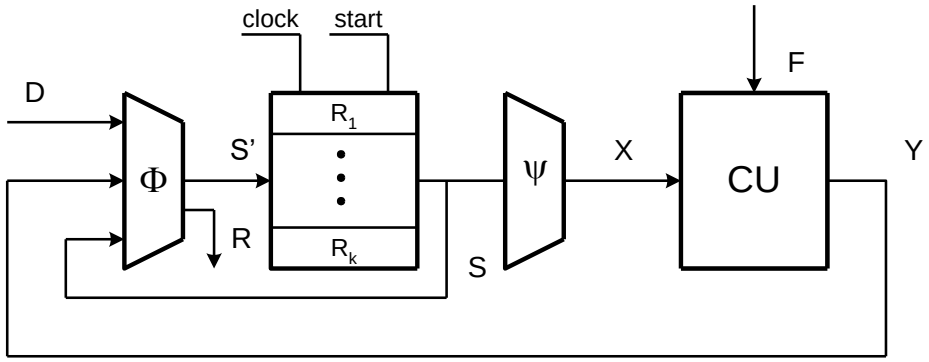


Рис. 8.1 – Структура операційного пристрою

Входами комбінаційної схеми Φ є операнди (вхідні слова) $D=\{d_1,\ \ldots,\ d_K\}$ і мікрооперації $Y=\{y_1,\ \ldots,\ y_N\}$, що формуються пристроєм управління для обробки інформації. Початкові значення вхідних слів завантажуються в пам'ять ОА, яка складається з K регістрів $R_1,\ \ldots,\ R_K$, кожен з яких має n розрядів. Виходи $S_1,\ \ldots,\ S_K$ цих регістрів формують множину проміжних результатів S, які надходять на схему Φ як сигнали зворотного зв'язку. Схема Φ формує нові значення слів $S_1^{'},...,S_K^{'}\in S^{'}$, які є функціями мікрооперацій $y_n\in Y$ і поточних значень слів $S_k\in S$. Нові значення слів $S_1^{'},...,S_K^{'}\in S^{'}$ завантажуються в пам'ять ОА за певним фронтом імпульсу синхронізації Clock. Зауважимо, що в початковий момент часу $t=0$ всі регістри пам'яті ОА обнуляються під впливом одиничного імпульсу Start. Вихідна інформація з пам'яті надходить у комбінаційну схему ψ, яка формує нові значення логічних

умов з множини $X = \{x_1, \ \ldots, \ x_L\}$. Логічні умови виступають в ролі інформаційних сигналів (прапорів), інформуючи пристрій управління про перебіг виконання алгоритму. Використовуючи цю інформацію, пристрій управління формує нові значення мікрооперацій Y, що є результатом аналізу як коду операції F, так і значень логічних умов X. Цей процес взаємодії автоматів відбувається аналогічно до того моменту, коли ОА сформує остаточні значення результатів $R = \{r_1, \ \ldots, \ r_k\}$, що відповідає кінцевому стану пристрою управління.

Число внутрішніх станів ОА можна визначити у такий спосіб:

$$M = 2^{nK}. \tag{8.1}$$

Наприклад, нехай пам'ять ОА $k = 16$ регістрів, кожен з яких має $n = 32$ розряди. Згідно з (8.1) такий автомат має $M = 2^{512}$ внутрішніх станів. Очевидно, що така величезна кількість станів виключає можливість безпосереднього застосування канонічного методу структурного синтезу для реалізації схеми ОА. З огляду на це схема ОА формується з використанням деякого заздалегідь визначеного набору елементів. Цей набір називається структурним базисом, який охоплює такі ресурси:

1. Ресурси пам'яті, що використовуються для отримання, зберігання і передачі слів інформації (операндів, проміжних і кінцевих результатів). Ці ресурси охоплюють регістри, реєстрові блоки (файли) і блоки пам'яті (постійної або оперативної). У цій книзі ми будемо розглядати тільки ОА з регістрами, а решту ресурсів пам'яті будемо вважати зовнішніми щодо ОА.

2. Функціональні ресурси, що використовуються для формування нових слів інформації. Ці ресурси будемо називати операційними елементами. Операційні елементи розподіляються на два класи. До першого класу належать так звані первинні ресурси, які представлені комбінаційними бібліотечними елементами систем автоматизованого проєктування (САПР), які використовуються для реалізації схем ОА. Такі бібліотечні елементи відповідають типовим мікрооперація м. Другий клас операційних елементів складається з нестандартних схем, які проєктуються для оптимізації певного операційного автомата.

3. Ресурси зв’язку (інтерфейси), що використовуються для передачі слів інформації між операційними елементами автомата. Ці ресурси вміщують шини та мультиплексори.

Усі ці ресурси наявні в структурній схемі, наведеній на рис. 8.1. Для формування мікрооперацій $y_n \in Y$ використовується пристрій управління (ПУ), входами якого є логічні умови з ОА. Розглянемо організацію шин ОА.

Будь-яка шина є набором провідників (проводів) і потрібна для передачі двійкової інформації. Кожен провід необхідний для передавання одного розряду (біта) інформації. Для паралельної передачі слова з K біт необхідна шина, що складається з K проводів. Домовимося позначати шину ім’ям слова інформації, що передається через неї.

Найпростішим видом шини є некерована шина, яка необхідна для виконання мікрооперації передачі даних виду $B = A$. Така шина зображена на рис. 8.2а.

Рис. 8.2 – Умовне позначення (а) й організація (б) некерованої шини

Кожен розряд цієї шини організований ідентично. При цьому провід номер k (рис. 8.2б) виконує операцію $b[k] = a[k],\ k = 1,\ \ldots,\ K$.

Керована або синхронізована шина виконує операцію передачі даних тільки за наявності спеціального сигналу управління. Ця шина виконує операцію такого виду:

$$B = \begin{cases} A, if\ y_{AB} = 1, \\ 0, if\ y_{AB} = 0, \end{cases}$$

де y_{AB} позначає сигнал управління передавання даних. Отже, має місце векторна формула $B = y_{AB}A$, яка може бути представлена в скалярній формі: $b[k] = y_{AB}a[k],\ k = 1,\ \ldots,\ K$. Умовне зображення керованої шини та організація її k-го розряду показані на рис. 8.3.

Рис. 8.3 – Умовне зображення (а) й організація k-го розряду (б) керованої шини

На практиці часто необхідно організувати передачу інформації від декількох джерел до одного приймача. З цією метою використовується мультиплексування шин, тобто мультиплексор. Наприклад, для передачі інформації від i джерел A_i до одного приймача B з використанням набору керуючих сигналів y_i $(i = 1, \ \ldots, \ I)$ необхідно виконати таку операцію:

$$B := \begin{cases} A_1, if\ y_1 = 1, \\ A_2, if\ y_2 = 1, \\ \vdots \\ A_I, if\ y_I = 1, \\ 0, if\ y_1 \vee y_2 \vee \ldots \vee y_I = 0. \end{cases}$$

Очевидно, кожен розряд такого передавача повинен реалізувати операцію $b[k] = y_1 a_1[k] \vee \ldots \vee y_I a_I[k]$. Умовне зображення й організація k-го розряду такої шини показані на рис. 8.4. Ці операції можуть бути виконані з використанням мультиплексора. Наприклад, якщо $I = 3$, то ми можемо закодувати керуючі сигнали $\langle y_1, \ y_2, \ y_3 \rangle$ з використанням двійкових кодів $z_1 z_2$: $100 \to 01, \ \ 010 \to 11, \ \ 001 \to 10, \ \ 000 \to 00$. В цьому разі ми приходимо до схеми, показаної на рис. 8.5а.

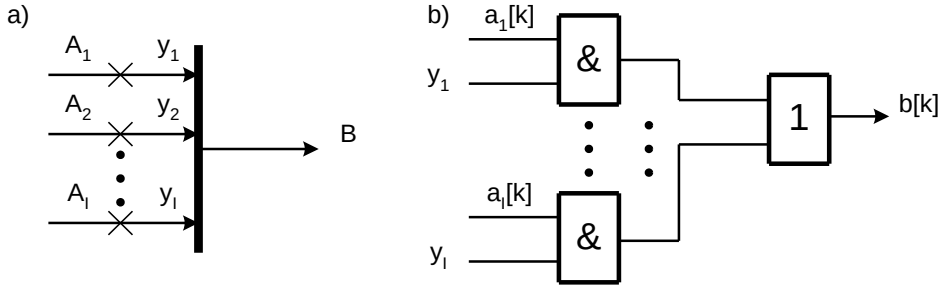


Рис. 8.4 – Умовне зображення (а) й організація (б) схеми мультиплексування даних

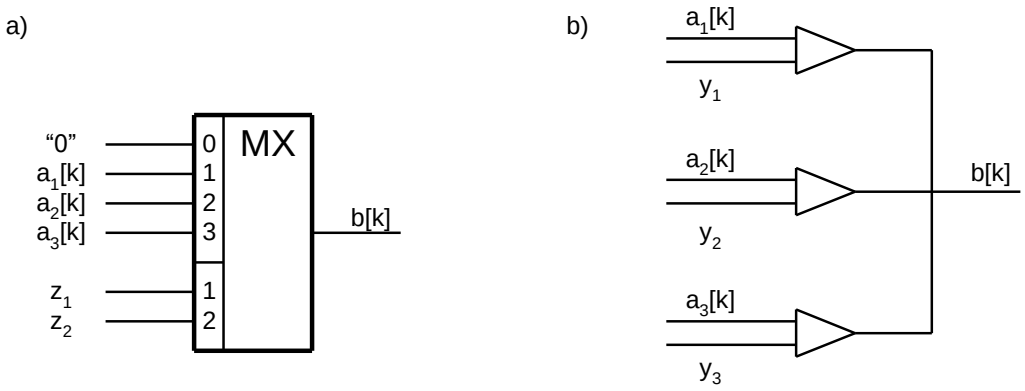


Рис. 8.5 – Реалізація схеми передачі даних з використанням мультиплексора (а) і тристабільних елементів (б)

Дуже часто для мультиплексування даних використовуються так звані тристабільні елементи, в яких здебільшого один стан відповідає логічному нулю, а другий відповідає логічній одиниці. Однак є ще й третій стан, що має великий опір, який наче вимикає вихід джерела від шини. Для установки елемента в третій стан використовується спеціальний керуючий сигнал. Наприклад, сигнал ініціалізації передачі даних можна використати для установки виходу в третій стан (рис. 8.5б). Якщо $y_i = 1$, то слово A_i передається на вихід B $(i = 1,\ \ 2,\ \ 3)$. Якщо $y_i = 0$, то відповідний вихід джерела буде встановлений у третій стан, тобто передачі даних з цього джерела не буде. Домовимося називати всі подібні пристрої мультиплексорами.

Між комбінаційною частиною автомата і його пам’яттю існує зворотний зв’язок. Це означає, що для стійкої роботи ОА його пам’ять повинна бути синтезована на тригерах, що спрацьовують по фронту сигналу синхронізації. В іншому разі існує небезпека переходу автомата в неправильний стан у тригерах пам’яті. Зауважимо, що в разі ОА під неправильним станом розуміється неправильне значення проміжних або кінцевих значень результатів виконання операції.

У цій книзі ми розглядаємо методи синтезу ОА, запропоновані професорами С. А. Майоровим і Г. І. Новіковим. Основою для синтезу

схеми операційного автомата є так звана функціональна мікропрограма. Її західним аналогом є так звана алгоритмічна машина станів (algorithmic state machines). За обох підходів закон поведінки автомата задається з використанням логічних умов і типових мікрооперацій.

8.2 Функціональні мікропрограми

У процесі розробки засобів обчислювальної техніки розробники сформували низку типових рішень для проєктування схем ОА, так званих операційних елементів. Ці елементи проєктуються тільки раз, їхні схеми оптимізуються для даного елементного базису (за витратами апаратури або за швидкодією) і поміщаються в бібліотеки САПР. За необхідності використання операційного елемента достатньо задати його розрядність, і оптимальна схема формується за допомогою САПР. Кожен операційний елемент виконує певну операцію елементарної обробки двійкової інформації, яку ми домовимося надалі називати типовою мікрооперацією (МО). Існують такі класи типових мікрооперацій:

1. Мікрооперації ініціалізації.

Подібні МО використовуються для запису деяких констант у регістри пам’яті ОА. Такими МО є, наприклад, $A := 01$, $B := 32$. За допомогою МО ініціалізації в пам’ять ОА заносяться початкові значення операндів.

2. Мікрооперації передачі даних.

Подібні МО використовуються для передачі слів інформації між регістрами пам’яті, наприклад, $A := B$.

3. Мікрооперації інвертування слова (порозрядне заперечення).

Виконання цієї МО приводить до заміни слова інформації, що опрацьовується, його порозрядною інверсією. Якщо результат операції повинен поміщатися в той самий регістр, то кінцеве значення слова надходить через ланцюг зворотного зв’язку. Наприклад, для виконання

операції інвертування слова A$(A := \neg A)$ використовується схема, показана на рис. 8.6. При надходженні сигналу $y_1 = 1$ багаторозрядний інвертор виконує порозрядну інверсію слова A і завантажує результат у регістр по фронту сигналу Clock.

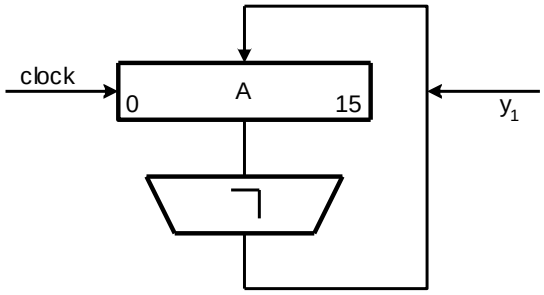


Рис. 8.6 – Виконання мікрооперації порозрядної інверсії слова

Знак: =, що використовується в раніше розглянутих мікрооперaціях, означає, що результат виконання такої мікрооперації заноситься до відповідного регістру за певним фронтом сигналу синхронізації Clock. При цьому ліва частина виразу $A := B$, наприклад, представляє вміст регістра A в момент часу t, тобто старий вміст. Права частина цього виразу означає, що після певного фронту імпульсу синхронізації Clock у момент часу $t+1$ $(t = 0,\ \ 1,\ \ \ldots)$ у регістрі A буде знаходиться слово B. В іншому прикладі при виконанні інверсії вмісту регістра A за $A^t = 0101$ отримаємо вміст цього регістра, що дорівнює $A^{t+1} = 1010$.

4. Мікрооперації зсуву.

Ці мікрооперації використовуються для зміни позиції деякого біта слова щодо його попередньої позиції шляхом переміщення кожного біта слова на n позицій праворуч або ліворуч. Наприклад, такі МО використовуються при виконанні деяких арифметичних операцій, при послідовно-паралельній обробці інформації тощо. При цьому розрізняють логічне й арифметичне зрушення інформації. Домовимося далі називати логічне зрушення просто зрушенням.

У разі зсуву на n розрядів праворуч $(A := R_n(A))$ крайні ліві n розряди слова будуть втрачені, а праві крайні n розряди обнуляться. Наприклад, $R_2(110110) = 001101$, $R_3(110110) = 000110$.

У разі зсуву на n розрядів ліворуч $(A := L_n(A))$ праві крайні n розряди слова будуть втрачені, крайні ліві n розряди обнуляться. Прикладами подібних МО будуть $L_1(110110) = 101100$, $L_2(110110) = 011000$.

Зрушення може виконуватися циклічно, що приводить врешті-решт до відновлення вихідного слова. Такі мікрооперації необхідні для запобігання втрати інформації. Наприклад, при множенні двійкових чисел аналізується кожен розряд множника. Щоб аналізувати завжди тільки один розряд слова, виконують його зрушення на один розряд. Якщо звільнені розряди заповнюються нулями, то множник буде втрачено. За циклічного зсуву після виконання всіх тактів множення множник має початкове значення і може бути далі використаний у програмі. За циклічного зсуву на n розрядів праворуч $(A := CR_n(A))$ крайні ліві n розряди слова замінюються значеннями його крайніх правих n розрядів. Наприклад, $CR_2(110110) = 101101$, тобто 4 старші розряди слова (1101) зсуваються на 2 позиції праворуч, а два молодші розряди (10) стають на місце двох старших розрядів.

У разі циклічного зсуву на n розрядів слова ліворуч $(A := CL_n(A))$ молодші n розряди слова замінюються на n старших розрядів вихідного слова. Наприклад, $CL_2(110110) = 011011$, тобто 4 молодші розряди слова (0110) зсуваються на 2 позиції ліворуч, а 2 старші розряди (11) замінюють 2 молодші розряди вихідного слова.

При виконанні операцій зсуву іноді необхідно дублювати деяку інформацію в декількох розрядах слова. Переважно, дублюється знак слова. Цьому відповідає МО арифметичного зсуву $(A := AR_n(A))$, що зводиться до копіювання старшого розряду слова (здебільшого, це знак

слова) в n старших розрядах вихідного слова. При цьому n молодших розрядів слова будуть втрачені. Прикладами таких мікрооперацій є $AR_3(100100)=111100,\ AR_2(100100)=111001.$

5. Мікрооперації рахунку.

При виконанні цих МО вміст слова змінюється на одиницю. При цьому операція складання з одиницею, $A:=A+1$, називається інкрементом, а операція віднімання одиниці, $A:=A-1$, називається декрементом слова A.

6. Логічні мікрооперації.

Ці МО є порозрядними і визначають, здебільшого, результати операцій кон'юнкції, диз'юнкції і виключення АБО ($\wedge,\vee,\oplus.$). Наприклад, при виконанні МО $C:=A\vee B$ отримаємо $C=0111$ за таких значень операндів: $A=0101$ і $B=0110$. Результати виконання логічних мікрооперацій для цих значень операндів показані на рис. 8.7а. Наприклад, порозрядна кон'юнкція приводить до $C=0100$, а порозрядне виконання операції EXOR приводить до $C=0011$. Частина ОА для виконання МО $S:=A\vee B$ показана на рис. 8.7б.

Для виконання МО $S:=A\vee B$ використовується керуючий сигнал y_2. Необхідно зазначити, що ця мікрооперація виконується постійно, але її результат записується в регістр S тільки за рівності $y_2=1$. Для виконання будь-якої логічної МО використовується подібна структура ОА, але комбінаційна схема повинна бути позначена відповідним символом (наприклад, $R_2, L_1, +1$ тощо).

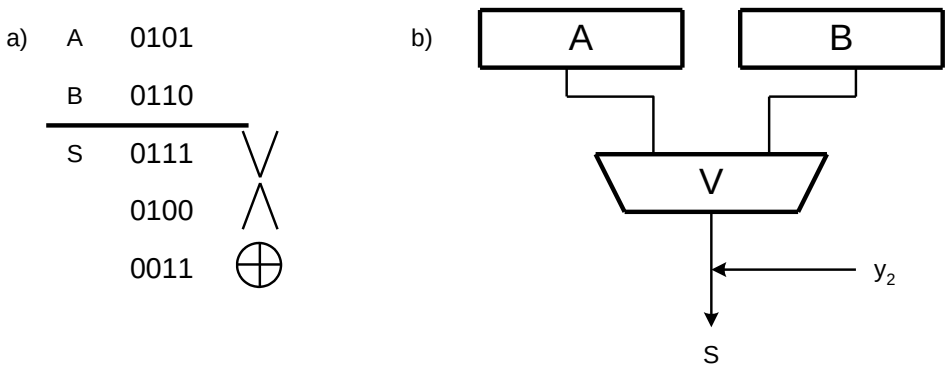


Рис. 8.7 – Виконання логічних мікрооперацій (а) і частина ОА (б)

7. Мікрооперації складання.

Ця МО використовується для знаходження результату підсумовування операндів з двох джерел; цей результат потім записується в регістр-приймач суми. При цьому можуть підсумовуватися операнди з джерел, які відмінні від приймача (наприклад, операція $S := A + B$). Якщо одним з операндів є сам приймач (наприклад, $S := S + A$), то подібна операція виконується на суматорі, що має зворотний зв'язок. Подібний пристрій використовується в усіх ЕОМ; такий акумулятор називається накопичувальним суматором або акумулятором. На рис. 8.8 показаний фрагмент схеми ОА, який використовується для виконання МО $y_3 \# S := A + B$. Зауважимо, що правильніше називати розглянуту МО підсумовуванням, оскільки операція додавання набагато складніша і виконується за кілька тактів.

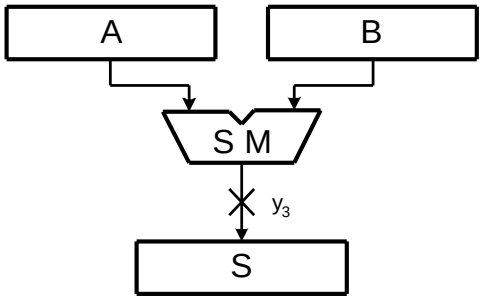


Рис. 8.8 – Фрагмент схеми ОА для виконання додавання

Здебільшого мікрооперація вирахування не належить до типових МО. Це зумовлене тим, що операція віднімання ($S := A - B$) може бути виконана як складання з протилежним знаком ($S := A + (-B)$). Крім того, до цього класу типових МО належить операція додавання трьох операндів, де як третій операнд завжди виступає одиниця, тобто $S := A + B + 1$. Подібні МО надзвичайно корисні у процесі організації багаторозрядних суматорів або для виконання операцій над числами, представленими в додатковому коді.

8. Комбіновані мікрооперації.

Комбіновані мікрооперації використовуються для збільшення продуктивності в спеціалізованих пристроях, що розробляються для деяких спеціальних цілей. Такі мікрооперації складаються з декількох типових мікрооперацій, що виконуються в одному такті роботи пристрою. Наприклад, комбінована МО $S := R_1(A+B)+C$ складається з трьох типових мікрооперацій (дві МО складання і зрушення праворуч). Зазначимо, що в загальному випадку комбіновані МО не належать до класу типових МО.

Домовимося називати мікрооперaціями як елементарні операції обробки двійкової інформації, так і сигнали (змінні), які ініціалізують виконання цих операцій.

Для передачі пристроєм управління інформації про стан процесу обробки інформації, про що вже йшлося, використовуються спеціальні логічні умови, що утворюють множину $X = \{x_1, \ \ldots, \ x_L\}$. У загальному випадку кожна логічна умова (ЛУ) представляється деякою булевої функцією $\psi_l\{S_1, \ \ldots, \ S_I\}$, де слова $S_1, \ \ldots, \ S_I$ є операндами операцій, що виконуються. Для обчислення значень ЛУ в ОА використовуються спеціальні комбінаційні схеми. Наприклад, нехай деяка логічна умова $x_1 = 1$, якщо $A[0] = 0$. Для її обчислення використовується комбінаційна схема (рис. 8.9а), що відповідає булевій функції $x_1 = \overline{A[0]}$. Нехай деякий ЛУ $x_2 = 1$, якщо значення розрядів $A[0]$ і $B[0]$ дорівнюють одиницям (обидва числа будуть при цьому негативними). Очевидно, що такій ЛУ відповідає функція $x_2 = \overline{A[0] \oplus B[0]}$, що веде до комбінаційної схеми, показаній на рис. 8.9б. Далі, нехай $x_3 = 1$, якщо всі розряди слова A дорівнюють нулям. Цій умові відповідає булева функція $x_3 = \overline{A[1] \vee \ldots \vee A[k]}$, де k – довжина слова A. Відповідна комбінаційна схема показана на рис. 8.9в.

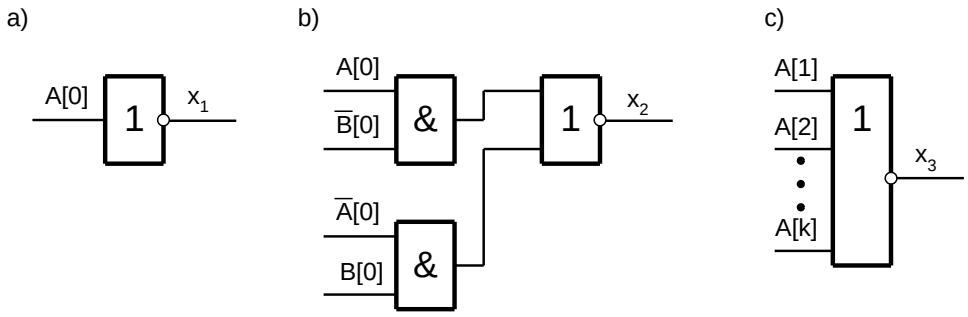


Рис. 8.9 – Приклади схем для обчислення логічних умов

Для проєктування цифрового пристрою необхідно мати задання закону його функціонування в часі. Така специфікація необхідна як для операційного, так і для керуючого автоматів операційного пристрою. У нашій книзі для специфікації поведінки ОА в часі використовується мова граф-схем алгоритму (ДСА). Граф-схема алгоритму – це орієнтований пов'язаний граф, що включає кінцеве число вершин. В ДСА входять вершини чотирьох типів: початкова, кінцева, операційна та умовна (рис. 8.10).

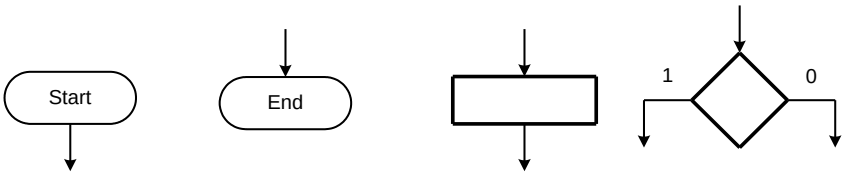


Рис. 8.10 – Типи вершин граф-схеми алгоритму

Початкова вершина не має входу, а тільки вихід, що відповідає початку виконання алгоритму. Кінцева вершина має тільки вхід і відповідає закінченню виконання алгоритму. Операторна вершина має один вхід і один вихід. Умовна вершина має два виходи, перший з яких позначений символом 1, а другий – символом 0. Один із виходів умовної вершини може бути з'єднаний з її входом, така вершина забезпечує очікування закінчення деякого процесу і називається вершиною очікування. Для операторної вершини таке з'єднання виходу і входу не допускається, оскільки воно призводить до зависання пристрою. Будь-яка ДСА задовольняє такі умови:

1. ДСА містить кінцеве число операторних і умовних вершин, одну початкову і одну кінцеву вершини.

2. Входи і виходи вершин ГСА з'єднані дугами, спрямованими від входу до виходу.

3. Кожен вихід з'єднується тільки з одним входом. В іншому разі виникає невизначеність з переходами автомата.

4. Вхід кожної вершини повинен бути з'єднаний принаймні з одним виходом. В іншому разі існують ситуації, коли функціонування ОА не може бути результативно завершено.

5. Існує принаймні один шлях з кожної вершини ГСА в її кінцеву вершину. В іншому разі існують ситуації, коли функціонування ОА не може бути результативно завершено.

6. Кожна умовна вершина містить одну логічну умову. Позначений символом 1 вихід умовної вершини відповідає істинності умови, що перевіряється. Позначений символом 0 вихід умовної вершини відповідає хибності умови, що перевіряється.

7. Кожна операторна вершина містить набір мікрооперацій, які виконуються одночасно. Отже, кожна операторна вершина відповідає одному такту автоматного часу. Існують порожні операторні вершини, вони використовуються для деяких спеціальних цілей.

Якщо всі мікрооперації, записані в операційних вершинах деякої ДСА, є типовими мікрооперaціями, то подібна ДСА називається функціональною граф-схемою алгоритму. Розглянемо приклад формування функціональної ДСА.

Приклад 8.1. Необхідно побудувати функціональну ДСА Г1 для обчислення функції $S_1 = 5(A + B) \vee \frac{(B \oplus C)}{4} + A \oplus B \oplus C.$

Процес розробки функціональної ДСА багато в чому подібний до процесу написання програми. Отже, це може розглядатися як певне мистецтво. Для успішного вирішення подібних завдань необхідні

тренування, які надають потрібні навички. Функціональна ДСА Γ_1 приведена на рис. 8.11.

Під час формування функціональних ДСА (як і під час написання програм) можуть висуватися певні умови. Наприклад, у процесі формування ДСА Γ_1 ми намагалися отримати мінімальне число операторних вершин без введення додаткових слів, відмінних від A, B, C і S. Це означає, що ми намагалися мінімізувати як час виконання алгоритму, так і розмір пам'яті ОА. Надалі буде видно, що кожне слово в ДСА відповідає одному регістру, а кожна операторна вершина відповідає одному циклу часу роботи ОА.

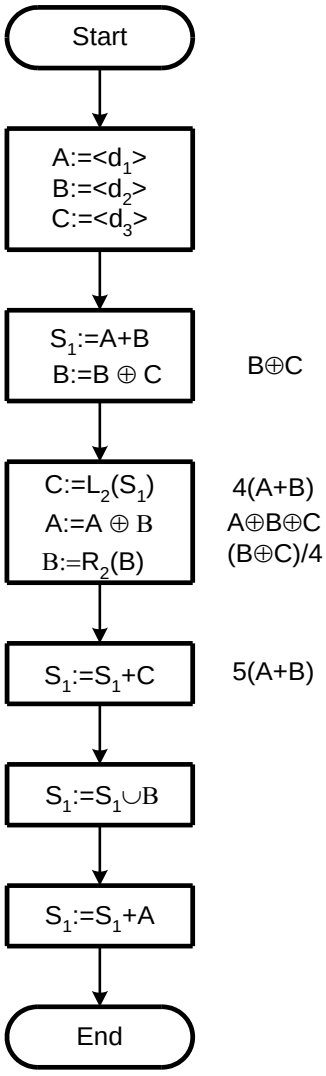


Рис. 8.11 – Функціональна граф-схема алгоритму Γ_1

Друга вершина ГСА на рис. 8.11 відповідає операції запису початкових значень операндів A, B, C. При цьому початкові значення

утворюють множину даних D. Отже, друга вершина – це вершина ініціалізації. Для виконання операції $S_1 := 5S_1$ необхідно представити її у вигляді $S_1 := 4S_1 + S_1 = L_2(S_1) + S_1$. Це уявлення дає змогу обійти відсутність операцій множення на константу серед типових мікрооперацій. Зазначимо, що мікрооперація $L_n(A)$ відповідає множенню числа A на константу 2^n. Аналогічно, зрушення на n розрядів праворуч аналогічне поділу операнда, що зрушується, на константу 2^n.

З огляду на це операція зсуву $B/4$ виконується як типова мікрооперація $R_2(B)$.

Приклад 8.2. Розробити функціональну ДСА Γ_2 для виконання операції $S_2 = \begin{cases} \frac{3(A+B)}{4} \vee \frac{\overline{C \oplus B}}{2}, & \text{якщо } A > B, \\ 5(A+B+C)3(C \oplus B) & \text{в іншому разі.} \end{cases}$

Отримана функціональна ДСА Γ_2 приведена на рис. 8.12. Як і в попередньому прикладі, ми намагалися мінімізувати число операторних вершин без введення додаткових слів. Оскільки у верхній і нижній частинах виразу S_2 можна виділити операції $A+B$ і $C \oplus B$, то їх доцільно виконати перед перевіркою умови $A > B$.

Завдання побудови функціональних ДСА ставиться для проблем синтезу. Для вирішення цієї проблеми розробник повинен знати:

1. Склад набору типових мікрооперацій, які можуть бути різними для різних бібліотек САПР.

2. Критерій ефективності розробленої ДСА. У наших прикладах використаний критерій мінімуму числа регістрів, тобто мінімум витрат апаратури. Можна використовувати критерій максимальної продуктивності, що приводить до збільшення витрат апаратури.

3. Однозначне формулювання задачі, що розв'язується.

Домовимося називати функціональні ДСА функціональними мікропрограмами, як це прийнято у С. А. Майорова і Г. І. Новікова. У

західній літературі функціональні мікропрограми відповідають так званим розкладам без обмежень. Існує теорія розкладів, яка допомагає створювати оптимальні прошивки, однак ці питання виходять за рамки нашої книги.

Необхідно зазначити три особливості, що відрізняють функціональні мікропрограми від розкладів:

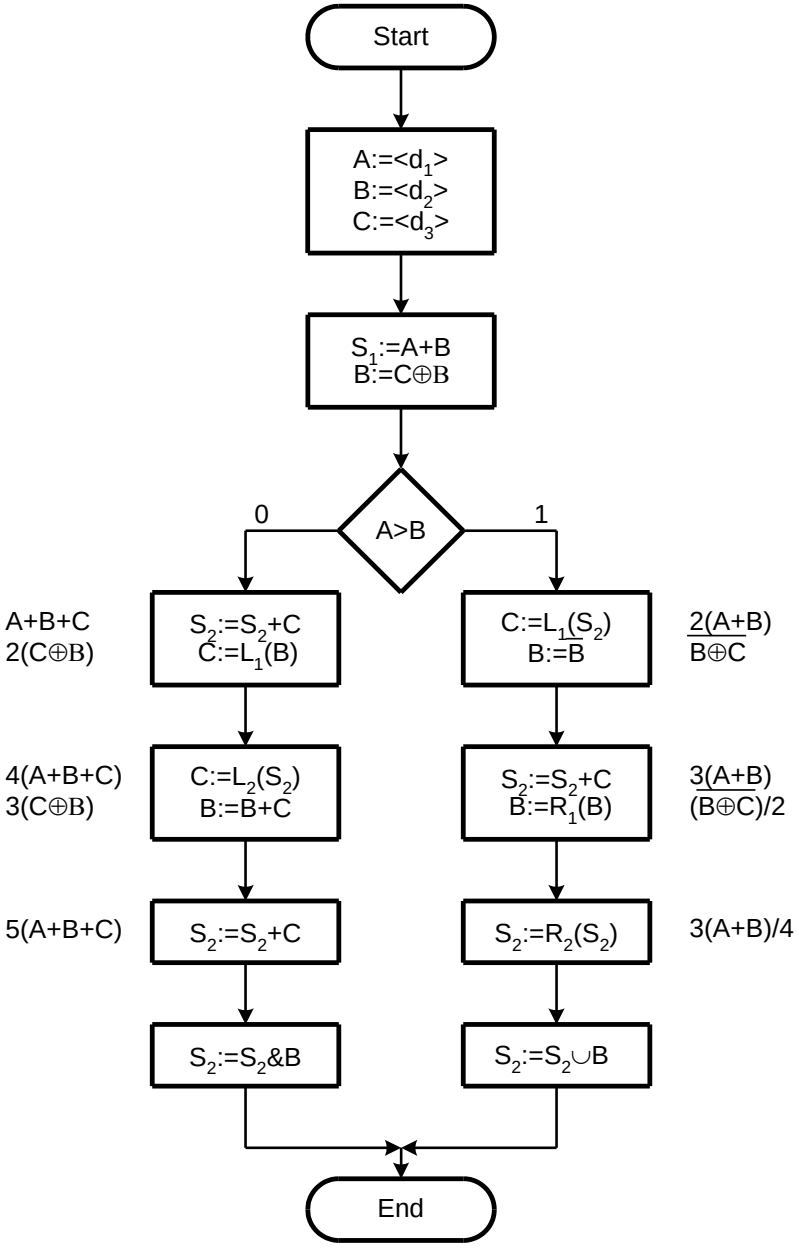


Рис. 8.12 – Функціональна ДСА Γ_2

– у функціональної мікропрограми є вершина «Start», а розклад починається з вершини ініціалізації;

– виходи вершин, пов’язаних з блоком ініціалізації розкладу, пов’язані з входом вершини «End» відповідної функціональної прошивки;

– блоки станів і умовні блоки розкладу відповідають операційним і умовним вершинам функціональної мікропрограми.

Отже, перехід між цими формами уявлення ОА може бути виконаний без труднощів. Наприклад, достатньо замінити блоки розкладу відповідними вершинами і ввести початкову і кінцеву вершини для переходу від розкладу до відповідної функціональної мікропрограми.

8.3 Базові структури операційних автоматів

Очевидно, що для виконання одного і того самого алгоритму обробки інформації можуть бути використані різні схеми операційних автоматів. При цьому розробник повинен мати деякі загальноприйняті критерії для оцінки якості окремих рішень. В цьому разі можна вибрати найкращі з них, оптимізувавши деякі критерії. Існують два універсальні критерії якості рішення – швидкодія ОА та апаратурні витрати (тобто ціна схеми ОА).

Для обчислення швидкодії схеми ОА необхідно знайти добуток часу циклу автомата на середню кількість циклів, необхідних для виконання цього алгоритму. У західній літературі ця оцінка також називається латентністю (latency time).

Час циклу автомата. Цей параметр залежить від структури автомата (від його організації). Він визначається шляхом підсумовування максимальних затримок (часів поширення сигналу) різних операційних елементів, що входять у схему ОА. Нагадаємо, що до таких елементів належать комбінаційні схеми, регістри і шини.

Середнє число циклів. Для визначення цього параметра необхідно виконати моделювання функціональної мікропрограми, змінюючи значення операндів за допомогою генераторів випадкових чисел. Також

для знаходження цього параметра можуть бути використані ланцюги Маркова.

Швидкодія. Цей параметр визначається середнім часом виконання алгоритму. Для обчислення швидкодії даного автомата необхідно перемножити час циклу ОА на середнє число циклів, необхідних для виконання алгоритму.

Апаратурні витрати. Цей параметр визначається сумарною вартістю всіх операційних елементів, що входять в схему ОА. При цьому як оцінка може виступати ціна схеми в грошових одиницях, а можна використовувати, наприклад, площу кристала, необхідну для реалізації схеми ОА. Очевидно, апаратурні витрати можуть визначатися просто сумарним числом операційних елементів у схемі ОА. Існує певна залежність між швидкодією ОА і апаратурними витратами, яка формулюється у такий спосіб: чим вище швидкодія, тим більше апаратурні витрати. Іноді існують винятки з цього правила, зумовлені грамотним використанням особливостей елементного базису.

Існують деякі додаткові параметри, які використовуються для порівняння різних схем ОА. До них належать продуктивність, регулярність і універсальність.

Продуктивність. Цей параметр визначається як середнє число мікрооперацій, що виконуються за один такт роботи ОА. Якщо пам’ять ОА охоплює K регістрів, то продуктивність знаходиться в межах від одиниці до K. Чим вище продуктивність ОА, тим складніше (і дорожче) його схема.

Регулярність. Структура об’єкта є регулярною, якщо об’єкт складається з множини ідентичних елементів, з’єднаних за допомогою ідентичних зв’язків. Серед пристроїв ЕОМ найбільшу регулярність має пам’ять, оскільки вона складається з ідентичних осередків. Очевидно, чим регулярнішою є структура об’єкта, тим простіше його виробництво, налагодження та подальший ремонт. Кажуть, що такий об’єкт більш технологічний.

Універсальність. Ця характеристика ОА визначає потенційну здатність схеми до реалізації довільного алгоритму обчислень без зміни числа і складу операційних елементів і (або) зв'язків між ними. Чим вище універсальність, тим нижче рівень спеціалізації пристрою. Наприклад, мікропроцесор може бути налаштований на виконання будь-яких обчислень за допомогою зміни програми. Однак, чим більше універсальність пристрою, тим менше його продуктивність, порівняно зі спеціалізованим пристроєм, призначеним тільки для реалізації певної функції (замовний БІС, наприклад).

Нехай пам'ять ОА складається з K регістрів R_1, …, R_K, позначимо їхні виходи відповідно символам S_1, …, S_K. Нехай кожен з цих регістрів складається з n тригерів (тобто довжина або розрядність регістра дорівнює n). В цьому разі ОА може бути реалізована як канонічний операційний автомат, структурна схема якого показана на рис. 8.13.

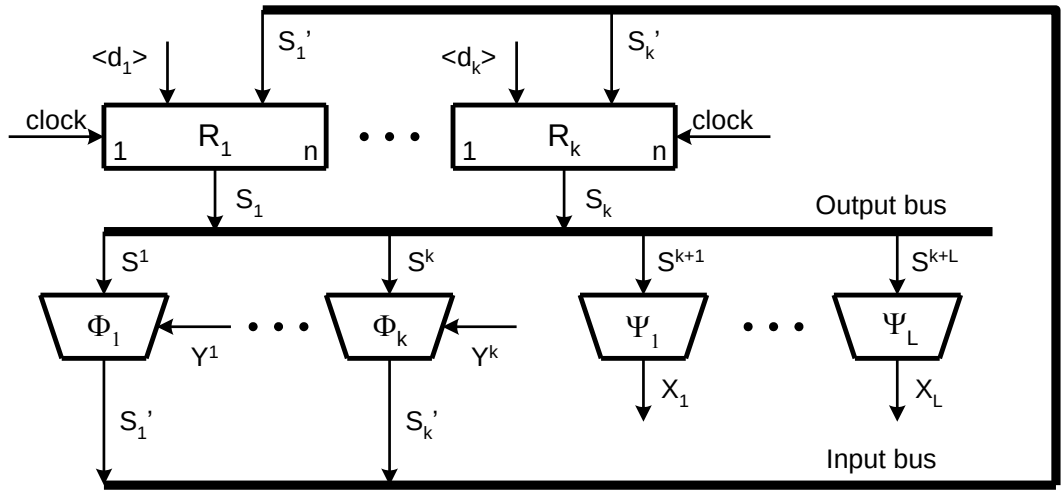


Рис. 8.13 – Структурна схема канонічного операційного автомата

Виходи S_1, …, S_K регістрів пам'яті ОА пов'язані з вихідною шиною, яка містить сумарно nK провідників (розрядів). Ця шина використовується для передачі інформації до комбінаційної частини ОА. Комбінаційна схема Φ_k формує нове значення слова S_k, використовуючи як

операнди слова з множини $S_k \subseteq S = \{S_1, \ \ldots, \ S_K\}$. Нове значення слова позначено символом $S_k^{'}$. Формування значення слова $S_k^{'}$ проводиться під управлінням мікрооперацій, що утворюють множину Y^k. Зазначимо, що виконується умова $Y^k \subseteq Y$, і при цьому множина $Y^k \ \ (k = 1, \ \ldots, \ K)$ є одним класом розбиття множини мікрооперацій Y. Комбінаційна схема ψ_l обчислює нове значення логічної умови $x_l \in X$ ($l = 1, \ \ldots, \ L$), для чого використовується підмножина S^{K+l} вихідної множини слів S. Під час появи фронту імпульсу синхронізації Clock нові значення слів $S_1^{'},...,S_k^{'}$ завантажуються у відповідні регістри через вхідну шину, яка має nK біт (проводів). Початкове значення $\langle d_k \rangle$ слова $S_k \ \ (k = 1, \ \ldots, \ K)$ записується в регістр R_k на першому кроці виконання деякого алгоритму.

Основним недоліком канонічного ОА є надмірність апаратури, що використовується, оскільки однакові операційні елементи (суматори, зсуви тощо) використовуються в різних підсистемах $\Phi_k \ \ (k = 1, \ \ldots, \ K)$, причому в кожній підсхемі може бути кілька однакових операційних елементів. Однак ця надмірність має і позитивну сторону, оскільки вона приводить до мінімального часу такту і максимальної продуктивності канонічної схеми ОА. Зауважимо, що продуктивність канонічного ОА може досягати до K мікрооперацій, а отже, слова з усіх регістрів ОА можуть оброблятися одночасно.

Апаратурні витрати можуть бути зменшені без зменшення продуктивності ОА. Для цього необхідно, щоб всередині кожної підсхеми використовувався не більше, ніж один операційний елемент кожного типу. Це означає, що в кожній схемі Φ_k може бути тільки один акумулятор, один зсувник на один розряд праворуч і так далі. При цьому в різних схемах Φ_k можуть використовуватися однакові операційні елементи. Такий підхід веде до автомата типу I, який ми будемо називати I-ОА (рис. 8.14).

У разі І-ОА для обчислення нового значення кожного слова інформації використовується індивідуальна комбінаційна схема. Назва автомата походить від першої літери слова «individual». Такі ОА також називають ОА із закріпленими мікрооперaціями. При цьому всередині кожної комбінаційної підсхеми Φ_k $(k=1, \ \ldots, \ K)$ необхідно поєднувати різні джерела з одним операційним елементом. Для цього використовуються $MX_1, \ \ldots, \ MX_K$. Кожен мультиплексор MX_k управляється мікрооперацією з множини Y^k; при цьому формується набір слів S_M^k, що використовуються як операнди схеми Φ_k.

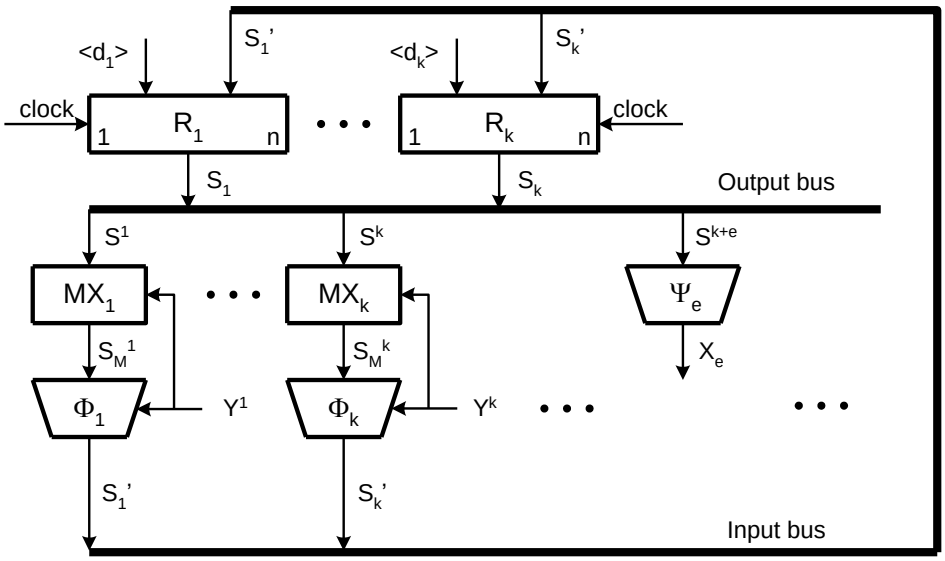


Рис. 8.14 – Структурна схема І-ОА

В іншому структура І-ОА збігається зі структурою канонічного ОА. Очевидно, перехід $S^k \to S_M^k$ вимагає деякого часу, що призводить до збільшення часу такту і зменшення швидкодії І-ОА щодо еквівалентного канонічного ОА. Тут під автоматною еквівалентністю розуміється виконання обома автоматами однакового алгоритму обчислень.

Очевидно, що І-ОА відповідає розкладу з мінімально можливим числом циклів і мінімальними апаратурними витратами, достатніми для

забезпечення заданої продуктивності. У загальному випадку швидкодія І-ОА менше, ніж у еквівалентного канонічного ОА.

Кількість типів операційних елементів можна зменшити до мінімуму (тобто, до одного), якщо кожен з них можна використовувати для визначення нових значень всіх слів прошивки. Це означає, що результат виконання мікрооперації на будь-якому операційному елементі може бути використаний будь-яким регістром пам'яті ОА. В цьому разі комбінаційна частина ОА включатиме лише один акумулятор, один зсувник і так далі. Такий підхід приводить до М-ОА, що наведено на рис. 8.15.

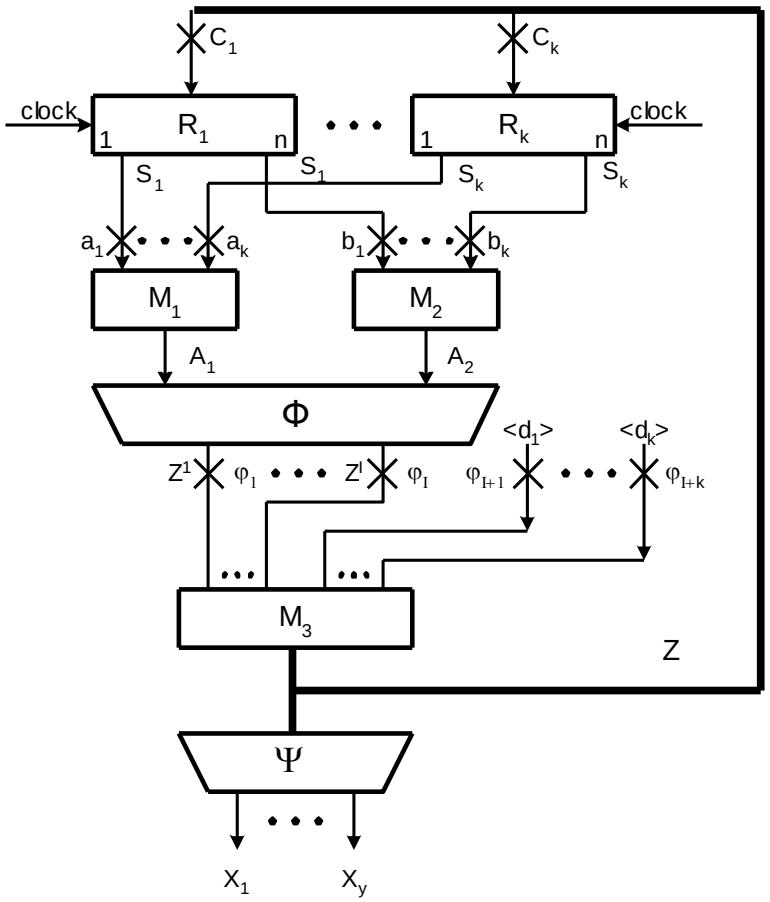


Рис. 8.15 – Структурна схема М-ОА

Буква М у назві типу ОА походить від слова «mutual». Також ці ОА називають автоматами з узагальненими мікрооперціями. Дані (слова) з

регістра R_k надходять у комбінаційну схему Ф або через шину M_1 (мультиплексор), використовуючи внутрішній керуючий сигнал (оператор) a_k, або через шину (мультиплексор) M_2, використовуючи оператор b_k. Інформація в схему Ф надходить з виходів цих мультиплексорів, позначених на структурній схемі символами A_1 і A_2. Далі, ця інформація обробляється схемою Ф і формуються нові значення вихідних слів Z^1, …, Z^I (наприклад, $Z^2 := A_1 \oplus A_2$). Результати обробки інформації надходять на шину Z через мультиплексор M_3, для чого використовуються оператори φ_i $(i = 1, \ldots, I)$. Початкові значення слів $\langle d_1 \rangle$, …, $\langle d_K \rangle$ також надходять через шину Z; для цього використовують відповідно оператори φ_{I+1}, …, φ_{I+K}. Нові значення слів завантажуються в регістри R_1, …, R_K за сигналами c_1, …, c_K відповідно. Нові значення прапорів (логічних умов) x_1, …, x_J формуються комбінаційною схемою ψ. Зауважимо, що $J \le L$, оскільки однакові перевірки для різних слів виконуються однією підсхемою. Наприклад, логічні умови $x_1 \# R_1[0] = 1$ і $x_2 \# R_2[0] = 1$ замінюються однією умовою $x_j \# Z[0] = 1$. В М-ОА кожна з шин A_1, A_2 і Z включає тільки n провідників, тобто вимагає в K разів менше обладнання, ніж шини еквівалентного І-ОА.

Як показує аналіз структурної схеми М-ОА, даний автомат може виконувати в кожному такті свого функціонування тільки одну МО вихідної прошивки. Отже, М-ОА має найменшу продуктивність, порівняно з раніше розглянутими типами ОА. Однак цей його недолік компенсується високою регулярністю й універсальністю, а також мінімумом витрат апаратури. Наявність мультиплексора M_3 призводить до збільшення часу такту, порівняно з еквівалентним І-ОА. Крім того, процес проєктування

схеми М-ОА ускладнюється, тому що вихідний розклад без обмежень має бути змінено. Ця зміна зумовлена тим, що в одному такті роботи ОА тепер виконується не більше однієї МО. Отже, вихідна функціональна мікропрограма повинна бути перетворена певним способом. Основним завданням цього перетворення є отримання розкладу з мінімальним числом тактів при виконанні обмеження на число, що виконується в одному такті МО. Природно, що апаратурні витрати повинні зберігатися мінімальними. Перетворена мікропрограма потім використовується для синтезу пристрою управління.

Існують операційні автомати, які мають середні показники швидкодії і витрат апаратури щодо таких крайніх точок простору можливих рішень, як М-ОА і І-ОА. Такі ОА отримали назву ІМ-ОА. Існує два основні типи ІМ-ОА, які відрізняються методом організації комбінаційної частини (КЧ).

В ІМ-ОА з паралельною комбінаційною частиною $(IM_p - OA)$ є комбінаційна схема Φ_1, яка виконує мікрооперації з двома операндами (додавання, кон'юнкція і так далі), і комбінаційна схема Φ_2, яка служить для виконання мікрооперацій з одним операндом (типу інверсія або зсув). Дві окремі комбінаційні схеми ψ_1 і ψ_2 використовуються для визначення значень логічних умов (рис. 8.16).

Очевидно, що $IM_p - OA$ можна розглядати як композицію двох автоматів типу М. Ці автомати мають загальну пам'ять, що складається з регістрів R_1, …, R_K. Перший автомат складається з мультиплексорів M_1, M_2, M_4 і комбінаційних схем Φ_1 (обробка слів) і ψ_1 (обчислення нових значень логічних умов). Другий операційний автомат містить мультиплексори M_3, M_5 і комбінаційні схеми Φ_2 і ψ_2. На схемі наявні оператори a, b, c, d, e, φ, ψ, призначення яких зрозуміле по аналогії з

М-ОА. Використовуючи таку організацію, $IM_p - OA$ може виконувати в одному такті до двох мікрооперацій, результати виконання яких паралельно з'являються на шинах Z_1 і Z_2. Отже, відбувається подвоєння продуктивності, порівняно з еквівалентним автоматом типу М-ОА. Однак це зумовлено ростом витрат апаратури через використання двох додаткових мультиплексорів.

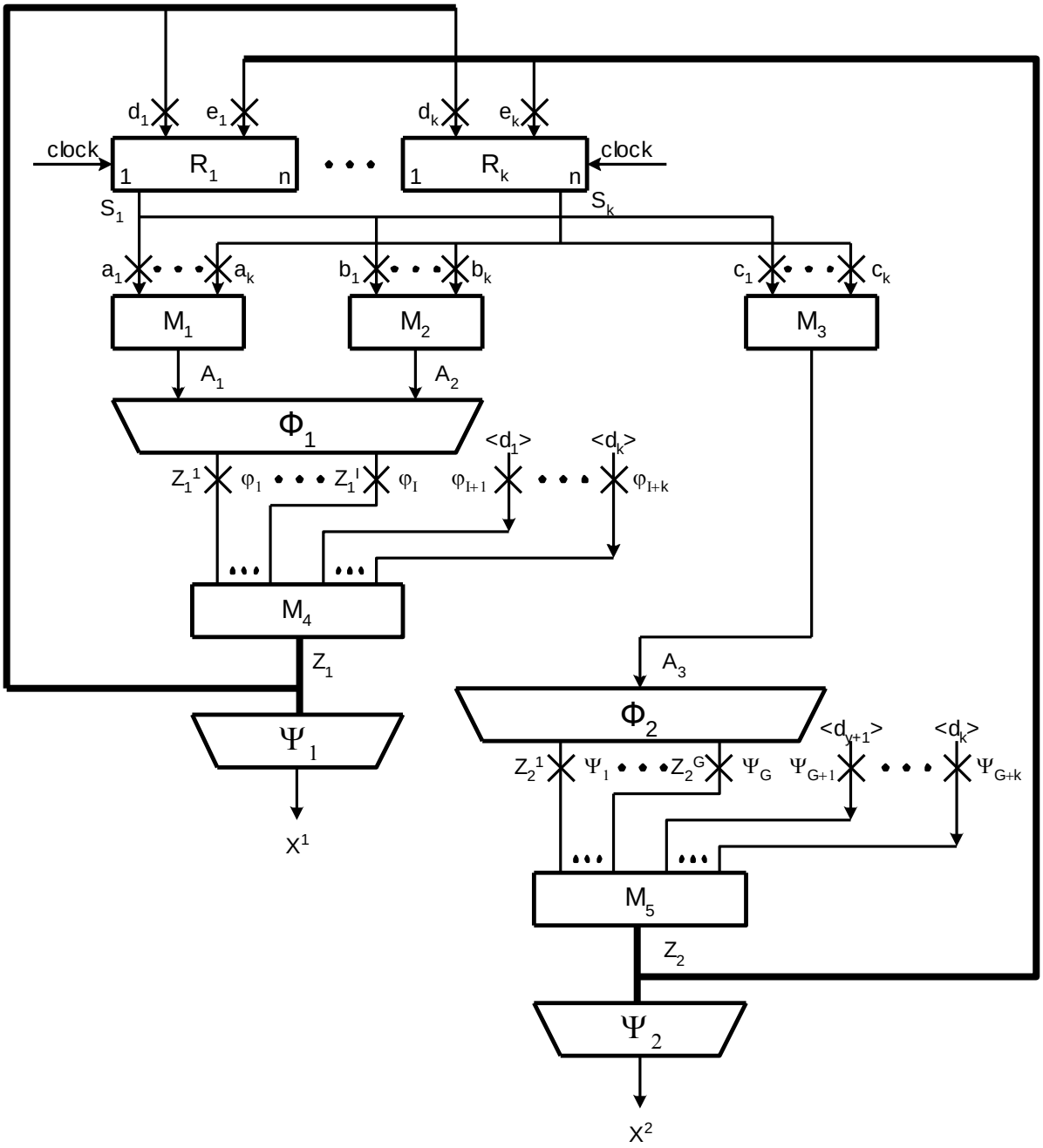


Рис. 8.16 – Структурна схема ІМ-ОА з паралельною КЧ

Очевидно, зростання продуктивності і, як наслідок, витрат апаратури в $IM_p - OA$ зростає в міру зростання числа схем Φ_i. При збігу числа схем і регістрів ($i = K$) відбувається трансформація $IM_p - OA$ в І-ОА. Як і в разі М-ОА, вихідна функціональна мікропрограма повинна бути перетворена з урахуванням обмежень, що накладаються структурою ОА.

Класичний $IM - OA$ з послідовною комбінаційною частиною ($IM_S - OA$) має три комбінаційні схеми $\Phi_1 - \Phi_3$ для виконання мікрооперацій і комбінаційну схему ψ для обчислення значень логічних умов (рис. 8.17). Цей автомат функціонує у такий спосіб.

Операнди з регістрів $S_1, \ \ldots, \ S_K$ передаються мультиплексорами M_1 і M_2 на шини A_1 і A_2. Схема Φ_1 виконує мікрооперації з одним із операндів (переважно, це інверсія слова або його частин). Результат виконання цих мікрооперацій $\alpha_i(A_1)$ надходить на шину A_3 через мультиплексор M_3 під керуванням операторів $\alpha_i \ (i = 1, \ \ldots, \ I)$. Схема Φ_2 виконує мікрооперації з двома операндами (додавання, диз'юнкція і так далі), і їхній результат $\beta_j(A_2, \ A_3)$ передається на шину A_4. Схема Φ_3 виконує різні мікрооперації зсуву (якщо вони потрібні) і формує остаточний результат виконання складної мікрооперації: $\gamma_g(\beta_j(A_2, \ \alpha_i(A_1)))$. Цей результат надходить на вхідну шину Z через мультиплексор M_5. Використовуючи цей мультиплексор, початкові значення операндів надходять у відповідні регістри пам'яті.

З аналізу схеми $IM_S - OA$ випливає, що автомат може виконувати до трьох мікрооперацій вихідної прошивки за один такт. Природно, це супроводжується зростанням часу циклу, порівняно з еквівалентним М-ОА. Як і раніше, необхідне перетворення вихідної прошивки.

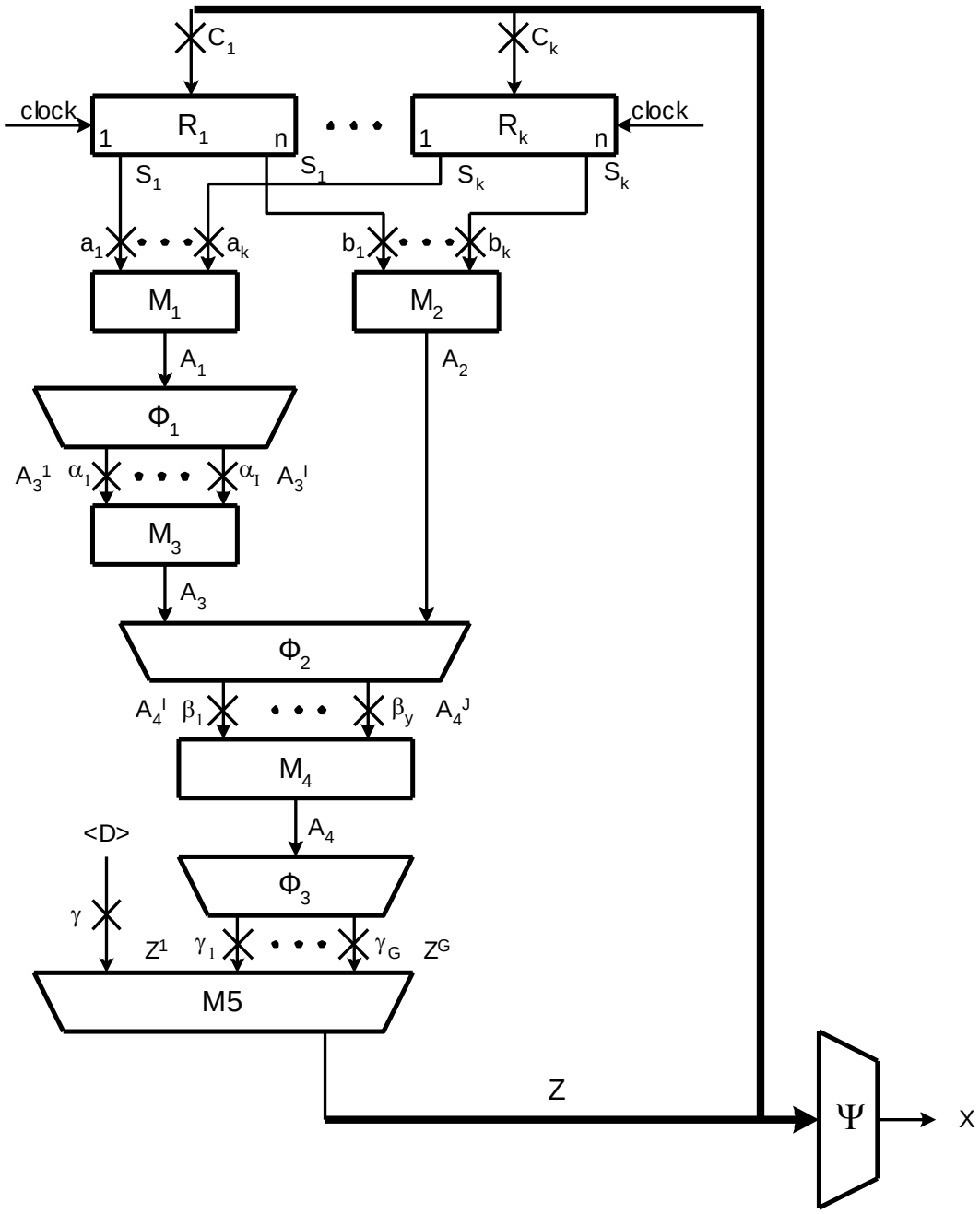


Рис. 8.17 – Структурна схема ІМ-ОА з послідовною КЧ

Продуктивність $IM_S - OA$ збільшується в міру зростання числа рівнів у комбінаційній частині. У межі $IM_S - OA$ є замовний цифровий пристрій, який виконує обчислення за один такт. Наприклад, замовна схема на рис. 8.18 за один такт виконує обчислення за формулою $S_1 = 5(A+B) \vee \frac{B \oplus C}{4} + A \oplus B \oplus C$, яка була раніше представлена функціональною мікропрограмою Γ_1 (рис. 8.11).

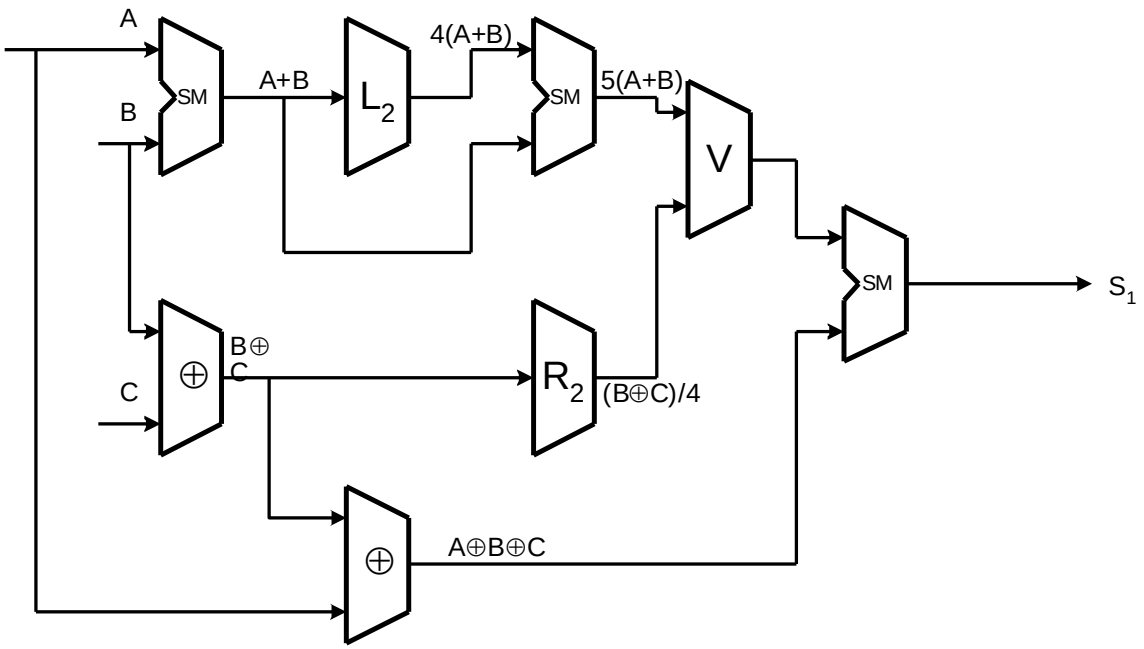


Рис. 8.18 – Рекомендована схема для обчислення функції S_1

Нехай t_S, t_i, t_{Sh}, t_M, t_R відповідно позначає час затримки для суматора, інвертора, зсувника, мультиплексора, а також час запису даних в регістр пам'яті ОА. Нехай C_K означає апаратурні витрати в схемі ОА з канонічною структурою, а C_M означає апаратурні витрати в схемі мультиплексора. Тоді характеристики різних операційних автоматів приведені в табл. 8.1.

У цій таблиці символ $C_{\min}$ означає мінімум витрат апаратури, досяжний тільки в разі, коли комбінаційна частина містить по одному різному операційному елементу (один акумулятор, один зсувник і так далі), t_ψ означає час формування значення логічних умов схемою ψ. Як відомо, операція підсумовування займає найбільше часу, порівняно з іншими типовими мікрооперaціями. Це зумовлене наявністю міжрозрядних переносів. З огляду на це час t_S було взято як визначальний для часу такту ОА. Символ L (H) використовується для позначення низького (високого) значення деякої характеристики автомата. Зауважимо, що табл. 8.1 дає тільки загальне уявлення про характеристики різних ОА.

На конкретні значення цих характеристик значно впливають особливості функціональних мікропрограм. З огляду на це остаточний вибір структури автомата проводиться на основі аналізу заданого критерію оптимальності (наприклад, мінімум витрат апаратури за середньої продуктивності), особливостей елементного базису і характеристик функціональної мікропрограми. Наприклад, обговоримо методи реалізації ОА всіх розглянутих типів для функціональної мікропрограми Γ_3 (рис. 8.19).

Таблиця 8.1

Характеристика операційних автоматів

Тип ОА	Продуктивність	Апаратура	Час циклу	Регуляр-ність	Універсаль-ність
I	$\le K$ max	$C_K + KC_M$ max	$t_R + t_M + t_S = t_I$ min	L	L
M	≤ 1 min	$C_K + 3C_M$ min	$t_I + t_M + t_\psi$	H	H
IM_p	≤ 2	$C_K + 5C_M$	$t_I + t_M + t_\psi$	H	H
IM_s	≤ 3	$C_{\min} + 5C_M$	$t_I + 3t_M + t_i + t_{Sh} + t_\psi$ max	H	H

Як це і повинно бути, у функціональній мікропрограмі Γ_3 кожна елементарна операція обробки даних відповідає типовій МО y_n, і множина мікрооперацій Y містить 18 різних типових мікрооперацій: $y_n \in Y = \{y_1, \ \ldots, \ y_{18}\}$. Природно, різні операційні вершини функціональної мікропрограми можуть містити мікрооперації з однаковими індексами, які відповідають однаковим операціям. Наприклад, операції елементарної обробки даних $S_2 := S_2 \oplus B$ всюди відповідає символ y_9.

У мікропрограмі Γ_3 є дві логічні умови, що визначають множину $X = \{x_1, \ \ x_2\}$. Очевидно, однакові перевірки відповідають однаковим логічним умовам.

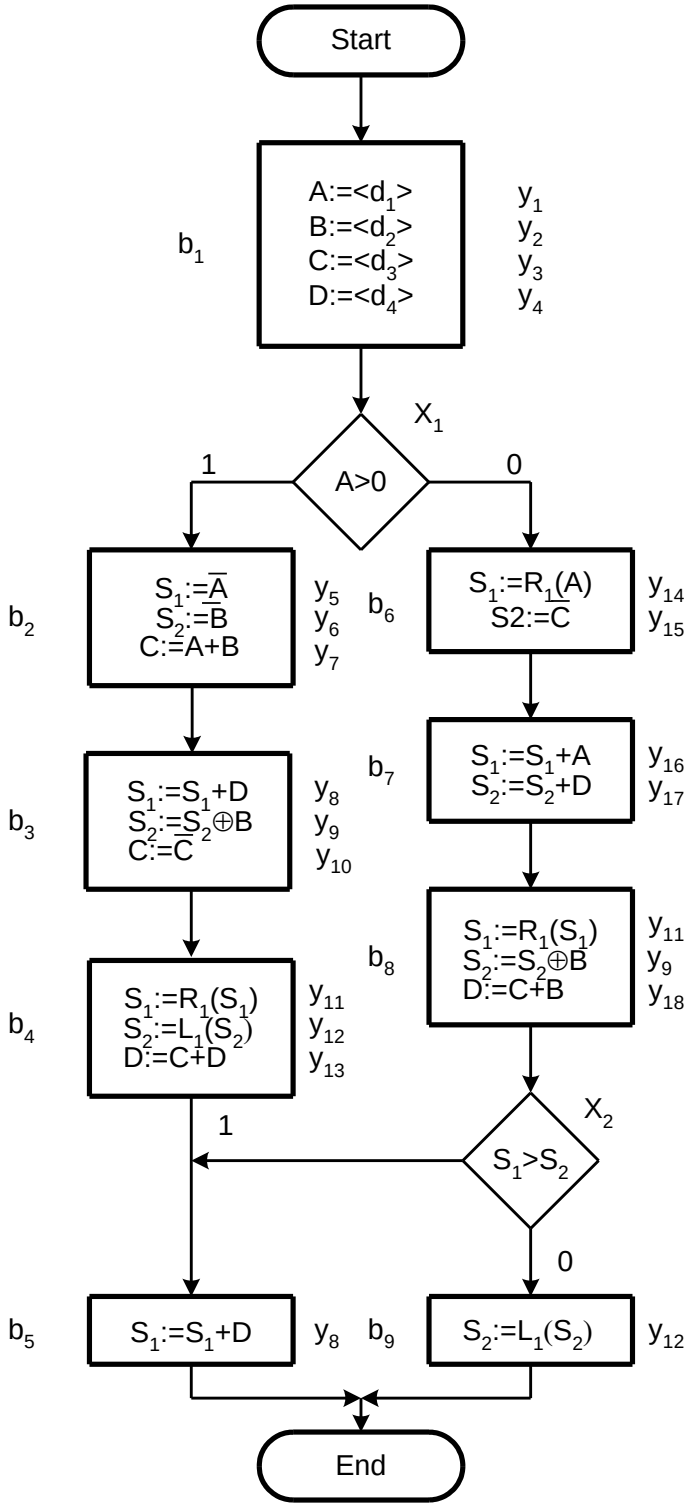


Рис. 8.19 – Функціональна мікропрограма Γ_3

Проєктування схеми будь-якого операційного автомата починається з проєктування його пам’яті. Для цього необхідно виконати аналіз слів, які записані у функціональній мікропрограмі. Існують слова трьох типів:

1. Вхідні слова (тип I). Ці слова беруть участь в МО ініціалізації і пов’язані з формуванням початкових значень операндів.

2. Вихідні слова (тип O). Ці слова відповідають результатам обчислень, що виконуються.

3. Внутрішні слова (тип L). Значення слів цього типу змінюються у процесі виконання обчислень. При цьому ініціалізація також розглядається як зміна значень слів.

Необхідно зазначити, що одне і те саме слово може належати декільком типам. Наприклад, можуть бути слова типу IL (вхідне і внутрішнє слово), типу LO (вихідне і внутрішнє слово) тощо.

Функціональна мікропрограма Γ_3 містить слова таких типів: A (тип IL), B (тип IL), C (тип IL), D (тип IL), S_1 (тип OL) і, нарешті, S_2 (тип OL). Кожному слову типу L відповідає регістр пам'яті автомата, при цьому зручно зберігати однакові позначення для слів і регістрів, які їм відповідають. Число тригерів в кожному регістрі збігається з розрядністю (числом біт) однойменного слова. Наприклад, для реалізації алгоритму, представленого функціональною мікропрограмою Γ_3, необхідно шість регістрів пам'яті. Позначимо їх символами RA, RB, RC, RD, RS_1 і RS_2, при цьому буква R відповідає слову «регістр». Подальші літери відповідають слову функціональної прошивки, яке зберігається у відповідному регістрі.

Список літератури до розділу 8

1. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

2. Майоров С. А., Новиков Г. И. Структура электронных вычислительных машин. Ленинград: Машиностроение, 1979. 384 с.

3. Adamski M., Barkalov A. Architectural and sequential synthesis of digital devices. Zielona Gora: University of Zielona Gora Press, 2006. 199 p.

4. Baranov S. Logic and system design of digital systems. Tallinn: TUT Press, 2008. 267 p.

5. Barkalov A. A., Titarenko L. A. Synthesis of operational and control automata. Donetsk: UNITECH, 2009. 256 p.

6. Gajski D. Principles of digital design. New Jersey: Prentice Hall, 1997. 318 p.

9 ПРОЄКТУВАННЯ СХЕМ ОПЕРАЦІЙНИХ АВТОМАТІВ

9.1 Проєктування операційного автомата типу I

Метод проєктування I-ОА охоплює такі етапи:

1. Формування пам'яті автомата. У загальному випадку число регістрів пам'яті ОА збігається з числом його внутрішніх слів. Нагадаємо, що ми розглядаємо методи синтезу схем ОА з використанням функціональної мікропрограми Γ_3. У цьому конкретному випадку пам'ять включає шість регістрів. Кожне вхідне слово визначає один вхідний полюс схеми, пов'язаний з відповідним регістром. Кожне вихідне слово відповідає вихідному полюсу схеми, пов'язаному з відповідним регістром. Зауважимо, що все це умовно, оскільки в конкретному проєкті слова зазвичай заносяться із зовнішньої пам'яті, куди потім заноситься і результат виконання обчислень, які визначаються функціональною мікропрограмою.

Домовимося позначати символом $I(\Gamma)$ той факт, що деякий I-ОА проєктується для обчислень за функціональною мікропрограмою Γ. Вхідні і вихідні полюси (порти, контакти) автомата $I(\Gamma_3)$ показані на рис. 9.1. Нехай всі слова, які використовуються в нашому прикладі, мають $n=16$ біт, тобто всі регістри автомата $I(\Gamma_3)$ мають по $n=16$ тригерів. Автомат $I(\Gamma_3)$ має чотири вхідні полюси $\langle d_1\rangle$, …, $\langle d_4\rangle$. Вихідні значення операндів завантажуються через ці порти відповідно до керуючих сигналів y_1, …, y_4. Автомат має також два вихідні полюси ($\langle r_1\rangle$ і $\langle r_2\rangle$), оскільки у функціональній мікропрограмі Γ_3 є тільки два слова типу О.

2. Розбиття множини мікрооперацій на класи еквівалентних мікрооперацій. Мікрооперації називаються еквівалентними, якщо вони

мають однаковий приймач інформації (обчислюють нове значення одного і того самого слова) і належать до одного класу типових мікрооперацій. Розглянемо, наприклад, такі типові мікрооперації: $y_1 \# S_1 := A + B$, $y_2 \# S_1 := R_1(A)$, $y_3 \# S_2 := A + B$, $y_4 \# S_1 := C + D$. Їхній аналіз показує, що тільки МО y_1 і y_4 є еквівалентними. Наприклад, МО y_1 і y_2 мають однаковий приймач, але належать до різних класів типових мікрооперацій, а МО y_1 і y_3 належать до одного класу типових МО, але мають різні приймачі результуючої інформації.

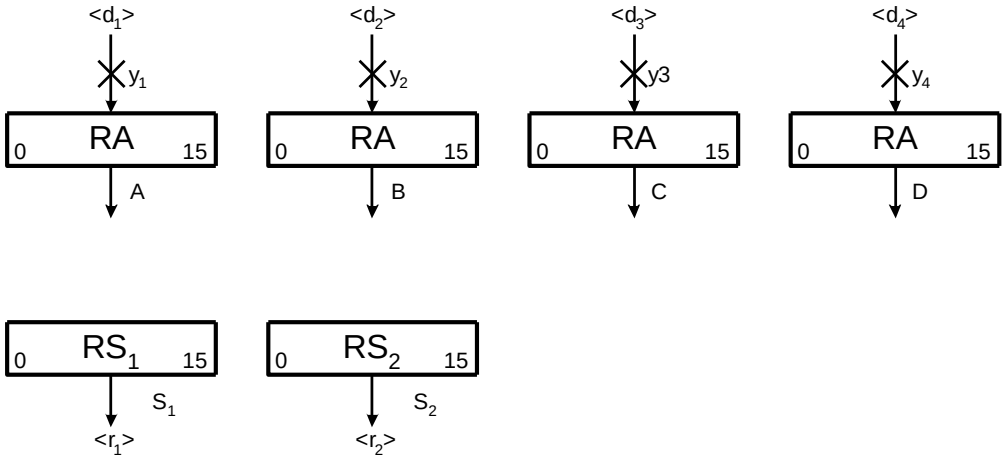


Рис. 9.1 – Реалізація пам'яті операційного автомата $I(\Gamma_3)$

Нехай $\Pi_Y = \{B_1, \ \ldots, \ B_J\}$ – розбиття множини мікрооперацій Y на класи (блоки) еквівалентних мікрооперацій $B_j \in \Pi_Y$. У разі ОА $I(\Gamma_3)$ це розбиття має вигляд $\Pi_Y = \{B_1, \ \ldots, \ B_{13}\}$, де $B_1 = \{y_1\}, \ \ldots, B_5 = \{y_5\}$, $B_6 = \{y_6, \ y_{15}\}, B_7 = \{y_7\}, \quad B_8 = \{y_8, \ y_{16}\}, \quad B_9 = \{y_9\}, \quad B_{10} = \{y_{10}\}$, $B_{11} = \{y_{11}, \ y_{14}\}, \ B_{12} = \{y_{12}, \ y_{17}\}, \ B_{13} = \{y_{13}, \ y_{18}\}$. Зауважимо, що МО y_{12} і y_{17} поміщені в один блок розбиття Π_Y, оскільки зсув ліворуч можна замінити складанням.

3. Формування узагальнених операторів. Узагальнений оператор – це спеціальна форма подання еквівалентних мікрооперацій. Використання цієї форми дає змогу більш компактно представити операції, що

виконуються автоматом. Наприклад, дві мікрооперації $y_1 \# S_1 := A + B$ і $y_2 \# S_1 := C + D$ можна уявити одним узагальненим оператором $S_1 := A_1 + A_2$, до якого входять такі вирази:

$$A_1 = \begin{cases} A, if\ y_1 = 1 \\ C, if\ y_2 = 1 \end{cases}, \quad A_2 = \begin{cases} B, if\ y_1 = 1 \\ D, if\ y_2 = 1 \end{cases}.$$

Узагальнений оператор $S_1 := A_1 + A_2$ відповідає схемі на рис. 9.2а, де вирази A_1 і A_2 реалізовані як мультиплексори. Використання узагальнених операторів дає змогу зменшити число операційних елементів у схемі ОА. Наприклад, у процесі реалізації схеми, що відповідає безпосередній реалізації на основі мікрооперацій $y_1 \# S_1 := A + B$ і $y_2 \# S_1 := C + D$, потрібно в два рази більше суматорів (рис. 9.2б). Вираз для оператора A_1 представимо у вигляді: $A_1 = Ay_1 \vee Cy_2$.

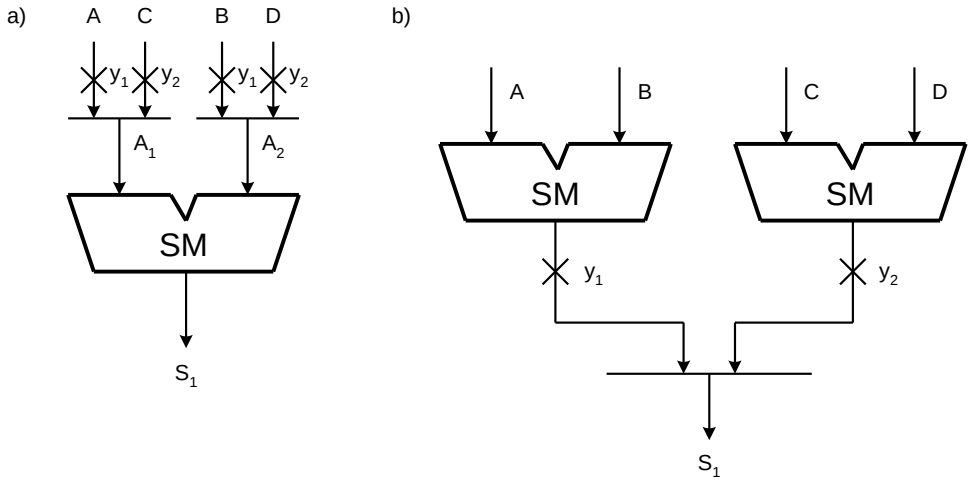


Рис. 9.2 – Реалізація схеми для еквівалентних мікрооперацій з використанням (а) і без використання (б) узагальнених операторів

Для автомата $I(\Gamma_3)$ можна отримати такі узагальнені оператори: $S_2 := \overline{A_1}$, де $A_1 := By_6 \vee Cy_{15}$, $S_1 := S_1 + A_2$, де $A_2 := Dy_8 \vee Ay_{16}$, $S_1 := R_1(A_3)$, де $A_3 := S_1 y_{11} \vee Ay_{14}$, $S_2 := S_2 + A_4$, де $A_4 := S_2 y_{12} \vee Dy_{17}$, і $D := C + A_5$, де $A_5 := Dy_{13} \vee By_{18}$.

4. Розбиття множини мікрооперацій щодо приймачів слів. Множину Y можна уявити як об'єднання її підмножин $Y_S \subseteq Y$, які визначаються приймачем інформації S. У розглянутому прикладі існують такі підмножини множини МО для вхідних слів: $Y_A = \{y_1\}$, $Y_B = \{y_2\}$, $Y_C = \{y_3, \ y_7, \ y_{10}\}$, $Y_D = \{y_4, \ y_{13}, \ y_{18}\}$, а також дві підмножини для результатів: $Y_{S_1} = \{y_5, \ y_8, \ y_{11}, \ y_{14}, \ y_{16}\}$, $Y_{S_2} = \{y_6, \ y_9, \ y_{12}, \ y_{15}, \ y_{17}\}$.

5. Синтез комбінаційної схеми автомата. Кожна множина Y_S відповідає одній частині (підсхемі) результуючої комбінаційної схеми І-ОА. Якщо множина Y_S включає більше одного елемента, то виходи відповідних операційних елементів об'єднуються мультиплексором. Наприклад, множина Y_D відповідає схемі, показаній на рис. 9.3. Тут узагальнений оператор A_S відповідає еквівалентним мікрооперaціям y_{13} і y_{18}. Зауважимо, що мікрооперація y_4 належить до мікрооперацій ініціалізації; реалізація частини схеми ОА для мікрооперації y_4 була показана на рис. 9.1. Об'єднання підсхем для всіх множин $Y_S \subseteq Y$ в єдину комбінаційну схему і її з'єднання з пам'яттю автомата призводить до результуючої схеми І-ОА $I(\Gamma_3)$, наведеної на рис. 9.4.

Комбінаційна частина (схема) операційного автомата $I(\Gamma_3)$ містить дві комбінаційні схеми, необхідні для обчислення значень логічних умов $x_1 \# A > 0$ і $x_2 \# S_1 > S_2$. Перша з них аналізує тільки знак числа, а друга є досить складною, тому що обробляє по 16 розрядів обох чисел.

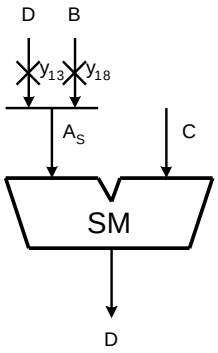


Рис. 9.3 – Реалізація підсхеми для множини Y_D

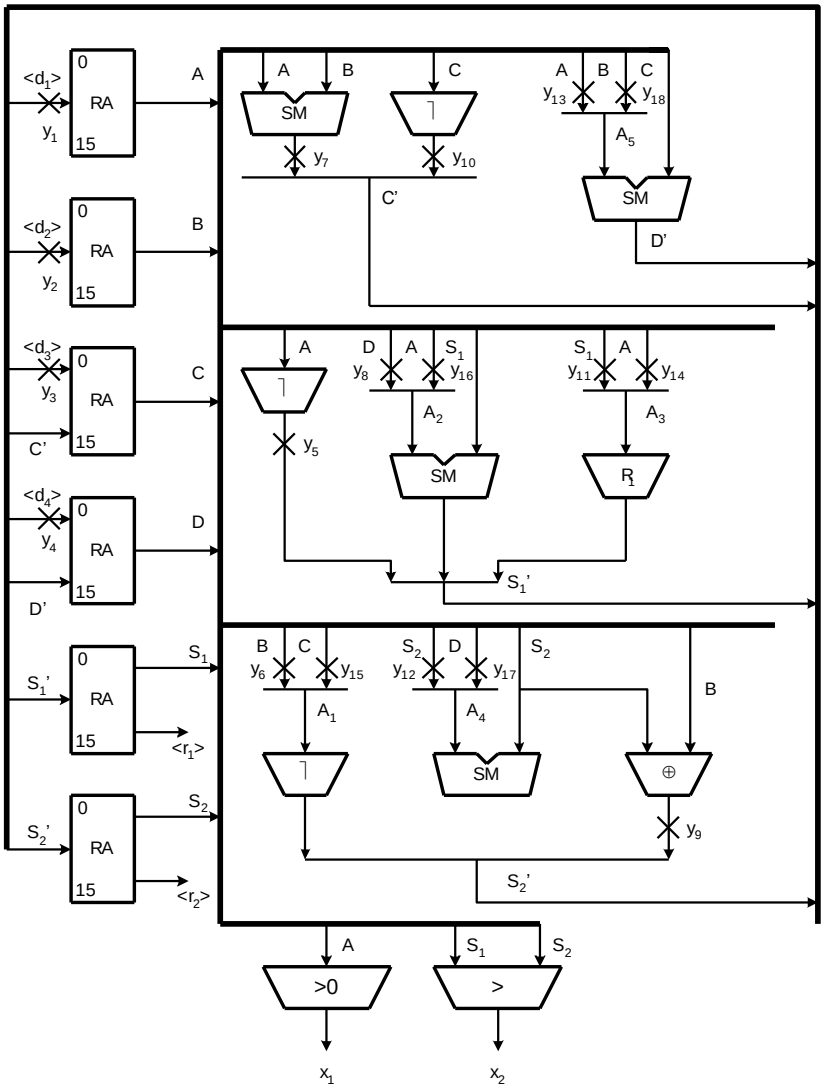


Рис. 9.4 – Функціональна схема операційного автомата $I(\Gamma_3)$

Визначимо основні характеристики цього автомата, а саме: продуктивність $E_I(\Gamma)$; апаратурні витрати $Q_I(\Gamma)$; час циклу $T_I(\Gamma)$ і швидкодія автомата $LT_I(\Gamma)$ (тобто середній час виконання алгоритму Γ).

Продуктивність ОА визначається за формулою

$$E(\Gamma) = N(\Gamma)/O(\Gamma), \tag{9.1}$$

де $N(\Gamma)$ позначає загальне число мікрооперацій в операційних вершинах функціональної мікропрограми Γ, $O(\Gamma)$ – число операторних вершин в алгоритмі. Для розглянутого прикладу ОА $I(\Gamma_3)$ маємо: $N(\Gamma) = 22$, $O(\Gamma) = 9$, отже, знаходимо продуктивність $E_I(\Gamma_3) = 2{,}44$.

Апаратурні витрати можна визначити як ціну обладнання, використаного в схемі ОА. Другий шлях – оцінка в деяких умовних одиницях, що ми і виберемо. Приймемо, що вартість суматора дорівнює 1, а решта операційних елементів коштують в 10 разів менше, тобто 0,1. Схема ОА $I(\Gamma_3)$ містить 4 суматори, 3 багаторозрядні інвертори, один суматор за модулем 2 і 3 мультиплексори. Отже, $Q_I(\Gamma_3) = 4{,}8$. Оскільки пам'ять еквівалентних автоматів різних типів (І-, М- і ІМ- ОА) має однакову кількість регістрів, то ми не будемо враховувати її ціну під час загальної оцінки різних ОА. Якщо апаратурні витрати оцінюються як використана частина площі кристала, то проєктувальник повинен брати витрати по площі для окремих операційних елементів з бібліотеки проєкту.

Щоб визначити час такту, необхідно знати час затримки для кожного операційного елемента. Для регістрів повинно бути відомо час перемикання. Нехай час затримки для вентиля І-НЕ становить τ деяких одиниць. Домовимося, що час затримки суматора $t_S = 20\tau$, а для інших елементів (зсувник, інвертор, мультиплексор) ці часи однакові і дорівнюють 2τ. Нехай тригери регістрів пам'яті переключаються за час $t_r = 6\tau$. У разі І-ОА час такту можна визначити у такий спосіб:

$$\mathrm{T}_I(\Gamma) = \tau_{CC} + \tau_r. \qquad (9.2)$$

У (9.2) час $\tau_{CC} = \max(\tau_s,\ \tau_{sh},\ \tau_i) + 2\tau_M = \tau_S + 2\tau_M$ позначає час затримки в комбінаційній частині ОА. Для розглянутого прикладу можна знайти, що $\mathrm{T}_I(\Gamma_3) = 22\tau + 6\tau = 28\tau$.

Для визначення швидкодії $LT_I(\Gamma)$ необхідно знайти середнє число циклів $n_I(\Gamma)$, необхідне для виконання обчислень за алгоритмом Γ для І-ОА. Потім ці параметри ($n_I(\Gamma)$ і $T_I(\Gamma)$) перемножуються, даючи значення $LT_I(\Gamma)$. Параметр $n_I(\Gamma)$ може бути знайдений статистично,

використовуючи моделювання з різними значеннями аргументів. Однак можна використати і таку процедуру:

1. Знайти всі шляхи $\alpha_1, \ \ldots, \ \alpha_J$ між вершинами Start і End.

2. Знайти число операторних вершин n_j для кожного шляху α_j.

3. Знайти параметр $n_I(\Gamma)$ як суму всіх n_j.

4. Знайти $LT_I(\Gamma)$ як результат ділення $n(\Gamma)$ на J.

Отже, швидкодію ОА можна знайти за формулою

$$LT_I(\Gamma) = \sum_{j=1}^{J} n_j \left(T_I(\Gamma) + t_{CU}\right) \Big/ J . \qquad (9.3)$$

У виразі (9.3) символ t_{CU} означає час циклу пристрою управління. При цьому ми припускаємо, що ймовірності для всіх шляхів α_j однакові.

У мікропрограмі Γ_3 існують три шляхи: $\alpha_1 = \langle b_1, \ \ldots, \ b_5 \rangle$ – за $x_1 = 1$; $\alpha_2 = \langle b_1, \ b_6, \ b_7, \ b_8, \ b_5 \rangle$ – за значення ЛУ $x_1 = 0$ і $x_2 = 1$; $\alpha_3 = \langle b_1, \ b_6, \ b_7, \ b_8, \ b_9 \rangle$ – за $x_1 = 0$ і $x_2 = 0$. Отже, маємо $J = 3$, при цьому $n_j = 5$, відтак можна отримати, що $LT_I(\Gamma_3) = 150\tau + 5t_{CU}$.

Отже, для І-ОА $I(\Gamma_3)$ знайдені такі значення його основних характеристик: продуктивність $E_I(\Gamma_3) = 2{,}44$, апаратурні витрати $Q_I(\Gamma_3) = 4{,}8$, час циклу $T_I(\Gamma_3) = 30\tau$ і латентність $LT_I(\Gamma_3) = 150\tau + 5t_{CU}$.

Усі розглянуті характеристики І-ОА можна знайти, використовуючи формули (9.1)–(9.3). Очевидно, для конкретних проєктів необхідно знати конкретні значення часових параметрів всіх операційних елементів і їхню вартість. Це дасть змогу знайти реальні значення всіх характеристик схеми автомата. Однак теоретичне визначення основних параметрів ОА дає змогу попередньо оцінити різні структури еквівалентних ОА. Переважно, такого порівняння достатньо для вибору оптимального рішення, що задовольняє задані критерії.

9.2 Проєктування операційного автомата типу М

Метод проєктування схеми М-ОА складається з таких кроків:

1. Формування пам'яті автомата. Цей етап виконується у такий самий спосіб, як і для І-ОА. Позначимо символом $M(\Gamma)$ той факт, що деякий М-ОА синтезується з функціональної мікропрограми Γ. Пам'ять автомата $M(\Gamma_3)$ складається з регістрів RA, RB, RC, RD, RS_1, RS_2, кожен з яких має по 16 розрядів.

2. Розподіл регістрів між шинами ОА. Зі структури ОА на рис. 8.15 випливає, що шини A_1 і A_2 використовуються для передачі інформації з регістрів пам'яті до операційних елементів. Ці шини зі свого боку є виходами мультиплексорів M_1 і M_2. Для оптимізації схем мультиплексорів необхідно розділити множину регістрів на два класи, в яких немає загальних елементів. Якщо таке розбиття неможливе через особливості реалізованого алгоритму обчислень, необхідно знайти покриття множини регістрів, що складається з підмножин з мінімальним числом загальних елементів. Отже, перетин $A_1 \cap A_2$ має бути мінімальним. Тут A_i означає множину регістрів, інформація з яких передається через мультиплексор means M_i. У західній літературі такий підхід називається «групуванням з'єднань».

При виконанні розподілу регістрів особлива увага повинна приділятися мікрооперації з двома операндами. Наприклад, для МО $S = A + B$ регістри RA і RB повинні бути розміщені в різних множинах A_1 і A_2. Прийнявши цей факт до уваги, можна знайти такий розподіл регістрів для $M(\Gamma_3)$: $A_1 = \{A,\ C,\ S_1,\ S_2\}$, $A_2 = \{A,\ B,\ C,\ D,\ S_2\}$. Пояснимо принцип його формування. Регістр S_2 повинен бути розташований в обох класах цього покриття, оскільки МО зсуву ліворуч зводиться до підсумовування: $L_1(S_2) = S_2 + S_2$. Зазначимо, що регістри A, B, C поміщаються в клас A_2, що необхідно для виконання порозрядної інверсії

$S_1 := \overline{A}$, $S_2 := \overline{B}$, $C := \overline{C}$ з використанням тільки одного інвертора, пов'язаного з шиною A_2.

У разі автомата $M(\Gamma_3)$ кожна з шин A_1 і A_2 має 16 розрядів, що збігається з розрядністю переданих слів. Якщо деякий регістр має менше розрядів, ніж шини автомата, то необхідно вирівнювати дані. Наприклад, молодші розряди регістра і шини будуть збігатися, що відповідає вирівнюванню по правій межі слів.

3. Формування таблиці операторів автомата. Як було зазначено раніше, для виконання мікрооперацій вихідної прошивки $y_n \in Y$ в М-ОА використовуються оператори a_i, b_j, c_k, φ_g (рис. 8.15). Для знаходження операторів необхідно побудувати таблицю зі стовпцями y_n, A_1, A_2, Z, RS. Ця таблиця відбиває структуру автомата і показує, які оператори використовуються для виконання вихідних мікрооперацій. Наприклад, для виконання мікрооперації $y_7 \# C := A + B$ необхідно виконати такі оператори: $A_1 := RA$, $A_2 := RB$, $Z := A_1 + A_2$, $RC := Z$. Ці оператори необхідно записати у відповідних колонках таблиці, що формується. У такий спосіб формується таблиця операторів, наведена для нашого прикладу в табл. 9.1. Таблиця операторів має стільки рядків, скільки мікрооперацій включає вихідна мікропрограма. Наприклад, для ОА $M(\Gamma_3)$ таблиця операторів має $N = 18$ рядків, що збігається з числом МО у функціональній мікропрограмі Γ_3.

4. Формування логічних умов. З аналізу структури ОА типу М випливає, що значення логічних умов обчислюються комбінаційною схемою, пов'язаною з шиною Z. Це означає, що формули для обчислення ЛУ повинні бути перетворені, порівняно з функціональною мікропрограмою. Наприклад, умову $x_1 \# A > 0$ необхідно представити у вигляді $x_1 \# Z > 0$. Для обчислення значень умови $x_2 \# S_1 > S_2$ обидва регістри RS_1 і RS_2 повинні бути наявними на шині Z одночасно, що для

М-ОА неможливо. Отже, в схему необхідно ввести додатковий регістр RZ, що пов'язаний з шиною Z і зберігає значення одного зі слів. Якщо регістр RZ зберігає значення слова S_2, то умова x_2 перетворюється в умову $x_2 \# Z > RZ$. Зазначимо, що слово S_2 має бути записано в регістр RZ (для чого використовується оператор $c_6 = 1$). Зауважимо, що в ЕОМ така перевірка виконується шляхом вирахування слів і аналізу знака різниці. Однак в наших прикладах ми використовуємо інші підходи, які більш характерні для спеціалізованої цифрової апаратури.

Таблиця 9.1

Таблиця операторів М-ОА $M(\Gamma_3)$

y_n	A_1	A_2	Z	RS
y_1	–	–	$\langle d_1 \rangle$ φ_1	RA c_1
y_2	–	–	$\langle d_2 \rangle$ φ_2	RB c_2
y_3	–	–	$\langle d_3 \rangle$ φ_3	RC c_3
y_4	–	–	$\langle d_4 \rangle$ φ_4	RD c_4
y_5	–	RA b_1	$\overline{A_2}$ φ_5	RS_1 c_5
y_6	–	RB b_2	$\overline{A_2}$ φ_5	RS_2 c_6
y_7	RA a_1	RB b_2	$A_1 + A_2$ φ_6	RC c_3
y_8	RS_1 a_2	RD b_3	$A_1 + A_2$ φ_6	RS_1 c_5
y_9	RS_2 a_3	RB b_2	$A_1 \oplus A_2$ φ_7	RS_2 c_6
y_{10}	–	RC b_4	$\overline{A_2}$ φ_5	RC c_3
y_{11}	RS_1 a_2	–	$R_1(A_1)$ φ_8	RS_1 c_5
y_{12}	RS_2 a_3	RS_2 b_5	$A_1 + A_2$ φ_6	RS_2 c_6
y_{13}	RC a_4	RD b_3	$A_1 + A_2$ φ_6	RD c_4
y_{14}	RA a_1	–	$R_1(A_1)$ φ_8	RS_1 c_5
y_{15}	–	RC b_4	$\overline{A_2}$ φ_5	RS_2 c_6
y_{16}	RS_1 a_2	RA b_1	$A_1 + A_2$ φ_6	RS_1 c_5
y_{17}	RS_2 a_3	RD b_3	$A_1 + A_2$ φ_6	RS_2 c_6
y_{18}	RC a_4	RB b_2	$A_1 + A_2$ φ_6	RD c_4

5. Синтез функціональної схеми автомата. Схема автомата типу М реалізується з використанням таблиці операторів. Стовпець RS

визначає пам’ять автомата та оператори передачі інформації в регістри з шини Z. Стовпці A_1, A_2 визначають входи відповідних мультиплексорів і оператори передачі інформації через ці мультиплексори до комбінаційної частини автомата. Стовпець Z визначає операційні елементи комбінаційної частини і оператори передачі інформації від цих елементів до мультиплексора Z. Схема формування логічних умов потрібна для обчислення значень ЛУ $x_l \in X$, використовуючи модифіковані формули для кожної з умов. Функціональна схема автомата $M(\Gamma_3)$ наведена на рис. 9.5.

6. Формування перетвореної прошивки. Очевидно, що пристрій управління, який входить до складу операційного пристрою, має формувати оператори a_i, b_j, c_k, φ_g замість мікрооперацій $y_n \in Y$, записаних у вихідній мікропрограмі. Ба більше, М-ОА може виконувати в кожному такті роботи тільки одну мікрооперацію. Крім того, необхідні додаткові такти для обчислення значень логічних умов. Наприклад, для обчислення умови $x_1 \# A > 0$ інформація з регістра RA повинна бути доставлена на шину Z. Отже, для формування перетвореної прошивки необхідно: подати кожну операційну вершину вихідної прошивки у вигляді послідовності операторних вершин, кожна з яких включає тільки одну мікрооперацію; замінити мікрооперації відповідними операторами з таблиці операторів; ввести додаткові вершини для обчислення значень ЛУ (якщо це необхідно).

При перетворенні прошивки необхідно враховувати залежність між мікрооперaціями за даними. Кажуть, що МО $y_n \in Y$ залежить за даними від МО $y_m \in Y$, якщо результат виконання першої з них є операндом другої МО. Ця залежність впливає на порядок проходження вершин в перетвореній мікропрограмі.

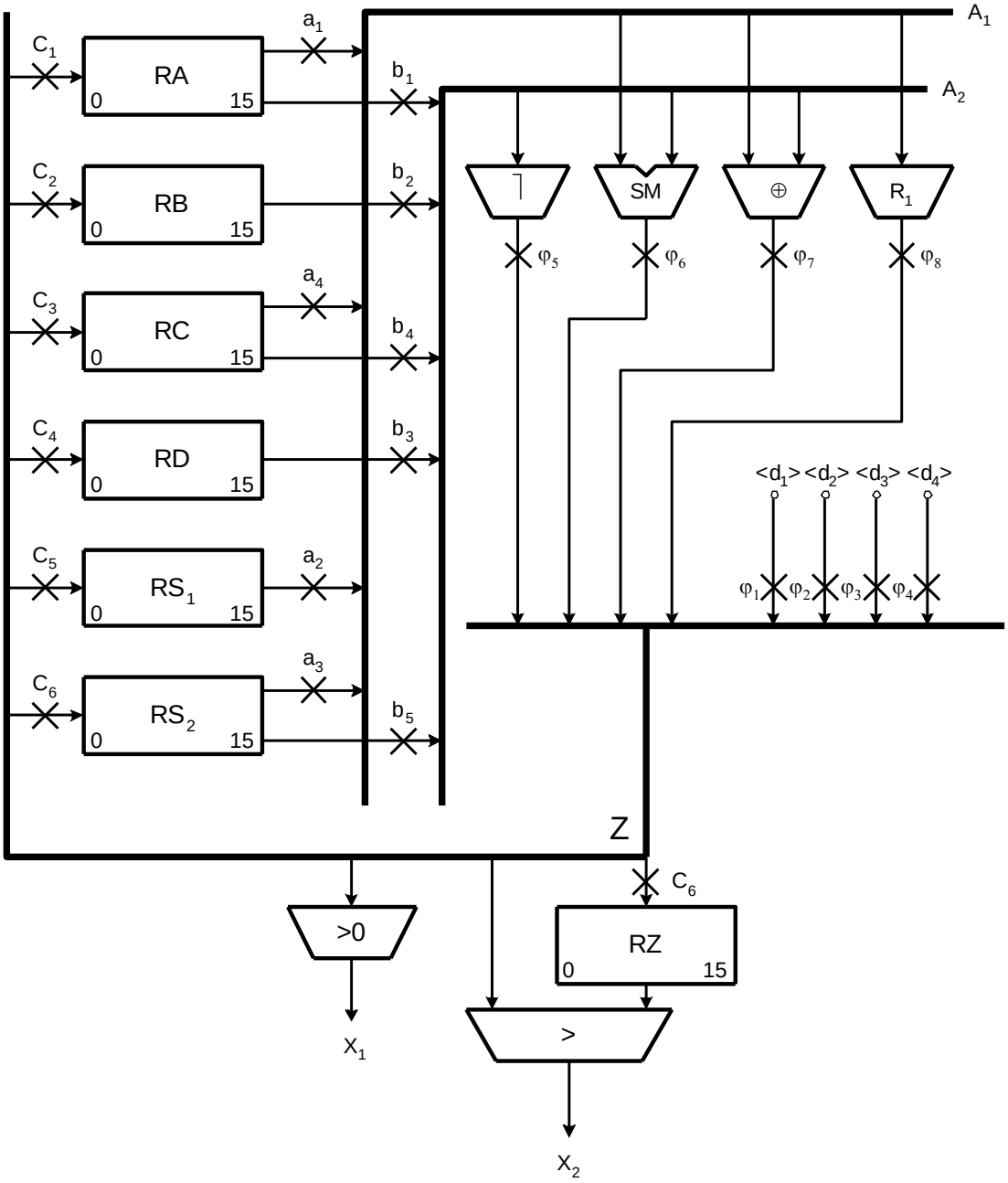


Рис. 9.5 – Функціональна схема М-ОА для прошивки Γ_3

Розглянемо деякі приклади, пов’язані з впливом залежності за даними (рис. 9.6). Мікрооперації y_3 і y_4 не залежать за даними, що випливає з рис. 9.6а. З огляду на це вони можуть слідувати в перетвореній мікропрограмі в будь-якому порядку. Наприклад, ці МО можуть слідувати в порядку $\langle y_3,\ y_4 \rangle$, чому відповідає фрагмент перетвореної прошивки, показаний на рис. 9.6б. Мікрооперація y_5 залежить за даними від МО y_3

(рис. 9.6в). Це означає, що МО y_5 повинна виконуватися першою, щоб використовувати значення слова S_1 до його зміни після виконання МО y_3 (рис. 9.6г). Розглянемо більш складний випадок, де МО y_6 і y_7 взаємозалежні за даними (рис. 9.6д). Очевидно, що будь-який порядок їхнього послідовного виконання ($\langle y_6,\ y_7\rangle$ або $\langle y_7,\ y_6\rangle$) призводить до помилок в обчисленні результату (рис. 9.6е). Для усунення такого конфлікту необхідно ввести додаткові слово D і МО $y_8 \# D := S_1$ (рис. 9.6ж). Для цього прикладу вихідна МО y_7 перетворюється в МО $y_9 \# S_2 := D + C$, що приводить до послідовності $\langle y_8,\ y_6,\ y_9\rangle$. Зрозуміло, що подібне перетворення веде до збільшення часу виконання алгоритму і пов'язане з введенням додаткового регістра RD.

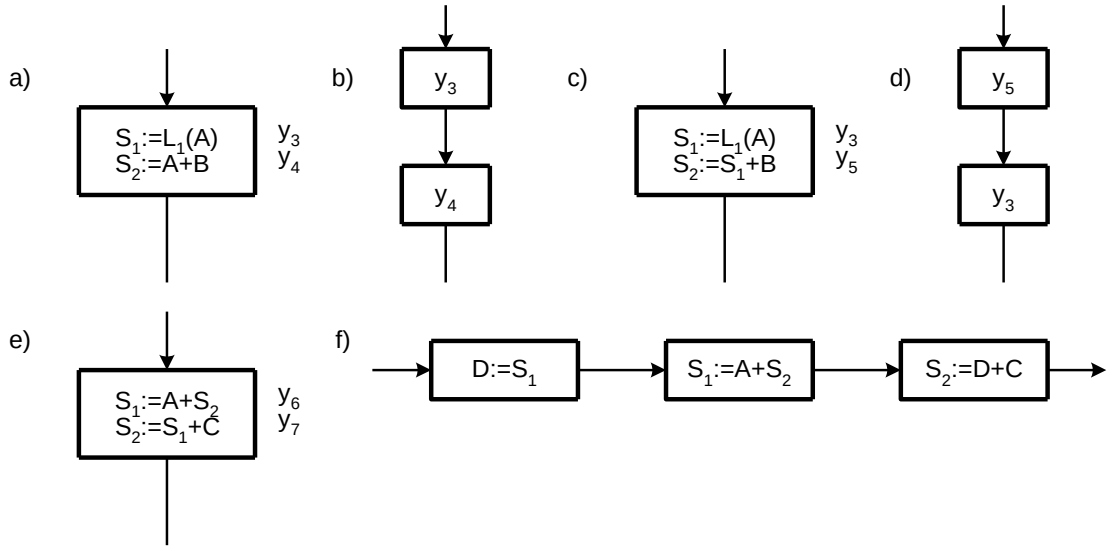


Рис. 9.6 – Облік залежності за даними в М-ОА

Додаткові операторні вершини можуть знадобитися і для перевірки значень логічних умов. Наприклад, для обчислення умови $x_1 \# A > 0$ (рис. 9.7а) в перетворену мікропрограму необхідно ввести вершину для передачі слова з регістру RA на шину Z (рис. 9.7б). Операнд повинен бути наявним на шині Z в момент перевірки пов'язаної з ним логічної умови. Отже, вершина з рис. 9.7в повинна бути перетворена в послідовність вершин з мікрооперацій $\langle y_4,\ y_3\rangle$, що показано на рис. 9.7г.

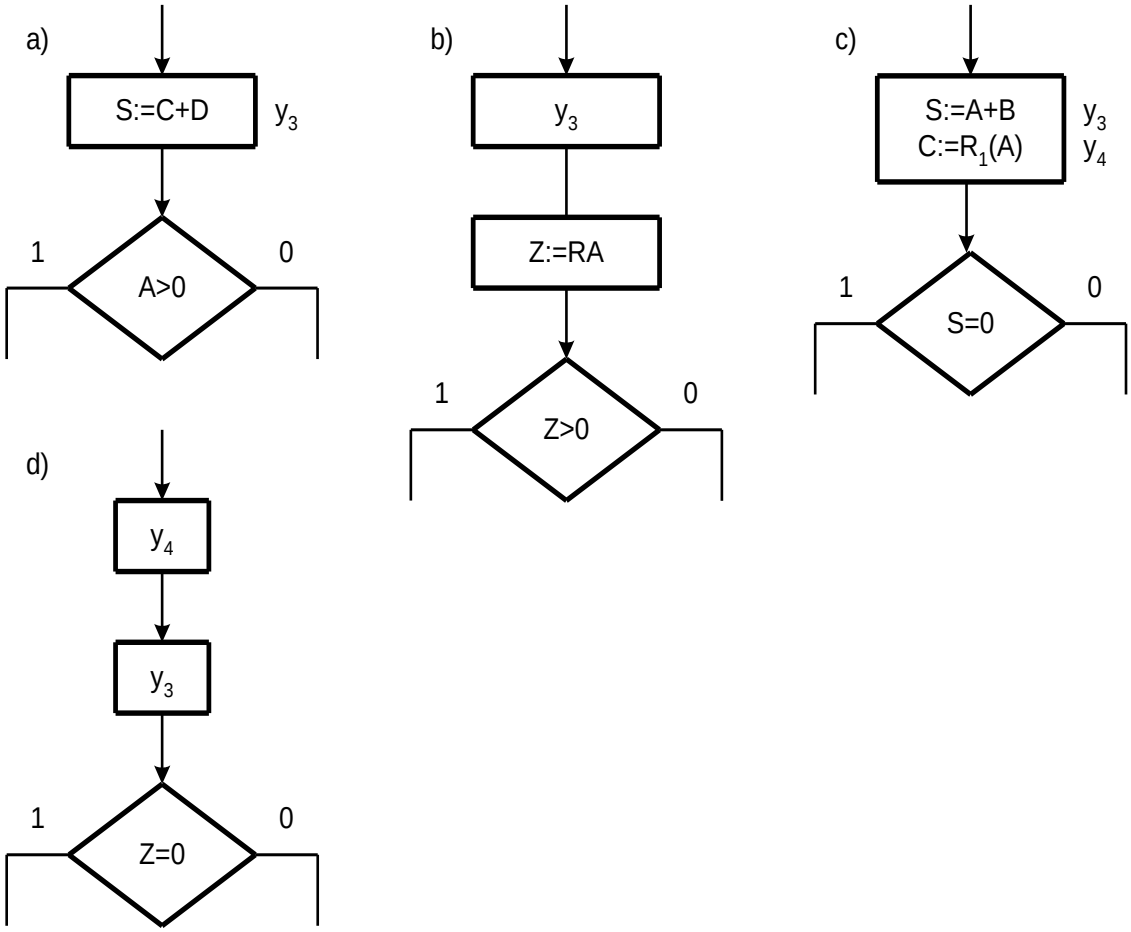


Рис. 9.7 – Формування логічних умов в М-ОА

У разі послідовності мікрооперацій $\langle y_3, \ y_4 \rangle$ операція $Z := RS$ повинна виконуватися перед обчисленням логічної умови, що призводить до збільшення числа тактів обчислень. Зазначимо, що порядок виконання мікрооперацій (тобто, $\langle y_3, \ y_4 \rangle$ або $\langle y_4, \ y_3 \rangle$) не є суттєвим, якщо пам'ять ОА включає додатковий регістр RZ, пов'язаний зі схемою обчислення логічних умов (рис. 9.8а). У цьому випадку вміст шини Z має бути записано в регістр RZ, що веде до двох варіантів перетвореної прошивки (рис. 9.8б, в). Розглянуті тут приклади дають достатнє уявлення про проблеми, які виникають при обчисленні логічних умов. Зауважимо, що на практиці деякі перевірки проводяться шляхом віднімання операндів.

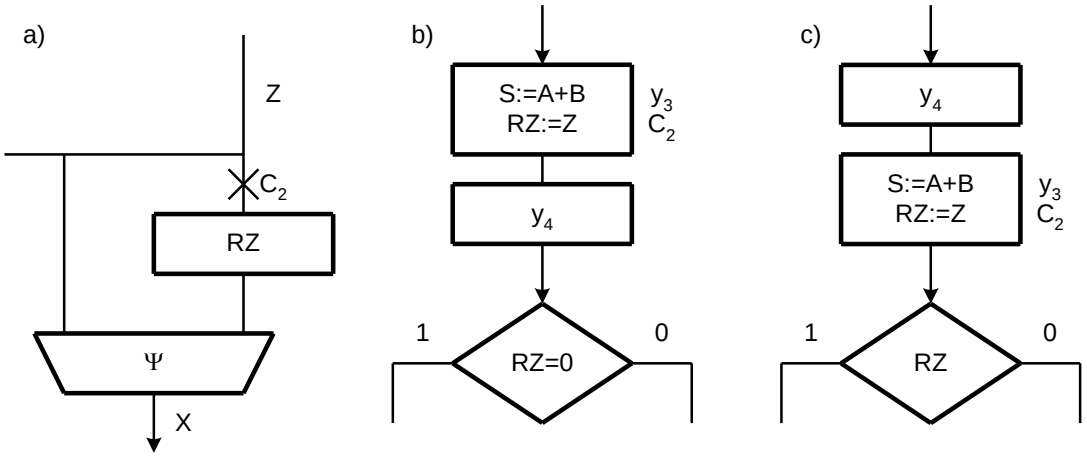


Рис. 9.8 – Формування логічних умов з додатковим регістром

Усі зазначені особливості були використані при перетворенні вихідної функціональної мікропрограми Γ_3. Для перевірки логічної умови $x_1 \# A > 0$ мікрооперація y_1 повинна виконуватися безпосередньо перед цією перевіркою. Це означає, що вершина b_1 повинна бути перетворена в таку послідовність $\langle y_2,\ y_3,\ y_4,\ y_1 \rangle$, тобто їй відповідають чотири вершини перетвореної прошивки. Для перевірки логічної умови $x_2 \# S_1 > S_2$ мікрооперація y_{11} з вершиною b_8 повинна виконуватися безпосередньо перед перевіркою. Отже, вершина b_8 перетворюється в послідовність вершин з МО $\langle y_9,\ y_{18},\ y_{11} \rangle$. Аналіз вершин $b_2 - b_9$ функціональної мікропрограми Γ_3 показує, що вони не містять мікрооперацій, які залежать за даними одна від одної. Отже, вершина b_2 представляється у вигляді послідовності вершин, що відповідає вектору $\langle y_5,\ y_6,\ y_7 \rangle$. Позначимо цей факт за допомогою рівності $b_2 = \langle y_5,\ y_6,\ y_7 \rangle$. За аналогією для інших вершин можна написати такі рівності: $b_3 = \langle y_8,\ y_9,\ y_{10} \rangle$, $b_4 = \langle y_{11},\ y_{12},\ y_{13} \rangle$, $b_5 = \langle y_8 \rangle$, $b_6 = \langle y_{14},\ y_{15} \rangle$, $b_7 = \langle y_{16},\ y_{17} \rangle$, $b_9 = \langle y_{12} \rangle$>. Перетворена мікропрограма $\Gamma_3(M)$ представлена на рис. 9.9.

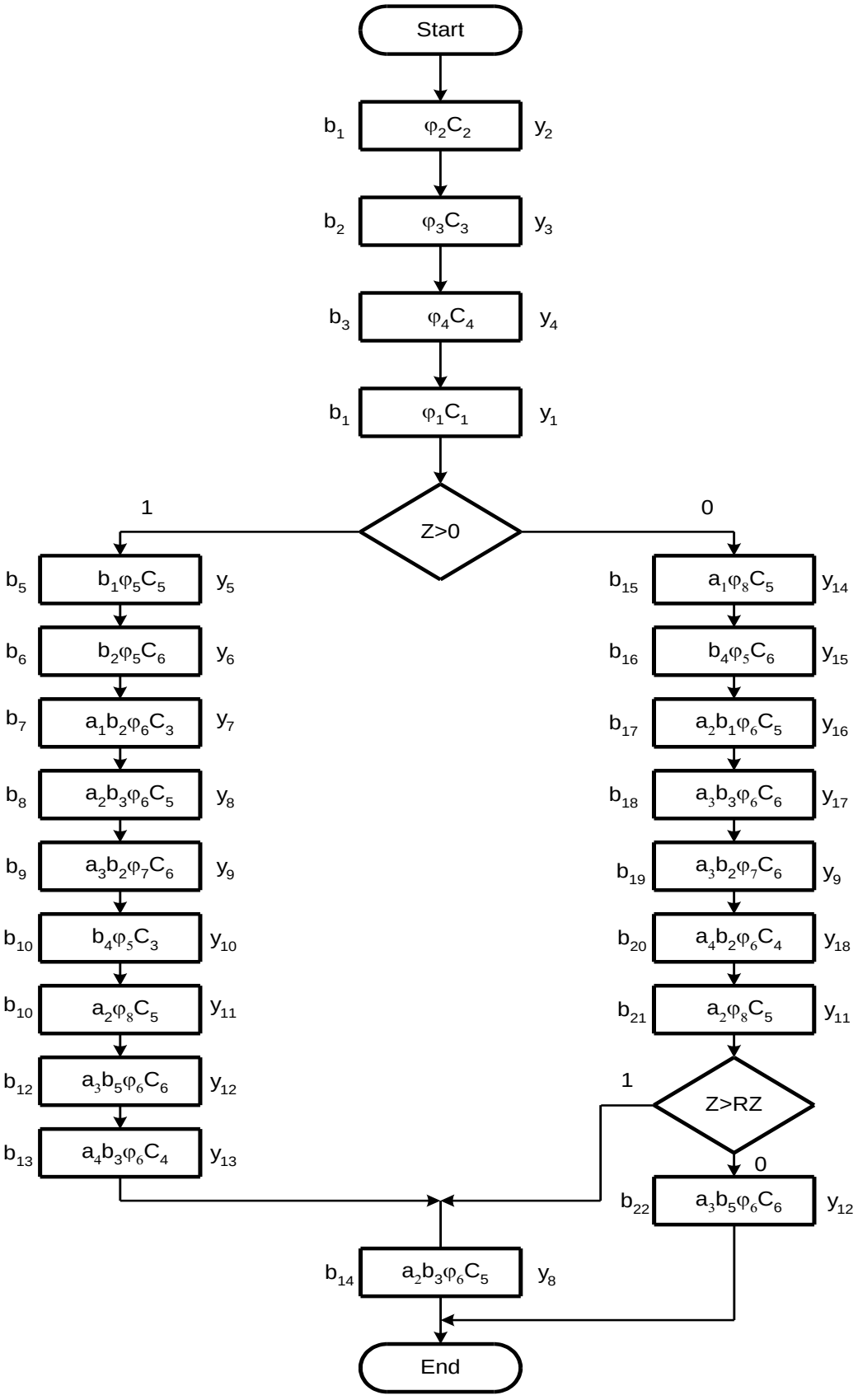


Рис. 9.9 – Перетворена мікропрограма $\Gamma_3(M)$

Для побудови цієї прошивки мікрооперації $y_n \in Y$ були замінені на відповідні сукупності операторів з табл. 9.1. Символ «М» в позначенні $\Gamma_3(M)$ на рис. 9.9 підкреслює той факт, що вихідна мікропрограма перетворена з урахуванням особливостей М-ОА. Визначимо характеристики автомата $M(\Gamma_3)$, використовуючи раніше розглянуті для І-ОА методи.

Продуктивність $E_M(\Gamma_3)=1$, оскільки кожна операторна вершина перетвореної прошивки містить тільки одну типову мікрооперацію, виражену через оператори автомата $M(\Gamma_3)$. Цей параметр може бути і менше одиниці, якщо в мікропрограму введені додаткові вершини (усунення залежності за даними, обчислення логічних умов з використанням додаткових операцій). У нашому прикладі таке перетворення не проводилося, тому $E_M(\Gamma_3)=1$. В комбінаційній частині автомата є 1 суматор, 1 зсувник, 1 суматор за модулем 2 і 3 мультиплексори. Отже, апаратурні витрати для нашого прикладу визначаються як $O_M(\Gamma_3)=1{,}6$. Для визначення часу циклу $T_M(\Gamma_3)$ застосована формула (9.2), яка дає змогу знайти $\tau_{cc}=30\tau$. Як це було і в разі еквівалентного І-ОА, для даного вбудованого М-ОА є три шляхи, але вони охоплюють більше вершин, ніж для І-ОА.

Отже, існує тільки три шляхи вбудованого $\Gamma_3(M)$: це шлях $\alpha_1=\langle b_1,\ \ldots,\ b_4,\ b_5,\ \ldots,\ b_{14}\rangle$, шлях $\alpha_2=\langle b_1,\ \ldots,\ b_4,\ b_{15},\ \ldots,\ b_{22}\rangle$, і шлях $\alpha_3=\langle b_1,\ \ldots,\ b_4,\ b_{15},\ \ldots,\ b_{21},\ b_{14}\rangle$, де $n_1=14$, $n_2=n_3=12$. Можна знайти, що $LT_M(\Gamma_3)=((14+12+12)/3)(30\tau+t_{CU})=378\tau+12{,}6t_{CU}$, для чого використовується формула (9.3). Для визначення більш точної оцінки швидкодії необхідно знати час циклу пристрою управління. Як видно, швидкодію операційного пристрою можна змінити завдяки зміні базису в схемі пристрою управління.

Отже, М-ОА $M(\Gamma_3)$ має такі базові характеристики: продуктивність $E_M(\Gamma_3)=1$; апаратурні витрати $O_M(\Gamma_3)=1{,}6$; час циклу $T_M(\Gamma_3)=30\tau$; швидкодія $LT_M(\Gamma_3)=378\tau+12{,}6t_{CU}$.

Зазначимо, що пам'ять ОА можна реалізувати на однотактних тригерах (latch memory elements), замінивши ними двотактні тригери (master-slave flip-flops). У цьому разі пам'ять повинна включати додатковий регістр RZ, стан якого змінюється за сигналом $Clock_1$ (рис. 9.10). Дані з виходу регістра RZ переписуються в регістри R_1, …, R_K синхросигналом $Clock_2$. Це дає змогу практично в два рази зменшити число тригерів в пам'яті ОА. Якщо пам'ять М-ОА (рис. 8.15) вимагає $2nK$ однотактних тригерів, то введення регістра RZ зменшує це число до $nK+n$, тобто майже вдвічі.

Очевидно, що ціна пам'яті автомата пропорційна кількості тригерів у ній. Це означає, що введення додаткового регістра вдвічі зменшує вартість пам'яті М-ОА щодо еквівалентного автомата типу І. Зазначимо, що для І-ОА така організація пам'яті неможлива. Це зумовлено тим, що інформація в кожен з регістрів пам'яті І-ОА заноситься з індивідуальної комбінаційної схеми. Це ще раз підтверджує той факт, що за збільшення швидкодії (зменшення часу обчислень) необхідно платити додатковою апаратурою.

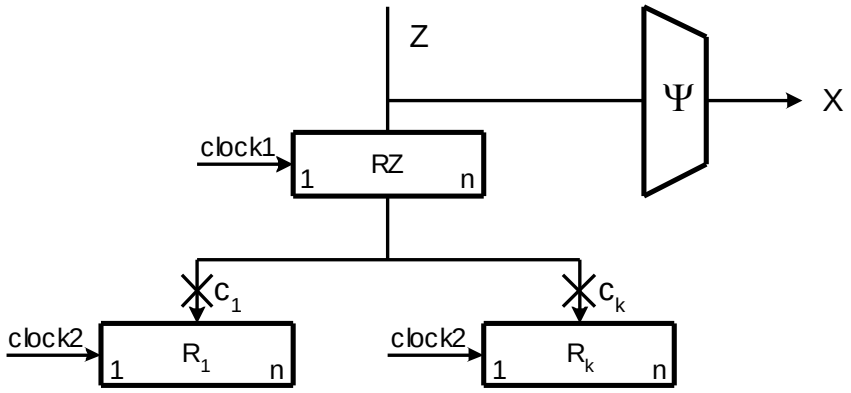


Рис. 9.10 – Оптимізація пам'яті М-ОА

9.3 Проєктування ІМ-ОА з паралельною комбінаційною частиною

Як було зазначено раніше, $IM_p - OA$ можна розглядати як два М автомати із загальною пам'яттю. Розглянемо метод його синтезу, використовуючи в прикладі функціональну мікропрограму Γ_3 (рис. 8.19).

1. Формування пам'яті автомата. Цей крок виконується так само, як і для інших типів ОА (І-ОА або М-ОА). Аналіз прошивки Γ_3 показує, що пам'ять автомата $IM_p(\Gamma_3)$ складається з 6 регістрів, кожен з яких має 16 розрядів.

2. Розбиття множини мікрооперацій. Як і для М-ОА, множина Y має бути розбита на два класи. Однак на відміну від М-ОА, один з класів включає тільки МО з одним операндом, а другий – з двома. Позначимо ці класи відповідно символами Y^1 і Y^2.

У разі прошивки Γ_3 формуються два класи мікрооперацій: клас $Y^1 = \{y_1,\ y_2,\ y_5,\ y_6,\ y_{10},\ y_{11},\ y_{14},\ y_{15}\}$ і клас $Y^2 = \{y_3,\ y_4,\ y_7,\ y_8,\ y_9,\ y_{12},\ y_{13},\ y_{16},\ y_{17},\ y_{18}\}$. Перший з них містить 8, а другий – 10 мікрооперацій. Зазначимо, що клас Y^2 включає МО y_{12}, оскільки зсув ліворуч замінюється складанням: $S_2 + S_2$. Далі, мікрооперації ініціалізації $y_1 - y_4$ рівномірно розподілені між класами Y^1 і Y^2. Це дає змогу підвищити продуктивність автомата, порівняно з еквівалентним М-ОА.

3. Розподіл регістрів за шинами. Цей розподіл приводить до множин $A_1 - A_3$, яким відповідають мультиплексори $M_1 - M_3$ з комбінаційної частини $IM_p - OA$ (рис. 8.16). Правила розподілу за шинами збігаються з раніше розглянутими для М-ОА. Це означає, що необхідно отримати розподіл, наближений до рівномірного розбиття, тобто з однаковим числом слів. У нашому прикладі для автомата $IM_p(\Gamma_3)$

формуються множини: $A_1 = \{A,\ C,\ S_1,\ S_2\}$, $A_2 = \{A,\ B,\ S_1,\ S_2\}$, $A_3 = \{A,\ B,\ S_1,\ C\}$.

4. Формування таблиці операторів для виконання МО з одним операндом. Ця таблиця містить стовпці y_n, A_3, Z_2, RS_1, які відбивають елементи з М-ОА, необхідні для виконання мікрооперацій з одним операндом. Нагадаємо, що до мікрооперацій цього типу належать різноманітні зрушення, а також зміна деяких розрядів слова шляхом їхнього інвертування. У разі розглянутого автомата $IM_p(\Gamma_3)$ ця таблиця має вісім рядків, кожен з яких відповідає одному елементу множини Y^1 (табл. 9.2).

Таблиця 9.2

Таблиця операторів для виконання МО з одним операндом

y_n	A_3	Z_2	RS_1
y_1	–	$\langle d_1 \rangle\ \psi_1$	$RA\ e_1$
y_2	–	$\langle d_2 \rangle\ \psi_2$	$RB\ e_2$
y_5	$RA\ c_1$	$\overline{A_3}\ \psi_3$	$RC\ e_3$
y_6	$RB\ c_2$	$\overline{A_3}\ \psi_3$	$RS_1\ e_4$
y_{10}	$RC\ c_3$	$\overline{A_3}\ \psi_3$	$RC\ e_3$
y_{11}	$RS_1\ c_4$	$R_1(A_3)\ \psi_4$	$RS_1\ e_4$
y_{14}	$RA\ c_1$	$R_1(A_3)\ \psi_4$	$RS_1\ e_4$
y_{15}	$RC\ c_3$	$\langle d_1 \rangle\ \psi_1$	$RS_2\ e_5$

5. Формування таблиці операторів для виконання МО з двома операндами. Ця таблиця має стовпці y_n, A_1, A_2, Z_1, RS_2, які відбивають елементи структури відповідного М-ОА. Цей ОА виконує двомісні операції. У разі автомата $IM_p(\Gamma_3)$ ця таблиця має 10 рядків, кожен з яких відповідає одному елементу множини Y^2 (табл. 9.3). Зазначимо, що сумарно обидві таблиці мають 18 рядків, що збігається з числом МО в множині Y.

Таблиця 9.3

Таблиця операторів для виконання МО з двома операндами

y_n	A_1	A_2	Z_1	RS_2
y_3	–	–	$\langle d_3 \rangle$ φ_1	RC d_1
y_4	–	–	$\langle d_4 \rangle$ φ_2	RD d_2
y_7	RA a_1	RB b_1	$A_1 + A_2$ φ_3	RC d_1
y_8	RS_1 a_2	RD b_2	$A_1 + A_2$ φ_3	RS_1 d_3
y_9	RS_2 a_3	RB b_1	$A_1 \oplus A_2$ φ_4	RS_2 d_4
y_{12}	RS_2 a_3	RS_2 b_3	$A_1 + A_2$ φ_3	RS_2 d_4
y_{13}	RC a_4	RD b_2	$A_1 + A_2$ φ_3	RD d_2
y_{16}	RS_1 a_2	RA b_4	$A_1 + A_2$ φ_3	RS_1 d_3
y_{17}	RS_2 a_3	RD b_2	$A_1 + A_2$ φ_3	RS_2 d_4
y_{18}	RC a_4	RB b_1	$A_1 + A_2$ φ_3	RD d_2

6. Формування логічних умов. Цей крок виконується за тією ж методикою, що і для М-ОА. У нашому прикладі інформація в регістр RA надходить з шини Z_2. Отже, логічну умову $x_1 \# A > 0$ має бути перетворено в формулу $x_1 \# Z_2 > 0$. Оскільки вміст регістрів RS_1 і RS_2 можна одночасно передати відповідно на шини Z_1 і Z_2, то логічна умова $x_2 \# S_1 > S_2$ має обчислюватися як $x_2 \# Z_1 > Z_2$. Це означає, що тепер немає необхідності у використанні додаткового регістра RZ, як це було в еквівалентному автоматі $M(\Gamma_3)$. Цей підхід можна застосовувати, постійно зменшуючи кількість регістрів у пам'яті операційного автомата. Однак при цьому можуть з'явитися зв'язки між регістрами і шинами, які необхідні тільки для виконання перевірок (обчислення значень логічних умов).

7. Реалізація функціональної схеми автомата. Цей крок синтезу ОА виконується за методикою, аналогічною методиці синтезу схеми М-ОА. На першому етапі формується пам'ять операційного автомата, а далі відбувається синтез комбінаційної частини схеми. Стовпець A_1 визначає зв'язки між регістрами пам'яті і мультиплексором A_1; стовпець A_2

визначає зв'язки між регістрами пам'яті і мультиплексором A_2; стовпець RS_1 визначає зв'язки між шиною Z_1 і регістрами пам'яті; стовпець Z_1 визначає частину схеми ОА (операційні елементи), яка служить для виконання мікрооперацій з двома операндами. Стовпець A_3 визначає зв'язки між регістрами пам'яті і мультиплексором A_3; стовпець Z_2 визначає частину схеми ОА, що виконує мікрооперації з одним операндом; стовпець RS_2 задає зв'язки між шиною Z_2 і регістрами пам'яті ОА.

Функціональна схема операційного автомата $IM_p(\Gamma_3)$ показана на рис. 9.11. Зауважимо, що, як і в попередньому прикладі, пам'ять $IM_p - OA$ може бути реалізована на однотактних тригерах (які спрацьовують по імпульсу синхронізації). Такий підхід пов'язаний з необхідністю введення двох додаткових регістрів RZ_1 і RZ_2, пов'язаних з шинами Z_1 і Z_2, що показано на рис. 9.12.

8. Формування перетвореної прошивки. Кожен цикл роботи $IM_p - OA$ може бути пов'язаний з виконанням двох сумісних мікрооперацій $\langle y_n, \ y_m \rangle$, де $y_n \in Y^1$, $y_m \in Y^2$. Отже, кожна операційна вершина перетвореної прошивки може включати до двох мікрооперацій, представлених за допомогою операторів автомата. Як і у разі М-ОА, необхідно враховувати залежність за даними між МО. Наприклад, мікрооперації $y_1 \# S_1 := A + B$ і $y_2 \# A := \overline{A}$ не можуть бути виконані одночасно. Перетворена мікропрограма $\Gamma_3(IM_p)$ наведена на рис. 9.13.

У мікропрограму $\Gamma_3(IM_p)$ включена вершина для передачі слова з регістру RS_1 на шину Z_1, а також для передачі слова з регістру RS_2 на шину Z_2. Крім того, додаткова вершина b_{14} введена для можливості перевірки логічної умови $x_2 \# Z_1 > Z_2$.

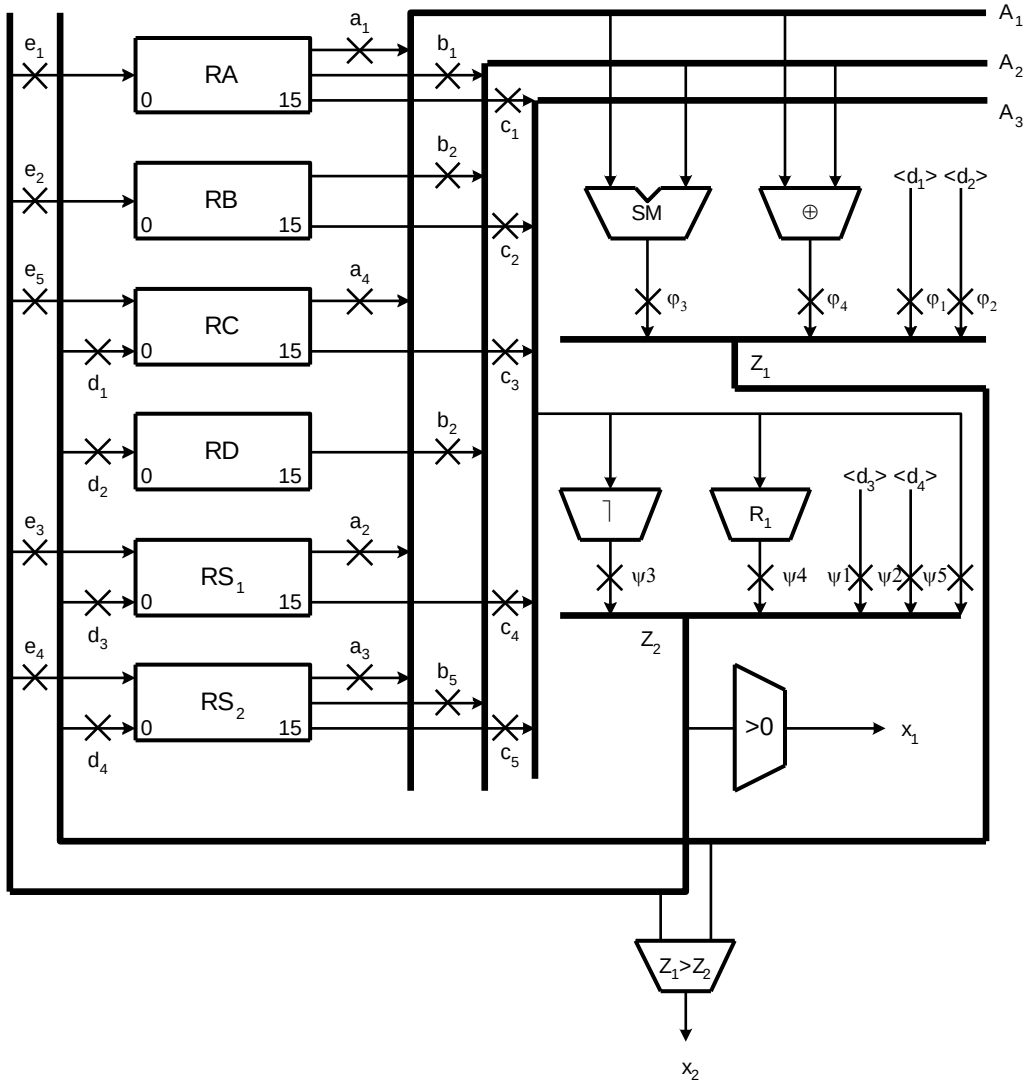


Рис. 9.11 – Функціональна схема автомата $IM_p(\Gamma_3)$

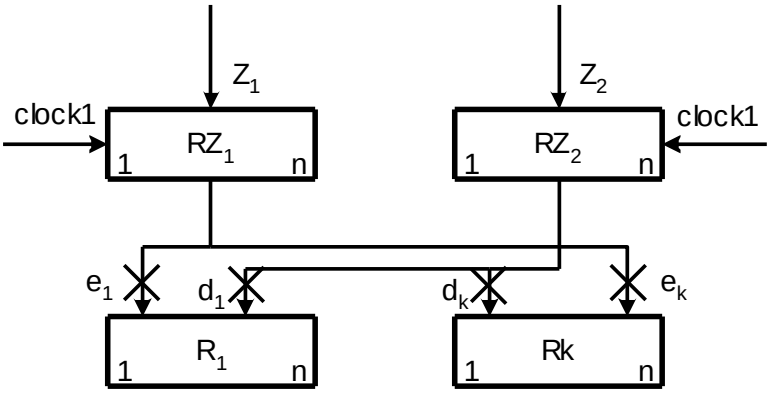


Рис. 9.12 – Оптимізація пам'яті $IM_p - ОА$

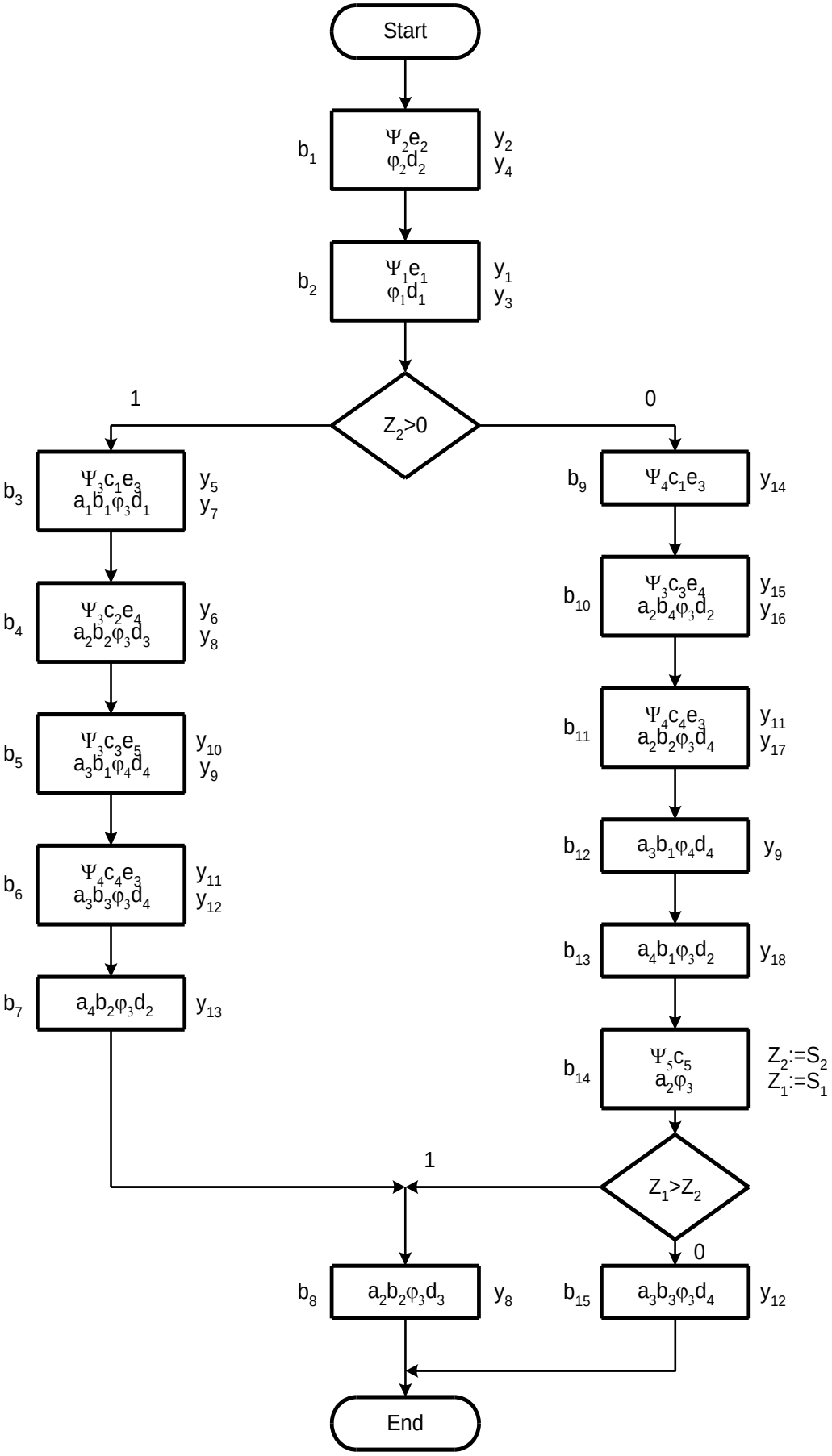


Рис. 9.13 – Перетворена мікропрограма $\Gamma_3(IM_p)$

Знайдемо основні характеристики автомата $IM_p(\Gamma_3)$, використовуючи формули (9.1)–(9.3).

Для оцінки продуктивності $E_p(\Gamma_3)$ автомата $IM_p(\Gamma_3)$ необхідно використовувати формулу (9.1). Перетворена мікропрограма $\Gamma_3(IM_p)$ включає $O(\Gamma_3)=15$ операторних вершин. У цих вершинах розташовані 22 мікрооперації вихідної прошивки Γ_3. Зауважимо, що b_{14} не включає вихідні мікрооперації. Частка від ділення 22 на 15 дає $E_p(\Gamma_3)=1{,}47$. У комбінаційну частину автомата $IM_p(\Gamma_3)$ входить акумулятор, зсувник, суматор за модулем 2, інвертор і 5 мультиплексорів. Це дає величину витрат апаратури $O_p(\Gamma_3)=1{,}8$. Час циклу $T_p(\Gamma_3)$ можна знайти як $\mathrm{T}_p(\Gamma_3)=\tau_S+2\tau_M+\tau_R=30\tau$.

Перетворена мікропрограма $\Gamma_3(IM_p)$ включає три шляхи, а саме: шлях $\alpha_1=\langle b_1,\ \ldots,\ b_8\rangle$, шлях $\alpha_2=\langle b_1,\ b_2,\ b_9,\ \ldots,\ b_{14},\ b_8\rangle$ і $\alpha_3=\langle b_1,\ b_2,\ b_9,\ \ldots,\ b_{14},\ b_{15}\rangle$, для яких число елементів дорівнює $n_1=8,\ n_2=n_3=9$. Використовуючи формулу (9.3), можна визначити, що ОА має $LT_M(\Gamma_3)=26(30\tau+t_{CU})/3=260\tau+8{,}7t_{CU}$.

Отже, операційний автомат $IM_p(\Gamma_3)$ має такі значення для своїх характеристик: продуктивність $E_p(\Gamma_3)=1{,}47$; витрати апаратури $O_p(\Gamma_3)=1{,}8$; швидкодія $LT_M(\Gamma_3)=260\tau+8{,}7t_{CU}$ і, нарешті, час циклу $\mathrm{T}_p(\Gamma_3)=30\tau$.

9.4 Проєктування ІМ-ОА з послідовною комбінаційною частиною

Комбінаційна частина автомата типу IM_S-OA містить три схеми $\Phi_1-\Phi_3$, які послідовно з'єднані (рис. 8.17). Ця організація комбінаційної частини вимагає формування складних мікрооперацій як суперпозицій

(послідовного з'єднання) типових мікрооперацій вихідної прошивки. Розглянемо метод синтезу автомата $IM_S(\Gamma_3)$. Як видно з цього позначення, схема автомата буде синтезована з використанням функціональної мікропрограми Γ_3 (рис. 8.19).

1. Формування пам'яті автомата. Цей етап виконується за тією ж методикою, як і для автоматів інших типів. Отже, пам'ять автомата $IM_S(\Gamma_3)$ складається з регістрів RA, RB, RC, RD, RS_1, RS_2.

2. Розподіл регістрів між шинами. Для розподілу треба сформувати дві множини A_1 і A_2, які відповідають зв'язкам між регістрами пам'яті і мультиплексорами M_1 і M_2.

Для автомата $IM_S(\Gamma_3)$ мультиплексор M_1 з'єднаний зі схемою Φ_1, яка виконує операцію порозрядної інверсії. Отже, регістри RA, RB і RC повинні бути включені в множину A_1. Відтак, регістри ОА $IM_S(\Gamma_3)$ розподіляються у такий спосіб: $A_1 = \{A,\ B,\ C,\ S_1,\ S_2\}$, $A_2 = \{A,\ B,\ D,\ S_2\}$.

3. Формування комплексних мікрооперацій. Структура $IM_S - OA$ дає змогу виконати послідовність таких трьох мікрооперацій: <інверсія, МО з двома операндами, зсув>. Саме послідовності мікрооперацій такого виду повинні бути сформовані за вихідної мікропрограми. У розглянутому прикладі (мікропрограма Γ_3) можна знайти такі послідовності МО: $S_1 = \langle y_5(b_2),\ y_8(b_3),\ y_{11}(b_4)\rangle$, $S_2 = \langle y_6(b_2),\ y_9(b_3),\ y_{12}(b_4)\rangle$, які охоплюють по три мікрооперації, і $S_3 = \langle y_{10}(b_3),\ y_{13}(b_4)\rangle$, $S_4 = \langle y_{15}(b_6),\ y_{17}(b_7)\rangle$, $S_5 = \langle y_{16}(b_7),\ y_{11}(b_8)\rangle$, які охоплюють по дві мікрооперації. Тут вираз $y_n(b_i)$ означає, що мікрооперація y_n записана у вершині b_i. Послідовність S_2 включає МО y_{12}, що зумовлено необхідністю використання додаткового операційного елемента в комбінаційній частині ОА. Цей елемент виконує зрушення на один розряд ліворуч. Введення цього елемента призводить до збільшення витрат

апаратури у схемі ОА, але збільшує швидкодію, порівняно з попереднім прикладом.

Отримані послідовності $S_1 - S_5$ відповідають комплексним мікрооперaціям $y_{19} \# S_1 := R_1\left(\overline{A} + D\right)$, $y_{20} \# S_2 := L_1\left(\overline{B} \oplus B\right)$, $y_{21} \# D := \overline{C} + D$, $y_{22} \# S_2 := \overline{C} + D$, $y_{23} \# S_1 := R_1\left(S_1 + A\right)$.

Побудуємо модифіковану множину мікрооперацій Y_0 автомата $IM_s - OA$. Для цього необхідно усунути з множини Y ті МО, які входять тільки в послідовності $S_1 - S_5$. Наприклад, мікрооперація y_5 входить тільки в S_1, і цю МО спостерігаємо тільки у вершині b_2. Відтак, ця мікрооперація ніколи не виконується самостійно і, отже, її необхідно виключити з множини Y. Водночас МО y_8 записана у вершині b_3, яка входить у послідовність S_1, і у вершині b_5, яка не входить у послідовності $S_1 - S_5$. Отже, мікрооперація y_8 повинна залишитися в множині Y. Застосування такого аналізу до всіх мікрооперацій прошивки дає змогу отримати множину Y_0 з мікрооперацій $y_1 - y_4$, $y_7 - y_9$, y_{12}, y_{14}, $y_{18} - y_{23}$, які виконуються комбінаційною частиною $IM_s - OA$.

4. Формування таблиці операторів. Як і раніше, ця таблиця відбиває порядок виконання мікрооперацій. Однак в разі $IM_s - OA$ таблиця містить мікрооперації $y_n \in Y_0$. Таблиця включає стовпці $y_n, A_1, A_2, A_3, A_4, Z, RS$ (табл. 9.4).

Як випливає з табл. 9.4, комбінаційна схема Φ_i потрібна тільки для передачі інформації, якщо мікрооперація y_n не використовує ресурси схеми Φ_i $(i = 1, 2, 3)$. У нашому ОА це відповідає, наприклад, безпосередньому зв'язку між шинами A_1 і A_3, який управляється оператором α_1. Для усунення аналогічного зв'язку в схемі Φ_2 використовується акумулятор, який виконує операцію $A_4 := A_1$ (для мікрооперації y_{14}). Оскільки інформація на шині A_2 відсутня, і за $\alpha_1 = 1$

виконується тотожність $A_3 \equiv A_1$, то маємо $\beta_1 \# A_4 := A_2 + A_3 = 0 + A_3 = A_1$. Такий підхід до виконання мікрооперації не приводить до збільшення часу циклу, оскільки цей параметр визначається на основі часів виконання найповільнішої мікрооперації в кожній з частин схеми ОА.

Таблиця 9.4

Таблиця операторів автомата $IM_s(\Gamma_3)$

y_n	A_1	A_2	A_3	A_4	Z	RS
y_1	–	–	–	–	$\langle d_1 \rangle\ \gamma_1$	$RA\ c_1$
y_2	–	–	–	–	$\langle d_2 \rangle\ \gamma_2$	$RB\ c_2$
y_3	–	–	–	–	$\langle d_3 \rangle\ \gamma_3$	$RC\ c_3$
y_4	–	–	–	–	$\langle d_4 \rangle\ \gamma_4$	$RD\ c_4$
y_7	$RA\ a_1$	$RB\ b_1$	$A_1\ \alpha_1$	$A_2 + A_3\ \beta_1$	$A_4\ \gamma_5$	$RC\ c_3$
y_8	$RS_1\ a_2$	$RD\ b_2$	$A_1\ \alpha_1$	$A_2 + A_3\ \beta_1$	$A_4\ \gamma_5$	$RS_1\ c_5$
y_9	$RS_2\ a_3$	$RB\ b_1$	$A_1\ \alpha_1$	$A_2 \oplus A_3\ \beta_2$	$A_4\ \gamma_5$	$RS_2\ c_6$
y_{12}	$RS_2\ a_3$	$RS_2\ b_3$	$A_1\ \alpha_1$	$A_2 + A_3\ \beta_1$	$A_4\ \gamma_5$	$RS_2\ c_6$
y_{14}	$RA\ a_1$	–	$A_1\ \alpha_1$	$A_2 + A_3\ \beta_1$	$R_1(A_4)\ \gamma_6$	$RS_1\ c_5$
y_{18}	$RC\ a_4$	$RB\ b_1$	$A_1\ \alpha_1$	$A_2 + A_3\ \beta_1$	$A_4\ \gamma_5$	$RD\ c_4$
y_{19}	$RA\ a_1$	$RD\ b_2$	$\overline{A}_1\ \alpha_2$	$A_2 + A_3\ \beta_1$	$R_1(A_4)\ \gamma_6$	$RS_1\ c_5$
y_{20}	$RB\ a_5$	$RB\ b_1$	$\overline{A}_1\ \alpha_2$	$A_2 \oplus A_3\ \beta_2$	$L_1(A_4)\ \gamma_7$	$RS_2\ c_6$
y_{21}	$RC\ a_4$	$RD\ b_2$	$\overline{A}_1\ \alpha_2$	$A_2 + A_3\ \beta_1$	$A_4\ \gamma_5$	$RD\ c_4$
y_{22}	$RC\ a_4$	$RD\ b_2$	$\overline{A}_1\ \alpha_2$	$A_2 + A_3\ \beta_1$	$A_4\ \gamma_5$	$RS_2\ c_6$
y_{23}	$RS_1\ a_2$	$RS_1\ b_4$	$A_1\ \alpha_1$	$A_2 + A_3\ \beta_1$	$R_1(A_4)\ \gamma_6$	$RS_1\ c_5$

5. Формування логічних умов. Цей етап виконується так само, як і для М-ОА. Схема формування логічних умов ψ пов'язана з шиною Z, отже, необхідна інформація повинна передаватися на цю шину. При цьому умова $x_1 \# A > 0$ обчислюється як $x_1 \# Z > 0$, а для формування умови $x_2 \# S_1 > S_2$ в схему ОА вводиться додатковий регістр RZ. Слово S_2 передається в цей регістр за $c_6 = 1$, що дає змогу обчислювати x_2 як таку функцію: $x_2 \# Z > RZ$.

6. Проєктування функціональної схеми автомата. Схема ОА синтезується з використанням правил, аналогічних правилам для автомата типу з паралельною комбінаційною частиною. Зазначимо, що схема автомата $IM_S(\Gamma_3)$ (рис. 9.14) охоплює додатковий зсувник на один розряд ліворуч (схема L_1), що дає змогу зменшити число вершин у перетвореній мікропрограмі.

7. Формування перетвореної прошивки. У разі ОА типу IM_S перетворена мікропрограма повинна включати тільки МО $y_n \in Y_0$, які представлені з використанням операторів $IM_S - OA$. Як і у випадках інших автоматів з узагальненими операціями, необхідно враховувати залежність мікрооперацій $y_n \in Y_0$ за даними. Для автомата $IM_S(\Gamma_3)$ перетворена мікропрограма $\Gamma_3(IM_S)$ показана на рис. 9.15.

Знайдемо основні характеристики автомата $IM_S(\Gamma_3)$, використовуючи формули (8.1)–(8.3). Перетворена мікропрограма $\Gamma_3(IM_S)$ має $O(\Gamma_3)=15$ операторних вершин, в яких сумарно записані $N(\Gamma_3)=22$ мікрооперації з вихідної прошивки Γ_3. Результат ділення 22 на 15 дає змогу знайти продуктивність автомата $IM_S(\Gamma_3)$, в нашому випадку це $E_S(\Gamma_3)=1{,}46$. Комбінаційна частина операційного автомата $IM_S(\Gamma_3)$ включає множник, множник по модулю 2, інвертор, зсувник праворуч, зсувник ліворуч і 5 мультиплексорів. Отже, апаратурні витрати для схеми автомата $IM_S(\Gamma_3)$ дорівнюють $O_S(\Gamma_3)=1{,}9$. Час циклу автомата $IM_S(\Gamma_3)$ визначається як результат додавання часів $4\tau_M$, τ_I, τ_S, τ_{Sh} і τ_R. Результат складання цих часів дає $T_S(\Gamma_3)=38\tau$. Як і для попередніх прикладів, мікропрограма $\Gamma_3(IM_S)$ має три шляхи, які вже раніше були знайдені. Загальна кількість вершин, через які проходять ці шляхи, дорівнює $n(\Gamma_3)=29$. Отже, середнє число вершин в кожному шляху визначається внаслідок ділення 29 на 3, тобто $n_S(\Gamma_3)=9{,}7$.

Використовуючи формулу (8.3), можна знайти значення швидкодії $LT_S(\Gamma_3) = 9{,}7(38\tau + t_{CU}) = 368{,}6\tau + 9{,}7t_{CU}$.

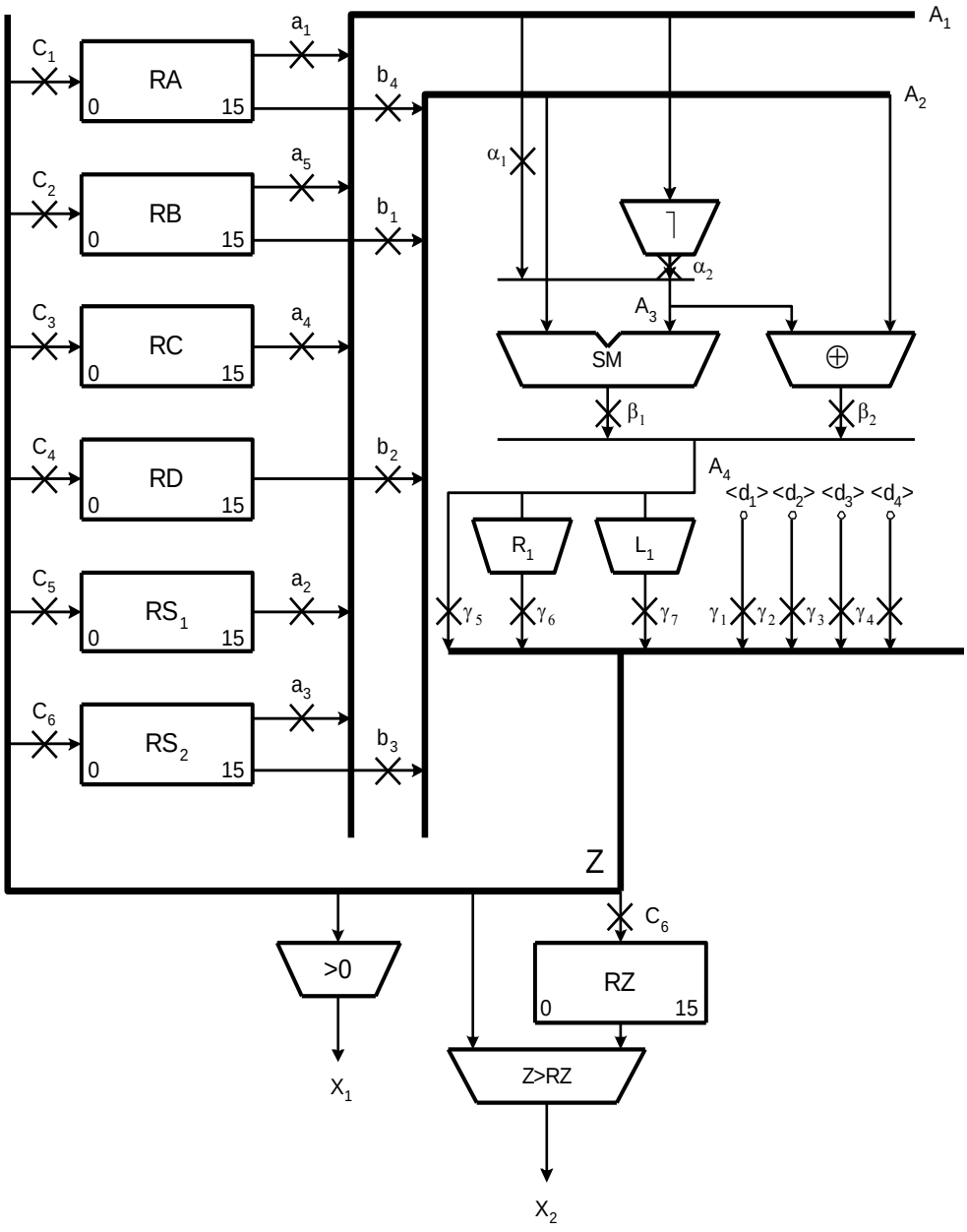


Рис. 9.14 – Функціональна схема автомата $IM_S(\Gamma_3)$

Для порівняння ефективності різних еквівалентних ОА треба знати їхні тимчасові характеристики. Нехай $t_{CU} = 10\tau$, тоді таблиця може бути побудована для порівняння автоматів, які реалізують алгоритм обчислень Γ_3 (табл. 9.5).

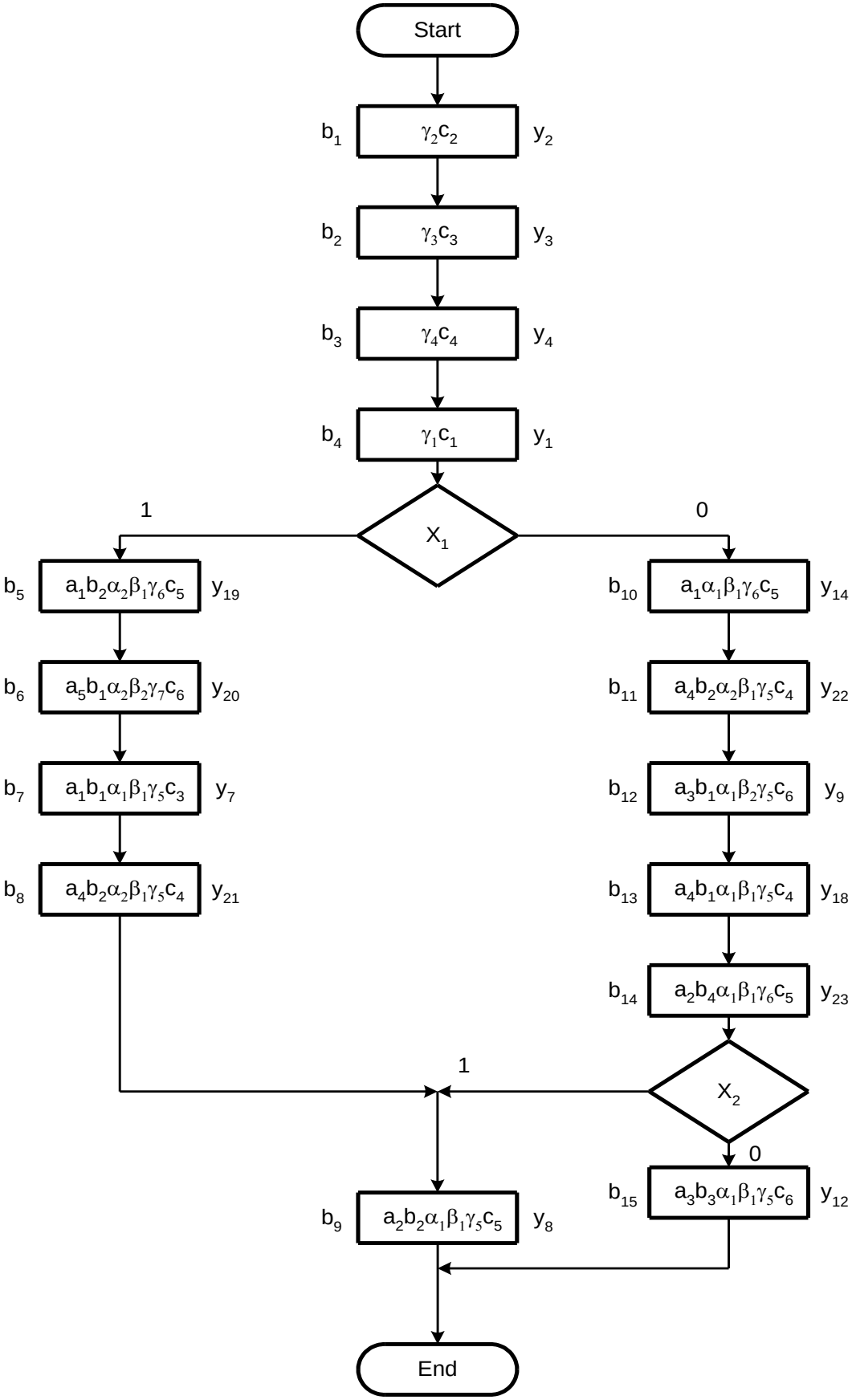


Рис. 9.15 – Перетворена мікропрограма $\Gamma_3(IM_s)$

Таблиця 9.5

Порівняння різних автоматів для прошивки Γ_3

Тип ОА	$E(\Gamma_3)$	$O(\Gamma_3)$	$T(\Gamma_3)$	$LT(\Gamma_3)$
$I(\Gamma_3)$	2,44	4,8	30τ	200τ
$M(\Gamma_3)$	1	1,6	30τ	504τ
$IM_p(\Gamma_3)$	1,47	1,8	30τ	374τ
$IM_s(\Gamma_3)$	1,46	1,9	38τ	425τ

Одним із найважливіших етапів проєктування цифрової апаратури загалом і операційних автоматів зокрема є етап вибору методу синтезу (або структури ОА), що дає змогу оптимізувати задані характеристики пристрою. Переважно, три основні практичні критерії використовуються для операційних автоматів:

1. Мінімальні апаратурні витрати. За виконання цієї умови схема ОА має мінімальну ціну (порівняно з іншими можливими рішеннями). Здебільшого, мінімальне значення відповідає М-ОА. Це підтверджується і в нашому прикладі, що видно з табл. 9.5. Причому, чим більша вартість суматора щодо вартості інших операційних елементів, тим ближча вартість еквівалентних автоматів типів $M-$, IM_p- і IM_s.

2. Максимальна швидкодія. Цей параметр аналогічний мінімуму часу для виконання алгоритму проведених обчислень. Здебільшого, максимум швидкодії забезпечує І-ОА. Однак це досягається за рахунок максимальної вартості автомата, порівняно з автоматами $M-$, IM_p- і IM_s. Як випливає з табл. 9.5, ціна автомата $I(\Gamma_3)$ в три рази більша, ніж автомата $M(\Gamma_3)$.

3. Максимальна ефективність. Під ефективністю будемо розуміти деяку інтегральну оцінку швидкодії і вартості ОА. Нехай вона визначається за формулою $e_i(\Gamma) = 1/(O_i(\Gamma)P_i(\Gamma))$, де індекс $i \in \{I,\ M,\ IM_p,\ IM_s\}$. Для нашого прикладу можна знайти:

$e_I(\Gamma_3) = 0{,}001$, $e_M(\Gamma_3) = 0{,}0012$, $e_P(\Gamma_3) = 0{,}0016$, $e_S(\Gamma_3) = 0{,}0012$. Аналіз цієї оцінки показує, що під час інтерпретації алгоритму Γ_3 найбільш ефективний $IM_p - OA$, а найгіршим буде I-OA.

Проєктування операційних пристроїв передбачає, що будуть розроблені схеми обох автоматів: операційного та керуючого. Отже, вибір оптимального рішення необхідно виконувати для всього операційного пристрою. Однак в цій книзі ми не розглядаємо методи синтезу пристроїв керування. Є багато книг, що детально розглядають ці питання.

Список літератури до розділу 9

1. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.

2. Майоров С. А., Новиков Г. И. Структура электронных вычислительных машин. Ленинград: Машиностроение, 1979. 384 с.

3. Adamski M., Barkalov A. Architectural and sequential synthesis of digital devices. Zielona Gora: University of Zielona Gora Press, 2006. 199 p.

4. Baranov S. Logic and system design of digital systems. Tallinn: TUT Press, 2008. 267 p.

5. Barkalov A. A., Titarenko L. A. Synthesis of operational and control automata. Donetsk: UNITECH, 2009. 256 p.

6. Gajski D. Principles of digital design. New Jersey: Prentice Hall, 1997. 318 p.

ВИСНОВКИ

Прикладна теорія цифрових автоматів знаходиться в постійному розвитку, що зумовлено насамперед успіхами мікроелектроніки. Сьогодні спостерігається інтенсивний розвиток методів синтезу цифрових пристроїв, орієнтованих на програмовані і замовні НВІС. Складність проєктованих систем зростає, як зростає і складність доступних для розробника НВІС. Ці схеми вже досягли складності в кілька мільярдів транзисторів, що не є межею.

Проєктування цифрових систем на основі таких мікросхем немислиме без застосування мов опису апаратури, систем автоматизованого проєктування, різних бібліотек. Однак все це не дає гарантії проєктування конкурентоспроможного продукту в прийнятний час. Для вирішення цієї проблеми необхідно, щоб розробник цифрової системи знав не тільки засоби, а й методи проєктування. Проєктування нових виробів, а також таких цифрових пристроїв, які будуть здатні конкурувати з найкращими аналогами, неможливе без фундаментальних знань в області теорії синтезу та оптимізації основних характеристик комбінаційних і послідовних схем.

Отже, щоб відповідати рівню сучасних складних проблем, фахівець в області інформатики повинен мати гарну базову підготовку. Автори сподіваються, що ця книга, яка містить класичні принципи проєктування цифрових пристроїв, допоможе у вирішенні завдань підготовки конкурентоспроможних фахівців в області розробки і застосування цифрових систем обробки інформації.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

Баркалов Олександр Олександрович

Титаренко Лариса Олександрівна

Баєв Артем Вікторович

Нескородєва Тетяна Василівна

ДИСКРЕТНИЙ АНАЛІЗ ПРИ ПРОЄКТУВАННІ І РЕАЛІЗАЦІЇ ЦИФРОВИХ АВТОМАТІВ

Навчальний посібник

Редактор А. О. Цяпало

Технічний редактор Т. О. Алимова

Підписано до друку 28.03.2022 р.
Формат 60x84/16. Папір офсетний.
Друк – цифровий. Умовн. друк. арк. 18,6
Тираж 100 прим. Зам. № 64

Донецький національний університет імені Василя Стуса,
21021, м. Вінниця, вул. 600-річчя, 21
Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру
серія ДК № 5945 від 15.01.2018 р.