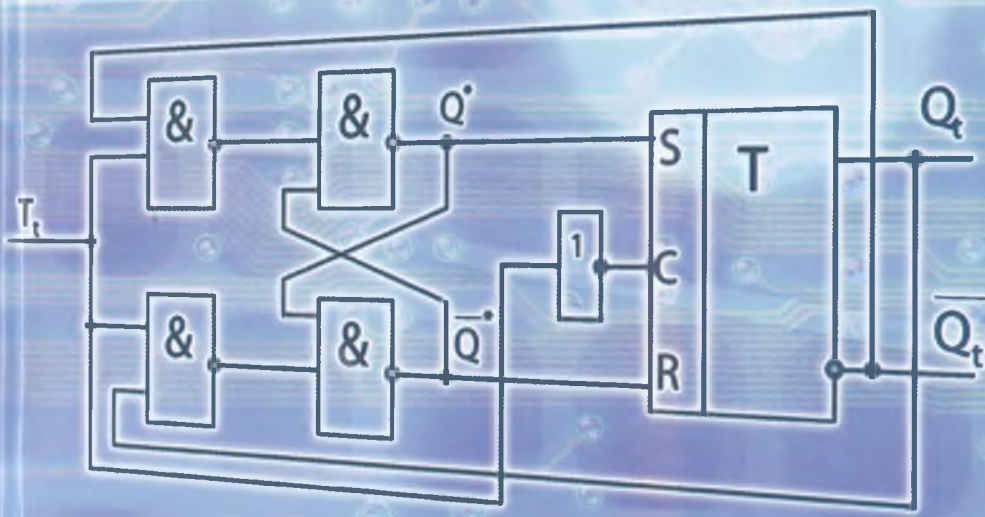
Матвієнко М.П.

КОМП'ЮТЕРНА ЛОГІКА



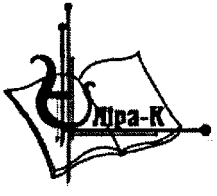
Ліра-К

МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ

Матвієнко М. П.

КОМП'ЮТЕРНА ЛОГІКА

Рекомендовано Міністерством освіти і науки, молоді та спорту України як навчальний посібник для студентів вищих навчальних закладів



Київ-2012

ББК 32.973
УДК 004.38
М 33

Рецензенти:

В.М. Міхайленко – завідувач кафедри математичних дисциплін Європейського університету, доктор технічних наук, професор
Г.С. Прокудін – доктор технічних наук, професор кафедри інформаційних систем і технологій Національного транспортного університету, м. Київ
М.М. Проценко – кандидат технічних наук, доцент кафедри комп'ютерних систем та мереж Національного Авіаційного університету, м. Київ.

Рекомендовано Міністерством освіти і науки, молоді та спорту України як навчальний посібник для студентів вищих навчальних закладів (лист Міністерства освіти і науки, молоді та спорту України №1/11-6538 від 22.07.2011 р.)

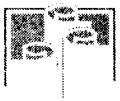
М 33 Матвієнко М. П. **Комп'ютерна логіка.** Навчальний посібник. — К.: Видавництво Ліра-К, 2012. — 288 с.
ISBN 966-2609-09-7

У навчальному посібнику викладено основні поняття комп'ютерної логіки і методи побудови різноманітних комп'ютерних схем та схем автоматики і управління. Теоретичний матеріал проілюстровано великою кількістю вправ та задач для набуття читачем практичного досвіду.

Навчальний посібник призначено для студентів, аспірантів і спеціалістів, які використовують відповідні математичні та комп'ютерні методи для побудови схем обчислювальної техніки і автоматики, а також окремі розділи посібника можуть бути використані студентами технічних навчальних закладів.

ББК 32.973
УДК 004.38

ISBN 966-2609-09-7



ЗМІСТ

ПЕРЕДМОВА

У навчальному посібнику приведені методи практичного застосування математичної логіки, теорії автоматів і регулярних подій у комп'ютерній інженерії, системах автоматики та управління.

Посібник ознайомлює читача з логікою арифметичних операцій у комп'ютерах, логіками Буля і Жегалкіна, логікою розкладання, дослідження та мінімізації булевих функцій, логікою часових і рекурентних булевих функцій, теорією автоматів та регулярних подій, а на їх основі і логікою побудови різноманітних комп'ютерних схем. Значна увага приділяється правильному застосуванню точних позначень, визначень, спрощеному доведенню теорем, логіці і алгоритмам побудови комп'ютерних схем.

Частина питань із розглянутих у посібнику недостатньо висвітлена в сучасній навчальній літературі. Зокрема, це стосується досліджень властивостей і повноти логічних функцій, перетворення автоматів із одного виду в інший, методів їх структурного синтезу, перетворення регулярних подій в абстрактні автомати, логіці і алгоритмам побудови різноманітних комп'ютерних схем.

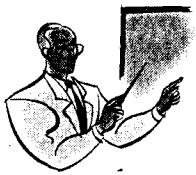
Усі визначення і теореми супроводжено ілюстрованими прикладами. Кожний параграф або розділ закінчується набором ретельно підібраних контрольних запитань та задач для самостійного розв'язування. Практичний матеріал є значним за обсягом і становить близько 40% загального матеріалу посібника. Це наведені приклади з рішеннями, контрольні запитання для закріплення та поглиблення розуміння теоретичних положень, задачі для самостійного розв'язування — їх практичне застосування дає можливість поліпшити організацію самостійної роботи студентів.

У навчальному посібнику зібрано як загальновідомий матеріал, так і розроблений останніми роками, а також подано і частково новий матеріал. Посібник розрахований, у першу чергу, на студентів, а також читачів, які бажають вивчити математичні методи комп'ю-

терної логіки і на їх основі навчитись будувати комп'ютерні схеми та схеми автоматики і управління. Від читача вимагаються знання в обсязі середньої школи, а всі подальші знання здобуваються в процесі роботи з книгою та запропонованими в ній вправами, що робить навчальний посібник привабливим для практичного використання.

Структура книги, яка складається із 13 розділів, їх перелік та наповнення випливає зі змісту. Вибір та викладання розділів комп'ютерної логіки виконано, враховуючи вимоги фундаментальної освіти з комп'ютерних дисциплін.

Посібник призначено для студентів спеціальностей вищих навчальних закладів III–IV рівнів акредитації базових напрямів «Комп'ютерна інженерія», «Комп'ютерні системи, автоматика і управління», «Комп'ютерні науки». Окремі розділи можуть бути використані також студентами відповідних середніх технічних навчальних закладів та коледжів.



Розділ 1

ЛОГІКА АРИФМЕТИЧНИХ ОПЕРАЦІЙ У КОМП'ЮТЕРАХ

1.1. Логіка систем числення

Означення 1.1.1. **Системою числення** називають систему відображення будь-яких чисел за допомогою обмеженого числа знаків.

Залежно від способів відображення чисел цифрами, системи числення діляться на позиційні і непозиційні.

Означення 1.1.2. **Непозиційною системою числення** називають систему, в якій кількісне значення кожної цифри не залежить від місця у відображенні числа, а визначається лише самим символом числа. Так, наприклад, число 30 десяткової системи числення в римській непозиційній системі позначають як число XXX, яке має у всіх розрядах один і той же самий символ X, що означає 10 одиниць незалежно від його позиції у відображенні числа.

Означення 1.1.3. **Позиційною системою числення** називають систему, в якій кількісне значення кожної цифри залежить від її місця у відображенні числа. Наприклад, число 575, представлене в десятковій системі числення, має в найстаршому і наймолодшому розрядах цифру 5. Цифра 5 у старшому розряді має вагу в 100 раз більшу, ніж у молодшому розряді.

У позиційній системі числення будь-яке число, яке має відображення

$$A_\alpha = \pm a_1 \cdot a_2 \cdot a_3 \ldots a_{k-1} \cdot a_k,$$

може бути представлене у вигляді такої суми

$$a_1 \cdot a_2 \cdot a_3 \ldots a_{k-1} \cdot a_k = a_1 \cdot q^{t-1} + a_2 \cdot q^{t-2} + a_3 \cdot q^{t-3} + \ldots + a_{k-1} \cdot q^{t-k+1} + a_k \cdot q^{t-k}, \quad (1.1.1)$$

де k — кінцева кількість розрядів у відображенні числа; a_i — цифра i-го розряду; q — основа системи числення; t — фіксоване число,

яке визначає положення коми; i — порядковий номер розряду; q^{l-i} — вага i-го розряду.

Цифри a_i повинні задовольняти нерівності $0 \le a_i \le q-1$.

Означення 1.1.4. **Основою системи числення** q називають кількість символів, які використовують для відображення числа в даній позиційній системі числення.

За основу системи числення q мають будь-яке число, яке задовольняє умові $q \ge 2$.

У двійковій системі числення $q = 2$ і для зображення чисел використовують символи (1, 0), у вісімковій $q = 8$ і для зображення чисел — символи (0, 1, 2, 3, 4, 5, 6, 7), а у шістнадцятковій — $q = 16$ і для зображення чисел символи — (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *E*, *F*), де $A = 10$; $B = 11$; $C = 12$; $D = 13$; $E = 14$; $F = 15$.

Двійкова система числення є основною системою числення, в якій виконують арифметичні і логічні операції в комп'ютерах, тому що для її технічної реалізації широке застосування знайшли двохпозиційні електронні елементи.

Суттєве значення при виконанні арифметичних та логічних операцій у комп'ютерах має переведення чисел із десяткової системи числення в двійкову, вісімкову та шістнадцяткову і навпаки. Для переведення цілих чисел найчастіше використовують алгоритм ділення заданого числа на основу числа, в систему якої його переводять. Даний алгоритм переведення чисел із системи числення з основою q є універсальним і найбільш широко застосовується на практиці. Він має такі кроки.

1. Розділити число, яке переводять, у системі числення з основою q на основу p за правилом системи числення з основою q.

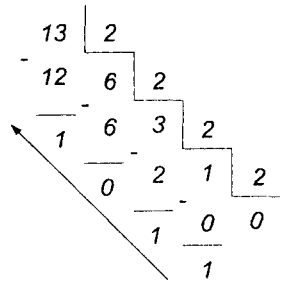
2. Перевірити, чи не дорівнює частка нулю. Якщо не дорівнює, то прийняти її за нове число й повернутися до кроку 1.

3. Якщо частка дорівнює нулю, то виписати всі отримані залишки від ділення в порядку, зворотному їх отриманню.

4. Отриманий запис є записом числа в системі числення з основою p.

Приклад 1.1.1. Перевести число $13_{(10)}$ десяткової системи числення у двійкову й виконати перевірку розв'язку.

Розв'язання. У відповідності з алгоритмом переведення цілих чисел ділимо послідовно число 13 десяткової системи числення на основу двійкової системи числення (число 2), в результаті чого отримаємо



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

Для перевірки, у відповідності з 1.1.1, записуємо формулу переведення чисел двійкової системи числення у десяткову

$$a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_1 \cdot 1^1 + a_0 \cdot 1^0 = A, \qquad (1.1.2)$$

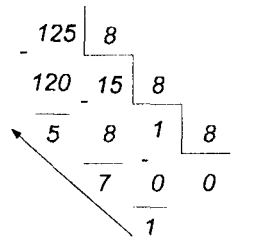
де a_n, a_{n-1},..., a_1, a_0 — цифри двійкового числа, які приймають значення 0 або 1;

A — ціле десяткове число.

Використовуючи формулу 1.1.2, отримаємо $1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + + 1 \cdot 2^0 = 13_{(10)}$

Приклад 1.1.2. Перевести число $125_{(10)}$ десяткової системи числення у вісімкову й виконати перевірку розв'язку.

Розв'язання. У відповідності з алгоритмом переведення цілих чисел послідовно ділимо число 125 десяткової системи числення на основу вісімкової системи числення (число 8), у результаті чого отримаємо



$$125_{(10)} = 175_{(8)}$$

Для перевірки застосуємо формулу 1.1.1 переведення числа із вісімкової системи числення у десяткову

$$b_n \cdot 8^n + b_{n-1} \cdot 8^{n-1} + \ldots + b_1 \cdot 8^1 + b_0 \cdot 8^0 = B, \qquad (1.1.3)$$

де b_n, b_{n-1}, …, b_1, b_0 — цифри числа вісімкової системи числення;

B — ціле десяткове число.

Використовуючи формулу 1.1.3, отримаємо $1 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 = 125_{(10)}$

Приклад 1.1.3. Перевести число $1822_{(10)}$ десяткової системи числення у шістнадцяткову і виконати перевірку розв'язку.

Розв'язання. У відповідності з алгоритмом переведення цілих чисел, послідовно ділимо число 1822 десяткової системи числення, на основу шістнадцяткової системи числення (число 16), у результаті чого отримаємо

1822	16		
1808	113	16	
14	112	7	16
	1	0	0
		7	

$$1822_{(10)} = 71E_{(16)}$$

Для перевірки застосуємо формулу 1.1.1 переведення числа із шістнадцяткової системи числення у десяткову

$$c_n \cdot 16^n + c_{n-1} \cdot 16^{n-1} + \ldots + c_1 \cdot 16^1 + c_0 \cdot 16^0 = C, \qquad (1.1.4)$$

де c_n, c_{n-1}, c_1, c_0 — цифри шістнадцяткової системи числення;

C — ціле десяткове число.

Використовуючи формулу 1.1.4, отримаємо

$$7 \cdot 16^2 + 1 \cdot 16^1 + 14 \cdot 16^0 = 1822_{(10)}.$$

Для переведення дробових чисел використовують алгоритм множення даного числа на основу числа, в систему якої його переводять. Даний алгоритм має такі кроки.

1. Помножити дробове число з основою q на основу системи числення p, у яку його переводять.

2. У кожному добутку виділити цілі частини, якщо такі є.

3. Виділені цілі частини добутку і є цифрами дробової частини числа.

Приклад 1.1.4. Перевести число $0{,}45_{(10)}$ десяткової системи числення у двійкову з точністю 10^{-4} й виконати перевірку розв'язку.

Розв'язання. Відповідно з алгоритмом переведення дробових чисел, послідовно чотири рази множимо дробове число 0,45 десяткової системи числення на основу двійкової системи числення (число 2), в результаті чого отримаємо

$$\begin{array}{cccc} 0{,}45 & 0{,}90 & 0{,}80 & 0{,}60 \\ \times \quad 2 & \times \quad 2 & \times \quad 2 & \times \quad 2 \\ \hline 0{,}90 & 1{,}80 & 1{,}60 & 1{,}20 \end{array}$$

$$0{,}45_{(10)} = 0{,}0111_{(2)}$$

Для перевірки застосуємо формулу 1.1.1 переведення дробових чисел двійкової системи числення у десяткові

$$a_1 \cdot 2^{-1} + a_2 \cdot 2^{-2} + \ldots + a_{n-1} \cdot 2^{-(n-1)} + a_n \cdot 2^{-n} = A, \quad (1.1.5)$$

де $a_1, a_2, \ldots a_{n-1}, a_n$ — цифри двійкового числа, які приймають значення 0 або 1;

A — дробове десяткове число.

Використовуючи формулу 1.1.5, отримаємо

$$0 \cdot 1^{-1} + 1 \cdot 1^{-2} + 1 \cdot 1^{-3} + 1 \cdot 1^{-4} = 0{,}4375_{(10)}$$

Неточність викликана обмеженням перетворення заданого десяткового дробового числа до четвертого знака.

Приклад 1.1.5. Перевести число $0{,}32_{(10)}$ десяткової системи числення у вісімкову з точністю 10^{-3} й виконати перевірку розв'язку.

Розв'язання. Відповідно з алгоритмом переведення дробових чисел, послідовно три рази множимо дробове число 0,32 десяткової системи числення на основу вісімкової системи числення (число 8), в результаті чого отримаємо

$$\begin{array}{ccc} 0,32 & 0,56 & 0,48 \\ \times \quad & \times \quad & \times \quad \\ \underline{\quad 8 \quad} & \underline{\quad 8 \quad} & \underline{\quad 8 \quad} \\ 2,56 & 4,48 & 3,48 \end{array}$$

$$0,32_{(10)} = 0,243_{(8)}.$$

Для перевірки застосуємо формулу 1.1.1 переведення дробових чисел із вісімкової системи числення у десяткову.

$$b_1 \cdot 8^{-1} + b_1 \cdot 8^{-2} + \ldots + b_{n-1} \cdot 8^{-(n-1)} + b_n \cdot 8^{-n} = B, \qquad (1.1.6)$$

де b_1, b_1, …, b_{n-1}, b_n — цифри вісімкового числа;

B — дробове десяткове число.

Використовуючи формулу 1.1.6, отримаємо $2 \cdot 8^{-1} + 4 \cdot 8^{-2} + 3 \cdot 8^{-3} = 0,3183_{(10)}$.

Приклад 1.1.6. Перевести число $0,64_{(10)}$ десяткової системи числення у шістнадцяткову з точністю до 10^{-3} й виконати перевірку розв'язку.

Розв'язання. Відповідно з алгоритмом переведення дробових чисел, послідовно три рази множимо дробове число 0,64 десяткової системи числення на основу шістнадцяткової системи числення (число 16), в результаті чого отримаємо

$$\begin{array}{ccc} 0,64 & 0,24 & 0,84 \\ \times \quad & \times \quad & \times \quad \\ \underline{\quad 16 \quad} & \underline{\quad 16 \quad} & \underline{\quad 16 \quad} \\ 10,24 & 3,84 & 13,44 \end{array}$$

$$0,64_{(10)} = 0,A3D_{(16)}.$$

Для перевірки застосуємо формулу 1.1.1 переведення дробових чисел із шістнадцятирічної системи числення в десяткову

$$c_1 \cdot 16^{-1} + c_1 \cdot 16^{-2} + \ldots + c_{n-1} \cdot 16^{-(n-1)} + c_n \cdot 16^{-n} = C, \quad (1.1.7)$$

де c_1, c_1, …, c_{n-1}, c_n — цифри шістнадцяткового числа;

C — дробове десяткове число.

Використовуючи формулу 1.1.7, отримаємо

$$10 \cdot 16^{-1} + 3 \cdot 16^{-2} + 13 \cdot 16^{-3} = 0,6398_{(10)}.$$

Для переведення змішаних чисел із десяткової системи числення в інші, необхідно виконати переведення цілої і дробової частин окремо згідно з їх алгоритмами, а отримані результати об'єднати.

Приклад 1.1.7. Перевести число $1822{,}64_{(10)}$ десяткової системи числення у шістнадцяткову з точністю 10^{-3}.

Розв'язання. Використовуючи дані переведення цілої частини числа (приклад 1.1.3) і дробової (приклад 1.1.6), отримаємо число в шістнадцятковій системі числення, яке матиме вигляд

$$1822{,}64_{(10)} = 71E{,}A3D_{(16)}.$$

Дуже часто при переведенні десяткових чисел у двійкові використовують проміжне переведення їх у вісімкову або шістнадцяткову систему числення з подальшим записом кожної їх цифри у вигляді трьох- або чотирьохзначного двійкового числа.

Приклад 1.1.8. Перевести число $257{,}36_{(8)}$ вісімкової системи числення у двійкову.

Розв'язання. Записуючи кожну цифру вісімкового числа у вигляді тріад двійкового числа, отримаємо

$$257{,}36_{(8)} = 010\ 101\ 111, 011\ 110_{(2)}.$$

Приклад 1.1.9. Перевести число $A7FD{,}Б5C_{(16)}$ шістнадцяткової системи числення у двійкову.

Розв'язання. Записуючи кожну цифру шістнадцяткової системи числення у вигляді тетрад двійкового числа, отримаємо

$$A7FD, Б5C_{(16)} = 1010\ \ 0111\ \ 1111\ \ 1101,\ 1011\ \ 0101\ \ 1100_{(2)}.$$

Переведення двійкових чисел у вісімкову систему числення відбувається таким чином. Задане двійкове число розподіляють на тріади справа наліво — для цілих чисел і зліва направо — для дробових чисел, а для змішаних — від коми, як уліво, так і вправо.

Переведення двійкових чисел у шістнадцяткову систему числення відбувається аналогічно, як і для вісімкової системи числення, але розподіл двійкового числа відбувається не на тріади, а на тетради.

Переведення двійкових чисел у двійково-десяткові відбувається шляхом поділу двійкового числа на тетради справа наліво для цілих чисел, а для змішаних від коми, як уліво, так і вправо.

1.2. Логіка арифметичних операцій над двійковими числами

В цьому параграфі розглядаються чотири арифметичні операції в двійковій системі числення (додавання, віднімання, множення і ділення).

Додавання. При додаванні двійкових чисел необхідно користуватись такими правилами

$$\text{а) } 0+0=0;\quad \text{б) } 0+1=1;\quad \text{в) } 1+0=1;\quad \text{г) } 1+1=10. \qquad (1.2.1)$$

Двозначна сума додавання в 1.2.1 г означає, що при додаванні двійкових цифр, які дорівнюють 1, виникає перенос 1 до сусіднього старшого розряду. Цей перенос повинен бути добавлений до суми цифр, які утворюються в сусідньому розряді зліва.

Приклад 1.2.1. Виконати додавання двох двійкових чисел $A_1 =$ 1101101 і $A_2 =$ 1001111.

Розв'язання. У відповідності з 1.2.1, виконуємо порозрядне додавання двійкових чисел A_1 і A_1

1101101	— A_1
1001111	— A_2
0100010	— порозрядна сума без урахування переносів
+	
1 11 1	— переноси
10111100	— $A_1 + A_2$

Безпосередньо під двома доданками записаний результат порозрядного додавання без урахування переносів. У тих розрядах, в яких обидва доданки дорівнюють 1, порозрядна сума дорівнює 0, і в цих розрядах утворилося перенесення в сусідній старший розряд, який відмічений у стрічці 1. У результаті додавання стрічки порозрядних сум із стрічкою переносів отримаємо кінцеву суму.

Віднімання. При відніманні двійкових чисел необхідно користуватись такими правилами:

$$\text{а) } 0-0=0;\quad \text{б) } 1-0=1;\quad \text{в) } 1-1=0;\quad \text{г) } 10-1=1. \qquad (1.2.2)$$

Крім цього, необхідно пам'ятати, що

$$1000\ldots0-1=111..1. \qquad (1.2.3)$$

n–нулів

Якщо при порозрядному відніманні необхідно відняти із нуля одиницю, то береться позичка в сусідньому розряді, тобто одиниця старшого розряду має вигляд двох одиниць даного розряду. Віднімання в цьому випадку виконується у відповідності з 1.2.2 г. Якщо в сусідньому старшому розряді або декількох старших розрядах містяться нулі, то позичка береться у найближчому старшому розряді з одиниці. Ця одиниця у відповідності з 1.2.3 має вигляд суми числа, яке складається із одиниць у всіх проміжних розрядах, в яких містились нулі, і двох одиниць у даному розряді. Віднімання у даному розряді виконується у відповідності з 1.2.2 г, а у всіх проміжних розрядах — згідно із 1.2.2 б і 1.2.2 в.

Приклад 1.2.2. Виконати віднімання двох двійкових чисел A_1 = = 11000011 і A_2 = 10100110.

Розв'язання. У відповідності з 1.2.2 виконаємо порозрядне віднімання двійкових чисел A_1 і A_2

$$\begin{array}{ll} 11000011 & -A_1 \\ - & \\ \underline{10100110} & -A_2 \\ 00011101 & -A_1 - A_2 \end{array}$$

У двох молодших розрядах не було необхідності робити позичку. Для третього розряду вона зроблена із сьомого розряду (найближча одиниця). В проміжних розрядах віднімання відбувається із одиниці.

Множення. Множення двох двійкових чисел виконується так само, як і множення двох десяткових. Тобто, множене послідовно множиться на кожну цифру множника, розпочинаючи з молодшої або із старшої. Для врахування ваги відповідної цифри множника результат рухається або вліво, якщо множення відбувається, починаючи з молодшого розряду множника, або вправо, якщо множення відбувається, починаючи із старшого розряду множника. При цьому рух відбувається на таке число розрядів, на яке відповідний розряд множника зсунутий відносно молодшого або старшого його розряду. Отримані в результаті множення і зсуву часткові добутки після додавання дають повний результат множення.

Приклад 1.2.3. Виконати множення двох двійкових чисел A_1 = = 10101 і A_2 = 1011 зсувом вліво і вправо.

Розв'язання. У відповідності з викладеними вище правилами множення, воно матиме такий вигляд для зсуву:

Вліво	Вправо
10101	10101
×	×
1011	1011
10101	10101
10101	10101
10101	10101
11100111	11100111

Якщо множене, або множник, або разом вони мають цілу і дробову частини, то множення таких чисел відбувається як множення цілих чисел, без урахування ком, а потім від отриманого добутку справа відділяється комою ($m + n$) від розрядів, де m — кількість дробових розрядів множеного, а n — кількість дробових розрядів множника.

Ділення. Ділення двох двійкових чисел відбувається аналогічно діленню двох десяткових. Спочатку розглянемо ділення цілого діленого на цілий дільник. Ділення розпочинається з того, що від цілого діленого зліва відділяється мінімально можлива група розрядів, яка дорівнює або перевищує численно дільник на один розряд. Якщо виділення такої групи можливо, то в старший розряд частки записують 1, якщо ні, то — 0.

Якщо виявилось, що частка має цілу частину, то утворюється нова група шляхом віднімання із виділеної групи дільника і приписування до різниці чергової цифри діленого. Якщо при цьому отримали число, яке перевищує дільник, то до частки записують наступну цифру, рівну 1, а якщо ні, то — 0.

У подальшому виконують ряд однакових циклів. Якщо остання цифра частки дорівнювала 1, то нова група утворюється шляхом віднімання дільника із старої групи і дописування наступної цифри діленого. Якщо в результаті отримали число, яке перевищує дільник, то до частки записують наступну цифру, яка дорівнює 1, а якщо ні, то – 0.

Якщо остання цифра частки дорівнює 1, то нова група утворюється шляхом віднімання дільника із старої групи і приписування наступної цифри діленого. Але якщо остання цифра частки дорівнює 0, то для утворення нової групи достатньо приписати до старої групи наступну цифру діленого.

Останню цифру цілої частини частки отримаємо тоді, коли після визначення чергової цифри частки (1 або 0) в діленому не зали-

шилося більше цифр для того, щоб приписувати їх до різниці між старою групою і дільником або до самої старої групи. Після цього розпочинається обчислення дробових членів частки. Воно відрізняється від обчислення цілих членів тільки тим, що замість чергових цифр дільника до старих груп приписують нулі.

Приклад 1.2.4. Виконати ділення двох двійкових чисел A_1 = = 110111 і A_2 = 10010 при умові, що в одному випадку діленим є A_1, а дільником A_2, а в іншому — навпаки

```
 110111 |10010          10010    |110111
-      |11                       |0,0101
  10010                  1001000
  10011                -
-                          110111
  10010                    01000100
      1 — залишок        -
                             110111
                              1101 — залишок
```

1.3. Логіка представлення двійкових чисел у прямому, додатковому, оберненому та модифікованому кодах

При виконанні арифметичних операцій у комп'ютерах числа кодуються спеціальними машинними кодами. Для цього використовують прямий, додатковий, обернений і модифікований коди, які дозволяють замінити операцію віднімання операцією додавання, що спрощує арифметико-логічний пристрій комп'ютера.

Прямий код заснований на представленні чисел у вигляді їх абсолютного значення з кодом знака плюс або мінус.

Формула для знаходження прямого коду двійкового числа $A_1 = 0$, $a_1\, a_1 \ldots a_n$ має такий вигляд:

$$A_n = \begin{cases} A, & \text{якщо } A \geq 0, \\ 1 - A, & \text{якщо } A \leq 0. \end{cases} \qquad (1.3.1)$$

Приклад 1.3.1. Записати двійкові числа A_1 = + 0,10110 і A_2 = = − 0,10110 в прямому коді.

Розв'язання. У відповідності з формулою 1.3.1, дані числа матимуть такий вигляд:

$$A_{1n} = 0{,}10110,\ A_{1n} = 1 - (-0{,}10110) = 1{,}10110.$$

Із формули 1.3.1 виходить, що нуль у прямому коді може бути як додатним, так і від'ємним

$$A = +0{,}00...00, \qquad A_n = 0{,}00...00,$$
$$A = -0{,}00...00, \qquad A_n = 1{,}00...00.$$

Додатковий код. Формула для зображення додаткового коду двійкового числа A має вигляд

$$A_д = \begin{cases} A, & \text{якщо } A \geq 0, \\ 10 + A, & \text{якщо } A < 0. \end{cases} \qquad (1.3.2)$$

Приклад 1.3.2. Записати двійкові числа $A_1 = +0{,}11010$ і $A_2 = 0{,}11010$ у додатковому коді.

Розв'язання. У відповідності з формулою 1.3.2 дані числа матимуть такий вигляд:

$$A_{1д} = 0{,}11010,\ A_{2д} = 10 + (-0{,}11010) = 1{,}00110.$$

Із формули 1.3.2 і прикладу випливає, що додатковий код додатного числа збігається із зображенням додатного числа прямого коду, а для зображення від'ємного числа в додатковому коді необхідно у знаковому розряді записати одиницю, а у всіх числових розрядах нулі замінити на одиниці, а одиниці – нулями і до отриманого результату додати одиницю до молодшого розряду.

В додатковому коді від'ємний нуль відсутній.

Обернений код. Формула для зображення оберненого коду двійкового числа A має такий вигляд:

$$A_0 = \begin{cases} A, & \text{якщо } A \geq 0, \\ 10 + A - 10^n, & \text{якщо } A \leq 0. \end{cases} \qquad (1.3.3)$$

Приклад 1.3.3. Записати двійкові числа $A_1 = +0{,}10110$ і $A_2 = 0{,}10110$ в оберненому коді.

Розв'язання. У відповідності з формулою 1.3.3 дані двійкові числа матимуть такий вигляд:

$$A_{1o} = 0{,}10110,\ A_{2o} = 10 - 0{,}10110 - 0{,}00001 = 1{,}01001.$$

Із формули 1.3.3 і прикладу випливає, що обернений код додатного числа збігається із зображенням додатного числа прямого ко-

ду, а для зображення від'ємного числа в оберненому коді необхідно у знаковому розряді записати одиницю, а у всіх числових розрядах нулі замінити на одиниці, а одиниці — нулями.

В оберненому коді відображення нуля неоднозначне і має такий вигляд:

$$A = +\,0{,}00\ldots00, \qquad A_o = 0{,}00\ldots00;$$
$$A = -\,0{,}00\ldots00, \qquad A_o = 1{,}11\ldots11.$$

Модифіковані коди. Ці коди використовують для виявлення переповнення розрядної сітки, яке може статися при додаванні двійкових чисел. Модифіковані коди відрізняються від простих машинних кодів тим, що на відображення знака в них відводяться два розряди. Плюс відображається двома нулями, а мінус — двома одиницями.

Переведення двійкових чисел у модифіковані прямі, додаткові та обернені коди відбувається за правилами наведеними в формулах 1.3.1, 1.3.2 і 1.3.3.

Приклад 1.3.4. Записати двійкові числа $A_1 = +\,0{,}11010$ і $A_2 = -\,0{,}11010$ в прямому, додатковому і оберненому модифікованих кодах.

Розв'язання. У відповідності з визначенням модифікованого коду, а також використовуючи формули 1.3.1, 1.3.2 і 1.3.3, дані коди матимуть такий вигляд:

$$A_{1n}^m = 00{,}11010,\ A_{1д}^m = 00{,}11010,\ A_{1o}^m = 00{,}11010,$$
$$A_{2n}^m = 11{,}11010,\ A_{2д}^m = 11{,}00110,\ A_{2o}^m = 11{,}00101.$$

1.4. Логіка додавання і віднімання двійкових чисел

Додавання чисел у модифікованому додатковому коді з фіксованою комою відбувається за правилами двійкової арифметики (§1.2). Одиниця переносу, яка виникає у старшому знаковому розряді суми, відкидається. Знаковим розрядом числа є другий зліва від коми розряд, а перший використовується для аналізу переповнення розрядної сітки.

Приклад 1.4.1. У модифікованому додатковому коді додати двійкові числа A_1 і A_2 при умові:

а) $A_1 > 0;\ A_2 > 0;\ A_1 + A_2 > 0;\quad A_1 = +0{,}1101;\ A_2 = +0{,}0001.$

$$\begin{array}{r} A^m_{1\partial} = 00{,}1101 \\ + \\ A^m_{2\partial} = 00{,}0001 \\ \hline A^m_{1\partial} + A^m_{2\partial} = 00{,}1110; \end{array} \qquad (A_1 + A_2)^m_{\text{п}} = 00{,}1110.$$

б) $A_1 > 0;\ A_2 < 0;\ A_1 + A_2 > 0;\quad A_1 = +0{,}1101;\ A_2 = -0{,}0001.$

$$\begin{array}{r} A^m_{1\partial} = 00{,}1101 \\ + \\ A^m_{2\partial} = 11{,}1111 \\ \hline A^m_{1\partial} + A^m_{2\partial} = 00{,}1100; \end{array} \qquad (A_1 + A_2)^m_{\text{п}} = 00{,}1100.$$

Одиниця переносу із старшого знакового розряду не враховується.

в) $A_1 < 0;\ A_2 > 0;\ A_1 + A_2 < 0;\quad A_1 = -0{,}1101;\ A_2 = +0{,}0001.$

$$\begin{array}{r} A^m_{1\partial} = 11{,}0011 \\ + \\ A^m_{2\partial} = 00{,}0001 \\ \hline A^m_{1\partial} + A^m_{2\partial} = 11{,}0100; \end{array} \qquad (A_1 + A_2)^m_{\text{п}} = 11{,}1100.$$

г) $A_1 < 0;\ A_2 < 0;\ A_1 + A_2 < 0;\quad A_1 = -0{,}1101;\ A_2 = -0{,}0001.$

$$\begin{array}{r} A^m_{1\partial} = 11{,}0011 \\ + \\ A^m_{2\partial} = 11{,}1111 \\ \hline A^m_{1\partial} + A^m_{2\partial} = 11{,}0010; \end{array} \qquad (A_1 + A_2)^m_{\text{п}} = 11{,}1110.$$

Одиниця переносу із старшого знакового розряду не враховується.

Додавання чисел у модифікованому оберненому коді з фіксованою комою виконується так, як і в додатковому коді. Різниця полягає тільки в тому, що одиницю переносу із старшого знакового розряду (якщо вона є) необхідно додати до молодшого розряду суми.

Приклад 1.4.2. В модифікованому оберненому коді додати двійкові числа A_1 і A_2 при умові :

а) $A_1 > 0;\ A_2 > 0;\ A_1 + A_2 > 0;\quad A_1 = +0{,}1101;\ A_2 = +0{,}0001.$

$$\begin{array}{r} A^m_{1o} = 00{,}1101 \\ + \\ A^m_{2o} = 00{,}0001 \\ \hline A^m_{1o} + A^m_{2o} = 00{,}1110; \end{array} \qquad (A_1 + A_2)^m_{\text{п}} = 00{,}1110.$$

б) $A_1 > 0; A_2 < 0; A_1 + A_2 > 0; A_1 = +0{,}1101; A_2 = -0{,}0001.$

$$\begin{array}{r} A_{1o}^{m} = \ 00{,}1101 \\ + \\ A_{2o}^{m} = \ 11{,}1110 \\ \hline A_{1o}^{m} + A_{2o}^{m} = 100{,}1011; \end{array}$$

$\longrightarrow +1$ — циклічний перенос

$$A_{1o}^{m} + A_{2o}^{m} = \ 00{,}1100; \qquad (A_1 + A_2)_{n}^{m} = 00{,}1100.$$

в) $A_1 < 0; A_2 > 0; A_1 + A_2 < 0; A_1 = -0{,}1101; A_2 = +0{,}0001.$

$$\begin{array}{r} A_{1o}^{m} = 11{,}0010 \\ + \\ A_{2o}^{m} = 00{,}0001 \\ \hline A_{1o}^{m} + A_{2o}^{m} = 11{,}0011; \end{array} \qquad (A_1 + A_2)_{n}^{m} = 11{,}1100.$$

г) $A_1 < 0; A_2 < 0; A_1 + A_2 < 0; A_1 = -0{,}1101; A_2 = -0{,}0001.$

$$\begin{array}{r} A_{1o}^{m} = 11{,}0010 \\ + \\ A_{2o}^{m} = 11{,}1110 \\ \hline A_{1o}^{m} + A_{2o}^{m} = 111{,}0000; \end{array}$$

$\longrightarrow +1$ — циклічний перенос

$$A_{1o}^{m} + A_{2o}^{m} = \ 11{,}0001; \qquad (A_1 + A_2)_{n}^{m} = 11{,}1110.$$

Переповнення розрядної сітки при додаванні в простих кодах. При додаванні двох двійкових чисел, які за абсолютною величиною менші одиниці, код суми за абсолютною величиною може бути більшим за одиницю або дорівнювати їй. У цьому випадку буде переповнення розрядної сітки, що приведе до спотворення результату додавання.

Приклад 1.4.3. *В додатковому коді додати двійкові числа* A_1 *і* A_2 при умові:

а) $A_1 > 0; A_2 > 0; A_1 + A_2 > 0; A_1 = +0{,}1101; A_2 = +0{,}0111.$

$$\begin{array}{r} A_{1\partial} = 0{,}1101 \\ + \\ A_{2\partial} = 0{,}0111 \\ \hline A_{1\partial} + A_{2\partial} = 1{,}0100; \end{array} \qquad (A_1 + A_2)_{n} = 1{,}1100.$$

б) $A_1 < 0; A_2 < 0; A_1 + A_2 < 0;\ \ A_1 = -0{,}1101; A_2 = -0{,}0111.$

$$\begin{array}{r} A_{1\partial} = \ 1{,}0011 \\ + \\ A_{2\partial} = \ 1{,}1001 \\ \hline A_{1\partial} + A_{2\partial} = 10{,}1100; \end{array} \qquad (A_1 + A_2)_{\text{п}} = 0{,}1100.$$

Як випливає із прикладу 1.4.3, додавання в додатковому двійковому простому коді привело до повного спотворення результату як за знаком, так і значенням числа через переповнення розрядної сітки. В першому випадку (1.4.3, *а*) переповнення розрядної сітки відбулося шляхом переносу в знаковий розряд при відсутності переносу із розряду знака, а в другому випадку (1.4.3, *б*) переповнення відбулося в знаковому розряді при відсутності його в розряді числа. Для уникнення таких спотворень і використовують модифіковані машинні коди.

Переповнення розрядної сітки при додаванні в модифікованих машинних кодах виявляють шляхом порівняння знакових розрядів отриманої суми. Дане твердження покажемо на такому прикладі.

Приклад 1.4.4. Виконати додавання двійкових чисел A_1 і A_2, заданих в умові прикладу 1.4.3, використовуючи модифікований машинний код.

Розв'язання. Використовуючи умови прикладу 1.4.3, а також представлення двійкових чисел у модифікованому коді, отримаємо:

а)
$$\begin{array}{r} A^m_{1\partial} = 00{,}1101 \\ + \\ A^m_{2\partial} = 00{,}0111 \\ \hline (A_1 + A_2)^m_{\partial} = 01{,}0100; \end{array}$$

б)
$$\begin{array}{r} A^m_{1\partial} = 11{,}0011 \\ + \\ A^m_{2\partial} = 11{,}1001 \\ \hline (A_1 + A_2)^m_{\partial} = 110{,}1100; \end{array}$$

↑ зникає

Як випливає із прикладу 1.4.4, в знакових розрядах отриманої суми додатних доданків маємо комбінацію «01», а від'ємних — «10», що є ознакою переповнення розрядної сітки.

Додавання і віднімання двійкових чисел із плаваючою комою відбувається за три кроки. На першому кроці необхідно виконати **вирівнювання порядків** чисел. У будь-якому числі з плаваючою комою вага N_i одиниці i-го розряду мантиси визначається не тільки позицією даного розряду, але і порядком p числа, тобто $N_i \quad 2^{p-i}$, де i — номер позиції, рахуючи вправо від коми.

При додаванні мантис необхідно, щоб вага одиниць однойменних розрядів мантиси чисел була однаковою. Для цього мантиси переміщуються відносно одна одної так, щоб їх порядки стали рівними. При вирівнюванні порядків отримувати мантиси більше одиниці не дозволяється, і тоді їх необхідно рухати у бік більшого порядку. Мантиси з меншим порядком рухаються вправо на кількість розрядів, яке дорівнює різниці порядків.

На другому кроці відбувається **додавання мантис** із вирівняними порядками, яке відбувається аналогічно додаванню чисел із фіксованою комою. Для додавання від'ємних мантис використовують додатковий або обернений модифіковані двійкові коди. Сума мантис є сумою чисел. Порядок суми чисел дорівнює спільному порядку доданків, тобто порядку більшого числа.

На третьому кроці відбувається **нормалізація отриманого результату**. При додаванні мантис за абсолютною величиною менших одиниці можливі випадки переповнення розрядної сітки — порушення нормалізації зліва від коми. Ознакою такого порушення нормалізації є поява в знакових розрядах модифікованого коду суми різних цифр (наприклад, 01,101... або 10,101...). Для встановлення нормалізації мантиса суми рухається на один розряд уліво, а порядок суми збільшується на одиницю.

При додаванні мантис можливі випадки отримання результату, меншого 0,1...(двійкове число), тобто порушення нормалізації справа від коми.

Виявом такого порушення є значення в мантисі суми одного або підряд декількох нулів справа від коми. При нормалізації вліво мантиса рухається вправо на таку кількість розрядів, щоб старша цифра мантиси була відмінна від нуля. Порядок суми зменшується на кількість одиниць, рівну числу рухів мантиси при нормалізації. Якщо всі розряди мантиси суми дорівнюють нулю, то нормалізація не відбувається, а порядок суми дорівнює нулю.

Приклад 1.4.5. Використовуючи плаваючу кому, додати два двійкові числа:

$$A_1 = +0{,}10100 \cdot 10^{+101} \text{ і } A_2 = -0{,}10110 \cdot 10^{+100}.$$

Розв'язання. Розв'язання виконаємо за три кроки. На першому кроці для вирівнювання порядків доданків необхідно із порядку числа A_1 відняти порядок числа A_2. Віднімання замінюємо додаванням порядків у модифікованому додатковому коді:

$$\begin{array}{r} P^m_{A_{1\partial}} = 00{,}101 \\ + \\ P^m_{A_{2\partial}} = 11{,}100 \\ \hline (P_{A_1} + P_{A_2})^m_{\partial} = 00{,}001; \end{array}$$

Так як P_{A_1} на одиницю більше P_{A_2}, то рухаємо мантису числа A_2 вираво на один розряд, тобто $M^m_{A_{2пр.}} = 1{,}10110 \rightarrow 1{,}01011$.

На другому кроці додаємо мантиси чисел A_1 і A_2 в модифікованому додатковому коді:

$$\begin{array}{r} M^m_{A_{1\partial}} = 00{,}10100 \\ + \\ M^m_{A_{2\partial}} = 11{,}10101 \\ \hline \left(M^m_{A_{1\partial}} + M^m_{A_{2\partial}}\right)^m_{\partial} = 00{,}01001. \end{array}$$

На третьому кроці нормалізуємо результат $0{,}01001 \cdot 10^{+101}$. Для цього рухаємо мантису суми на один розряд вліво і із значення порядку суми віднімаємо одиницю. В результаті цього отримаємо нормалізований результат додавання чисел $A_1 + A_2 = -0{,}1001 \cdot 10^{+100}$.

Приклад 1.4.6. Використовуючи плаваючу кому, додати два двійкових числа $A_1 = +0{,}10100 \cdot 10^{+011}$ і $A_2 = +0{,}11100 \cdot 10^{+101}$.

Розв'язання. Вирівнюємо порядки доданків

$$\begin{array}{r} P^m_{A_{1\partial}} = 00{,}011 \\ + \\ P^m_{A_{2\partial}} = 11{,}011 \\ \hline (P_{A_1} + P_{A_2})^m_{\partial} = 11{,}110; \quad (P_{A_1} + P_{A_2})^m_{n} = 11{,}010. \end{array}$$

Як і в попередньому прикладі, для віднімання використаємо модифікований код. Так як P_{A_1} на дві одиниці менше P_{A_2}, то рухаємо вираво мантису $M_{A_{1пр.}} = 0{,}10100 \rightarrow 0{,}00101$. Тепер додаємо мантиси чисел A_1 і A_2 та отримаємо

$$M^m_{A_{1\partial}} = 00\,00101$$

$$+$$

$$M^m_{A_{2\partial}} =$$

$$(M_{A_1} + M_{A_2})^n_\partial =$$

Нормалізуємо отриманий результат додавання чисел $A_1 + A_2 = 01{,}00001 \cdot 10^{+101}$, для чого рухаємо мантису на один розряд вправо і збільшуємо порядок числа на одиницю. В результаті цього отриманий результат матиме слідуючий вигляд $A_1 + A_2 = 0{,}100001 \cdot 10^{+110}$.

Контрольні запитання

1. Які системи числення застосовують у комп'ютерах?
2. Яку систему числення називають двійковою, вісімковою, шістнадцятковою?
3. Як переводять цілі і дробові числа із десяткової системи числення у двійкову і навпаки?
4. Як переводять цілі і дробові числа з десяткової системи числення у вісімкову і навпаки?
5. Як переводять цілі і дробові числа з десяткової системи числення у шістнадцяткову і навпаки?
6. Чим відрізняються між собою двійкова, вісімкова і шістнадцяткова системи числення і в яких частинах комп'ютера вони застосовуються?
7. Які арифметичні операції можна виконувати у двійковій системі числення?
8. Що таке прямий, додатковий і обернений коди?
9. Для яких цілей застосовують додатковий і обернений коди?
10. Яка різниця між додатковим і оберненим кодом?
11. Що таке модифікований код і для яких цілей його застосовують?
12. Що називають мантисою і порядком у двійковому числі?
13. Що таке переповнення розрядної сітки при додаванні в модифікованих машинних кодах?
14. Яка різниця в додаванні і відніманні двійкових чисел з фіксованою і плаваючою комою?
15. Що називають нормалізацією числа у двійковому коді і для яких цілей її застосовують?
16. Чому для додавання і віднімання двійкових чисел у комп'ютерах застосовують плаваючу кому?

Задачі для самостійного розв'язування

1. Перевести числа 117 і 0,117 з десяткової у двійкову системи числення з точністю 10^{-4} й виконати перевірку.

2. Перевести числа 411 і 0,411 з десяткової у вісімкову системи числення з точністю 10^{-5} й виконати перевірку.

3. Перевести числа 1015 і 0, 1015 з десяткової в шістнадцяткову системи числення з точністю 10^{-5} й зробити перевірку.

4. Перевести число 30,156 з десяткової в двійкову систему числення з точністю 10^{-6} й виконати перевірку.

5. Перевести число 97,11 з десяткової у вісімкову систему числення з точністю 10^{-4} й зробити перевірку.

6. Перевести число 1115,48 з десяткової в шістнадцяткову систему числення з точністю 10^{-3} і виконати перевірку.

7. Виконати додавання трьох двійкових чисел $A_1 = 101001$, $A_2 = 111001$, $A_3 = 100101$.

8. Виконати віднімання двох двійкових чисел $A_1 = 110010$, $A_2 = 101101$.

9. Виконати множення і ділення двох двійкових чисел $A_1 = 11010$, $A_2 = 10101$.

10. Представити двійкові числа $A_1 = +\,0{,}10011$ і $A_2 = -\,0{,}01100$ у прямому, додатковому і оберненому кодах.

11. Представити двійкові числа, наведені в задачі 10, у модифікованому прямому, додатковому і оберненому кодах.

12. Виконати додавання двійкових чисел $A_1 = 0{,}01101$ і $A_2 = 0{,}10111$ у модифікованому додатковому коді при умові :

а) $A_1 > 0$; $A_2 > 0$; $A_1 + A_2 > 0$;

б) $A_1 > 0$; $A_2 < 0$; $A_1 + A_2 < 0$;

в) $A_1 < 0$; $A_2 > 0$; $A_1 + A_2 > 0$;

г) $A_1 < 0$; $A_2 < 0$; $A_1 + A_2 < 0$.

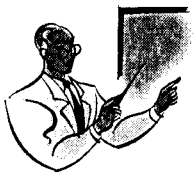
13. Виконати додавання двійкових чисел, наведених в задачі 12, у модифікованому оберненому коді.

14. Використовуючи плаваючу кому, виконати додавання і нормалізацію двійкових чисел $A_1 = +\,0{,}010101 \cdot 10^{+101}$ і $A_2 = 0{,}101101 \cdot 10^{+100}$

15. Використовуючи плаваючу кому, виконати додавання і нормалізацію двійкових чисел $A_1 = -0{,}1010101 \cdot 10^{+110}$ і $A_2 = -0{,}0110110 \cdot 10^{+111}$.

Коментарі

У даному розділі логіка систем числення взята із [25], логіка арифметичних операцій випливає із [22], а логіка представлення двійкових чисел, їх додавання і віднімання взяті з [22, 25].



Розділ 2

ЛОГІКА БУЛЯ

2.1. Основні визначення

Означення 2.1.1. Функцію $f(x_1, x_2, \ldots, x_n)$ називають **логічною (булевою)**, якщо вона також, як і її змінні $<x_1, x_2, \ldots, x_n>$, приймає лише два значення 0 і 1, які називають логічними константами.

Означення 2.1.2. Дві логічні функції $f_1(x_1, x_2, \ldots, x_n)$ і $f_2(x_1, x_2, \ldots, x_n)$ називають **рівними**, якщо вони приймають однакові значення на всіх можливих наборах їх змінних, виконуючи при цьому умову

$$f_1(x_1, x_2, \ldots, x_n) = f_2(x_1, x_2, \ldots, x_n).$$

Означення 2.1.3. Змінну x_i у логічній функції називають **істотною**, якщо для неї виконується умова

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \neq f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

Означення 2.1.4. Змінну x_i у логічній функції називають **неістотною (фіктивною)**, якщо

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

при будь-яких значеннях решти змінних, тобто якщо значення змінної x_i у будь-якому наборі не змінює значення функції. В цьому випадку функція $f(x_1, x_2, \ldots, x_n)$ фактично залежить від $n-1$ змінної.

Значення логічної функції може бути задано не на всіх можливих наборах її змінних. На деяких наборах значення логічної функції можуть бути неповністю визначені. Такі логічні функції називають **неповністю визначеними**, або **недовизначеними**.

Означення 2.1.5. **Інтерпретацією** логічної функції $f(x_1, x_2, \ldots, x_n)$ називають конкретне (індивідуальне) значення логічного набору $< x_1, x_2, \ldots, x_n >$.

Множину всіх двійкових слів позначають через A^n і називають ***n*-вимірним логічним кубом**, який містить 2^n елементів слів $|A^n| = 2^n$.

Наприклад, для $n = 1$ є всього $2^1 = 2$ слова — (0) і (1), а за рахунок збільшення довжини слова на один символ кількість слів збільшується в два рази. Для наочності зобразимо в одному рядку набори логічних змінних зростаючої довжини, а в іншому — кількість різних наборів відповідної довжини

$$\langle x_1 \rangle, \ \langle x_1, x_2 \rangle, \ \langle x_1, x_2, x_3 \rangle, \ \ldots, \ \langle x_1, x_2, \ldots, x_n \rangle$$
$$2^1, \quad 2 \cdot 2 = 2^2, 2 \cdot 2 \cdot 2 = 2^3, \ \ldots, \ 2 \cdot 2 \cdot \ldots \cdot 2 = 2^n.$$

Тобто, кількість логічних функцій, які можна отримати з набору змінних $\langle x_1, x_2, \ldots, x_n \rangle$, дорівнює 2^n. Але кожна логічна функція може приймати два значення 0 і 1, тоді число всіх можливих функцій буде дорівнювати 2^{2^n}.

2.2. Способи задання логічних функцій

Логічні функції можуть бути задані такими способами:

1. Таблицею істинності;
2. Порядковим номером, який має ця функція;
3. Аналітично (у вигляді формули).

Таблицею істинності називають таблицю, в якій кожну інтерпретацію логічної функції поставлено у відповідність її значення.

Форма таблиці істинності приведена на рис. 2.2.1

x_1	x_2	....	x_n	$f(x_1, x_2, \ldots, x_n)$
0	0	0	0	0
1	0	0	0	.....
0	1	0	0	.....
....	....	....	....	.....
1	1	1	1	1

Рис. 2.2.1

В таблиці істинності, рис. 2.2.1, кожній змінній та значенню самої функції відповідає по одному стовпчику, а кожній інтерпретації — по одному рядку. Кількість рядків таблиці відповідає кількості різних інтерпретацій логічної функції. Логічні функції $\varphi(x)$, які залежать від однієї змінної, наведені в табл. 2.2.1

Таблиця 2.2.1

x	φ_0	φ_1	φ_2	φ_3
0	0	0	1	1
1	0	1	0	1

Кожній функції, відповідно до її значень, можна приписати такі назви:

$\varphi_0 = 0$ — функція константи 0;

$\varphi_1 = x$ — функція повторення змінної;

$\varphi_2 = \overline{x}$ — функція інверсії або заперечення змінної;

$\varphi_3 = 1$ — функція константи 1.

Для двох змінних $f(x, y)$ різні логічні функції приведені в табл. 2.2.2.

Таблиця 2.2.2

x	y	f_0	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}
0	0	0	0	0	0	0	0	0	0	1	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

Їх кількість дорівнює $2^{2^2} = 16$. Більшість із цих функцій використовують у математиці, програмуванні і техніці, тому їх позначення, назви і прочитання приведені в табл. 2.2.3

Таблиця 2.2.3

Функція	Позначення	Назва	Прочитання
$f_0(x, y)$	0	константа 0	константа 0
$f_1(x, y)$	$x \,\&\, y = x \wedge y = x \cdot y$	кон'юнкція (логічне «і»)	x і y

Закінчення табл. 2.2.3

Функція	Позначення	Назва	Прочитання
$f_2(x, y)$	$x \Delta y$	заборона або заперечення імплікації	x і не y
$f_3(x, y)$	x	повторення першого аргументу	як x
$f_4(x, y)$	$y \Delta x$	заборона або заперечення оберненої імплікації	y і не x
$f_5(x, y)$	y	повторення другого елемента	як y
$f_6(x, y)$	$x \oplus y$	що виключає «або» (сума за модулем 2)	x не як y
$f_8(x, y)$	$x \downarrow y$	стрілка Пірса (заперечення диз'юнкції)	не x і не y
$f_9(x, y)$	$x \sim y$	еквівалентність	x як y
$f_{10}(x, y)$	$\overline{y}$	заперечення другого елемента	не y
$f_{11}(x, y)$	$y \to x$	обернена імплікація	x, якщо y (x або не y)
$f_{12}(x, y)$	$\overline{x}$	заперечення першого елемента	не x
$f_{13}(x, y)$	$x \to y$	імплікація	якщо x, то y (не x або y)
$f_{14}(x, y)$	$x \mid y$	функція Шеффера (заперечення кон'юнкції)	не x або не y
$f_{15}(x, y)$	1	константа 1	константа 1

При заданні логічної функції **порядковим номером** кожній функції привласнюють порядковий номер у вигляді натурального числа, двійковий код якого зображує стовпчик значень функції у таблиці істинності. Вказаний порядковий номер функції, як двійковий так і десятковий, повністю визначає логічну функцію.

Приклад 2.2.1. Знайти порядковий номер функції $f(x, y)$, що приймає такі значення : $f(0, 0) = 1, f(0, 1) = 1, f(1, 0) = 0, f(1, 1) = 1$.

Розв'язання. Будуємо таблицю істинності для заданої функції (x, y), табл. 2.2.4.

Таблиця 2.2.4

x	y	$f(x, y)$
0	0	1
0	1	1
1	0	0
1	1	1

Як випливає з табл. 2.2.4, двійковий код, що відповідає значенням функції, дорівнює 1101. Перевівши двійкове число 1101_2 у десяткову систему числення, отримаємо :

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13_{10}$$

Таким чином, цьому числу відповідає розглянута функція імплікації

$f_{13}(x, y) = x \rightarrow y$ (табл. 2.2.2 і табл. 2.2.3).

Приклад 2.2.2. Побудувати таблицю істинності для логічної функції з порядковим номером 14.

Розв'язання. Знайдемо двійкове число, що відповідає десятковому числу 14, представимо його, як суму степенів числа 2

$14_{(10)} = 8 + 4 + 2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 1110_{(2)}$.

Будуємо шукану таблицю істинності, розташувавши отримане число у стовпчику значення функції таким чином, щоб молодший розряд виявився у нижчому рядку, табл. 2.2.5.

Таблиця 2.2.5

x	y	$f_{14}(x, y)$
0	0	1
0	1	1
1	0	1
1	1	0

Таким чином, функції Шеффера $f_{14}(x, y) = x \mid y$ відповідає таблиця істинності, табл. 2.2.5.

Логічні функції можуть бути задані і **аналітично**, тобто **формулами**. Наприклад, задана логічна функція $f(x, y, z)$ формулою

$$f(x, y, z) = x \wedge \bar{y} \vee z.$$

Якщо у формулі відсутні дужки, то операції необхідно виконувати у такій послідовності: заперечення, кон'юнкція, диз'юнкція, імплікація та еквівалентність

$$< \bar{}, \wedge, \vee, \rightarrow, \sim >.$$

Приклад 2.2.3. У заданій логічній функції $f(x, y, z) = x \sim y \rightarrow$ $\ldots \vee \ldots$ розставити дужки.

Розв'язання. Враховуючи пріоритет виконання логічних операцій, розставимо дужки для заданої функції

$$f(x, y, z) = x \sim (y \rightarrow (z \vee \bar{x})).$$

Приклад 2.2.4. У заданій логічній функції $f(x, y, z) = z \rightarrow \overline{x \wedge y \vee z} \vee x$ розставити дужки.

Розв'язання. Враховуючи пріоритет виконання логічних операцій, розставимо дужки для заданої функції

$$f(x, y, z) = z \rightarrow (\overline{((x \wedge y) \vee z)} \vee x).$$

Перехід від заданої формули логічної функції до таблиці істинності виконують таким чином. Установлюють значення функції на всіх її інтерпретаціях. Потім розміщують в таблиці істинності інтерпретації в порядку збільшення відповідних їм двійкових номерів і записують отримані значення функції на кожній інтерпретації.

Приклад 2.2.5. Побудувати таблицю істинності для функції

$$f(x, y, z) = (x \vee y) \rightarrow z.$$

Розв'язання. Функція залежить від трьох змінних і, отже, для неї є $2^3 = 8$ інтерпретацій. Установлюємо значення функції на всіх інтерпретаціях:

$$f(0, 0, 0) = (0 \vee 0) \rightarrow 0 = \bar{0} \vee 0 = 1,$$
$$f(0, 0, 1) = (0 \vee 0) \rightarrow 1 = \bar{0} \vee 1 = 1,$$
$$f(0, 1, 0) = (0 \vee 1) \rightarrow 0 = \bar{1} \vee 0 = 0,$$
$$f(0, 1, 1) = (0 \vee 1) \rightarrow 1 = \bar{1} \vee 1 = 1,$$
$$f(1, 0, 0) = (1 \vee 0) \rightarrow 0 = \bar{1} \vee 0 = 0,$$
$$f(1, 0, 1) = (1 \vee 0) \rightarrow 1 = \bar{1} \vee 1 = 1,$$
$$f(1, 1, 0) = (1 \vee 1) \rightarrow 0 = \bar{1} \vee 0 = 0,$$
$$f(1, 1, 1) = (1 \vee 1) \rightarrow 1 = \bar{1} \vee 1 = 1.$$

Отримані значення функцій, у відповідності з їх інтерпретаціями, заносимо в таблицю істинності, табл. 2.2.6.

Таблиця 2.2.6

x	y	z	$f(x, y, z) = (x \vee y) \rightarrow z$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

2.3. Елементарні логічні функції

До елементарних логічних функцій, розглянутих у табл. 2.2.3, належать дванадцять функцій, які відіграють суттєву роль у теорії функцій алгебри логіки, а також їх практичному застосуванні при побудові різних електронних пристроїв і систем.

Функцію $f_0(x, y) = 0$ називають **константою нуль.** Вона завжди дорівнює «0».

Функцію $f_1(x, y) = x \& y$ називають **кон'юнкцією** або **логічним множенням**. Ця функція приймає значення «1» тоді і тільки тоді, коли логічні змінні x і y приймають значення «1» одночасно. Для її позначення використовують символи $< \&, \wedge, \cdot >$.

Функцію $f_2(x, y) = x \Delta y$ називають **забороною** за y або **запереченням імплікації**. Логічна функція заборони за y приймає логічне значення «1» тоді і тільки тоді, коли логічна змінна x приймає значення «1», а логічна змінна y – «0». Для її позначення використовують символи $< \Delta, \leftarrow >$.

Функцію $f_3(x, y) = x$ називають **повторенням** першого аргументу і позначають x.

Функцію $f_6(x, y) = x \oplus y$ називають **виключаючим «або»**, **сумою за модулем два** або функцією **нерівнозначності**. Ця функція приймає значення «1» тоді і тільки тоді, коли значення логічної

змінної x дорівнює «1», а y = «0» і навпаки. Для її позначення застосовують символи $<\oplus, \neq>$.

Функцію $f_7(x, y) = x \vee y$ називають **диз'юнкцією** або **логічним додаванням**. Ця функція приймає значення «1» тоді і тільки тоді, коли хоча б одна змінна приймає значення «1». Для її позначення застосовують символи $<\vee, +>$.

Функцію $f_8(x, y) = x \downarrow y$ називають **стрілкою Пірса** або **запереченням диз'юнкції**. Ця функція приймає значення «1» тоді і тільки тоді, коли логічні змінні x і y приймають значення «0» одночасно. Для її позначення використовують символ $<\downarrow>$.

Функцію $f_9(x, y) = x \sim y$ називають **еквівалентністю** або **рівнозначністю**. Ця функція приймає значення «1» тоді і тільки тоді, коли логічні змінні x і y приймають значення «0» або «1» одночасно. Для її позначення використовують символи $<\sim, \leftrightarrow, \equiv>$.

Функцію $f_{12}(x, y) = \bar{x}$ називають **запереченням** або **інверсією** першого елемента. Для її позначення використовують символи $<\ ^{-}, \neg>$.

Функцію $f_{13}(x, y) = x \rightarrow y$ називають **імплікацією**. Ця функція приймає значення «0» тоді і тільки тоді, коли логічна змінна x приймає значення «1», а змінна y – «0». Для її позначення використовують символи $<\rightarrow, \Rightarrow>$.

Функцію $f_{14}(x, y) = x|y$ називають функцією **Шеффера** або **запереченням кон'юнкції**. Ця функція приймає логічне значення «0» тоді і тільки тоді, коли логічні змінні x і y приймають значення «1» одночасно. Для її позначення використовують знак $<|>$.

Функцію $f_{15}(x, y) = 1$ називають **константою** «1».

Не розглянуті функції $f_4(x, y)$, $f_5(x, y)$, $f_{10}(x, y)$, $f_{11}(x, y)$ теж належать до елементарних, але вони повторюють розглянуті елементарні функції $f_2(x, y), f_3(x, y)$, $f_{12}(x, y)$, $f_{13}(x, y)$ відповідно.

2.4. Основні закони алгебри логіки

Означення 2.4.1. **Двохелементною булевою алгеброю** називають алгебраїчну структуру $< A, \cdot, \vee, \bar{}, 0, 1 >$, що створена двійковою множиною $A = \{1, 0\}$, операціями " $\cdot$ " — кон 'юнкція, " $\vee$ " — диз'юнкція і " $\bar{}$ " — заперечення або інверсія та константами 0 і 1.

Означення 2.4.2. Алгеброю логіки називають двохелементну булеву алгебру $< A, \cdot, \vee, \bar{}, \rightarrow, \sim, 0, 1 >$, в якій множину її операцій доповнено двома бінарними операціями: імплікацією та еквівалентністю.

У двохелементній булевій алгебрі, а також в алгебрі логіки необхідно виділити такі основні закони: комутативність; асоціативність; дистрибутивність; закони для заперечення; нуля та одиниці.

Доведення істинності цих законів, а також законів, які є наслідком інших, розглянемо за допомогою таблиць істинності.

1. Комутативність кон'юнкції та диз'юнкції

$$x \cdot y = y \cdot x; \; x \vee y = y \vee x.$$

Доведення комутативності кон'юнкції та диз'юнкції приведено в таблиці істинності, табл. 2.4.1.

Таблиця 2.4.1

x	y	$x \cdot y$	$y \cdot x$	$x \vee y$	$y \vee x$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	1

Стовпчики, які відповідають лівій та правій частинам рівнянь у таблиці істинності, містять однакові значення, що доводить комутативність операцій кон'юнкції та диз'юнкції.

2. Асоціативність кон'юнкції та диз'юнкції

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z; \; x \vee (y \vee z) = (x \vee y) \vee z.$$

Доведення асоціативності кон'юнкції та диз'юнкції приведено в таблиці істинності, табл. 2.4.2.

Таблиця 2.4.2

x	y	z	$y \cdot z$	$x \cdot (y \cdot z)$	$x \cdot y$	$(x \cdot y) \cdot z$	$y \vee z$	$x \vee (y \vee z)$	$x \vee y$	$(x \vee y) \vee z$
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	1	1	1	1
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	1	1	1
1	1	0	0	0	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Стовпчики, які відповідають лівій та правій частинам рівнянь, містять однакові значення в таблиці істинності, що доводить асоціативність операції кон'юнкції та диз'юнкції.

3. Дистрибутивність кон'юнкції та диз'юнкції відносно одна одної

$$x \cdot (y \vee z) = (x \cdot y) \vee (x \cdot z); \; x \vee (y \cdot z) = (x \vee y) \cdot (x \vee z).$$

Доведення дистрибутивності цих операцій приведено в таблиці істинності, табл. 2.4.3.

Таблиця 2.4.3

x	y	z	$y \vee z$	$x \cdot (y \vee z)$	$x \cdot y$	$x \cdot z$	$(x \cdot y) \vee \vee (x \cdot z)$	$y \cdot z$	$x \vee (y \cdot z)$	$x \vee y$	$x \vee z$	$(x \vee y) \cdot \cdot (x \vee z)$
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	1 ·	1	1	1
1	0	0	0	0	0	0	0	0	1	1	1	1
1	0	1	1	1	0	1	1	0	1	1	1	1
1	1	0	1	1	1	0	1	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1

Стовпчики, які відповідають лівій та правій частинам рівнянь, містять однакові значення в таблиці істинності, що доводить їх дистрибутивність.

4. Ідемпотентність кон'юнкції та диз'юнкції

$$x \cdot x = x;\ x \vee x = x.$$

Доведення ідемпотентності цих операцій приведено в таблиці істинності, табл. 2.4.4

Таблиця 2.4.4

x	$x \cdot x$	$x \vee x$	$\overline{x}$	$x \vee \overline{x}$
0	0	0	1	1
1	1	1	0	1

Із таблиці істинності випливає, що значення всіх стовпчиків однакові, і вони збігаються із значенням змінної x, що і підтверджує ідемпотентність кон'юнкції та диз'юнкції.

5. Закон виключеного третього

$$x \vee \overline{x} = 1.$$

Доведення цього закону приведено в таблиці істинності, табл. 2.4.4. Із таблиці істинності випливає, що стовпчик, який зображує ліву частину тотожності, дорівнює константі одиниці, що і треба було довести.

6. Закон протиріччя $x \cdot \overline{x} = 0$.

Доведення цього закону приведено в таблиці істинності, табл. 2.4.5.

Таблиця 2.4.5

x	$\overline{x}$	$x \cdot \overline{x}$	$x \cdot 0$	$x \cdot 1$	$x \vee 0$	$x \vee 1$
0	1	0	0	0	0	1
1	0	0	0	1	1	1

Із таблиці істинності випливає, що стовпчик, який зображує ліву частину тотожності, дорівнює константі нуль, що і треба було довести.

7. Закон тотожності з константами

$$x \cdot 0 = 0;\ x \cdot 1 = x;\ x \vee 0 = x;\ x \vee 1 = 1.$$

Доведення цієї тотожності приведено в таблиці істинності, табл. 2.4.5. Отримана таблиця істинності доводить справедливість даних тотожностей.

8. Закон елімінації

$$x \cdot (x \vee y) = x;\ x \vee (x \cdot y) = x.$$

Доведення цього закону приведено в таблиці істинності, табл. 2.4.6.

Таблиця 2.4.6

x	y	$x \vee y$	$x \cdot (x \vee y)$	$x \cdot y$	$x \vee (x \cdot y)$
0	0	0	0	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	1	1	1	1	1

Із таблиці істинності випливає, що стовпчики, які відповідають лівій та правій частинам рівнянь, містять однакові значення в таблиці істинності, що доводить закон елімінації.

9. Закон подвійного заперечення

$$\bar{\bar{x}} = x.$$

Побудуємо відповідну таблицю істинності, табл. 2.4.7.

Таблиця 2.4.7

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

Отримана таблиця істинності доводить справедливість закону.

10. Закон де Моргана

$$\overline{x \cdot y} = \bar{x} \vee \bar{y};\ \overline{x \vee y} = \bar{x} \cdot \bar{y}.$$

Доведення цього закону приведено в таблиці істинності, табл. 2.4.8.

Таблиця 2.4.8

x	y	$x \cdot y$	$\overline{x \cdot y}$	$\bar{x}$	$\bar{y}$	$\bar{x} \vee \bar{y}$	$x \vee y$	$\overline{x \vee y}$	$\bar{x} \cdot \bar{y}$
0	0	0	1	1	1	1	0	1	1
0	1	0	1	1	0	1	1	0	0
1	0	0	1	0	1	1	1	0	0
1	1	1	0	0	0	0	1	0	0

Із таблиці істинності випливає, що стовпчики, які відповідають лівій та правій частинам рівнянь, містять однакові значення в таблиці істинності, що доводить закон де Моргана.

2.5. Перетворення логічних функцій

Для розглянутих дванадцяти елементарних логічних функцій необхідно знайти логічні формули їх перетворення, що дасть математичний апарат для їх практичного застосування в науці і техніці. Ці формули легко отримати за допомогою безпосередньої перевірки в таблицях істинності за збіжністю значень. Так, наприклад, доказ перетворення розглянутої логічної функції рівнозначність приведено в табл. 2.5.1.

Із табл. 2.5.1 випливає, що логічна функція рівнозначність може мати таке перетворення:

$$x_1 \sim x_2 = (x_1 \vee \bar{x}_2) \cdot (\bar{x}_1 \vee x_2).$$

Таблиця 2.5.1

x_1	x_2	$x_1 \sim x_2$	$\bar{x}_1$	$\bar{x}_1 \vee x_2$	$\bar{x}_2$	$x_1 \vee \bar{x}_2$	$(x_1 \vee \bar{x}_2) \cdot (\bar{x}_1 \vee x_2)$
0	0	1	1	1	1	1	1
0	1	0	1	1	0	0	0
1	0	0	0	0	1	1	0
1	1	1	0	1	0	1	1

Аналогічно за допомогою таблиць істинності (табл. 2.5.2, табл. 2.5.3, табл. 2.5.4, табл. 2.5.5, табл. 2.5.6) можна довести перетворення функцій: суми за модулем два; імплікації; заборони; стрілки Пірса і функції Шеффера відповідно.

Таблиця 2.5.2

x_1	x_2	$x_1 \oplus x_2$	$\overline{x}_1$	$\overline{x}_1 \cdot x_2$	$\overline{x}_2$	$x_1 \cdot \overline{x}_2$	$\overline{x}_1 \cdot x_2 \vee x_1 \cdot \overline{x}_2$
0	0	0	1	0	1	0	0
0	1	1	1	1	0	0	1
1	0	1	0	0	1	1	1
1	1	0	0	0	0	0	0

Таблиця 2.5.3

x_1	x_2	$x_1 \to x_2$	$\overline{x}_1$	$\overline{x}_1 \vee x_2$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	0	1

Таблиця 2.5.4

x_1	x_2	$x_1 \Delta x_2$	$\overline{x}_2$	$x_1 \cdot \overline{x}_2$
0	0	0	1	0
0	1	0	0	0
1	0	1	1	1
1	1	0	0	0

Таблиця 2.5.5

x_1	x_2	$x_1 \downarrow x_2$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_1 \cdot \overline{x}_2$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0

Таблиця 2.5.6

x_1	x_2	$x_1 \mid x_2$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_1 \vee \overline{x}_2$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

Із табл. 2.5.2, ..., табл. 2.5.6 випливає, що логічні функції сума за модулем два, імплікація, заборона, стрілка Пірса та функція Шеффера при перетворенні матимуть такий вигляд відповідно:

$$x_1 \oplus x_2 = \bar{x}_1 \cdot x_2 \vee x_1 \cdot \bar{x}_2,$$
$$x_1 \to x_2 = \bar{x}_1 \vee x_2,$$
$$x_1 \Delta x_2 = x_1 \cdot \bar{x}_2,$$
$$x_1 \downarrow x_2 = \bar{x}_1 \cdot \bar{x}_2,$$
$$x_1 \mid x_2 = \bar{x}_1 \vee \bar{x}_2.$$

Для перетворення елементарних логічних функцій широко використовують закон подвійності (теорему де Моргана). Так, наприклад, доказ правильності перетворення логічних функцій

$$\overline{x_1 \cdot x_2} = \bar{x}_1 \vee \bar{x}_2;$$
$$\overline{x_1 \vee x_2} = \bar{x}_1 \cdot \bar{x}_2;$$
$$\overline{x_1 \oplus x_2} = x_1 \sim x_2;$$
$$\overline{x_1 \to x_2} = x_1 \Delta x_2;$$
$$\overline{x_1 \downarrow x_2} = x_1 \mid x_2,$$

із застосуванням закону де Моргана приведений у таблицях істинності, табл. 2.5.7, ..., табл. 2.5.11 відповідно.

Таблиця 2.5.7

x_1	x_2	$x_1 \cdot x_2$	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_1 \vee \bar{x}_2$	$\overline{x_1 \vee x_2}$
0	0	0	1	1	1	0
0	1	0	1	0	1	0
1	0	0	0	1	1	0
1	1	1	0	0	0	1

Таблиця 2.5.8

x_1	x_2	$x_1 \vee x_2$	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_1 \cdot \bar{x}_2$	$\overline{x_1 \cdot x_2}$
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	1	1	0	0	0	1

Таблиця 2.5.9

x_1	x_2	$x_1 \oplus x_2$	$x_1 \sim x_2$	$\overline{x_1 \sim x_2}$
0	0	0	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	0

Таблиця 2.5.10

x_1	x_2	$x_1 \to x_2$	$x_1 \Delta x_2$	$\overline{x_1 \Delta x_2}$
0	0	1	0	1
0	1	1	0	1
1	0	0	1	0
1	1	1	0	1

Таблиця 2.5.11

x_1	x_2	$x_1 \downarrow x_2$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_1 \mid \overline{x}_2$	$\overline{\overline{x}_1 \mid \overline{x}_2}$
0	0	1	1	1	0	1
0	1	0	1	0	1	0
1	0	0	0	1	1	0
1	1	0	0	0	1	0

2.6. Властивості логічних функцій

Функції кон'юнкція і диз'юнкція мають:

комутативну

$$x_1 \cdot x_2 = x_2 \cdot x_1; \qquad x_1 \vee x_2 = x_2 \vee x_1;$$

асоціативну

$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3; \qquad x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3;$$

і дистрибутивну

$$x_1 \cdot (x_2 \vee x_3) = x_1 \cdot x_2 \vee x_1 \cdot x_3; \qquad x_1 \vee x_2 \cdot x_3 = (x_1 \vee x_2) \cdot (x_1 \vee x_3)$$

властивості. Перевірка дистрибутивної властивості цих логічних функцій приведена в таблиці істинності, табл. 2.6.1 і табл. 2.6.2.

Таблиця 2.6.1

x_1	x_2	x_3	$x_2 \vee x_3$	$x_1 \cdot (x_2 \vee x_3)$	$x_1 \cdot x_2$	$x_1 \cdot x_3$	$x_1 \cdot x_2 \vee x_1 \cdot x_3$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Таблиця 2.6.2

x_1	x_2	x_3	$x_2 \cdot x_3$	$x_1 \vee x_2 \cdot x_3$	$x_1 \vee x_2$	$x_1 \vee x_3$	$(x_1 \vee x_2) \cdot (x_1 \vee x_3)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Із табл. 2.6.1 і табл. 2.6.2 випливає справедливість виконання дистрибутивної властивості кон'юнкції і диз'юнкції.

Ряд простих, але досить важливих співвідношень для диз'юнкції і кон'юнкції приведені в табл. 2.6.3.

Таблиця 2.6.3

Для диз'юнкції	Для кон'юнкції
$x \vee x = x$	$x \cdot x = x$
$x \vee 1 = 1$	$x \cdot 1 = x$
$x \vee 0 = x$	$x \cdot 0 = 0$
$x \vee \bar{x} = 1$	$x \cdot \bar{x} = 0$

Із табл. 2.6.3 випливає, що для диз'юнкції і кон'юнкції справедливі такі співвідношення:

$$x \vee x \vee ... \vee x = x; \; x \cdot x \cdot ... x = x.$$

Виходячи із теореми де Моргана, для диз'юнкції і кон'юнкції справедливі такі співвідношення:

$$\overline{x_1 \vee x_2 \vee ... \vee x_n} = \overline{x}_1 \cdot \overline{x}_2 \cdot ... \cdot \overline{x}_n;$$

$$\overline{x_1 \cdot x_2 \cdot ... \cdot x_n} = \overline{x}_1 \vee \overline{x}_2 \vee ... \vee \overline{x}_n.$$

Властивості логічної функції суми за модулем два, а також функцій імплікації та заборони часто можуть бути корисні для аналізу і синтезу різних дискретних пристроїв і систем.

Логічна функція **сума за модулем два** має **комутативну** й **асоціативну** властивості, а також дистрибутивну властивість відносно кон'юнкції:

$$x_1 \oplus x_2 = x_2 \oplus x_1;$$

$$x_1 \oplus (x_2 \oplus x_3) = (x_1 \oplus x_2) \oplus x_3;$$

$$x_1 \cdot (x_2 \oplus x_3) = (x_1 \cdot x_2) \oplus (x_1 \cdot x_3).$$

Для цієї логічної функції характерні також такі співвідношення:

$$x \oplus x = 0; \qquad x \oplus \overline{x} = 1;$$

$$x \oplus 0 = x; \qquad x \oplus 1 = \overline{x};$$

$$x_1 \vee x_2 = x_1 \oplus x_2 \oplus x_1 \cdot x_2.$$

На відміну від розглянутих раніше функцій, логічні **функції імплікації і заборони не мають комутативної і асоціативної властивостей**, але їх характеризують такі співвідношення:

$$x \to x = 1; \qquad x \; \Delta \; x = 0;$$

$$x \to \overline{x} = \overline{x}; \qquad x \; \Delta \; \overline{x} = x;$$

$$x \to 1 = 1; \qquad x \; \Delta \; 1 = 0;$$

$$x \to 0 = \overline{x}; \qquad x \; \Delta \; 0 = x;$$

$$1 \to x = x; \qquad 1 \; \Delta \; x = \overline{x};$$

$$0 \to x = 1; \qquad 0 \; \Delta \; x = 0;$$

$$x_1 \to x_2 = \overline{x}_2 \to \overline{x}_1; \qquad x_1 \; \Delta \; \overline{x}_2 = x_2 \Delta \overline{x}_1;$$

$$(x_1 \to x_2) \to x_1 = x_1; \qquad x_1 \; \Delta \; (x_2 \; \Delta \; x_1) = x_1.$$

Логічні функції диз'юнкції і кон'юнкції можуть бути виражені через імплікацію і заборону таким чином:

$$x_1 \vee x_2 = \overline{x_1} \to x_2; \qquad \overline{x_1} \cdot \overline{x_2} = \overline{\overline{x_1} \to x_2};$$

$$x_1 \vee x_2 = \overline{x_1 \,\Delta\, x_2}; \qquad x_1 \cdot x_2 = x_1 \,\Delta\, \overline{x_2}.$$

Логічні функції Шеффера і стрілка Пірса мають комутативну властивість:

$$x_1 \mid x_2 = x_2 \mid x_1; \qquad x_1 \downarrow x_2 = x_2 \downarrow x_1,$$

але у них відсутня асоціативна властивість

$$x_1 \mid (x_2 \mid x_3) \neq (x_1 \mid x_2) \mid x_3; \qquad x_1 \downarrow (x_2 \downarrow x_3) \neq (x_1 \downarrow x_2) \downarrow x_3.$$

Логічні функції Шеффера і стрілка Пірса мають ряд співвідношень, які приведені в табл. 2.6.4.

Таблиця 2.6.4

Для функції Шеффера	Для стрілки Пірса
$x \mid x = \overline{x}$	$x \downarrow x = \overline{x}$
$x \mid \overline{x} = 1$	$x \downarrow \overline{x} = 0$
$x \mid 1 = \overline{x}$	$x \downarrow 1 = 0$
$x \mid 0 = 1$	$x \downarrow 0 = \overline{x}$

Логічні функції Шеффера і стрілки Пірса зв'язані між собою співвідношеннями, аналогічними формулам де Моргана для логічних функцій кон'юнкції і диз'юнкції:

$$x_1 \mid x_2 = \overline{\overline{x_1} \downarrow \overline{x_2}}, \qquad x_1 \downarrow x_2 = \overline{\overline{x_1} \mid \overline{x_2}}.$$

Так як асоціативної властивості для функції Шеффера і стрілки Пірса не існує, тобто не можна розкривати дужки і виносити за них змінні, то **доказ тотожностей, які складаються тільки з цих функцій, можна виконувати двома** шляхами. **Шлях перший** — за допомогою таблиць істинності, а **шлях другий** — за допомогою окремого перетворення лівої і правої частин тотожності в диз'юнктивно-кон'юнктивну формулу з наступним їх порівнянням.

Приклад 2.6.1. Доказати тотожність

$$(x_1 \downarrow x_2) \mid (x_3 \downarrow x_4) = \overline{x_1} \mid \overline{x_2} \mid \overline{x_3} \mid \overline{x_4}.$$

Розв'язання. Окремо перетворюємо ліву і праву частини рівняння в диз'юнктивно-кон'юнктивну форму:

$$(x_1 \downarrow x_2) \mid (x_3 \downarrow x_4) = \overline{(x_1 \vee x_2)} \mid \overline{(x_3 \vee x_4)} = (\overline{x_1} \cdot \overline{x_2}) \mid (\overline{x_3} \cdot \overline{x_4}) =$$
$$= \overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}} = x_1 \vee x_2 \vee x_3 \vee x_4$$
$$\overline{x_1} \mid \overline{x_2} \mid \overline{x_3} \mid \overline{x_4} = \overline{x_1} \mid \overline{x_2} \mid (\overline{x_3} \cdot \overline{x_4}) = \overline{x_1} \mid (\overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}) =$$
$$= \overline{\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}} = x_1 \vee x_2 \vee x_3 \vee x_4$$

Так як перетворені ліва і права частини рівняння збігаються, то тотожність правильна.

Приклад 2.6.2. Довести тотожність

$$(x_1 \downarrow x_2) \mid (x_1 \downarrow x_3) = \overline{x_1 \downarrow (\overline{x_2} \mid \overline{x_3})}.$$

Розв'язання. Окремо перетворюємо ліву і праву частини рівняння в диз'юнктивно-кон'юктивну форму:

$$(x_1 \downarrow x_2) \mid (x_1 \downarrow x_3) = \overline{(x_1 \vee x_2)} \mid \overline{(x_1 \vee x_3)} = \overline{\overline{(x_1 \vee x_2)} \cdot \overline{(x_1 \vee x_3)}} =$$
$$= x_1 \vee x_2 \vee x_1 \vee x_3 = x_1 \vee x_2 \vee x_3.$$
$$\overline{x_1 \downarrow (\overline{x_2} \mid \overline{x_3})} = \overline{x_1 \downarrow (x_2 \vee x_3)} = \overline{\overline{x_1 \vee x_2 \vee x_3}} = x_1 \vee x_2 \vee x_3 = x_1 \vee x_2 \vee x_3.$$

Так як перетворені ліва і права частини рівняння збігаються, то тотожність правильна.

2.7. Суперпозиція логічних функцій

Означення 2.7.1. Функцію $f = f(f_1, f_2, \ldots, f_n)$, яка отримана із функцій $f_1, f_2, \ldots, f_n$, називають **суперпозицією** функцій $f_1, f_2, \ldots, f_n$.

Приклад 2.7.1. Задана суперпозиція функцій $f((f_1 \vee f_2) \to f_3)$ і самі функції: $f_1(x_1, x_2) = x_1 \downarrow x_2$; $f_2(x_1, x_2) = x_1 \mid x_2$; $f_3(x_3) = x_3$. Необхідно знайти функцію $f(x_1, x_2, x_3)$, як суперпозицію функцій f_1, f_2, f_3.

Розв'язання. У відповідності з означенням 2.7.1, отримаємо

$$f(x_1, x_2, x_3) = [\,(x_1 \downarrow x_2) \vee (x_1 \mid x_2)\,] \to x_3 = [\,\overline{(x_1 \vee x_2)} \vee \overline{(x_1 \cdot x_2)}\,] \to x_3 =$$
$$= \overline{(\bar{x}_1 \cdot \bar{x}_2 \vee \bar{x}_1 \vee \bar{x}_2)} \vee x_3 = (x_1 \vee x_2) \cdot x_1 \cdot x_2 \vee x_3 = x_1 \cdot x_2 \vee x_3$$

Приклад 2.7.2. Задана суперпозиція функцій $f((f_1 \mid f_2)\ \Delta\ f_3)$ і самі функції: $f_1(x_1, x_2) = x_1 \to x_3$; $f_2 = x_1 \downarrow x_2$; $f_3 = x_2$. Необхідно знайти функцію $f(x_1, x_2, x_3)$, як суперпозицію функцій f_1, f_2, f_3.

Розв'язання. У відповідності з означенням 2.7.1, отримаємо

$$f(x_1, x_2, x_3) = [(x_1 \to x_3) \mid (x_1 \downarrow x_2)]\ \Delta x_2 = [(\bar{x}_1 \vee x_3) \mid \overline{(x_1 \vee x_2)}] \cdot \bar{x}_2 =$$
$$= [\,\overline{(\bar{x}_1 \vee x_3)} \vee \overline{\overline{(x_1 \vee x_2)}}\,] \cdot \bar{x}_2 = [\,x_1 \cdot \bar{x}_3 \vee (x_1 \vee x_2)\,] \cdot \bar{x}_2 =$$
$$= x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \vee x_1 \cdot \bar{x}_2 = x_1 \cdot \bar{x}_2.$$

2.8. Аналітичне представлення логічних функцій

В цьому розділі викладені універсальні методи переходу від табличного задання функції алгебри логіки до їх аналітичного представлення.

Нехай маємо двійковий набір $< x_1^*, x_2^*, ..., x_n^* >$. Порівняємо йому число i, визначене таким чином:

$$i = x_1^* \cdot 2^{n-1} + x_2^* \cdot 2^{n-2} + ... + x_n^* \cdot 2^0.$$

Число i назвемо номером набору $< x_1^*, x_2^*, ..., x_n^* >$. Розглянемо функцію $f(x_1, x_2, ..., x_n)$ визначену таким співвідношенням

$$f_i = \begin{cases} 1, \text{ якщо номер набору є} \\ 0, \text{ в протилежному випадку.} \end{cases} \qquad (2.8.1)$$

Таку функцію називають **характеристичною функцією одиниці.** Припустимо, що нам удалось побудувати аналогічний вираз для функції f_i. Тоді має місце така теорема.

Теорема 2.8.1. Будь-яка таблично задана логічна функція може бути записана в такому аналітичному вигляді.

$$f(x_1, x_2, ..., x_n) = f_{i_1} \vee f_{i_2} \vee ... \vee f_{i_n} = \bigvee_{j \in T} f_{ij}, \qquad (2.8.2)$$

де T — множина номерів наборів, на яких логічна функція перетворюється в одиницю.

Доведення. Якщо взяти довільний набір аргументів логічної функції із таблиці, то можливі два випадки. В першому випадку, якщо функція дорівнює одиниці, то в співвідношенні 2.8.2 в правій частині найдеться таке f_{ij}, у якому ij відповідає номеру даного набору. Але тоді у відповідності 2.8.1 на даному наборі f_{ij} буде дорівнювати одиниці. У відповідності з $x \vee 1 = 1$ уся права частина 2.8.2 також буде дорівнювати одиниці. Якщо на даному наборі логічних аргументів функція f дорівнює нулю, то, як це випливає із формулювання теореми, серед характеристичних функцій f_{ij}, що входять у співвідношення 2.8.2, не буде такої, в якої індекс співпадає з номером даного набору аргументів. Але тоді у відповідності 2.8.1 усі члени диз'юнкції в правій частині 2.8.2 будуть дорівнювати нулю, що еквівалентно перетворенню правої частини в нуль. Оскільки ліва і права частини співвідношення 2.8.2 збігаються на даному наборі аргументів, то теорема 2.8.1 доведена.

У теоремі 2.8.1 було використано диз'юнкцію характеристичних функцій. Розглянемо використання замість диз'юнкції іншої логічної функції. Але для того, щоб виконувалось співвідношення, подібне співвідношенню 2.8.2, необхідно, щоб при перетворенні будь-якої із характеристичних функцій в одиницю, весь вираз перетворювався в одиницю, а при перетворенні в нуль увесь вираз також дорівнював нулю. Сформульовані умови приведені в табл. 2.8.1,

Таблиця 2.8.1

f_i	f_j	$f_i \nabla f_j$
0	0	0
0	1	1
1	0	1
1	1	?

де ∇ — знак невідомої логічної операції.

Знак питання в останньому рядку таблиці відповідає такій комбінації значень характеристичних функцій, яка ніколи не може зу-

стрітися, тобто $(i \neq j)$. Тому шукана функція ∇ може бути знайдена двома різними шляхами. Якщо в таблицю замість знака "?" вписати одиницю, то одержана функція є диз'юнкцією, яка розглянута в теоремі 2.8.1, а якщо нуль, то отримаємо логічну функцію суми за модулем два.

Наслідок. Будь-яка таблично задана логічна функція може бути представлена у такій аналітичній формі

$$f(x_1, x_2, ..., x_n) = f_{i_1} \oplus f_{i_2} \oplus ... \oplus f_{i_k} = \bigoplus_{i_j \in T} f_{ij} \qquad (2.8.3)$$

Представлення логічної функції у вигляді 2.8.2 називають **диз'юнктивним**, а у вигляді 2.8.3 — **поліноміальним.**

Для другого аналітичного представлення розглянемо функцію $\Phi_i(x_1, x_2, ..., x_n)$ визначену, як

$$\Phi_i = \begin{cases} 0, \text{ якщо номер набору є} \\ 1, \text{ в протилежному випадку.} \end{cases} \qquad (2.8.4)$$

Такі функції називають ***характеристичними функціями нуля.*** Із співвідношень 2.8.1 і 2.8.4 випливає, що

$$f(x_1, x_2, ..., x_n) = \Phi_i(x_1, x_2, ..., x_n)$$

Теорема 2.8.2. Будь-яка таблична функція алгебри логіки може бути задана у такій аналітичній формі

$$f(x_1, x_2, ..., x_n) = \Phi_{i_1} \cdot \Phi_{i_2} \cdot ... \Phi_{i_k} = \mathop{\&}_{i_j \in T_0} \Phi_{i_j}, \qquad (2.8.5)$$

де T_0 — множина номерів наборів, на яких логічна функція перетворюється в нуль.

Доведення теореми 2.8.2 виконується аналогічно доведенню теореми 2.8.1

Замість кон'юнкції в співвідношенні 2.8.5 можна взяти будь-яку логічну функцію, яка задовольняє умови, приведені в табл. 2.8.2,

Таблиця 2.8.2

Φ_i	Φ_j	$\Phi_i \square \Phi_j$
0	0	?
0	1	0
1	0	0
1	1	1

де □ — знак невідомої логічної операції.

Якщо в першій стрічці таблиці 2.8.2 записати нуль, то отримаємо кон'юнкцію, яка розглянута в теоремі 2.8.1, а якщо одиницю, то — функцію еквівалентності.

Наслідок. Будь-яка таблична функція алгебри логіки може бути представлена в такій аналітичній формі

$$f(x_1, x_2, ...x_n) = \Phi_{i_1} \sim \Phi_{i_2} \sim ... \sim \Phi_{i_k}, \qquad (2.8.6)$$

де $i_j \in T_0$.

Представлення логічної функції у вигляді 2.8.5 називають **кон'юнктивним**, а у вигляді 2.8.6 не має традиційної назви.

Знайдені чотири представлення характеристичних логічних функцій є основними для їх подальшого використання при дослідженні в алгебрі логіки.

Розглянемо тепер побудову аналітичних виразів найбільш характеристичних логічних функцій. Для цього введемо у розгляд «*степінь*» аргументу x, яку позначимо x^σ

$$x^\sigma = \begin{cases} x, \sigma = 1, \\ \bar{x}, \sigma = 0. \end{cases} \qquad (2.8.7)$$

Розглянемо кон'юнкцію

$$x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \qquad (2.8.8)$$

Двійковий набір $\sigma_1, \sigma_2, ..., \sigma_n$ має 2^n різних таких наборів, тому і кількість різних кон'юнкцій 2.8.8 також дорівнює 2^n.

Співставивши кожній кон'юнкції 2.8.8 номер, визначений номером набору $\sigma_1, \sigma_2, ..., \sigma_n$, можна записати

$$\bigvee_{i \in \beta} (x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n})_i, \qquad (2.8.9)$$

Тоді запис 2.8.9 означає, що це є диз'юнкція всіх кон'юнкцій з номерами із множини β.

Аналогічний вираз

$$\&_{i\in\beta}(x_1^{\sigma_1} \vee x_2^{\sigma_2} \vee \ldots \vee x_n^{\sigma_n})_i$$

означає кон'юнкцію всіх диз'юнкцій з номерами із множин β.

Покажемо, що $x_1^{\sigma_1} = 1$ тоді і тільки тоді, коли виконується рівність $x_i = \sigma_i$.

Це випливає із розгляду таких чотирьох можливих випадків:

$$1.\ \left.\begin{matrix} x_i = 0 \\ \sigma_i = 0 \end{matrix}\right\} x_i^{\sigma_i} = \bar{x}_i = 1; \qquad 2.\ \left.\begin{matrix} x_i = 0 \\ \sigma_i = 1 \end{matrix}\right\} x_i^{\sigma_i} = x_i = 0;$$

$$3.\ \left.\begin{matrix} x_i = 1 \\ \sigma_i = 0 \end{matrix}\right\} x_i^{\sigma_i} = \bar{x}_i = 0; \qquad 4.\ \left.\begin{matrix} x_i = 1 \\ \sigma_i = 1 \end{matrix}\right\} x_i^{\sigma_i} = x_i = 1. \qquad (2.8.10)$$

Звідси можна зробити висновок, що бінарна функція $f(x_i \cdot \sigma_i) = x_i^{\sigma_i}$ зображується формулою

$$x_i^{\sigma_i} = x_i \cdot \sigma_i \vee \bar{x}_i \cdot \bar{\sigma}_i = 1. \qquad (2.8.11)$$

Таким чином, кон'юнкція $x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot \ldots \cdot x_n^{\sigma_n}$ не перетворюється в нуль, якщо одночасно виконуються умови:

$$x_1 = \sigma_1;\ x_2 = \sigma_2;\ \ldots;\ x_n = \sigma_n; \qquad (2.8.12)$$

Тому із 2.8.12 випливає, що

$$f(x_1, x_2, \ldots x_n) = x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot \ldots \cdot x_n^{\sigma_n}.$$

Тоді у відповідності з теоремою 2.8.1 можна стверджувати, що будь-яка функція алгебри логіки, крім нуля, може бути представлена в такому вигляді

$$f(x_1, x_2, \ldots x_n) = \bigvee_1 (x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot \ldots \cdot x_n^{\sigma_n}), \qquad (2.8.13)$$

де символ $\bigvee_1$ означає, що диз'юнкція береться тільки за тими наборами $\sigma_1, \sigma_2, \ldots, \sigma_n$, для яких виконується умова $f(\sigma_1, \sigma_2, \ldots, \sigma_n) = 1$.

Тобто диз'юнкція в правій частині 2.8.11 береться тільки за тими номерами наборів аргументів, на яких функція, задана табли-

цею, перетворюється в одиницю. Таке представлення логічної функції 2.8.13 називають **досконалою диз'юнктивною нормальною формою**, скорочено **ДДНФ**.

Якщо замість співвідношення (2.8.2) використати співвідношення (2.8.3), то отримаємо **досконалу поліноміальну нормальну форму логічної функції**, скорочено **ДПНФ**, тобто ДПНФ випливає із ДДНФ шляхом заміни знака диз'юнкції на знак суми за модулем два.

Теорема 2.8.3. Будь-яка функція алгебри логіки, крім одиниці, може бути представлена в такому вигляді

$$f(x_1, x_2, ...x_n) = \underset{0}{\&}(\overline{x_1^{\sigma_1}} \vee \overline{x_2^{\sigma_2}} \vee ... \vee \overline{x_n^{\sigma_n}}), \qquad (2.8.14)$$

де символ $\underset{0}{\&}$ означає, що кон'юнкція береться тільки за тими наборами $\sigma_1, \sigma_2, ..., \sigma_n$, для яких виконується умова $f(\sigma_1, \sigma_2, ..., \sigma_n) = 0$.

Доведення. Застосовуючи теорему де Моргана для співвідношення 2.8.14, отримаємо

$$\overline{f}(x_1, x_2, ..., x_n) = \bigvee_{1}(x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n})$$

або

$$\overline{f}(x_1, x_2, ..., x_n) = \bigvee_{1} x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \cdot \overline{f}(\sigma_1, \sigma_2, ..., \sigma_n).$$

Взявши заперечення від лівої і правої частин рівняння, отримаємо

$$f(x_1, x_2, ..., x_n) = \overline{\bigvee_{1} x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \cdot \overline{f}(\sigma_1, \sigma_2, ..., \sigma_n)}.$$

Застосовуючи до правої частини співвідношення закон де Моргана, одержимо

$$f(x_1, x_2, ..., x_n) = \overline{\&\, x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \cdot \overline{f}(\sigma_1, \sigma_2, ..., \sigma_n)}.$$

Застосуємо до кон'юнкцій $x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \overline{f}(\sigma_1, \sigma_2, ..., \sigma_n)$ знову закон де Моргана, отримаємо

$$f(x_1, x_2, ..., x_n) = \&(\overline{x_1^{\sigma_1}} \vee \overline{x_1^{\sigma_2}} \vee ... \vee \overline{x_n^{\sigma_n}} \vee f(\sigma_1, \sigma_2, ..., \sigma_n)).$$

На тих наборах, для яких $f(\sigma_1, \sigma_2, ..., \sigma_n) = 1$ відповідна диз'юнкція матиме вигляд

$$\overline{x_1^{\sigma_1}} \vee \overline{x_1^{\sigma_2}} \vee \ldots \vee \overline{x_n^{\sigma_n}} \vee f(\sigma_1, \sigma_2, \ldots, \sigma_n) = 1.$$

Такі набори через $(x \vee 1 = 1,\ x \cdot 1 = x)$ на значення кон'юнкції не впливають, а тому ними можна знехтувати і отримати

$$f(x_1, x_2, \ldots x_n) = \underset{0}{\&}(\overline{x_1^{\sigma_1}} \vee \overline{x_2^{\sigma_2}} \vee \ldots \vee \overline{x_n^{\sigma_n}}),$$

що дійсно для $n \geq 1$, а для функції f, тотожно рівній нулю, на основі $x \vee \overline{x} = 1,\ x \cdot \overline{x} = 0$ маємо $0 = x_1 \cdot \overline{x}_1$, що і потрібно було довести.

Представлену функцію алгебри логіки у вигляді 2.8.14 називають **досконалою кон'юнктивною нормальною формою**, скорочено **ДКНФ**.

Контрольні запитання

1. Дайте визначення логічної функції?
2. Які змінні називають логічними або булевими?
3. Поясніть сенс поняття істотної і неістотної змінної.
4. Які логічні функції називають рівними?
5. Що називають інтерпретацією логічної функції?
6. Що називають *n*-вимірним логічним кубом?
7. Які є способи задання логічних функцій?
8. Яким чином будується таблиця істинності логічних функцій?
9. Назвіть логічні функції від однієї змінної.
10. Назвіть основні логічні функції від двох змінних.
11. Яким чином здійснюють перехід від порядкового номера логічної функції до таблиці істинності і навпаки?
12. Який пріоритет для операцій алгебри логіки?
13. Для чого використовують пріоритет операцій в алгебрі логіки?
14. Яким чином здійснюють перехід від формули до таблиці істинності логічної функції?
15. Що називають елементарною логічною функцією?
16. Скільки може бути елементарних логічних функцій від двох і трьох змінних?
17. Яка різниця між функціями: диз'юнкція і кон'юнкція, нерівнозначність і рівнозначність, стрілка Пірса і функція Шеффера, імплікації і заборона?
18. Назвіть основні закони алгебри логіки.
19. Якими способами можна довести закони алгебри логіки ?
20. Сформулюйте кожен із десяти законів алгебри логіки.

21. Запишіть тотожності для кожного із десяти законів алгебри логіки.
22. Що потрібно розуміти під перетворенням логічних функцій?
23. Якими способами можна робити перетворення логічних функцій?
24. Які закони і теореми використовують для перетворення логічних функцій?
25. Що розуміють під комутативною властивістю логічної функції?
26. Що таке асоціативна властивість логічної функції?
27. Що розуміють під дистрибутивною властивістю логічної функції?
28. Які властивості мають функції диз'юнкції і кон'юнкції?
29. Які властивості мають функції: імплікація, заборона, еквівалентність і сума за модулем два?
30. Які властивості мають функції Шеффера і стрілка Пірса?
31. Що називають суперпозицією логічної функції?
32. Чи може бути суперпозиція обернених логічних функцій?
33. Яку функцію називають характеристичною функцією одиниці?
34. Яку функцію називають характеристичною функцією нуля?
35. Назвіть форми представлення логічних функцій.

Задачі для самостійного розв'язування

1. Скільки інтерпретацій має логічна функція від двох змінних $f(x, y)$? Назвіть їх.

2. Скільки всіх логічних функцій можна отримати від двох змінних?

3. Скільки логічних функцій від n-змінних приймають однакові значення на протилежних наборах? Протилежними наборами, наприклад, для двох змінних вважати набори: <0, 0> – <1, 1>; <1, 0> – <0, 1>.

4. Побудуйте таблиці істинності функцій та визначте їх порядковий номер:

а) $f(x, y) = (x \to y) \cdot (y \to x)$;

б) $f(x, y) = x \downarrow (x \downarrow y)$;

в) $f(x, y) = (x \mid y) \downarrow (x \mid y)$;

г) $f(x, y, z) = (x \cdot y) \vee (x \cdot z) \vee (y \cdot z)$;

д) $f(x, y, z) = (x \cdot y) \oplus (x \cdot z) \oplus (y \cdot z)$;

є) $f(x, y, z) = (x \to y) \vee (x \downarrow \bar{y}) \vee (x \mid y)$.

5. Визначте інтерпретації, на яких виконуються співвідношення:

а) $x\,(x \to y) \vee (x \mid y) \to (x \to z) = 0$;

б) $(x \downarrow y) \cdot (\overline{x \mid y}) \vee (z \to \overline{x}) = 1$;

в) $(\overline{x} \downarrow y) \cdot (x \oplus y) \cdot (\overline{z} \to x) = 1$.

6. Опустіть максимально можливе число дужок у формулах з урахуванням пріоритету виконання операцій:

а) $((x \to y) \to ((y \oplus (\overline{z})) \vee t)) \vee (\overline{(t \vee (\overline{y}))} \sim x)$;

б) $((\overline{x}) \to (\overline{y})) \downarrow ((y \vee (\overline{z})) \vee (x \Delta y))$;

в) $((x \sim y) \oplus ((x \cdot z) \vee t)) \vee (((\overline{x}) \to y) \vee (\overline{z}))$;

7. Перевірте за допомогою таблиці істинності справедливість таких співвідношень:

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

б) $x \to (y \vee z) = (x \to y) \vee (x \to z)$;

в) $x \cdot (y \sim z) = (x \cdot y) \sim (x \cdot z)$;

г) $x \vee (y \sim z) = (x \vee y) \sim (x \vee z)$;

д) $x \to (y \to z) = (x \to y) \to (x \to z)$.

8. Спростіть за допомогою законів алгебри логіки нижченаведені вирази і за допомогою таблиць істинності порівняйте отримані вирази з вихідними.

а) $(x \vee \overline{z}) \cdot (\overline{x} \vee \overline{y}) \cdot (\overline{y} \wedge z) \cdot (\overline{x} \vee y) \cdot (y \vee z)$;

б) $(x \cdot \overline{z}) \vee (\overline{x} \cdot \overline{y}) \vee (y \cdot z) \vee (\overline{x} \cdot y) \vee (z \cdot \overline{y})$;

в) $((x \vee \overline{x}) \cdot (\overline{y} \vee \overline{z}) \cdot (\overline{y} \vee \overline{t}) \cdot (\overline{z} \vee t)) \vee ((\overline{y} \vee z) \cdot (z \vee t)$;

г) $(\overline{y} \vee t) \cdot (\overline{x} \cdot z) \vee (x \cdot z) \vee (\overline{t} \cdot \overline{z}) \vee (x \cdot \overline{z})$.

9. Доведіть за допомогою перетворення такі логічні вирази:

а) $x \downarrow x = x \mid x = \overline{x}$;

б) $x \mid \overline{x} = x \mid 0 = 1$;

в) $x \downarrow \overline{x} = x \downarrow 1 = 0$;

г) $x \mid 0 = x \downarrow 1 = 0$;

д) $x_1 \mid x_2 = \overline{\overline{x_1} \downarrow \overline{x_2}}$;

є) $x_1 \downarrow x_2 = \overline{\overline{x_1} \mid \overline{x_2}}$.

10. Довести, що функція еквівалентності має комутативну і асоціативну властивості.

11. Довести, що логічна функція сума за модулем два має комутативну і асоціативну властивості, а також дистрибутивну властивість відносно кон'юнкції.

12. Довести, що функції імплікації і заборона не мають комутативної і асоціативної властивостей.

13. Довести, що логічні функції Шеффера і стрілка Пірса не мають комутативної властивості.

14. Довести, що логічні функції Шеффера і стрілка Пірса не мають асоціативної властивості.

15. Для заданої суперпозиції функцій $f(f_1 \downarrow f_2) \to (f_2 \,\Delta\, f_3)$ при: $f_1(x_1,\ x_2) = x_1 \mid x_2$; $f_2(x_1,\ x_3) = x_1 \oplus x_3$; $f_3(x_2,\ x_3) = x_2 \vee \overline{x}_3$ знайти функцію $f(x_1,\ x_2,\ x_3)$.

16. Для заданої суперпозиції функцій $f(f_1 \oplus f_2) \mid (f_2 \sim f_3)$ при: $f_1(x_1,\ x_2) = x_1 \cdot \overline{x}_2$; $f_2(x_1,\ x_2) = x_1 \mid x_2$; $f_3(x_2,\ x_3) = x_2 \to x_3$ знайти функцію $f(x_1,\ x_2,\ x_3)$.

17. Задані логічні функції перевести із ДНФ у КНФ:

а) $x \cdot y \cdot \overline{z} \vee x \cdot \overline{y} \cdot z \vee \overline{x} \cdot y \cdot z$;

б) $x \cdot \overline{y} \cdot z \vee \overline{(x \cdot y)} \cdot \overline{z} \vee x \cdot \overline{(y \cdot z)}$;

в) $\overline{(x \vee y \vee z)} \vee \overline{(x \vee y)} \vee z \vee \overline{x} \cdot \overline{(y \vee z)}$.

18. Задані логічні функції перевести із КНФ у ДНФ:

а) $(\overline{x} \vee y \vee z) \cdot (x \vee \overline{y} \vee \overline{z}) \cdot (\overline{x} \vee \overline{y} \vee z)$;

б) $(x \vee \overline{y} \vee \overline{z}) \cdot \overline{(x \cdot y \vee z)} \cdot \overline{(x \vee y \cdot z)}$;

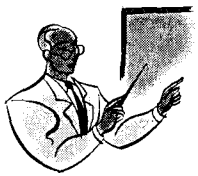
в) $\overline{(x \cdot y \cdot z)} \cdot \overline{(x \cdot y \vee z)} \cdot (x \vee \overline{y} \vee \overline{z})$.

19. Задані логічні функції перевести із ДНФ у поліноміальну:

а) $\overline{x} \vee y \vee \overline{z} \vee t$; б) $x \vee y \vee \overline{z \cdot t}$; в) $\overline{x \cdot y \cdot z \cdot t}$.

Коментарі

У даному розділі основні визначення, способи задання логічних функцій і основні закони алгебри логіки взяті з [5, 6, 12], їх властивості, суперпозиція і перетворення — з [8, 30], а аналітичне представлення — з [24].



Розділ 3
ЛОГІКА ЖЕГАЛКІНА

3.1. Основні визначення

Означення 3.1.1. **Логікою Жегалкіна** називають алгебру виду $< B;\ \wedge,\ \oplus,\ 0,\ 1 >$, що утворена множиною $B = \{0, 1\}$ разом з операціями кон'юнкції, суми за модулем два (mod 2), яку позначають знаком $\oplus$ і констант 0, 1.

Наприклад, формула $(x \oplus y \oplus z) \cdot (x \oplus y \oplus 1) \oplus y \cdot x \oplus 1$ є прикладом формули алгебри Жегалкіна, тому що вона містить булеві змінні, операцію кон'юнкція і суму за mod 2.

В алгебрі Жегалкіна операція кон'юнкція повністю ідентична логічному множенню, а операція суми за mod 2 зображає додавання за модулем два для скінченних множин.

Операція суми за mod 2 (операція алгебри Жегалкіна) відіграє важливу роль у програмуванні, обробці інформації і особливо при її кодуванні. Вона має важливу властивість – наявність оберненого елемента для кожного $x \in \{0, 1\}$. В алгебрі Жегалкіна кожний елемент є оберненим до самого себе. Це дозволяє розв'язувати рівняння шляхом додавання до обох частин однакових елементів. Наприклад, рівняння виду $x \oplus a = b$ розв'язується так:

$$x \oplus a \oplus a = b \oplus a;$$
$$x \oplus 0 = b \oplus a;$$
$$x = b \oplus a.$$

Можливість розв'язку подібних рівнянь, що відсутнє у булевій алгебрі, обумовила широке застосування операції суми за mod 2, зокрема, при кодуванні інформації.

3.2. Закони алгебри Жегалкіна

1. **Комутативність операції суми за mod 2**

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

Доведення комутативності операції приведено в таблиці істинності, табл. 3.2.1.

Таблиця 3.2.1

x	y	$x \oplus y$	$y \oplus x$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Табл. 3.2.1 показує, що ліва і права частини тотожності мають однакові таблиці істинності, тому тотожність доведена.

2. **Асоціативність операції суми за mod 2**

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

Доведення асоціативності даної операції приведено в таблиці істинності, табл. 3.2.2.

Таблиця 3.2.2

x	y	z	$y \oplus z$	$x \oplus (y \oplus z)$	$x \oplus y$	$(x \oplus y) \oplus z$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	0	0	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	0	0 ·	0
1	1	1	0	1	0	1

Із табл. 3.2.2 випливає, що ліва і права частини тотожності мають однакові таблиці істинності, що свідчить — тотожність доведена.

3. **Дистрибутивна кон'юнкція відносно суми за mod 2**

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

Доведення дистрибутивної кон'юнкції відносно суми за mod 2 приведено в таблиці істинності, табл. 3.2.3.

Таблиця 3.2.3

x	y	z	$y \oplus z$	$x \cdot (y \oplus z)$	$x \cdot y$	$x \cdot z$	$x \cdot y \oplus x \cdot z$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	0	0	1	1	0

Із табл. 3.2.3 випливає, що ліва і права частини тотожності мають однакові таблиці істинності, що свідчить — тотожність доведена.

4. **Закон зведення подібних доданків**

$$x \oplus x = 0.$$

Доведення цього закону приведено в таблиці істинності, табл. 3.2.4.

Таблиця 3.2.4

x	$x \oplus x$	$x \oplus 0$
0	0	0
1	0	1

Із табл. 3.2.4 випливає, що, незалежно від значення змінної x, їх сума за mod 2 завжди дорівнює 0, що свідчить — тотожність доведена.

5. **Операція з константою 0**

$$x \oplus 0 = x.$$

Доведення правильності цієї операції приведено в таблиці істинності, табл. 3.2.4. Із табл. 3.2.4 випливає, що змінна x при підсумуванні її за mod 2 з 0 завжди дорівнює сама собі.

Для доведення зображення будь-якої логічної функції формулою алгебри Жегалкіна достатньо виразити диз'юнкцію і заперечення через кон'юнкцію і суму за mod 2.

Зображення заперечення в алгебрі Жегалкіна формулою $\overline{x} = x \oplus 1$ виходить із таблиці істинності, табл. 3.2.5,

Таблиця 3.2.5

x	$\overline{x}$	$x \oplus 1$
0	1	1
1	0	0

а зображення диз'юнкції реалізується формулою $x \vee y = x \cdot y \oplus x \oplus y$.

Доведемо цю формулу аналітично:

$$x \vee y = \overline{\overline{x \vee y}} = \overline{\overline{x} \vee \overline{y}} = (x \oplus 1) \cdot (y \oplus 1) \oplus 1 =$$

$$= x \cdot y \oplus y \oplus x \oplus 1 \oplus 1 = x \cdot y \oplus y \oplus x.$$

Таким чином, будь-яка логічна функція може бути зображена формулою в алгебрі Жегалкіна.

Серед усіх еквівалентних зображень функції в алгебрі Жегалкіна слід виділити особливий вид формул, який називають поліномом Жегалкіна.

3.3. Поліном Жегалкіна

Означення 3.3.1. **Поліномом Жегалкіна** називають скінченну суму за mod 2 попарно різних елементарних кон'юнкцій над множиною змінних $\{x_1, x_2, ..., x_n\}$.

Для побудови полінома Жегалкіна функції, що задана певною формулою алгебри Жегалкіна, необхідно розкрити всі дужки в даній формулі за законом дистрибутивності і виконати всі можливі спрощення з використанням законів дій із константами, ідемпотентності та зведення подібних доданків.

Приклад 3.3.1. Зобразити поліномами Жегалкіна логічні функції $(x \to y)$ і еквівалентність $(x \sim y)$.

Розв'язання. Спочатку дані функції запишемо через диз'юнктивну нормальну форму. Потім виразимо операції диз'юнкції та заперечення через операції кон'юнкції та суми за mod 2. Користуючись формулою алгебри Жегалкіна і описаним вище правилом, отримаємо поліном Жегалкіна для кожної з даних функцій:

$$x \to y = \overline{x} \vee y = (x \oplus 1) \vee y = (x \oplus 1) \cdot y \oplus (x \oplus 1) \oplus y =$$
$$= x \cdot y \oplus y \oplus x \oplus 1 \oplus y = x \cdot y \oplus x \oplus 1.$$
$$x \sim y = x \cdot y \vee \overline{x} \cdot \overline{y} = x \cdot y \cdot \overline{x} \cdot \overline{y} \oplus x \cdot y \oplus \overline{x} \cdot \overline{y} = x \cdot y \oplus \overline{x} \cdot \overline{y} =$$
$$x \cdot y \oplus (x \oplus 1) \cdot (y \oplus 1) = x \cdot y \oplus x \cdot y \oplus y \oplus x \oplus 1 = y \oplus x \oplus 1.$$

Теорема 3.3.1. Будь-яку логічну функцію можна єдиним способом подати у вигляді полінома Жегалкіна.

Доведення. Нехай задана довільна логічна функція $f(x_1, x_2, ..., x_n)$.

Перетворимо її у ДДНФ, тобто у диз'юнкцію конституент одиниці $f = K_1 \vee K_2 \vee ... \vee K_m$. Замінивши знак $\vee$ на $\oplus$, отримаємо формулу $\varphi = K_1 \oplus K_2 \oplus ... \oplus K_m$. Покажемо, що функції f і φ рівні. На жодному наборі $(a_1, a_2, ..., a_n)$ значень змінних дві різні конституенти одиниці не можуть водночас перетворитися на 1, бо i-та конституента одиниці набуває значення 1 лише на i-му наборі. Отже, обчислюючи значення функцій f і φ, потрібно встановити значення багатомісної диз'юнкції чи, відповідно, суми за mod 2 змінних, із яких тільки один може дорівнювати 1, а решта – 0. Але тоді диз'юнкція та сума за mod 2 дають однаковий результат (0, *якщо всі змінні дорівнюють* 0 *і* 1, *якщо одна і тільки одна змінна дорівнює* 1). У формулі $K_1 \oplus K_2 \oplus ... \oplus K_m$ за тотожністю $\overline{x} = 1 \oplus x$ замінимо всі заперечення змінних. У результаті цього отримаємо формулу алгебри Жегалкіна. Для перетворення її у поліном Жегалкіна розкриваємо дужки, застосувавши закон ідемпотентності для кон'юнкції та зводимо подібні члени. Для того, щоб довести, що таке подання єдине, підрахуємо кількість поліномів Жегалкіна від n змінних: $x_{i_1}, x_{i_2}, ..., x_{i_m}$ дорівнює кількості підмножин $\{i_1, i_2, ..., i_m\}$. множини $\{1, 2, ..., n\}$, тобто 2^n. Кожна з цих кон'юнкцій може входити в поліном з коефіцієнтом 0 або 1. Отже кількість поліномів Жегалкіна від n змінних дорівнює кількості кортежів довжиною 2^n із компонентами 0 або 1, тобто 2^{2^n}. Кількість різних булевих

функцій від n змінних також дорівнює 2^{2^n}. Оскільки одному поліному не можуть відповідати дві різні логічні функції, то кожній логічній функції відповідає єдиний поліном Жегалкіна, що і потрібно було довести.

3.4. Методи побудови полінома Жегалкіна

Метод невизначених коефіцієнтів. У цьому методі для логічної функції $f(x_1, x_2,..., x_n)$ записують найзагальніший вигляд полінома Жегалкіна $P(x_1, x_2,..., x_n)$ із невизначеними коефіцієнтами (2^n). Даний поліном для двох змінних має загальний вигляд

$$P(x, y) = c_0 \oplus c_1 \cdot x \oplus c_2 \cdot y \oplus c_3 \cdot x \cdot y,$$

а для трьох змінних —

$$P(x, y, z) = c_0 \oplus c_1 \cdot x \oplus c_2 \cdot y \oplus c_3 \cdot z \oplus c_4 \cdot x \cdot y \oplus c_5 \cdot x \cdot z \oplus c_6 \cdot y \cdot z \oplus c_7 \cdot x \cdot y \cdot z$$

Для кожного двійкового набору $<a_1, a_2,..., a_n>$ значень змінних записують 2^n рівнянь $f(a_1, a_2,..., a_n) = P(a_1, a_2,..., a_n)$. Розв'язавши їх, отримують коефіцієнти полінома $P(x_1, x_2,..., x_n)$.

Приклад 3.4.1. Побудувати поліном Жегалкіна для функції $f(x, y) = x \sim y$.

Розв'язання. Запишемо поліном для даної функції у вигляді суми за mod 2 всіх можливих елементарних кон'юнкцій для x, y із невизначеними коефіцієнтами

$$f(x, y) = c_0 \oplus c_1 \cdot x \oplus c_2 \cdot y \oplus c_3 \cdot x \cdot y,$$

де коефіцієнти c_0, c_1, c_2, c_3 приймають значення з множини {0, 1} і засвідчують присутність або відсутність елементарної кон'юнкції в поліномі Жегалкіна. Шукаємо послідовно значення коефіцієнтів, підставляючи значення змінних і функції на різних інтерпретаціях

$$f(0, 0) = x \sim y = x \cdot y \vee \bar{x} \cdot \bar{y} = 1.$$
$$1 = c_0 \oplus c_1 \cdot 0 \oplus c_2 \cdot 0 \oplus c_3 \cdot 0 \cdot 0 = c_0.$$

Отже, $c_0 = 1$.

Далі

$$f(0,\ 1) = x \sim y = x \cdot y \vee \overline{x} \cdot \overline{y} = 0.$$

$$0 = c_0 \oplus c_1 \cdot 0 \oplus c_2 \cdot 1 \oplus c_3 \cdot 0 \cdot 1 = c_0 \oplus c_2 \cdot 1 = c_2 \oplus 1,$$

звідки маємо $c_2 = 1$.

На такій інтерпретації

$$f(1,\ 0) = x \sim y = x \cdot y \vee \overline{x} \cdot \overline{y} = 0.$$

$$0 = c_0 \oplus c_1 \cdot 1 \oplus c_2 \cdot 0 \oplus c_3 \cdot 1 \cdot 0 = c_0 \oplus c_2 \cdot 1 = c_0 \oplus c_1 = c_1 \oplus 1,$$

звідки маємо $c_1 = 1$.

Нарешті,

$$f(1,\ 1) = x \sim y = x \cdot y \vee \overline{x} \cdot \overline{y} = 1.$$

$$1 = c_0 \oplus c_1 \cdot 1 \oplus c_2 \cdot 1 \oplus c_3 \cdot 1 \cdot 1 = 1 \oplus 1 \oplus 1 \oplus c_3 = c_3 \oplus 1,$$

звідки маємо $c_3 = 0$.

Підставивши отримані значення коефіцієнтів c_0, c_1, c_2, c_3, знаходимо поліном Жегалкіна для логічної функції $f(x,\ y) = x \sim y$, який буде дорівнювати

$$f(x,\ y) = c_0 \oplus c_1 \cdot x \oplus c_2 \cdot y \oplus c_3 \cdot x \cdot y = 1 \oplus 1 \cdot x \oplus 1 \cdot y \oplus x \cdot y \cdot 0 = 1 \oplus x \oplus y.$$

Метод рівносильних перетворювань. У цьому методі спочатку будують рівносильну формулу, в якій є лише операції кон'юнкції та заперечення, а потім всюди змінюють $\overline{x}$ на $1 \oplus x$. Після цього тривіальними перетвореннями отримують поліном Жегалкіна.

Приклад 3.4.2. Побудувати поліном Жегалкіна для функції $f(x,\ y) = x \to y$.

Розв'язання. Використовуючи введені раніше еквівалентності в алгебрі логіки, отримаємо

$$x \to y = \overline{x} \vee y = \overline{x \cdot \overline{y}} = 1 \oplus x \cdot (1 \oplus y) = 1 \oplus x \oplus x \cdot y.$$

Метод із використанням ДДНФ логічної функції. Цей метод ґрунтується на теоремі 3.4.1. Його доцільно застосовувати тоді, коли функцію задано ДДНФ або цю формулу легко знайти.

Приклад 3.4.3. Побудувати поліном Жегалкіна для логічної функції

$$f(x,\ y,\ z) = x \cdot y \cdot \overline{z} \vee x \cdot \overline{y} \cdot z \vee y \cdot z.$$

Розв'язання. Перетворюємо задану функцію на ДДНФ

$$x\cdot y\cdot\overline{z}\vee x\cdot\overline{y}\cdot z\vee y\cdot z = x\cdot y\cdot\overline{z}\vee x\cdot\overline{y}\cdot z\vee y\cdot z\cdot(x\vee\overline{x}) =$$
$$= x\cdot y\cdot\overline{z}\vee x\cdot\overline{y}\cdot z\vee x\cdot y\cdot z\vee\overline{x}\cdot y\cdot z.$$

Використовуючи доведення теореми 3.4.1, отримаємо

$$x\cdot y\cdot\overline{z}\vee x\cdot\overline{y}\cdot z\vee x\cdot y\cdot z\vee\overline{x}\cdot y\cdot z = x\cdot y\cdot\overline{z}\oplus x\cdot\overline{y}\cdot z\oplus x\cdot y\cdot z\oplus\overline{x}\cdot y\cdot z =$$
$$= x\cdot y\cdot(1\oplus z)\oplus x\cdot(1\oplus y)\cdot z\oplus x\cdot y\cdot z\oplus(1\oplus x)\cdot y\cdot z = x\cdot y\oplus x\cdot y\cdot z\oplus$$
$$\oplus x\cdot z\oplus x\cdot y\cdot z\oplus x\cdot y\cdot z\oplus y\cdot z\oplus x\cdot y\cdot z = x\cdot y\oplus x\cdot z\oplus y\cdot z.$$

Таким чином, поліном Жегалкіна для логічної функції $x\cdot y\cdot\overline{z}\vee$ $\vee x\cdot\overline{y}\cdot z\vee y\cdot z$ матиме вигляд:

$$f(x,\ y,\ z) = x\cdot y\oplus x\cdot z\oplus y\cdot z.$$

Контрольні запитання

1. Назвіть структуру алгебри Жегалкіна.
2. Запишіть основні закони алгебри Жегалкіна.
3. Дайте визначення полінома Жегалкіна.
4. Які є методи побудови полінома Жегалкіна?
5. Сформулюйте метод невизначених коефіцієнтів для побудови полінома Жегалкіна.
6. У чому різниця методу рівносильних перетворювань і методу з використання ДДНФ для побудови полінома Жегалкіна?
7. Скільки різних поліномів Жегалкіна можна побудувати для довільної логічної функції?

Задачі для самостійного розв'язування

1. Використовуючи метод невизначених коефіцієнтів, представити поліномом Жегалкіна такі логічні функції:

а) $((x\vee\overline{y})\cdot(\overline{y}\vee\overline{z})\cdot(\overline{z}\vee x))\vee(\overline{x}\vee y)$;

б) $(\overline{x}\cdot y\vee x\cdot\overline{y})\vee((\overline{x}\vee y)\cdot(z\vee\overline{t})\cdot(\overline{x}\vee\overline{y})\cdot(t\vee z))$;

в) $(x\vee y)\cdot z\to x\to y$;

г) $(x\vee(z\vee\overline{y}\cdot\overline{z}))\cdot(z\vee x\cdot\overline{y}\vee\overline{z})$;

д) $(y\cdot z\vee x\cdot\overline{z})\cdot(x\vee y\vee z)\cdot(y\cdot z\vee x\cdot\overline{y})$.

2. Використовуючи метод рівносильних перетворювань, представити поліномом Жегалкіна такі логічні функції:

а) $x \mid y$; б) $x \Delta y$; в) $\overline{\overline{x} \cdot \overline{y} \cdot z}$;

г) $x \downarrow \overline{y}$; д) $\overline{x \to y}$; є) $\overline{x \Delta y}$.

3. Застосовуючи метод із використанням ДДНФ, представити поліномом Жегалкіна такі логічні функції:

а) $x \cdot \overline{y} \cdot z \vee \overline{x} \cdot y \cdot z \vee \overline{x} \cdot \overline{y} \cdot z \vee x \cdot \overline{y} \cdot \overline{z}$;

б) $(\overline{z} \vee x) \cdot (\overline{x} \vee y) \cdot (z \vee \overline{x} \cdot y)$;

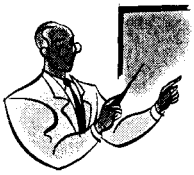
в) $z \cdot \overline{x} \vee y \cdot \overline{z} \vee x \cdot y \vee x \cdot y \cdot z$;

г) $(\overline{x} \vee y) \cdot (z \vee x) \vee (\overline{z} \vee \overline{x}) \cdot (y \vee \overline{z}) \cdot (x \vee \overline{y})$;

д) $x \cdot y \vee \overline{y} \cdot z \vee x \cdot y \cdot \overline{z} \vee x \vee y \vee z$.

Коментарі

Основні свідчення, які викладені в цьому розділі з логічних функцій Жегалкіна, взяті з [8], а побудова поліномів Жегалкіна випливає із [21].



Розділ 4

ЛОГІКА РОЗКЛАДАННЯ БУЛЕВИХ ФУНКЦІЙ

4.1. Диз'юнктивне розкладання логічних функцій

Розглянемо диз'юнктивне розкладання логічної функції $f(x_1, x_2,..., x_n)$ за k-змінними.

Теорема 4.1.1. Будь-яку булеву функцію $f(x_1, x_2,..., x_n)$ при розкладанні за k-змінними, можна зобразити у такій формі:

$$f(x_1, x_2, ..., x_k, x_{k+1}, ..., x_n) =$$
$$= \bigvee_{\sigma_1,...,\sigma_k} x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_k^{\sigma_k} \cdot f(\sigma_1, \sigma_2, ..., \sigma_k, \sigma_{k+1}, ..., x_n) \qquad (4.1.1)$$

Доведення. Для доведення необхідно перевірити, що на довільній інтерпретації $< a_1, a_2,..., a_n >$ ліва і права частини рівності (4.1.1) приймають однакові значення. Підставивши значення $< a_1, a_2,..., a_n >$ замість $< x_1, x_2 ,..., x_n >$, отримаємо

$$f(a_1, a_2 ,..., a_n) = \bigvee_{\sigma_1...\sigma_k} a_1^{\sigma_1} \cdot a_2^{\sigma_2} \cdot ... \cdot a_k^{\sigma_k} \cdot f(\sigma_1, \sigma_2,...\sigma_k, a_{k+1},..., a_n).$$

Кон'юнкція $a_1^{\sigma_1} \cdot a_2^{\sigma_2} \cdot ... \cdot a_k^{\sigma_k}$ дорівнює нулю, якщо хоча б один елемент кон'юнкції $a_i^{\sigma_i}$ дорівнює нулю, кон'юнкція відбудеться при $a_i \neq \sigma_i$ (4.1.1). Якщо ж $< a_1, a_2 ,..., a_k >$ збігається з $< \sigma_1, \sigma_2,..., \sigma_k >$, то $a_1^{\sigma_1} \cdot a_2^{\sigma_2} \cdot ... \cdot a_k^{\sigma_k} = 1$.

Звідси випливає, що

$$a_1^{\sigma_1} \cdot a_2^{\sigma_2} \cdot ... \cdot a_k^{\sigma_k} \cdot f(\sigma_1, \sigma_2 ,..., \sigma_k, a_{k+1} ,..., a_n) =$$
$$= 0 \cdot f(\sigma_1, \sigma_2,..., \sigma_k, a_{k+1} ,..., a_n) = 0$$

при $< a_1, a_2,..., a_k > \neq < \sigma_1, \sigma_2,..., \sigma_k >$.

Якщо $<a_1, a_2,..., a_k> = <\sigma_1, \sigma_2,..., \sigma_k>$, то

$$a_1^{\sigma_1} \cdot a_1^{\sigma_2} \cdot ... \cdot a_k^{\sigma_k} \cdot f(\sigma_1, \sigma_2,..., \sigma_k, a_{k+1},..., a_n) =$$
$$= 1 \cdot f(\sigma_1, \sigma_2,..., \sigma_k, a_{k+1},..., a_n) = f(a_1, a_2,..., a_n).$$

Виходячи з цього, запишемо значення правої частини тотожності 4.1.1

$$\bigvee_{\sigma_1, ..., \sigma_k} a_1^{\sigma_1} \cdot a_2^{\sigma_2} \cdot ... \cdot a_k^{\sigma_k} \cdot f(\sigma_1, \sigma_2,..., \sigma_k, a_{k+1},..., a_n) =$$
$$= 0 \vee ... \vee 0 \vee f(a_1, a_2, ..., a_n) \vee 0 \vee ... \vee 0 = f(a_1, a_2,..., a_n).$$

Отже, значення правої частини тотожності 4.1.1 збігається із значенням лівої частини на довільній інтерпретації, що і потрібно було довести.

Приклад 4.1.1. Знайти диз'юнктивне розкладання функції $f(x, y, z, t) = \left(\overline{x \vee \bar{y} \cdot z}\right) \cdot t$ за змінними x, z.

Розв'язання. Використовуючи теорему 4.1.1, отримаємо

$$f(x, y, z, t) = \bigvee_{\sigma_1, \sigma_2} a_1^{\sigma_1} \cdot a_2^{\sigma_2} \cdot f(\sigma_1 y, \sigma_2, t) = \bar{x} \cdot \bar{z} \cdot f(0, y, 0, t) \vee$$
$$\vee \bar{x} \cdot z \cdot f(0, y, 1, t) \vee x \cdot \bar{z} \cdot f(1, y, 0, t) \vee x \cdot z \cdot f(1, y, 1, t).$$

Обчислимо значення функцій:

$$f(0, y, 0, t) = \left(\overline{0 \vee \bar{y} \cdot 0}\right) \cdot t = t;$$
$$f(0, y, 1, t) = \left(\overline{0 \vee \bar{y} \cdot 1}\right) \cdot t = y \cdot t;$$
$$f(1, y, 0, t) = \left(\overline{1 \vee \bar{y} \cdot 0}\right) \cdot t = 0;$$
$$f(1, y, 1, t) = \left(\overline{1 \vee \bar{y} \cdot 1}\right) \cdot t = 0;$$

Підставивши отримані значення

$$f(0, y, 0, t), f(0, y, 1, t), f(1, y, 0, t), f(1, y, 1, t)$$

у формулу диз'юнктивного розкладання за змінними x та z, отримаємо

$$f(x, y, z, t) = \bar{x} \cdot \bar{z} \cdot t \vee \bar{x} \cdot z \cdot y \cdot t \vee x \cdot \bar{z} \cdot 0 \vee x \cdot z \cdot 0 =$$
$$= \bar{x} \cdot \bar{z} \cdot t \vee \bar{x} \cdot y \cdot z \cdot t.$$

Наслідок 4.1.1. Будь-яка булева функція $f(x_1, x_2, ..., x_n)$ має таке розкладання за однією змінною:

$$f(x_1, x_2, ..., x_n) = \bigvee_{\sigma_i} x_i^{\sigma_i} \cdot f(x_1, x_2, ..., x_{i-1}, \sigma_i, x_{i+1}, ..., x_n).$$

Приклад 4.1.2. Отримати диз'юнктивне розкладання функції

$$f(x, y, z, t) = \overline{(x \vee \bar{y} \cdot z)} \cdot t \text{ за змінною } x.$$

Розв'язання. У відповідності з наслідком теореми 4.1.1 отримаємо

$$f(x, y, z, t) = \bar{x} \cdot f(0, y, z, t) \vee x \cdot f(1, y, z, t).$$

Обчислюючи значення функції $f(x, y, z, t)$ при $x = 0$ і $x = 1$, маємо

$$f(0, y, z, t) = \left(\overline{0 \vee \bar{y} \cdot z}\right) \cdot t = \left(y \vee \bar{z}\right) \cdot t;$$
$$f(1, y, z, t) = \left(\overline{1 \vee \bar{y} \cdot z}\right) \cdot t = 0$$

і, підставивши їх значення у формулу диз'юнктивного розкладання отримаємо

$$f(x, y, z, t) = \bar{x} \cdot \left(y \vee \bar{z}\right) \cdot t \vee x \cdot 0 = \bar{x} \cdot \left(y \vee \bar{z}\right) \cdot t.$$

Наслідок 4.1.2. Будь-яка логічна функція $f(x_1, x_2, ..., x_n) \neq 0$ має таке розкладання за всіма змінними

$$f(x_1, x_2, ..., x_n) = \bigvee_{\substack{<\sigma_1, ..., \sigma_n> \\ f(\sigma_1, ..., \sigma_n)=1}} x_i^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n}, \quad (4.1.2)$$

де запис $\bigvee_{\substack{<\sigma_1, ..., \sigma_n> \\ f(\sigma_1, ..., \sigma_n)=1}}$ означає, що диз'юнкція береться за всіма можливими наборами значень $<\sigma_1, \sigma_2, ..., \sigma_n>$, на яких $f(\sigma_1, \sigma_2, ..., \sigma_n)=1$.

Замінимо співвідношення (4.1.2) для випадку $k = n$

$$f(x_1, x_2, ..., x_n) = \bigvee_{\sigma_1, ..., \sigma_n} x_i^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \cdot f(\sigma_1, \sigma_2, ..., \sigma_n)$$

і якщо $f(\sigma_1, \sigma_2, ..., \sigma_n) = 0$, то $x_i^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \cdot f(\sigma_1, \sigma_2, ..., \sigma_n) = 0$, але якщо $f(\sigma_1, \sigma_2, ..., \sigma_n) = 1$, то

$$x_i^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n} \cdot f(\sigma_1, \sigma_2, ..., \sigma_n) = x_i^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n}.$$

Тому диз'юнктивне розкладання за всіма змінними містить тільки кон'юнкції, що відповідають наборам $< \sigma_1, \sigma_2, ..., \sigma_n >$, для яких $f(\sigma_1, \sigma_2, ..., \sigma_n) = 1$.

Приклад 4.1.3. Отримати диз'юнктивне розкладання функції $f(x, y, z) = \bar{x}\, y \vee \bar{z}$ за всіма змінними.

Розв'язання. Встановимо значення функції на кожній з інтерпретацій:

$$f(0, 0, 0) = \bar{0} \cdot 0 \vee \bar{0} = 1;$$
$$f(0, 0, 1) = \bar{0} \cdot 0 \vee \bar{1} = 0;$$
$$f(0, 1, 0) = \bar{0} \cdot 1 \vee \bar{0} = 1;$$
$$f(0, 1, 1) = \bar{0} \cdot 1 \vee \bar{1} = 1;$$
$$f(1, 0, 0) = \bar{1} \cdot 0 \vee \bar{0} = 1;$$
$$f(1, 0, 1) = \bar{1} \cdot 0 \vee \bar{1} = 0;$$
$$f(1, 1, 0) = \bar{1} \cdot 1 \vee \bar{0} = 1;$$
$$f(1, 1, 1) = \bar{1} \cdot 1 \vee \bar{1} = 0.$$

Використовуючи формулу 4.1.2, отримаємо

$$f(x, y, z) = x^0 \cdot y^0 \cdot z^0 \vee x^0 \cdot y^1 \cdot z^0 \vee x^0 \cdot y^1 \cdot z^1 \vee x^1 \cdot y^0 \cdot z^0 \vee x^1 \cdot y^1 \cdot z^0 =$$
$$= \bar{x} \cdot \bar{y} \cdot \bar{z} \vee \bar{x} \cdot y \cdot \bar{z} \vee \bar{x} \cdot y \cdot z \vee x \cdot \bar{y} \cdot \bar{z} \vee x \cdot y \cdot \bar{z}.$$

Означення 4.1.1. **Елементарною кон'юнкцією** називають кон'юнкцію будь-якого числа булевих змінних, що взяті із запереченням або без нього, в якій кожна змінна зустрічається не більше одного разу.

Наприклад, елементарною кон'юнкцією для функції від трьох змінних можуть бути: $\bar{x} \cdot y \cdot \bar{z}$, $x \cdot \bar{y} \cdot z$, $x \cdot \bar{y} \cdot \bar{z}$.

Означення 4.1.2. **Диз'юнктивною нормальною формою (ДНФ)** називають формулу, що зображена у вигляді диз'юнкції елементарних кон'юнкцій. Наприклад, формула $\bar{x} \cdot y \vee x \cdot \bar{y} \vee x \cdot \bar{y} \cdot \bar{z}$ є ДНФ функції $f(x, y, z)$.

Означення 4.1.3. **Конституентою одиниці** функції $f(x_1, x_2, ..., x_n)$ називають елементарну кон'юнкцію $x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n}$, якщо $f(\sigma_1, \sigma_2, ..., \sigma_n) = 1$, тобто інтерпретація, яка обертає в одиницю дану елементарну кон'юнкцію, обертає в одиницю і функцію f.

Означення 4.1.4. **Досконалою диз'юнктивною нормальною формою (ДДНФ)** булевої функції називають формулу, що зображена у вигляді диз'юнкції конституент одиниці даної функції. Наприклад, формула $\bar{x} \cdot y \cdot z \vee x \cdot \bar{y} \cdot \bar{z} \vee x \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot z \vee x \cdot \bar{y} \cdot \bar{z}$ є ДДНФ функції $f(x, y, z)$.

4.2. Кон'юнктивне розкладання логічних функцій

Розглянемо кон'юнктивне розкладання логічної функції $f(x_1, x_2, ..., x_n)$ за k змінними.

Теорема 4.2.1. Будь-яку логічну функцію $f(x_1, x_2, ..., x_n)$, при розкладанні за k змінними, можна зобразити у такій формі

$$f(x_1, ..., x_k, x_{k+1}, ..., x_n) = \bigwedge_{\sigma_1, ..., \sigma_k} (x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_k^{\bar{\sigma}_k} \vee \vee f(\sigma_1, \sigma_2, ..., \sigma_k, x_{k+1}, ..., x_n)) \qquad (4.2.1)$$

Запис $\bigwedge_{\sigma_1, ..., \sigma_k}$ означає, що кон'юнкція береться за всіма можливими наборами значень $<\sigma_1, \sigma_2, ..., \sigma_k>$.

Доведення. Для доведення теореми використаємо тотожність 4.1.1 і принцип двоїстості. Запишемо диз'юнктивне розкладання функції $f^*(x_1, x_2, ..., x_n)$, яка двоїста функції $f(x_1, x_2, ..., x_n)$,

$$f^*(x_1, x_2, ..., x_n) = \bigvee_{\sigma_1, ..., \sigma_k} x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_k^{\sigma_k} \cdot f(\sigma_1, \sigma_2, ..., \sigma_k, x_{k+1}, ..., x_n).$$

Тотожність, двоїста до останньої, має вигляд

$$f^{**}(x_1, x_2, ..., x_n) = \bigwedge_{\sigma_1, ..., \sigma_k} \Big[\left(x_1^{\sigma_1}\right)^* \vee \left(x_2^{\sigma_2}\right)^* \vee ... \vee \left(x_k^{\sigma_k}\right)^* \vee$$
$$\vee f^{**}(\sigma_1, \sigma_2, ..., x_{k+1}, ..., x_n) \Big].$$

В лівій частині рівності ми одержали $f(x_1, x_2, ..., x_n)$, а у правій частині, використовуючи формулу (4.2.1), знайдемо двоїсту до неї шляхом заміни всіх вхідних до неї функцій на двоїсті $\left(x_i^{\sigma_i} = 1\right)^* = (1)^* = 0 = x_i^{\overline{\sigma_i}}$.

Таким чином, рівність з f^{**} приймає форму 4.2.1, що і потрібно було довести.

Приклад 4.2.1. Виконати кон'юнктивне розкладання функції $f(x, y, z) = \left(x \cdot y \vee \overline{y}\right) \cdot z$ за змінними x, y.

Розв'язання. Використовуючи формулу 4.2.1 маємо

$$f(x, y, z) = \left(x^{\overline{0}} \vee y^{\overline{0}} \vee f(0, 0, z)\right) \cdot \left(x^{\overline{0}} \vee y^{\overline{1}} \vee f(0, 1, z)\right) \cdot$$
$$\cdot \left(x^{\overline{1}} \vee y^{\overline{0}} \vee f(1, 0, z)\right) \cdot \left(x^{\overline{1}} \vee y^{\overline{1}} \vee f(1, 1, z)\right).$$

Підставляючи значення у формулу:

$$f(0, 0, z) = (0 \cdot 0 \vee \overline{0}) \cdot z = z;$$
$$f(0, 1, z) = (0 \cdot 1 \vee \overline{1}) \cdot z = 0;$$
$$f(1, 0, z) = (1 \cdot 0 \vee \overline{0}) \cdot z = z;$$
$$f(1, 1, z) = (1 \cdot 1 \vee \overline{1}) \cdot z = z.$$

отримаємо шукане кон'юнктивне розкладання

$$f(x, y, z) = (x \vee y \vee z) \cdot (x \vee \overline{y} \vee 0) \cdot (\overline{x} \vee y \vee z) \cdot (\overline{x} \vee \overline{y} \vee z) =$$
$$= (x \vee y \vee z) \cdot (x \vee \overline{y}) \cdot (\overline{x} \vee y \vee z) \cdot (\overline{x} \vee \overline{y} \vee z).$$

Наслідок 4.2.1. Для будь-якої логічної функції $f(x_1, x_2, ..., x_n)$ кон'юнктивне розкладання за однією змінною можна зобразити у такій формі

$$f(x_1, x_2, ..., x_n) = \hat{\sigma}_i \; x_i^{\sigma_i} \vee f(x_1, x_2, ..., x_{i+1}, \sigma_i, x_i + 1, ..., x_n).$$

Запис $\hat{\sigma}_i$ — означає, що кон'юнкція береться за двома можливими значеннями σ_i, тобто 0 і 1.

Приклад 4.2.2. Виконати кон'юнктивне розкладання функції $f(x, y, z) = \bar{x} \vee y \cdot z$ за змінною y.

Розв'язання. Використовуючи наслідок теореми 4.2.1 отримаємо

$$f(x, y, z) = (y^{\bar{0}} \vee f(x, 0, z))\,(y^{\bar{1}} \vee f(x, \bar{1}, z)) = (y \vee f(x, 0, z)) \cdot$$
$$\cdot (\bar{y} \vee f(x, 1, z)) = (y \vee \bar{x} \vee 0 \cdot z) \cdot (\bar{y} \vee \bar{x} \vee 1 \cdot z) = (\bar{x} \vee y) \cdot (\bar{x} \vee \bar{y} \vee z).$$

Наслідок 4.2.2. Для будь-якої булевої функції $f(x_1, x_2, ..., x_n)$ кон'юнктивне розкладання за всіма змінними можна зобразити у такій формі:

$$f(x_1, x_2, ..., x_n) = \bigwedge_{\substack{<\sigma_1, ..., \sigma_n> \\ f(\sigma_1, ..., \sigma_n)}} (x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_n^{\bar{\sigma}_n}). \qquad (4.2.2)$$

У правій частині кон'юнкція береться за всіма наборами значень $< \sigma_1, \sigma_2, ..., \sigma_k >$, на яких $f(x_1, x_2, ..., x_n) = 0$.

Нехай у формулі (4.2.1) $k = n$, тоді

$$f(x_1, x_2, ..., x_n) = \bigwedge_{\substack{<\sigma_1, ..., \sigma_n> \\ f(\sigma_1, ..., \sigma_n)=0}} \left[x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_n^{\bar{\sigma}_n} \vee f(\sigma_1, \sigma_2, ..., \sigma_n) \right].$$

Але якщо $f(\sigma_1, \sigma_2, ..., \sigma_n) = 0$, то

$$x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_n^{\bar{\sigma}_n} \vee f(\sigma_1, \sigma_2, ..., \sigma_n) = x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee x_n^{\bar{\sigma}_n}$$

і якщо $f(\sigma_1, \sigma_2, ..., \sigma_n) = 1$, то $x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee ... \vee \sigma_n^{\bar{\sigma}_n} \vee f(\sigma_1, \sigma_2, ..., \sigma_n) = 1$.

Тобто кон'юнктивне розкладання за всіма змінними містить тільки диз'юнкції, що відповідають наборам $(\sigma_1, \sigma_2, ..., \sigma_n)$, для яких

$$f(\sigma_1, \sigma_2, ..., \sigma_n) = 0.$$

Приклад 4.2.3. Виконати кон'юнктивне розкладання функції

$$f(x, y, z) = x \vee \bar{y} \cdot z$$ за всіма змінними.

Розв'язання. Вирахуємо значення функції на кожній з інтерпретацій

$$f(0, 0, 0) = 0 \vee \bar{0} \cdot 0 = 0,$$
$$f(0, 0, 1) = 0 \vee \bar{0} \cdot 1 = 1,$$
$$f(0, 1, 0) = 0 \vee \bar{1} \cdot 0 = 0,$$
$$f(0, 1, 1) = 0 \vee \bar{1} \cdot 1 = 0,$$
$$f(1, 0, 0) = 1 \vee \bar{0} \cdot 0 = 1,$$
$$f(1, 0, 1) = 1 \vee \bar{0} \cdot 1 = 1,$$
$$f(1, 1, 0) = 1 \vee \bar{1} \cdot 0 = 1,$$
$$f(1, 1, 1) = 1 \vee \bar{1} \cdot 1 = 1.$$

Використовуючи формулу 4.2.2, отримаємо

$$f(x, y, z) = (x^0 \vee y^0 \vee z^0) \cdot (x^0 \vee y^1 \vee z^0) \cdot (x^0 \vee y^1 \vee z^1) =$$
$$= (x \vee y \vee z) \cdot (x \vee \bar{y} \vee z) \cdot (x \vee \bar{y} \vee \bar{z}).$$

Означення 4.2.1. **Елементарною диз'юнкцією** називають диз'юнкцію будь-якого числа булевих змінних, що взяті із запереченням або без нього, в якій кожна змінна зустрічається не більше одного разу.

Наприклад, елементарними диз'юнкціями трьох змінних є:

$$\bar{x} \vee y \vee \bar{z}, \quad x \vee y \vee z, \quad x \vee \bar{y} \vee \bar{z}.$$

Означення 4.2.2. **Кон'юнктивною нормальною формою (КНФ)** називають формулу, що зображена у вигляді кон'юнкції елементарних диз'юнкцій.

Наприклад, формула $(x \vee \bar{y} \vee z) \cdot (\bar{x} \vee y \vee z) \cdot (\bar{x} \vee y \vee \bar{z})$ є КНФ функції $f(x, y, z)$.

Означення 4.2.3. **Конституентою нуля** називають елементарну диз'юнкцію $x_1^{\bar{\sigma}_1} \vee x_2^{\bar{\sigma}_2} \vee \ldots \vee x_n^{\bar{\sigma}_n}$ функції $f(x_1, x_2, \ldots, x_n)$, якщо

$f(\sigma_1, \sigma_2, ..., \sigma_n) = 0$, тобто інтерпретація, яка обертає в нуль дану елементарну диз'юнкцію, обертає в нуль і функцію f.

Наприклад, елементарна диз'юнкція $x \vee \bar{y} \vee z$ є конституентою нуля функції трьох змінних $f(x, y, z)$ на інтерпретації (0, 1, 0).

Дійсно, елементарна диз'юнкція $x \vee \bar{y} \vee z = x^1 \vee y^0 \vee z^1 = \bar{x^0} \vee \bar{y^1} \vee \bar{z^0}$ дорівнює нулю на інтерпретації (0, 1, 0).

Означення 4.2.4. **Досконалою кон'юнктивною нормальною формою (ДКНФ)** функції називають формулу, що зображена у вигляді кон'юнкції конституент нуля даної функції.

ДКНФ функції є результатом кон'юнктивного розкладання функції за всіма змінними і відповідає формулі (4.2.1). З аналізу формул 4.1.1 і 4.2.1, відповідних ДДНФ і ДКНФ функції, можна зробити такі **висновки**.

1. Для кожної логічної функції, що не є конституентою нуль, існує зображення у вигляді ДДНФ.

2. Для кожної логічної функції, що не є конституентою одиниці, існує зображення у вигляді ДКНФ.

3. Дві різні логічні функції не можуть мати однакові ДДНФ або ДКНФ.

4. Для кожної логічної функції існує зображення у вигляді формули алгебри логіки, що містить тільки операції диз'юнкції, кон'юнкції та заперечення.

4.3. Нормальні форми зображення логічних функцій

Для кожної логічної функції $f(x_1, x_2, ..., x_n)$ існують конституенти одиниці та нуля стільки, скільки й інтерпретацій цієї функції, тобто 2^n. Знайдемо, скільки існує різних ДДНФ і ДКНФ для булевих функцій n змінних. ДДНФ функції для n змінних можна представити у вигляді

$$f(x_1, x_2, ..., x_n) = f_1 \cdot M_1 \vee f_2 \cdot M_2 \vee ... \vee f_{2^n} \cdot M_{2^n},$$

де f_i — значення функції $f(x_1, x_2, ..., x_n)$ на i-й інтерпретації.

Так, як $M_1, M_2, ..., M_{2^n}$ однакові для всіх функцій n змінних, то вигляд ДДНФ залежить від набору 2^n булевих констант $< f_1, f_2, ..., f_n >$. З цього маємо, що кількість різних ДДНФ дорівнює кількості впорядкованих наборів 2^n булевих констант і дорівнює 2^{2^n} (§2.2). Таким чином, для кожної булевої функції існує єдина ДДНФ. Аналітично можемо довести, що існує 2^{2^n} різних ДКНФ функцій від n змінних. Тобто, кожній булевій функції відповідає єдина ДКНФ.

Наприклад, конституенти одиниці й нуля, що відповідають інтерпретаціям функцій двох змінних, приведені в табл. 4.3.1.

Таблиця. 4.3.1

Номер інтерпретації	Інтерпретація		Конституента одиниці	Конституента нуля
	x_1	x_2		
0	0	0	$\overline{x_1} \cdot \overline{x_2}$	$x_1 \vee x_2$
1	0	1	$\overline{x_1} \cdot x_2$	$x_1 \vee \overline{x_2}$
2	1	0	$x_1 \cdot \overline{x_2}$	$\overline{x_1} \vee x_2$
3	1	1	$x_1 \cdot x_2$	$\overline{x_1} \vee \overline{x_2}$

Алгоритм переходу від таблиці істинності логічної функції до ДДНФ

1. Виділити всі інтерпретації $< \sigma_1, \sigma_2, ..., \sigma_n >$, на яких значення функції дорівнює одиниці.

2. Записати конституенти одиниці виду $x_1^{\sigma_1} \cdot x_2^{\sigma_2} \cdot ... \cdot x_n^{\sigma_n}$, що відповідають відміченим інтерпретаціям.

3. Отримати ДДНФ функції за допомогою з'єднання операцією диз'юнкції записаних конституент одиниці.

Алгоритм переходу від таблиці істинності логічної функції до ДКНФ

1. Виділити всі інтерпретації $< \sigma_1, \sigma_2, ..., \sigma_n >$, на яких значення функції дорівнює нулю.

2. Записати конституенти нуля виду $x_1^{\overline{\sigma_1}} \vee x_2^{\overline{\sigma_2}} \vee \ldots \vee x_n^{\overline{\sigma_n}}$, що відповідають виділеним інтерпретаціям.

3. Отримати ДКНФ функції, записавши кон'юнкцію конституент нуля.

Приклад 4.3.1. Отримати ДДНФ і ДКНФ для функції $f(x_1, x_2)$ приведеної в табл. 4.3.2.

Таблиця 4.3.2

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	0
1	0	0
1	1	1

Розв'язання. Для отримання ДДНФ відмітимо інтерпретації, на яких $f(x_1, x_2) = 1$, і запишемо відповідні конституенти одиниці, об'єднавши їх знаком диз'юнкції $f(x_1, x_2) = x^0 \cdot y^0 \vee x^1 \cdot y^1 = \overline{x} \cdot \overline{y} \vee x \cdot y$. Для отримання ДКНФ відмітимо інтерпретації, на яких $f(x_1, x_2) = 0$, і запишемо відповідні конституенти нуля, з'єднавши їх знаком кон'юнкції.

$$f(x_1, x_2) = (x_1^{\overline{0}} \vee x_2^{\overline{1}}) \cdot (x_1^{\overline{1}} \vee x_2^{\overline{0}}) = (x_1 \vee \overline{x_2}) \cdot (\overline{x_1} \vee x_2).$$

Приклад 4.3.2. Для функції $f(x, y, z) = x \cdot \overline{y} \cdot z \vee x \cdot \overline{y} \cdot \overline{z} \vee \overline{x} \cdot y \cdot z$, що задана ДДНФ, побудувати таблицю істинності.

Розв'язання. Розглядувана функція має три конституенти одиниці: $x \cdot \overline{y} \cdot z$; $x \cdot \overline{y} \cdot \overline{z}$; $\overline{x} \cdot y \cdot z$, яким відповідають інтерпретації (1, 0, 1), (1, 0, 0), (0, 1, 1). На даних інтерпретаціях функція дорівнює одиниці, а на решті — нулю. Дані значення функції $f(x, y, z)$ і змінних x, y, z утворюють таблицю істинності функції $f(x, y, z)$, табл. 4.3.3.

Таблиця 4.3.3

x	y	z	$f(x, y, z)$	$\varphi(x, y, z)$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

Приклад 4.3.3. Для функції

$$\varphi(x, y, z) = (x \vee \overline{y} \vee z) \cdot (x \vee \overline{y} \vee \overline{z}) \cdot (\overline{x} \vee \overline{y} \vee \overline{z})$$

побудувати її таблицю істинності.

Розв'язання. Розглядувана функція має три конституенти нуля: $x \vee \overline{y} \vee z$, $x \vee \overline{y} \vee \overline{z}$, $\overline{x} \vee \overline{y} \vee \overline{z}$, яким відповідають інтерпретації (0, 1, 0), (0, 1, 1), (1, 1, 1). На даних інтерпретаціях функція дорівнює нулю, а на решті — одиниці. Дані значення функції $\varphi(x, y, z)$ і змінних x, y, z створюють таблицю істинності функції $\varphi(x, y, z)$, табл. 4.3.3.

Алгоритм переходу від довільної формули алгебри логіки до ДДНФ

1. Виключити константи, використовуючи закони дій з константами.
2. Опустити знаки заперечення безпосередньо на змінні, використовуючи закон де Моргана.
3. За допомогою дистрибутивного закону розкрити дужки і до отриманих елементарних кон'юнкцій застосувати закони ідемпотентності та протиріччя, спростити їх і звести подібні.
4. Побудувати конституенти одиниці функції введенням у кожну елементарну кон'юнкцію відсутніх змінних, використовуючи закон виключеного третього.
5. За допомогою дистрибутивного закону розкрити дужки і звести подібні, використовуючи закон ідемпотентності. Отримана формула відповідає ДДНФ функції.

Алгоритм переходу від довільної формули алгебри логіки до ДКНФ

1. Виключити константи, використовуючи закон дій із константами.

2. Опустити знаки заперечення безпосередньо на змінні, використовуючи закон де Моргана.

3. За допомогою дистрибутивного закону звести функцію до виду кон'юнкції елементарних диз'юнкцій, застосовувати до них закони ідемпотентності й виключеного третього, спростити їх і звести подібні.

4. Побудувати конституенти нуля функції, введенням у кожну елементарну диз'юнкцію відсутніх змінних, використовуючи закон протиріччя.

5. За допомогою дистрибутивного закону звести функцію до виду кон'юнкції конституенти нуля і спростити формулу, використовуючи закон ідемпотентності. Отримана формула відповідає ДКНФ функції.

Приклад 4.3.4. Побудувати ДДНФ функції

$$f(x, y, z) = x \cdot \bar{y} \vee \overline{(x \cdot (y \vee z) \vee y \cdot z)}.$$

Розв'язання. Скористаємося передостаннім алгоритмом. Опускаємо заперечення на змінні, використовуючи закон де Моргана

$$x \cdot \bar{y} \vee \overline{(x \cdot (y \vee z) \vee y \cdot z)} = x \cdot \bar{y} \vee \overline{(x \cdot (y \vee z))} \cdot (\bar{y} \vee \bar{z}) =$$
$$= x \cdot \bar{y} \vee (\bar{x} \vee \overline{y \vee z}) \cdot (\bar{y} \vee \bar{z}) = x \cdot \bar{y} \vee (\bar{x} \vee \bar{y} \cdot \bar{z}) \cdot (\bar{y} \vee \bar{z}).$$

Побудуємо ДНФ, використовуючи дистрибутивний закон, закони ідемпотентності і протиріччя

$$x \cdot \bar{y} \vee (\bar{x} \vee \bar{y}\,\bar{z}) \cdot (\bar{y} \vee \bar{z}) = x \cdot \bar{y} \vee \bar{x} \cdot \bar{y} \vee \bar{y} \cdot \bar{z} \vee \bar{y} \cdot \bar{z} \vee \bar{x} \cdot \bar{z} =$$
$$= x \cdot \bar{y} \vee \bar{x} \cdot \bar{y} \vee \bar{y} \cdot \bar{z} \vee \bar{x} \cdot \bar{z}.$$

Використовуючи закон виключеного третього, введемо до елементарних кон'юнкцій відсутні змінні

$$x \cdot \bar{y} \vee \bar{x} \cdot \bar{y} \vee \bar{y} \cdot \bar{z} \vee \bar{x} \cdot \bar{z} =$$
$$= x \cdot \bar{y}(z \vee \bar{z}) \vee \bar{x} \cdot \bar{y}\,(z \vee \bar{z}) \vee \bar{y} \cdot \bar{z}(x \vee \bar{x}) \vee \bar{x} \cdot \bar{z}(y \vee \bar{y}).$$

Використовуючи дистрибутивний закон, розкриємо дужки і зведемо подібні для отримання ДДНФ

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

Одержана ДДНФ заданої функції матиме вигляд

$$f(x,y,z)=x\cdot\bar{y}\cdot z\vee x\cdot\bar{y}\cdot\bar{z}\vee\bar{x}\cdot\bar{y}\cdot z\vee\bar{x}\cdot\bar{y}\cdot\bar{z}\vee\bar{x}\cdot y\cdot\bar{z}.$$

Приклад 4.3.5. Побудувати ДКНФ функції

$$f(x,y,z)=x\cdot\bar{y}\vee\overline{(x\cdot(y\vee\bar{z})\vee y\cdot z)}.$$

Розв'язання. Користуючись останнім алгоритмом, опустимо заперечення безпосередньо на змінні і, використовуючи закон де Моргана, отримаємо

$$f(x,y,z)=x\cdot\bar{y}\vee(\bar{x}\vee(\bar{y}\cdot z))\cdot(\bar{y}\vee\bar{z}).$$

Побудуємо КНФ, використовуючи дистрибутивний закон, закони ідемпотентності й виключеного третього,

$$x\cdot\bar{y}\vee(\bar{x}\vee(\bar{y}\cdot z))\cdot(\bar{y}\vee\bar{z})=x\cdot\bar{y}\vee(\bar{x}\vee\bar{y})\cdot(\bar{x}\vee z)\cdot(\bar{y}\vee\bar{z})=$$
$$=(x\vee(\bar{x}\vee\bar{y})\cdot(\bar{x}\vee z)\cdot(\bar{y}\vee\bar{z})\cdot\bar{y}\vee(\bar{x}\vee\bar{y})\cdot(\bar{x}\vee z)\cdot(\bar{y}\vee\bar{z}))=$$
$$=(x\vee\bar{x}\vee\bar{y})\cdot(x\vee\bar{x}\vee z)\cdot(x\vee\bar{y}\vee\bar{z})\cdot(\bar{y}\vee\bar{x}\vee\bar{y})\cdot(\bar{y}\vee\bar{x}\vee z)\cdot$$
$$\cdot(\bar{y}\vee\bar{y}\vee\bar{z})=1\cdot1\cdot(x\vee\bar{y}\vee\bar{z})\cdot(\bar{x}\vee\bar{y})\cdot(\bar{x}\vee\bar{y}\vee z)\cdot(\bar{y}\vee\bar{z})=$$
$$=(x\vee\bar{y}\vee\bar{z})\cdot(\bar{x}\vee\bar{y})\cdot(\bar{x}\vee\bar{y}\vee z)\cdot(\bar{y}\vee\bar{z}).$$

Дана функція залежить від трьох змінних, тому до елементарних диз'юнкцій $(\bar{x}\vee\bar{y})$ та $(\bar{y}\vee\bar{z})$ необхідно ввести змінні z і x відповідно, використовуючи закон протиріччя. Після чого, застосовуючи дистрибутивний закон, доведемо функцію до виду кон'юнкції конституенти нуля

$$(x\vee\bar{y}\vee\bar{z})\cdot(\bar{x}\vee\bar{y}\vee z\cdot\bar{z})\cdot(\bar{x}\vee\bar{y}\vee z)\cdot(\bar{y}\vee\bar{z}\vee x\cdot\bar{x})=$$
$$=(x\vee\bar{y}\vee\bar{z})\cdot(\bar{x}\vee\bar{y}\vee z)\cdot(\bar{x}\vee\bar{y}\vee\bar{z})\cdot(\bar{y}\vee\bar{z}\vee x)\cdot(\bar{y}\vee\bar{z}\vee\bar{x}).$$

Після зведення подібних за допомогою закону індемпотентності отримаємо ДКНФ

$$f(x,y,z)=(x\vee\bar{y}\vee\bar{z})\cdot(\bar{x}\vee\bar{y}\vee z)\cdot(\bar{x}\vee\bar{y}\vee\bar{z}).$$

Контрольні запитання

1. Дайте визначення елементарної диз'юнкції та конституенти одиниці.
2. Запишіть формулу диз'юнктивного розкладання булевих функцій від n змінних за k змінними $(k < n)$, за всіма n змінними, за однією змінною.
3. Яку властивість має конституента одиниці?
4. Дайте визначення елементарної диз'юнкції та конституенти нуля.
5. Запишіть формули кон'юнктивного розкладання логічних функцій від n змінних за k змінними $(k < n)$, за всіма n змінними, за однією змінною.
6. Яку властивість має конституента нуль?
7. Скільки існує конституент одиниці та нуля для n змінних?
8. Опишіть алгоритм переходу від таблиці істинності булевої функції до ДДНФ та ДКНФ.
9. Сформулюйте алгоритми переходу від ДДНФ та ДКНФ логічної функції до таблиці істинності.
10. Дайте порівняльну характеристику алгоритмів переходу від довільної логічної функції до ДДНФ і ДКНФ.

Задачі для самостійного розв'язування

1. Випишіть конституенти одиниці булевої функції $f(x, y, z)$:

   а) $x, \bar{y}, \bar{z}$; б) $\bar{x}, y, \bar{z}$; в) $x, \bar{y}, z$; г) $\bar{x}, \bar{y}$.

2. Знайдіть диз'юнктивне розкладання функції $f(x, y, z)$ за змінними x, y:

   а) $(\bar{x} \cdot y \vee \bar{x} \cdot z) \cdot (y \vee \bar{z}) \cdot (x \vee y \cdot z)$;
   $x \cdot y \vee x \cdot z) \cdot (y \vee z) \cdot (x \vee y \cdot z)$;

   б) $(x \cdot y \vee y \cdot z) \cdot (\bar{x} \vee z) \cdot (x \cdot y \vee x \cdot \bar{y})$;

   в) $(y \vee \bar{x} \cdot \bar{z}) \cdot (x \vee \overline{y \cdot z}) \cdot (y \vee z)$;

   г) $(\overline{\bar{y} \vee z}) \cdot (\overline{\bar{x} \vee y}) \cdot (z \vee \bar{y})$;

   д) $(x \cdot \vee \bar{y} \vee z) \cdot (\bar{x} \vee \overline{y \vee z}) \cdot (x \cdot y \vee \bar{y} \cdot \bar{z})$;

   є) $(\overline{\bar{y} \vee z}) \cdot (\overline{x \vee \bar{z}}) \cdot (x \cdot \bar{y} \vee z)$;

ж) $\left(\overline{\overline{x} \vee y} \vee \overline{z}\right) \cdot \left(y \vee \overline{z}\right) \cdot \left(x \cdot \overline{y} \cdot z\right)$.

3. Випишіть конституенти нуля логічної функції $f(x, y, z)$:

а) $x \vee y \vee \overline{z}$; б) $\overline{x} \vee \overline{y} \vee z$; в) $x \vee \overline{y} \vee \overline{z}$; г) $x \vee z$.

4. Знайти кон'юнктивне розкладання функції $f\left(x, y, z\right)$ за змінними y, z:

а) $(x \cdot z \vee \overline{y}) \cdot (x \cdot \overline{y} \vee \overline{x} \cdot y) \cdot (x \vee \overline{y} \vee z)$;

б) $(x \vee y \vee \overline{x} \cdot z) \cdot (x \cdot \overline{y} \vee z) \cdot (\overline{x} \cdot \overline{y} \vee x \cdot y \vee z)$;

в) $(x \vee \overline{y} \vee z) \cdot (\overline{x} \vee y \vee \overline{z}) \cdot (\overline{x} \vee \overline{y} \vee \overline{z})$;

г) $(\overline{\overline{x} \vee y \vee \overline{z}}) \cdot (\overline{\overline{y} \vee z}) \cdot (x \vee y \vee \overline{x} \cdot z)$;

д) $(\overline{y} \vee z) \cdot (\overline{x \vee \overline{y} \vee \overline{z}}) \cdot (x \cdot \overline{y} \vee z)$;

є) $(x \vee y \vee \overline{x} \cdot z) \cdot (\overline{z \vee y}) \cdot (\overline{x} \vee \overline{y} \vee \overline{z})$;

ж) $(x \cdot y \vee z) \cdot (\overline{x} \cdot \overline{y} \vee z) \cdot (x \vee \overline{y} \vee \overline{z})$.

5. Побудувати таблиці істинності для функцій, що задані ДДНФ:

а) $f_1(x, y, z) = x \cdot \overline{y} \cdot z \vee \overline{x} \cdot y \cdot \overline{z} \vee x \cdot \overline{y} \cdot \overline{z}$;

б) $f_2(x, y, z) = x \cdot y \cdot z \vee \overline{x} \cdot y \cdot \overline{z} \vee x \cdot \overline{y} \cdot z$;

в) $f_3(x, y, z) = \overline{x} \cdot \overline{y} \cdot z \vee x \cdot \overline{y} \cdot z \vee x \cdot \overline{y} \cdot \overline{z}$.

6. Побудувати таблиці істинності для функцій, що задані ДКНФ:

а) $f_1(x, y, z) = (x \vee \overline{y} \vee z) \cdot (\overline{x} \vee y \vee \overline{z}) \cdot (x \vee \overline{y} \vee \overline{z})$;

б) $f_2(x, y, z) = (x \vee y \vee z) \cdot (\overline{x} \vee y \vee z) \cdot (x \vee \overline{y} \vee \overline{z})$;

в) $f_3(x, y, z) = (\overline{x} \vee \overline{y} \vee z) \cdot (x \vee \overline{y} \vee z) \cdot (x \vee \overline{y} \vee \overline{z})$.

7. Знайти ДДНФ таких функцій, заданих стовпчиком у таблиці істинності:

а) $f_1(x, y, z) = (0, 1, 1, 0, 0, 1, 1, 0)$;

б) $f_2(x, y, z) = (1, 1, 0, 0, 1, 0, 1, 0)$;

в) $f_3(x, y, z) = (1, 0, 1, 0, 1, 0, 1, 0)$.

8. Отримати ДКНФ таких функцій, заданих стовпчиком у таблиці істинності:

а) $f_1(x, y, z) = (1, 1, 0, 0, 1, 0, 0, 1)$;

б) $f_2(x, y, z) = (1, 0, 1, 0, 1, 1, 0, 0)$;

в) $f_3(x, y, z) = (0, 1, 0, 0, 1, 1, 0, 1)$.

9. Знайти ДДНФ таких функцій:

а) $f_1(x, y, z) = x \cdot y \vee (z \to \bar{y}) \cdot \overline{(x \vee y \vee z)}$;

б) $f_2(x, y, z) = x \oplus y \oplus z$;

в) $f_3(x, y, z) = x \cdot \bar{y} \vee y \cdot z \vee \bar{z}$.

10. Визначити ДКНФ таких функцій:

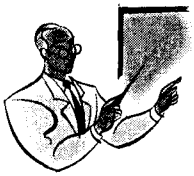
а) $f_1(x, y, z) = x \vee y \vee z$;

б) $f_2(x, y, z) = x \oplus y \oplus z$;

в) $f_3(x, y, z) = x \vee \bar{y} \cdot z \vee \bar{z}$.

Коментарі

Диз'юнктивне і кон'юнктивне розкладання логічних функцій, нормальні форми їх зображення випливають із [8, 30].



Розділ 5

ЛОГІКА ДОСЛІДЖЕННЯ БУЛЕВИХ ФУНКЦІЙ

5.1. Дослідження логічних функцій на двоїстість

Означення 5.1.1. Функцію $\overline{f}(x_1, x_2, ..., x_n)$ називають ***двоїстою*** до довільної логічної функції $f(x_1, x_2, ..., x_n)$ і позначають $[f(x_1, x_2, ..., x_n)]^*$ або f^*.

Оскільки $[f(x_1, x_2, ..., x_n)]^* = \overline{f(x_1, x_2, ..., x_n)}$, то використання операції побудови двоїстої функції складається з наступних кроків.

1. Інвертування всіх аргументів функції.
2. «Начеплення» інверсії над усією функцією.

Набір функції $< \overline{x}_1, \overline{x}_2, ..., \overline{x}_n >$ називають протилежним набору функції $< x_1, x_2, ..., x_n >$. Тому таблиця для двоїстої функції f^* випливає із таблиці для функції $f(x_1, x_2, ..., x_n)$ шляхом інвертування в стовпчику кожного значення змінної функції на протилежне і запису отриманого стовпчика у зворотній послідовності, табл. 5.1.1.

Таблиця 5.1.1

x_1	x_2	$x_1 \cdot x_2$	$(x_1 \cdot x_2)^* = \overline{\overline{x}_1 \cdot \overline{x}_2} = x_1 \vee x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Описаний стовпчик табл. 5.1.1 відповідає визначенню диз'юнкції.

В табл. 5.1.2 приведені двоїсті функції до деяких часто використовуваних елементарних логічних функцій.

Таблиця 5.1.2

f	1	x	$\bar{x}$	$x_1 \cdot x_2$	$x_1 \downarrow x_2$	$x_1 \to x_2$	$x_1 \oplus x_2$
f^*	0	x	$\bar{x}$	$x_1 \vee x_2$	$x_1 \mid x_2$	$x_2 \Delta x_1$	$x_1 \sim x_2$

Із означення двоїстості функції випливає, що $(f^*)^* = f^{**} = f$.

Приклад 5.1.1. Знайти двоїсту функцію для логічної функції $f(x_1, x_2) = \bar{x}_1 \cdot x_2$.

Розв'язання. У відповідності з означенням 5.1.1, отримаємо

$$f^*(x_1, x_2) = \overline{\overline{\bar{x}_1 \cdot x_2}} = \bar{x}_1 \vee x_2.$$

Теорема 5.1.1. Якщо задана логічна функція $f(x_1, x_2, ..., x_n) = f(f_1(x_1, x_2, ..., x_n), f_2(x_1, x_2, ..., x_n), ..., f_n(x_1, x_2, ..., x_n))$, то її двоїста функція буде дорівнювати

$$f^*(x_1, x_2, ..., x_n) = f^*(f_1^*(x_1, x_2, ..., x_n), f_2^*(x_1, x_2, ..., x_n), ..., f_n^*(x_1, x_2, ..., x_n))$$

Доведення. Доведення випливає з таких рівностей:

$$f^*(x_1, x_2, ..., x_n) = \bar{f}(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n) = \bar{f}(\bar{f}_1(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n),$$
$$\bar{\bar{f}}_2(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n), ..., \bar{\bar{f}}_n(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)) =$$
$$= \bar{f}(\bar{f}_1^*(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n), \bar{f}_2^*(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n), ..., \bar{f}_n^*(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)) =$$
$$= f^*(f_1^*(x_1, x_2, ..., x_n), f_2^*(x_1, x_2, ..., x_n), ..., f_n^*(x_1, x_2, ..., x_n)),$$

що і потрібно було довести.

Наслідок. Виходячи із теореми 5.1.1, можна сказати, що в алгебрі логіки виконання принципу двоїстості пов'язано з одночасною заміною всіх диз'юнкцій на кон'юнкції, всіх кон'юнкцій на диз'юнкції, всіх нулів на одиниці і всіх одиниць на нулі. Так як $x^* = x$ і $(\bar{x})^* = \bar{x}$, то символи змінних і їх інверсії при отриманні двоїстої функції не змінюються.

Приклад 5.1.2. Для функції $f(x_1, x_2) = x_1 \cdot x_2 \vee \bar{x}_1 \cdot \bar{x}_2$ знайти двоїсту до неї функцію.

Розв'язання. У відповідності з означенням 5.1.1, теоремою 5.1.1, двоїста функція буде дорівнювати

$$f^* = \overline{\overline{x_1} \cdot \overline{x_2} \vee x_1 \cdot x_2} = (\overline{x_1} \vee \overline{x_2}) \cdot (x_1 \vee x_2) = \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2}.$$

5.2. Дослідження логічних функцій на збереження нуля та одиниці

Означення 5.2.1. Логічну функцію $f(x_1, x_2, ..., x_n)$ називають **функцією, що зберігає 0,** якщо на нульовому наборі вона дорівнює 0, $f(0, 0, ..., 0) = 0$.

Означення 5.2.2. Логічну функцію $f(x_1, x_2, ..., x_n)$ називають **функцією, що зберігає 1,** якщо на одиничному наборі вона дорівнює 1, $f(1, 1, ..., 1) = 1$.

Функції $x \cdot y$ і $x \vee y$ зберігають 0, оскільки $0 \cdot 0 = 0$ і $0 \vee 0 = 0$. Крім того, дані функції також зберігають 1, оскільки $1 \cdot 1 = 1$ і $1 \vee 1 = 1$. Функція $\overline{x}$ не зберігає 0 і не зберігає 1, оскільки $\overline{0} = 1$, а $\overline{1} = 0$.

Приклад 5.2.1. Для функції $f(x, y, z) = x \vee \overline{y} \cdot \overline{z} \vee y \cdot z$ визначити збереження нею 0 та 1.

Розв'язання. Перевіримо значення даної функції на нульовому та одиничному наборах.

$$f(0,\ 0,\ 0) = 0 \vee \overline{0} \cdot \overline{0} \vee 0 \cdot 0 = 0 \vee 1 \vee 0 = 1,$$
$$f(1,\ 1,\ 1) = 1 \vee \overline{1} \cdot \overline{1} \vee 1 \cdot 1 = 1 \vee 0 \vee 1 = 1.$$

Отже дана функція зберігає 1 і не зберігає 0.

5.3. Дослідження логічних функцій на монотонність

Означення 5.3.1. Логічну функцію $f(x_1, x_2, ..., x_n)$ називають **монотонною,** якщо для будь-яких пар наборів значень змінних $(a_1, a_2, ..., a_n)$ і $(b_1, b_2, ..., b_n)$, для яких виконується відношення $(a_1, a_2, ..., a_n) \le (b_1, b_2, ..., b_n)$, правильна і нерівність $f(a_1, a_2, ..., a_n) \le$ $\le f(b_1, b_2, ..., b_n)$.

Для порівняння логічних констант уводиться відношення порядку, яке виражають знаком « $\le$ ».

Наприклад, відношення порядку для функції з одного змінного $f(x)$ має вигляд: $(0) \le (0)$, $(0) \le (1)$, $(1) \le (1)$, а для функції з двома змінними $f(x, y)$ встановлюється таким чином:

$(0, 0) \le (0, 0)$; $(0, 0) \le (0, 1)$; $(0, 0) \le (1, 0)$; $(0, 0) \le (1, 1)$;

$(0, 1) \le (0, 1)$; $(0, 1) \le (1, 0)$; $(0, 1) \le (1, 1)$;

$(1, 0) \le (1, 0)$; $(1, 0) \le (1, 1)$;

$(1, 1) \le (1, 1)$.

Для перевірки функції на монотонність необхідно дослідити виконання нерівності $f(a_1, a_2, ..., a_n) \le f(b_1, b_2, ..., b_n)$ для всіх пар наборів $(a_1, a_2, ..., a_n) \le (b_1, b_2, ..., b_n)$, крім випадків $(a_1, a_2, ..., a_n) = (b_1, b_2, ..., b_n)$, у яких значення функцій збігаються.

Приклад 5.3.1. Дослідити на монотонність функції $f(x, y) = x \cdot y$ та $\varphi(x, y) = x \oplus y$.

Розв'язання. Для функції $f(x, y)$ запишемо всі набори значень змінних, для яких виконується відношення порядку, становимо значення функції на даних наборах і порівняємо їх:

$(0, 0) \le (0, 1)$, $f(0, 0) = 0, f(0, 1) = 0$, $f(0, 0) \le f(0, 1)$;

$(0, 0) \le (1, 0)$, $f(0, 0) = 0, f(1, 0) = 0$, $f(0, 0) \le f(1, 0)$;

$(0, 0) \le (1, 1)$, $f(0, 0) = 0, f(1, 1) = 1$, $f(0, 0) \le f(1, 1)$;

$(0, 1) \le (1, 0)$, $f(0, 1) = 0, f(1, 0) = 0$, $f(0, 1) \le f(1, 0)$;

$(0, 1) \le (1, 1)$, $f(0, 1) = 0, f(1, 1) = 1$, $f(0, 1) \le f(1, 1)$;

$(1, 0) \le (1, 1)$, $f(1, 0) = 0, f(1, 1) = 1$, $f(1, 0) \le f(1, 1)$.

Висновок: функція $f(x, y) = x \cdot y$ є монотонною.

Аналогічно виконуємо дослідження для функції $\varphi(x, y)$:

$(0, 0) \le (0, 1)$, $\varphi(0, 0) = 0$, $\varphi(0, 1) = 1$, $\varphi(0, 0) \le \varphi(0, 1)$;

$(0, 0) \le (1, 0)$, $\varphi(0, 0) = 0$, $\varphi(1, 0) = 1$, $\varphi(0, 0) \le \varphi(1, 0)$;

$(0, 0) \le (1, 1)$, $\varphi(0, 0) = 0$, $\varphi(1, 1) = 0$, $\varphi(0, 0) \le \varphi(1, 1)$;

$(0, 1) \le (1, 1)$, $\varphi(0, 1) = 1$, $\varphi(1, 1) = 0$, $\varphi(0, 1) \not\le \varphi(1, 0)$.

Висновок: функція $\varphi(x, y) = x \oplus y$ не є монотонною.

Теорема 5.3.1. Булева функція, відмінна від констант 0 і 1, є монотонною, якщо вона припускає зображення формулою алгебри логіки без заперечень.

Доведення. Нехай деяка формула алгебри логіки без заперечень зображує деяку функцію f від n змінних. Перетворимо її на ДНФ. Вона також не буде містити заперечень. Припустимо, що функція f не монотонна, тоді на якійсь інтерпретації $a \le b$ значення $f(a) \le f(b)$, тобто $f(a)=1$, $f(b)=0$. Рівність $f(a)=1$ означає, що ДНФ функція f містить елементарну кон'юнкцію, яка дорівнює *1* на наборі a. Оскільки дана кон'юнкція не містить заперечень, її можна позначити через $x_1 \cdot x_2 \cdot ... \cdot x_k$. Оскільки $x_1 \cdot x_2 \cdot ... \cdot x_k = 1$ на наборі $a_1 \cdot a_2 \cdot ... \cdot a_k = 1$, то і $a_1 = a_2 = ... = a_k = 1$. Так як було зроблене припущення, що $a \le b$, то $b_1 = b_2 = ... = b_k = 1$. Тоді кон'юнкція $x_1 \cdot x_2 \cdot ... \cdot x_k = 1$ при підстановці значень із набору b дорівнює одиниці, звідки випливає, що $f(b)=1$. Однак з початку доведення було зроблене припущення $f(b)=0$. Значить, припущення помилкове, і функція f монотонна.

Нехай тепер f монотонна функція від n змінних. Зобразимо її у ДНФ. Припустимо, що вона містить елементарну кон'юнкцію із запереченням. Позначимо її $\overline{x}_1 \cdot k$, де k елементарна кон'юнкція змінних $x_2, ..., x_k$. Розглянемо певний набір $(0, a_2, ..., a_k, a_{k+1}, ..., a_n)$, в якому $\overline{x}_1 \wedge k = 1$, а отже і $f(a_1)=1$. Тепер розглянемо набір $(1, a_2, ..., a_k, a_{k+1}, ..., a_n)$. Оскільки $a_1 \le a_2$, і функція f монотонна, то $f(a_2)=1$. Останнє твердження буде справедливе тільки тоді, коли ДНФ функції містить елементарну кон'юнкцію $x_1 \cdot k$, яку набір $(1, a_2, ..., a_n)$ обертає на одиницю. Таким чином, ДНФ функції f можна записати у вигляді: $\overline{x}_1 \cdot k \vee x_1 \cdot k$. Застосувавши дистрибутивний закон і закон виключеного третього, отримаємо:

$$\overline{x}_1 \cdot k \vee x_1 \cdot k = (\overline{x}_1 \vee x) \cdot k = k.$$

Таким чином, змінна із запереченням завжди може бути видалена з ДНФ монотонної функції, що і потрібно було довести.

Приклад 5.3.2. Визначити, чи є функція $f(x, y, z) = (\overline{x} \vee \overline{z}) \to y$ монотонною.

Розв'язання. Виразимо задану функцію через елементарні функції алгебри логіки:

$$(\overline{x} \vee \overline{z}) \to y = \overline{(\overline{x} \vee \overline{z})} \vee y = x \cdot z \vee y.$$

Отримана формула алгебри логіки не містить заперечень, отже функція $f(x,\ y,\ z)$ є монотонною.

5.4. Дослідження логічних функцій на лінійність

Означення 5.4.1. Логічну функцію називають **лінійною**, якщо її поліном Жегалкіна (§ 3.3) не містить кон'юнкції змінних.

Приклад 5.4.1. Дослідити на лінійність функцію $f(x,\ y,\ z) = (x \downarrow y) \vee \overline{z}$.

Розв'язання. Побудуємо поліном Жегалкіна функції $f(x,\ y,\ z)$, використовуючи такі тотожності: $x \downarrow y = \overline{x \vee y}$, $x \vee y = x \cdot y \oplus x \oplus y$, $\overline{x} = x \oplus 1$ (§ 3.2), тоді

$$f(x,\ y,\ z) = (x \downarrow y) \vee \overline{z} = \overline{(x \vee y)} \vee \overline{z} = \overline{(x \vee y)} \cdot \overline{z} \oplus \overline{(x \vee y)} \oplus z \oplus 1 =$$

$$= (x \cdot y \oplus x \oplus y \oplus 1) \cdot (z \oplus 1) \oplus (x \cdot y \oplus x \oplus y \oplus 1) \oplus z \oplus 1 =$$

$$= x \cdot y \cdot z \oplus x \cdot z \oplus y \cdot z \oplus z \oplus x \cdot y \oplus x \oplus y \oplus 1 \oplus x \cdot y \oplus x \oplus y \oplus 1 \oplus z \oplus 1 =$$

$$= x \cdot y \cdot z \oplus x \cdot z \oplus y \cdot z \oplus 1.$$

Висновок: функція $f(x,\ y,\ z) = (x \downarrow y) \vee \overline{z}$ не є лінійною, оскільки її поліном містить кон'юнкції змінних.

5.5. Дослідження на замкнутість класів і повноту логічних функцій

Означення 5.5.1. Множнику K логічних функцій називають **замкненим класом**, якщо довільна суперпозиція функції із K також належить множині K.

Замкнення системи функцій K позначають $[K]$. Очевидно, що коли K — замкнений клас, то $[K] = K$.

Існує п'ять замкнених класів логічних функцій:

K_0 — функції, що зберігають 0;

K_1 — функції, що зберігають 1;

S — самодвоїсті функції;

M — монотонні функції;

L — лінійні функції.

До класу K_0 належать логічні функції $f(x_1, x_2,..., x_n)$, **які зберігають 0**, якщо $f(0, 0,..., 0)=0$. Наприклад, функції 0, x, $x\cdot y$, $x\vee y$, $x\oplus y$ зберігають 0, тобто належать класу K_0, а функції 1, $\overline{x}$, $x\to y$ не зберігають 0, тобто не належать класу K_0.

Клас K_0 містить $2^{2^n-1}=1/2\cdot 2^{2^n}$ логічних функцій, що залежать від n змінних $(x_1, x_2,..., x_n)$.

Теорема 5.5.1. K_0 — замкнений клас.

Доведення. Нехай клас K_0 містить тотожну функцію. Тоді функція $F(x_1, x_2,..., x_n)=f(f_1(x_1, x_2,..., x_n), f_2(x_1, x_2,..., x_n), ..., f_m(x_1, x_2,..., x_n))$ зберігає 0, якщо функції f, f_1, f_2, ..., f_m зберігають 0.

Дійсно, $f(f_1(0, 0,..., 0), f_2(0, 0,..., 0)..., f_m(0, 0,..., 0))=f(0, 0,..., 0)=0$, що і потрібно було довести.

Наслідок. Повна система логічних функцій має містити хоча б одну функцію, яка не зберігає 0.

До класу K_1 належать логічні функції $f(x_1, x_2,..., x_n)$, які **зберігають 1**, якщо $f(1, 1,..., 1)=1$. Наприклад, функції 1, x, $x\cdot y$, $x\vee y$, $x\to y$ належать класу K_1, а функції 0, $\overline{x}$, $x\oplus y$ — ні.

Теорема 5.5.2. K_1 — замкнений клас.

Доведення. Нехай клас K_1 містить тотожну функцію. Тоді функція $F(x_1, x_2,..., x_n)=f(f_1(x_1, x_2,..., x_n), f_2(x_1, x_2,..., x_n),..., f_m(x_1, x_2,..., x_n))$ зберігає 1, якщо функції f, f_1, f_2, ..., f_m зберігають 1.

Дійсно, $f(f_1(1, 1, ..., 1), f_2(1, 1,..., 1)..., f_m(1, 1,..., 1))=f(1, 1,..., 1)=1$, що і потрібно було довести.

Наслідок. Повна система функцій має містити хоча б одну функцію, яка не зберігає 1.

Клас K_1 містить $2^{2^n-1}=1/2\cdot 2^{2^n}$ логічних функцій, що залежать від n змінних $(x_1, x_2,..., x_n)$.

До класу S належать **самодвоїсті функції**, тобто функції, які набувають протилежних значень на протилежних наборах значень змінних $f(x_1, x_2,..., x_n) = \overline{f}(\overline{x}_1, \overline{x}_2,..., \overline{x}_n)$.

Теорема 5.5.3. Клас S — самодвоїстих функцій замкнений.

Доведення. Нехай клас S містить тотожну функцію. Тоді функція $F(x_1, x_2,..., x_n) = f(f_1(x_1, x_2,..., x_n), f_2(x_1, x_2,..., x_n), ..., f_m(x_1, x_2,..., x_n))$ — самодвоїста, якщо функції f, f_1, f_2,..., f_m самодвоїсті. Застосувавши принцип двоїстості, отримаємо:

$$F(x_1, x_2,..., x_n) = f^*(f_1^*(x_1, x_2,..., x_n), f_2^*(x_1, x_2,..., x_n),...$$
$$..., f_m^*(x_1, x_2,..., x_n)) = f(f_1(x_1, x_2,..., x_n), f_2(x_1, x_2,..., x_n),...,$$
$$f_m(x_1, x_2,..., x_n)) = F(x_1, x_2,..., x_n),$$

що і потрібно було довести.

Наслідок. Повна система логічних функцій має містити хоча б одну несамодвоїсту функцію.

Клас S самодвоїстих функцій від n змінних $(x_1, x_2,..., x_n)$ дорівнює кількості двійкових наборів довжиною 2^{n-1}, тобто $2^{2^{n-1}} = \sqrt{2^{2^n}}$.

До класу M належать **монотонні функції**, якщо для будь-яких двійкових пар наборів значень змінних $(a_1, a_2,..., a_n)$ і $(b_1, b_2,..., b_n)$, для яких виконується відношення $(a_1, a_2,..., a_n) \le (b_1, b_2,..., b_n)$, випливає, що $f(a_1, a_2,..., a_n) \le f(b_1, b_2,..., b_n)$.

Наприклад, x, $x \cdot y$, $x \vee y$ — монотонні функції, а $\overline{x}$, $x \to y$, $x \mid x$ — немонотонні.

Теорема 5.5.4. Клас M — монотонних функцій замкнений.

Доведення. Нехай клас M містить тотожну функцію. Тоді функція $F(x_1, x_2,..., x_n) = f(f_1(x_1, x_2,..., x_n), f_2(x_1, x_2,..., x_n),..., f_m(x_1, x_2,..., x_n))$ монотонна в разі монотонності функцій f, f_1, f_2,..., f_m.

Нехай $(a_1, a_2,..., a_n) \le (b_1, b_2,..., b_n)$, тоді $f_i(a_1, a_2,..., a_n) \le f_i(b_1, b_2,..., b_n)$ для всіх $i = 1, 2,..., n$. Звідси випливає, що

$$F(a_1, a_2,..., a_n) = f(f_1(a_1, a_2,..., a_n), f_2(a_1, a_2,..., a_n),...,$$
$$f_m(a_1, a_2,..., a_n)) \le f(f_1(b_1, b_2, ..., b_n), f_2(b_1, b_2,..., b_n), ...,$$
$$f_m(b_1, b_2, ..., b_n)) = F(b_1, b_2, ..., b_n).$$

Таким чином, $F(a_1, a_2,..., a_n) \le F(b_1, b_2,..., b_n)$, що і потрібно було довести.

Наслідок. Повна система логічних функцій має містити хоча б одну немонотонну функцію.

До класу *L* належать **лінійні логічні функції**, в яких поліном Жегалкіна має $f(x_1, x_2,..., x_n) = c_0 \oplus c_1 \cdot x_1 \oplus c_2 \cdot x_2 \oplus ... \oplus c_n \cdot x_n$.

Коефіцієнти $c_0, c_1, c_2,..., c_n$ лінійного полінома можуть утворювати довільний набір значень з $n+1$ нулів і одиниць. Унаслідок єдиності полінома Жегалкіна різним набором коефіцієнтів відповідають різні логічні функції. Таких функцій може бути 2^{n+1} від n змінних. Прикладом лінійних логічних функцій можуть бути функції x, $\bar{x}$, $x \sim y$, а нелінійними — $x \vee y$, $x \to y$.

Теорема 5.5.5. Клас L — лінійних функцій замкнений.

Доведення. Множина L усіх лінійних функцій є замкнений клас, тому що підстановка формул вигляду $c_0 \oplus c_1 \cdot x_1 \oplus c_2 \cdot x_2 \oplus ... \oplus c_n \cdot x_n$ у формулу такого самого вигляду знову дає формулу того самого вигляду.

Наслідок. Повна система логічних функцій має містити хоча б одну нелінійну функцію.

Замкнуті п'ять класів логічних функцій називають **класами Поста**. Розглянемо критерій функціональної повноти системи логічних функцій, використовуючи класи Поста.

Означення 5.5.1. Система логічних функцій повна, якщо вона містить хоча б одну функцію, що не зберігає нуль, хоча б одну функцію, що не зберігає одиницю, хоча б одну несамодвоїсту функцію, хоча б одну немонотонну функцію і хоча б одну нелінійну функцію.

Тобто, виходячи із зазначеного критерію Поста, випливає, що для повноти будь-якої логічної функції необхідно й достатньо, щоб для кожного з п'яти замкнених класів вона містила функцію, яка цьому класу не належить.

Необхідно зауважити, що одна й та ж логічна функція може зображати у функціонально повній системі одну або кілька необхідних властивостей. Тому мінімальне число логічних функцій у фун-

кціонально повному наборі дорівнює одиниці, але одна функція може мати всі п'ять необхідних властивостей.

Означення 5.5.2. Повну систему логічних функцій називають **нескоротною**, якщо з неї неможливо виключити жодної логічної функції без утрати властивості повноти. Максимальна кількість логічних функцій у нескоротному функціонально повному наборі дорівнює чотирьом.

Приклад 5.5.1. За допомогою критерію Поста визначити функціональну повноту системи $< \bar{}, \vee >$.

Розв'язання. Заперечення зображуване за допомогою полінома Жегалкіна $\bar{x} = x \oplus 1$. Звідси випливає, що функція заперечення — лінійна. Будуємо таблиці істинності для функції заперечення і диз'юнкції, табл. 5.5.1 і табл. 5.5.2 відповідно

Таблиця 5.5.1

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

Таблиця 5.5.2

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Із табл. 5.5.1 випливає, що функція заперечення не зберігає 0 і 1, є самодвоїста і немонотонна, а із табл. 5.5.2 те, що функція диз'юнкція зберігає 0 і 1, є несамодвоїста, не лінійна, але монотонна.

Побудуємо табл. 5.5.3, в якій помітимо знаком «+» наявність відповідних властивостей у функції, а знаком «–» — її відсутність.

Таблиця 5.5.3

Назва властивості	$\bar{x}$	$x \vee y$
Зберігання 0	–	+
Зберігання 1	–	+
Самодвоїстість	+	–
Монотонність	–	+
Лінійність	+	–

У кожному рядку таблиці присутній знак «–». Отже, для кожного класу Поста в даній системі є хоча б одна логічна функція, яка цьому класу не належить. Тому за критерієм Поста така система логічних функцій є функціонально повною.

В табл. 5.5.4 приведені результати дослідження щодо належності до п'яти класів відповідних елементарних логічних функцій двох змінних.

Належність логічної функції до розглядуваного класу в таблиці відмічено знаком «+», а неналежність — знаком «–».

Таблиця 5.5.4

Назва функції	Символічне позначення функції	Представлення функції в алгебрі логіки	Класи логічних функцій				
			K_0	K_1	S	M	L
Константа 0	0	0	+	–	–	+	+
Константа 1	1	1	–	+	–	+	+
Змінна	x	x	+	+	+	+	+
Заперечення	$\overline{x}$	$\overline{x}$	–	–	+	–	+
Рівнозначність	$x_1 \sim x_2$	$\overline{x}_1\overline{x}_2 \vee x_1x_2$	–	+	–	–	+
Нерівнозначність	$x_1 \oplus x_2$	$\overline{x}_1x_2 \vee x_1\overline{x}_2$	+	–	–	–	+
Диз'юнкція	$x_1 \vee x_2$	$x_1 \vee x_2$	+	+	–	+	–
Кон'юнкція	$x_1 \cdot x_2$	$x_1 \cdot x_2$	+	+	–	+	–
Імплікація	$x_1 \to x_2$	$\overline{x}_1 \vee x_2$	–	+	–	–	–
Заборона	$x_1 \Delta x_2$	$x_1 \cdot \overline{x}_2$	+	–	–	–	–
Стрілка Пірса	$x_1 \downarrow x_2$	$\overline{x}_1 \cdot \overline{x}_2$	–	–	–	–	–
Функція Шеффера	$x_1 \mid x_2$	$\overline{x}_1 \vee \overline{x}_2$	–	–	–	–	–

Означення 5.5.3. Систему логічних функцій називають **функціонально повною в слабкому розумінні**, якщо будь-яка логічна функція може бути зображена суперпозицією функцій із множини, що отримана шляхом додавання до множини системи констант 0 і 1.

З критерію Поста для означення 5.5.1 випливає, що для функціональної повноти системи функцій у слабкому розумінні достатньо, щоб у системі містилися нелінійна та немонотонна функції, оскільки решта необхідних властивостей присутня у констант 0 і 1, табл. 5.5.4.

Функціональна повнота системи означає можливість реалізувати будь-яку логічну функцію при синтезі логічних елементів за допомогою елементів відповідної повної системи. Константи 0 і 1 не потребують спеціальних елементів для реалізації, тому часто їх вважають даними. В такому випадку систему типів логічних елементів досліджують на функціональну повноту в слабкому розумінні.

Приклад 5.5.2. За допомогою критерію Поста визначити функціональну повноту систем $< \bar{}, \Delta >$, $< \bar{}, \to >$, $< \bar{}$, змінна $>$.

Розв'язання. Користуючись даними табл. 5.5.4, побудуємо табл. 5.5.5, у якій для кожної системи знаком «+» відмітимо наявність відповідних властивостей у функції, а знаком «–» — їх відсутність

Таблиця 5.5.5

Назва властивості	$\bar{x}$	$x \Delta y$	$\bar{x}$	$x \to y$	$\bar{x}$	x
Зберігання 0	–	+	–	–	–	+
Зберігання 1	–	–	–	+	–	+
Самодвоїстість	+	–	+	–	+	+
Монотонність	–	–	–	–	–	+
Лінійність	+	–	+	–	+	+

Із табл. 5.5.5 випливає, що для систем $< \bar{}, \Delta >$, $< \bar{}, \to >$ у кожному рядку присутній знак «–», тому згідно з критерієм Поста такі системи логічних функцій є функціонально повними. У системі $< \bar{}$, змінна $>$ при самодвоїстості і лінійності знак «–» відсутній, і тому згідно з критерієм Поста така система логічних функцій не є функціонально повною.

Контрольні запитання

1. Дайте визначення двоїстої функції.
2. Яку функцію називають самодвоїстою?
3. Сформулюйте принципи двоїстості.
4. Яким чином формується таблиця істинності двоїстої функції?
5. Сформулюйте правило отримання аналітичним шляхом із заданою формулою функції, двоїстої до неї.
6. Дайте визначення логічних функцій, що зберігають нуль та одиницю.
7. Як за таблицею істинності функції визначити, чи зберігає вона нуль та одиницю?
8. Назвіть функції двох змінних, які зберігають нуль та одиницю.
9. Дайте визначення монотонності логічної функції.
10. Якщо функція зображена формулою алгебри логіки із запереченням, то f монотонна чи ні? Відповідь обґрунтуйте.
11. Чи можливо за видом ДНФ логічної функції стверджувати про її монотонність?
12. Яку логічну функцію називають лінійною?
13. Яку логічну функцію називають нелінійною?
14. Сформулюйте означення про замкнені класи логічних функцій?
15. Скільки замкнених класів логічних функцій існує? Назвіть їх.
16. Сформулюйте критерії Поста про повноту системи логічних функцій.
17. Яка повна система логічних функцій називається нескоротною?
18. Дайте визначення функціонально повної у слабкому розумінні системи логічних функцій.

Задачі для самостійного розв'язування

1. Знайти двоїсті формули до таких функцій:
   а) $x \cdot y \vee \overline{y} \cdot z \vee \overline{x} \cdot z$;
   б) $x \cdot (\overline{y} \vee z) \vee (x \vee \overline{y} \vee \overline{z})$;
   в) $(x \vee \overline{y}) \cdot z \vee y \vee (\overline{x} \cdot y \vee z)$.
2. Визначити самодвоїстість у наступних функціях:
   а) $f(x, y) = x \vee \overline{(x \cdot y \vee \overline{y})}$;
   б) $f(x, y) = (x \vee \overline{y}) \cdot (\overline{x} \vee y)$;
   в) $f(x, y, z) = \overline{x} \cdot \overline{y} \vee x \cdot z \vee \overline{y} \cdot \overline{z}$;
   г) $f(x, y, z) = x \cdot \overline{y} \vee x \cdot \overline{z} \vee y \cdot \overline{z}$.

3. В наведених нижче функціях визначити, чи зберігають вони 0 чи 1:

а) $x \cdot y \vee \overline{x} \cdot \overline{z} \vee \overline{x} \cdot \overline{y} \cdot \overline{z}$; б) $(x \to y) \vee (x \sim y)$; в) $(x \oplus y) \to \overline{x} \cdot \overline{y}$.

4. Дослідити такі функції на монотонність:

а) $x \cdot y \sim y$; б) $(x \,\Delta\, y) \to x$; в) $x \cdot y \oplus x \oplus y$;

г) $(x \downarrow y) \,\Delta\, z$; д) $(x \to y) \to z$; є) $(x \vee y) \to (x \mid y)$.

5. Дослідити на лінійність такі логічні функції:

а) $(x \to y) \mid \overline{(x \,\Delta\, \overline{y})}$; б) $(x \oplus y) \to (\overline{x} \vee z)$;

в) $(x \downarrow y) \vee (\overline{x} \oplus y) \to (y \vee \overline{z})$; г) $(x \mid y) \to (y \to x) \to (y \vee \overline{z})$;

д) $(x \sim y) \to (z \mid y) \,\Delta\, (x \downarrow y)$.

6. За допомогою критерію Поста перевірити на функціональну повноту такі системи:

а) $<0, 1, \overline{\ }>$; б) $<0, 1, \sim>$; в) $<\sim, 1, \vee>$;

г) $<\cdot, \oplus, 1>$; д) $<\leftrightarrow, \Delta>$; є) $<1, \oplus, \overline{\ }>$;

ж) $<\leftrightarrow, \overline{\ }>$; з) $<\vee, \cdot, \overline{\ }>$; і) $<\vee, \oplus>$.

7. Довести функціональну повноту функції Шеффера і стрілки Пірса.

Коментарі

У даному розділі дослідження на двоїстість, монотонність, лінійність, а також дослідження на замкнутість і повноту логічних функцій взяті з [8, 21].

Розділ 6

ЛОГІКА МІНІМІЗАЦІЇ БУЛЕВИХ ФУНКЦІЙ

6.1. Основні визначення

Задача мінімізації складається з пошуку найпростішої формули логічної функції за обраним критерієм. У якості критерію може виступати кількість змінних у формулі, кількість знаків кон'юнкції та диз'юнкції або комбінація подібних критеріїв.

У цьому розділі розглянуті різні методи логіки мінімізації на множині ДНФ і КНФ. Мінімальні форми, які отримані в процесі мінімізації називають мінімальними ДНФ і КНФ.

Означення 6.1.1. **Імплікантою** деякої логічної функції f називають таку функцію φ, яка на всіх інтерпретаціях, на яких вона дорівнює одиниці, f теж дорівнює одиниці.

Елементарні кон'юнкції, що входять до складу ДНФ функції, також є її імплікантами.

Означення 6.1.2. **Простою імплікантою** логічної функції f називають таку імпліканту, що ніяка її власна частина не є імплікантою даної функції.

Означення 6.1.3. **Скороченою ДНФ** називають диз'юнкцію всіх простих імплікант логічної функції.

Означення 6.1.4. **Тупиковою ДНФ** називають ДНФ даної логічної функції, яка складається тільки з простих імплікант.

На відміну від скороченої ДНФ, тупикова ДНФ може не містити деякі із простих імплікант функції. Кожна логічна функція має єдину скорочену ДНФ і може мати декілька тупикових ДНФ.

Означення 6.1.5. **Мінімальною ДНФ(МДНФ)** логічної функції називають одну з тупикових ДНФ, якій відповідає найменше значення критерію мінімізації ДНФ.

Якщо логічна функція задана ДНФ, то для знаходження її простих імплікант використовують такі операції для перетворення формул алгебри логіки.

Операція **неповного диз'юнктивного склеювання:**

$$A \cdot x \vee A \cdot \bar{x} = A \vee A \cdot x \vee A \cdot \bar{x}.$$

Операція **диз'юнктивного поглинання:**

$$A \vee A \cdot x = A$$

Операція **повного диз'юнктивного склеювання:**

$$A \cdot x \vee A \cdot \bar{x} = A,$$

де A — певна елементарна кон'юнкція змінних, а x — логічна змінна.

Приклад 6.1.1. Виконати операції повного склеювання логічної функції, заданої в ДДНФ різними способами:

$$f(x, y, z) = x \cdot y \cdot z \vee \bar{x} \cdot y \cdot z \vee \bar{x} \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot \bar{z}.$$

Розв'язання. Виконаємо операції повного склеювання першим способом:

$$f(x, y, z) = x \cdot y \cdot z \vee \bar{x} \cdot y \cdot z \vee \bar{x} \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot \bar{z} = (x \cdot y \cdot z \vee \bar{x} \cdot y \cdot z) \vee$$
$$\vee (\bar{x} \cdot y \cdot z \vee \bar{x} \cdot \bar{y} \cdot z) \vee (\bar{x} \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot \bar{z}) = y \cdot z \vee \bar{x} \cdot z \vee \bar{x} \cdot \bar{y}.$$

Тепер виконаємо операції склеювання іншим способом, інакше компонуючи імпліканти:

$$f(x, y, z) = (x \cdot y \cdot z \vee \bar{x} \cdot y \cdot z) \vee (\bar{x} \cdot \bar{y} \cdot \bar{z} \vee \bar{x} \cdot \bar{y} \cdot z) = y \cdot z \vee \bar{x} \cdot \bar{y}.$$

В результаті склеювання отримані дві різні тупикові ДНФ логічної функції $f(x, y, z)$. Друга тупикова ДНФ простіша за першу, оскільки містить меншу кількість символів змінних і знаків операцій.

Означення 6.1.6. **Імпліцентою** певної логічної функції f називають таку функцію φ, яка на всіх інтерпретаціях, на яких вона дорівнює нулю, f теж дорівнює нулю.

Елементарні диз'юнкції, що входять до складу КНФ функції, також є її імпліцентами.

Означення 6.1.7. **Простою імпліцентою** логічної функції f називають таку імпліценту, що ніяка її власна частина не є імпліцентою даної функції.

Означення 6.1.8. **Скороченою КНФ** називають кон'юнкцію всіх простих імпліцент логічної функції.

Означення 6.1.9. **Тупиковою КНФ** називають КНФ даної логічної функції, яка складається тільки з простих імпліцент.

Означення 6.1.10. **Мінімальною КНФ (МКНФ)** логічної функції називають одну із тупикових КНФ, якій відповідає найменше значення критерію мінімізації КНФ.

Для мінімізації КНФ використовують такі операції склеювання.

Операція **неповного кон'юктивного склеювання:**

$$(A \vee x) \cdot (A \vee \bar{x}) = A \cdot (A \vee x) \cdot (A \vee \bar{x}).$$

Операція **кон'юктивного поглинання:**

$$A \cdot (A \vee x) = A.$$

Операція **повного кон'юктивного поглинання:**

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

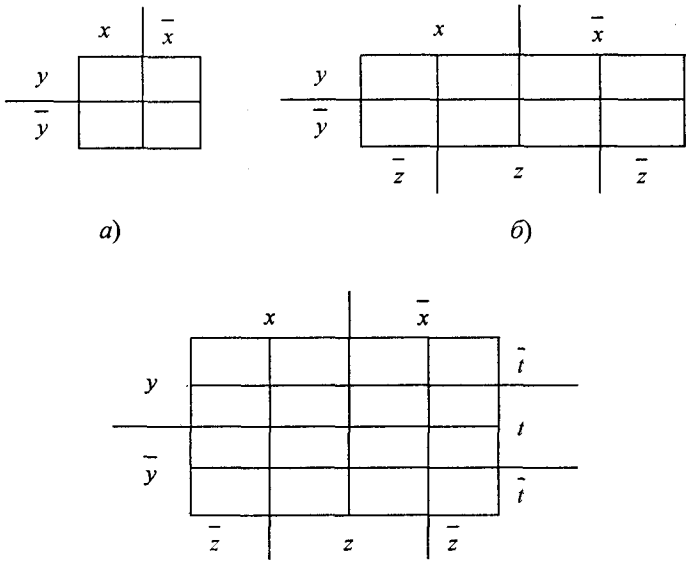
де A — деяка елементарна диз'юкція, а x — логічна змінна.

6.2. Метод Вейча

Мінімізацію логічних функцій методом Вейча застосовують для функцій $f(x_1, x_2, ..., x_n)$, які містять, як правило, **не більше чотирьох змінних** і повинні бути задані в аналітичній формі у вигляді ДДНФ або ДКНФ. Однак, цей метод може застосовуватись і для більшого числа змінних.

У методі Вейча для мінімізації використовують таблиці, які являють собою прямокутник, що вміщує n клітинок, до яких заносять одиниці при мінімізації логічних функцій, які задані в ДДНФ або нулі, у випадку мінімізації логічних функцій, поданих у ДКНФ. Якщо функція записана в ДНФ або КНФ, то її слід попередньо перевести до ДДНФ або ДКНФ.

Структури таблиць Вейча для двох, трьох і чотирьох логічних змінних приведені на рис. 6.2.1.



а) *б*) *в*)

Рис. 6.2.1

Як випливає із рис. 6.2.1, структура таблиці для двох змінних має вид таблиці 2×2, для трьох — 2×4, а для чотирьох — 4×4.

Із таблиць випливає, що кожна змінна і її заперечення містяться з одного боку таблиці. За такого розміщення дві різні змінні із запереченням чи без покривають 2^{n-2} клітинки. Такі покриття відповідають логічним добуткам n змінних — конституентам одиниці функції. Наприклад, структура таблиці Вейча з відповідними мінтернами і абстрактними змінними в клітинках для трьох зміних показана на рис. 6.2.2.

	x		$\bar{x}$	
y	$x \cdot y \cdot \bar{z}$	$x \cdot y \cdot z$	$\bar{x} \cdot y \cdot z$	$\bar{x} \cdot y \cdot \bar{z}$
$\bar{y}$	$x \cdot \bar{y} \cdot \bar{z}$	$x \cdot \bar{y} \cdot z$	$\bar{x} \cdot \bar{y} \cdot z$	$\bar{x} \cdot \bar{y} \cdot \bar{z}$
	$\bar{z}$	z		$\bar{z}$

Рис. 6.2.2

Із рис. 6.2.2 випливає, що важливою властивістю цих мінтерн є те, що ті з них, які належать до сусідніх клітинок і до клітинок, що містяться з краю одних і тих же рядків і стовпчиків таблиці, дозволяє здійснювати мінімізацію функції безпосередньо за таблицею в наочній формі.

Для того, щоб занести ДДНФ функції $f(x_1, x_2, ..., x_n)$ в таблицю Вейча необхідно розмістити одиниці в клітинках, що відповідають її конституентам одиниці. Якщо одиниці розташовані в сусідніх клітинках, наприклад, таких, що відповідають добуткам $x \cdot y \cdot z$ і $\bar{x} \cdot y \cdot z$, то внаслідок того, що вони відрізняються знаком заперечення лише в одній змінній (у даному випадку x), відбувається операція склеювання за змінною x. Результатом склеювання для цього випадку буде $x \cdot y \cdot z \vee \bar{x} \cdot y \cdot z = y \cdot z\,(x \vee \bar{x}) = y \cdot z$. Дві змінні y і z покривають спільно разом дві клітинки.

Якщо одиниці розташовані в чотирьох клітинках, то це означає, що відбувається склеювання чотирьох сусідніх конституент за стовпчиками і рядками. У результаті цього буде отримана одна змінна, яка покриває всі чотири сусідні клітинки. Тобто, склеювання як у першому, так і у другому випадках проводиться графічно за допомогою об'єднання клітинок у групи.

Склеювання клітинок у таблиці Вейча і отримання мінімальної ДНФ відбувається за таким правилом.

1. Клітинки об'єднуються у групи, що позначають операції склеювання. В об'єднанні беруть участь тільки ті сусідні клітинки, в яких містяться одиниці.

2. В групу дозволяється об'єднувати кількість клітинок 2^n, $n = 1, 2, 3...$ При цьому група може мати лише прямокутну або квадратну форму.

3. При склеюванні необхідно знайти набір максимальних груп клітинок. Під максимальною групою розуміють групу, яка не входить цілком у жодну іншу групу і відповідає простій імпліканті функції. Кількість груп у такому наборі повинна бути мінімальною, оскільки така група відповідає мінімальній тупиковій ДНФ. Кожна одиниця таблиці Вейча повинна входити хоча б до однієї групи, що забезпечує покриття функції отриманим набором імплікант.

4. Кожна група клітинок, що отримана після склеювання, відповідає тій імпліканті функції, реальні змінні якої мають однакове значення для всіх клітинок групи.

5. Диз'юнкція всіх отриманих простих імплікант зображує результат мінімізації формули і є мінімальною ДНФ.

Оскільки при мінімізації на множині ДНФ у склеюванні беруть участь тільки клітинки, які містять одиниці, то нулі в таблиці Вейча, як правило, не вказують, а мають на увазі, що порожні клітинки містять нулі.

Приклад 6.2.1. Знайти МДНФ логічної функції

$$f(x, y, z) = x \cdot y \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z} \vee x \cdot \bar{y} \cdot z \vee \bar{x} \cdot y \cdot \bar{z} \vee \bar{x} \cdot \bar{y} \cdot \bar{z}.$$

Розв'язання. У відповідності з рис 6.2.1б, будуємо таблицю Вейча, до якої заносимо одиниці згідно із заданою логічною функцією, рис. 6.2.3.

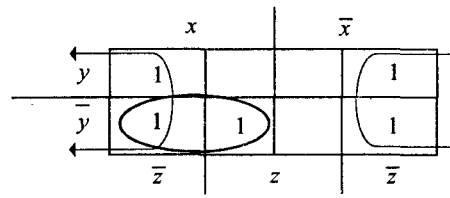


Рис. 6.2.3

Виконуємо склеювання клітинок таблиці відповідно з розглянутими вище правилами. При склеюванні необхідно об'єднати чотири клітинки, де містяться чотири одиниці, що перебувають у двох клітинках першого і останнього стовпчиків, які накриваються змінною $\bar{z}$, а одиниця, яка залишилась у другому стовпчику, склеюється з одиницею першого стовпчика нижнього рядка таблиці. У результаті цього отримаємо ДНФ логічної функції $f(x, y, z) = \bar{z} \vee x \cdot \bar{y}$.

Для мінімізації логічної функції $f(x_1, x_2, ..., x_n)$, поданої у ДКНФ, таблицю Вейча заповнюють нулями, які заносять у клітинки, що відповідають логічним сумам, на яких функція f дорівнює нулю. У всьому іншому процедура мінімізації відбувається за тими ж правилами, що були розглянуті для логічної функції, поданої у ДНФ.

Приклад 6.2.2. Знайти МКНФ логічної функції

$$f(x, y, z) = (x \vee y \vee z) \cdot (\bar{x} \vee y \vee z) \cdot (x \vee \bar{y} \vee \bar{z}) \cdot (\bar{x} \vee \bar{y} \vee \bar{z}).$$

Розв'язання. У відповідності з рис. 6.2.1б будуємо таблицю Вейча, до якої заносимо нулі згідно із заданою логічною функцією, рис. 6.2.4.

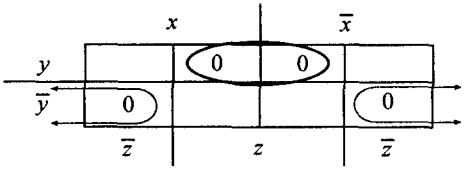


Рис. 6.2.4

Для отримання МКНФ цієї функції слід склеїти нулі, які розміщені поряд, і нулі, які розміщені в лівому і правому нижньому углу таблиці. Окреслені суцільними лініями по два нулі, покриваються відповідними логічними сумами $y \vee z$ і $\bar{y} \vee \bar{z}$. Звідси випливає, що МКНФ заданої функції матиме вигляд

$$f(x, y, z) = (y \vee z) \cdot (\bar{y} \vee \bar{z}).$$

Застосування методу Вейча для часткового визначених логічних функцій розглянемо на прикладі.

Приклад 6.2.3. Знайти МДНФ логічної функції

$$f(x, y, z, t) = x \cdot \bar{y} \cdot \bar{z} \cdot \bar{t} \vee \bar{x} \cdot \bar{y} \cdot \bar{z} \cdot \bar{t} \vee x \cdot y \cdot \bar{z} \cdot \bar{t},$$

яка невизначена при $\bar{x} \cdot y = 1$.

Розв'язання. У відповідності з рис. 6.2.1в, будуємо таблицю Вейча, до якої згідно із заданою логічною функцією заносимо одиниці і її невизначеність при $\bar{x} \cdot y = 1$, рис. 6.2.5

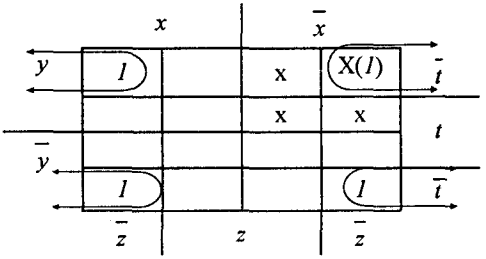


Рис. 6.2.5

де x — знак невизначеності функції.

За рахунок невизначеності функції при $\bar{x} \cdot y = 1$ в крайню праву верхню клітинку ставимо одиницю. Виконуємо склеювання клітинок таблиці відповідно з розглянутими правилами. У результаті цього отримаємо МДНФ частково визначеної логічної функції

$$f(x, y, z, t) = y \cdot \bar{z} \cdot \bar{t} \vee \bar{y} \cdot \bar{z} \cdot \bar{t} = \bar{z} \cdot \bar{t}\left(y \vee \bar{y}\right) = \bar{z} \cdot \bar{t}.$$

Мінімізація частково визначених КНФ логічних функцій відбувається аналогічно мінімізації частково визначених ДНФ.

Метод Вейча може використовуватись і для більшого числа змінних. Так, наприклад, на рис. 6.2.6 приведена розмітка таблиці Вейча при її використанні для мінімізації логічних функцій, які мають п'ять змінних.

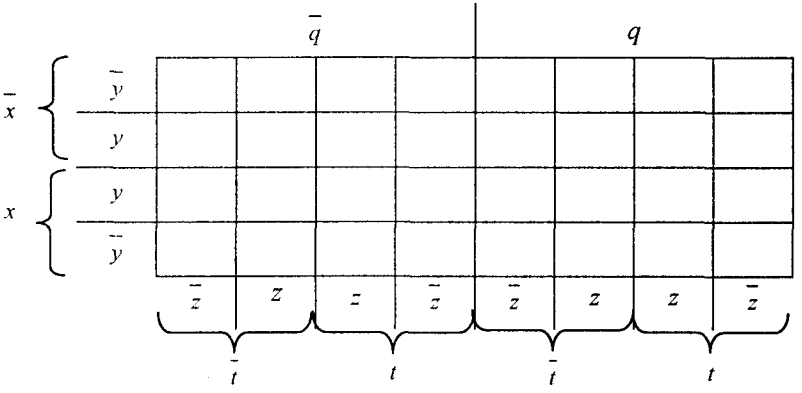
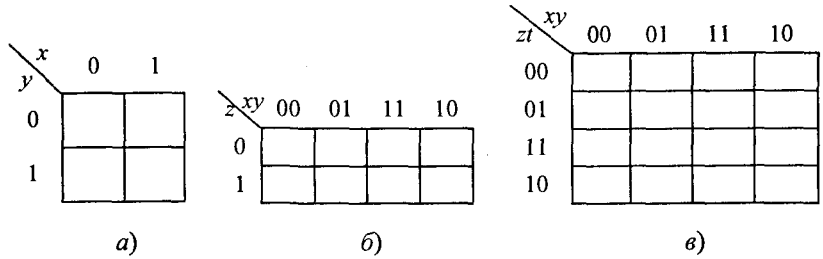


Рис. 6.2.6

Вона має 32 клітинки для розміщення конституент одиниці при мінімізації ДНФ або конституент нуля при мінімізації КНФ. Аналогічно будують таблиці Вейча для мінімізації логічних функцій, у яких кількість змінних більша п'яти. Вони матимуть 64 клітинки для шести змінних, 128 кітинок для 7 змінних і т.д.

6.3. Метод Карно

Цей метод був запропонований у 1953 році Карно після раніше опублікованого методу Вейча. За своєю суттю він повторює метод Вейча, але має дещо відмінну структуру таблиць за формою, які приведені на рис. 6.3.1.



а) *б*) *в*)

Рис. 6.3.1

На рис. 6.3.1 *а* показана таблиця Карно для двох змінних, на рис. 6.3.1 *б* — для трьох, а на рис. 6.3.1 *в* — для чотирьох змінних. Для двох змінних структура таблиці Карно має вигляд 2×2, для трьох — 2×4, а для чотирьох — 4×4.

На відміну від таблиці Вейча, в таблицях Карно значення змінних розташовані у заголовках рядків і стовпчиків. Кожній конституенті *одиниці або нуль функції відповідає одна клітинка* таблиці. Нуль або одиниця в клітинці показує значення функції на даній інтерпретації. Значення змінних розташовані так, щоб сусідні рядки і стовпчики таблиці відрізнялися значенням тільки однієї змінної, тобто так само, як і у таблиці Вейча. На відміну від таблиці Вейча, для інтерпретації змінних у таблицях Карно застосовуються двійкові послідовності (0, 0), (0, 1), (1,1), (1,0). У цій послідовності перша та остання інтерпретації також відрізняються значенням тільки однієї змінної, тому перший і останній рядки (стовпчики) вважаються сусідніми. При такому розташуванні мінтерми, до яких застосована операція склеювання, розташовуються у сусідніх клітинках таблиці, і склеювання проводиться графічно за допомогою об'єднання клітинок у групи. Так, наприклад, структура таблиці Карно з відповідними мінтернами і абстрактними змінними в клітинках для трьох змінних показана на рис. 6.3.2.

zt \ xy	00	01	11	10
0	$\bar{x}\cdot\bar{y}\cdot\bar{z}$	$\bar{x}\cdot y\cdot\bar{z}$	$x\cdot y\cdot\bar{z}$	$x\cdot\bar{y}\cdot\bar{z}$
1	$\bar{x}\cdot\bar{y}\cdot z$	$\bar{x}\cdot y\cdot z$	$x\cdot y\cdot z$	$x\cdot\bar{y}\cdot z$

Рис. 6.3.2

Заповнення таблиць Карно для логічної функції аналогічне заповненню таблиць Вейча.

Всі правила склеювання клітинок і запису МДНФ і МКНФ в таблицях Карно аналогічні правилам склеюванням клітинок і запису МДНФ і МКНФ в таблицях Вейча.

Приклад 6.3.1. Знайти МДНФ для функції

$$F(x, y, z) = x \cdot y \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z} \vee x \cdot \bar{y} \cdot z \vee \bar{x} \cdot y \cdot \bar{z} \vee \bar{x} \cdot \bar{y} \cdot \bar{z}.$$

Розв'язання. У відповідності з рис 6.3.1б будуємо таблицю Карно, до якої заносимо одиниці згідно із заданою логічною функцією, рис. 6.3.3.

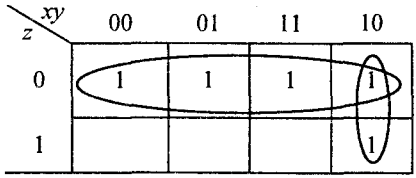


Рис. 6.3.3

Виконуємо склеювання клітинок таблиці у відповідності з розглянутими в § 6.2 правилами. При склеюванні об'єднаємо чотири клітинки, де містяться чотири одиниці першої стрічки, які накриваються змінною $\bar{z}$, а одиниця, яка залишилась в останньому стовпчику, склеюється з одиницею останнього стовпчика першої стрічки. У результаті цього отримаємо МДНФ логічної функції

$$f(x, y, z) = \bar{z} \vee x \cdot \bar{y}.$$

Приклад 6.3.2. Знайти МКНФ для функції

$$f(x, y, z) = (x \vee y \vee z) \cdot (\bar{x} \vee y \vee z) \cdot (x \vee \bar{y} \vee \bar{z}) \cdot (\bar{x} \vee \bar{y} \vee \bar{z}).$$

Розв'язання. У відповідності з рис 6.3.1б, будуємо таблицю Карно, до якої заносимо нулі згідно із заданою логічною функцією, рис. 6.3.4.

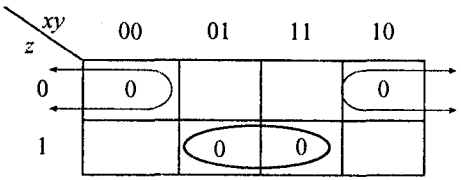


Рис. 6.3.4

Для отримання МКНФ функції необхідно склеїти нулі, які стоять поряд, і нулі, які розміщені в лівому і правому кутках першої стрічки таблиці. Окреслені суцільними лініями по два нулі покриваються відповідними логічними сумами $y \vee z$ і $\bar{y} \vee \bar{z}$. Звідси випливає, що МКНФ заданої логічної функції матиме вигляд

$$f(x, y, z) = (y \vee z) \cdot (\bar{y} \vee \bar{z}).$$

Застосування методу Карно для частково визначених логічних функцій аналогічне методу Вейча.

Приклад 6.3.3. Знайти МДНФ для частково визначеної логічної функції, яка не визначена, якщо $x \cdot y = 1$

$$f(x, y, z, t) = \bar{x} \cdot \bar{y} \cdot z \cdot \bar{t} \vee \bar{x} \cdot y \cdot z \cdot \bar{t} \vee x \cdot \bar{y} \cdot z \cdot \bar{t} \vee x \cdot \bar{y} \cdot \bar{z} \cdot \bar{t}.$$

Розв'язання. У відповідності з рис 6.3.1в, будуємо таблицю Карно, до якої згідно із заданою логічною функцією заносимо одиниці і її невизначеність при $x \cdot y = 1$, рис. 6.3.5.

zt \ xy	00	01	11	10
00			x(1)	1
01			x	
11			x	
10	1	1	x(1)	1

Рис. 6.3.5

Виконуємо склеювання клітинок таблиці у відповідності з розглянутими правилами в §.6.2. У результаті цього отримаємо МДНФ частково визначеної логічної функції

$$f(x, y, z, t) = z \cdot \bar{t} \vee x \cdot \bar{t}.$$

Мінімізація частково визначених КНФ логічних функцій відбувається аналогічно мінімізації частково визначених ДНФ.

Метод Карно може використовуватись і для більшого числа змінних. Так, наприклад на рис 6.3.6 приведена розмітка таблиць Карно при їх використанні для мінімізації логічних функцій, які мають п'ять змінних.

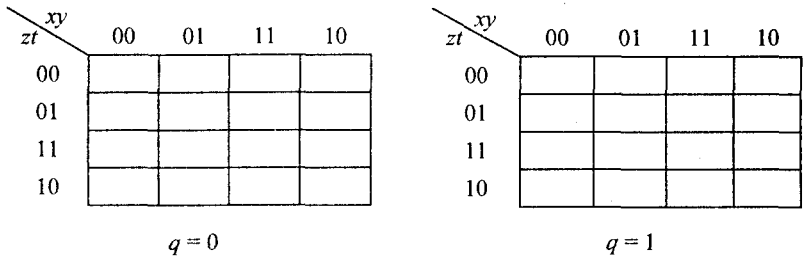


Рис. 6.3.6

Дана таблиця має дві таблиці. В одній із них значення п'ятої змінної $q = 0$, а в іншій – $q = 1$. В сумі вона має 32 клітинки для розміщення конституент одиниці при мінімізації ДНФ або конституент нуля при мінімізації КНФ. Склеювання клітинок таблиці відбувається у відповідності з розглянутими правилами в §.6.2.

6.4. Метод Квайна

Метод мінімізації Квайна реалізує перехід від ДДНФ (ДКНФ) до ДНФ (КНФ) з використанням операцій склеювання та поглинання.

Основу алгоритму Квайна складають такі кроки.

1. Записати ДДНФ (ДКНФ) заданої функції.

2. Виконати всі можливі операції неповного диз'юнктивного (кон'юктивного) склеювання. Отримана формула є диз'юнкцією (кон'юнкцією) всіх можливих імплікант (імпліцент) даної функції.

3. Виконати всі можливі операції диз'юнктивного (кон'юнктивного) поглинання. Результуюча формула є скороченою ДНФ (КНФ) даної функції.

4. Скласти імплікантну (імпліцентну) таблицю і шляхом її покриття за рахунок мінімальної кількості простих імплікант (імпліцент) знайти всі тупикові ДНФ (КНФ) даної логічної функції.

5. Серед тупикових ДНФ (КНФ) знайти мінімальну ДНФ (КНФ) логічної функції.

Приклад 6.4.1. Знайти методом Квайна МДНФ функції

$$f(x, y, z) = x \cdot y \cdot z \vee x \cdot y \cdot \bar{z} \vee \bar{x} \cdot y \cdot z \vee \bar{x} \cdot \bar{y} \cdot z \vee x \cdot \bar{y} \cdot \bar{x} \vee \bar{x} \cdot \bar{y} \cdot \bar{z}.$$

Розв'язання. Виконаємо всі можливі операції диз'юнктивного склеювання і поглинання

$$x \cdot y \cdot z \vee x \cdot y \cdot \bar{z} = x \cdot y;$$
$$x \cdot y \cdot z \vee \bar{x} \cdot y \cdot z = y \cdot z;$$
$$x \cdot y \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z} = x \cdot \bar{z};$$
$$\bar{x} \cdot y \cdot z \vee \bar{x} \cdot \bar{y} \cdot z = \bar{x} \cdot z;$$
$$\bar{x} \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot \bar{z} = \bar{x} \cdot \bar{y};$$
$$x \cdot \bar{y} \cdot \bar{z} \vee \bar{x} \cdot \bar{y} \cdot \bar{z} = \bar{y} \cdot \bar{z}.$$

В результаті цього отримаємо формулу

$$f(x,y,x) = x \cdot y \cdot z \vee x \cdot y \cdot \bar{z} \vee \bar{x} \cdot y \cdot x \vee \bar{x} \cdot \bar{y} \cdot z \vee x \cdot \bar{y} \cdot \bar{z} \vee \bar{x} \cdot \bar{y} \cdot \bar{z} =$$
$$= x \cdot y \vee y \cdot z \vee x \cdot \bar{z} \vee \bar{x} \cdot z \vee \bar{x} \cdot \bar{y} \vee \bar{y} \cdot \bar{z}.$$

Із аналізу формули випливає, що отримані інші елементарні кон'юнкції не підлягають подальшому склеюванню, і тому диз'юнкція всіх можливих імплікант даної функції матиме вигляд

$$f(x, y, z) = x \cdot y \vee y \cdot z \vee x \cdot \bar{z} \vee \bar{x} \cdot z \vee \bar{x} \cdot y \vee \bar{y} \cdot \bar{z}.$$

Отримана формула є скороченою ДНФ заданої функції. Для знаходження МДНФ складемо імплікантну таблицю, табл. 6.4.1.

Таблиця 6.4.1

•	$x \cdot y \cdot z$	$x \cdot y \cdot \bar{z}$	$\bar{x} \cdot y \cdot z$	$\bar{x} \cdot \bar{y} \cdot z$	$x \cdot \bar{y} \cdot \bar{z}$	$\bar{x} \cdot \bar{y} \cdot \bar{z}$
$x \cdot y$	*	*				
$y \cdot z$	*		*			
$x \cdot \bar{z}$		*			*	
$\bar{x} \cdot z$			*	*		
$\bar{x} \cdot \bar{y}$				*		*
$\bar{y} \cdot \bar{z}$					*	*

Рядки заданої таблиці є прості імпліканти, а стовпчики–конституентами одиниці досліджуваної функції. Зірочкою в таблиці по-

значається кожна клітинка, для якої імпліканта рядка є власною частиною конституенти із стовпчика. Користуючись кроком чотирі алгоритму, шляхом покриття за рахунок мінімальної кількості простих імплікант, знаходимо дві тупикові ДНФ, які мають такий вид:

ДНФ1: $f(x, y, z) = x \cdot y \vee \bar{x} \cdot z \vee \bar{y} \cdot \bar{z}$;

ДНФ2: $f(x, y, z) = y \cdot z \vee x \cdot \bar{z} \vee \bar{x} \cdot \bar{y}$.

Вказані ДНФ містять по 6 символів змінних, а також однакову кількість знаків операцій диз'юнкції (2), кон'юнкції (3) і заперечення (3). В якості МДНФ можна вибрати ДНФ1 або ДНФ2.

Приклад 6.4.2 Знайти методом Квайна МКНФ функції

$$f(x, y, z) = (x \vee y \vee z) \cdot (x \vee y \vee \bar{z}) \cdot (\bar{x} \vee y \vee z) \cdot (\bar{x} \vee \bar{y} \vee z) \cdot (x \vee \bar{y} \vee \bar{z}) \cdot (\bar{x} \vee \bar{y} \vee \bar{z}).$$

Розв'язання. Виконаємо всі можливі операції кон'юнктивного склеювання і поглинання:

$$(x \vee y \vee x) \cdot (x \vee y \vee \bar{x}) = (x \vee y);$$
$$(x \vee y \vee z) \cdot (\bar{x} \vee y \vee z) = (y \vee z);$$
$$(x \vee y \vee \bar{z}) \cdot (x \vee \bar{y} \vee \bar{z}) = (x \vee \bar{z});$$
$$(\bar{x} \vee y \vee z) \cdot (\bar{x} \vee \bar{y} \vee z) = (\bar{x} \vee z);$$
$$(\bar{x} \vee \bar{y} \vee z) \cdot (\bar{x} \vee \bar{y} \vee \bar{z}) = (\bar{x} \vee \bar{y});$$
$$(x \vee \bar{y} \vee \bar{z}) \cdot (\bar{x} \vee \bar{y} \vee \bar{z}) = (\bar{y} \vee \bar{z}).$$

В результаті цього отримаємо формулу

$$f(x, y, z) = (x \vee y \vee z) \cdot (\bar{x} \vee y \vee \bar{z}) \cdot (\bar{x} \vee y \vee z) \cdot (\bar{x} \vee \bar{y} \vee z) \cdot (z \vee \bar{y} \vee \bar{z}) \cdot (\bar{x} \vee \bar{y} \vee \bar{z}) =$$
$$= (x \vee y) \cdot (y \vee z) \cdot (x \vee \bar{z}) \cdot (\bar{x} \vee z) \cdot (\bar{x} \vee \bar{y}) \cdot (\bar{y} \vee \bar{x}).$$

Із аналізу формули випливає, що отримані інші елементи диз'юнкції не підлягають подальшому склеюванню, і тому кон'юнкція всіх можливих імпліцент даної функції матиме вигляд

$$f(x, y, z) = (x \vee y) \cdot (y \vee z) \cdot (x \vee \bar{z}) \cdot (\bar{x} \vee z) \cdot (\bar{x} \vee \bar{y}) \cdot (\bar{y} \vee \bar{z}).$$

Отримана формула є скороченою КНФ заданої функції. Для знаходження МКНФ складемо імпліценту таблицю, табл. 6.4.2.

Таблиця 6.4.2

	$x \vee y \vee z$	$x \vee y \vee \bar{z}$	$\bar{x} \vee y \vee z$	$\bar{x} \vee \bar{y} \vee z$	$x \vee \bar{y} \vee \bar{z}$	$\bar{x} \vee \bar{y} \vee \bar{z}$
$x \vee y$	*	*				
$y \vee z$	*		*			
$x \vee \bar{z}$		*			*	
$\bar{x} \vee z$			*	*		
$\bar{x} \vee \bar{y}$				*		*
$\bar{y} \vee \bar{z}$					*	*

Рядки заданої таблиці є прості імпліценти, а стовпчики — конституентами нуля. Зірочкою в таблиці позначають кожну клітинку, для якої імпліцента рядка є власною частиною конституенти із стовпчика. Користуючись кроком чотири алгоритму, шляхом покриття за рахунок мінімальної кількості простих імпліцент знаходимо дві тупикові КНФ, які мають такий вигляд:

$$\text{КНФ1: } f(x, y, z) = (x \vee y) \cdot (\bar{x} \vee z) \cdot (\bar{y} \vee \bar{z});$$

$$\text{КНФ2: } f(x, y, z) = (y \vee z) \cdot (x \vee \bar{z}) \cdot (\bar{x} \vee \bar{y}).$$

Вказані КНФ містять по 6 символів змінних, а також однакову кількість знаків операцій кон'юнкції (2), диз'юнкції (3) і заперечення (3). В якості МКНФ можна вибрати КНФ1 або КНФ2.

6.5. Метод Мак-Класкі

У методі Квайна є істотна незручність, пов'язана з необхідністю повного попарного порівняння простих імплікант (імпліцент) на етапі їх визначення. Зі зростанням кількості мінтермів застосування методу Квайна стає складним. У 1956 р. Мак-Класкі запропонував модернізацію, яка істотно зменшує кількість порівнянь мінтермів.

Ідея її полягає в тому, що мінтерми записують у вигляді їх двійкових номерів, а всі номери розбивають за числом одиниць у них на непересічні групи. При цьому до i-ої групи ввійдуть усі номери, що мають у своєму двійковому запису i одиниць. Попарно порівнювати можна тільки сусідні за номером групи, оскільки саме

вони різні для тих мінтермів, що входять до них в одному розряді. Під час створення мінтермів із рангом, вищим від нульового, до розрядів, що відповідають вилученим змінним, записують знак "*тире*". Така модифікація на практиці дає змогу уникнути виписування громіздких мінтермів, залишаючи це їх двійковим номерам.

Приклад 6.5.1. Знайти за допомогою методу Мак-Класкі МДНФ функції $f(x,y,z,t)$, що задана такою ДДНФ

$$f(x,y,z,t)=\bar{x}\cdot\bar{y}\cdot\bar{z}\cdot\bar{t}\vee\bar{x}\cdot\bar{y}\cdot\bar{z}\cdot t\vee\bar{x}\cdot\bar{y}\cdot z\cdot\bar{t}\vee\bar{x}\cdot\bar{y}\cdot z\cdot t\vee\bar{x}\cdot y\cdot\bar{z}\cdot t\vee$$
$$\vee\bar{x}\cdot y\cdot z\cdot t\vee x\cdot\bar{y}\cdot\bar{z}\cdot\bar{t}\vee x\cdot\bar{y}\cdot z\cdot\bar{t}\vee x\cdot\bar{y}\cdot z\cdot t.$$

Розв'язання. Для знаходження МДНФ заданої функції за методом Мак-Класкі необхідно використати такий алгоритм.

1. Записати конституенти одиниці заданої функції у вигляді двійкового коду.

2. Згрупувати двійкові коди імплікант з однаковою кількістю одиниць і присвоїти їм індекс групи k.

3. Починаючи з $k=0$, зробити порівняння кожного двійкового коду в групі з індексом k з кожним кодом з групи з індексом $k+1$. Якщо порівнювані двійкові коди різні тільки в одному розряді, то у наступний стовпчик таблиці записати відповідний їм двійковий код з порожньою позначкою «–» на місці зазначеного розряду. Напроти кожного нового коду записати номери кодів двох імплікант, які брали участь у порівнянні, і у наступному стовпчику дані імпліканти помітити знаком «*V*» — для непростих і знаком «*» — для простих імплікант.

5. Серед однакових отриманих імплікант залишити тільки одну.

6. Циклічно повторювати кроки алгоритму 2...5 доти, доки існує можливість отримувати нові коди імплікант.

У відповідності з кроком 1 алгоритму, запишемо конституенти одиниці заданої функції $f(x,y,z,t)$ у вигляді двійкового коду і занесемо в табл. 6.5.1.

Таблиця 6.5.1

Десяткові номери інтерпретацій	Двійкові коди конституент одиниці	Конституенти одиниці функції $f(x,y,z,t)$
0	0 0 0 0	$\bar{x}\cdot\bar{y}\cdot\bar{x}\cdot\bar{t}$
1	0 0 0 1	$\bar{x}\cdot\bar{y}\cdot\bar{z}\cdot t$
2	0 0 1 0	$\bar{x}\cdot\bar{y}\cdot z\cdot\bar{t}$

Закінчення табл. 6.5.1

Десяткові номери інтерпретацій	Двійкові коди конституент одиниці	Конституенти одиниці функції $f(x, y, z, t)$
3	0 0 1 1	$\bar{x} \cdot \bar{y} \cdot z \cdot t$
5	0 1 0 1	$\bar{x} \cdot y \cdot \bar{z} \cdot t$
7	0 1 1 1	$\bar{x} \cdot y \cdot z \cdot t$
8	1 0 0 0	$x \cdot \bar{y} \cdot \bar{z} \cdot \bar{t}$
10	1 0 1 0	$x \cdot \bar{y} \cdot z \cdot \bar{t}$
11	1 0 1 1	$x \cdot \bar{y} \cdot z \cdot t$

Виконуючи кроки алгоритму 2…6, заносимо дані в табл. 6.5.2.

Таблиця 6.5.2

Цикл 0			
k	Код імпл.	№ імпл.	Вид імпл.
0	0 0 0 0	0	V
1	0 0 0 1 0 0 1 0 1 0 0 0	1 2 8	V V V
2	0 0 1 1 0 1 0 1 1 0 1 0	3 5 10	V V V
3	0 1 1 1 1 0 1 1	7 11	V V

Цикл 1			
k	Код імпл.	№ імпл.	Вид імпл.
0	0 0 0 – 0 0 – 0 – 0 0 0	0,1 0,2 0,8	V V V
1	0 0 – 1 0 – 0 1 0 0 1 – – 0 1 0 1 0 – 0	1,3 1,5 2,3 2,10 8,10	V V V V V
2	0 – 1 1 – 0 1 1 0 1 – 1 1 0 1 –	3,7 3,11 5,7 10,11	V V V V

Цикл 2

k	Код імпл.	№ імпл.	Вид імпл.
0	0 0 – –	(0, 1), (2, 3)	*
	0 0 – –	(0, 2, (1, 3)	*
	– 0 – 0	(0, 2), (8, 10)	*
	– 0 – 0	(0, 8), (2, 10)	*
1	0 – – 1	(1, 3), (5, 7)	*
	0 – – 1	(1, 5), (3, 7)	*
	– 0 1 –	(2, 3), (10, 11)	*
	– 0 1 –	(2, 10), (3, 11)	*

Заповнення табл. 6.5.2 за циклами було зроблено таким чином. Спочатку заповнили три стовпчики нульового циклу : k (позначення індексу групи); двійковий код імпліканти; номер імпліканти в десятковій системі числення. В подальшому імпліканти ділимо на групи за значенням k. Потім здійснюємо порівняння коду першої імпліканти з групи $k = 0$ (0 0 0 0) з кодами інших імплікант з групи $k = 1$. Вони відрізняються тільки в одному розряді, тому робимо їх склеювання та одержуємо нові імпліканти з кодами 0 0 0 –, 0 0 – 0, – 0 0 0, які записуємо у стовпчик коду імпліканти циклу 1 в групу $k = 0$.

На наступному кроці заповнення табл. 6.5.2 попарно порівнюємо між собою коди групи 1 та 2 нульового циклу й отримаємо стовпчик коду імпліканти циклу 1, який належить групі 1. Аналогічно попарно порівнюючи між собою коди групи 2 та 3 нульового циклу, отримаємо стовпчик коду імпліканти циклу 1, який належить групі 2.

Аналогічно будуються стовпчики і рядки циклу 2. Для цього використовують коди циклу 1. Але при цьому порівнянні кодів необхідно дотримуватись такого правила: коди, що містять знак «–», можуть утворювати нові імпліканти, тільки якщо вони містять знаки «–» в одних і тих же рядках. Закінчивши порівняння кодів у циклах 0 і 1, виділяємо їх знаком «V», бо вони не є простими імплікантами. У стовпчику «код імплікант» циклу 2 є однакові коди імплікант, тому довільно викреслюємо один з однакових рядків, щоб кожна імпліканта зустрічалася тільки один раз. Порівнюючи коди имплікант у циклі 2, приходимо до висновку, що методом склеювання з них неможливо отримати нові імпліканти. Тому коди циклу 2 відповідають простим імплікантам і їх позначаємо зірочками.

Для знаходження тупикових ДНФ функції $f(x, y, z)$ будуємо імплікатну таблицю, табл. 6.5.3.

Таблиця 6.5.3

	0000	0001	0010	1000	0011	0101	1010	0111	1011
00– –	*	*	*		*				
– 0 – 0	*		*	*			*		
0 – – 1		*			*	*		*	
– 01 –			*		*		*		*

В рядках табл. 6.5.3 розташовані двійкові коди, що відповідають простим імплікантам, а в стовпчиках — коди, що відповідають конституентам одиниці. Якщо двійковий код рядку є частиною коду стовпчика (позиції із знаком «–» не порівнюються), то у відповідну клітинку таблиці заноситься зірочка. Із табл. 6.5.3 знаходимо МДНФ шляхом вибору із неї мінімальної кількості імплікант, які повністю перекривають усі двійкові коди конституент одиниці

$$f(x, y, z, t) = \bar{y} \cdot \bar{t} \vee \bar{x} \cdot t \vee \bar{y} \cdot z.$$

Таким чином, за допомогою методу Мак-Класкі отримали МДНФ функції $f(x, y, z, t)$, яка містить лише три елементарні кон'юнкції замість дев'яти конституент одиниці ДДНФ функції.

Знаходження МКНФ логічної функції за методом Мак-Класкі аналогічне знаходженню МДНФ.

6.6. Метод невизначених коефіцієнтів

Цей метод може бути застосований при мінімізації логічних функцій будь-якого числа аргументів. Для простоти викладення розглянемо його застосування для мінімізації функції, яка має три змінні.

Нехай задана логічна функція $f(x, y, z)$ у вигляді ДНФ таким чином

$$f(x,y,z) = K_1^1 x \vee K_1^0 \bar{x} \vee K_2^1 y \vee K_2^0 \bar{y} \vee K_3^1 z \vee K_3^0 \bar{z} \vee K_{12}^{11} x \cdot y \vee K_{12}^{10} x \cdot \bar{y} \vee$$
$$\vee K_{12}^{01} \bar{x} \cdot y \vee K_{12}^{00} \bar{x} \cdot \bar{y} \vee K_{13}^{11} x \cdot z \vee K_{13}^{10} x \cdot \bar{z} \vee K_{13}^{01} \bar{x} \cdot z \vee K_{13}^{00} \bar{x} \cdot \bar{z} \vee K_{23}^{11} y \cdot z \vee$$
$$\vee K_{23}^{10} y \cdot \bar{z} \vee K_{23}^{01} \bar{y} \cdot z \vee K_{23}^{00} \bar{y} \cdot \bar{z} \vee K_{123}^{111} x \cdot y \cdot z \vee K_{123}^{110} x \cdot y \cdot \bar{z} \vee K_{123}^{101} x \cdot \bar{y} \cdot z \vee$$
$$\vee K_{123}^{100} x \cdot \bar{y} \cdot \bar{z} \vee K_{123}^{011} \bar{x} \cdot y \cdot z \vee K_{123}^{010} \bar{x} \cdot y \cdot \bar{z} \vee K_{123}^{001} \bar{x} \cdot \bar{y} \cdot z \vee K_{123}^{000} \bar{x} \cdot \bar{y} \cdot \bar{z}.$$

В цьому виразі представлені всі можливі кон'юнктивні члени, які входять до ДНФ функції $f(x, y, z)$, коефіцієнти k з різними індексами є невизначеними і підбираються так, щоб була отримана МДНФ.

Якщо тепер задавати всі можливі набори значень змінних (x, y, z) і порівняти отримані після цього вирази зі значенням функції на вибраних наборах, то отримаємо 2^3 рівнянь для знаходження k:

$$f(0, 0, 0) = K_1^0 \vee K_2^0 \vee K_3^0 \vee K_{12}^{00} \vee K_{13}^{00} \vee K_{23}^{00} \vee K_{123}^{000};$$

$$f(0,0,1)=K_1^0\vee K_2^0\vee K_3^1\vee K_{12}^{00}\vee K_{13}^{01}\vee K_{23}^{01}\vee K_{123}^{001};$$
$$f(0,1,0)=K_1^0\vee K_2^1\vee K_3^0\vee K_{12}^{01}\vee K_{13}^{00}\vee K_{23}^{10}\vee K_{123}^{010};$$
$$f(0,1,1)=K_1^0\vee K_2^1\vee K_3^1\vee K_{12}^{01}\vee K_{13}^{01}\vee K_{23}^{11}\vee K_{123}^{011};$$
$$f(1,0,0)=K_1^1\vee K_2^0\vee K_3^0\vee K_{12}^{10}\vee K_{13}^{10}\vee K_{23}^{00}\vee K_{123}^{100};$$
$$f(1,0,1)=K_1^1\vee K_2^0\vee K_3^1\vee K_{12}^{10}\vee K_{13}^{11}\vee K_{23}^{01}\vee K_{123}^{101};$$
$$f(1,1,0)=K_1^1\vee K_2^1\vee K_3^0\vee K_{12}^{11}\vee K_{13}^{10}\vee K_{23}^{10}\vee K_{123}^{110};$$
$$f(1,1,1)=K_1^1\vee K_2^1\vee K_3^1\vee K_{12}^{11}\vee K_{13}^{11}\vee K_{23}^{11}\vee K_{123}^{111}.$$

Від задання деякої конкретної функції із розглянутих рівнянь залежить значення набору (x, y, z) розглядуваної функції. Якщо набір (x, y, z) такий, що функція на ньому дорівнює нулю, то в лівій частині відповідного рівняння буде стояти нуль.

Для розв'язання цього рівняння необхідно прирівняти нулю всі коефіцієнти k, які входять до правої частини розглядуваного рівняння. Розглянувши всі набори, на яких дана функція перетворюється в нуль, отримаємо всі нульові коефіцієнти K. У рівняннях, в яких зліва стоять одиниці, викреслим справа всі нульові коефіцієнти K. Із коефіцієнтів, що залишили, прирівнюємо одиниці коефіцієнт, який визначає кон'юнкцію найменшого можливого рангу, а решта коефіцієнтів у правій частині даного рівняння позначимо рівним нулю. Це можна зробити, оскільки диз'юнкція перетворюється в одиницю, якщо хоча б один елемент її дорівнював одиниці. Знайдені одиничні коефіцієнти K_i і визначають відповідну ДНФ функції.

Приклад 6.6.1. Задана логічна функція в ДДНФ

$$f(x,y,z)=x\cdot y\cdot z\vee x\cdot y\cdot\bar{z}\vee x\cdot\bar{y}\cdot z\vee x\cdot\bar{y}\cdot\bar{z}\vee\bar{x}\cdot\bar{y}\cdot\bar{z}.$$

Необхідно знайти її МДНФ методом невизначених коефіцієнтів.

Розв'язання. У відповідності із заданою логічною функцією, складемо систему рівнянь для знаходження коефіцієнтів k:

$$f(0,0,0)=1=K_1^0\vee K_2^0\vee K_3^0\vee K_{12}^{00}\vee K_{13}^{00}\vee K_{23}^{00}\vee K_{123}^{000};$$
$$f(0,0,1)=0=K_1^0\vee K_2^0\vee K_3^1\vee K_{12}^{00}\vee K_{13}^{01}\vee K_{23}^{01}\vee K_{123}^{001};$$
$$f(0,1,0)=0=K_1^0\vee K_2^1\vee K_3^0\vee K_{12}^{01}\vee K_{13}^{00}\vee K_{23}^{10}\vee K_{123}^{010};$$
$$f(0,1,1)=0=K_1^0\vee K_2^1\vee K_3^1\vee K_{12}^{01}\vee K_{13}^{01}\vee K_{23}^{11}\vee K_{123}^{011};$$
$$f(1,0,0)=1=K_1^1\vee K_2^0\vee K_3^0\vee K_{12}^{10}\vee K_{13}^{10}\vee K_{23}^{00}\vee K_{123}^{100};$$

$$f(1,0,1)=1=K_1^1\vee K_2^0\vee K_3^1\vee K\vee K_{12}^{10}\vee K_{13}^{11}\vee K_{23}^{01}\vee K_{123}^{101};$$
$$f(1,1,0)=1=K_1^1\vee K_2^1\vee K_3^0\vee K_{12}^{11}\vee K_{13}^{10}\vee K_{23}^{10}\vee K_{123}^{100};$$
$$f(1,1,1)=1=K_1^1\vee K_2^1\vee K_3^1\vee K_{12}^{11}\vee K_{13}^{11}\vee K_{23}^{11}\vee K_{123}^{111}.$$

Із другого, третього і четвертого рівнянь за властивістю диз'юнкції випливає, що $K_1^0=K_2^0=K_2^1=K_3^0=K_3^1=K_{12}^{00}=K_{12}^{01}=K_{13}^{00}=K_{13}^{01}=$ $=K_{23}^{01}=K_{23}^{10}=K_{23}^{11}=K_{123}^{001}=K_{123}^{010}=K_{123}^{011}=0.$

Після цього дана система рівнянь буде мати вигляд:

$$f(0,0,0)=1=K_{23}^{00}\vee K_{123}^{000};$$
$$f(1,0,0)=1=K_1^1\vee K_{12}^{10}\vee K_{13}^{10}\vee K_{23}^{00}\vee K_{133}^{100};$$
$$f(1,0,1)=1=K_1^1\vee K_{12}^{10}\vee K_{13}^{11}\vee K_{123}^{101};$$
$$f(1,1,0)=1=K_1^1\vee K_{12}^{11}\vee K_{13}^{10}\vee K_{123}^{110};$$
$$f(1,1,1)=1=K_1^1\vee K_{12}^{11}\vee K_{13}^{11}\vee K_{123}^{111}.$$

Прирівняємо нулю в кожному рівнянні всі відповідні коефіцієнти, які мають найбільше число змінних:

$$K_{12}^{11}=K_{12}^{10}=K_{13}^{11}=K_{13}^{10}=K_{123}^{111}=K_{123}^{110}=K_{123}^{101}=K_{123}^{100}=K_{123}^{000}=0.$$

В результаті цього отримаємо систему:

$$K_{23}^{00}=1$$
$$K_1^1\vee K_{23}^{00}=1$$
$$K_1^1=1$$
$$K_1^1=1$$
$$K_1^1=1$$

Із даної системи знаходимо МДНФ заданої логічної функції, яка дорівнює

$$f(x,y,z)=x\vee\overline{y}\cdot\overline{z}.$$

6.7. Метод Блейка–Порецького

Недоліком розглянутих методів є те, що для їх застосування перед мінімізацією необхідно представити логічну функцію у вигляді ДДНФ. Але процес такого представлення логічної функції з великим числом змінних є дуже громіздким. Тому вчені А. Блейк і П. С.

Порецький запропонували перехід від довільної ДНФ функції до скороченої ДНФ за допомогою операцій узагальненого склеювання і поглинання.

Операція узагальненого склеювання має вигляд

$$A \cdot x \vee B \cdot \bar{x} = A \cdot x \vee B \cdot \bar{x} \vee A \cdot B.$$

Доведення справедливості даної тотожності

$$A \cdot x \vee B \cdot \bar{x} = A \cdot x \cdot (1 \vee B) \vee B \cdot \bar{x} \cdot (1 \vee A) = A \cdot x \vee A \cdot B \cdot x \vee B \cdot \bar{x} \vee A \cdot B \cdot \bar{x} =$$
$$= A \cdot x \vee B \cdot \bar{x} \vee A \cdot B \cdot (x \vee \bar{x}) = A \cdot x \vee B \cdot \bar{x} \vee A \cdot B\,.$$

Метод Блейка–Порецького полягає у застосуванні різних операцій узагальненого склеювання до ДНФ функцій. Потім у отриманій формулі здійснюються всі можливі операції поглинання.

Приклад 6.7.1. Знайти скорочену ДНФ для функції $f(x, y, z) = \bar{x} \cdot y \cdot z \vee x \cdot y \vee x \cdot \bar{y}$ за методом Блейка–Порецького.

Розв'язання. Здійснимо операції узагальненого склеювання:

$$\bar{x} \cdot y \cdot z \vee x \cdot y = x \cdot y \vee \bar{x} \cdot y \cdot z \vee y \cdot z\,,\ \text{тут } A = y,\ B = y \cdot z\,;$$
$$\bar{x} \cdot y \cdot z \vee x \cdot \bar{y} = x \cdot \bar{y} \vee \bar{x} \cdot y \cdot z \vee 0\,,\ \text{тут } A = \bar{y},\ B = y \cdot x;$$
$$x \cdot y \vee x \cdot \bar{y} = x \cdot y \vee x \cdot \bar{y} \vee x\,,\ \text{тут } A = x,\ B = x.$$

Так як диз'юнкція лівих частин співпадає з функцією f, то до знову отриманих імплікант може бути застосована операція повного диз'юктивного склеювання

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

Виходячи із цього, вихідна формула прийме такий вигляд

$$f(x, y, z) = \bar{x} \cdot y \cdot z \vee y \cdot z \vee x \cdot \bar{y} \vee x = y \cdot z \cdot (\bar{x} \vee 1) \vee x \cdot (\bar{y} \vee 1) = x \vee y \cdot z.$$

Отже, $f(x, y, z) = x \vee y \cdot z$ є скорочена ДНФ.

? Контрольні запитання

1. В чому полягає мінімізація логічних функцій?
2. Дайте визначення імпліканти та імпліценти логічної функції.
3. Яку імпліканту та імпліценту називають простою?

4. Що таке скорочена ДНФ та КНФ?
5. Дайте визначення МДНФ та МКНФ?
6. Яку логічну функцію називають тупиковою?
7. Що розуміють під операцією повного і неповного склеювання логічної функції?
8. Що розуміють під операцією диз'юнктивного і кон'юнктивного поглинання?
9. Запишіть формули операцій диз'юнктивного склеювання і поглинання.
10. Запишіть формули операцій кон'юнктивного склеювання і поглинання.
11. Що таке імпікантна (імпліцентна) таблиця і для яких цілей вона використовується?
12. Сформулюйте правила склеювання клітинок і отримання МДНФ (МКНФ) за методом Вейча.
13. Сформулюйте правила склеювання клітинок і отримання МДНФ (МКНФ) за методом Карно.
14. У чому відмінність методів Вейча і Карно, а також їх таблиць для мінімізації?
15. Яким чином здійснюється мінімізація частково визначених функцій?
16. Назвіть основні кроки алгоритму мінімізації Квайна.
17. Які модифікації запропонував внести Мак-Класкі в метод Квайна?
18. У чому полягає суть методу мінімізації логічних функцій за невизначеними коефіцієнтами?
19. В чому суть методу мінімізації логічних функцій методом Блейка-Порецького?
20. Запишіть формулу узагальненого склеювання.
21. Дайте порівняльну характеристику методів мінімізації Вейча та Карно.
22. Дайте порівняльну характеристику методів мінімізації Квайна та Мак-Класкі.
23. Дайте порівняльну характеристику методів мінімізації за допомогою невизначених коефіцієнтів, Квайна, Мак-Класкі та Блейка-Порецького.

Задачі для самостійного розв'язування

1. Побудувати таблиці Вейча і Карно та мінімізувати за ними такі логічні функції:

а) $f(x, y, z) = x \cdot \bar{y} \cdot z \vee \bar{x} \cdot y \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z} \vee x \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot \bar{z}$;

б) $f(x, y, z) = x \cdot y \cdot z \vee \bar{x} \cdot y \cdot z \vee x \cdot \bar{y} \cdot z \vee x \cdot y \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z}$;

в) $f(x, y, z, t) = x \cdot y \cdot \bar{z} \cdot \bar{t} \vee x \cdot \bar{y} \cdot \bar{z} \cdot t \vee x \cdot \bar{y} \cdot z \cdot \bar{t} \vee x \cdot y \cdot \bar{z} \cdot t \vee x \cdot y \cdot z \cdot \bar{t}$;

г) $f(x, y, z, t) = \bar{x} \cdot \bar{y} \cdot z \cdot t \vee \bar{x} \cdot y \cdot \bar{z} \cdot t \vee x \cdot \bar{y} \cdot z \cdot \bar{t} \vee x \cdot y \cdot \bar{z} \cdot t \vee \bar{x} \cdot y \cdot z \cdot \bar{t}$.

2. Знайти мінімальні ДНФ і КНФ, що задані такими таблицями Вейча:

а)

	x		$\bar{x}$		
y	1	1	1	0	$\bar{t}$
	0	0	1	1	t
$\bar{y}$	0	0	1	1	
	1	1	1	0	$\bar{t}$
	$\bar{z}$		z		$\bar{z}$

б)

	x		$\bar{x}$		
	0	0	0	0	$\bar{t}$
y	1	1	0	1	
	1	1	0	1	t
$\bar{y}$	1	1	1	1	$\bar{t}$
	$\bar{z}$		z		$\bar{z}$

в)

	x		$\bar{x}$		
	1	0	0	1	$\bar{t}$
y	0	1	1	0	
	0	1	1	0	t
$\bar{y}$	1	0	0	1	$\bar{t}$
	$\bar{z}$		z		$\bar{z}$

г)

	x		$\bar{x}$		
	0	1	1	0	$\bar{t}$
y	1	1	0	1	
	0	0	1	1	
$\bar{y}$	1	1	0	1	$\bar{t}$
	$\bar{z}$		z		$\bar{z}$

3. Знайти мінімальні ДНФ і КНФ, що задані такими таблицями Карно:

а)

zt \ xy	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	0	0	1	1
10	0	0	0	0

б)

zt \ xy	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	1	1	0	1
10	1	1	1	1

в)

zt \ xy	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	1	0
10	1	0	1	0

г)

zt \ xy	00	01	11	10
00	1	0	1	0
	0	1	1	1
01	0	1	1	1
11	0	1	1	1
10	1	0	1	0

4. Знайти мінімальні ДНФ функцій методом Квайна, що задані номерами конституент одиниці таким чином:

а) $f(x, y, z, t) = 1, 2, 3, 5, 6, 7, 10, 12, 15$;

б) $f(x, y, z, t) = 0, 1, 5, 7, 9, 10, 13, 14, 15$;

в) $f(x, y, z, t) = 0, 2, 3, 4, 5, 6, 7, 10, 11$;

г) $f(x, y, z, t) = 1, 3, 4, 6, 8, 9, 10, 11, 14$.

5. Знайти ДНФ функцій методом Мак-Класкі, що задані номерами конституент одиниці, приведених у завданні 4.

6. Знайти мінімальні ДНФ функцій методом невизначених коефіцієнтів, що задані логічними фунціями:

а) $f(x, y, z,) = x \cdot \bar{y} \cdot z \vee x \cdot \bar{y} \cdot \bar{z} \vee \bar{x} \cdot y \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z}$;

б) $f(x, y, z) = \bar{x} \cdot y \cdot z \vee x \cdot \bar{y} \cdot x \vee x \cdot y \cdot \bar{z} \vee \bar{x} \cdot \bar{y} \cdot \bar{z}$;

в) $f(x, y, z) = x \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot z \vee \bar{x} \cdot y \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z}$;

г) $f(x, y, z) = \bar{x} \cdot \bar{y} \cdot \bar{z} \vee x \cdot \bar{y} \cdot \bar{z} \vee x \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{y} \cdot z$.

7. Знайти скорочені ДНФ методом Блейка–Порецького для таких функцій:

а) $f(x, y, z, t) = \bar{x} \cdot \bar{t} \vee \bar{x} \cdot z \cdot t \vee x \cdot \bar{y} \cdot z \vee x \cdot y \cdot z \cdot \bar{t}$;

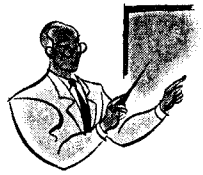
б) $f(x, y, z, t) = x \cdot y \cdot \bar{z} \vee \bar{x} \cdot z \cdot \bar{t} \vee \bar{x} \cdot \bar{y} \cdot \bar{z} \cdot \bar{t}$;

в) $f(x, y, z, t) = x \cdot \bar{y} \cdot z \vee \bar{x} \cdot \bar{z} \vee \bar{x} \cdot y \vee x \cdot \bar{y} \cdot \bar{z} \cdot \bar{t}$;

г) $f(x, y, z, t) = x \cdot \bar{y} \cdot \bar{t} \vee \bar{y} \cdot x \cdot t \vee \bar{x} \cdot t \vee x \cdot y \cdot z \cdot t$.

Коментарі

В даному розділі мінімізація логічних функцій методами Вейча і Карно викладена у відповідності з [5], а методи Квайна і Мак-Класкі взяті з [5, 21]. Метод невизначених коефіцієнтів випливає з [24], а метод Блейка-Порецького — з [8].



Розділ 7

ЛОГІКА ЧАСОВИХ І РЕКУРЕНТНИХ БУЛЕВИХ ФУНКЦІЙ

7.1. Логіка часових булевих функцій

Означення 7.1.1. **Часовою булевою функцією (ЧБФ)** називають функцію, яка дає однозначне відображення двійкових наборів $R = < x_1, x_2, ..., x_n, t >$ на множину $Y = \{0, 1\}$, де $x_1, x_2, ..., x_n$ — аргументи функції, які приймають значення 0 або 1; t — часова координата або просто час, який приймає цілочисленні значення, наприклад, від 0 до m.

Якщо ЧБФ $y = \varphi (x_1, x_2, ..., x_n, t)$ залежить від часу не суттєво, то вона перетворюється у звичайну функцію логіки Буля.

Теорема 7.1.1. Кількість різних ЧБФ функцій виду

$$y = \varphi (x_1, x_2, ..., x_n, t), \quad 0 \le t \le m - 1$$

дорівнює $2^{m \cdot 2^n}$.

Доведення. Так як кожна змінна набору $< x_1, x_2, ..., x_n >$ ЧБФ приймає значення $< 0, 1 >$, то число всіх можливих наборів буде дорівнювати $N = 2^n$. Але кожна часова логічна функція приймає також значення $< 0, 1 >$ і, у відповідності до n аргументів і часової координати, t дорівнюватиме $M = 2^{m \cdot N}$. Тоді загальна кількість різних ЧБФ буде дорівнювати $M = 2^{m \cdot 2^n}$, що і потрібно було довести.

Із теореми 7.1.1 випливає, що будь-яку ЧБФ можна повністю задати за допомогою таблиці істинності.

Приклад 7.1.1. Для ЧБФ заданої у вигляді

$$y = \varphi (x_1, x_2, t), \quad 0 \le t \le 2$$

знайти загальну її кількість і побудувати таблицю істинності для одної із них.

Розв'язання. Використовуючи теорему 7.1.1, знаходимо загальну кількість ЧБФ, яка дорівнює $2^{3 \cdot 2^2} = 2^{12} = 4096$. Таблиця істинності для однієї із них матиме вигляд, приведений у табл. 7.1.1.

Таблиця 7.1.1

x_1	x_2	t	$\varphi(x_1, x_2, t)$	x_1	x_2	t	$\varphi(x_1, x_2, t)$
0	0	0	0	1	0	1	1
0	1	0	1	1	1	1	1
1	0	0	0	0	0	2	0
1	1	0	0	0	1	2	0
0	0	1	0	1	0	2	1
0	1	1	1	1	1	2	0

Як випливає із прикладу 7.1.1, запис усіх ЧБФ за допомогою таблиці істинності є дуже громіздким і для практичних цілей не застосовується. Тому для ЧБФ, подібно як і для булевих функцій, необхідно ввести більш компактну форму запису. Для цього розглянемо будь-яку ЧБФ, яка має такий вигляд:

$$y = \varphi(x_1, x_2, ..., x_n, t), \ 0 \le t \le m - 1. \qquad (7.1.1)$$

Якщо тепер дати t будь-яке фіксоване значення ($t = k$, $0 \le k \le m - 1$), то ця функція прийме вигляд

$$y_k = \varphi_k(x_1, x_2, ..., x_n) \qquad (7.1.2)$$

Функція 7.1.2 є тепер звичайною функцією алгебри Буля і може вивчатися за допомогою методів, які викладені в розділі 2. Примушуючи t пробігати всю послідовність допускних значень, ми отримаємо послідовність функцій алгебри Буля.

$$\varphi_0, \varphi_1, ..., \varphi_{m-1}. \qquad (7.1.3)$$

Таким чином, будь-якій ЧБФ можна співпіставити послідовність функцій алгебри Буля.

Приклад 7.1.2. Записати ЧБФ прикладу 7.1.1 за допомогою послідовності функцій алгебри Буля.

Розв'язання. Використовуючи 7.1.2 та 7.1.3, а також дані табл. 7.1.1, ЧБФ буде мати такий вигляд:

$$\varphi_0 = \overline{x}_1 \cdot x_2; \quad \varphi_1 = x_1 \vee x_2; \quad \varphi_2 = x_1 \cdot \overline{x}_2. \tag{7.1.4}$$

Для більшої зручності запису ЧБФ введемо спеціальну функцію τ_α, яку визначають таким чином:

$$\tau_\alpha = \begin{cases} 1, & t = \alpha \\ 0, & t \neq \alpha \end{cases} \tag{7.1.5}$$

Тоді у нових позначеннях ЧБФ виду 7.1.1 можна записати у вигляді

$$\varphi = \varphi_0 \cdot \tau_0 \vee \varphi_1 \cdot \tau_1 \vee \ldots \vee \varphi_{m-1} \cdot \tau_{m-1} \tag{7.1.6}$$

Приклад 7.1.3. Записати ЧБФ прикладу 7.1.1 у позначеннях виразу 7.1.6.

Розв'язання. Використовуючи умову прикладу 7.1.1, а також табл. 7.1.1, вирази 7.1.4, 7.1.5 і 7.1.6, отримаємо

$$\varphi = \overline{x}_1 \cdot x_2 \cdot \tau_0 \vee (x_1 \vee x_2) \cdot \tau_1 \vee x_1 \cdot \overline{x}_2 \cdot \tau_2$$

Означення 7.1.2. Якщо в ЧБФ $\varphi = \varphi_0 \cdot \tau_0 \vee \varphi_1 \cdot \tau_1 \vee \ldots \varphi_{m-1} \cdot \tau_{m-1}$ всі функції φ_i, $i = 0, 1, 2, \ldots, m-1$, представлені в ДДНФ, то відповідний вираз для φ називають **досконалою диз'юнктивною нормальною формою часової булевої функції φ.**

Означення 7.1.3. Якщо в ЧБФ $\varphi = \varphi_0 \cdot \tau_0 \vee \varphi_1 \cdot \tau_1 \vee \ldots \vee \varphi_{m-1} \cdot \tau_{m-1}$ всі функції φ_i, $i = 0, 1, 2, \ldots, m-1$, представлені в ДКНФ, то відповідний вираз для φ називають **досконалою кон'юнктивною нормальною формою часової булевої функції φ.**

Приклад 7.1.4. Записати в ДДНФ і ДКНФ часову булеву функцію, яка задана табл. 7.1.2.

Розв'язання. Так як $0 \le t \le 1$, то, використовуючи табл. 7.1.2, знаходимо ДДНФ і ДКНФ для функцій φ_0 і φ_1, які матимуть такий вигляд:

$$\varphi_0 = \overline{x}_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee x_1 \cdot \overline{x}_2 \cdot x_3;$$

$$\varphi_0 = (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee x_2 \vee \overline{x}_3) \cdot (x_1 \vee \overline{x}_2 \vee x_3) \cdot (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \cdot (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3);$$

$$\varphi_1 = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot x_2 \cdot \overline{x_3} \vee \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3;$$

$$\varphi_1 = (\overline{x_1} \vee x_2 \vee x_3) \cdot (\overline{x_1} \vee x_2 \vee \overline{x_3}) \cdot (\overline{x_1} \vee \overline{x_2} \vee x_3).$$

Таблиця 7.1.2

x_1	x_2	x_3	t	$\varphi(x_1, x_2, x_3, t)$	x_1	x_2	x_3	t	$\varphi(x_1, x_2, x_3, t)$
0	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	1	1	1
0	1	0	0	0	0	1	0	1	1
0	1	1	0	1	0	1	1	1	1
1	0	0	0	1	1	0	0	1	0
1	0	1	0	1	1	0	1	1	0
1	1	0	0	0	1	1	0	1	0
1	1	1	0	0	1	1	1	1	1

Часові булеві функції, записані в ДДНФ $(\varphi_д)$ і ДКНФ $(\varphi_к)$, з урахуванням формули 7.1.6 і наведених вище їх складових φ_0 і φ_1, матимуть такий вигляд:

$$\varphi_д = (\overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \vee x_1 \cdot \overline{x_2} \cdot x_3) \cdot \tau_0 \vee (\overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee$$
$$\vee \overline{x_1} \cdot x_2 \overline{x_3} \vee \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3) \cdot \tau_1.$$

$$\varphi_к = [(x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee x_2 \vee \overline{x_3}) \cdot (x_1 \vee \overline{x_2} \vee x_3) \cdot (\overline{x_1} \vee \overline{x_2} \vee x_3) \cdot$$
$$\cdot (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})] \cdot \tau_0 \vee [(\overline{x_1} \vee x_2 \vee x_3) \cdot (\overline{x_1} \vee x_2 \vee \overline{x_3}) \cdot (\overline{x_1} \vee \overline{x_2} \vee x_3)] \cdot \tau_1.$$

Мінімізацію ЧБФ виконують аналогічно мінімізації функцій алгебри логіки, яка була розглянута в розділі 6. Але мінімізація у випадку ЧБФ функцій при кількості змінних x більше трьох або при великій кількості значень t є великою трудністю. Тому для практичних потреб при мінімізації ЧБФ застосовують не повну (часткову) мінімізацію. Вона проводиться наступним чином. Нехай ДДНФ часової булевої функції φ має такий вигляд:

$$\varphi = \varphi_0 \cdot \tau_0 \vee \varphi_1 \cdot \tau_1 \vee \ldots \vee \varphi_{m-1} \cdot \tau_{m-1}.$$

Тоді визначаємо за звичайними правилами МДНФ для φ_0, $\varphi_1, \ldots, \varphi_{m-1}$ і за наближений мінімальний вираз для φ приймаємо

$$\varphi = \varphi^*_0 \cdot \tau_0 \vee \varphi^*_1 \cdot \tau_1 \vee \ldots \vee \varphi^*_{m-1} \cdot \tau_{m-1} \qquad (7.1.7)$$

де φ^*_i — МДНФ функції φ_i, $i = 0, 1, 2, \ldots, m-1$. При великому значенні $m-1$ і малої кількості аргументів x у функції φ_i такий метод є досить ефективним.

Приклад 7.1.5. Для ЧБФ заданої у вигляді ДДНФ,

$$\varphi = (\overline{x_1} \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \vee x_1 \cdot x_2) \cdot \tau_0 \vee (\overline{x_1} \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2) \cdot \tau_1$$

виконати її часткову мінімізацію.

Розв'язання. Застосовуючи метод неповної мінімізації, отримаємо

$$\varphi_0 = \overline{x_1} \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \vee x_1 \cdot x_2 =$$
$$= \overline{x_1} \cdot (\overline{x_2} \vee x_2) \vee x_1 \cdot (\overline{x_2} \vee x_2) = \overline{x_1} \vee x_1 = 1;$$
$$\varphi_1 = \overline{x_1} \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2 = \overline{x_1} \cdot (\overline{x_2} \vee x_2) = \overline{x_1}$$

Тоді за часткову мінімальну форму приймаємо вираз

$$\varphi = \tau_0 \vee \overline{x_1} \cdot \tau_1.$$

Приклад 7.1.6 Для ЧБФ, заданої у вигляді ДДНФ,

$$\varphi_0 = \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \vee x_1 \cdot \overline{x_2} \cdot x_3;$$
$$\varphi_1 = x_1 \cdot \overline{x_2} \cdot \overline{x_3} \vee x_1 \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot x_2 \cdot \overline{x_3} \vee \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot x_2 \cdot x_3;$$
$$\varphi_2 = \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3};$$
$$\varphi_3 = \overline{x_1} \cdot x_2 \cdot \overline{x_3} \vee \overline{x_1} \cdot x_2,$$

виконати її часткову мінімізацію.

Розв'язання. Застосовуючи метод неповної мінімізації, до кожної із наведених функцій, отримаємо

$$\varphi_0 = \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot (\overline{x_3} \vee x_3) = \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2};$$
$$\varphi_1 = x_1 \cdot \overline{x_2} \cdot (\overline{x_3} \vee x_3) \vee \overline{x_1} \cdot x_2 \cdot (\overline{x_3} \vee x_3) \vee x_1 \cdot x_2 \cdot x_3 =$$
$$= x_1 \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot x_2 \cdot x_3;$$
$$\varphi_2 = \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3};$$
$$\varphi_3 = \overline{x_1} \cdot x_2 \cdot (\overline{x_3} \vee 1) = \overline{x_1} \cdot x_2.$$

Застосовуючи формулу 7.1.7, отримаємо мінімальну формулу для ЧБФ, яка матиме такий вигляд:

$$\varphi = (\overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2}) \cdot \tau_0 \vee (x_1 \cdot \overline{x_2} \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot x_2 \cdot x_3) \cdot \tau_1 \vee$$
$$(\overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3}) \cdot \tau_2 \vee \overline{x_1} \cdot x_2 \cdot \tau_3 .$$

Означення 7.1.4. Часову булеву функцію називають **періодичною, з періодом *q***, якщо для будь-якого t має місце співвідношення $\varphi_{t+q} = \varphi_t$.

При заданні ЧБФ таблицею істинності достатньо мати в ній тільки q стрічок. Тобто можна сказати, що періодична ЧБФ не має границі зверху. Це достатньо важливо для аналізу і синтезу схем, робота яких описується за допомогою ЧБФ.

Приклад 7.1.7. Для ЧБФ заданої таблицею істинності, табл. 7.1.3.

Таблиця 7.1.3

x_1	x_2	t	$\varphi(x_1, x_2, t)$	x_1	x_2	t	$\varphi(x_1, x_2, t)$
0	0	0	0	0	0	2	0
0	1	0	0	0	1	2	0
1	0	0	0	1	0	2	0
1	1	0	1	1	1	2	1
0	0	1	0	0	0	3	0
0	1	1	1	0	1	3	1
1	0	1	1	1	0	3	1
1	1	1	1	1	1	3	1

знайти період функції.

Розв'язання. Із розгляду табл. 7.1.3 випливає, що

$$\varphi_{2k}(x_1, x_2) = x_1 \cdot x_2, \qquad k = 0, 1, ...;$$
$$\varphi_{2k+2}(x_1, x_2) = x_1 \vee x_2, \quad k = 0, 1, ...;$$
$$\varphi_{2k+3}(x_1, x_2) = x_1 \vee x_2, \quad k = 0, 1, ...$$

Тобто, період заданої таблицею істинності ЧБФ дорівнює двом.

Приклад 7.1.8. Для ЧБФ, заданої формулою

$$\varphi = x_1 \cdot \tau_0 \vee x_2 \cdot \tau_1 \vee x_3 \cdot \tau_2 \vee x_1 \cdot x_2 \cdot x_3 \cdot \tau_3 \vee x_1 \cdot \tau_4 \vee x_2 \cdot \tau_5,$$

знайти її період.

Розв'язання. Задана за умовою ЧБФ не є періодичною, так як вона визначена лише для значень $t \le 5$. Але її можна задовільним чином довизначити для значень $t > 5$. Якщо воно буде зроблено належним чином, то можна отримати періодичну ЧБФ. У нашому випадку найбільш природно отримати періодичну ЧБФ із періодом, який дорівнює чотирьом

$$\varphi_{4k}(x_1, x_2, x_3) = x_1, \qquad k = 0, 1, \ldots;$$
$$\varphi_{4k+1}(x_1, x_2, x_3) = x_2, \qquad k = 0, 1, \ldots;$$
$$\varphi_{4k+2}(x_1, x_2, x_3) = x_3, \qquad k = 0, 1, \ldots;$$
$$\varphi_{4k+3}(x_1, x_2, x_3) = x_1 \cdot x_2 \cdot x_3, \quad k = 0, 1, \ldots.$$

Приклад 7.1.9. Для ЧБФ, заданої формулою

$$\varphi = x_1 \cdot x_2 \cdot \tau_0 \vee \tau_1 \vee x_2 \cdot \tau_3 \vee (x_1 \vee x_2) \cdot \tau_4,$$

знайти її період.

Розв'язання. Ця ЧБФ може бути довизначена до періодичної з періодом, який дорівнює п'яти

$$\varphi_{5k}(x_1, x_2) = x_1 \cdot x_2, \qquad k = 0, 1, \ldots;$$
$$\varphi_{5k+1}(x_1, x_2) = 1, \qquad k = 0, 1, \ldots;$$
$$\varphi_{5k+2}(x_1, x_2) = 0, \qquad k = 0, 1, \ldots;$$
$$\varphi_{5k+3}(x_1, x_2) = x_2, \qquad k = 0, 1, \ldots;$$
$$\varphi_{5k+4}(x_1, x_2) = x_1 \vee x_2, \quad k = 0, 1, \ldots.$$

Із прикладу 7.1.9 випливає, що будь-яка ЧБФ $\varphi(x_1, x_2, \ldots, x_n, t)$, $0 \le t \le m-1$, може бути **довизначена до періодичної ЧБФ із періодом, більшим за $m-1$.**

7.2. Логіка рекурентних булевих функцій

Нехай задана певна функція φ, яка залежить від часу. Позначимо через y_t значення цієї функції в момент часу t, яка може приймати два значення 0 або 1. Розглянемо множину двійкових наборів

$$Q = \{< x_{1t}, x_{2t}, ..., x_{nt}, y_{t-1}, ..., y_{t-k} >\},$$

координати яких приймають значення 0 або 1. Набором Q^* будемо називати фіксований двійковий набір із двійкових наборів множини Q, де $Q^* = < x^*_{1t}, x^*_{2t}, ..., x^*_{nt}, y^*_{t-1}, ..., y^*_{t-k} >$. Виконаємо однозначне відображення двійкового набору Q на множину $Y = \{0, 1\}$.

Означення 7.2.1. **Рекурентною булевою функцією першого виду (РБФ–1)** називають функцію, яка дає однозначне відображення множини Q на множину Y.

Її записують у найбільш загальному вигляді таким чином:

$$y_t = \varphi(x_{1t}, x_{2t}, ..., x_{nt}, y_{t-1}, ..., y_{t-k}), \quad 1 \le k \le t.$$

Якщо $k = t$, то в число аргументів φ входить значення y_0, яке збігається зі значенням y_t на початку процесу при $t = 0$. В подальшому завжди будемо враховувати, що початкове значення є заданим. Тому РБФ – 1 з урахуванням початкової умови можна записати так:

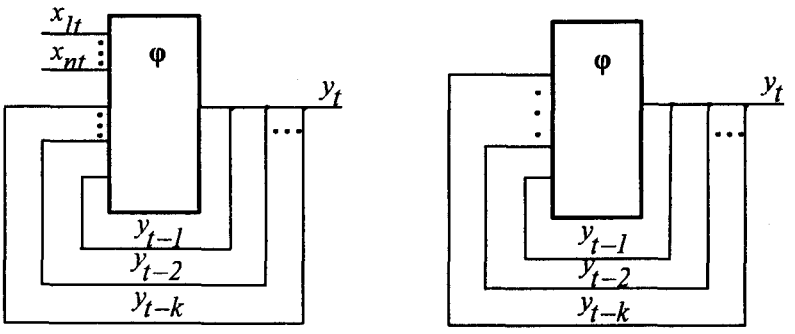
$$\left.\begin{aligned} y_0 &= a \\ y_t &= \varphi(x_{1t} + x_{2t} + \ldots + x_{nt}, y_{t-1}, \ldots, y_{t-k}), \end{aligned}\right\}$$

де $a = \{0, 1\}$.

При знаходженні значень y_t, для яких $t < k$, серед аргументів φ будемо мати значення функції, які необхідно знаходити і у від'ємні моменти часу. Тому для всіх випадків необхідно враховувати, що для будь-якого $t < 0$ маємо рівність $y_t = y_0$. Звідси випливає, що кінцеве значення РБФ–1 має такий вигляд:

$$\left.\begin{aligned} y_t &= a, t \le 0; \\ y_t &= \varphi(x_{1t}, x_{2t}, ..., x_{nt}, y_{t-1}, y_{t-k}), t > 0 \end{aligned}\right\}, \qquad (7.2.1)$$

Фізично цю РБФ–1 реалізують у вигляді схем з n входами і k лініями зворотного зв'язку, рис. 7.2.1а.



а) *б*)

Рис. 7.2.1

Означення 7.2.2. **Виродженою рекурентною булевою функцією першого виду (ВРБФ–1)** називають РБФ–1, яка не залежить від аргументів x_{it}, де $i = 0, 1, 2, \ldots, n$.

ВРБФ–1 задають таким співвідношенням:

$$\left.\begin{aligned} &y_t = a,\ t \le 0; \\ &y = \varphi(y_{t-1}, y_{t-2}, \ldots, y_{t-k}),\ t > 0 \end{aligned}\right\}$$

ВРБФ–1 фізично представлено на рис. 7.2.1б.

Приклад 7.2.1. За заданим співвідношенням $y = 0$, $y_t = y_{t-1} \vee 1$ необхідно знайти ВРБФ–1.

Розв'язання. Виходячи із означення 7.2.2, ВРБФ–1 буде визначена співвідношенням $y = 0$, $y_t = 1$ $(t = 1, 2, \ldots)$.

Приклад 7.2.2. За заданим співвідношенням

$$y_1 = x_{11} \vee y_0 = x_{11} \vee 0 = x_{11};$$

$$y_2 = x_{12} \vee y_1 = x_{11} \vee x_{12};$$

$$y_3 = x_{13} \vee y_2 = x_{11} \vee x_{12} \vee x_{13}$$

знайти РБФ–1.

Розв'язання. Використовуючи означення 7.2.1 і задані за умовою співвідношення, РБФ–1 для будь-якого t буде мати вигляд

$$y_t = \bigvee_{m=1}^{t} x_{1m}.$$

Розглянемо тепер набори такого виду:

$$U = \langle x^*_{1t}, x^*_{2t}, \dots, x^*_{nt}, x^*_{1(t-1)}, x^*_{2(t-2)}, \dots,$$
$$x^*_{n(t-1)}, \dots, x^*_{1(t-r)}, x^*_{2(t-r)}, \dots, x^*_{n(t-r)} \rangle.$$

У цих наборах $x_{j(t-i)}$ відповідає значенню j-го вхідного аргументу в момент часу $t-i$. При цьому $x_{j(t-i)}$ дорівнює або *1*, або *0*.

Означення 7.2.3. **Рекурентною булевою функцією другого виду (РБФ–2)** називають функцію, яка дає однозначне відображення множини U на множину $Y = \{0, 1\}$.

Приклад 7.2.3. Для заданої РБФ–2 $y_t = x_{1t} \vee x_{2t} \cdot x_{2(t-1)}$ при $\{x_{1t}\} = 0, 1, 0, 1, 0, 1 \dots$; і $\{x_{2t}\} = 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, \dots$ знайти y_0, y_1, y_2, y_3, y_4 і т. д.

Розв'язання. Використовуючи означення 7.2.3, формули 7.2.1 і 7.2.2 отримаємо

$$y_0 = 0 \vee 0 \cdot 0 = 0;$$
$$y_1 = 1 \vee 0 \cdot 0 = 1;$$
$$y_2 = 0 \vee 0 \cdot 0 = 0;$$
$$y_3 = 1 \vee 1 \cdot 0 = 1;$$
$$y_4 = 0 \vee 1 \cdot 1 = 1;$$

і т. д.

Приклад 7.2.4. Для заданої системи рекурентних булевих функцій

$$y_{1t} = x_{1t} \vee y_{2(t-1)};$$
$$y_{2t} = \bar{x}_{1t} \cdot y_{2(t-1)} \vee y_{1(t-2)};$$
$$y_{3t} = y_{1(t-1)} \cdot \bar{y}_{3(t-1)} \vee y_{2(t-1)}.$$

знайти декілька перших значень функцій при: $y_{10} = 0$; $y_{20} = 0$; $y_{30} = 1$; $\{x_{1t}\} = 0, 1, 0, 1, 0 \dots$.

Розв'язання. Знаходимо значення функцій y_{1t}, y_{2t}, *і* y_{3t} для $t = 1$

$$y_{11} = x_{11} \vee y_{20} = 0;$$

$$y_{21} = \overline{x_{11}} \cdot y_{20} \vee y_{1(-1)} = 0;$$

$$y_{31} = y_{10} \cdot \overline{y_{30}} \vee y_{20} = 0;$$

для $t = 2$

$$y_{12} = x_{12} \vee y_{21} = 1;$$

$$y_{22} = \overline{x_{12}} \cdot y_{21} \vee y_{10} = 0;$$

$$y_{32} = y_{11} \cdot \overline{y_{31}} \vee y_{21} = 0;$$

для $t = 3$

$$y_{13} = x_{13} \vee y_{22} = 0;$$

$$y_{23} = \overline{x_{13}} \cdot y_{22} \vee y_{11} = 0;$$

$$y_{33} = y_{12} \cdot \overline{y_{32}} \vee y_{22} = 1.$$

і т. д.

Контрольні запитання

1. Що називають часовою булевою функцією?
2. Чим відрізняється часова булева функція від звичайної булевої функції?
3. Яку функцію називають ДДНФ часової булевої функції?
4. Яку функцію називають ДКНФ часової булевої функції?
5. Чи можна мінімізувати часові булеві функції?
6. Яку мінімізацію доцільно застосовувати для часових булевих функцій?
7. Яку часову булеву функцію називають періодичною?
8. Що називають рекурентною булевою функцією?
9. Яку рекурентну булеву функцію називають функцією першого виду?
10. Що таке вироджена рекурентна булева функція першого виду?
11. Яку рекурентну булеву функцію називають функцією другого виду?
12. Чим відрізняються між собою рекурентні булеві функції першого і другого видів?

Задачі для самостійного розв'язування

1. Для заданої часової булевої функції $y = \varphi(x_1, x_2, t)$, $0 \le t \le 3$ знайти загальну її кількість і побудувати таблицю істинності для одної із них.

2. Записати в ДДНФ і ДКНФ часову булеву функцію, яка наведена в таблиці

x_1	x_2	x_3	t	$\varphi(x_1, x_2, x_3, t)$	x_1	x_2	x_3	t	$\varphi(x_1, x_2, x_3, t)$
0	0	0	0	1	0	0	0	1	0
0	0	1	0	1	0	0	1	1	0
0	1	0	0	0	0	1	0	1	1
0	1	1	0	0	0	1	1	1	1
1	0	0	0	0	1	0	0	1	1
1	0	1	0	0	1	0	1	1	1
1	1	0	0	1	1	1	0	1	0
1	1	1	0	0	1	1	1	1	1

3. Для часової булевої функції, заданої у вигляді ДДНФ

$$\varphi = (x_1 \cdot \overline{x_2} \vee x_1 \cdot x_2 \vee \overline{x_1} \cdot \overline{x_2}) \cdot \tau_0 \vee (\overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_2} \vee \overline{x_1} \cdot \overline{x_2}) \cdot \tau_1,$$

виконати її часткову мінімізацію.

4. Для часової булевої функції, заданої у вигляді ДДНФ

$$\varphi_0 = x_1 \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3};$$

$$\varphi_1 = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \vee \overline{x_1} \cdot x_2 \cdot \overline{x_3} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3;$$

$$\varphi_2 = x_1 \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot x_3,$$

виконати її часткову мінімізацію.

5. Для часової булевої функції, заданої у вигляді ДКНФ

$$\varphi_0 = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \cdot (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \cdot (\overline{x_1} \vee \overline{x_2} \vee x_3);$$

$$\varphi_1 = (x_1 \vee x_2 \vee \overline{x_3}) \cdot (x_1 \vee \overline{x_2} \vee x_3);$$

$$\varphi_2 = (\overline{x_1} \vee \overline{x_2} \vee x_3) \cdot (x_1 \vee \overline{x_2} \vee x_3) \cdot (\overline{x_1} \vee x_2 \vee \overline{x_3}) \cdot (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}),$$

виконати її часткову мінімізацію.

6. Для часової булевої функції $\varphi = x_1 \cdot \tau_0 \vee x_1 \cdot x_2 \cdot \tau_1 \vee x_1 \cdot x_2 \cdot x_3 \cdot \tau_2 \vee \vee x_1 \cdot \tau_4$ знайти її період.

7. Для часової булевої функції $\varphi = \tau_0 \vee x_1 \cdot x_2 \cdot \tau_1 \vee x_1 \cdot \tau_2 \vee \vee (x_1 \vee x_2)\,\tau_3$ знайти її період.

8. За заданим співвідношенням $y = 0;\ y_t = 1\ (t = 1, 2, \ldots)$ знайти вироджену рекурентну булеву функцію.

9. Для заданої рекурентної булевої функції другого виду

$$y_t = x_{0t} \vee x_{1t} \cdot x_{1(t-1)} \vee x_{2t} \cdot x_{2(t-1)},$$

при $\{x_{0t}\} = 0, 1, 0, 1, 0, 1 \ldots$, $\{x_{1t}\} = 1, 0, 1, 0, 1, 0 \ldots$ і $\{x_{2t}\} = 0, 0, 1, 1, 0, 0, 1, 1 \ldots$ знайти $y_0, y_1, y_2, y_3, y_4, y_5$ і т. д.

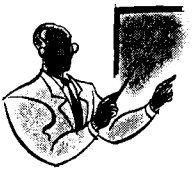
10. Для заданої системи рекурентних булевих функцій

$$y_{1t} = x_{1t} \vee x_{2t} \vee y_{2(t-1)};$$
$$y_{2t} = x_{1t} \cdot y_{2(t-1)} \vee y_{1(t-2)};$$
$$y_{3t} = \overline{x}_{2t} \cdot y_{1(t-1)} \cdot \overline{y}_{2(t-1)} \vee y_{3(t-1)}.$$

знайти декілька перших значень функцій при $y_{10} = 0$; $y_{20} = 1$; $y_{30} = 0$; $\{x_{1t}\} = 0, 1, 0, 1, 0 \ldots$, $\{x_{2t}\} = 1, 0, 1, 0, 1, 0, \ldots$.

Коментарі

Основні відомості, які викладені у цьому розділі за часовими булевими функціями, взяті із [17], а рекурентні булеві функції випливають із [24].



Розділ 8

ЛОГІКА ЦИФРОВИХ АВТОМАТІВ

8.1. Основні визначення

Теорія автоматів має справу з математичними моделями, які використовують у будь-якій галузі дослідження, наприклад, для *моделювання* багатьох процесів, алгоритмів, розробки компонентів комп'ютерних систем і таке інше.

Означення 8.1.1. **Автоматом** називають певну систему, яка має n–входів і m–виходів і яка робить відповідне перетворення множини вхідних сигналів $A = \{a_1, a_2, \ldots, a_n\}$, в множину вихідних сигналів $B = \{в_1, в_2, \ldots, в_m\}$, рис. 8.1.1.

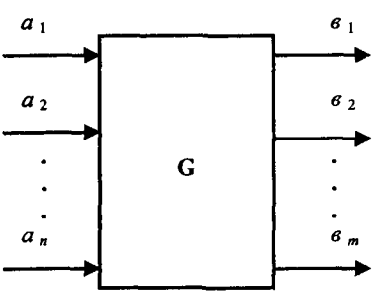


Рис. 8.1.1

Якщо характер перетворень усередині автомата в дискретні моменти часу $t = 0, 1, 2, \ldots$ залежить тільки від комбінації сигналів, які поступають на його входи, то такі автомати називають **комбінаційними** (тривіальними). У таких автоматах вихідні сигнали є функцією вхідних сигналів, тобто $B_i = G(a_i)$.

Якщо характер перетворень усередині автомата в дискретні моменти часу $t = 0, 1, 2, \ldots$ залежить не тільки від вхідних сигналів,

але і від сигналів, які надійшли до автомата раніше, то такий автомат називають **автоматом з пам'яттю** (в подальшому просто автомат), рис. 8.1.2.

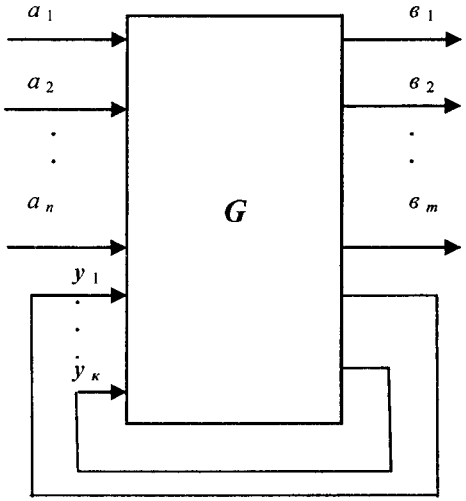


Рис. 8.1.2

В таких автоматах вихідні сигнали залежать не тільки від множини вхідних сигналів $A = \{a_1, a_2, \ldots, a_n\}$, але і від множини внутрішніх станів автомата $Q = \{q_0, q_1, q_2, \ldots, q_s\}$. У них вихідні сигнали і внутрішні стани є функціями станів автомата і множини вхідних сигналів.

Відображення, яке індуктионується таким автоматом, незаперечно визначається заданням у ньому двох функцій, які називають:

1. функцією переходів $\delta(q(t), a(t))$, яка визначає внутрішні стани автомата згідно із значенням вхідних сигналів $a(t)$ і попереднього стану автомата $q(t-1)$;

2. функцією виходів $\lambda(q(t), a(t))$, яка визначає вихідні сигнали $Z(t) = \lambda(q(t), a(t))$ залежно від внутрішнього стану автомата $q(t)$ і вхідних сигналів $a(t)$.

Отже, такий автомат можна задати шестикомпонентним вектором $W = \{A, B, Q, \delta, \lambda, q_0\}$, у якого:

1. $A = \{a_1, a_2, \ldots, a_n\}$ — множина вхідних сигналів;
2. $B = \{в_1, в_2, \ldots, в_m\}$ — множина вихідних сигналів;

3. $Q = \{q_0, q_1, q_2, \ldots, q_s\}$ — множина внутрішніх станів;
4. δ — функція переходів;
5. λ — функція виходів;
6. $q_0 \in Q$ — початковий стан.

Якщо кількість вхідних, вихідних і внутрішніх станів (у подальшому станів) автомата є кінечна величина, то такі автомати називають **скінченими.** В даному розділі розглядаються тільки скінченні автомати.

Означення 8.1.2. **Абстрактним автоматом** називають математичну модель дискретного пристрою задану шестикомпонентним вектором $W = \{A, B, Q, \delta, \lambda, q_0\}$, яка абстрагується від його змісту в частині визначення природи внутрішньої структури, вхідних і вихідних сигналів.

Більшість задач, з якими має справу теорія автоматів, можна поділити на дві категорії: задачі, які пов'язані з аналізом, що складається в прогнозуванні поведінки досліджуваної дискретної системи, і задачі синтезу дискретної системи, яка базується на побудові системи згідно з результатами її аналізу.

Означення 8.1.3. **Структурним автоматом** називають математичну модель дискретного пристрою, задану шестикомпонентним вектором $W = \{A, B, Q, \delta, \lambda, q_0\}$, в якої визначені природа і структура побудови внутрішніх станів, вхідних і вихідних сигналів.

Автомати за своєю структурою можуть бути повністю або частково визначені.

Означення 8.1.4. **Повністю визначеним автоматом** називають автомат, у якого ділянка визначення переходів $\delta(q(t), a(t))$ і виходів $\lambda(q(t), a(t))$ збігаються з множиною $A \times B$ усіх пар виду (a_i, b_j), де $i \in n$, $j \in m$.

Означення 8.1.5. **Частково визначеним автоматом** називають автомат, у якого ділянки визначення функції переходів $\delta(q(t), a(t))$ і виходів $\lambda(q(t), a(t))$ визначені не для всіх пар $(a_i, b_j) \in A \times B$, де $i \in n$, а $j \in m$.

В природі розрізняють синхронні і асинхронні автомати.

Означення 8.1.6. **Синхронним автоматом** називають автомат, робота якого, тобто виконання функції переходів $\delta(q(t), a(t))$ і

виходів $\lambda(q(t), a(t))$, відбувається через рівні інтервали часу $t = 0, 1, 2, \ldots$.

Означення 8.1.7. **Асинхронним автоматом** називають автомат, виконання функції переходів $\delta(q(t), a(t))$ і виходів $\lambda(q(t), a(t))$ у якого відбувається тільки при зміні вектора вхідного сигналу $a(t)$.

8.2. Автомати Мілі, Мура, *С*-автомати

З практичної точки зору найбільше розповсюдження (застосування) мають автомати Мілі і Мура, які одержали назву на честь їх перших дослідників, американських учених G.H. Mealy і E.F. Moore.

Означення 8.2.1. **Автоматом Мілі** називають математичну модель дискретного пристрою задану шестикомпонентним вектором $W = \{A, B, Q, \delta, \lambda, q_0\}$, у якої закон функціонування описують слідуючою системою рівнянь

$$\left.\begin{aligned} q(t+1) &= \delta(q(t),\ a(t)); \\ в(t) &= \lambda(q(t),\ a(t)), \end{aligned}\right\} \qquad (8.2.1)$$

де $\delta\ (q\ (t), a\ (t))$ — функція переходів;

$\lambda\ (q\ (t), a\ (t))$ — функція виходів;

$q(t\ +\ 1)$ — стан системи, в який вона переходить із стану $q\ (t)$ під дією вхідних сигналів $a(t)$;

$в(t)$ — вихідний сигнал, який система видає при переході її із стану $q(t)$ в стан $q(t\ +1)$ під дією вхідних сигналів $a\ (t)$.

Означення 8.2.2. **Автоматом Мура** називають математичну модель дискретного пристрою, задану шестикомпонентним вектором $W = \{A, B, Q, \delta, \lambda, q_0\}$, у якої закон функціонування описують такою системою рівнянь

$$\left.\begin{aligned} q(t+1)\ &= \delta(q(t),\ a(t)); \\ Z(t)\ &= \lambda(q(t)), \end{aligned}\right\} \qquad (8.2.2)$$

де $Z(t)$ — вихідний сигнал, який система видає при знаходженні її в стані $q(t)$.

Означення 8.2.3. ***С*-автоматом** називають математичну модель дискретного пристрою, задану восьмикомпонентним вектором

$W = \{A, B, Z, Q, \lambda_1, \lambda_2, \delta, q_0\}$, закон функціонування якої описують такою системою рівнянь

$$\left.\begin{aligned} q(t+1) &= \delta(q(t), a(t)); \\ в(t) &= \lambda_1(q(t), a(t)); \\ Z(t) &= \lambda_2(q(t)) \end{aligned}\right\}, \qquad (8.2.3)$$

де $B = \{в_1, в_2, \ldots, в_m\}$ — множина вихідних сигналів типу 1;

$Z = \{z_1, z_2, \ldots, z_p\}$ — множина вихідних сигналів типу 2;

$\lambda_1(q(t), a(t))$ — функція виходів яку реалізує автомат Мілі;

$\lambda_2(q(t))$ — функція виходів, яку реалізує автомат Мура.

Із закону функціонування автомата Мілі (система рівнянь 8.2.1) випливає, що його вихідний сигнал $в(t) = \lambda_1(q(t), a(t))$ видається в час дії вхідного сигналу $a(t)$ і знаходженні автомата в стані $q(t)$, а автомата Мура (система рівняння 8.2.2), вихідний сигнал його $Z(t) = \lambda_2(q(t))$ видається весь час, поки автомат перебуває в стані $q(t)$.

В подальшому для кращого розуміння задання і перетворення автоматів буде використовуватись також сигнал $в(t)$ в якості вихідного сигналу $Z(t)$.

С-автомат індукує вихідні стани автомата Мілі і Мура одночасно. З практичної точки зору це вигідно використовувати при проектуванні електронних обчислювальних машин. Так, наприклад, частина станів пристрою керування, таких, як передача із одного регістра в інший, нарощування лічильників і т.п., за часом збігаються із вхідним сигналом, який трактується синхроімпульсом, тоді як, наприклад, сигнали подачі на комбінаційний суматор або дешифратор пам'яті значно довші в часі і визначаються часом спрацювання суматора та дешифратора. Ці сигнали зручно ототожнювати із станами автомата Мілі і Мура. Таким чином, з практичної точки зору доцільно розглядати С-автомат, моделі автоматів Мілі і Мура.

8.3. Способи задання автоматів

Для задання автоматів застосовують три способи: табличний, графічний і за допомогою матриці переходів. Серед них найбільш часто використовують табличний і графічний способи.

Опис роботи повністю визначеного автомата Мілі **табличним способом** за допомогою таблиць переходів і виходів ілюструється на прикладі автомата W_1, приведеного в табл. 8.3.1 і табл. 8.3.2.

Таблиця 8.3.1

a_i \ q_j	q_0	q_1	q_2
a_1	q_2	q_2	q_1
a_2	q_1	q_1	q_0

Таблиця 8.3.2

q_i \ q_j	q_0	q_1	q_2
a_1	$в_1$	$в_1$	$в_3$
a_2	$в_2$	$в_3$	$в_2$

Стрічки цих таблиць відповідають вхідним сигналам, а стовпчики — станам. Крайній зліва стовпчик позначений початковим станом q_0. На пересіченні стовпчика q_j і стрічки a_i в таблиці переходів ставиться стан $q_s = \delta(a_i, q_j)$, в який автомат переходить із стану q_j у стан q_s, а в таблиці виходів — відповідний цьому переходу вихідний сигнал $в_к = \lambda(a_i, q_j)$.

Оскільки в автоматі Мура вихідний сигнал залежить тільки від стану, то цей автомат задається одною **відміченою** таблицею переходів, в якій кожному її стовпчику приписано, крім стану q_j, ще і вихідний сигнал $в_к = \lambda(q_j)$, відповідний даному стану. Приклад табличного задання повністю визначеного автомата Мура W_2, приведений у відміченій таблиці переходів, табл. 8.3.3.

Таблиця 8.3.3

q_j \ $в_к$	$в_0$	$в_1$	$в_2$	$в_3$	$в_4$
a_i \ q_j	q_0	q_1	q_2	q_3	q_4
a_1	q_1	q_0	q_3	q_1	q_3
a_2	q_2	q_3	q_4	q_0	q_2

Для часткового визначеного автомата Мілі, в якого функції переходів $\delta(a_i(t), q_j(t))$ і виходів $\lambda(a_i(t), q_j(t))$ визначені не для всіх пар $(a_i, в_j) \in A \times B$, де $i \in n$, а $j \in m$, то на містах невизначених станів і вихідних сигналів ставлять прочерк. У табл. 8.3.4 і табл. 8.3.5 приведений частково визначений автомат Мілі W_3.

Таблиця 8.3.4

a_i \ q_j	q_0	q_1	q_2	q_2
a_1	q_1	q_0	–	q_2
a_2	q_2	–	q_3	–

Таблиця 8.3.5

a_i \ q_j	q_0	q_1	q_2	q_3
a_1	$в_1$	$в_2$	–	$в_2$
a_2	$в_1$	–	$в_3$	–

Аналогічно табличним способом задають і частково визначені автомати Мура.

Інколи для задання автоматів Мілі використовують одну спільну таблицю переходів і виходів, у якій на пересіченні стовпчика q_j і стрічки a_i записують наступний стан і видаваний на переході q_s вихідний сигнал $в_к$. Автомат W_4, заданий спільною таблицею переходів і виходів, приведений у табл. 8.3.6.

Таблиця 8.3.6

a_i \ q_j	q_0	q_1	q_2	q_2
a_1	$q_1/в_1$	$q_0/в_2$	$q_0/в_2$	$q_1/в_2$
a_2	$q_2/в_2$	—	$q_1/в_1$	—

Задання C — автоматів табличним способом аналогічно заданню автоматів Мілі і Мура цим способом.

Графічний спосіб задання автоматів Мілі такий. Стани автоматів зображують вершинами графа, які з'єднують дугами. Дві вершини графа автомата q_0 і $q_к$ (початковий стан і стан переходу) з'єднують дугою, направленою від q_0 до $q_к$, якщо в автоматі є пере-

хід від q_0 до $q_к$, тобто якщо $q_к = \delta\,(q_0\,(t), a_i\,(t))$ при деякому $a_i \in A$. Дузі $(q_0, q_к)$ графа автомата приписують вихідний сигнал $в_j = \lambda\,(q_0\,(t), a_i(t))$, якщо він визначений, і роблять прочерк — у противному разі рис. 8.3.1. Якщо перехід автомата із стану q_0 до стану $q_к$ відбувається під дією декількох вхідних сигналів, то тоді дузі $(q_0, q_к)$ для автомата Мілі приписують усі ці вхідні і відповідні вихідні сигнали.

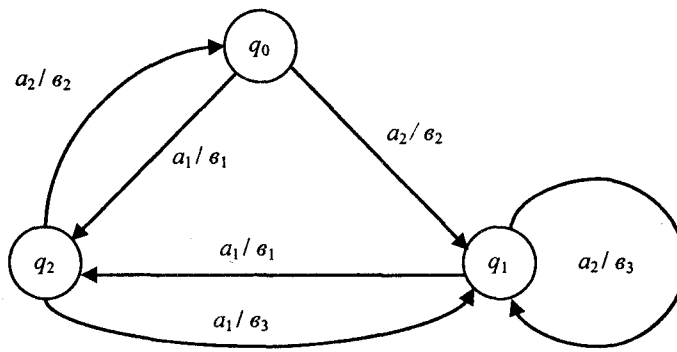


Рис. 8.3.1.

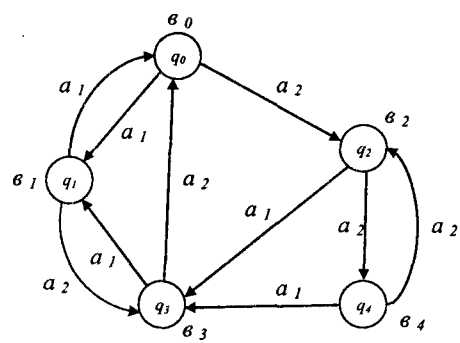


Рис. 8.3.2

При заданні автомата Мура за допомогою графа вихідний сигнал $в_к = \lambda\,(q_s)$ записують, як правило, поруч із його станом, рис. 8.3.2.

На рис. 8.3.1 і рис. 8.3.2 приведені автомати W_1 (Мілі) і W_2 (Мура), задані у вигляді графів, які раніше були задані за допомогою таблиць переходів і виходів.

При задані *С*-автоматів у вигляді графів вихідні сигнали типу 1 приписують дугам графа, поруч із вхідними сигналами, а вихідні сигнали типу 2 — станам, яким вони відповідають.

Тобто, графічне задання *С*-автомата аналогічне спільному графічному заданню автоматів Мілі та Мура.

Матричний спосіб задання автоматів є математичною копією графа переходів. За допомогою цього способу є можливість формалізувати ряд операцій, які на графі переходів можуть бути виконані візуально. Тому матриці переходів мають переваги в тих випадках, коли ці операції не можуть бути виконані людиною – дослідником, тобто коли вони не можуть бути виконані візуально або коли граф переходів настільки складний, що використання візуальної методики є марним.

Якщо автомат W має S станів, то матриця переходів складається із S стрічок і S стовпчиків і позначається $[W]$. Нехай $Q = \{q_0, q_1, q_2, \ldots, q_s\}$ — множина станів автомата Мілі W, а перехід автомата із стану q_i до стану q_j позначимо дугою ℓ_{ij}. Тоді елемент (i, j), який позначає дугу ℓ_{ij} графа переходів автомата із стану q_i до стану q_j, буде розміщений на пересіченні i-ої стрічки і j-го стовпчика матриці $[W]$ та знаходиться як

$$l_{ij} = \begin{cases} l_{ij}, \text{ якщо } l_{ij} \text{ є}, \\ 0, \text{ якщо } l_{ij} \text{ відсутня}. \end{cases}$$

Матриця переходів для автомата Мілі W_1, який заданий за допомогою таблиці переходів і виходів, або графічно (рис. 8.3.1), приведена на рис. 8.3.3.

$$[W_1] = \begin{array}{c} \\ q_0 \\ q_1 \\ q_2 \end{array} \begin{array}{c} \begin{array}{ccc} q_0 & q_1 & q_2 \end{array} \\ \begin{pmatrix} 0 & 0 & a_2 / в_2 \\ a_2 / в_2 & a_2 / в_3 & a_1 / в_3 \\ a_1 / в_1 & a_1 / в_1 & 0 \end{pmatrix} \end{array}$$

Рис. 8.3.3

Автомати Мура і *С*-автомати матрицею переходів не задаються.

8.4. Перетворення автоматів Мура в автомати Мілі

Нехай задано автомат Мура

$$W_y = \{A_y, B_y, Q_y, \delta_y, \lambda_y, q_{oy}\},$$

у якого:

$$A_y = \{a_1, a_2, \ldots, a_n\};$$
$$B_y = \{в_1, в_2, \ldots, в_m\};$$
$$Q_y = \{q_0, q_1, q_2, \ldots, q_S\};$$
$$\delta_y = \delta\,(a_i\,(t), q_j\,(t));$$
$$\lambda_y = \lambda\,(q_j\,(t));$$

q_{oy} — початковий стан.

Побудуємо такий автомат Мілі

$$W_i = \{A_i, B_i, Q_i, \delta_i, \lambda_i, q_{oi}\},$$

у якого: $A_i = A_y$; $B_i = B_y$; $Q_i = Q_y$; $\delta_i = \delta_y$; $q_{oi} = q_{oy} = q_o$. Функцію його виходів λ_i визначимо таким чином: якщо в автоматі Мура $\delta_y = \delta\,(a_i, q_j) = q_s$ і $\lambda_y = \lambda\,(q_s) = в_к$, то в автоматі Мілі $\lambda_y = \lambda\,(a_i, q_j) = q_s = в_к$.

Перехід від автомата Мура до автомата Мілі ілюструється за допомогою графічного способу, рис. 8.4.1.

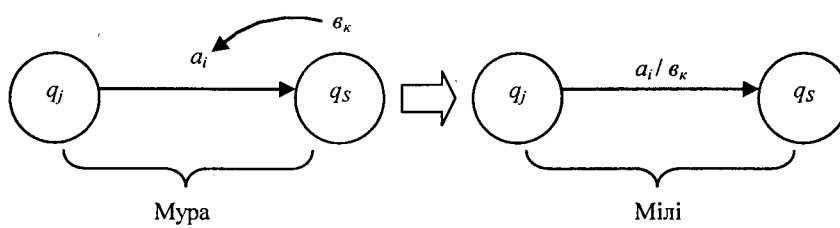


Рис. 8.4.1

Із рис. 8.4.1 випливає, що вихідний сигнал $в_к$ автомата Мура, записаний поруч із станом q_s, переноситься на всі дуги, які входять у цей стан.

В результаті цього переносу ми отримали фрагмент автомата Мілі з цими ж станами, вхідними і вихідними змінними.

При табличному способі задання автомата Мура таблиця переходів автомата Мілі збігається з таблицею переходів автомата Му-

ра, а таблиця виходів автомата Мілі випливає із таблиці переходів заміною символів q_s, які містяться на пересіченні стрічки a_i і стовпчика q_j, символом вихідного сигналу $в_к$, що відмічає стовпчик q_s у таблиці переходів автомата Мура.

Із самого способу побудови автомата Мілі W_i випливає, що він еквівалентний автомату Мура W_y. Дійсно, якщо деякий вхідний сигнал $a_i \in A$ подається на вхід автомата Мура W_y, який перебуває в стані q_j, то він перейде в стан $q_S = \delta_y (a_i, q_j)$ і видає вихідний сигнал $в_к = \lambda_y (q_s)$. Але відповідний автомат Мілі W_i із стану q_j також перейде в стан q_S, оскільки $\delta_i = (a_i, q_j) = \delta_y = (a_i, q_j) = q_S$ і видає той же вихідний сигнал $в_к$ згідно зі способом побудови функції виходів λ_i.

Таким чином, для вхідної послідовності однієї довжини поведінка автоматів Мура W_y і Мілі W_i повністю збігає. Якщо вхідна послідовність буде другої кінечної довжини, то за індукцією можна показати, що те слово, яке подається на входи автомата W_y і W_i призведе до появи однакових вихідних слів, і тому автомати Мура W_y і Мілі W_i є еквівалентні.

Приклад 8.4.1. Для автомата Мура, приведеного на рис. 8.4.2

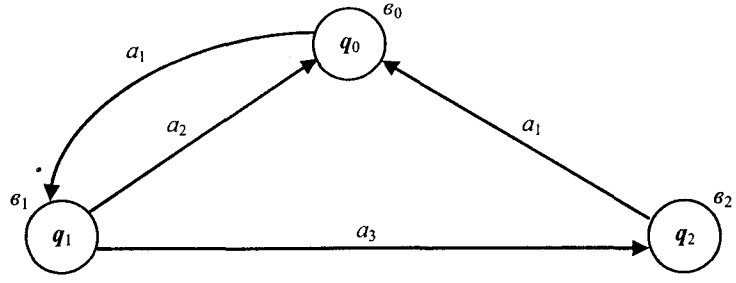


Рис. 8.4.2

знайти еквівалентний йому автомат Мілі.

Розв'язання. У відповідності з алгоритмом перетворення автомата Мура в еквівалентний йому автомат Мілі він буде мати вигляд, приведений на рис. 8.4.3.

Із прикладу 8.4.1 випливає, що при перетворенні автомата Мура в автомат Мілі кількість станів в автоматах не змінюється.

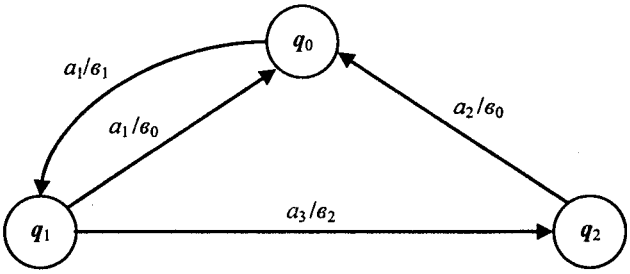


Рис. 8.4.3

8.5. Перетворення автоматів Мілі в автомати Мура

Нехай задано автомат Мілі

$$W_i = \{A_i, B_i, Q_i, \delta_i, \lambda_i, q_{oi}\},$$

у якого:

$$A_i = \{a_1, a_2, \ldots, a_n\};$$
$$B_i = \{в_1, в_2, \ldots, в_m\};$$
$$Q_i = \{q_0, q_1, q_2, \ldots, q_S\};$$
$$\delta_i = \delta\,(a_i\,(t), q_j\,(t));$$
$$\lambda_i = \lambda\,(a_i\,(t), q_j\,(t));$$

q_{oi} — початковий стан.

В першому випадку при перетворенні автомата Мілі в автомат Мура розглянемо автомати Мілі, які не мають перехідних станів, тобто станів, в які не входить жодна дуга, але які мають принаймні одну вихідну дугу.

Виходячи із цих умов, і побудуємо автомат Мура

$$W_y = \{A_y, B_y, Q_y, \delta_y, \lambda_y, q_{oy}\},$$

у якого: $A_y = A_i$; $B_y = B_i$

Для знаходження станів автомата Мура Q_y кожному стану $q_p \in Q_i$ поставим у відповідність множину Q_p можливих пар виду $(q_p, в_j)$, де $в_j$ — вихідний сигнал, приписаний вхідній дузі стану q_p, рис. 8.5.1.

Тоді множина можливих пар виду $q_p, в_j$ буде дорівнювати Q_p = $= \{(q_p, в_1), (q_p, в_2), (q_p, в_3)\}$. Число елементів у множині Q_p дорівнює числу різних вихідних сигналів на дугах автомата, які входять до стану q_p.

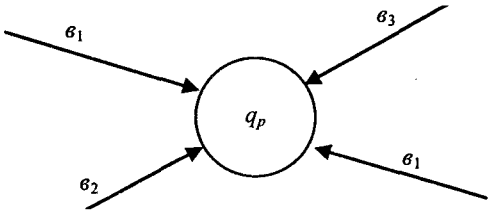


Рис. 8.5.1

Множину станів автоматів Мура із станів автомата Мілі отримаємо, як об'єднання множин Q_p таких станів, де $p = 0, 1, 2, \ldots, s$. Тобто

$$Q_y = \bigcup_{p=o}^{S} Q_p.$$

Функції виходів λ_y і переходів δ_y для пошукового автомата Мура знайдемо таким чином. Кожному стану автомата Мура, який являє собою пару $(q_p, в_m)$, поставим у відповідність вихідний сигнал $в_m$. Якщо в автоматі Мілі був перехід $\delta_i\ (a_i, q_j) = q_p$ і при цьому видавався вихідний сигнал $\lambda_i\ (a_i, q_j) = в_к$, то в автоматі Мура буде перехід із множини станів Q_j, породжених q_j у стані $(q_p, в_к)$ під дією того ж вхідного сигналу a_i, рис. 8.5.2.

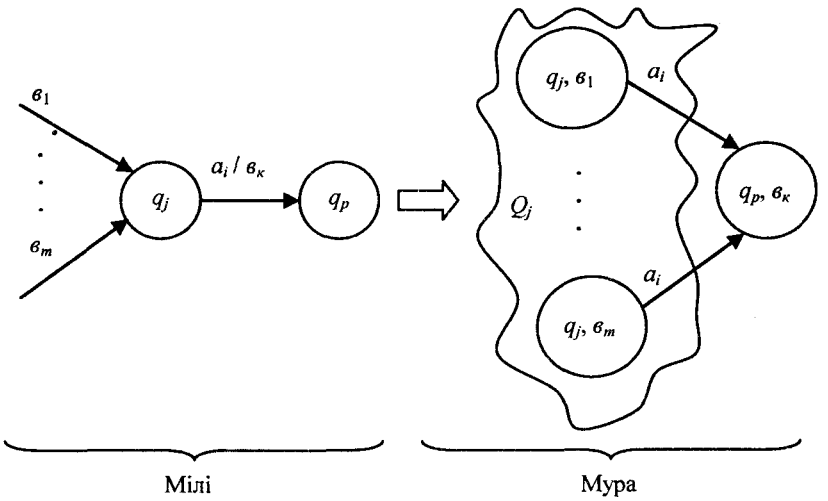


Рис. 8.5.2

В якості початкового сигналу пошукового автомата Мура q_{oy} може бути будь-який із станів множини, породжуваної початковим станом автомата Мілі. Але при цьому необхідно знати, що при порівнянні реакції автоматів W_i і W_y на будь-які вхідні слова не повинен ураховуватись вихідний сигнал у момент часу $t = 0$, який зв'язаний із станом q_{oy} автомата Мура.

Приклад 8.5.1. Для автомата Мілі, приведеного на рис. 8.5.3,

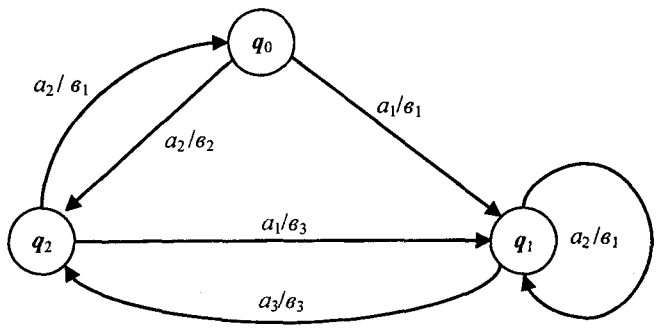


Рис. 8.5.3

знайти еквівалентний йому автомат Мура, де $A_i = \{a_1, a_2, a_3\}$; $B_i = \{в_1, в_2, в_3\}$; $Q_i = \{q_0, q_1, q_2\}$; δ_i і λ_i — позначають графом заданого автомата.

Розв'язання. За умовою побудови еквівалентного автомата Мура в ньому повина збігатися кількість вхідних і вихідних сигналів із заданим автоматом Мілі, тобто: $A_y = A_i = \{a_1, a_2, a_3\}$; $B_y = B_i = \{в_1, в_2, в_3\}$. Для знаходження кількості станів Q_y пошукового автомата Мура знайдемо множину пар, яка породжується кожним станом автомата Мілі. Така кількість станів автомата Мура дорівнює $Q_y = \bigcup_{p=o}^{S} Q_p$, де Q_p — стани автомата Мура, які породжуються кожним станом автомата Мілі. Вони, відповідно до станів автомата Мілі, визначаються як: $q_0 \Rightarrow Q_p^0 = (q_0, в_1) = C_0$; $q_1 \Rightarrow Q_p^1 = \{(q_1, в_1), (q_1, в_3)\} = C_1, C_2$; $q_2 \Rightarrow Q_p^2 = \{(q_2, в_2), (q_2, в_3)\} = C_3, C_4$, де q_0 — стан автомата Мілі, який породжує стан автомата Мура C_0; q_1 — стан автомата Мілі, який породжує стани автомата Мура C_1 і C_2; q_2 — стан автомата Мілі, який породжує стани автомата Мура C_3 і C_4.

Побудова функцій виходів автомата Мура λ_y відбувається таким чином. Кожному знайденому стану автомата Мура виду $(q_p, в_m)$ по-

ставим у відповідність вихідний сигнал $в_m$, який є другим елементом цієї пари. В нашому випадку це:

$$\lambda_y{}^0(C_0) = \lambda_y{}^1(C_1) = в_1;\quad \lambda_y{}^2(C_2) = \lambda_y{}^4(C_4) = в_3;\quad \lambda_y{}^3(C_3) = в_2.$$

Тобто ми знайшли, який вихідний сигнал в автоматі Мура відповідає кожному знайденому його стану із множини

$$Q_p = \{C_0, C_1, C_2, C_3, C_4\}.$$

Побудова функції переходів автомата Мура δ_y відбувається таким чином. Так як в автоматі Мілі є перехід із стану q_2 в стан q_1 під дією вхідного сигналу a_1 з видачею вихідного сигналу $в_3$, то із множини станів $Q_p{}^1 = \{C_1, C_2\}$, яка породжується станом q_1 автомата Мілі, в автоматі Мура повинен бути перехід із стану $(q_1, в_1) = C_1$ в стан $(q_2, в_3) = C_4$ під дією вхідного сигналу a_1. Аналогічно із стану $(q_1, в_3) = C_2$ множини $Q_p{}^1$ повинен бути перехід у стан $(q_1, в_1) = C_1$ під дією вхідного сигналу a_2. Аналогічно будується функції переходів автомата Мура δ_y і для інших станів: $C_0 \xrightarrow{a_1} C_1$; $C_4 \xrightarrow{a_1} C_2$; $C_2 \xrightarrow{a_1} C_4$; $C_4 \xrightarrow{a_2} C_0$; $C_0 \xrightarrow{a_2} C_3$; $C_3 \xrightarrow{a_2} C_0$; $C_3 \xrightarrow{a_1} C_2$, де зверху стрілки показана вхідна змінна, під дією якої цей перехід відбувається. В якості початкового стану можна вибрати тільки стан C_0, так як це єдиний стан, який породжується початковим станом q_0 автомата Мілі. Замінивши множину станів $\{C_0, C_1, C_2, C_3, C_4\}$ автомата Мура і накресливши в ньому знайдені функції переходів — виходів, отримаємо еквівалентний автомат Мура, приведений на рис. 8.5.4.

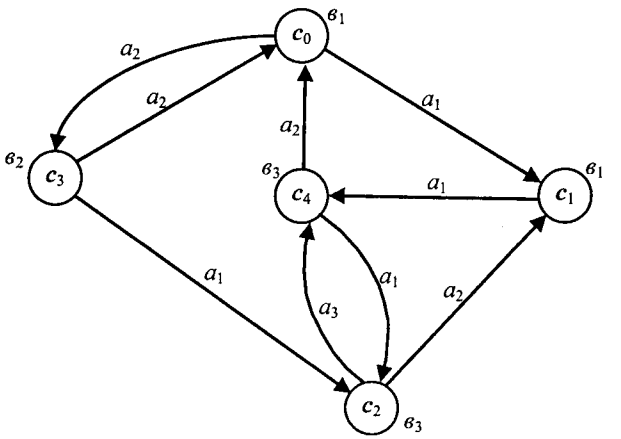


Рис. 8.5.4

Розглянемо тепер випадок перетворення автомата Мілі з перехідним станом в автомат Мура.

Приклад 8.5.2. Для автомата Мілі з перехідним станом, рис. 8.5.5.

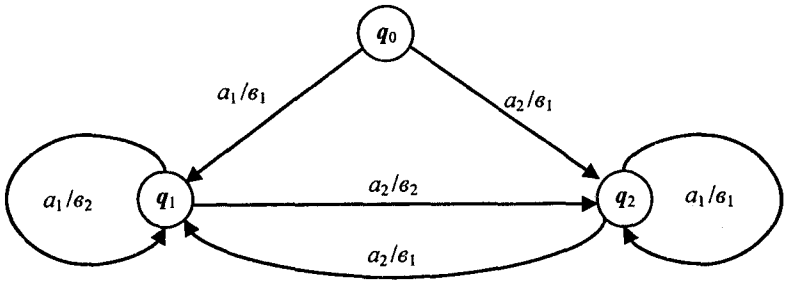


Рис. 8.5.5

знайти еквівалентний йому автомат Мура, де $A_i = \{a_1, a_2\}$; $B_i = \{в_1, в_2\}$; $Q_i = \{q_0, q_1, q_2\}$; δ_i і λ_i — визначають графом заданого автомата.

Розв'язання. За умови побудови еквівалентного автомата Мура в ньому повинна збігатися кількість вхідних і вихідних сигналів із заданим автоматом Мілі, тобто: $A_y = A_i = \{a_1, a_2\}$; $B_y = B_i = \{в_1, в_2\}$. Для знаходження кількості станів Q_y пошукового автомата Мура встановимо множину пар, яка породжується кожним станом автомата Мілі, крім перехідного стану q_0. Як і у попередньому випадку, вони визначаються, як: $q_1 \Rightarrow Q_p^{\,1} = \{(q_1, в_1), (q_1, в_2)\} = C_1, C_2$; $q_2 \Rightarrow Q_p^{\,2} = \{(q_2, в_1), (q_2, в_2)\} = C_3, C_4$. До цієї множини станів автомата Мура додамо стан $Q_p^{\,0} = (q_0, -) = C_0$, який породжений перехідним станом q_0 автомата Мілі і невизначеним вихідним сигналом у стані C_0. Тобто, множина станів пошукового автомата Мілі буде дорівнювати $Q_p = \{C_0, C_1, C_2, C_3, C_4\}$. Побудова функцій виходів і переходів пошукового автомата Мура відбувається аналогічно прикладу 8.5.1. Так для функцій виходів це буде:

$$\lambda_y^{\,1}(C_1) = \lambda_y^{\,3}(C_3) = в_1;\ \lambda_y^{\,2}(C_2) = \lambda_y^{\,4}(C_4) = в_2;$$

$\lambda_y^{\,0}(C_0)$ — у цьому стані вихідний сигнал є невизначеним, а функції переходів будують мати слідуючий вигляд:

$$\delta_y^{\,1} = C_0 \xrightarrow{a_1} C_1;\ \delta_y^{\,2} = C_0 \xrightarrow{a_2} C_3;\ \delta_y^{\,3} = C_3 \xrightarrow{a_2} C_1;$$
$$\delta_y^{\,4} = C_4 \xrightarrow{a_1} C_3;\ \delta_y^{\,5} = C_4 \xrightarrow{a_2} C_1;\ \delta_y^{\,6} = C_1 \xrightarrow{a_2} C_4;$$

$$\delta_y^{\,7} = C_1 \xrightarrow{a_1} C_2;\ \delta_y^{\,8} = C_2 \xrightarrow{a_2} C_4.$$

В якості початкового стану необхідно взяти породжений стан C_0. Тоді пошуковий автомат Мура, отриманий із автомата Мілі із перехідним станом, буде мати вигляд, приведений на рис. 8.5.6.

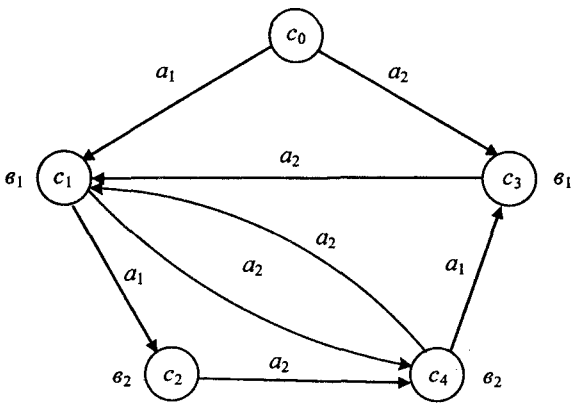


Рис. 8.5.6

Із прикладів 8.5.1 і 8.5.2 випливає, що при перетворенні автомата Мілі в автомат Мура, число станів в автоматі Мура збільшується за рахунок породжування окремими станами автомата Мілі не одного, а двох станів автомата Мура.

8.6. Ізоморфізм автоматів

Означення 8.6.1. Два автомати називають **ізоморфними** один до одного, якщо їх функції переходів і виходів однакові за винятком можливих різниць у позначенні їх станів.

Отже, якщо заданий автомат W, який представляє певну систему, то будь-який автомат, ізоморфний до W, може також представляти дану систему.

Приклад 8.6.1. Для заданого таблицею переходів і виходів автомата W, табл. 8.6.1 і табл. 8.6.2.

Таблиця 8.6.1

a_i \ q_j	q_1	q_2	q_3
a_1	q_2	q_1	q_2
a_2	q_1	q_3	q_3
a_3	q_3	q_2	q_1

Таблиця 8.6.2

a_i \ q_j	q_1	q_2	q_3
a_1	1	0	1
a_2	0	1	0
a_3	1	0	0

знайти ізоморфний до нього автомат W_1.

Розв'язання. У відповідності з означенням 8.6.1 будуємо автомат W_1, ізоморфний W. Для цього стани q_1, q_2, q_3 автомата W замінимо на стани q_3, q_2, q_1 в автоматі W_1. В результаті цього отримаємо таблиці переходів і виходів ізоморфного автомата W_1, табл. 8.6.3 і табл. 8.6.4

Таблиця 8.6.3

a_i \ q_j	q_1	q_2	q_3
a_1	q_2	q_3	q_2
a_2	q_3	q_1	q_1
a_3	q_1	q_2	q_3

Таблиця 8.6.4

a_i \ q_j	q_1	q_2	q_3
a_1	1	0	1
a_2	0	1	0
a_3	1	0	0

із яких випливає, що автомат W_1 видає на виході ті ж сигнали під дією вхідних сигналів, що й автомат W. Множина всіх різних автоматів, яку можна отримати в результаті всіх можливих перестановок станів автомата, дорівнює $n!$ (в нашому випадку 3!). Цю множину називають **сімейством перестановок** автомата.

8.7. Еквівалентність автоматів

Означення 8.7.1. Два автомата W_1 і W_2 називають **еквівалентними**, якщо кожному стану q_i автомата W_1 відповідає, принаймні, один еквівалентний йому стан в автоматі W_2 і якщо кожному стану q_j в автоматі W_2 відповідає, принаймні, один еквівалентний йому стан в автоматі W_1.

Таким чином, автомати W_1 і W_2 є еквівалентними тоді і тільки тоді, коли, спостерігаючи сигнали на їх виходах, неможливо відріз-

нити автомат W_1 у будь-якому із його станів від автомата W_2 і автомат W_2 у будь-якому із його станів від автомата W_1.

Автомати W_1 і W_2 будуть **розпізнавані** тоді і тільки тоді, коли буде хоча б один стан у W_1, який не еквівалентний ніякому стані в W_2 або коли буде хоча б один стан у W_2, який не еквівалентний ніякому стану в автоматі W_1.

Еквівалентність автоматів W_1 і W_2 позначають $W_1 = W_2$ або $W_1 \sim W_2$.

Еквівалентність автоматів має властивості рефлексивності, симетричності і транзитивності. Тому еквівалентність автоматів можна розглядати як звичайне відношення еквівалентності і застосовувати його до множин автоматів будь-якої потужності.

Визначення еквівалентності автоматів означає, що два автомати, які мають однакові таблиці переходів-виходів або графи чи матриці, повинні бути еквівалентними. Так як еквівалентність або розпізнаваність пари станів не залежать від позначень станів то два ізоморфних автоматів також повинні бути еквівалентними.

Приклад 8.7.1. Показати, що автомати, приведені на рис. 8.7.1, еквівалентні.

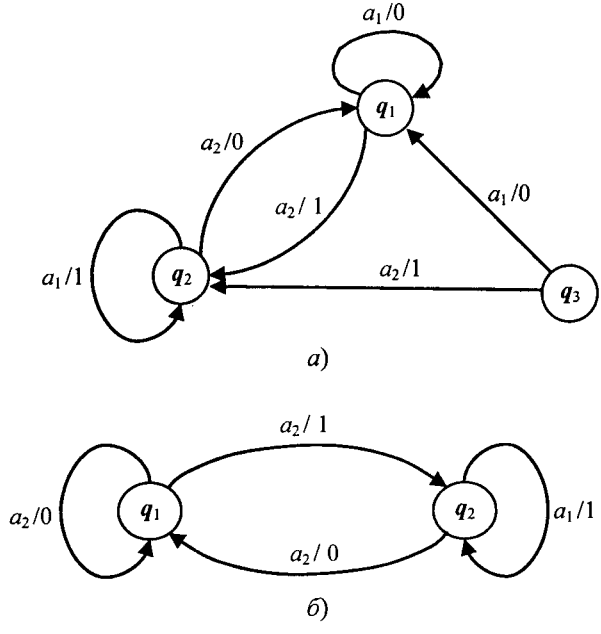


Рис. 8.7.1

Розв'язання. Із рисунка випливає, що стани q_1 і q_2 автомата, рис. 8.7.1 *а*, еквівалентні відповідно станам q_1 і q_2 автомата, рис. 8.7.1 *б*. Крім того, стани q_1 і q_3 автомата, рис. 8.7.1 а, еквівалентні між собою, а тому еквівалентні стану q_1 автомата, рис. 8.7.1 *б*. Таким чином для кожного стану автомата, рис. 8.7.1 *а*, можна знайти еквівалентний стан в автоматі, рис. 8.7.1 *б*, і навпаки. А це означає, що автомати, приведені на рис. 8.7.1, є еквівалентні.

Приклад 8.7.2. Показати, що автомати, приведені на рис. 8.7.2, неекві валентні.

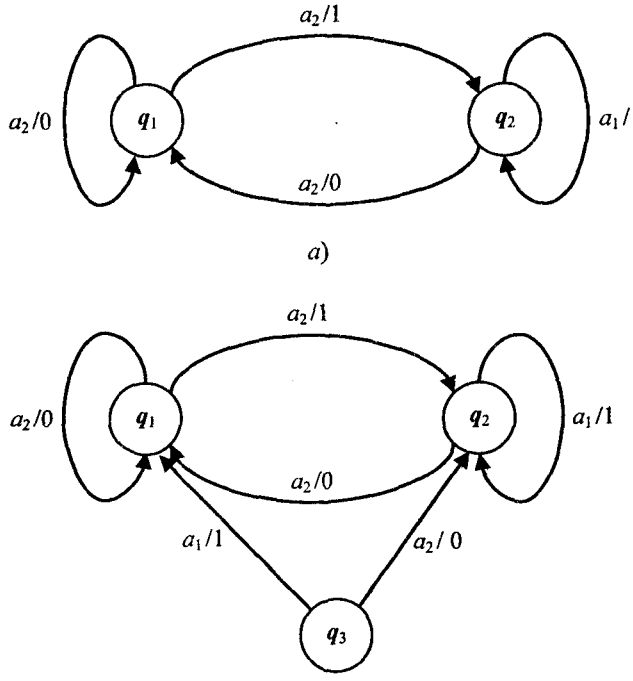


Рис. 8.7.2

Із рисунка випливає, що при порівнянні двох автоматів, вони є однаковими, якщо не враховувати стан q_3 автомата, рис. 8.7.2 *б*. Тому без урахування стану q_3 можна сказати, що для кожного стану автомата, рис. 8.7.2 *а*, ми знаходимо еквівалентний стан автомата,

рис. 8.7.2 *б*. Однак, оскільки пари станів (q_1, q_3) і (q_2, q_3) автомата, рис. 8.7.2 *б*, є явно не еквівалентними і тому розпізнавані, а це засвідчує те, що дані автомати не є еквівалентні.

8.8. Мінімізація автоматів

В даному параграфі розглядається мінімізація повністю визначених автоматів.

Означення 8.8.1. Два стани автомата q_i і q_j називають **еквівалентними**, якщо при $(q_i, a) = (q_j, a)$ для будь-яких вхідних слів, реакції автомата в цих станах збігається.

При мінімізації повністю визначених автоматів множину станів заданого автомата розбивають на непересічні класи еквівалентних між собою станів. У кожному класі всі стани, крім одного, є збитковими, і автомат їх не розпізнає, так як реагує на будь-які вхідні слова у будь-якому із класів однаково.

Автомат W буде мінімальним, якщо із $q_i \sim q_j$ випливає $q_i = q_j$.

Для знаходженя мінімального автомата необхідно розділити P множин повністю визначеного автомата на класи еквівалентних між собою станів. Для цього введемо більш послаблене поняття відношення еквівалентності.

Означення 8.8.2. Два стани автомата q_i і q_j називають ***k*-еквівалентними**, якщо при $(q_i, a_k) = (q_j, a_k)$ для будь-яких вхідних слів a_k довжини k, вихідні сигнали автомата збігаються. Якщо стани автомата не k-еквівалентні, то вони k-розпізнавані.

Алгоритм знаходження P множин станів пошукового мінімального автомата згідно з Ауфенкампом–Хоном має такий вигляд.

1. Множину станів заданого автомата Q послідовно розділяють на класи одно -, двох -, ..., $k + 1$ — еквівалентних між собою станів до тих пір, поки на певному $k + 1$ кроці не виявиться, що $P_{k+1} = P_k$.

2. В кожному із знайдених на кроці один алгоритму класі еквівалентності довільно вибирають один стан, який є представником цього класу. Вибрані представники класів і є множиною станів пошукового мінімального автомата, яку позначають Q^*.

3. В пошуковому мінімальному автоматі $\{A, B, Q^*, \delta^*, \lambda^*, q_0\}$ будуємо функції переходів δ^* і виходів λ^*, для чого в матрицях переходів і виходів викреслюються стовпчики, які відповідають тим

станам, що відсутні в Q^*, і в стовпчиках матриці переходів, що залишилися, всі стани заміняють на еквівалентні із Q^*.

4. Початковим станом може бути будь-який із класу станів, еквівалентних q_0, наприклад, сам q_0.

Приклад 8.8.1. Для автомата Мілі, приведеного на рис. 8.8.1,

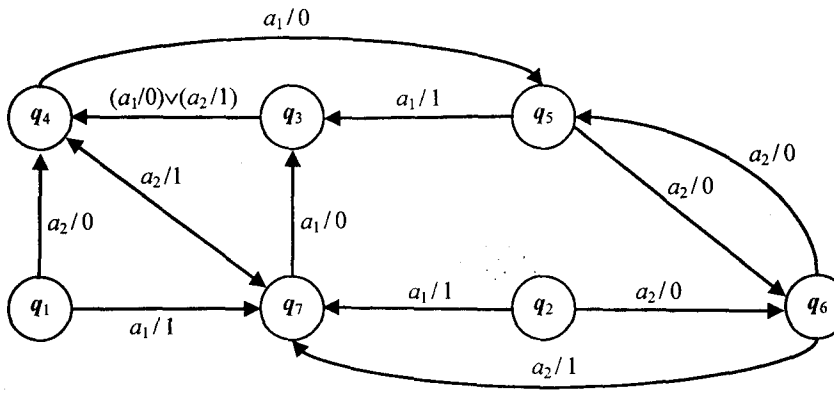


Рис. 8.8.1

знайти графічний мінімальний автомат, еквівалентний заданому.

Розв'язання. Для заданого графічного автомата будуємо таблицю переходів і виходів, у якому: $A = \{a_1, a_2\}$; $B = \{0, 1\}$; $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$; $q_0 = q_1$, а функції переходів і виходів, які окреслені гріфічно, представимо відповідними таблицями — табл. 8.8.1 і табл. 8.8.2.

Таблиця 8.8.1

a_i \ q_j	q_1	q_2	q_3	q_4	q_5	q_6	q_7
a_1	q_7	q_7	q_4	q_5	q_3	q_5	q_3
a_2	q_4	q_6	q_4	q_7	q_6	q_7	q_4

Таблиця 8.8.2

a_i \ q_j	q_1	q_2	q_3	q_4	q_5	q_6	q_7
a_1	1	1	0	0	1	0	0
a_2	0	0	1	1	0	1	1

Об'єднанням однакових стовпчиків у таблиці виходів, табл. 8.8.2, отримаємо одноеквівалентні розбиття P_1

$$P_1 = \{\pi_{11}, \pi_{12}\}, \text{ де } \pi_{11} = \{q_1, q_2, q_5\}, \pi_{12} = \{q_3, q_4, q_6, q_7\}.$$

За допомогою таблиці переходів, табл. 8.8.1, будуємо таблицю для знаходження P_1, табл. 8.8.3,

Таблиця 8.8.3

	π_{11}			π_{12}			
a_i \ q_j	q_1	q_2	q_5	q_3	q_4	q_6	q_7
a_1	π_{12}	π_{12}	π_{12}	π_{12}	π_{11}	π_{11}	π_{12}
a_2	π_{12}	π_{12}	π_{12}	π_{12}	π_{12}	π_{12}	π_{12}

до якої заносимо значення одноеквівалентного розбиття, отриманого раніше. Із табл. 8.8.3 знаходимо двохеквіваленті розбиття, так як одноеквівалентні будуть двохеквівалентні тоді і тільки тоді, коли вони переводяться будь-яким вхідним сигналом (буквою) в стан свого еквівалентного класу. Тому із табл. 8.8.3 випливає, що двохеквівалентне розбиття P_2 буде дорівнювати

$$P_2 = \{\pi_{21}, \pi_{22}, \pi_{23}\}, \text{ де } \pi_{21} = \{q_1, q_2, q_5\}; \pi_{22} = \{q_3, q_7\}; \pi_{23} = \{q_4, q_6\}.$$

Знову за допомогою таблиці переходів, табл. 8.8.1, будуємо таблицю для знаходженя P_2, табл. 8.8.4,

Таблиця 8.8.4

	π_{21}			π_{22}		π_{23}	
a_i \ q_j	q_1	q_2	q_5	q_3	q_7	q_4	q_6
a_1	π_{22}	π_{22}	π_{22}	π_{23}	π_{23}	π_{21}	π_{21}
a_2	π_{23}	π_{23}	π_{23}	π_{23}	π_{23}	π_{22}	π_{22}

за якою знаходимо трьохеквівалентні розбиття P_3, які матимуть вигляд

$$P_3 = \{\pi_{31}, \pi_{32}, \pi_{33}\}, \text{ де } \pi_{31} = \{q_1, q_2, q_5\}; \pi_{32} = \{q_3, q_7\}; \pi_{33} = \{q_4, q_6\}.$$

Оскільки еквівалентні розбиття $P_2 = P_3$, то, згідно з кроком алгоритму один, множина станів шуканого мінімального автомата розділена на еквівалентні між собою класи, які дорівнюють

$$\pi_{21} = \pi_{31} = \{q_1, q_2, q_5\}; \pi_{22} = \pi_{32} = \{q_3, q_7\}; \pi_{23} = \pi_{33} = \{q_4, q_6\}.$$

Отже, вихідне розбиття на класи еквівалентності матиме вигляд

$$P = \{\{q_1, q_2, q_5\}, \{q_3, q_7\}, \{q_4, q_6\}\}.$$

Для побудови шуканого автомата довільно виберем по одному стану з кожного класу для знаходження множини станів Q^* мінімального автомата

$$Q^* = \{q_1, q_3, q_4\}.$$

Початковим станом автомата залишимо стан q_1.

Використовуючи крок три алгоритму, будуємо таблиці переходів і виходів мінімального автомата, табл. 8.8.5 і табл. 8.8.6.

Таблиця 8.8.5

a_i \ q_j	q_1	q_3	q_4
a_1	q_3	q_4	q_1
a_2	q_4	q_4	q_3

Таблиця 8.8.6

a_i \ q_j	q_1	q_3	q_4
a_1	1	0	0
a_2	0	1	1

За знайденими таблицями переходів і виходів накреслимо мінімальний автомат Мілі, який приведений на рис. 8.8.2.

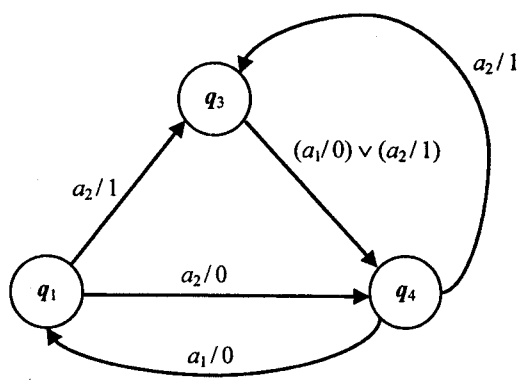


Рис. 8.8.2

Початковим станом знайденого мінімального автомата Мілі є стан q_1. Мінімізацію автоматів Мура виконують аналогічно описаній мінімізації автомата Мілі.

8.9. Канонічний метод структурного синтезу автоматів

Основним завданням **структурної теорії автоматів** є пошук спільних дій для побудови структурних схем автоматів на основі тієї чи іншої вибраної схемо-технічної елементної бази. В структурних автоматах ураховується структура вхідних і вихідних сигналів автомата, а також його внутрішній стан на рівні структурних схем заданої елементної бази.

Тепер найбільш розповсюдженим структурним алфавітом для синтезу автоматів є **двійковий**, що пояснюється простотою його використання в сучасних схемо-технічних елементах, системах і електронних обчислювальних машинах (ЕОМ). Крім того, для двійкового алфавіту розроблений математичний апарат булевих функцій, який дозволяє виконувати більшість операцій над схемами формально. Тому для рішення задач структурного синтезу автоматів будемо використовувати двійковий структурний алфавіт.

На етані структурного синтезу завчасно вибирають елементарні автомати, із яких потім шляхом їх композиції будують структурну схему отриманого на етапі абстрактного синтезу автомата Мілі, Мура або С–автомата. Якщо рішення задачі структурного синтезу є, то кажуть, що задана система автоматів **структурно повна**.

Певна система елементарних автоматів, що містить автомат Мура з нетрівіальною пам'ятю, яка володіє повною системою переходів, повною системою виходів і містить певну функціональну повну систему логічних елементів, є структурно повною.

Для правильної роботи схеми автомата не можна дозволити, щоб сигнали на вході запамятовуючих елементів безпосередньо приймали участь в утворенні вихідних сигналів, які в цей же час подавались на ці входи. У зв'язку із цим запам'ятовуючими елементами повинні бути використані не автомати Мілі, а автомати Мура.

Таким чином, структурно повна система елементарних автоматів повинна мати хоча б один автомат Мура. В той же час для синтезу будь-яких автоматів з мінімальним числом елементів пам'яті необхідно в якості таких елементів вибирати автомати Мура, які

мають повну систему переходів і повну систему виходів, — так звані **повні автомати.**

Канонічний метод структурного синтезу розглянемо для *С* — автомата, оскільки ця модель є об'єднанням моделі Мілі та Мура, рис. 8.9.1.

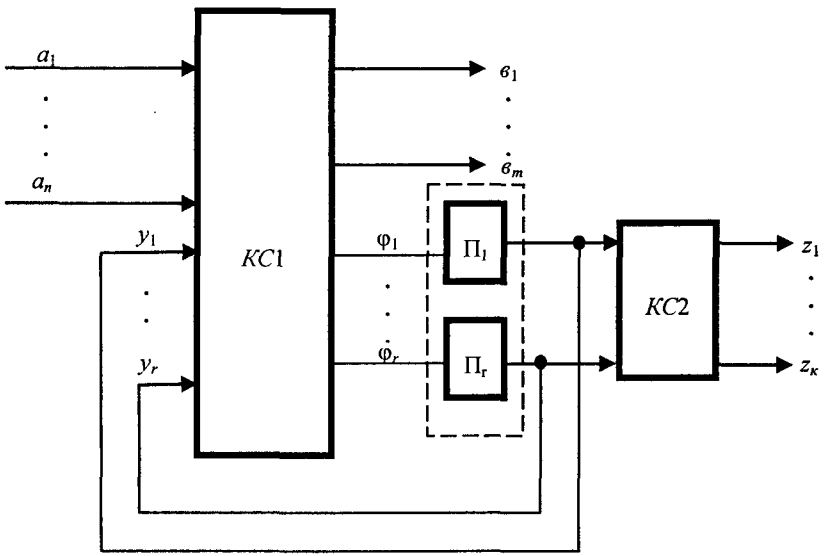


Рис. 8.9.1

Структурна схема *С*–автомата складається з трьох частин: пам'яті і двох комбінаційних схем КС1 та КС2. Якщо необхідно синтезувати автомат Мілі, то в *С*–автоматі функцію λ_2 не задають і буде відсутня комбінаційна схема КС2. У випадку моделі Мура незаданою буде функція λ_1 і в комбінаційній схемі КС1 будуть відсутні вихідні сигнали $в_1 \ldots в_m$.

Комбінаційна схема КС1 служить для формування вихідних сигналів типу 1 і вхідних сигналів для автоматів пам'яті, а КС2 для формування сигналів типу 2.

Пам'ять автомата складається з вибраних автоматів пам'яті — елементарних автоматів Мура $П_1, \ldots, П_r$. Після вибору елементів пам'яті кожний стан абстрактного *С* — автомата кодується в структурному автоматі. Якщо всі автомати $П_1, \ldots, П_r$ однакові, то їх число буде дорівнювати

$$n =] \log_{\Theta} M [, \qquad (8.9.1)$$

де Θ — число станів елементарного автомата пам'яті;

M — число станів абстрактного C – автомата;

] [— знак означає, що береться найбільше ціле число від результату логарифму.

При двійковій системі числення $\Theta = 2$.

Таким чином, після вибору елементів пам'яті і кодування станів синтез структурного автомата зводиться до синтезу комбінаційної схеми, яка реалізує функції:

$$\begin{aligned}
&в_1 = в_1 (a_1, \ldots, a_n, y_1, \ldots, y_r),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&в_m = в_m (a_1, \ldots, a_n, y_1, \ldots, y_r),\\
&\varphi_1 = \varphi_1 (a_1, \ldots, a_n, y_1, \ldots, y_r),\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&\varphi_r = \varphi_r (a_1, \ldots, a_n, y_1, \ldots, y_r),\\
&z_1 = z_1 (y_1, \ldots, y_r),\\
&\ldots\ldots\ldots\ldots\\
&z_к = z_к (y_1, \ldots, y_r),
\end{aligned}$$

де $y = (y_1, \ldots, y_r)$ — функції зворотного зв'язку від пам'яті автомата до комбінаційної схеми КС1;

$\varphi = (\varphi_1, \ldots, \varphi_r)$ — функція збудження пам'яті автомата.

Тобто, канонічний метод структурного синтезу автоматів дозволяє звести завдання структурного синтезу довільних автоматів до завдання синтезу комбінаційних схем. Результатом канонічного методу структурного синтезу є система логічних рівнянь, яка виражає залежність вихідних сигналів автомата і сигналів, що видаються на входи запам'ятовуючих елементів, від сигналів, які поступають на вхід автомата в цілому, і сигналів, що знімаються з виходів елементів пам'яті. Ці рівняння називають канонічними.

8.10. Графічний метод структурного синтезу автоматів

При графічному методі синтезу структурний C–автомат (автомат Мілі, Мура) представляють у вигляді графа. Дугам графа автомата Мура приписують кон'юнкції (диз'юнкції) змінних, які пере-

водять автомат із одного стану в інший, а станам — вихідні змінні, які автомат індуктує в цьому стані. Якщо розглядувати автомат Мілі, то його дугам приписують не тільки кон'юнкції (диз'юнкції) змінних, які переводять його з одного стану в інший, але ще і вихідний сигнал, який він при цьому індуктує.

Кожен стан структурного автомата кодується відповідним кодом. Розмірність коду залежить від кількості станів абстрактного автомата і визначається формулою 8.9.1. Для автоматів, які за умовою не є самокорегуючі, як правило, використовують звичайний двійковий код.

Приклад 8.10.1. Розробити схему управління комп'ютерним пристроєм, алгоритм роботи якого заданий абстрактним автоматом, рис. 8.10.1,

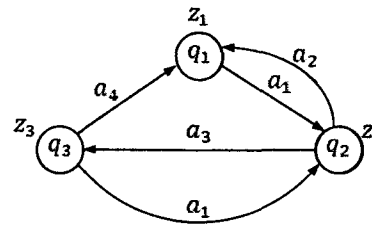


Рис. 8.10.1

де $a_1, ..., a_4$ — вхідні змінні пристрою; $z_1, ..., z_3$ — сигнали, які видає пристрій при управлінні; $q_1, ..., q_3$ — абстрактні позначення станів в яких перебуває пристрій при управлінні.

Розв'язання. Для забезпечення реалізації трьох станів абстрактного автомата q_1, q_2, q_3 необхідно в структурному автоматі, згідно із формулою $n =]\log_2 3[= 2$, мати два елементи пам'яті, які можуть задовольнити реалізацію чотирьох станів: 00; 01; 10; 11. Для кодування трьох станів абстрактного автомата використаємо кодові комбінації: $q_1 \rightarrow 00$; $q_2 = 01$; $q_3 = 10$. У результаті цього структурний автомат матиме вигляд, наведений на рис 8.10.2.

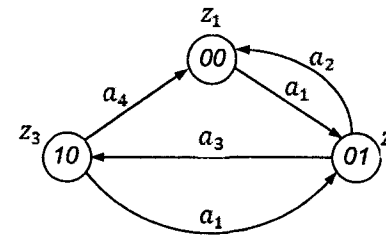


Рис. 8.10.2

Використовуючи відмічену для автомата Мура таблицю переходів, табл. 8.10.1,

Таблиця 8.10.1

q_j \ z_k	z_1	z_2	z_3
a_i \ q_j	00	01	10
a_1	01	-	01
a_2	-	00	-
a_3	-	10	-
a_4	-	-	00

отримаємо канонічні рівняння роботи схеми управління комп'ютерним пристроєм:

$$z_1 = \overline{y}_1 \cdot \overline{y}_2;\ z_2 = \overline{y}_1 \cdot y_2;\ z_3 = y_1 \cdot \overline{y}_2;$$
$$\varphi_1^1 = a_3;\ \varphi_1^0 = a_1 \vee a_4 \cdot \overline{y}_2;$$
$$\varphi_2^1 = a_1 \cdot \overline{y}_1 \vee a_1 = a_1 \cdot (\overline{y}_1 \vee 1) = a_1;$$
$$\varphi_2^0 = a_2 \cdot \overline{y}_1 \vee a_3.$$

де φ_1^1, φ_2^1 і φ_1^0, φ_2^0 — функції включення і виключення відповідно першого і другого елементів пам'яті структурного автомата Мура;

y_1, y_2 і $\overline{y}_1$, $\overline{y}_2$ — сигнали на виходах першого і другого елементів пам'яті, які відповідають логічним сигналам «1» і «0».

Функція φ_1 відповідає стану коду розряду, розміщеного зліва, а φ_2 — справа. Рівняння включення першого елемента пам'яті φ_1^1 отримують таким чином. У відміченій таблиці переходів розглядують усі переходи кодових станів цієї функції з «0» до «1» під дією вхідних змінних. До кон'юнкції вхідних змінних також записують і змінну другого елемента пам'яті, якщо вона не міняє свій знак при цьому переході. Якщо цей перехід для функції φ_1^1 відбувається не один раз, а, наприклад, два, то знайдені кон'юнкції змінних об'єднують знаком диз'юнкції.

Рівняння виключення першого елемента пам'яті φ_1^0 отримують аналогічно описаному з тою лише різницею, що при цьому розгля-

дують лише переходи із стану «1» до стану «0». Рівняння для функції φ_2 отримують аналогічно описаному для функції φ_1.

Схема управління комп'ютерним пристроєм, яка реалізує канонічні рівняння роботи структурного автомата Мура, приведена на рис. 8.10.3.

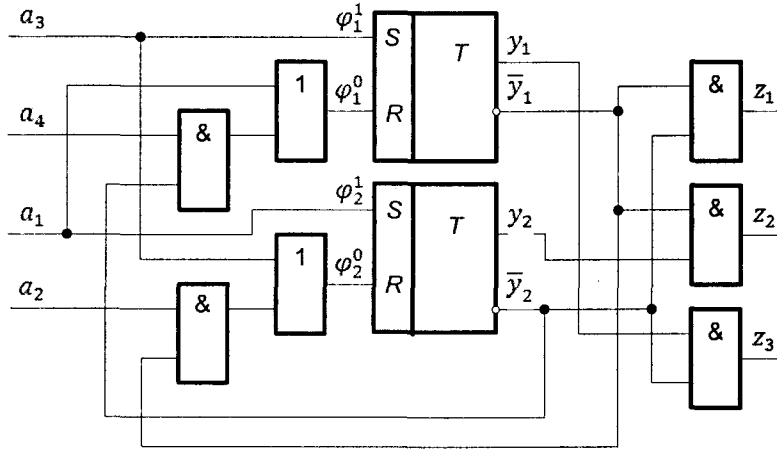
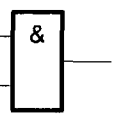
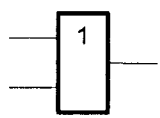


Рис. 8.10.3

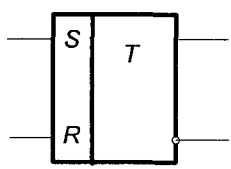
де



— функціональний елемент, який реалізує кон'юнкцію;



— функціональний елемент, який реалізує диз'юнкцію;



— функціональний елемент, який реалізує елемент памяті (*RS*–тригер).

Контрольні запитання

1. Що називають автоматом?
2. Які бувають автомати ?
3. Чим відрізняються тривіальні автомати від автоматів з пам'яттю?
4. Що розуміють під абстрактним автоматом?
5. Що таке структурний автомат?
6. Який автомат називають скінченим?
7. Що розуміють під функціями переходів і виходів автомата?
8. Який автомат називають повністю визначеним?
9. Що розуміють під частково визначеним автоматом?
10. Що таке синхронний автомат?
11. Який автомат називають асинхронним?
12. Що називають автоматом Мілі?
13. Що називають автоматом Мура?
14. Що називають *С*-автоматом?
15. Яка різниця між автоматами Мілі і Мура?
16. Яка різниця між автоматом Мілі і *С*- автоматом?
17. Яка різниця між автоматом Мура і *С*- автоматом?
18. Де можуть бути застосовані автомати Мілі, Мура та *С*-автомати?
19. Скількома способами можна задавати автомати?
20. Які автомати можна задавати табличним, графічним і матричним способами?
21. Чому не можна задавати автомати Мура і *С*-автомати матричним способом?
22. Якими способами можна задавати частково визначені автомати?
23. Визначте переваги і недоліки графічного способу задання автоматів перед табличним та матричним?
24. Які необхідні обов'язкові умови для перетворення автомата Мура в автомат Мілі ?
25. Сформулюйте алгоритм перетворення автомата Мура в автомат Мілі.
26. Як визначити початковий стан пошукового автомата Мілі ?
27. Як отримати таблицю виходів пошукового автомата Мілі ?
28. Сформулюйте алгоритм перетворення автомата Мілі в автомат Мура.
29. Як визначають число станів пошукового автомата Мура?
30. Як знаходять функції переходів і виходів пошукового автомата Мура?
31. Як визначають початковий стан пошукового автомата Мура?
32. Які автомати називають ізоморфними?
33. Чому дорівнює сімейство перестановок ізоморфного автомата?
34. Чи можуть бути ізоморфними автомати, в яких функції переходів різні?

35. Які автомати називають еквівалентними?
36. Які автомати називають розпізнаваними?
37. Які властивості має еквівалентність автоматів?
38. Що розуміють під еквівалентністю станів автомата?
39. Що розуміють під еквівалентністю двох автоматів?
40. Сформулюйте алгоритм мінімізації автомата Мілі?
41. Що розуміють під класами еквівалентних між собою станів в автоматі?
42. Як будують функції переходів і виходів при мінімізації автомата?
43. Який стан необхідно вибрати початковим у знайденому мінімальному автоматі?
44. Що є завданням структурної теорії автоматів?
45. Який алфавіт використовують для структурної теорії автоматів і чому?
46. Яку систему автоматів називають структурно повною?
47. Як визначають кількість елементів пам'яті структурного автомата, якщо відома кількість елементів пам'яті абстрактного автомата?
48. До чого зводиться синтез структурного автомата?
49. Які рівняння при структурному синтезі автомата називають канонічними?
50. Опишіть графічний метод структурного синтезу автомата.
51. Чим відрізняються автомати Мілі від автоматів Мура при графічному методі структурного синтезу автоматів?
52. Як можна кодувати внутрішні стани при графічному методі структурного синтезу автомата?

Задачі для самостійного розв'язування

1. За допомогою автомата Мілі, Мура або *C*-автомата описати системи:

а) управління вантажним ліфтом для трьохповерхового комплексу, який має кнопку виклику на кожному поверсі і працює таким чином, що: якщо натиснута одна кнопка, то ліфт рухається до поверху, на який вона натиснута, якщо натиснути дві або три кнопки, то він рухається до найнижчого поверху, на який натиснута кнопка.

б) накопичення числа одиниць за модулем 2, які надходять до неї у вигляді двійкових сигналів 0 і 1.

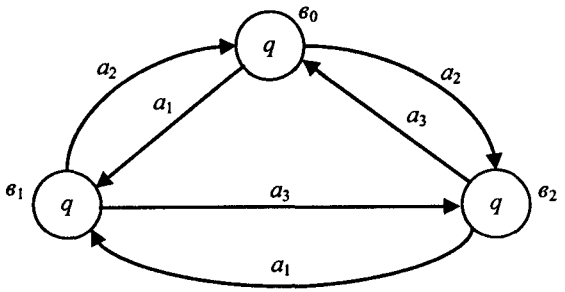
в) накопичення числа одиниць за модулем 3, які надходять до неї у вигляді двійкових сигналів 0 і 1.

2. Автомат Мілі заданий табличним способом, представити графічно і матрично.

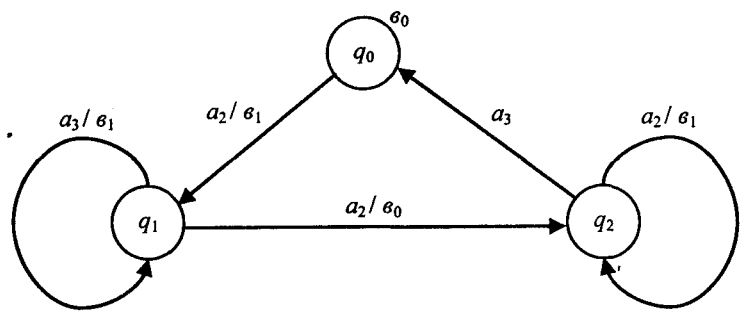
a_i \ q_j	q_0	q_1	q_2	q_3
a_1	q_3	q_2	—	q_3
a_2	—	q_0	q_0	—
a_3	q_2	—	q_1	q_2

a_i \ q_j	q_0	q_1	q_2	q_3
a_1	$в_3$	$в_2$	—	$в_3$
a_2	—	$в_0$	$в_0$	—
a_3	$в_2$	—	$в_1$	$в_2$

3. Автомат Мура, заданий графічно, представити табличним способом.



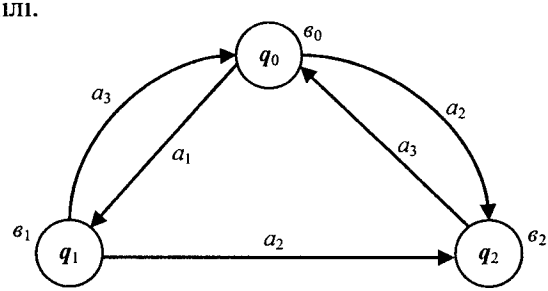
4. *C*-автомат, заданий графічно, представити табличним способом.



5. Автомат Мілі, заданий матрично, представити графічно і таблично.

$$\begin{pmatrix} a_1 / в_1 & a_2 / в_2 & a_3 / в_3 \\ 0 & a_1 / в_0 & a_2 / в_1 \\ a_2 / в_3 & 0 & a_2 / в_1 \\ a_3 / в_0 & 0 & 0 \end{pmatrix}$$

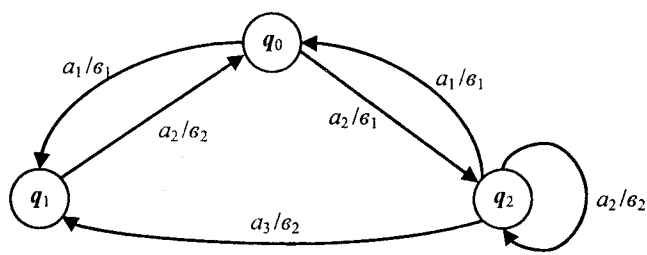
6. Заданий графічно автомат Мура перетворити в еквівалентний автомат Мілі.



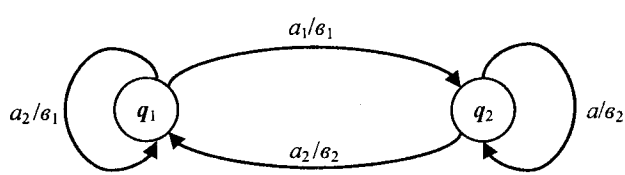
7. Заданий таблично автомат Мура перетворити в еквівалентний табличний автомат Мілі.

q_j \ $в_k$	$в_0$	$в_1$	$в_2$
a_i \ q_j	q_0	q_1	q_2
a_1	q_1	q_0	—
a_2	—	q_2	q_1
a_3	q_3	q_1	q_0

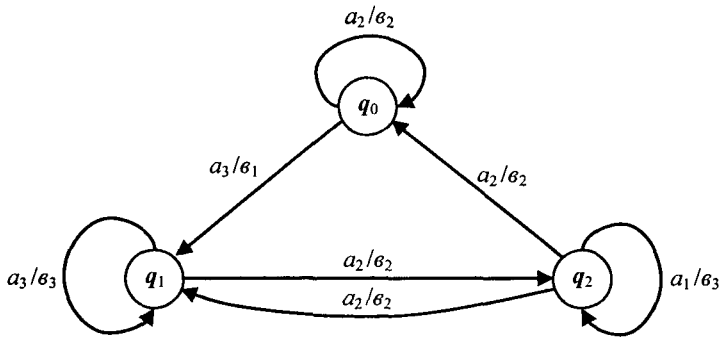
8. Задані графічно автомати Мілі перетворити в еквівалентні графічні автомати Мура:



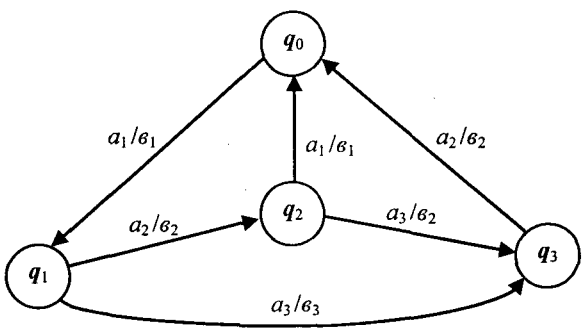
а)



б)



в)



г)

9. Для автоматів заданих таблицями переходів і виходів побудувати автомати, ізоморфні заданим:

a_i \ q_j	q_1	q_2	q_3	q_4	q_5
a_1	q_2	q_1	q_2	q_1	q_2
a_2	q_3	q_5	q_4	q_5	q_3

a_i \ q_j	q_1	q_2	q_3	q_4	q_5
a_1	0	1	1	0	1
a_2	1	0	1	1	0

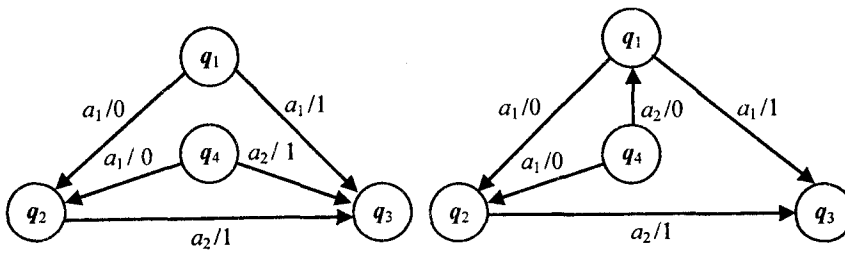
а)

a_i \ q_j	q_1	q_2	q_3	q_4
a_1	q_2	q_1	q_2	q_1
a_2	q_3	q_4	q_4	q_2
a_3	q_1	q_3	q_1	q_4

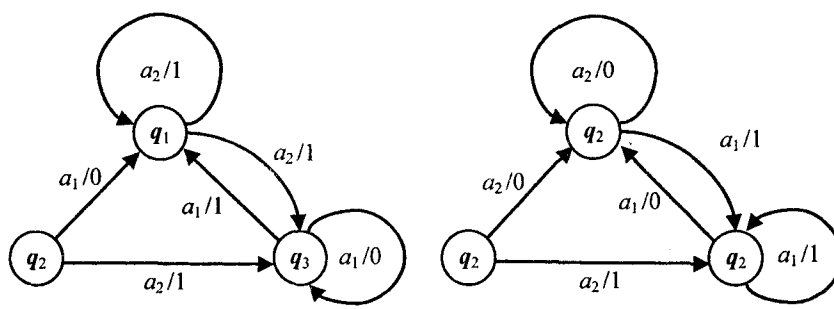
a_i \ q_j	q_1	q_2	q_3	q_4
a_1	1	0	1	0
a_2	0	1	1	1
a_3	1	0	0	0

б)

10. Для автоматів, заданих графічно, довести або спростувати їх еквілентність



а)



б)

11. Для повністю визначених автоматів Мілі, заданих таблицями переходів і виходів, знайти еквівалентні їм мінімальні автомати. Зробити графічне відображення заданого і мінімального автомата. Отримані результати порівняти і зробити висновки.

а)

a_i \ q_j	q_1	q_2	q_3	q_4	q_5	q_6	q_7
a_1	q_1	q_2	q_4	q_7	q_6	q_7	q_1
a_2	q_2	q_1	q_6	q_6	q_4	q_6	q_2
a_3	q_6	q_7	q_7	q_7	q_4	q_7	q_6

a_i \ q_j	q_1	q_2	q_3	q_4	q_5	q_6	q_7
a_1	1	1	0	0	0	1	1
a_2	0	0	1	1	1	0	0
a_3	1	1	0	0	0	1	1

б)

a_i \ q_j	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
a_1	q_7	q_7	q_4	q_5	q_3	q_5	q_3	q_7
a_2	q_4	q_6	q_4	1	q_6	1	q_4	q_2

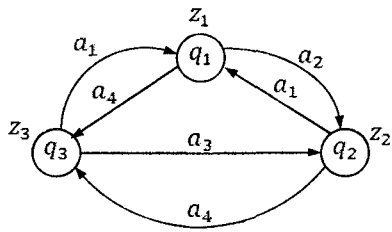
a_i \ q_j	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
a_1	1	1	0	0	1	0	0	1
a_2	0	0	1	1	0	1	1	0

12. Використовуючи графічний метод структурного синтезу автоматів, розробити:

а) схему підсумування за модулем 3 двійкових сигналів «0» і «1»

б) схему підсумування за модулем 2 двійкових сигналів «0» і «1»

в) схему управління комп'ютерним пристроєм заданою у вигляді абстрактного автомата



Коментарі

В цьому розділі для визначення автоматів Мілі, Мура та *С*–автоматів, способів їх задання і перетворення використані підручники [4, 10, 14, 15]. Ізоморфізм та еквівалентність автоматів взяті з [10], мінімізація — з [10, 13], а канонічний і графічний метод структурного синтезу автоматів — із [4, 11, 14, 15].

Розділ 9

ЛОГІКА РЕГУЛЯРНИХ ПОДІЙ

9.1. Основні визначення

Зручною формою задання довільних автоматних відображень є задання їх за допомогою подій.

Нехай $A = \{a_1, a_2, \ldots, a_n\}$ — довільний вхідний алфавіт, а $\sigma(A)$ — множина всіх слів цього алфавіту. Тоді будь-яку підмножину множини $\sigma(A)$ називають **подією в алфавіті A.**

Нехай W — будь-який абстрактний автомат із вхідним алфавітом $A = \{a_1, a_2, \ldots, a_n\}$ і вихідним алфавітом $B = \{в_1, в_2, \ldots, в_m\}$, який індуктує часткове відображення F множини $\sigma(A)$ в множину $\sigma(B)$. Тобто, функція переходів і виходів зазначеного автомата визначена не для всіх пар $(a_i, в_j) \in A \times B$, де $i \in n$, a $j \in m$. Тоді **подією R_k, представленою в автоматі W вихідним сигналом $в_j$,** називають множину всіх слів $p \in \sigma\ (A)$ цього автомата, для яких слово. $F(p)$ визначено і закінчується буквою $в_j$. Якщо $M \subseteq B$ — деяка підмножина вихідних сигналів, то подією, представленою в автоматі W множиною $M \in \{1, \ldots, m\}$, називають об'єднання подій, представлених усіма елементами цієї множини. В такому випадку, коли M збігається з алфавітом B, то відповідну йому подію називають **канонічною множиною подій $R_1, R_2, \ldots, R_m$ автомата W.**

Теорема 9.1.1. Задання **часткового автоматного відображення F** множини $\sigma(A)$ в $\sigma(B)$ довільного абстрактного автомата W з вхідним алфавітом $A = \{a_1, a_2, \ldots, a_n\}$ і вихідним алфавітом $B = \{в_1, в_2, \ldots, в_m\}$ еквівалентно заданню канонічної множини подій $R_1, R_2, \ldots, R_m$ цього автомата.

Для доведення цієї теореми достатньо показати справедливість двох лем.

Лема 9.1.1. Задання часткового автоматного відображення F множини $\sigma(A)$ в $\sigma(B)$ довільного абстрактного автомата W незаперечно визначає канонічна множина подій $R_1, R_2, \ldots, R_m$ автомата W.

Доведення. Нехай F є часткове автоматне відображення $\sigma(A)$ в $\sigma(B)$ і p — задовільне слово із $\sigma(A)$. Якщо образ $F(p)$ є і закінчується буквою $в_j$, то вхідне слово p належить до множини R_j, відміченої вихідним словом $в_j$. Якщо хоча б для однієї вхідної букви слова p не визначена вихідна буква, то слово p є забороненим і його відносять до множини S — ділянки заборони автомата W. Легко побачити, що в результаті перегляду всіх слів $p \in \sigma(A)$ множина $\sigma(A)$ розбивається на $m+1$ підмножин $R_1, R_2, \ldots, R_m$ і S що попарно не перетинаються. Множини $R_1, R_2, ..., R_m$ є подіями, які представлені в автоматі W вихідними сигналами $в_1, в_2, ..., в_m$, і тому утворюють канонічну множину подій автомата W, а множина S — події, що складаються із усіх тих слів $p \in \sigma(A)$, які не увійшли ні до одної із подій R_j, де $j = \{1, 2, \ldots, m\}$, що підтверджує справедливість леми 7.1.1.

Лема 9.1.2. Задання канонічної множини подій $R_1, R_2, \ldots, R_m$ довільного абстрактного автомата W однозначно задає часткове відображення F, яке індуктується автоматом W.

Доведення. Нехай задана канонічна множина подій $R_1, R_2, \ldots, R_m$ абстрактного автомата W. Покажемо, яким чином визначити часткове відображення множини $\sigma(A)$ в $\sigma(B)$ індуктоване автоматом W, не використовуючи відображення F' множини станів Q в собі або, іншими словами, не застосовуючи таблиць переходів і виходів автомата W. Допустимо, що $p = x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ — довільне вхідне слово із $\sigma(A)$. Тоді для кожного x_{i_e} $(1 \le \ell \le k)$ знайдемо вихідну букву $в_{j_e}$ по слідуючому правилу: $в_{j_e}$ є вихідний сигнал, що представляє в автоматі W подію R_{j_e}, яка має початковий відрізок $x_{i_1}, x_{i_2}, \ldots, x_{i_e}$ довжини ℓ вхідного слова p. Якщо для всіх $\ell = 1, 2, 3, \ldots, k$

є відповідне їм $в_{j_e}$, то тоді необхідно вважати, що $F(p) = F(x_{i_1}, x_{i_2}, \ldots, x_{i_k}) = в_{j_1}, в_{j_2}, \ldots, в_{j_k}$. Якщо хоча б для одного ℓ немає $в_{j_e}$, то необхідно сказати, що відображення F на слові p не визначено і $p \in S$, що підтверджує справедливість леми 9.1.2 і доведення теореми 9.1.1.

Таким чином із теореми 9.1.1 можна зробити висновок, що довільне автоматне відображення можна задавати за допомогою розділення множини $\sigma(A)$ всіх слів вхідного алфавіту на кінцеве число подій, що попарно не перетинаються.

9.2. Алгебра подій

Для ефективного опису скінченних і деяких класів нескінченних подій розглянемо алгебру подій.

Означення 9.2.1. **Алгеброю подій в алфавіті** називають множину всіх подій у цьому алфавіті, в якому задана система трьох операцій: двох бінарних, які називають **диз'юнкцією** і **добутком**, і одної унарної, яку називають **ітерацією**.

Означення 9.2.2. **Диз'юнкцією подій *R* і *S*** називають подію P, що позначають як $P = R \vee S$, яку утворюють теоретико-множинним об'єднанням подій R і S.

Означення 9.2.3. **Добутком подій *R* і *S*** називають подію U, позначену як $U = R \cdot S$, яка складається із всіх слів виду $u = r \cdot s$, де $u \in U$, $r \in R$ і $s \in S$. Отже, слова події U утворюються приписом справа будь-якого слова події S до будь-якого слова події R, але не навпаки.

Означення 9.2.4. **Ітерацією події *R*** називають подію, яку позначають як $\{R\}^*$, що є диз'юнкцією порожнього слова ℓ, події R, події $R \cdot R$, події $R \cdot R \cdot R$ і т. д. до безмежності, тобто

$$\{R\}^* = \ell \vee R \cdot R \vee R \cdot R \cdot R \vee \ldots$$

Необхідно зазначити, що подія ℓ, яка утворена порожнім словом ℓ, складається із одного слова нульової довжини і відіграє допоміжну роль у теорії автоматів. Тому в подальшому не будемо вважати

різними події та, які відрізняються одна від одної тільки порожнім словом ℓ. Крім події ℓ, будемо також розглядати порожню подію $\varnothing$, яка складається із порожньої множини букв вхідного алфавіту, і тому є частиною будь-якої події.

Означення 9.2.5. **Регулярним виразом в алфавіті** $A=\{a_1, a_2, \ldots, a_n\}$ називають вираз, який знаходиться рекурсивно таким чином:

1. символи $a_1, a_2, \ldots, a_n, \ell$ і $\varnothing$ є регулярними виразами;

2. якщо R і S є регулярні вирази, то таким є і вирази R і S, $R \cdot S$ і $\{R^*\}$;

3. ніякий інший вираз не є регулярним, якщо він не отриманий шляхом застосування скінченного числа правил 1 і 2.

Таким чином, регулярний вираз — це формула в алгебрі подій, причому одна й та ж подія може бути по-різному виражена через одноелементні події і операції диз'юнкції, добутку й ітерації.

Означення 9.2.6. **Регулярною подією** називають подію, яка має регулярний вираз. Інакше таку подію називають **нерегулярною.**

Приклад 9.2.1. Для алфавіту $A=\{a_1, a_2\}$ і заданих подій $R=\{a_1, a_2 a_1\}$ та $S=\{a_2 a_2, a_1 a_2\}$ в алфавіті A побудувати події $R \vee S$, $R \cdot S$, $\{R\}^*$.

Розв'язання. Користуючись означеннями 9.2.2, 9.2.3 і 9.2.4, отримуємо відповідні події:

$$R \vee S = \{a_1, a_2\, a_1, a_2\, a_2, a_1\, a_2\},$$

$$R \cdot S = \{a_1\, a_2\, a_2,\ a_1\, a_1\, a_2,\ a_2\, a_1\, a_2\, a_2,\ a_2\, a_1\, a_1\, a_2\},$$

$$\{R\}^* = \{\ell,\ a_1,\ a_2\, a_1,\ a_1\, a_1,\ a_1\, a_2\, a_1,\ a_2\, a_1\, a_1,\ a_2\, a_1\, a_2\, a_1,\ a_1\, a_1\, a_1,$$

$$a_1\, a_2\, a_1\, a_1, a_2\, a_1\, a_1\, a_1, a_2\, a_1\, a_2\, a_1\, a_1, a_1\, a_1\, a_2\, a_1, a_1\, a_2\, a_1\, a_2\, a_1,$$

$$a_2\, a_1\, a_1\, a_2\, a_1, a_2\, a_1\, a_2\, a_1\, a_2\, a_1, \ldots\}.$$

Приклад 9.2.2. Для алфавіту $A=\{a_1, a_2, a_3\}$ побудувати чотири регулярні події.

Розв'язання. Використовуючи означення 9.2.5 і 9.2.6, можна отримати такі чотири регулярні події із алфавіту $A=\{a_1, a_2, a_3\}$.

1. $P=\{a_1 \vee a_2 \vee a_3\}^*$. Таку регулярну подію називають універсальною, тому що вона складається із усіх слів алфавіту A.

2. $R=(a_1 \vee a_2 \vee a_3)\cdot(a_1 \vee a_2 \vee a_3)\cdot(a_1 \vee a_2 \vee a_3)$. Така регулярна подія містить тільки три буквені слова алфавіту A.

3. $S=\{a_1 \vee a_2 \vee a_3\}^*\cdot a_1 a_2 a_1 \cdot \{a_1 \vee a_2 \vee a_3\}^*$. Ця регулярна подія складається із усіх слів, в яких хоча б один раз зустрічається відрізок $a_1 a_2 a_1$ алфавіту A.

4. $T=\{a_1 \vee a_2\}^*\cdot(a_2 \vee a_3)\cdot(a_2 \vee a_3)\cdot a_1 \cdot \{a_1\}^*$. Дана регулярна подія складається із таких слів, початок яких є будь-яке слово із букв a_1 або a_2, потім йде двохбуквене слово із a_2 або a_3, а закінчення має хоча б одну букву a_1.

9.3. Закони еквівалентного перетворення регулярних подій

Велике значення в алгебрі подій має встановлення законів еквівалентного перетворення регулярних виразів. При цьому ми обмежимося розглядом основних властивостей операцій $<\vee,\ \cdot,\ \{\}^*>$, які випливають із їх визначень.

1. Закони комутативності

$P \vee R = R \vee P$ — для диз'юнкції,

$P\cdot\{P\}^* = \{P\}^*\cdot P$ — для ітерації.

2. Закони асоціативності

$P \vee (R \vee S) = (P \vee R) \vee S$ — для диз'юнкції,

$P\cdot(R\cdot S) = (P\cdot R)\cdot S$ — для добутку.

3. Закони дистрибутивності

$P\cdot(R \vee S) = (P\cdot R) \vee (P\cdot S)$ — для лівої дистрибутивності добутку відносно диз'юнкції,

$(P \vee R)\cdot S = (P\cdot S) \vee (R\cdot S)$ — для правої дистрибутивності добутку відносно диз'юнкції.

4. Закони ідемпотентності

$P \vee P = P$ — для диз'юнкції,

$\{\{P\}^*\}^* = \{P\}^*$ — для ітерації.

5. Закон розгортання ітерації

$\{P\}^* = \ell \vee P\cdot\{P\}^*$.

6. Диз'юнктивне поглинання ітерації

$\{P\}^* \vee P = \{P\}^*$.

7. Мультиплікативне поглинання ітерації

$\{P\}^* \cdot \{P\}^* = \{P\}^*$.

Для довільних регулярних подій P, R і S із множини $\sigma(A)$ мають місце такі співвідношення

$$P \vee \varnothing = \varnothing \vee P = P,$$

$$P \cdot \varnothing = \varnothing \cdot P = \varnothing,$$

$$P \cdot \ell = \ell \cdot P = P,$$

$$\{\varnothing\}^* = \ell, \{\ell\}^* = \ell.$$

Фігурні дужки із зірочкою $\{\ \}^*$ використовують для позначення ітерації і їх називають ітераційними дужками. Для визначення порядку операцій в алгебрі подій вводяться звичайні дужки. При відсутності звичайних дужок першим виконують ітерацію, потім добуток і в останню чергу диз'юнкцію.

Означення 9.3.1. **Циклічною глибиною регулярного виразу** називають максимальне число вкладених один в одного пар ітераційних дужок.

Означення 9.3.2. **Циклічною глибиною регулярної події** називають мінімальну циклічну глибину регулярних виразів, що представляють його.

Приклад 9.3.1. Для регулярного виразу $\{a_1\{a_2\{a_1\}^* a_1\}^* \vee a_3\}^* \vee a_2$ знайти його циклічну глибину.

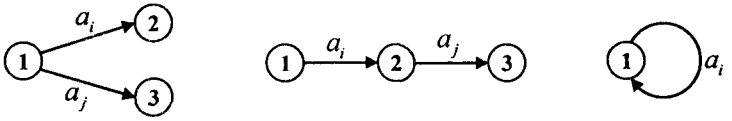
Розв'язання. Із означення 9.3.1 випливає, що циклічна глибина цього регулярного виразу дорівнює 3.

Приклад 9.3.2. Для регулярної події $(\{a_2\}^* \cdot \{a_1 a_3\}^*) \cdot a_2 \vee \vee \{a_1 \cdot a_2\}^*$ знайти її циклічну глибину.

Розв'язання Із означення 9.3.2 випливає, що циклічна глибина даної регулярної події дорівнює 1.

9.4. Задання регулярних подій графами

Означення 9.4.1. Будь-який регулярний вираз (подію) можливо представити у вигляді графа. Зображення елементарних регулярних виразів диз'юнкції $a_i \vee a_j$, добутку $a_i \cdot a_j$, та ітерації $\{a_i\}^*$ у вигляді графів приведені відповідно на рис. 9.4.1.



а) *б)* *в)*

Рис. 9.4.1

Як правило, вершинам графа приписують номери із множини $N = \{1, 2, 3, \ldots, n\}$. У кожному графі регулярного виразу фіксують вершину, яка є початком (звичайно, це номер один) і вершини (одна або декілька), які служать кінцем. Однак у графах регулярних виразів можливі випадки утворення хибних шляхів, які відповідають вхідним словам, що не належать початковим регулярним виразам. Тому, щоб для унеможливити це, розглянемо твердження про повноту системи зв'язків, які ліквідують хибні шляхи в графах регулярних виразів за допомогою введення порожніх стрілок.

Така система правил, яка визначає типи регулярних виразів, графи яких повинні мати порожні стрілки, має такий зміст.

Правило 1. Порожні стрілки на графі регулярного виразу S вводяться у випадку добутку двох або більше ітерацій

$$S = \prod_{i \in N} \{R_i\}^*,$$

де $N = \{1, 2, \ldots, n\}$, а R_i — довільний регулярний вираз.

Графічна інтерпретація правила для $N = \{1, 2, 3\}$ приведена на рис. 9.4.1.

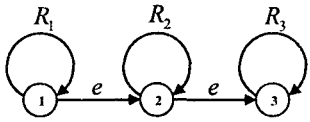


Рис. 9.4.1

Правило 2. Порожні стрілки на графі регулярного виразу S, які розпочинаються і закінчуються ітераційними дужками, вводять в таких випадках:

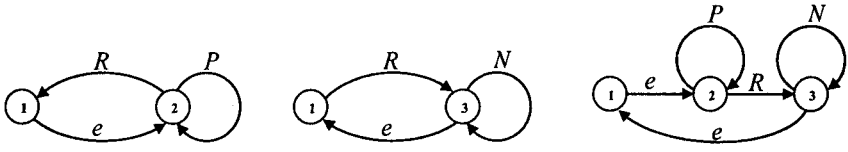
а) $S = \{\{P\}^* \cdot R\}^*$,

б) $S = \{R \cdot \{N\}^*\}^*$,

в) $S = \{\{P\}^* \cdot R \cdot \{N\}^*\}^*$,

де P, R, N — будь-які регулярні вирази.

Графічна інтерпретація цього правила приведена на рис. 9.4.2.



а) *б)* *в)*

Рис. 9.4.2

Правило 3. Порожні стрілки на графі регулярного виразу S вводять у випадку диз'юнкції, якщо хоча б один із диз'юнктивних членів розпочинається з ітерації

$$S = \{R\}^* \cdot Q \vee \{P\}^* \vee \ldots \vee Q,$$

де Q — регулярний вираз, який не має ітераційних дужок.

Графічна інтерпретація правила 3 приведена на рис. 9.4.3.

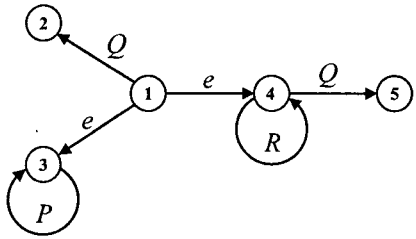


Рис. 9.4.3

Правило 4. Порожні стрілки на графі регулярного виразу S вводять при добутку зліва на диз'юнкцію, якщо хоча б один із диз'юнктивних членів закінчується ітерацією

$$S = (Q \cdot \{P\}^* \vee \{R\}^* \vee \ldots \vee Q) \cdot N.$$

Графічна інтерпретація правила 4 приведена на рис. 9.4.4.

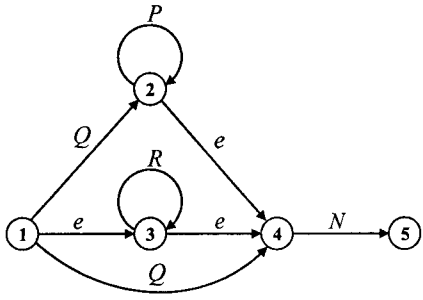


Рис. 9.4.4

Твердження 9.4.1. Система правил 1...4 введення порожніх стрілок для унеможливлення хибних шляхів у графах регулярних виразів є повною.

Доведення. Будь-який регулярний вираз в алгебрі подій утворюється в результаті застосування скінченного числа операцій диз'юнкції, добутку й ітерації. Єдина операція ітерації як при однократному, так і багатократному застосуванні не веде до появи порожніх стрілок у графі регулярного виразу. Такі стрілки в графі появляються лише тоді, коли в регулярному виразі є поєднання таких операцій: $(\cdot, \{\}^*)$, $(\vee, \{\}^*)$, $(\cdot, \vee, \{\}^*)$.

Розглянемо регулярні вирази, отримані застосуванням усіх трьох операцій. Позначимо регулярні вирази у базисі $(\cdot, \vee)$ через P, у базисі $(\cdot, \{\}^*)$ — через Q, у базисі $(\vee, \{\}^*)$ — через R, а у базисі $(\cdot, \vee, \{\}^*)$ — через V.

Із базисів випливає, що ітерація над P не потребує використання порожньої стрілки. Застосування ітерації до R, Q, V приводить до появи таких стрілок. Ці випадки описуються правилом 2.

Розглядаючи попарно диз'юнкції всіх регулярних виразів, можна побачити, що $P \vee P$ не потребує використання порожніх стрілок. Всі інші випадки описуються правилом 3.

Розберемо тепер усі можливі попарні добутки із P, R, Q і V. Зрозуміло, що $P \cdot P$, $P \cdot R$, $P \cdot Q$, $P \cdot V$ і $Q \cdot P$ не потребують використання порожніх стрілок. Введення їх у графах, які реалізують добутки $Q \cdot Q$, $Q \cdot R$ і $Q \cdot V$, описуються правилом 1. Добутки $R \cdot P$, $V \cdot P$ описуються правилом 4, а $R \cdot Q$, $R \cdot R$, $R \cdot V$, $V \cdot Q$, $V \cdot R$ і $V \cdot V$ — правилами 1 і 4.

Таким чином, розглянуті всі можливі види регулярних виразів та показано, що поєднання операції $< \cdot , \vee , \{ \}^* >$, які приводять до появи порожніх стрілок у графах регулярних виразів, повністю описуються правилами 1...4, що і потрібно було довести.

Приклад 9.4.1. Виконати графічне зображення події, яка задана регулярним виразом

$$R = \{ a_1 \, a_2 \}^* \cdot \{ a_3 \}^* \cdot \{ a_1 \vee a_3 \}^*.$$

Розв'язання. Використовуючи означення 9.4.1, а також правило 1 до заданого регулярного виразу, будуємо графічне зображення події, рис. 9.4.5.

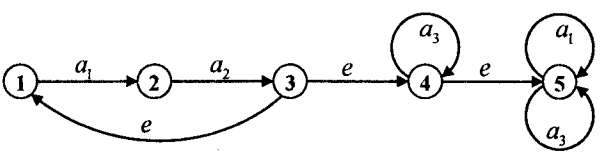


Рис. 9.4.5

Приклад 9.4.2. Виконати графічне зображення події, яка задана регулярним виразом

$$R = \{ a_2 \vee a_1 \}^* \cdot (a_3 \{ a_3 \}^* \cdot \{ a_2 \}^* \cdot a_1 \vee a_3 \{ a_2 \}^* \cdot a_1 \vee a_1 \{ a_2 \}^* a_3).$$

Розв'язання. Використовуючи означення 9.4.1, а також правило 2 до заданого регулярного виразу будуємо графічне зображення події, рис. 9.4.6.

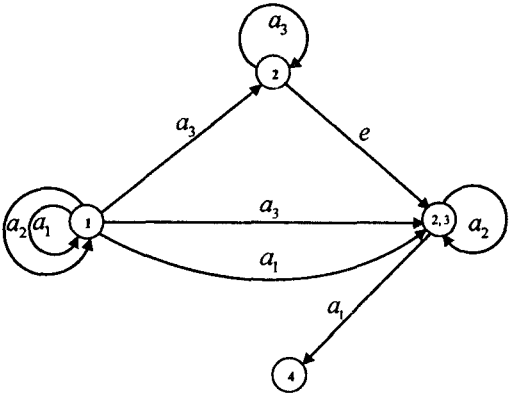


Рис. 9.4.6

Приклад 9.4.3. Виконати графічне зображення події, яка задана регулярним виразом

$$R = a_1 a_3 \cdot \{ a_2 a_1 \}^* \vee a_4 \{ a_1 \}^* a_2 \vee \{ a_3 \}^*.$$

Розв'язання. Використовуючи означення 9.4.1, а також правило 3 до заданого регулярного виразу, будуємо графічне зображення події, рис. 9.4.7.

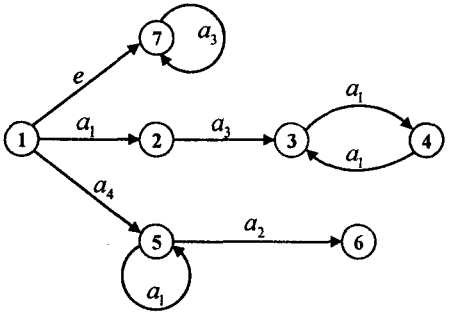


Рис. 9.4.7

Приклад 9.4.4. Виконати графічне зображення події, яка задана регулярним виразом

$$R = (a_1 a_2 \cdot \{ a_3 \}^* \vee \{ a_2 \}^* \vee a_1 \vee a_2) \cdot a_3.$$

Розв'язання. Використовуючи означення 9.4.1, а також правило 4 до заданого регулярного виразу, будуємо графічне зображення події, рис. 9.4.8.

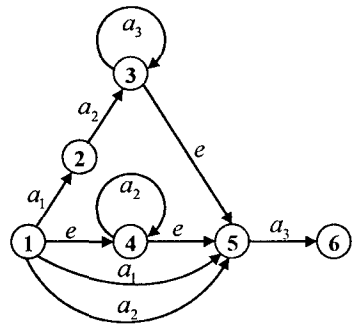


Рис. 9.4.8

9.5. Синтез автоматів за графами регулярних подій

З точки зору практичного застосування найбільше значення мають алгоритми абстрактного синтезу автоматів із застосуванням таблиць переходів за графами регулярних виразів. Кроки подібного алгоритму мають такий зміст.

1. Перед побудовою графа перетворюють кожний регулярний вираз шляхом виносу спільних множників вправо за дужку об'єднують їх знаком диз'юнкції.

2. Використовуючи правила 1...4 (§ 9.4), будують граф за регулярним виразом, отриманим на кроці один алгоритму.

3. Приписують вершинам графа індекси із множини $N=\{1, 2, 3, \ldots, n\}$.

4. Будують таблицю переходів автомата. Стрічки таблиці відповідають різним буквам a_i вхідного алфавіту, який є в регулярному виразі R. Перший стовпчик таблиці позначають символом «1» початкової вершини графа і із неї розпочинають побудову таблиці переходів автомата. До клітинки таблиці, яка розміщена на перетині q_j стовпчика і a_i стрічки, записують диз'юнкцію індексів тих вершин графа, в які входять стрілки з буквами a_i, що виходять із будь-якої вершини, індекси якої включені в множину індексів

станів q_j. На початку побудови таблиці в стан q_j входить один індекс початкової вершини. Якщо таких стрілок нема, то в клітинку записують «–», що означає порожній стан автомата. Після заповнення клітинок стовпчика таблиці склад кожної клітинки, якщо він повністю не збігається зі складом станів, що відмічають стовпчики, виписують як новий відмічений стан. Процес побудови таблиці переходів уважають закінченим, якщо весь вміст кожної клітинки таблиці виписаний як відмічений стан. Якщо вершина графа має декілька індексів, то в клітинку таблиці переходів записують тільки крайній справа індекс.

5. Стан q_j відмічають вихідним сигналом y_i, який відповідає даній кінцевій вершині графа. Якщо в множину індексів входять індекси декількох кінцевих вершин, то такий стан відмічають диз'юнкцією вихідних сигналів.

6. Для отримання відміченої таблиці переходів автомата виконують перекодування станів і вихідних букв.

Розглянемо роботу алгоритму на прикладах.

Приклад 9.5.1. Для події, заданої регулярним виразом

$$R = \{a_1 \vee a_2\}^* (a_1 \{a_1\}^* \cdot \{a_3\}^* a_2 \vee a_1 \{a_3\}^* a_2 \vee a_3 \{a_3\}^* a_2)$$

побудувати автомат Мура, представляющий цю подію вихідною буквою y.

Розв'язання. Використовуючи перший крок алгоритму, виносимо спільні множники за дужки, в результаті чого отримаємо регулярний вираз у такому вигляді:

$$R = \{a_1 \vee a_2\}^* \cdot (a_1 \{a_1\}^* \vee a_1 \vee a_3) \cdot \{a_3\}^* a_2.$$

Користуючись другим кроком алгоритму, будуємо граф регулярного виразу R, який приведений на рис. 9.5.1.

Згідно з третім кроком алгоритму, приписуємо номери вершинам графа. Початкову вершину графа позначимо першим номером, а кінцеву — четвертим. Користуючись четвертим кроком алгоритму, будуємо відмічену таблицю переходів, табл. 9.5.1.

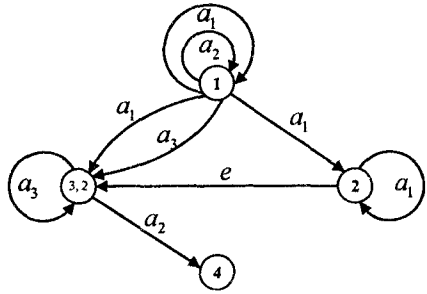


Рис. 9.5.1

Таблиця 9.5.1

y_k	—	—	—	y	y
a_i \ q_j	1	$1 \vee 2 \vee 3$	3	$1 \vee 4$	4
a_1	$1 \vee 2 \vee 3$	$1 \vee 2 \vee 3$	—	$1 \vee 2 \vee 3$	—
a_2	1	$1 \vee 4$	4	1	—
a_3	3	3	3	3	—

Користуючись п'ятим кроком алгоритму, робимо перекодування станів: $1 \to q_1$; $1 \vee 2 \vee 3 \to q_2$; $3 \to q_3$; $1 \vee 4 \to q_4$; $4 \to q_5$ і отримаємо відмічену таблицю переходів автомата Мура, табл. 9.5.2.

Таблиця 9.5.2

y_k	—	—	—	y	y
a_i \ q_j	q_1	q_2	q_3	q_4	q_5
a_1	q_2	q_2	—	q_2	—
a_2	q_1	q_4	q_5	q_1	—
a_3	q_3	q_3	q_3	q_3	—

Граф автомата Мура, побудований на відміченій таблиці переходів, буде мати вигляд, приведений на рис. 9.5.2.

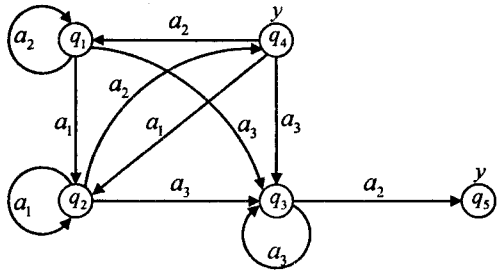


Рис. 9.5.2

Приклад 9.5.2. Для події, заданої регулярним виразом

$$R = \{a_1\}^* \cdot (a_3 \cdot \{a_1 \vee a_2\}^* \vee (a_1 \vee a_2)) \cdot \{a_3\}^*,$$

побудувати автомат Мура, що представляє цю подію вихідними буквами y_1, y_2, y_3.

Розв'язання. Так як спільних множників у регулярному виразу немає, то, згідно із розглядуваним алгоритмом, переходимо до другого кроку, на якому будуємо граф регулярного виразу R, який приведений на рис. 9.5.3.

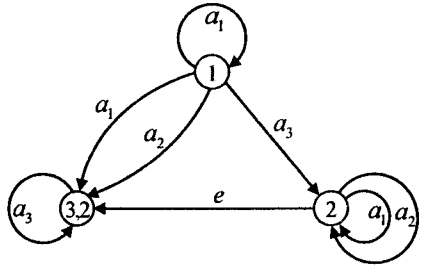


Рис. 9.5.3

Користуючись третім кроком алгоритму, приписуємо номери вершинам графа. Початкову вершину графа позначаємо першим номером, а кінцеву — третім.

Згідно із четвертим кроком алгоритму за графом, рис. 9.5.3, будуємо відмічену таблицю переходів, табл. 9.5.3.

Таблиця 9.5.3

y_k	—	—	y_1	y_2	y_3
a_i \ q_j	1	$1 \vee 3$	3	$2 \vee 3$	2
a_1	$1 \vee 3$	$1 \vee 3$	—	2	2
a_2	3	3	—	2	2
a_3	$2 \vee 3$	$2 \vee 3$	3	3	3

Використовуючи п'ятий крок алгоритму, виконуємо перекодування станів: $1 \to q_1$; $1 \vee 3 \to q_2$; $3 \to q_3$; $2 \vee 3 \to q_4$; $2 \to q_5$ і отримаємо відмічену таблицю переходів автомата Мура, табл. 9.5.4.

Таблиця 9.5.4

y_k	—	—	y_1	y_2	y_3
a_i \ q_j	q_1	q_2	q_3	q_4	q_5
a_1	q_2	q_2	—	q_5	q_5
a_2	q_3	q_3	—	q_5	q_5
a_3	q_4	q_4	q_3	q_3	q_3

Граф автомата Мура, побудований на відміченій таблиці переходів, буде мати вигляд, приведений на рис. 9.5.4.

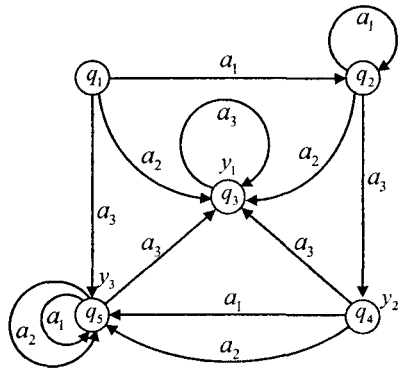


Рис. 9.5.4

Приклад 9.5.3. Необхідно синтезувати автомат, що моделює вироблення умовного рефлексу із забуванням. Вхідний алфавіт автомата позначимо через $A = \{ a_1, a_2, a_3 \}$, де a_1 — наявність умовного подразника та відсутність безумовного, a_2 — наявність безумовного подразника і відсутність умовного, a_3 — одночасна подача умовного та безумовного подразника. Вихідний алфавіт автомата складається із двох букв $Y = \{ y_1, y_2 \}$.

Автомат виробляє умовний рефлекс після певного стану навчання і реагує вихідною буквою y_2 на дію умовного подразника a_1 так само, як і на дію безумовного a_2 або спільну дію обох подразників a_3. В решті випадків на виході автомата з'являється буква y_1. Етап навчання полягає в одночасній дії умовного і безумовного подразників. Таких збігів за час навчання автомата повинно бути не менше n.

Якщо в процесі навчання між двома послідовностями збігу подразників відбулося більше k збігів, то процес навчання порушується і його необхідно розпочинати заново. Якщо після вироблення автоматом умовного рефлексу відбудеться більше чим m послідовних збігів дій умовного та безумовного подразника, то рефлекс забувається.

Позначимо, що умовний рефлекс із забуванням виробляється при таких параметрах: $k \le 1$, $n \ge 2$, $m > 2$.

Переходячи від описової форми алгоритму роботи автомата до задання алгоритму на мові регулярних виразів, отримаємо

$$R = \{ a_1 \vee a_2 \vee a_3 \}^* [(a_2 \vee a_3) \vee a_3 (a_1 \vee a_2)\, a_3 \vee a_3) \cdot \{ a_3 \}^* \{ (a_1 \vee a_2) \cdot a_3 \}^* \cdot ((a_1 \vee a_2) \vee (a_1 \vee a_2) \cdot (a_1 \vee a_2))] .$$

Розв'язання. Використовуючи другий крок алгоритму будуємо граф регулярного виразу R, рис. 9.5.5.

Згідно із третім кроком алгоритму, приписуємо номери вершинам графа. Початкову вершину графа позначимо першим номером, а кінцеву — восьмим.

За графом, рис. 9.5.5, будуємо відмічену таблицю переходів, табл. 9.5.5. і табл. 9.5.6.

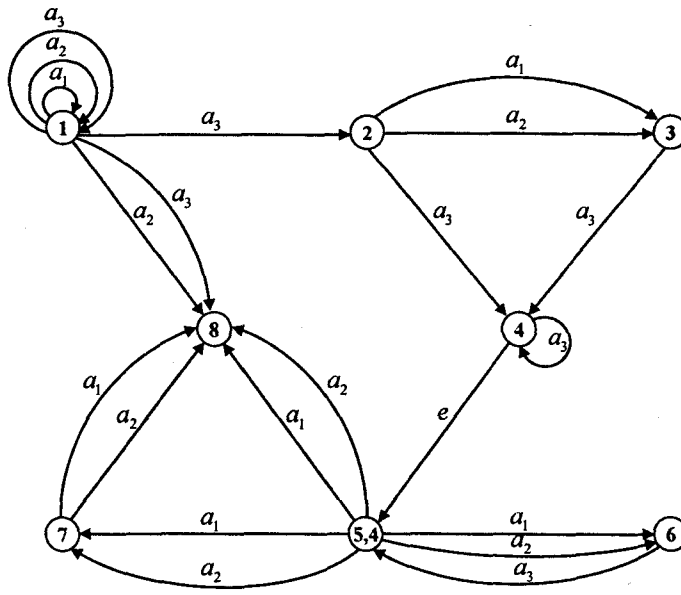


Рис. 9.5.5

Таблиця 9.5.5

y_i	y_1	y_2	y_2	y_1
a_i \ q_j	1	1 ∨ 8	1 ∨ 2 ∨ 8	1 ∨ 3
a_1	1	1	1 ∨ 3	1
a_2	1 ∨ 8	1 ∨ 8	1 ∨ 3 ∨ 8	1 ∨ 8
a_3	1 ∨ 2 ∨ 8	1 ∨ 2 ∨ 8	1 ∨ 2 ∨ 4 ∨ 8	1 ∨ 2 ∨ 4 ∨ 8

Таблиця 9.5.6

y_i	y_1	y_2	y_2	y_2
a_i \ q_j	1 ∨ 3 ∨ 8	1 ∨ 2 ∨ 4 ∨ 8	1 ∨ 3 ∨ 6 ∨ 7 ∨ 8	1 ∨ 2 ∨ 4 ∨ 5 ∨ 8
a_1	1	1 ∨ 3 ∨ 6 ∨ 7 ∨ 8	1 ∨ 8	1 ∨ 3 ∨ 6 ∨ 7 ∨ 8
a_2	1 ∨ 8	1 ∨ 3 ∨ 6 ∨ 7 ∨ 8	1 ∨ 8	1 ∨ 3 ∨ 6 ∨ 7 ∨ 8
a_3	1 ∨ 2 ∨ 4 ∨ 8	1 ∨ 2 ∨ 4 ∨ 8	1 ∨ 2 ∨ 4 ∨ 5 ∨ 8	1 ∨ 2 ∨ 4 ∨ 8

Використовуючи п'ятий крок алгоритму, виконуємо перекодування станів: $1 \to q_1$; $1 \vee 8 \to q_2$; $1 \vee 2 \vee 8 \to q_3$; $1 \vee 3 \to q_4$; $1 \vee 3 \vee 8 \to q_5$; $1 \vee 2 \vee 4 \vee 8 \to q_6$; $1 \vee 3 \vee 6 \vee 7 \vee 8 \to q_7$; $1 \vee 2 \vee 4 \vee 5 \vee 8 \to q_8$; і отримаємо відмічену таблицю переходів автомата Мура, табл. 9.5.7.

Таблиця 9.5.7

y_i	y_1	y_2	y_2	y_1	y_2	y_2	y_2	y_2
a_i \ q_j	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
a_1	q_1	q_1	q_4	q_1	q_1	q_7	q_2	q_7
a_2	q_2	q_2	q_5	q_2	q_2	q_7	q_2	q_7
a_3	q_3	q_3	q_6	q_6	q_6	q_6	q_8	q_6

Граф автомата Мура, побудований за відміченою таблицею переходів, буде мати вигляд, приведений на рис. 9.5.6.

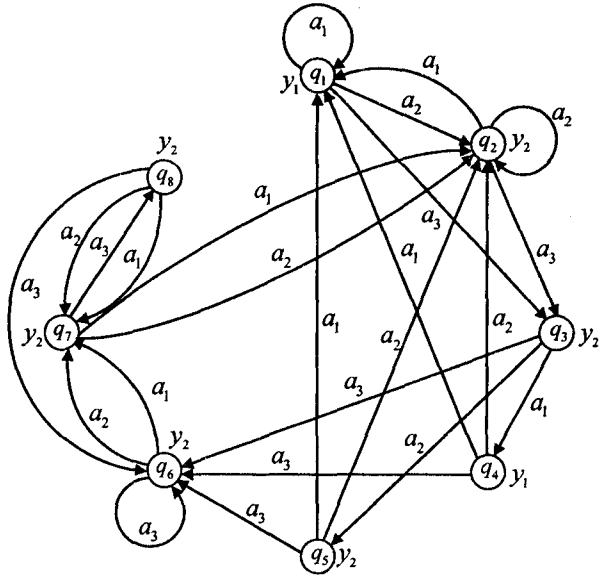


Рис. 9.5.6

Контрольні запитання

1. Що називають подією в алфавіті A?
2. Що називають відображенням в автоматі?
3. Яке відображення називають частковим?
4. Яку подію називають канонічною множиною подій в автоматі?
5. Чому еквівалентно задання канонічної множини подій в автоматі?
6. За допомогою чого можна задавати довільне автоматичне відображення?
7. Що називають алгеброю подій в алфавіті A?
8. Що називають диз'юнкцією подій T і Q?
9. Що називають добутком подій T і Q?
10. Що називають ітерацією події T?
11. Що таке подія ℓ і яким словом вона утворюється?
12. Що називають регулярним виразом в алфавіті A?
13. Що таке регулярна і нерегулярна подія, як їх розпізнати?
14. Що необхідно розуміти під еквівалентним перетворенням регулярних подій?
15. Сформулюйте закони комутативності для диз'юнкції й ітерації.
16. Сформулюйте закони асоціативності для диз'юнкції і добутку.
17. Сформулюйте закони ідемпотентності для диз'юнкції та ітерації.
18. Сформулюйте закони для лівої і правої дистрибутивності добутку відносно диз'юнкції.
19. В чому суть закону розгортання ітерації?
20. В чому суть закону диз'юнктивного і мультиплікативного поглинання ітерації?
21. Який порядок операцій діє в алгебрі подій?
22. Що називають циклічною глибиною регулярного виразу?
23. Що називають циклічною глибиною регулярної події?
24. Які регулярні події можуть бути зображені графом ?
25. Яка вершина графа регулярної події може бути початковою, а яка кінцевою?
26. Які правила застосовують для унеможливлення хибних шляхів у графах, які відображають регулярні події ?
27. Які типи стрілок застосовують у графах регулярних подій, використовуючи повну систему зв'язків ?
28. Назвіть кроки алгоритму синтезу автоматів за графами регулярних подій.
29. Яка різниця між таблицею переходів і відміченою таблицею переходів автомата Мура?
30. Що необхідно зробити в регулярному виразі перед побудовою автомата Мура?

31. Скільки станів графа автомата Мура можна відмічати вихідними буквами (сигналами)?
32. Для чого потрібно робити перекодування станів графа регулярних подій?

Задачі для самостійного розв'язування

1. Для алфавіту $A=\{a_1, a_2\}$ і заданих подій $R=\{a_1 a_1, a_2\}$, $S=\{a_1, a_2 a_2\}$ в ньому побудувати події $\{R\}^*$, $R\cdot S$ і $R\vee S$.

2. Для алфавіту $A=\{a_1, a_2, a_3\}$ і заданих подій $R=\{a_1, a_3\}$ і $S=\{a_1 a_2, a_2 a_3\}$ в ньому побудувати події $R\vee S$, $R\cdot S$ і $\{R\}^*$.

3. Для алфавіту $A=\{a_1, a_2\}$ побудувати чотири регулярні події в ньому.

4. Для алфавіту $A=\{a_1, a_2, a_3\}$ побудувати сім регулярних подій у ньому.

5. Для регулярних виразів:

а) $\{a_1\vee\{a_1\}^*\cdot\{a_1\{a_2\}^* a_3\}^*\cdot a_2$;

б) $\{a_1\}^* a_2\{a_2\}^* a_3\{a_1\}^*\}^*$;

в) $\{\{a_1\vee a_2\}^* a_1\{a_1\vee a_2\}^*\}^*$

знайти їх циклічну глибину.

6. Для регулярної події $(\{a_1\}^*\{a_2 a_3\}^*)\cdot\{a_1\}^*\vee\{a_2\}^*\vee\{a_3\}^*$ знайти її циклічну глибину.

7. Виконати графічне зображення подій, які задані такими регулярними виразами:

а) $R=\{a_1\vee a_2\vee a_3\}^*\cdot\{a_2\}^*\vee a_1 a_2\cdot\{a_2\}^*$;

б) $R=\{(a_1\vee a_2)\cdot a_3\}^*\cdot a_1 a_2\vee a_3\vee\{a_2\}^*$;

в) $R=\{a_1\vee a_2\}^*\cdot(a_3\cdot\{a_4 a_5\}^*\vee(a_1\vee a_2))\cdot\{a_3\}^*$;

г) $R=\{a_1\}^*\cdot a_2\cdot\{a_3\}^*\vee a_2 a_1\cdot\{a_3\vee a_4\}^*\cdot a_3$;

д) $R=\{a_1\vee a_2\vee a_3\}^*\cdot a_3\cdot\{a_1\vee a_2\}^*\vee a_3\cdot\{a_3 a_4\}^*$;

є) $R=\{a_1 a_2 a_3\}^*\cdot a_1\vee a_3\cdot\{a_2\cdot a_1\}^*\vee(a_1\vee a_3)\cdot\{a_3\}^*$.

8. Для подій, заданих регулярними виразами:

а) $R = a_1 \cdot \{a_2 a_3 a_1\}^* \vee \{a_1 a_2 a_3\}^* \vee a_1 a_2 \{a_3\}^*$;

б) $R = a_2 a_3 \{a_2 a_1\}^* \vee a_3 \{a_1\}^* \vee \{a_3\}^*$;

в) $R = a_1 a_2 \cdot \{a_3 \vee a_2\}^* \vee a_3 \{a_1 \vee a_2 \vee a_3\}^* a_2 \vee (a_1 \vee a_2) \cdot \{a_2 a_3\}^*$;

г) $R = \{a_1 a_2 a_3\}^* \cdot a_1 \vee a_3 \{a_2 a_1\}^* \vee (a_1 \vee a_3) \cdot \{a_3\}^*$;

д) $R = \{a_1 \vee a_2 \vee a_3\}^* \cdot a_3 \cdot \{a_1 \vee a_2\}^* \vee a_3 \cdot \{a_3 a_4\}$,

побудувати автомати Мура з вихідними буквами: y_1, y_2 для варіанта *а)*; y_1, y_2, y_3 для варіанта *б)*; y_1, y_2, y_3, y_4 для варіанта *в)*; y_1, y_2, y_3 для варіанта *г)* і $y_1 y_2 y_3 y_4$ для варіанта *д)*.

Коментарі

В заданому розділі основні визначення і алгебра подій узяті з [20], закони еквівалентного перетворення регулярних подій з [2, 20], а синтез автоматів за графами регулярних подій випливає [9, 11].

Розділ 10

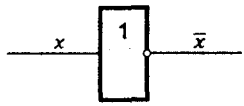
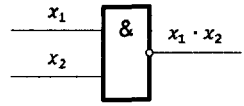
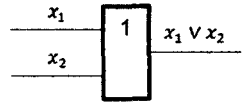
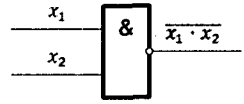
ЛОГІКА ПОБУДОВИ КОМБІНАЦІЙНИХ СХЕМ

10.1. Логічні елементи елементарних булевих функцій

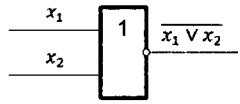
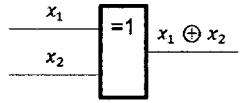
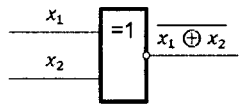
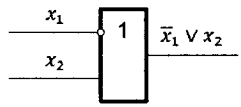
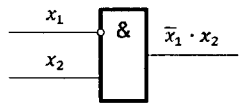
Булеві функції від одного або двох аргументів називають **елементарними**. Елементарні булеві функції, їх означення, позначення і властивості приведені в § 2.1, § 2.3 і § 2.6. Схему, яка реалізує елементарну логічну операцію, називають **елементом**.

Найменування й умовні позначення основних логічних елементів, згідно із нормативно-технічною документацією, наведені в табл. 10.1.1

Таблиця 10.1.1

Найменування операції	Найменування елемента	Умовне графічне позначення
1	*2*	*3*
Заперечення	«*НІ*»	
Кон'юнкція	«*І*»	
Диз'юнкція	«*АБО*»	
Заперечення кон'юнкції (функція Шеффера)	«*І-НІ*»	

Закінчення табл. 10.1.1

1	*2*	*3*
Заперечення диз'юнкції (стрілка Пірса)	«*АБО-НІ*»	
Нерівнозначність	«*Виключаюче АБО*»	
Еквівалентність	«*Еквівалентність*»	
Імплікація	«*Якщо, то*»	
Заборона	«*Заборона*»	

Наведені в табл. 10.1.1 логічні елементи є базовими при побудові різноманітних схем комп'ютерної логіки. Для побудови логічних елементів, представлених у табл. 10.1.1, використовують різноманітні системи елементів, розгляд яких виходить за межі даного навчального посібника.

Логічні елементи наведені в табл. 10.1.1 працюють наступним чином. Якщо на вхід елемента «*НІ*» подати логічний сигнал $x = 1$, то на його виході буде логічний сигнал «0» і навпаки. При подачі на логічний елемент «*І*» сигналів $x_1 = 1$ і $x_2 = 1$ на його виході буде сигнал $x_1 \cdot x_2 = 1$, а у всіх інших випадках – сигнал «0». Виконання операції диз'юнкції на елементі «*АБО*» відбувається так. При подачі сигналів $x_1 = 1$, $x_2 = 0$, або $x_1 = 0$, $x_2 = 1$, або $x_1 = 1$, $x_2 = 1$ на його виході буде сигнал $x_1 \vee x_2 = 1$, тобто на виході елемента «*АБО*» в єдиному випадку буде сигнал «0», якщо на його входах x_1 і x_2 буде одночасно присутнім сигнал «0».

При подачі на вхід елемента «*І-НІ*» сигналів $x_1 = 1$ і $x_2 = 1$ на його виході буде сигнал $\overline{x_1 \cdot x_2} = 0$, а у всіх інших випадках — сигнал «1». Елемент «*І-НІ*» реалізує функцію Шеффера. Він широко застосовується при побудові інтегральних мікросхем.

Елемент «*АБО-НІ*» працює наступним чином. При подачі на нього хоча б одного сигнала «1» на його виході буде сигнал $\overline{x_1 \vee x_2} = 0$. Сигнал «1» на виході елемента «*АБО-НІ*» буде в єдиному випадку, коли на його входах x_1 і x_2 одночасно присутній сигнал «0». Даний елемент реалізує функцію «*стрілка Пірса*».

Елемент «*Виключаюче АБО*» виконує функцію нерівнозначності і працює наступним чином. На виході даного елемента виникає сигнал «1» тільки тоді, коли хоча б на одному вході x_1 або x_2 є сигнал «1». У всіх інших випадках на виході елемента «*Виключаюче АБО*» присутній сигнал «0».

Елемент «*Еквівалентність*» є елементом заперечення «*Виключаючого АБО*» і на його виході з'являється сигнал «1» тоді, коли на його входах x_1 та x_2 одночасно присутні сигнали «0» або «1». У всіх інших випадках на його виході присутній сигнал «0».

Елемент «*Якщо, то*» реалізує логічну функцію імплікації і працює наступним чином. На виході даного елемента сигнал буде відсутній тільки в єдиному випадку, коли на його вході x_1 отримують сигнал «1», а на вході x_2 — сигнал «0». У всіх інших випадках елемент «*Якщо, то*» видає сигнал «1».

Елемент «*Заборона*» працює наступним чином. Якщо на вході даного елемента x_1 присутній сигнал «1», то незалежно від значення сигналу на вході x_2, на його виході завжди буде присутній сигнал «0». Тобто, вхід x_1 елемента «*Заборона*» є дозволяючим або забороняючим залежно від значення сигналу на даному вході.

Розглянуті логічні елементи широко застосовуються для побудови логічних комбінаційних схем у комп'ютерах, таких, як дешифратори, шифратори, мультиплексори, демультиплексори, суматори та компаратори.

10.2. Логіка побудови дешифраторів та шифраторів

Означення 10.2.1. **Дешифратором** називають функціональний пристрій комп'ютера, призначений для перетворення кожної ком-

бінації вхідного двійкового коду в управляючий сигнал тільки на одному із своїх виходів.

Функціонування повного дешифратора описується такою системою логічних функцій

$$\begin{aligned} f_0 &= \overline{x}_n \cdot \overline{x}_{n-1} \cdot \ldots \cdot \overline{x}_2 \cdot \overline{x}_1, \\ f_1 &= \overline{x}_n \cdot \overline{x}_{n-1} \cdot \ldots \cdot \overline{x}_2 \cdot x_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f_{m-1} &= x_n \cdot x_{n-1} \cdot \ldots \cdot x_2 \cdot x_1, \end{aligned} \qquad (10.2.1)$$

$x_1, x_2, \ldots, x_n$ — вхідні двійкові змінні дешифратора;

$f_0, f_1, \ldots, f_{m-1}$ — вихідні логічні функції дешифратора;

Вихід дешифратора, на якому з'являється управляючий сигнал, називають активним. Якщо значення сигналу на активному виході дорівнює «1», то на всіх інших пасивних виходах дешифратора сигнали дорівнюють «0».

Активний вихід дешифратора в інтегральному виконанні часто відображається значенням логічного «0», а на решті пасивних виходах він дорівнює «1». Тоді функціонування повного дешифратора з інверсними виходами описується такою системою логічних функцій:

$$\begin{aligned} \varphi_0 &= \overline{x}_n \vee \overline{x}_{n-1} \vee \ldots \vee \overline{x}_2 \vee \overline{x}_1, \\ \varphi_1 &= \overline{x}_n \vee \overline{x}_{n-1} \vee \ldots \vee \overline{x}_2 \vee x_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \varphi_{m-1} &= x_n \vee x_{n-1} \vee \ldots \vee x_2 \vee x_1, \end{aligned} \qquad (10.2.2)$$

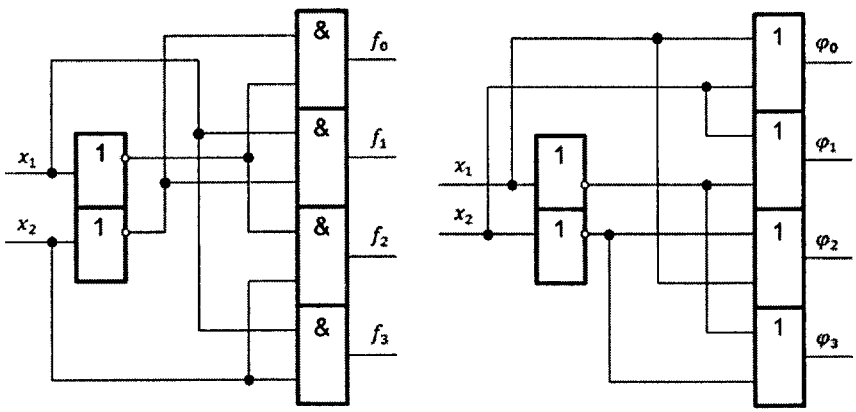
де $f_0, f_1, \ldots, f_{m-1}$ — вихідні логічні функції дешифратора.

Для побудови дешифратора використовують такі кроки. На першому кроці із системи логічних функцій 10.2.1 або 10.2.2 обирають функції, які необхідно використати для проектування заданого дешифратора. На другому кроці за вибраними рівняннями в необхідній елементній базі будують потрібний дешифратор. Так, наприклад, необхідно побудувати дешифратор із чотирма активними вихідними сигналами «1» і дешифратор з чотирма активними вихідним сигналами «0». Для цього на першому кроці їх побудови складають рівняння їх роботи, які мають такий вигляд відповідно

$$f_0 = \overline{x}_2 \cdot \overline{x}_1;\ f_1 = \overline{x}_2 \cdot x_1;\ f_2 = x_2 \cdot \overline{x}_1;\ f_3 = x_2 \cdot x_1. \qquad (10.2.3)$$

$$\varphi_0 = \overline{x}_2 \vee \overline{x}_1;\ \varphi_1 = \overline{x}_2 \vee x_1;\ \varphi_2 = x_2 \vee \overline{x}_1;\ \varphi_3 = x_2 \vee x_1. \qquad (10.2.4)$$

На другому кроці для побудови дешифраторів використовуємо логічні елементи, які наведені в табл. 10.1.1. Схеми дешифраторів, побудовані за логічними рівняннями 10.2.3 і 10.2.4, відповідно, приведені на рис. 10.2.1а і рис. 10.2.1б.



а) *б)*

Рис. 10.2.1

На рис. 10.2.1а приведена схема дешифратора з активним вихідним сигналом «1», а на рис. 10.2.1б — з активним вихідним сигналом «0». Дані дешифратори є лінійними. Логіка їх роботи на два входи x_1 і x_2, чотири прямих виходи (f_0, f_1, f_2, f_3) і чотири інверсних виходи $(\varphi_0, \varphi_1, \varphi_2, \varphi_3)$ приведена в табл. 10.2.1 і табл. 10.2.2 відповідно.

Таблиця 10.2.1

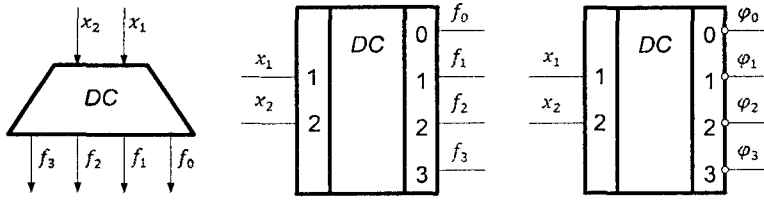
x_2	x_1	f_0	f_1	f_2	f_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Таблиця 10.2.2

x_2	x_1	φ_0	φ_1	φ_2	φ_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Як видно із табл. 10.2.1 і табл. 10.2.2, логіка функціонування дешифраторів відповідає рівнянням 10.2.3 і 10.2.4, за якими будувались дані дешифратори.

Умовно графічне позначення дешифратора на електричних схемах приведено на рис. 10.2.2.



а) *б)* *в)*

Рис. 10.2.2

На рис. 10.2.2а приведено функціональне позначення дешифратора на схемах, а на рис. 10.2.2б і рис. 10.2.2в — на принципових схемах. Логічну функцію дешифратора позначають буквами *DC* (*decoder*). Мітки лівого додаткового поля в умовному позначенні дешифратора на принципових схемах відтворюють вагу вхідних двійкових змінних, а мітки правого додаткового поля відповідають десятковим еквівалентам вхідних комбінацій двійкових змінних.

Дешифратори у комп'ютерах використовують для виконання наступних операцій:

а) дешифрації коду операції, записаного до регістра команд процесора, що забезпечує вибір необхідної мікропрограми;

б) перетворення коду адреса операнда команди в управляючі сигнали вибору заданої комірки пам'яті в процесі запису або читання інформації;

в) забезпечення візуалізації на зовнішніх пристроях;

г) реалізації логічних операцій, побудови мультиплексорів і демультиплексорів.

Означення 10.2.2. **Шифратором** називають функціональний пристрій комп'ютера, призначений для перетворення вхідного m-розрядного унітарного позиційного коду у вихідний n-розрядний двійковий позиційний код.

Двійкові шифратори виконують функцію, обернену функції дешифратора. При активізації одного із входів шифратора на його виходах формується код, який відображає номер активного входу. Повний двійковий шифратор має $m = 2^n$ входів і n виходів.

Для побудови шифраторів використовують три кроки. На першому кроці за допомогою таблиці функціонування описують роботу необхідного шифратора. На другому кроці, користуючись таблицею, знаходять логічні рівняння роботи шифратора, за якими на третьому кроці будують сам шифратор, вибравши необхідну елементну базу.

Так, наприклад, необхідно побудувати двійковий шифратор з $m = 8$ і $n = 3$. Тоді, виконуючи крок перший, будуємо таблицю функціонування такого шифратора, яка для $m = 8$ і $n = 3$ наведена в табл. 10.2.3.

Таблиця 10.2.3

Вхід	Вихід		
	x_3	x_2	x_1
φ_0	0	0	0
φ_1	0	0	1
φ_2	0	1	0
φ_3	0	1	1

Вхід	Вихід		
	x_3	x_2	x_1
φ_4	1	0	0
φ_5	1	0	1
φ_6	1	1	0
φ_7	1	1	1

На другому кроці, користуючись таблицею функціонування шифратора, знаходимо логічні рівняння його роботи:

$$x_1 = \varphi_1 \vee \varphi_3 \vee \varphi_5 \vee \varphi_7;\ \ x_2 = \varphi_2 \vee \varphi_3 \vee \varphi_6 \vee \varphi_7;$$
$$x_3 = \varphi_4 \vee \varphi_5 \vee \varphi_6 \vee \varphi_7. \qquad (10.2.5)$$

Для реалізації рівнянь 10.2.5 на елементній базі «*I-HI*», яка широко застосовується в периферійних пристроях комп'ютерів, перетворюємо їх за законами алгебри логіки до виду:

$$x_1 = \overline{\overline{\varphi}_1 \vee \overline{\varphi}_3 \vee \overline{\varphi}_5 \vee \overline{\varphi}_7};\ \ x_2 = \overline{\overline{\varphi}_2 \vee \overline{\varphi}_3 \vee \overline{\varphi}_6 \vee \overline{\varphi}_7};$$
$$x_3 = \overline{\overline{\varphi}_4 \vee \overline{\varphi}_5 \vee \overline{\varphi}_6 \vee \overline{\varphi}_7}. \qquad (10.2.6)$$

На третьому кроці, використовуючи рівняння 10.2.6, будуємо сам шифратор, наведений на рис. 10.2.3

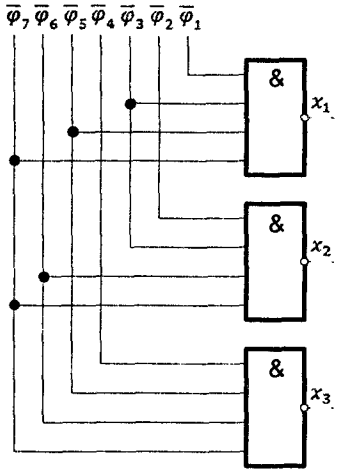
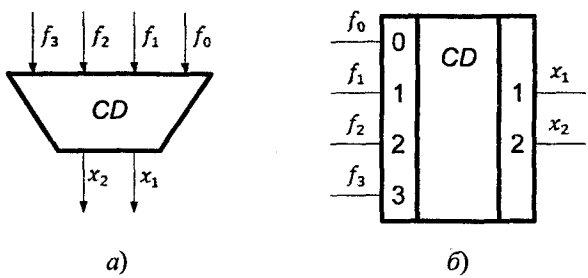


Рис. 10.2.3

Умовно графічне позначення шифраторів на схемах наведені на рис. 10.2.4.



а) *б)*

Рис. 10.2.4

На рис. 10.2.4а приведено функціональне позначення шифратора, а на рис. 10.2.4б — принципіальне. Логічну функцію шифратора позначають буквами *СД* (*coder*). Входи шифратора нумерують цифрами (0, 1, 2, ..., $m-1$), а мітки виходів відтворюють вагу вихідних двійкових змінних (1, 2, 4, ..., 2^{n-1}).

В комп'ютерах шифратори використовують для виконання наступних функцій:

а) перетворення унітарного вхідного коду у вихідний двійковий позиціонний код;

б) вводу десяткових даних з клавіатури комп'ютера;

в) показу старшої одиниці в слові;

г) передачі інформації між різними пристроями при обмеженні на кількість ліній зв'язку .

10.3. Логіка побудови мультиплексорів та демультиплексорів

Означення 10.3.1. **Мультиплексором** називають функціональний пристрій комп'ютера, призначений для послідовної комутації інформації від одного із n входів на його спільний вихід.

Входи мультиплексора поділяють на інформаційні та керуючі (адресні). Конкретний вхід мультиплексора, що підключаєьться до його виходу, визначається адресним кодом $A_0, A_1, ..., A_{n-1}$. Зв'язок між числом інформаційних n і адресних m входів визначаються із співвідношення $n = 2^m$.

Для побудови мультиплексорів використовують три кроки. На першому кроці за допомогою таблиці функціонування описують логіку роботи мультиплексора. На другому кроці, користуючись таблицею функціонування, визначають вихідну функцію роботи мультиплексора, за якою на третьому кроці, вибравши елементну базу, будують необхідний мультиплексор.

Так, наприклад, необхідно побудувати мультиплексор, який має чотири входи. Тоді, згідно із першим кроком побудови, користуючись логікою роботи мультиплексора, будуємо таблицю його функціонування, табл. 10.3.1.

Таблиця 10.3.1

A_1	A_0	F_0	F_1	F_2	F_3	D
0	0	1	0	0	0	$F_0 \cdot x_0$
0	1	0	1	0	0	$F_1 \cdot x_1$
1	0	0	0	1	0	$F_2 \cdot x_2$
1	1	0	0	0	1	$F_3 \cdot x_3$

В таблиці прийняті такі позначення: A_1, A_0 — адресні входи мультиплексора; F_0, F_1, F_2 і F_3 — виходи внутрішнього дешифратора; x_0, x_1, x_2, x_3 — вхідна інформація мультиплексора; D — спільний інформаційний вихід мультиплексора.

На другому кроці, користуючись таблицею функціонування мультиплексора, визначимо його вихідну функцію D

$$D = F_0 \cdot x_0 \vee F_1 \cdot x_1 \vee F_2 \cdot x_2 \vee F_3 \cdot x_3. \qquad (10.3.1)$$

У формулі 10.3.1 використані виходи $F_0 ... F_3$ внутрішнього дешифратора мультиплексора. За табл. 10.3.1, формулою 10.3.1 можна також записати його вихідну функцію D із застосуванням змінних адресного входу

$$D = \overline{A}_1 \cdot \overline{A}_0 \cdot x_0 \vee \overline{A}_1 \cdot A_0 \cdot x_1 \vee A_1 \cdot \overline{A}_0 \cdot x_2 \vee A_1 \cdot A_0 \cdot x_3. \qquad (10.3.2)$$

На третьому кроці, використовуючи логічні рівняння 10.3.1 і 10.3.2, будуємо необхідні мультиплексори на елементах «*І*», «*НІ*», «*АБО*», рис. 10.3.1.

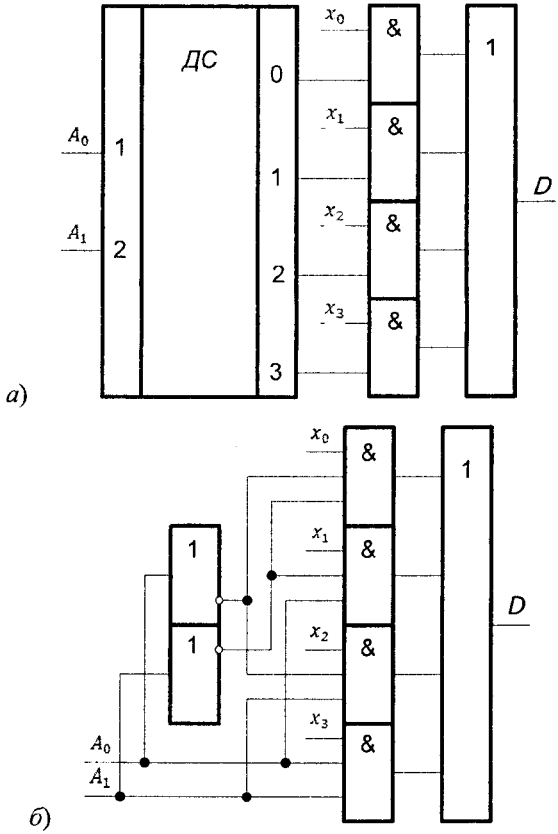
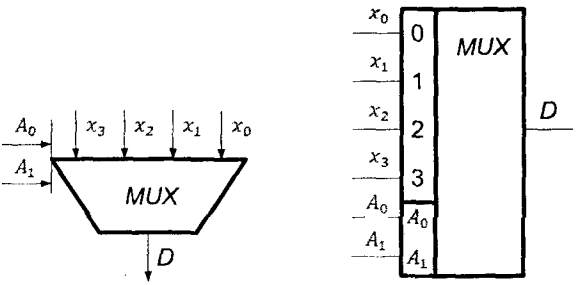


Рис. 10.3.1

Умовно графічне позначення мультиплексорів приведено на рис. 10.3.2.



а) *б)*

Рис. 10.3.2

На рис. 10.3.2а приведено позначення мультиплексора на функціональних, а на рис. 10.3.2б — принципових схемах. Функцію мультиплексора позначають буквами *MUX* (*multiplexor*).

Мультиплексори в комп'ютерах використовують для виконання таких операцій:

а) комутації як окремих ліній, так і шин передачі інформації;

б) перетворення паралельного коду в послідовний;

в) реалізації логічних функцій.

Означення 10.3.2. **Демультиплексором** називають функціональний пристрій комп'ютера, призначений для комутації сигналу з одного інформаційного входу D на один із n інформаційних виходів.

Входи демультиплексора, як і в мультиплексорі, поділяють на інформаційні та керуючі. Номер виходу, на який у кожний момент часу передається значення вхідного сигналу, визначається адресним кодом $A_0, A_1, ..., A_{n-1}$. Адресні входи m та інформаційні виходи n пов'язані співвідношенням $n = 2^m$.

Для побудови демультиплексорів використовують також три кроки. На першому кроці за допомогою таблиць функціонування описують логіку роботи демультиплексора. На другому, користуючись таблицею функціонування, визначають вихідні функції роботи демультиплексора, за яким на третьому кроці, вибравши елементну базу, будують необхідний демультиплексор.

Так, наприклад, необхідно побудувати демультиплелексор, який має чотири входи. Тоді, згідно із першим кроком, користуючись логікою роботи демультиплексора, будуємо таблицю його функціонування, табл. 10.3.2.

Таблиця 10.3.2

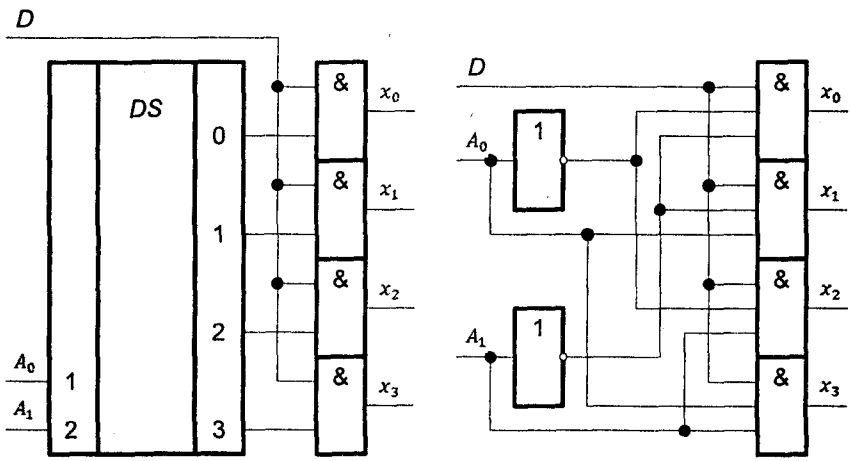
A_1	A_0	F_0	F_1	F_2	F_3	x_0	x_1	x_2	x_3
0	0	1	0	0	0	F_0D	—	—	—
0	1	0	1	0	0	—	F_1D	—	—
1	0	0	0	1	0	—	—	F_2D	—
1	1	1	0	0	1	—	—	—	F_3D

В таблиці прийняті такі позначення: F_0, F_1, F_2, F_3 — виходи внутрішнього дешифратора; D — інформаційний вхід; x_0, x_1, x_2, x_3 — інформаційні виходи демультиплексора.

На другому кроці, користуючись таблицею функціонування демультиплексора, визначимо його вихідні функції x_0, x_1, x_2, x_3:

$$x_0 = F_0 \cdot D = \overline{A}_1 \cdot \overline{A}_0 \cdot D; \quad x_1 = F_1 \cdot D = \overline{A}_1 \cdot A_0 \cdot D;$$
$$x_2 = F_2 \cdot D = A_1 \cdot \overline{A}_0 \cdot D; \quad x_3 = F_3 \cdot D = A_1 \cdot A_0 \cdot D. \qquad (10.3.3)$$

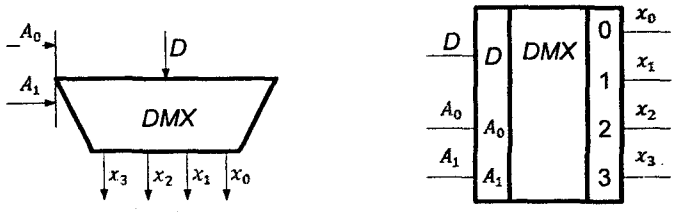
На третьому кроці, на базі рівнянь 10.3.3 будуємо схеми демультиплексорів із внутрішнім дешифратором рис. 10.3.3а і з адресним входами змінних на трьохвходових елементах «І», рис. 10.3.3б.



а) *б)*

Рис. 10.3.3

Умовне графічне зображення демультиплексорів приведено на рис. 10.3.4.



а) *б*)

Рис. 10.3.4

На рис. 10.3.4а приведено позначення демультиплексора на функціональних, а на рис. 10.3.4б — принципових схемах. Функцію демультиплексора позначають буквами *DMX* (*demultiplexor*).

Демультиплексор використовують для виконання таких операцій:

а) комутації як окремих ліній, так і шин передачі інформації в комп'ютерах;

б) перетворення послідовного коду в паралельний;

в) реалізації логічних функцій.

10.4. Логіка побудови суматорів

Означення 10.4.1. **Суматором** називають комбінаційний логічний пристрій комп'ютера, призначений для виконання операцій арифметичного додавання чисел, поданих у вигляді двійкових кодів.

Операція віднімання в суматорі заміняється операцією додавання двійкових чисел у додатковому або оберненому коді, §1.3 і §1.4. Операція множення та ділення зводиться до багатократного додавання і зсуву. Тому суматор є важливою частиною арифметично-логічного пристрою комп'ютера. Функцію суматора позначають буквами *SM* або $\sum$.

Суматор складається з окремих схем, які називаються **одноразрядними суматорами.** Ці схеми виконують усі дії із додавання значень однойменних розрядів двійкових чисел. Суматори класифікують за такими ознаками:

а) способу додавання — паралельні, послідовні, паралельно-послідовні;

б) числу входів — напівсуматори, однорозрядні і багаторозрядні суматори;

в) організації переносу зберігання результату додавання — комбінаційні, накопичувальні, комбіновані;

г) організації переносу між розрядами — з послідовним, паралельним або комбінованим переносом;

д) способу представлення від'ємних чисел — у додатковому або оберненому кодах, а також в їх модифікаціях;

є) часу додавання — синхронні, асинхронні.

В паралельних суматорах значення всіх розрядів операндів надходять одночасно на відповідні входи однорозрядних підсумовуючих схем. В послідовних суматорах значення розрядів операндів і перенос, який запам'ятався в попередньому такті, надходить послідовно в напрямку від молодших розрядів до старших на входи одного однорозрядного суматора. В паралельно-послідовних суматорах числа дробляться на частини, наприклад, байти, розряди байтів, надходять на входи восьмирозрядного суматора паралельно (одночасно), а самі байти — послідовно, в напрямку від молодших до старших з урахуванням запам'ятовуючого переносу.

В комбінаційніх суматорах результат операції додавання запам'ятовується в регістрі результату. В накопичувальних суматорах процес додавання об'єднується зі зберіганням результату, що пояснюється використанням Т-тригерів як однорозрядних схем додавання.

Організація переносу в суматорі практично визначає час виконання операції додавання. Послідовні переноси схемно утворюються просто, але вони не є швидкодійні. Паралельні переноси схемно значно складніші, але є помітно швидкодійніші.

Суматори, які мають постійний інтервал часу для додавання називають **синхронними**, а суматори, в яких додавання визначається моментом фактичного закінчення операції, — **асинхронним**.

Означення 10.4.2. **Однорозрядним суматором** називають логічну схему, яка виконує додавання значень i–x розрядів x_i і y_i двійкових чисел з урахуванням переносу z_i із молодшого сусіднього розряду і виробляє на виходах функції результат S_i і перенос P_i в старший сусідній розряд.

На основі однорозрядних схем додавання на три входи і два виходи будуються багаторозрядні суматори будь-якого типу. Логіка їх побудови аналогічна логіці побудови комбінаційних схем,

яка розглянута у попередніх розділах. Отже, спочатку визначають алгоритм функціонування, який, звичайно, подають таблицею, а потім за допомогою рівняння (системи рівнянь), які отримують із таблиці функціонування, будують необхідний за умовою суматор.

Так, наприклад, алгоритм роботи однорозрядного суматора відображається таблицею істинності, табл. 10.4.1.

Таблиця 10.4.1

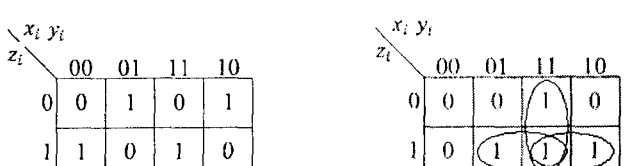
x_i	y_i	z_i	S_i	P_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Із таблиці 10.4.1 випливає система логічних функцій для результату S_i і переносу в P_i в *ДДНФ*.

$$S_i = \overline{x}_i \cdot \overline{y}_i \cdot z_i \vee \overline{x}_i \cdot y_i \cdot \overline{z}_i \vee x_i \cdot \overline{y}_i \cdot \overline{z}_i \vee x_i \cdot z_i \cdot y_i. \qquad (10.4.1)$$

$$P_i = \overline{x}_i \cdot y_i \cdot z_i \vee x_i \cdot \overline{y}_i \cdot z_i \vee x_i \cdot y_i \cdot \overline{z}_i \vee x_i \cdot y_i \cdot z_i. \qquad (10.4.2)$$

Мінімізація функцій 10.4.1 і 10.4.2 за допомогою карт Карно приведено на рис. 10.4.1а і рис. 10.4.1б відповідно



а) *б)*

Рис. 10.4.1

Як випливає із карт Карно, функція S_i не мінімізується, а функція P_i мінімізується зі зменшенням рангу кон'юнкції і використовує тільки прямі значення змінних

$$P_i = x_i \cdot y_i \vee x_i \cdot z_i \vee y_i \cdot z_i = x_i \cdot y_i \vee (x_i \vee y_i) \cdot z_i. \quad (10.4.3)$$

При проектуванні комбінаційних однорозрядних суматорів необхідно враховувати такі фактори:

а) схема повинна мати однаковість структури і мінімальну вартість, тобто мати по можливості мінімальну кількість входів у всіх елементів;

б) для швидкодії багаторозрядного суматора необхідний мінімальний час отримання функції переносу $t_n = k \cdot t_p$, де k — кількість послідовно включених елементів від входів до виходів P_i або $\overline{P}_i$, t_p — середня затримка розповсюдження сигналу одним логічним елементом;

в) для схем однорозрядних суматорів на основі рівнянь 10.4.1 і 10.4.2 необхідно проектувати як прямі P_i, так і інверсні $\overline{P}_i$ значення функції переносу.

Для побудови схеми однорозрядного суматора на логічних елементах «*І-НІ*», рівняння 10.4.1 і 10.4.3, із використанням подвійної інверсії і правил де Моргана перетворюємо на такий вигляд:

$$S_i = \overline{\overline{x}_i \cdot \overline{y}_i \cdot z_i \cdot \overline{x}_i \cdot y_i \cdot \overline{z}_i \cdot x_i \cdot \overline{y}_i \cdot \overline{z}_i \cdot x_i \cdot y_i \cdot z_i},$$
$$P_i = \overline{\overline{x_i \cdot y_i} \cdot \overline{x_i \cdot z_i} \cdot \overline{y_i \cdot z_i}}. \quad (10.4.4)$$

Схема однорозрядного суматора, побудована на елементах «І-НІ» у відповідності з рівняннями 10.4.4, приведена на рис. 10.4.2 *а*.

Рівняння 10.4.1 і 10.4.2 можуть бути також виражені через функцію «*Виключаюче АБО*»

$$S_i = (x_i \oplus y_i) \cdot \overline{z}_i \vee \overline{(x_i \oplus y_i)} \cdot z_i = x_i \oplus y_i \oplus z_i, \quad (10.4.5)$$

$$P_i = x_i \cdot y_i \vee (\overline{x}_i \cdot y_i \vee x_i \cdot \overline{y}_i) \cdot z_i = x_i \cdot y_i \vee (x_i \oplus y_i) \cdot z_i. \quad (10.4.6)$$

Схема однорозрядного суматора на елементах «*Виключаюче АБО*», згідно із рівнянням 10.4.5 і 10.4.6, приведена на рис. 10.4.2 *б*, а його функціональна схема на рис. 10.4.2 *в*.

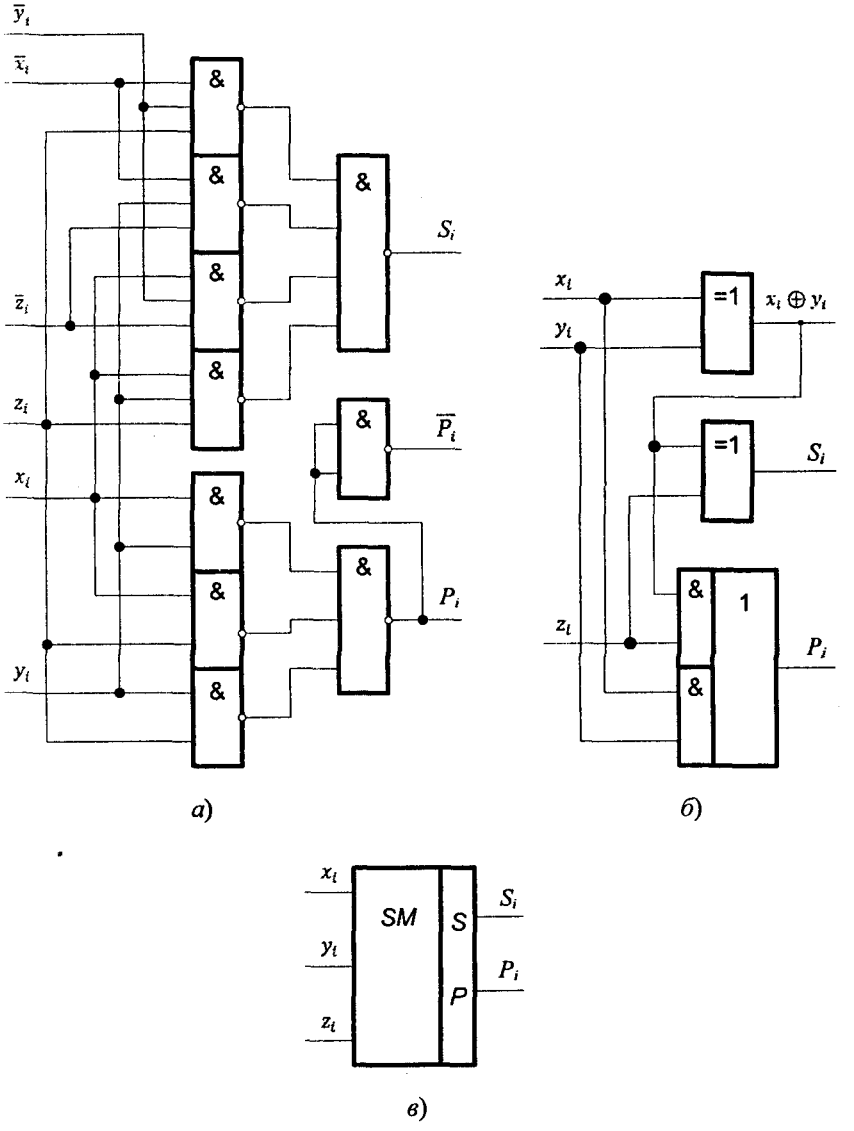


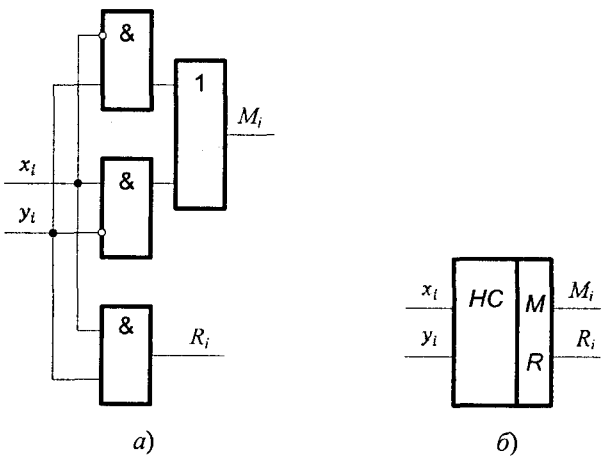
Рис. 10.4.2

Означення 10.4.3. **Напівсуматором** називають логічну схему, яка виконує додавання значень i-х розрядів x_i і y_i двійкових чисел x,

y і реалізує на вході значення результату M_i і перенесення в старший сусідній розряд R_i

$$M_i = \overline{x}_i \cdot y_i \vee x_i \cdot \overline{y}_i = x_i \oplus y_i;\ R_i = x_i \cdot y_i. \qquad (10.4.7)$$

Таким чином, напівсуматор виконує частину підсумування в i-му розряді, оскільки не враховує перенесення із сусіднього молодшого розряду. Схема напівсуматора, побудована на основі рівнянь 10.4.7 приведена на рис. 10.4.3 а, а його функціональна схема — на рис. 10.4.3 б.



а) *б)*

Рис. 10.4.3

10.5. Логіка побудови компараторів

Означення 10.5.1. **Компаратором (схемою порівняння)** називають комбінаційний логічний пристрій для порівняння чисел, поданих у двійковому коді.

Основними відношеннями при порівнянні слід уважати «*дорівнює*», «*більше*» і «*менше*». Ці відношення широко використовують у мікропроцесорах, а також у пристроях контролю та діагностики комп'ютерів. Після виконання конкретної команди в комп'ютері автоматично формулюються ознаки результатів операції. Ці ознаки називають **прапорцями** і їх розміщують у спеціальний регістр. До прапорців, наприклад, належать ознаки нульового результату, переповнення розрядної сітки, знак результату операції, наявність перенесення із старшого розряду суматора, парне і непарне число одиниць в результаті і таке інше.

Логіка побудови компараторів аналогічна логіці побудови комбінаційних схем, розглянутих у попередніх параграфах. Тобто, спочатку визначають алгоритм функціонування, який може бути заданий або словесно, або таблицею, а потім за допомогою рівняння (системи рівнянь) будують необхідний за умовою компаратор. Розглянемо принцип побудови компараторів на прикладах.

Приклад 10.5.1. Побудувати схему порівняння двійкового слова $A = A_2\ A_1\ A_0$ із слідуючими заданими константами

$$F_1 := (A = 001);\ F_2 := (A \le 011).$$

Розвязання. Будуємо таблицю істинності, табл. 10.5.1, для заданого вхідного слова. Виходячи з умови задачі і таблиці істинності це відношення буде мати такий вигляд:

$$F_1 := \overline{A}_2\ \overline{A}_1\ A_0;\ F_2 = \overline{A}_2. \qquad (10.5.1)$$

Таблиця 10.5.1

A_2	A_1	A_0	F_1	F_2
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

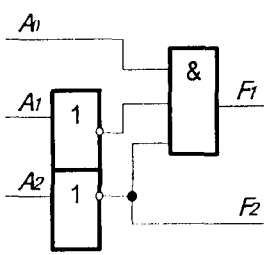


Рис. 10.5.1

Схема порівняння слова з константою, відповідно до рівнянь 10.5.1, приведена на рис. 10.5.1.

Приклад 10.5.2. Побудувати схему порівняння двох *i-x* розрядів $A_i = B_i$. Для порівняння будуємо таблицю істинності, табл. 10.5.2

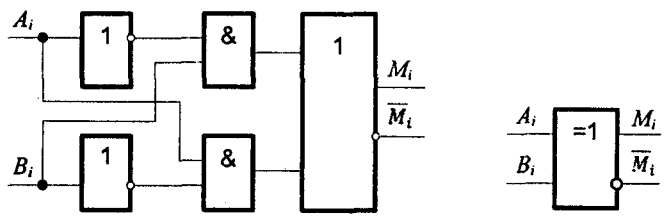
Із таблиці істинності, яка задає умову рівності M_i двох *i-x* розрядів A_i і B_i, отримаємо

$$r_i = A_i \cdot B_i \vee \overline{A}_i \cdot \overline{B}_i = \overline{A_i \oplus B_i} = \overline{M}_i, \qquad (10.5.2)$$

де M_i — функція додавання за модулем два (*mod*2 — *«Виключаюче АБО»*).

Таблиця 10.5.2

A_i	B_i	r_i
0	0	1
0	1	0
1	0	0
1	1	1



а) *б)*

Рис. 10.5.2

Принципова схема, яка реалізує цю функцію, показана на рис. 10.5.2а, а її функціональна схема приведена на рис. 10.5.2б.

Приклад 10.5.3. Побудувати схему порівняння двох чотирьохрозрядних двійкових слів A та B.

Розв'язання. Ознака рівності двох чотирьохрозрядних слів A та B визначається добутком порозрядних умов r_i.

$$F_{A=B} = r_1 \cdot r_2 \cdot r_3 \cdot r_4 = \overline{M}_1 \cdot \overline{M}_2 \cdot \overline{M}_3 \cdot \overline{M}_4. \qquad (10.5.3)$$

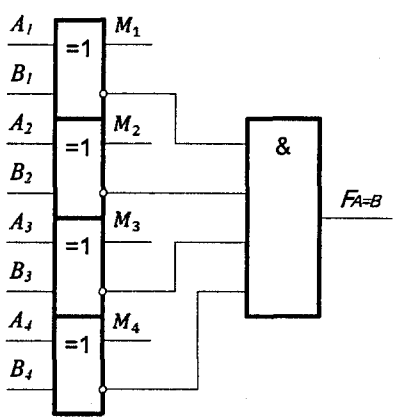


Рис. 10.5.3

Схема порівняння двох чотирьох розрядних слів A та B, згідно із виразом 10.5.3, приведена на рис. 10.5.3.

При більшій розрядності порівняння використовують різні групові рівні, на яких реалізують спільні логічні множення групових ознак.

Контрольні запитання

1. Назвіть основні логічні елементи булевих функцій.
2. Яка різниця між логічними елементами «*І*» і «*АБО*», «*Виключаюче АБО*» і «*Еквівалентність*», «*Якщо, то*» і «*Заборона*»?
3. Що називають дешифратором?
4. Що називають шифратором?
5. Яка різниця між дешифратором і шифратором?
6. Де використовують дешифратор і шифратор?
7. Як позначають на схемах дешифратори і шифратори?
8. Накресліть функціональну і принципову схему дешифратора і шифратора.
9. Що називають мультиплексором?
10. Що називають демультиплексором?
11. Яка різниця між мультиплексором і демультиплексором?
12. Як позначають на схемах демультиплексор та мультиплексор?
13. Накресліть функціональну і принципову схему демультиплексора та мультиплексора.
14. Де використовують демультиплексор і мультиплексор?
15. Що називають суматором?
16. Які операції виконує суматор?
17. Як позначають суматори на схемах?
18. Які бувають суматори?
19. За якими ознаками класифікують суматори?
20. Що називають напівсуматором і чим він відрізняється від суматора?
21. Як позначають напівсуматор на схемах?
22. Що називають компаратором і як його позначають?
23. Для яких цілей використовують компаратор?

Коментарі

В даному розділі для логіки побудови дешифраторів, шифраторів, мультиплексорів, демультиплексорів використані матеріали літератури [22, 27], а логіка побудови суматорів і компараторів випливає з [3, 27].

Розділ 11

ЛОГІКА ПОБУДОВИ КОМБІНАЦІЙНИХ СХЕМ НА ПРОГРАМОВАНИХ ЛОГІЧНИХ МАТРИЦЯХ

11.1. Призначення і ділянки застосування

Програмовані логічні матриці (ПЛМ) являють собою логічну схему для перетворення множини вхідних значень $X = \{x_1, x_2, ..., x_m\}$ у відповідну множину вихідних даних $Y = \{y_1, y_2, ..., y_m\}$ у двійковому коді. Правило перетворення вхідних змінних у функціях задається таблицею істинності. ПЛМ реалізує систему булевих функцій, представлених в ДДНФ або МДНФ, або в ДНФ.

Програмовані логічні матриці знайшли широке застосування в логічних інтегральних схемах (ПЛІС). Наприклад, ПЛІС із плавкими запобіжниками за технологією ТТЛШ, які виготовляються в НДУМЕ, м. Зеленоград Росія. В їх складі уже давно відомі ПЛМ К556РТ1, КР556РТ2, КР556РТ21.

Завжди виникає питання, де можливо застосовувати ПЛІС?

По-перше, при розробці оригінальних та нестандартних пристроїв у комп'ютерах і системах управління, а також для заміни звичайних інтегральних мікросхем малої та середнього ступеня інтеграції. При цьому значно зменшуються розміри, потужність споживання і збільшується надійність пристроїв і систем, там де вони використовуються.

По-друге, використання ПЛІС дає можливість значно зменшити час і затрати на проектування схем, розширити можливості розробки модифікацій комп'ютерів, налагодження комп'ютерних пристроїв, що особливо суттєво в стендовому обладнанні, на етапах розробки і виготовлення дослідних партій нових виробів, а також емуляції схем.

По-третє, при проектуванні на основі ПЛІС пристроїв для захисту програмного забезпечення виробів від несанкціонованого доступу і копіювання. ПЛІС має таку технологічну особливість, як

«біт секретності», після програмування якого схема стає недоступною для читання.

Але найбільше ПЛІС використовують у мікропроцесорній і обчислювальній техніці. На їх основі розробляють контролери, адресні дешифратори, логіку обладнання мікропроцесора й інш. На основі ПЛІС часто виготовляють мікропрограмні автомати, спеціалізовані пристрої, схеми обробки сигналів та відображення, процесори швидкого перетворення функцій Фур'є і т. д.

Якщо за кордоном ПЛІС уже зайняли достойне місце а арсеналі розробника, то в пострадянських країнах ці технології тільки починають по-справжньому розвиватися. Відставання пояснюється рядом причин. По-перше, дуже звужена номенклатура ПЛІС на нашому ринку елементної бази. По-друге, практична відсутність у наших спеціалістів сучасних систем проектування. По-третє, недостатність інформації в технічній літературі про ПЛІС, їх застосування і методи програмування.

11.2. Принципи побудови базової програмованої логічної матриці

Виготовлювані електронною промисловістю ПЛІС мають базову структуру програмованої логічної матриці, яка включає матрицю кон'юнкторів (матриця «*І*») і матрицю диз'юнкторів (матриця «*АБО*»). Принцип побудови таких ПЛМ розглянемо на ПЛІС серії К556РТ1. Структурна схема даної ПЛІС приведена на рис. 11.2.1.

Дана ПЛІС включає матрицю кон'юнкторів (матрицю «*І*»), матрицю диз'юнкторів (матриця «*АБО*»), блок вхідних підсилювачів (ВП), блок вихідних каскадів (ВК), схему дозволу вибірки кристалу (ДВ), програмований дешифратор, програмовані адресні формувачі (АФ1, АФ2). Вхідні підсилювачі формують прямі й інверсні значення вхідних змінних за всіма шістнадцятьма входами (А1...А16).

Програмований дешифратор (ДС) і програмовані адресні формувачі (АФ1, АФ2) використовують тільки в режимах програмування та контролю ПЛІС. Організація цих режимів достатньо складна і в даному підручнику не розглядується.

Для наочності та більш повного розуміння принципу побудови ПЛМ розглянемо базову функціональну схему ПЛІС серії К556РТ1, яка включає в себе лише основні вузли схеми матриці «*І*», «*АБО*», вхідні і вихідні каскади, рис. 11.2.2.

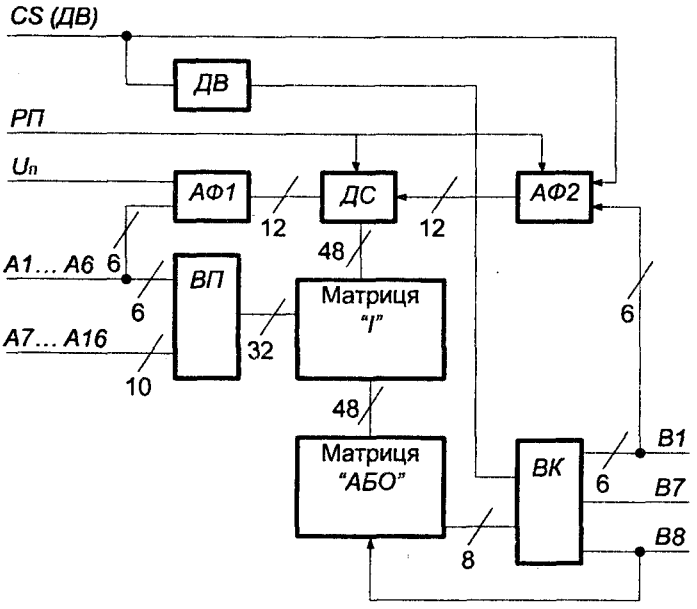


Рис 11.2.1

Вхідні підсилювачі (ВП1...ВП16) формують прямі й інверсні значення вхідних змінних, які надходять в матрицю «*І*». Для управління вхідними підсилювачами є шістнадцять входів (А1...А16). Вхідні підсилювачі побудовані на основі двох включених послідовно буферних логічних схем «*І-НІ*».

Основними вузлами мікросхеми К556РТ1 є матриці «*І*» і «*АБО*», які реалізують двохрівневі логічні функції. Перший рівень ПЛМ складається із 48 кон'юнкторів (матриця «*І*»), які з'єднані за допомогою плавких ніхромових перемичок із будь-яким із шістнадцяти спільних входів через буферні схеми. В матриці «*І*» реалізують кон'юнкції вхідних змінних, причому кожна вхідна змінна може входити в кон'юнкцію або прямим або інверсним значенням, або не входити зовсім. Вхідні сигнали, які з'являються на вхідних шинах матриці «*І*», вводяться в матрицю «*АБО*», яка утворює другий логічний рівень і реалізує диз'юнкції заданих кон'юнкцій. Матриця «*АБО*» утворює вісім диз'юнкторів (по одному «*АБО*» на виході ПЛІС), кожний із яких може бути вибірково з'єднаний із будь-яким із сорока восьми кон'юнкторів.

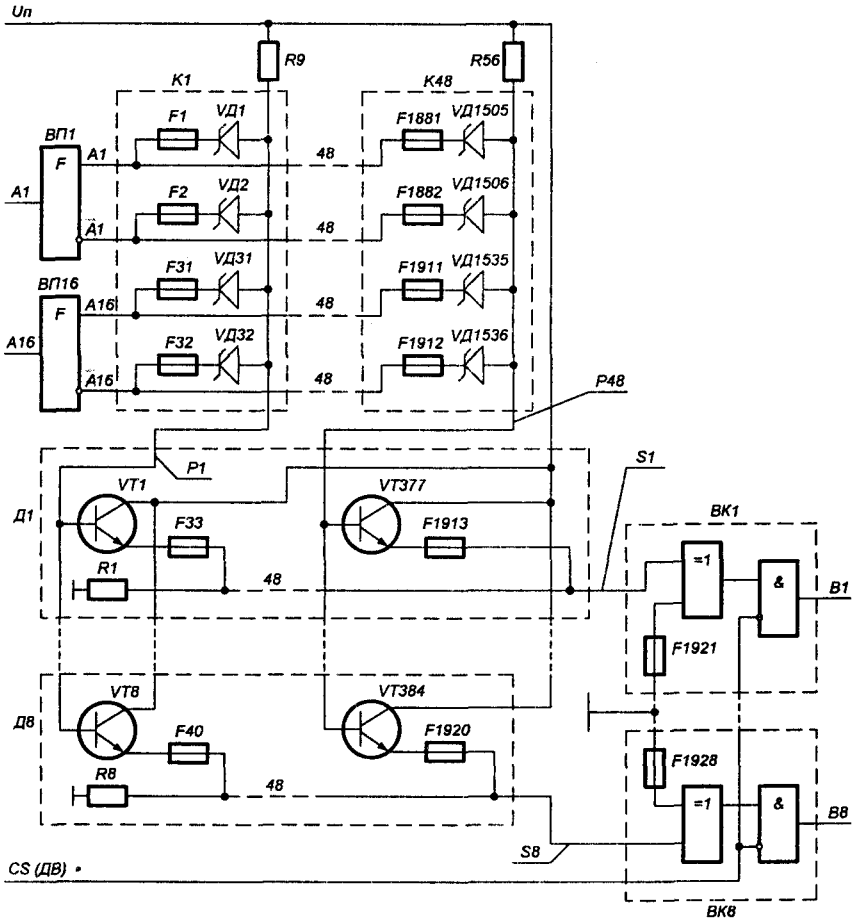


Рис. 11.2.2

де ВП1…ВП16 — вхідні підсилювачі;
К1…К48 — кон'юнктори матриці «*І*»;
Д1… Д8 — диз'юнктори матриці «*АБО*»;
ВК1… ВК8 — вихідні каскади;
Р1…Р48 — шини кон'юнкцій;
S1…S8 — шини диз'юнкцій;
F1…F1928 — плавкі ніхромові перемикачі;

VD1...VD1536 — діоди Шотки;
VT1...VT34 — транзистори;
R1...R6 — резистори.

Шини, які з'єднують ці дві матриці, називають шинами кон'юнкцій і позначають P1...P48, а шини, які з'єднують матрицю «*АБО*» з вихідними каскадами, називають шинами диз'юнкцій і позначають S1...S8.

Програмованим елементом матриці «*І*» є діод Шотки з плавкою ніхромовою перемичкою, а матриці «*АБО*» включені за схемою емітерного повторювача, *n-p-n* транзистор з плавкою ніхромовою перемичкою в емітері.

Вихідні каскади ВК1...ВК8 включають логічні схеми «*Виключаюче АБО*» і підсилювачі зчитування. Наявність на вході каскаду логічної схеми «*Виключаюче АБО*» дозволяє інвертувати рівень вихідного сигналу залежно від сигналу на вході, тобто дозволяє програмувати або активний високий, або активний низький рівень вихідного сигналу. Заземлення (підключення до сигналу «О») одного із двох входів логічної схеми «*Виключаюче АБО*» через плавку перемичку веде до того, що активним рівнем виходу стає вихідна напруга високого рівня, а виплавлення цієї перемички веде до того, що активним рівнем стає вихідна напруга низького рівня.

Підсилювачі зчитування побудовані на логічних схемах, що управляють сигналами, які надходять від матриці «*АБО*» і від схеми дозволу вибірки.

ПЛІС як базова програмована логічна матриця в режимі обробки інформації працює таким чином. Вхідні змінні A1...A16 через блок вхідних підсилювачів в прямому й інверсному значенні надходять на матрицю «*І*», де за допомогою діодів Шотки і плавких ніхромових перемичок утворюють потрібні кон'юнкції P1...P48, які логічно підсумовуються матрицею «*АБО*», утворюючи проміжні логічні функції S1...S8. Дані функції надходять у вихідні каскади для подальшого їх перетворення і видачі на виходи B1...B8 ПЛМ.

Умовне графічне позначення мікросхеми К556РТ1 приведено на рис. 11.2.3,

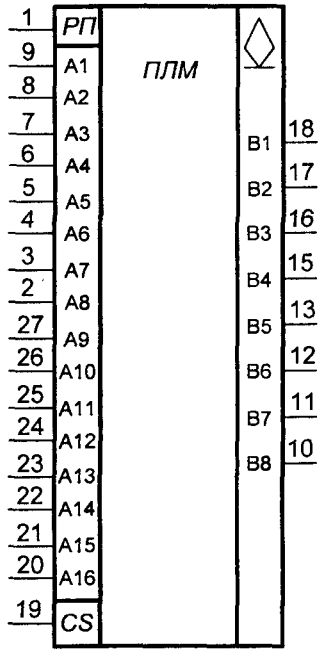


Рис. 11.2.3

де входи і виходи мікросхеми визначають:

1 — вхід програмування РП;

2...9 — входи підключення вхідних змінних А1...А8;

10...13 — виходи отриманих функцій В8...В5;

14 — спільний вихід (вихід подачі «О» В);

15...18 — виходи отриманих функцій В4...В1;

19 — вхід дозволу роботи (вибору) мікросхеми;

20...27 — входи підключення вхідних змінних А16...А9;

28 — вхід подачі джерела живлення (+5В).

11.3. Рекомендації із програмування базової логічної матриці

Програмування і контроль базової логічної матриці розглянемо на ПЛІС серії К556РТ1. Дана ПЛІС виготовляється і поставляється споживачу не запрограмованою, тобто в такому стані, що кожний

кон'юнктор отримує як прямі, так і інверсні значення від кожної вхідної змінної A_i, кожний диз'юнктор має всі сорок вісім кон'юнкцій, а для кожного виходу активним рівнем є високий і на всіх виходах присутня напруга низького рівня при напрузі на вході CS (0В).

Кожний програмований кон'юнктор P_n формує необхідну кон'юнкцію від вхідних змінних, причому кожна змінна може входити в кон'юнкцію прямим значенням, інверсним значенням або не входити зовсім. Ці стани реалізують за допомогою відповідних плавких перемичок у матриці «*І*». Якщо кон'юнктор P_n має вхідну змінну A_i, то перемичка, що з'єднує цей кон'юнктор із шиною вхідної змінної $\overline{A}_i$, повинна бути розплавлена і навпаки. Якщо змінна A_i не повинна входити в кон'юнктор P_n, то дві перемички вхідних змінних A_i і $\overline{A}_i$ повинні бути розплавлені.

Якщо число використаних вхідних змінних A_i менше шістнадцяти, то невикористані змінні повинні бути виключені у всіх аналогічних кон'юнкторах, тобто відповідні їм плавкі перемички в матриці «*І*» повинні бути розплавлені в процесі програмування.

Програмування диз'юнкторів виконується тільки для тих випадків, коли кон'юнкція не включається у вхідну функцію. Якщо кількість використаних функцій менше восьми, то всі плавкі перемички в матриці «*АБО*», що з'єднують невикористані диз'юнктори і використані або невикористані кон'юнктори, переплавляти не потрібно.

11.4. Програмування базової логічної матриці

Програмування базової логічної матриці розглянемо окремо для матриць «*І*», «*АБО*» і активного рівня виходу мікросхеми К556РТ1.

Програмування активного рівня виходів В1...В8 відбувається перед програмуванням матриць «*І*» і «*АБО*». В початковому стані всі ніхромові перемички вихідних каскадів цілі, при цьому рівень вихідного сигналу у вихідному каскаді не інвертується, і тому рівень активності виходів В1...В8 – високий. Переплавлення одної перемички відбувається при подачі на відповідний вихід напруги $U_{вих.пр.}$ При цьому спрацьовує схема програмування перемички у вихідному каскаді, і через перепалювану перемичку протікає руйнуючий її імпульс струму.

При даному програмуванні необхідно:

– виводи 14,1 мікросхеми підключити до ОВ, а до виводу 28 подати напругу 0...0,4В;

– виводи 10...13, 15...18, крім програмуваного, через резистор 10кОм підключити до джерела живлення 5В±10%;

– до виводів 2...9, 19...27 підключити напругу 2,4...4,5 В;

– на програмований вивід подати напругу 17±1В і утримувати її 1...5 мс;

– через 10...15 мкс після зняття напруги з програмованого виводу напругу на виводі 28 збільшити до 9,0±0,5В;

– через 10...15 мкс після збільшення напруги на виводі 28 напругу на виводі 19 зменшити до 0...0,4В, на виводах 2, 3, 20...27 установити напругу 0...0,4В, на виводах 4...9 — напругу 2,4...4,5В, а на програмованому виводі виконати контроль напруги, величина якої при позитивному результаті повинна бути 2,4...4,5В.

У випадку негативного результату програмування, тобто при $U^{1}_{вих} = 0 \ldots 0,4В$, необхідно ще раз виконати програмування через $t \geq 10$ мс після закінчення контролю.

Програмування матриці *«I»* відбувається наступним чином. Для вибору потрібної перемички в мікросхемі є дешифратор *DC*, рис. 11.2.1, який підключає до джерела програмованого стуму відповідну складову частину матриці «*I*». Для управління дешифратором використовують шість адресних формувачів АФ2, адресація яких відбувається з вихідних виводів В1...В6. А це в свою чергу вимагає, щоб усі виводи програмованої мікросхеми були в закритому стані, для цього на вхід *CS* необхідно подати напругу $U_{вхDB}$, що приведе до закриття транзисторів усіх підсилювачів зчитування і на виводи В1...В8 можна подавати адресний код, що відповідає номеру програмованої діодної частини.

Для забезпечення розплавлення тільки потрібної перемички із числа перемичок вибраної дешифратором діодного складання необхідно забезпечити закриття всіх виходів вхідних підсилювачів (як прямих, так і інверсних), крім програмованого. Це забезпечується подачею напруги на входи всіх вхідних підсилювачів, крім одного. На вході вибраного вхідного підсилювача подають напругу високого рівня $U_{вх}$, якщо необхідно переплавити перемичку, з'єднаного з інверсним виходом, або напругу низького рівня $U^{0}_{вх}$ — для прямого виходу.

За кожний цикл програмування переплавляється тільки одна перемичка. Імпульс програмованого струму формується при подачі на програмований РП напруги $U_{вх.рп}$.

При даному програмуванні необхідно:

– вивід 14 мікросхеми підключити до ОВ, а на вивід 28 подати напругу 5 В ± 5%;

– на виводі 19 установити напругу 2,4...4,5 В, а на виводі 1 — 0...0,4 В;

– виводи 2...9, 2...27, крім програмованого, підключити до джерела 10 В ± 5%;

– на кожний вивід 12, 13, 15...18 (18 — молодший розряд) подати напругу 0...0,4 В або 2,4...4,5 В у відповідності з кодом адреса кон'юнкції;

– кон'юнкцією включається пряме значення вхідної змінної або напруга 0...0,4 В, якщо в кон'юнкцію включається інверсне значення вхідної змінної;

– через 10...15 мкс напруга на виході 1 збільшується до 17 ± 1 В і утримується в процесі всього наступного переходу;

– через 10...15 мкс напруга на виводі 19 збільшується до 10 В ± 5 % і утримується 1...5 мс;

– через 10...15 мкс після зняття напруги на виводі 19 напруга на виводі 1 знижується до 0...0,4 В;

– через 10...15 мкс напруга на виводі 19 збільшується до 10 В ± 5 % і на виводі 10 відбувається контроль напруги, величина якої при позитивному результаті програмування повинна бути 2,4...4,5 В.

За негативного результату програмування (0...0,4 В) необхідно виконати повторне програмування шляхом його одно-, двохкратного повторення через $t \geq 10$мс після закінчення контролю.

Якщо вхідна змінна і при цьому не включається в кон'юнкцію, то її необхідно виключити із кон'юнкції шляхом подачі на програмований вхід напруги 2,4...4,5 В, а потім 0...0,4 В, і вхідних значень згідно з описаним вище.

Програмування матриці «*АБО*» відбувається таким чином. У початковому стані всі ніхромові перемички матриці «*АБО*» цілі. Для формування потрібних функцій необхідно в кожний із них включити ті кон'юнкції, які не повинні входити у відповідну функцію, тобто розплавити деякі перемички матриці.

Для програмування матриці «*АБО*» використовують той же дешифратор *DS*, рис. 11.2.1, що і при програмуванні матриці «*І*», але управління ним відбувається через інші групи програмованих адресних формувачів АФ1 із боку вхідних виводів А1...А6. Підклю-

чення АФ1 до джерела живлення і установка вхідних підсилювачів в необхідний стан відбувається при подачі на мікросхему збільшеної напруги. На виводи А1...А6 подається код, що відповідає номеру логічного добутку, який необхідно виключити із даної функції, а на вхід «*CS*» (DB) — напругу 2,4...4,5В, яка встановлює виходи всіх підсилювачів зчитування в закритий стан. На вихід відповідної функції, із якої виключається вибрана кон'юнкція, подається напруга $U_{вих.ф.}$. Імпульс програмованого струму, який протікає по вибраній перемичці, формується при подачі на програмований вхід *РП* напруги $U_{вих.пр.}$, а на вхід *CS* (*DB*) — $U_{вих.DB}$. За кожний цикл програм програмується тільки одна перемичка.

При даному програмуванні необхідно:

– вивід 14 підключити до 0В, вивід 28 — до 9,0 ± 0,5В, вивід 19 — до 2,4...4,5В, а виводи 1...3, 20...27 — до 0...0,4В;

– на виводи 4...9 подати напругу 0...0,4В і 2,4...4,5В у відповідності з кодом адреси кон'юнкції, яку не включають у вхідну функцію;

– виводи 10...13, 15...18, крім програмованого, підключити до джерела постійної напруги 4,5 В ± 10%;

– на програмованому виводі виставляється напруга 4,5 ± 5%;

– через 10...15 мкс після подачі напруги на програмований вивід напруга на виводі 1 збільшується до10В ± 5% і утримується до *наступного переходу*;

– через 10...15 мкс після збільшення напруги на виводі 1 напруга на виводі 19 збільшується до 10В ± 5% і утримується 1...5 мс, після чого зменшується до 2,4...4,5 В;

– через 10...15 мкс після зняття напруги на виводі 19, напруга на виводі 1 знижується до 0...0,4В

– через 10...15 мкс, після зниження напруги на виводі 1, напруга зовнішнього джерела 10 ± 5 %, від програмованого виводу відключається, а на вивід 19 виставляєтья напруга 0...0,4В, після чого на програмованому виводі відбувається контроль напруги, величина якої при довільному результаті програмування повинна бути 2,4...4,5В для виводу з активним низьким рівнем або 0...0,4В для виводу з активним високим рівнем.

За негативного результату програмування, тобто при 0...0,4В для виводу з активним рівнем, необхідно повторити його через $t \geq$ 10 мс після контролю, згідно із наведеною вище програмою.

11.5 Логіка побудови комбінаційних схем на програмованих логічних матрицях

На ПЛМ зручно будувати як одновихідні так і особливо багатовихідні логічні комбінаційні схеми.

Логіка побудови комбінаційних схем на програмованих логічних матрицях має такі кроки. На першому кроці визначають кількість вхідних і вихідних змінних, а також кон'юнкцій, які необхідно реалізувати на ПЛМ. На другому кроці, використовуючи дані логічних функцій першого кроку, визначають дану ПЛМ (їх сукупність) для реалізації заданих функцій. На третьому кроці, згідно з алгоритмом роботи системи, визначають, яку кількість логічних функцій необхідно отримати з високим рівнем активності, а яку — з низьким. На четвертому кроці заданим логічним функціям присвоюють номери їх кон'юнкторів. На п'ятому кроці, використовуючи рекомендації із програмування §11.3 і дані §11.4, програмують утворені на четвертому кроці кон'юнктори логічних функцій, а також і самі функції. Невикористані вхідні входи ПЛМ повинні бути виключені у всіх використаних кон'юнкторах шляхом розплавлення їх ніхромових перемичок у матриці «*I*» у процесі програмування. На шостому кроці контролюють правильність програмування ПЛМ, і якщо воно відбулося правильно, то запрограмовану мікросхему використовують для реалізації заданих логічних функцій, а якщо ні, то необхідно повернутись до п'ятого кроку.

Приклад 11.5.1. Побудувати (реалізувати) на ПЛМ К556РТ1 таку систему логічних функцій

$$f_1 = x_2 \cdot \overline{x}_4 \cdot x_5 \vee \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \cdot \overline{x}_4 \vee x_1 \cdot \overline{x}_3 \cdot \overline{x}_5;$$

$$f_2 = x_1 \cdot \overline{x}_2 \cdot x_3 \vee x_2 \cdot \overline{x}_3 \cdot \overline{x}_4 \cdot x_5 \vee \overline{x}_1 \cdot x_2 \cdot x_4 \cdot \overline{x}_5;$$

$$f_3 = \overline{x}_1 \cdot \overline{x}_4 \cdot x_5 \vee x_4 \cdot \overline{x}_5 \vee x_2 \cdot \overline{x}_4 \cdot \overline{x}_5 \vee x_3 \cdot x_4 \cdot \overline{x}_5;$$

$$f_4 = x_3 \cdot \overline{x}_4 \cdot \overline{x}_5 \vee x_1 \cdot \overline{x}_2 \cdot x_3 \vee x_1 \cdot \overline{x}_4 \cdot x_5 \vee x_1 \cdot \overline{x}_2 \cdot \overline{x}_4 \cdot x_5;$$

із високим рівнем активності для перших двох функцій і низьким — для решти двох.

Розв'язання. Для реалізації цієї системи логічних функцій на ПЛМ, згідно із першим кроком, визначаємо необхідну кількість входів, виходів і кон'юнкцій — для програмування їх в мікросхемі К556РТ1. Кількість входів дорівнює 5, виходів — 4, а кон'юнкцій — 15. Так як задана ПЛМ має 16 входів, 8 виходів і може реалізувати 48 кон'юнкцій, §11.2, то робимо висновок, що на ній можливо за-

програмувати задану систему логічних рівнянь, що відповідає другому кроку.

Згідно із заданими функціями $f_1 \ldots f_4$, присвоюємо номера їх кон'юнкторів:

$k_1 = x_2 \cdot \overline{x}_4 \cdot x_5;$ $k_6 = x_2 \cdot \overline{x}_3 \cdot \overline{x}_4 \cdot x_5;$ $k_{11} = x_3 \cdot x_4 \cdot \overline{x}_5;$

$k_2 = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4;$ $k_7 = \overline{x}_1 \cdot x_2 \cdot x_4 \cdot \overline{x}_5;$ $k_{12} = x_3 \cdot \overline{x}_4 \cdot \overline{x}_5;$

$k_3 = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4;$ $k_8 = \overline{x}_1 \cdot \overline{x}_4 \cdot x_5;$ $k_{13} = x_1 \cdot \overline{x}_2 \cdot x_3;$

$k_4 = x_1 \cdot \overline{x}_3 \cdot \overline{x}_5;$ $k_9 = x_4 \cdot \overline{x}_5;$ $k_{14} = x_1 \cdot \overline{x}_4 \cdot x_5;$

$k_5 = x_1 \cdot \overline{x}_2 \cdot x_3;$ $k_{10} = x_2 \cdot \overline{x}_4 \cdot \overline{x}_5;$ $k_{15} = x_1 \cdot \overline{x}_2 \cdot \overline{x}_4 \cdot x_5;$

Використовуючи рекомендації із програмування, наведені в §11.3 і дані §11.4, програмуємо задані функції і їх дані заносимо в табл. 11.5.1.

Таблиця 11.5.1

Номер кон'юнктора	Кон'юнктори					Рівень активності			
	Вхідна змінна					1	1	0	0
	x_1	x_2	x_3	x_4	x_5	Вихідна функція			
	Номер програмованого входу					f_1	f_2	f_3	f_4
	A1	A2	A3	A4	A5	B1	B2	B3	B4
$k_1 = x_2 \cdot \overline{x}_4 \cdot x_5$	*	1	*	0	1	A	*	*	*
$k_2 = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4$	0	1	0	1	*	A	*	*	*
$k_3 = \overline{x}_1 \cdot x_2 \cdot \overline{x}_3 \cdot x_4$	1	0	1	0	*	A	*	*	*
$k_4 = x_1 \cdot \overline{x}_3 \cdot \overline{x}_5$	1	*	0	*	0	A	*	*	*
$k_5 = x_1 \cdot \overline{x}_2 \cdot x_3$	1	0	1	*	*	*	A	*	*
$k_6 = x_2 \cdot \overline{x}_3 \cdot \overline{x}_4 \cdot x_5$	*	1	0	0	1	*	A	*	*
$k_7 = \overline{x}_1 \cdot x_2 \cdot x_4 \cdot \overline{x}_5$	0	1	*	1	0	*	A	*	*
$k_8 = \overline{x}_1 \cdot \overline{x}_4 \cdot x_5$	0	*	*	0	1	*	*	A	*
$k_9 = x_4 \cdot \overline{x}_5$	*	*	*	1	0	*	*	A	*
$k_{10} = x_2 \cdot \overline{x}_4 \cdot \overline{x}_5$	*	1	*	0	0	*	*	A	*
$k_{11} = x_3 \cdot x_4 \cdot \overline{x}_5$	*	*	1	1	0	*	*	A	*
$k_{12} = x_3 \cdot \overline{x}_4 \cdot \overline{x}_5$	*	*	1	0	0	*	*	*	A
$k_{13} = x_1 \cdot \overline{x}_2 \cdot x_3$	1	0	1	*	*	*	*	*	A
$k_{14} = x_1 \cdot \overline{x}_4 \cdot x_5$	1	*	*	0	0	*	*	*	A
$k_{15} = x_1 \cdot \overline{x}_2 \cdot \overline{x}_4 \cdot x_5$	1	0	*	0	1	*	*	*	A

При програмуванні кон'юнкторів змінну x_i, яка входить в кон'юнкцію, прямим значенням програмують «1», якщо інверсним, то — «0», а якщо не входить зовсім, то — «*», що означає переплавлення перемички.

При програмуванні диз'юнкторів належність k_j до даної функції позначають у таблиці знаком «A», а його відсутність — знаком «*», що засвідчує перепалювання перемички.

Високий рівень активності виходу програмується «1», а низький — «0».

На всіх невикористаних входах А6...А16 і виходах В5...В8 ПЛМ із номерами кон'юнкторів К1...К15 перемички перепалюються.

Схема, яка реалізує систему рівнянь прикладу 11.5.1, приведена на рис. 11.5.1, а їх програмування в ПЛМ К556РТ1 — в табл. 11.5.1.

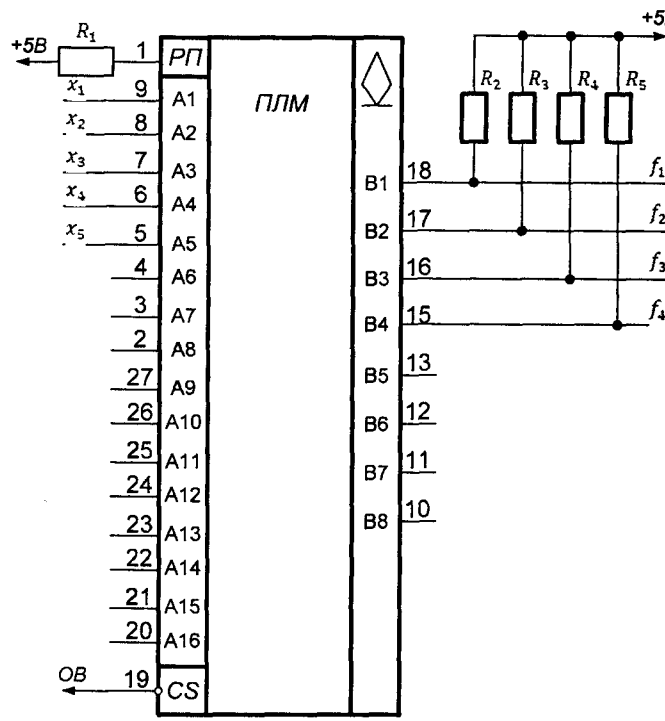


Рис. 11.5.1

де $R_1...R_5$ — резистор 1 кОм.

Як випливає із табл. 11.5.1 і рис. 11.5.1, реалізація будь-яких логічних функцій дуже зручна на ПЛМ.

Контрольні запитання

1. Яке призначення має програмована логічна матриця?
2. Які галузі застосування програмованої логічної матриці?
3. Для яких цілей застосовують програмовані логічні матриці?
4. З яких блоків складається програмована логічна матриця?
5. Яка максимальна кількість кон'юнкторів може бути запрограмована в ПЛІС К556РТ1?
6. Яка максимальна кількість логічних функції може бути запрограмована в ПЛІС К556РТ1?
7. Який процес відбувається при програмуванні в ПЛІС К556РТ1?
8. Які дві матриці є обов'язковими в базовій програмованій матриці, назвіть їх?
9. В якій формі можна реалізовувати логічні функції у базовій програмованій логічній матриці і чому?
10. Назвіть переваги побудови комбінаційних схем на програмованих логічних матрицях перед побудовою комбінаційних логічних схем на елементах елементарних булевих функцій.

Коментарі

В даному розділі використані матеріали галузевого стандарту [23].

Розділ 12

ЛОГІКА ПОБУДОВИ ТИПОВИХ СХЕМ ІЗ ПАМ'ЯТТЮ

12.1. Логіка побудови *RS*–тригерів

***Означення 12.1.1. RS*–тригером** називають запам'ятовуючий пристрій із двома стійкими станами і з різними інформаційними входами для установки його в стан «0» (R — вхід) і стан «1» (S — вхід). Назву *RS* — тригера утворено від перших букв слов *Reset* (виключити) і *Set* (включити).

Логіку побудови *RS*–тригера виконують за п'ять кроків. На першому кроці, досліджуючи логіку роботи *RS*–тригера, будують таблицю його переходів. На другому кроці, використовуючи таблицю переходів, будують карти Карно, а третьому, за допомогою карт Карно, отримують логічні рівняння роботи *RS*–тригера, які на четвертому кроці за допомогою законів алгебри логіки перетворюють на вигляд, на якому корисно його реалізувати, застосувавши один із видів основних логічних елементів, наведених у §8.1. На п'ятому кроці, за отриманим на четвертому кроці логічним рівнянням, будують *RS*–тригер на вибраній елементній базі.

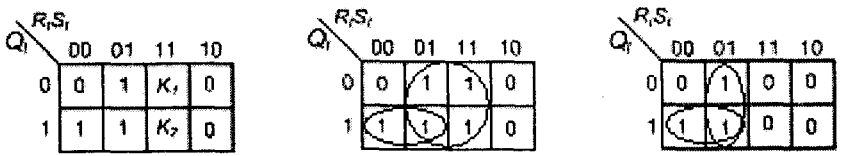
За першим кроком, досліджуючи логіку роботи *RS*–тригера, будуємо таблицю його переходів, табл. 12.1.1.

Таблиця 12.1.1

R_t	S_t	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	K_1
1	1	1	K_2

У таблиці переходів прийняті такі позначення: R_t, S_t, Q_t — значення логічних змінних у момент часу t на входах R, S і виході Q–тригера; Q_{t+1} — стан тригера після переключення; K_1, K_2 — невизначені коефіцієнти на тих наборах, де вхідні сигнали R_t і S_t одночасно приймають значення одиниці (заборонена комбінація сигналів).

На другому кроці, користуючись таблицею переходів *RS*–тригера, будуємо карти Карно, які приведені на рис. 12.1.1а



а) *б)* *в)*

Рис. 12.1.1

Припустимо, що комбінації вхідних сигналів $R_tS_t = 11$ не має, тоді одержимо карти Карно для $K_1 = K_2 = 1$, рис. 10.1.1б, і $K_1 = K_2 = 0$, рис. 12.1.1в.

На третьому кроці побудови *RS*–тригера із карт Карно, рис. 12.1.1б і рис. 12.1.1в, отримаємо логічні рівняння асинхронного *RS*–тригера

$$Q_{t+1} = S_t \vee \overline{R}_t Q_t \text{ при } K_1 = K_2 = 1 \qquad (12.1.1)$$

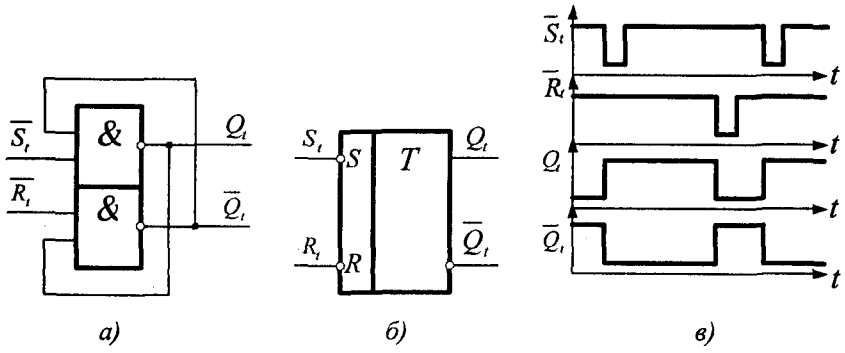
$$Q_{t+1} = \overline{R}_t (S_t \vee Q_t) \text{ при } K_1 = K_2 = 0 \qquad (12.1.2)$$

Логічні рівняння 12.1.1 і 12.1.2 визначають новий стан тригера Q_{t+1} залежно від старого стану Q_t і вхідних сигналів R_t і S_t.

На четвертому кроці побудови *RS*–тригера перетворюємо логічне рівняння 12.1.1 до вигляду, на якому корисного його реалізувати на елементах «*I-HI*»

$$Q_{t+1} = S_t \vee \overline{R_t} \cdot Q_t = \overline{\overline{S_t \vee \overline{R}_t \cdot Q_t}} = \overline{\overline{S_t} \cdot \overline{\overline{R_t} \cdot Q_t}}. \qquad (12.1.3)$$

Використовуючи логічні зв'язки рівняння 12.1.3, на п'ятому кроці будуємо схему синхронного *RS*–тригера на елементах «*I-HI*», яка матиме вигляд, наведений на рис. 12.1.2а.



а) *б)* *в)*

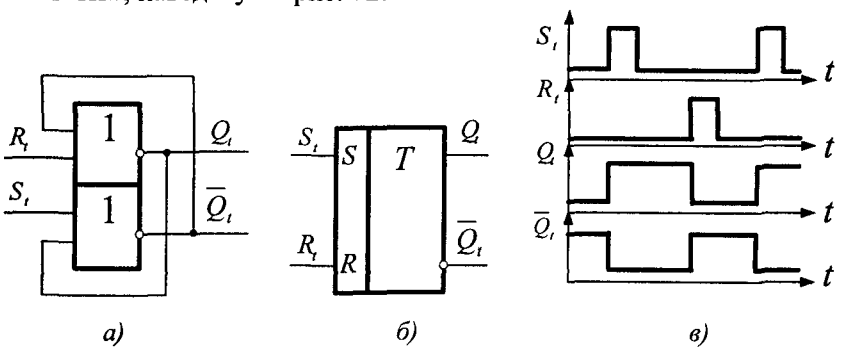
Рис. 12.1.2

Особливістю даного *RS*–тригера є інверсне управління на його інформаційних входах, що показано на часовій діаграмі роботи, рис. 12.1.2в. На рис. 12.1.2б приведена функціональна схема *RS*–тригера.

Для побудови *RS*–тригера на елементах «*АБО-НІ*» необхідно, згідно з кроком чотири, перетворити логічне рівняння 12.1.2 до такого виду

$$Q_{t+1} = \overline{R_t} \cdot (S_T \vee Q_t) = \overline{\overline{R_T} \cdot (S_t \vee Q_t)} = R_t \vee \overline{(S_t \vee Q_t)}. \qquad (12.1.4)$$

Тоді, використовуючи логічні зв'язки рівняння 12.1.4, на п'ятому кроці будуємо схему асинхронного *RS*–тригера на елементах «*АБО-НІ*», наведену на рис. 12.1.3а.



а) *б)* *в)*

Рис. 12.1.3

Часова діаграма роботи даного тригера приведена на рис. 12.1.3в, а зображення функціональної схеми на рис. 12.1.3б.

Для побудови синхронного *RS*–тригера необхідно в рівнянні 12.1.3 ввести змінну сигналу синхронізації C_t.

У логічному рівнянні 12.1.3 змінні S_t і R_t скомпонуємо із сигналом синхроімпульсу C_t таким чином: $C_t \cdot S_t$ і $C_t \cdot R_t$, у результаті чого отримаємо логічне рівняння

$$Q_{t+1} = \overline{\overline{C_t \cdot S_t} \cdot \overline{C_t \cdot R_t} \cdot Q_t}. \qquad (12.1.5)$$

Використовуючи рівняння 12.1.5 будуємо схему з логічними зв'язками синхронного *RS*–тригера на елементах «*I-HI*», яка приведена на рис. 12.1.4а.

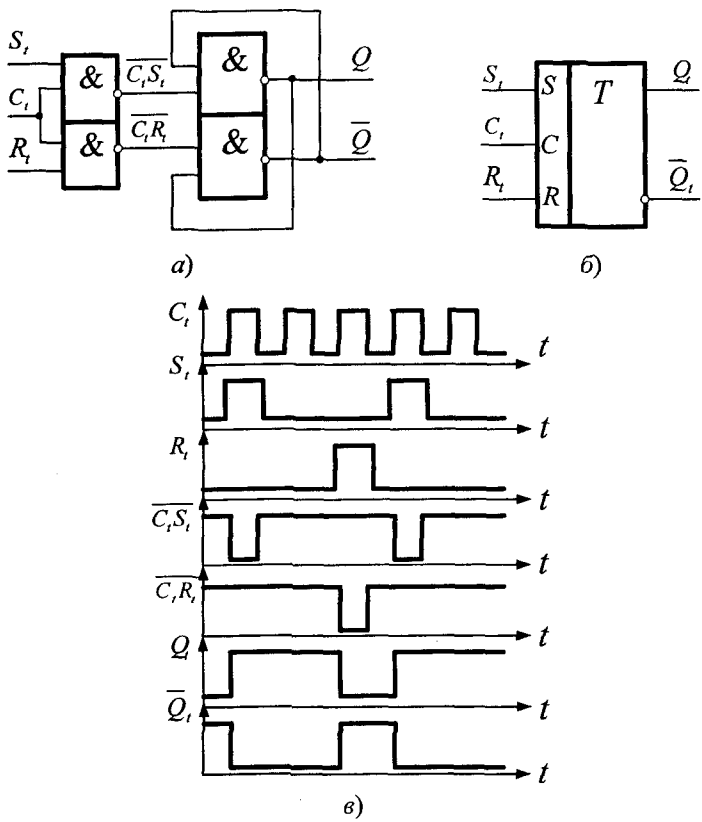


Рис. 12.1.4

Часова діаграма роботи синхронного *RS*–тригера приведена на рис. 12.1.4в, а функціональна схема — на рис. 12.1.4б. Із часової діаграми випливає, що комбінація сигналів $C_t = S_t = R_t = 1$ на вході тригера заборонена, бо веде до невизначеності його станів.

Для побудови синхронного *RS*–тригера на елементах «*АБО-НІ*» необхідно в логічному рівнянні 12.1.4 замінити змінні S_t і R_t на $\overline{C_t \cdot S_t}$ і $\overline{C_t \cdot R_t}$. В результаті цього отримаємо логічне рівняння роботи синхронного *RS*–тригера на елементах «*АБО-НІ*»

$$Q_{t+1} = \overline{\overline{C_t \cdot R_t} \vee (\overline{C_t \cdot S_t} \vee Q_t)} = \overline{\overline{C_t} \vee \overline{R_t} \vee (\overline{C_t} \vee \overline{S_t} \vee Q_t)}. \qquad (12.1.6)$$

Схема синхронного *RS*–тригера на елементах «*АБО-НІ*» з логічними зв'язками на основі логічного рівняння 12.1.6 наведена на рис. 12.1.5а.

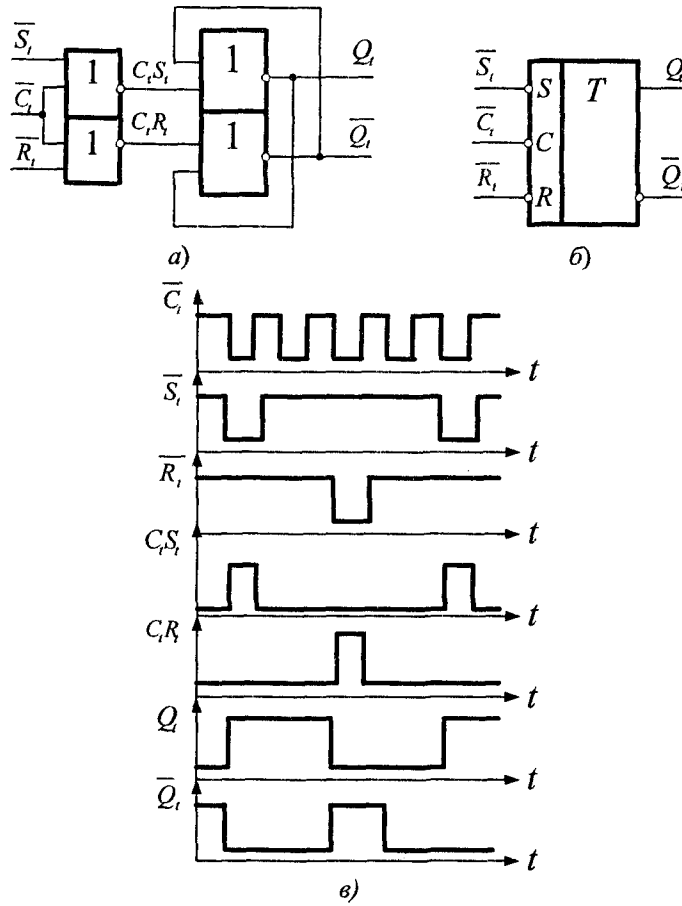


Рис. 12.1.5

Часова діаграма роботи синхронного *RS*–тригера приведена на рис. 12.1.5в, а функціональна схема на рис. 12.1.5б. Із часової діаграми випливає, що комбінація сигналів $C_t = S_t = R_t = 0$ на вході тригера заборонена, так як веде до невизначеності його станів.

12.2. Логіка побудови *D*–тригер

Означення 12.2.1. *D*–тригером називають синхронний запам'ятовуючий пристрій із двома стійкими станами і одним інформаційним *D*–входом.

Як випливає із означення 12.2.1, логіка функціонування *D*–тригера може бути взята з логіки функціонування синхронного *RS*–тригера, якщо в ньому вхід S замінити на вхід D, а вхід R — на $\overline{D}$. Тобто, синхронний *RS*–тригер можна зробити тригером з одним входом *D*. Для цього в рівнянні 12.1.5 замінимо змінні S на D і R на, $\overline{D}$, в результаті чого отримаємо

$$Q_{t+1} = \overline{\overline{C_t \cdot D_t} \cdot \overline{C_t \cdot \overline{D} \cdot Q_t}}. \qquad (12.2.1)$$

Звідси випливає, що логіка побудови *D*–тригера може бути взята з логіки побудови *RS*–тригера, у зв'язку з чим, використовуючи рівняння 12.2.1, будуємо схему з логічними зв'язками *D*–тригера на елементах «*І-НІ*», яка приведена на рис. 12.2.1а.

Часова діаграма роботи *D*–тригера приведена на рис. 12.2.1в, а функціональна схема — на рис. 12.2.1б. Із часової діаграми випливає, що *D*–тригер слідкує за зміною сигналу на *D*–вході в час дії сигналу синхронізації C_t і зберігає ту інформацію, яка була в момент його кінця.

З логіки функціонування *D*–тригера, рівняння 12.2.1, також випливає й інша схема *D*–тригера, яка побудована з використанням функціональної схеми синхронного *RS*–тригера й інвертора, рис. 12.2.2.

Часова діаграма роботи цієї схеми аналогічна часовій діаграмі, приведеній на рис. 12.2.1в.

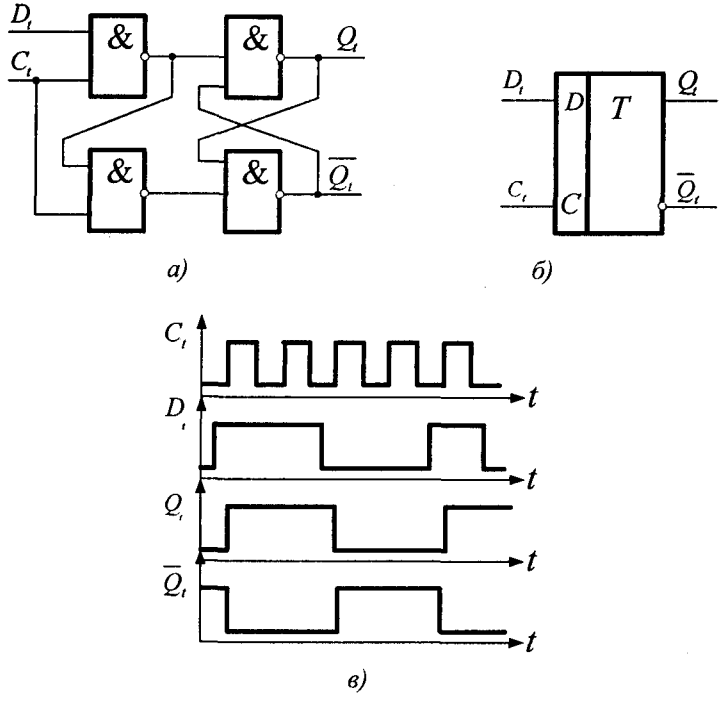


Рис. 12.2.1

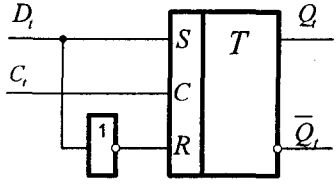


Рис. 12.2.2

12.3. Логіка побудови *T*–тригера

Означення 12.3.1. ***T*–тригером** називають запам'ятовуючий пристрій із двома стійкими станами, які міняються на протилежні після кожного надходження сигналу на його інформаційний вхід *T*.

Логіку побудови *T*–тригера виконують за чотири кроки. На першому кроці, досліджуючи його роботу, будують таблицю переходів. На другому — із таблиці переходів отримують логічні рівняння роботи *T*–тригера, які на третьому кроці за допомогою законів алгебри логіки надають йому вигляду, на якому корисно його реалізувати, застосовуючи один із видів основних логічних елементів, наведених у §8.1. На четвертому кроці, за отриманим логічним рівнянням, будують *T*–тригер на вибраній елементній базі.

Згідно з першим кроком, досліджуючи логіку функціонування *T*–тригера, будуємо таблицю його переходів, табл. 12.3.1.

Таблиця 12.3.1

T_t	Q_t	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

Із таблиці переходів випливає логічне рівняння роботи *T*–тригера, яке має вигляд

$$Q_{t+1} = \overline{T_t} \cdot Q_t \vee T_t \cdot \overline{Q_t}. \qquad (12.3.1)$$

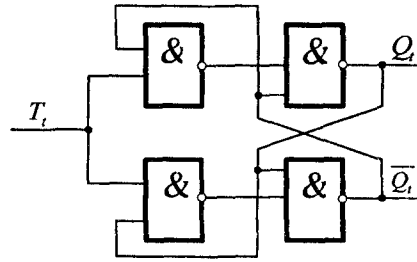
На третьому кроці побудови *T*–тригера, використовуючи закони алгебри логіки (подвійного заперечення і закон де Моргана), перетворюємо рівняння 12.3.1 до вигляду

$$Q_{t+1} = \overline{T_t} \cdot Q_t \vee T_t \cdot \overline{Q_t} = \overline{\overline{T_t} \cdot Q_t \cdot T_t \cdot \overline{Q_t}}. \qquad (12.3.2)$$

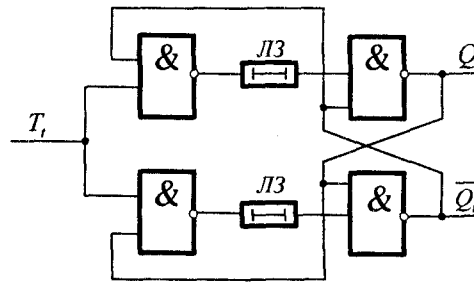
В рівнянні 12.3.2 для виключення інверсії сигналу T_t використовуємо логічну тотожність $\overline{T_t} \cdot Q_t = (\overline{T_t \cdot Q_t}) \cdot Q_t$. Підставивши її в рівняння 12.3.2, отримаємо

$$Q_{t+1} = \overline{\overline{T_t \cdot Q_t} \cdot Q_t \cdot T_t \cdot \overline{Q_t}}. \qquad (12.3.3)$$

На четвертому кроці будуємо схему *T*–тригера на елементах «*І-НІ*» із використанням рівняння 12.3.3, рис. 12.3.1а.



а)

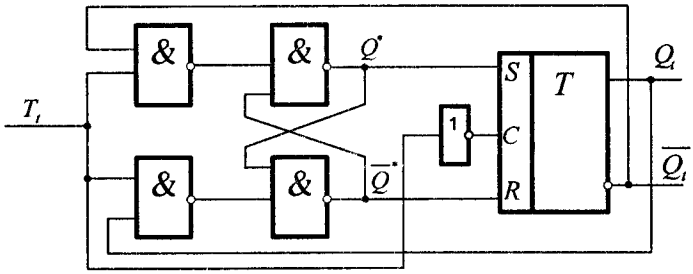


б)

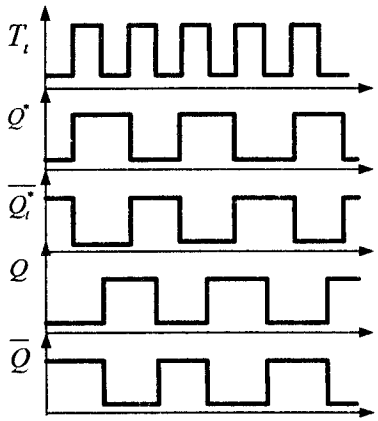
Рис. 12.3.1

Із аналізу роботи схеми *Т*–тригера (рис. 12.3.1а) випливає, що сигнали оберненого зв'язку в тригері міняються в час дії сигналу T_t. Тобто робота елемента пам'яті *Т*–тригера не є стійкою (може виникнути коливальний режим). Уникнути цього явища можна двома шляхами. Перший шлях полягає у затримці сигналів, які надходять на елемент пам'яті *Т*–тригера, на час дії сигналу T_t використовуючи для цього лінію затримки (*ЛЗ*), а при другому необхідно застосовувати додатковий асинхронний *RS*–тригер. Схема *Т*–тригера, яка реалізує перший шлях, приведена на рис. 12.3.1б, а другий — на рис. 12.3.2а.

Схему, приведену на рис. 12.3.2а, називають схемою двохступінчатого асинхронного *Т*–тригера на елементах «*І-НІ*» з логічними зв'язками відповідно рівняння 12.3.3. На рис. 12.3.2б приведена часова діаграма роботи даної схеми, яка є стійкою в роботі і виключає різні «*гонки*» в схемі за рахунок затримки сигналів на її елементах.



а)



б)

Рис. 12.3.2

12.4. Логіка побудови *JK*–тригера

Означення 12.4.1. ***JK*–тригером** називають запам'ятовуючий пристрій із двома стійкими станами й інформаційними входами J (аналог S), K (аналог R), які забезпечують незалежну установку станів «1» і «0», а при збігу сигналів $JK = \{1, 0\}$ він переключається в протилежні стани, тобто реалізує додавання сигналів за модулем два.

Логіку побудови *JK*–тригера виконують за чотири кроки. На першому кроці, досліджуючи логіку *JK*–тригера, будують таблицю його переходів. На другому кроці за таблицею переходів отриму-

ють карти Карно, за якими на третьому отримують логічне рівняння роботи *JK*–тригера, а на четвертому за отриманим логічним рівнянням будують *JK*–тригер на вибраній елементній базі.

Згідно з першим кроком, досліджуючи логіку роботи *JK*–тригера, будуємо таблицю його переходів, табл. 12.4.1.

Таблиця 12.4.1

K_t	J_t	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

В таблиці переходів прийняті такі позначення: K_t, J_t, Q_t — значення логічних змінних у момент часу t на входах K_t, J_t і виході Q_t; Q_{t+1} — стан тригера після переключення.

На другому кроці, користуючись таблицею переходів *JK*–тригера, будуємо карту Карно, рис. 12.4.1.

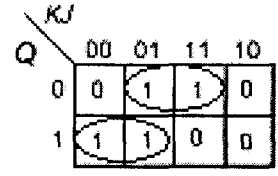


Рис. 12.4.1

Із карти Карно отримаємо логічне рівняння роботи *JK*–тригера

$$Q_{t+1} = \overline{K_t} \cdot Q_t \vee J_t \cdot \overline{Q_t}, \qquad (12.4.1)$$

яке за законами алгебри логіки (закон подвійного заперечення і закон де Моргана) перетворимо на таке:

$$Q_{t+1} = \overline{K_t} \cdot Q_t \vee J_t \cdot \overline{Q_t} = \overline{\overline{K_t} \cdot Q_t \vee J_t \cdot \overline{Q}}. \qquad (12.4.2)$$

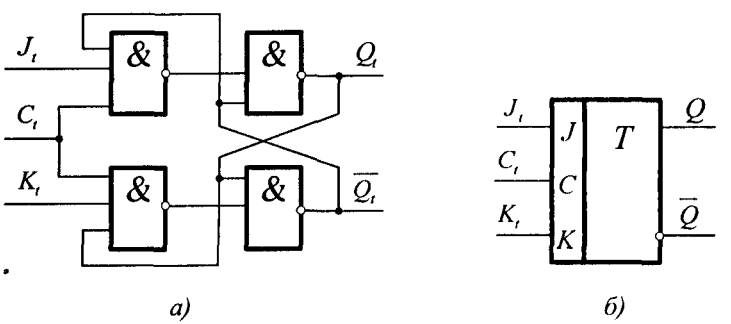
Для виключення заперечення сигналу K_t у рівнянні 12.4.2 використовуємо логічну тотожність $\overline{K_t} \cdot Q_t = \overline{(K_t \cdot Q_t)} \cdot Q_t$. Підставивши її в рівняння 12.4.2, отримаємо

$$Q_{t+1} = \overline{\overline{(K_t \cdot Q_t)} \cdot Q_t \cdot J_t \cdot \overline{Q_t}}. \qquad (12.4.3)$$

Для побудови синхронного *JK*–тригера на елементах «*І-НІ*» необхідно замінити в рівнянні 12.4.3 змінні K_t і J_t на $C_t \cdot K_t$ і $C_t \cdot J_t$ відповідно. В результаті цього отримаємо

$$Q_{t+1} = \overline{\overline{(C_t \cdot K_t \cdot Q_t)} \cdot Q_t \cdot \overline{C_t \cdot J_t \cdot \overline{Q_t}}}. \qquad (12.4.4)$$

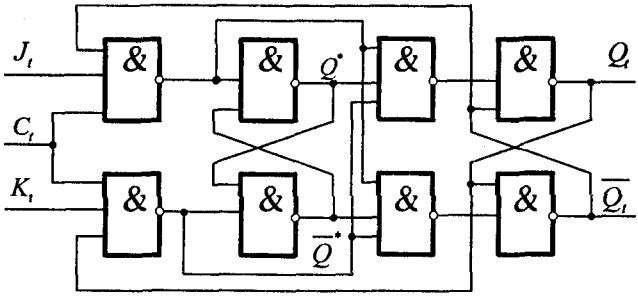
На четвертому кроці, використовуючи логічні зв'язки рівняння 12.4.4, будуємо одноступінчатий асинхронний *JK*–тригер, який приведений на рис. 12.4.2а.



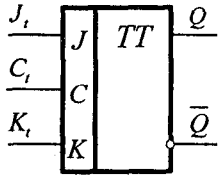
а) *б)*

Рис. 12.4.2

На рис. 12.4.2б приведена функціональна схема одноступінчатого *JK*–тригера. Дана схема працює надійно в режимі роз'єднаних входів *J* і *K*, тобто в режимі *RS*–тригера, а при їх об'єднанні, тобто в режимі *T*–тригера, необхідно використати один із двох шляхів ліквідації «*гонок*» сигналів, які були розглянуті в §12.3. На практиці, як правило, застосовують двохступінчаті синхронні *JK*–тригери. Схема такого тригера із застосуванням рівняння 12.4.4 в кожному ступені приведена на рис. 12.4.3а,



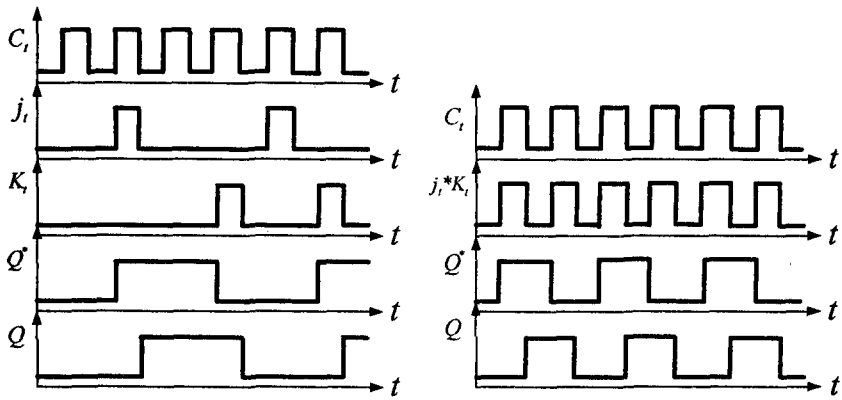
а)



б)

Рис. 12.4.3

його функціональна схема — на рис. 12.4.3б, а часова діаграма роботи в режимі *RS*–тригера — на рис. 12.4.4а.



а) *б)*

Рис. 12.4.4

На рис. 12.4.4б наведена часова діаграма роботи двохступінчатого *JK*–тригера при умові, коли входи J_t, і K_t — об'єднані, тобто *JK*–тригер уключений у лічильному режимі, режимі роботи *T*–тригера.

12.5. Логіка побудови лічильників

Означення 12.5.1. **Лічильником** називають функціональний пристрій комп'ютера, призначений для підрахунку вхідних імпульсів.

Лічильники являють собою зв'язаний ланцюг *T*–тригерів, що утворюють пам'ять із заданою кількістю сталих станів, рис. 12.5.1.

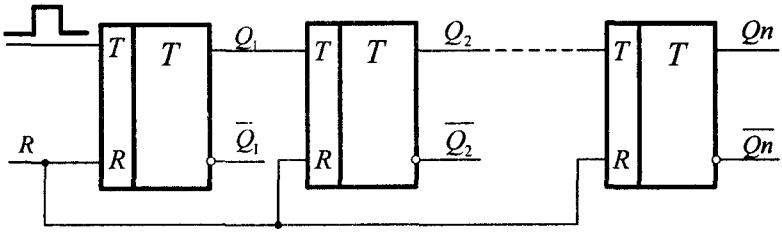


Рис. 12.5.1

Розрядність лічильника *n* дорівнює числу *T*–тригерів. Кожний вхідний імпульс змінює стан лічильника, який зберігається до надходження наступного сигналу. Значення виходів тригерів лічильника Q_n, Q_{n-1}, …, Q_1 відображають результат лічби в прийнятій системі числення. Логічну функцію лічильника позначають буквами *CT* (counter). До мікрооперацій лічильника включають попередню установку в початковий стан, інкремент або декремент слова, що зберігається, видачу слова паралельним кодом та ін.

Вхідні імпульси можуть надходити на лічильник як періодично, так і довільно розміщені в часі.

Лічильник є одним із основних функціональних пристроїв комп'ютера і застосовується в утворенні послідовності адреси команд програми, підрахунку числа циклів при виконанні операцій ділення, множення, зсуву, а також отримання сигналів мікрооперацій і синхронізації.

Лічильники характеризуються модулем і ємністю лічення. Модуль лічення $K_{сч}$ визначає кількість станів лічильника. Модуль двійкового *n*–розрядного лічильника визначається цілим степенем

двійки $M = 2^n$. У лічильниках інших типів справедлива нерівність $K_{сч} \le M$. Після лічення числа імпульсів $N_{вх} = K_{сч}$ лічильник повертається в початковий стан. Таким чином, модуль лічення, який часто називають коефіцієнтом перерахунку, визначає цикл роботи лічильника, після якого його стан повторюється. Тому число вхідних імпульсів і стан лічильника незаперечно визначені тільки для першого циклу.

Ємність лічення N_{max} визначається максимальною кількістю вхідних імпульсів, які може зафіксувати лічильник при одному циклі роботи.

В лічильниках використовуються три режими роботи: управління, накопичення і ділення. При управлінні зчитування інформації відбувається після кожного вхідного зліченого імпульсу. В режимі накопичення головним є підрахунок заданого числа імпульсів, а в режимі ділення — зменшення частоти надходження імпульсів, наприклад, у $K_{сч}$ раз.

Лічильники за логікою побудови класифікують за такими ознаками:

а) способом кодування — позиційні і непозиційні;

б) модулем лічення — двійкові, десяткові, із довільним постійним або змінним модулем;

в) напрямком лічення — прості (сумуючі, віднімаючі) та реверсивні;

г) способом організації міжрозрядних зв'язків — із послідовним, крізним, паралельним і комбінованим переносами;

д) типом використаних тригерів — T, JK, D в лічильному режимі.

У лічильниках із позиційним кодуванням числове значення поточного стану лічильника визначається за формулою

$$N = \sum_{i=1}^{n} a_i \cdot Q_i = a_n \cdot Q_n + a_{n-1} \cdot Q_{n-1} + \ldots + a_i \cdot Q_i,$$

де a_i — вага і-го розряду; Q_i — значення виходу i-го розряду; n — число розрядів.

Прості лічильники за видом переходів розділяються на підсумуючі і віднімаючі. Граф переходів підсумуючого лічильника приведений на рис. 12.5.2а, віднімаючого — на рис. 12.5.2б, а реверсивного — на рис. 12.5.2в.

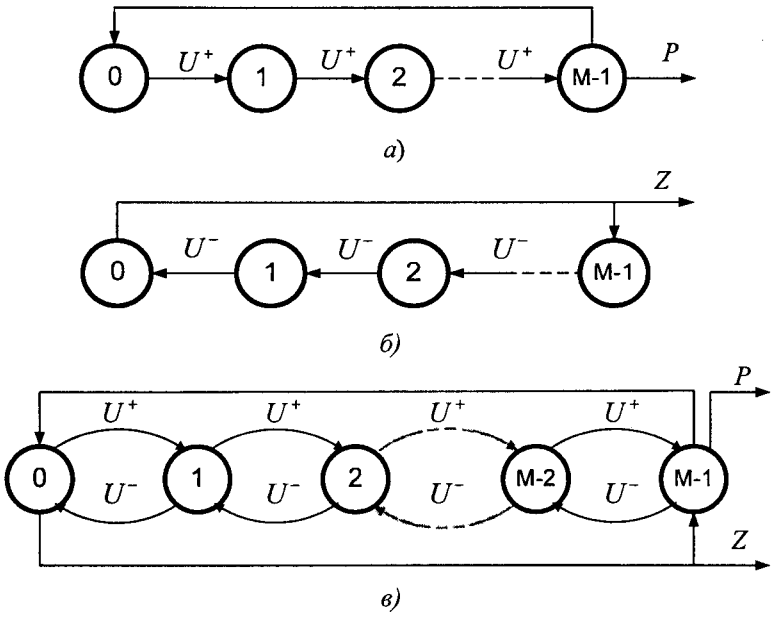


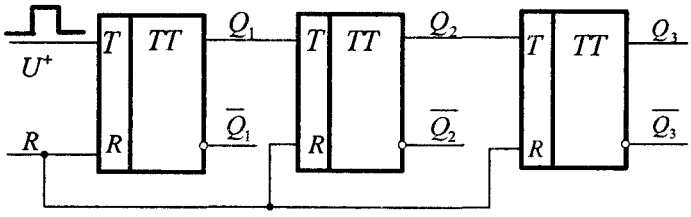
Рис. 12.5.2

Реверсивні лічильники мають переходи в прямому і оберненому напрямках, що дозволяє рахувати додавані і віднімані імпульси, рис. 12.5.2в. В процесі лічення повинна виконуватись умова $\sum U^+ + N_n \geq \sum U^-$, де N_n — попередньо записане число. За поточним станом виходів лічильника визначають результат реверсивного лічильника

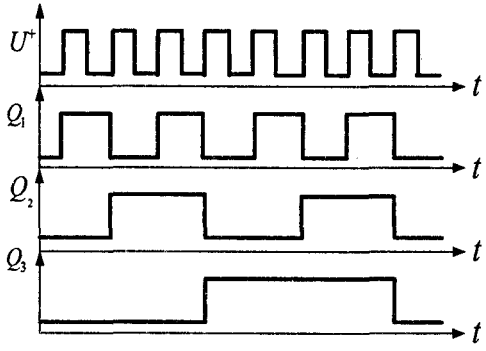
$$\Delta N = \sum U^+ + N_n \geq \sum U^-.$$

Використовуючи рис. 12.5.2, розглянемо побудову асинхронних двійкових підсумуючих, віднімаючих і реверсивних лічильників із використанням *T*–тригерів.

Асинхронні підсумуючі лічильники на двохступінчатих *T*–тригерах будують таким чином, щоб вхідні імпульси U^+ надходили тільки на лічильний вхід першого розряду. Сигнали переносу в лічильнику передаються асинхронно з прямих виходів молодших розрядів на *T*–входи сусідніх старших. Схема такого лічильника на три розряди приведена на рис. 12.5.3а, а часова діаграма його роботи — на рис. 12.5.3б.



а)



б)

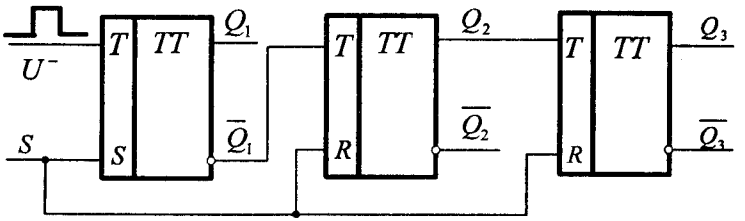
Рис. 12.5.3

Асинхронні віднімаючі лічильники на двохступінчатих *T*–тригерах будують аналогічно підсумуючим, але з тою різницею, що сигнали переносу в лічильнику передаються асинхронно з інверсних виходів молодших розрядів на *T*–входи сусідніх старших. Схема такого лічильника на три розряди приведена на рис. 12.5.4а, а часова діаграма його роботи — на рис. 12.5.4б.

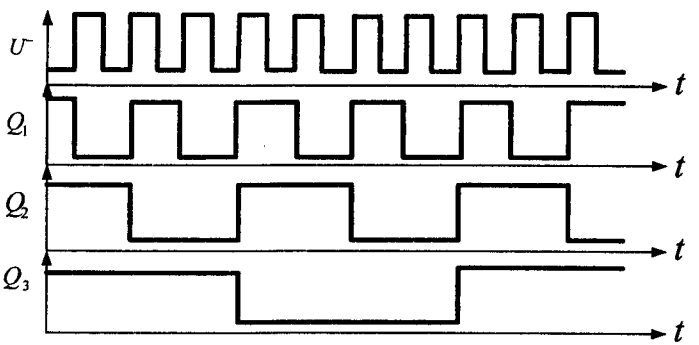
Реверсивний лічильник повинен суміщати принципи побудови підсумуючого і віднімаючого лічильників. Схема його повинна задовольняти переключення із режиму додавання в режим віднімання і мати окремі установки лічильника в «0» і «1». Схема на три розряди, яка відповідає цим принципам, приведена на рис. 12.5.5.

У даній схемі міжрозрядні зв'язки комутуються за допомогою логічних елементів «*2І — АБО*». На виході цих логічних елементів виробляється сигнал T_i для лічильних входів старших розрядів.

$$T_i = Y_+ \cdot Q_i \vee Y_- \cdot \overline{Q}_i,\ i = 1, 2, 3, \ldots, n.$$



а)



б)

Рис. 12.5.4

Тобто, якщо управляючий *RS*–тригер міститься у стані «1», то лічильник реалізує режим підрахунку вхідних імпульсів (додавання), а в протилежному випадку — забезпечує режим віднімання. В обох режимах роботи *T*–тригери переключаються асинхронно, а їх робота описується часовими діаграмами, приведеними на рис. 12.5.3б і рис. 12.5.4б.

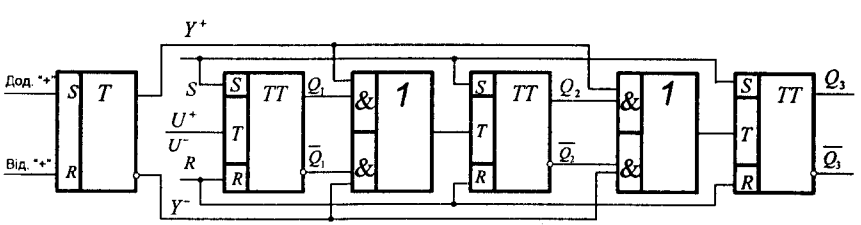


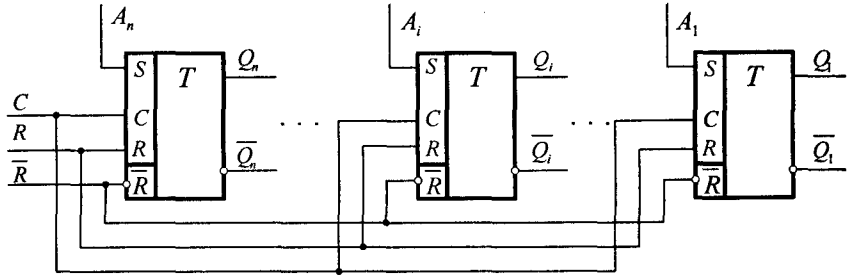
Рис. 12.5.5

12.6. Логіка побудови регістрів

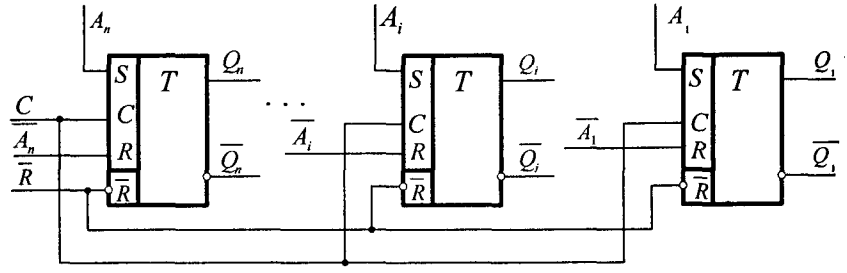
Означення 12.6.1. Регістром називають функціональний пристрій комп'ютера, призначений для приймання, тимчасового зберігання, перетворення і видачі n-розрядного двійкового коду.

Регістр включає регулярний набір однотипних тригерів, у кожному із яких зберігається один біт інформації. Для регістрів найчастіше використовують *RS*, *D* і *JK*–тригери.

Регістр, який призначений тільки для приймання, зберігання і передачі інформації, називають елементарним. Регістр, в якому зберігання даних суміщається з мікрооперaціями зсуву, називають зсувовим. Регістри позначають буквами *RG* (*register*). Побудова елементарних регістрів з однофазним і парафазним способом запису інформації на *RS*–тригерах приведена на рис. 12.6.1а і 12.6.1б відповідно.



а)



б)

Рис. 12.6.1

Як випливає із рис. 12.6.1а, при однофазному запису інформації значення кожного слова $A_n...A_i...A_1$ подається по одній лінії зв'язку на входи S і RS–тригерів, а після зчитування записаної інформації регістри повинні обнулитися за допомогою спільного $\overline{R}$–входу. При парафазному способі запису значення кожного розряду слова передається за двома лініями зв'язку: пряме значення A_i надходить на вхід S, а інверсне A_i — на вхід R, рис. 12.6.1б. В цьому випадку не потрібна попередня установка регістра в стан «0», що дає можливість збільшити швидкість роботи регістра.

Зсувові регістри використовують у процесі виконання команд множення, ділення, нормалізації чисел, перетворення паралельного коду в послідовний і навпаки. Зсувові регістри проектують на двохступінчатих RS, JK або D–тригерах. На рис. 12.6.2 приведена побудова регістра зсуву, що складається з n послідовно з'єднаних D–тригерів, функції збудження яких мають вигляд

$$D_1 = A,\ D_i = Q_{i-1},\ i = 2, 3, ..., n \qquad (12.6.1)$$

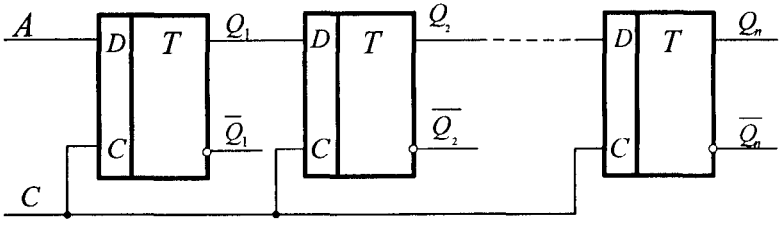


Рис. 12.6.2

Із співвідношення 12.6.1 випливає, що інформація яка зберігається в деякому такті в тригері Q_{i-1}, під дією імпульсу синхронізації передається в наступному такті в тригер Q_i, тобто відбувається зсув інформації від тригера до тригера.

Приклад побудови реверсивного трьох розрядного регістру зсуву на D — тригерах з динамічним управлінням приведена на рис. 12.6.3.

Реверсивний регістр зсуву працює таким чином. При значенні сигналу $Y_{зп} = 1$ в регістр записується інформація паралельним однофазним кодом ($A_3\ A_2\ A_1$). При значенні сигналу $R_n = 1$ інформація, що зберігається одночасно рухається в бік молодшого розряду, при цьому розряд тригера Q_3 обнуляється. При значенні сигналу

$L_л = 1$ інформація в регістрі одночасно рухається у бік старших розрядів, при цьому розряд Q_1 обнуляється. Запис і зсув інформації відбувається за переднім фронтом імпульсу (динамічний вхід C D–тригера).

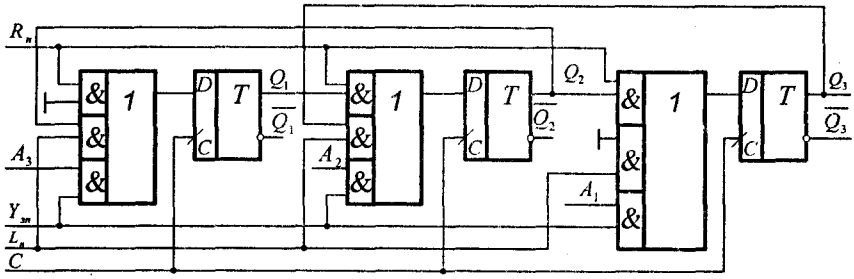


Рис. 12.6.3

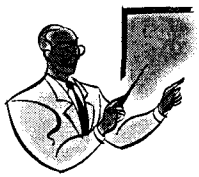
Контрольні запитання

1. Що називають *RS*–тригером?
2. Скільки етапів має логіка побудови *RS*–тригера?
3. Сформулюйте етапи логіки побудови *RS*–тригера.
4. Яка різниця між синхронним і асинхронним *RS*–тригером?
5. Де застосовують *RS*–тригери?
6. Який тригер називають *D*–тригером?
7. Яка різниця між *RS* і *D*–тригером?
8. Скільки етапів має логіка побудови *D*–тригера? Назвіть їх.
9. Де застосовують *D*–тригери?
10. Що називають *T*–тригером?
11. В чому полягає різниця між *RS*, *D* і *T*–тригером?
12. В чому полягає логіка побудови *T*–тригера?
13. Де застосовують *T*–тригер?
14. Які бувають *T*–тригери?
15. Який тригер називають *JK*–тригером?
16. Сформулюйте етапи логіки побудови *JK*–тригера.
17. Які бувають *JK*–тригери?
18. Чому двохступінчатий *JK*–тригер знайшов більше застосування, ніж одноступінчатий?
19. Що називають лічильником?
20. На яких тригерах проектують лічильники?

21. Які є лічильники і чим вони характеризуються?
22. За якими ознаками класифікують лічильники?
23. Що розуміють під модулем і ємністю лічильника?
24. Який лічильник називають реверсивним?
25. Які лічильники називають підсумуючими і віднімаючими?
26. Що називають регістром?
27. Які є регістри?
28. Для яких цілей застосовують регістри?
29. Яка різниця між однофазною і парафазною формами запису інформації в регістр?
30. За якою формою запису інформації в регістр його швидкість збільшується?

Коментарі

В даному розділі для логіки побудови *RS*, *D*, *T* і *JK* — тригерів використані матеріали літератури [3, 16, 22], а логіка побудови лічильників і регістрів випливає з [26, 27, 28, 29].



Розділ 13

ЛОГІКА ПОБУДОВИ КОМП'ЮТЕРНИХ СХЕМ

13.1. Логіка побудови одновихідних комбінаційних схем на елементах логіки Буля

Означення 13.1.1. **Комбінаційною** називають схему, вихідний сигнал якої визначається вхідним набором її змінних у даний момент часу.

Такі схеми «*не пам'ятають*» значення вхідних наборів у попередні моменти часу, а тому їх нерідко називають автоматами без пам'яті або однотактними схемами. При наявності одного виходу комбінаційні схеми описуються одним рівнянням (функцією) виду

$$y = f(x_1, x_2, ..., x_n) \qquad (13.1.1)$$

де x_i та y можуть прийняти лише значення логічного *"0"* або логічної «1».

Рівняння 13.1.1 отримують, як правило, за допомогою таблиці істинності, § 2.2, а форму цього рівняння називають ДНФ (ДДНФ), КНФ (ДКНФ), розділ 4. Інколи таке рівняння можна отримати на основі логіки роботи об'єкта.

За отриманим тим чи іншим способом рівнянням необхідно розробити схему, яка б реалізувала даний алгоритм. Методи проектування залежать від того, на якій елементній базі буде реалізована дана схема. Тому логіка розробки комбінаційних схем на елементах логіки Буля має такі кроки.

1. За допомогою таблиці істинності або таблиці функціонування, або іншим способом описують алгоритм роботи необхідної схеми управління.

2. З таблиці функціонування або таблиці істинності знаходять логічне рівняння для схеми управління.

3. Виконують мінімізацію логічного рівняння одним із методів, наведених у розділі 5.

4. При необхідності мінімізоване логічне рівняння перетворюють до певного базису елементів, на якому буде відбуватися побудова схеми.

5. Вибирають логічні елементи і будують принципову схему, яка відтворює отримане логічне рівняння.

Логіку побудови одновихідної комбінаційної схеми розглянемо на конкретних прикладах.

Приклад 13.1.1. В базисі Шеффера побудувати одновихідну комбінаційну схему, робота якої описується такою таблицею істинності, табл. 13.1.1.

Таблиця 13.1.1

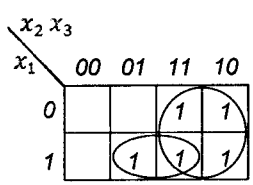
x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1

x_1	x_2	x_3	y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Розв'язання. Використовуючи табл. 13.1.1, знаходимо логічне рівняння роботи схеми в ДДНФ, крок 2

$$y = \overline{x_1} \cdot x_2 \cdot \overline{x_3} \vee \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3 \cdot \vee x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot x_3.$$

На третьому кроці виконуємо методом Карно мінімізацію отриманого логічного рівняння



$$y = x_2 \vee x_1 \cdot x_3. \qquad (13.1.2)$$

Отримане рівняння 13.1.2 перетворюємо в базис Шеффера, крок 4

$$y = x_2 \vee x_1 \cdot x_3 = \overline{\overline{x_2} \cdot \overline{x_1 \cdot x_3}}. \qquad (13.1.3)$$

Користуючись рівнянням 13.1.3, а також елементами «*І-НІ*», які відповідають базису Шеффера, будуємо одновихідну комбінаційну схему, рис. 13.1.1, крок 5

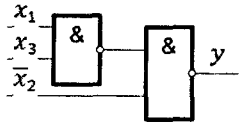


Рис. 13.1.1

Примітка. Вибір логічних елементів означає, перш за все, вибір серії мікросхем. Існують два основних класи мікросхем. Це мікросхеми на основі транзисторно-транзистерної логіки (ТТЛ) і мікросхеми на основі МОП-структури.

Реальні елементи обох класів мають також обмежену кількість входів, часову затримку і відповідні коефіцієнти розгалуження. При проектуванні схем конкретні параметри елементів в даному підручнику не розглядуються.

Приклад 13.1.2. В базисі Пірса побудувати одновихідну комбінаційну схему автомата, логіка роботи якого описується такою функцією в ДНФ

$$y = x_1 \cdot \bar{x}_2 \vee x_3 \cdot x_4 \vee \bar{x}_1 \cdot x_2 \cdot x_3. \qquad (13.1.4)$$

Розв'язання. Мінімізацію заданої функції виконати неможливо, так як тоді два із трьох мінтерми (елементарні кон'юнкції) не відрізняються один від одного тільки однією змінною, тобто при винесені за дужки загальних змінних неможливо отримати вираз $x \vee \bar{x} = 1$. Тому, користуючись кроком чотири, перетворимо рівняння 13.1.4 в базис Пірса, в результаті чого отримаємо

$$y = \overline{\overline{\bar{x}_2 \vee x_2} \vee \overline{\bar{x}_3 \vee \bar{x}_4} \vee \overline{x_1 \vee \bar{x}_2 \vee \bar{x}_3}}. \qquad (13.1.5)$$

Вибір логічних елементів необхідно робити так само, як і у прикладі 13.1.1. Для побудови схеми в базисі Пірса використаємо рівняння 13.1.5, логічні зв'язки якого реалізуємо на елементах «*АБО-НІ*», рис. 13.1.2.

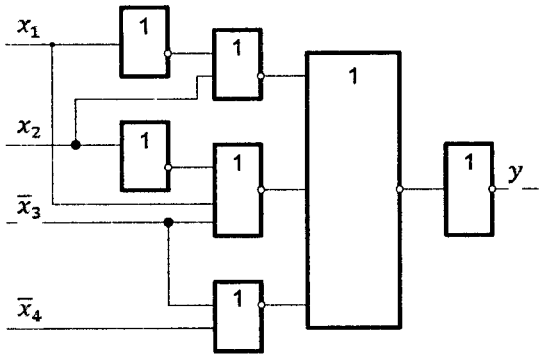


Рис. 13.1.2

Приклад 13.1.3. В базисі елементів «*І*», «*АБО*», «*НІ*» побудувати одновихідну комбінаційну схему, робота якої описується такою таблицею істинності, табл. 13.1.2

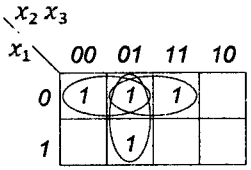
Таблиця 13.1.2

x_1	x_2	x_3	y	x_1	x_2	x_3	y
0	0	0	1	1	0	0	0
0	0	1	1	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	0

Розв'язання. Використовуючи табл. 13.1.2, знаходимо логічне рівняння роботи схеми в ДДНФ

$$y = \overline{x}_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 x_3 \vee x_1 \cdot \overline{x}_2 \cdot x_3.$$

Виконаємо мінімізацію отриманого логічного рівняння методом Карно



в результаті чого одержимо

$$y = \overline{x}_1 \cdot \overline{x}_2 \vee \overline{x}_1 \cdot x_3 \vee \overline{x}_2 \cdot x_3. \qquad (13.1.6)$$

Вибір логічних елементів необхідно робити так само, як і у прикладі 13.1.1. Для побудови схеми використаємо рівняння 13.1.6, логічні зв'язки якого реалізуємо за допомогою елементів «*І*», «*АБО*», «*НІ*», рис. 13.1.3.

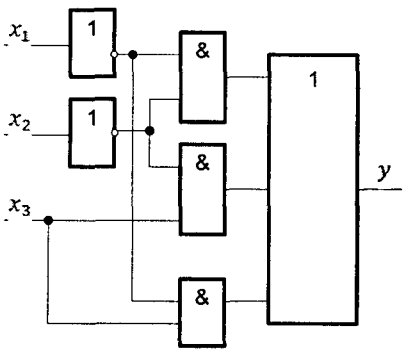


Рис. 13.1.3

13.2. Логіка побудови одновихідних комбінаційних схем на мультиплексорах

У § 10.3 було показано, що в мультиплексорі номер набору на управляючому вході дорівнює номеру інформаційного входу x_i, який підключений до виходу D. Тоді загальне рівняння роботи мультиплексора матиме такий вигляд:

$$D = \bigvee_{i=0}^{2^n} A_i \cdot b_i, \qquad (13.2.1)$$

де A_i — номер вхідного набору на управляючих входах;

b_i — змінна на i-му інформаційному вході;

D — вихідне значення функції мультиплексора.

В подальшому вихідне значення функції мультиплексора D будемо позначати через y.

Із рівняння 13.2.1 випливає, що мультиплексор розкладає логічну функцію за n змінними. Подібні властивості мультиплексорів і використовують при синтезі комбінаційних схем.

Логіка побудови комбінаційних схем на мультиплексорах залежить від кількості змінних у функції та від кількості управляючих входів мультиплексора. Тому є декілька варіантів логіки побудови комбінаційних схем на мультиплексорах.

Варіант перший. У даному варіанті кількість змінних у функції m дорівнює кількості управляючих входів n. При цьому розклад функції з m змінними за n змінними дає мінтерми, які можуть бути нульовими або одиночними. Звідси випливає, що на входи мультиплексора необхідно подати або логічний нуль, або логічну одиницю. Оскільки розглядувана логічна функція є сумою одиничних мінтермів, то числа в ній є номерами тих інформаційних входів b_i, на які необхідно подати логічну одиницю, а решту входів з'єднати з логічним нулем. У такому випадку вхідні змінні x_i необхідно подати на управляючі входи мультиплексора.

Приклад 13.2.1. Реалізувати логічну функцію

$$y = x_1 \cdot \overline{x}_2 \cdot \overline{x}_3 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot x_3 \vee \overline{x}_1 \cdot x_2 \cdot \overline{x}_3$$

на мультиплексорі.

Розв'язання. У даній функції кількість змінних $m = 3$, тому необхідно застосувати мультиплексор для її реалізації з кількістю управляючих входів $n = 3$.

Із логічного рівняння прикладу знаходимо десяткові числа, яким відповідають мінтерми в функції $y = \vee(1, 2, 4)$. Тобто, на інформаційні входи 1, 2, 4 мультиплексора необхідно подати логічні одиниці, а на решту — нулі.

Тоді схема реалізації заданої функції на мультиплексорі при подачі на його управляючі входи змінних x_1, x_2, x_3, матиме вигляд, приведений на рис. 13.2.1.

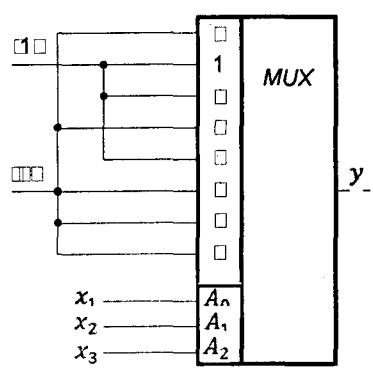


Рис. 13.2.1

Варіант другий. У даному варіанті кількість змінних $m = n + 1$.

Це говорить за те, що кількість змінних у функції більша за кількість управляючих входів мультиплексора. В даному випадку задану функцію можна розкласти за будь-якою змінною, але найбільш доцільно зробити це або за старшою, або за молодшою змінною. Якщо функцію розкласти за старшою змінною, то будемо мати

$$f(x_1, x_2, ..., x_m) = \overline{x_1} \cdot f_1(0, x_2, ..., x_m) \vee x_1 \cdot f_2(1, x_2, ..., x_m).$$

Якщо функції f_1 і f_2 мають однакові мінтерми, то це означає, що на відповідний b_i вхід мультиплексора необхідно подати логічну одиницю.

Решта мінтермів у функції f_1 є номери інформаційних входів, на які подають $\overline{x_1}$. Інші мінтерми в функції f_2 є номерами інформаційних входів, на які подають x_1. На решту входів мультиплексора подають логічний нуль, а на управляючі входи мультиплексора — змінні $x_2, ..., x_m$.

Приклад 13.2.2. Реалізувати логічну функцію

$$y = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \cdot x_4 \vee \overline{x_1} \cdot x_2 \cdot \overline{x_3} \cdot x_4 \vee$$
$$\overline{x_1} \cdot x_2 \cdot x_3 \cdot \overline{x_4} \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 \vee x_1 \cdot \overline{x_2} \cdot x_3 \cdot x_4 \vee$$
$$x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} \vee x_1 \cdot x_2 \cdot x_3 \cdot x_4$$

на мультиплексорі, який має три управляючі входи.

Розв'язання. Так як мультиплексор має три управляючі входи, тобто $n = 3$, а $m = 4$, то для розв'язку прикладу необхідно використати другий варіант. Для наочності і спрощення рішення перетворимо мінтерми заданої функції у десяткові числа

$$y = \vee(1, 3, 5, 6, 9, 11, 12, 15).$$

Розкладемо функцію за старшою змінною, в результаті чого рівняння матиме вигляд

$$y = \overline{x_1} \cdot [\vee(1, 3, 5, 6)] \vee x_1 \cdot [\vee(1, 3, 4, 7)].$$

В обох частинах розкладання є спільні числа 1 та 3, тому:

$$b_1 = b_3 = 1; \quad b_5 = b_6 = \overline{x_1}; \quad b_4 = b_7 = x_1.$$

На решту входів мультиплексора необхідно подати логічний нуль, а на його управляючі входи змінні x_2, x_3, x_4. Виходячи із цього, реалізація заданої логічної функції на мультиплексорі з трьома управляючими входами матиме вигляд, приведений на рис. 13.2.2.

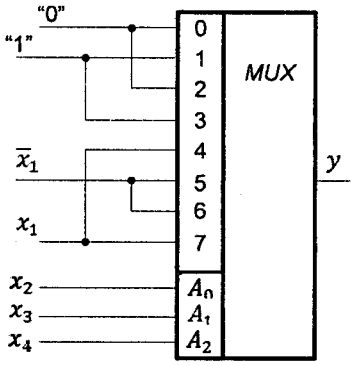


Рис. 13.2.2

При подальшому збільшенні кількості змінних у логічній функції необхідно зробити її розкладання за двома і т.д. змінними, із наступним використанням розглянутої вище логіки побудови одновихідних комбінаційних схем на мультиплексорах.

13.3. Логіка побудови багатовихідних комбінаційних схем на елементах логіки Буля

Означення 13.3.1. **Багатовихідною** комбінаційною схемою називають схему, яка управляє декількома елементами (механізмами) одночасно.

Вона задається системою булевих функцій

$$y_1 = f_1(x_1, x_2, ..., x_n),$$
$$y_2 = f_2(x_1, x_2, ..., x_n),$$
$$.....$$
$$y_m = f_m(x_1, x_2, ..., x_n).$$

Особливістю цих рівнянь є те, що вони реалізують різні функції від одних і тих же змінних.

Початковий стан проектування багатовихідних схем такий же, як і при проектуванні одновихідних (за допомогою таблиці істинності або іншим шляхом отримують конкретні функції для кожного виходу).

Кожна функція, незалежно одна від одної, може бути реалізована окремо як одновихідна функція. Але така система може бути не оптимальною через те, що окремі рівняння можуть мати деяку кількість однакових мінтермів, які доцільно реалізувати на спільних елементах. Тому пошук спільних імплікант і становить суть методу проектування подібних схем.

Даний метод оснований на пошуку спільних імплікант заданих функцій шляхом їх спільної мінімізації. Логіка послідовності цього методу має такі кроки.

1. Знайти прості імпліканти кожної із функцій.

2. Знайти прості імпліканти добутку всіх можливих пар заданих функцій

$$f_1 \cdot f_2;\ f_1 \cdot f_3;\ \dots\ f_1 \cdot f_m;\ f_2 \cdot f_3;\ f_2 \cdot f_4;\ \dots\ f_2 \cdot f_m \dots f_{m-1} \cdot f_m.$$

3. Знайти прості імпліканти добутку всіх можливих поєднань трьох, чотирьох і т.п. функцій. Останнім буде знайдено прості імпліканти добутку всіх функцій. Під добутком функцій розуміють їх спільні мінтерми.

4. Залишити серед отриманих однакових імплікант тільки одні і позначити буквами.

5. Серед отриманих імплікант методом Квайна вилучити зайві.

6. Розподілити отримані імпліканти за функціями таким чином, щоб кожна функція була сумою деякої кількості імплікант.

У подальшому проектування багатовихідної комбінаційної схеми виконують аналогічно проектуванню одновихідних комбінаційних схем.

Приклад 13.3.1. Побудувати двохвихідну комбінаційну схему, яка задана системою функцій у вигляді десяткових чисел

$$\begin{aligned} y_1 &= \vee(6, 7, 8, 9, 10, 11, 13, 14, 15), \\ y_2 &= \vee(1, 3, 6, 7, 8, 9, 10, 11). \end{aligned} \qquad (13.3.1)$$

Розв'язання. За допомогою карт Карно, рис. 13.3.1, знайдемо прості імпліканти кожної із функцій, відмічаючи мінтерми, з яких вони отримані.

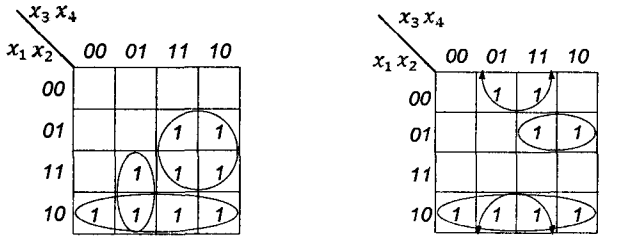


Рис. 13.3.1

Із рис. 13.3.1 маємо

$$y_1 = x_1 \cdot \overline{x_2}(8, 9, 10, 11) \vee x_2 \cdot x_3(6, 7, 14, 15) \vee x_1 \cdot \overline{x_3} \cdot x_4\ (9, 13);$$

$$y_2 = x_1 \cdot \overline{x_2}(8, 9, 10, 11) \vee \overline{x_2} \cdot x_4(1, 3, 9, 11) \vee \overline{x_1} \cdot x_2 \cdot x_3\ (6, 7).$$

Користуючись рис. 13.3.1 знайдемо спільні імпліканти добутку функцій y_1 та y_2. Відповідна карта Карно для добутку цих функцій приведена на рис. 13.3.2.

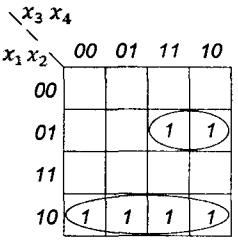


Рис. 13.3.2

Із рис. 13.3.2 маємо

$$y_{12} = x_1 \cdot \overline{x_2}(8, 9, 10, 11) \vee \overline{x_1} \cdot x_2 \cdot x_3(6, 7)$$

Позначимо кожний імплікант латинськими буквами, пропускаючи ті із них, які вже зустрічались раніше

$$A = x_1 \cdot \overline{x_2}(8, 9, 10, 11);$$

$$B = x_2 \cdot x_3(6, 7, 14, 15);$$

$$C = x_1 \cdot \overline{x_3} \cdot x_4(9, 13);$$

$$D = \overline{x_2} \cdot x_4(1, 3, 9, 11);$$

$$E = \overline{x_1} \cdot x_2 \cdot x_3(6, 7).$$

Будуємо таблицю Квайна, в якій стовпчики відповідають мінтермам функцій, а рядки — імплікантам, табл. 13.3.1.

Таблиця 13.3.1

Імпліканти	Мінтерми функцій															
	y_1									y_2						
	6	7	8	9	10	11	13	14	15	1	3	6	7	8	9	10	11
A			*	*	*	*								*	*	*	*
B	*	*						*	*								
C				*			*										
D										*	*				*		*
E												*	*				

Заповнення таблиці здійснюють таким чином. Для кожного відібраного імпліканта зірочку ставлять у стовпчику, номер якого є в списку мінтермів функцій цього імпліканта. Так, наприклад, імплікант *A* належить функції y_1 і y_2, тому зірочку проставляють у стопчиках 8, 9, 10, 11 першої і другої функцій. Аналогічно заповнюють зірочками таблицю і для решти імплікант.

В таблиці Квайна в першу чергу розглядуємо стовпчики, сума зірочок у яких дорівнює одиниці. Так, наприклад, для імпліканти *A* першим розглядуємо стовпчик 8 функції y_1, бо він здатний забезпечити одиничний сигнал для даної функції.

Решту стовпчиків даної імпліканти викреслюємо як для функції y_1, так і для функції y_2. Імпліканта *B* теж у кожному стовпчику має одну зірочку, тому можна розглянути стовпчик 6, а решту викреслити. Для імпліканти *C* першим необхідно розглянути стовпчик 13, бо він має одну зірочку, а решту викреслити, а для імплікант *D* і *E* — стовпчики 1 і 6 функції y_2 відповідно, а решту викреслити.

Таким чином для реалізації схеми необхідно використати п'ять імплікант — *A, B, C, D, E*. Зайвих імплікант при мінімізації не виявлено. Тепер необхідно задані імпліканти розподілити між функціями. Із табл. 13.3.1 випливає, що імплікант *A* належить обом функціям, імпліканти *B* і *C* тільки функції y_1, а *D* і *E* — лише функції y_2. Функції, як сума відповідних імплікант, матимуть вигляд

$$y_1 = A \vee B \vee C = x_1 \cdot \overline{x_2} \vee x_2 \cdot x_3 \vee x_1 \cdot \overline{x_3} \cdot x_4;$$

$$y_2 = A \vee D \vee E = x_1 \cdot \overline{x_2} \vee \overline{x_2} \cdot x_4 \vee \overline{x_1} \cdot x_2 \cdot x_3.$$

Побудова комбінаційної схеми з використанням рівнянь 13.3.2 відбувається аналогічно описаному в § 13.3. Дана схема на елементах логіки Буля має вигляд, рис. 13.3.3.

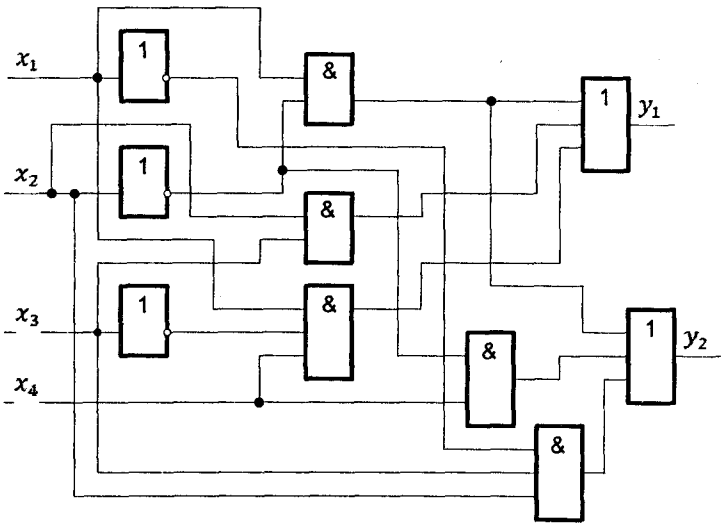


Рис. 13.3.3

13.4. Логіка побудови багатовихідних комбінаційних схем на дешифраторах

Значна кількість кроків у побудові схем робить класичний метод проектування багатовихідних комбінаційних схем інколи незручним для застосування. Обминути цей метод можна шляхом

використання дешифраторів. Із § 10.2 випливає, що кожний вихід дешифратора відповідає певному вхідному набору змінних, тобто певному мінтерму. Оскільки кожна функція є сума одиночних мінтермів, то для її реалізації достатньо об'єднати елементом «*АБО*» ті виходи дешифратора, які аналогічні одиночним мінтермам. Невикористані набори, якщо вони є, слід віднести до нульових.

Приклад 13.4.1. Побудувати чотирьохвихідну комбінаційну схему, яка задана системою функцій у вигляді десяткових чисел

$$y_1 = \vee(0, 1, 3, 4, 5);$$
$$y_2 = \vee(1, 2, 4, 5, 6);$$
$$y_3 = \vee(0, 2, 3, 6);$$
$$y_4 = \vee(0, 1, 3, 5, 6).$$

Розв'язання. Використовуючи § 10.2, будуємо двійковий дешифратор, який забезпечує на своїх виходах реалізацію чотирьох функцій, у вигляді десяткових чисел від 0 до 6. Отже, дешифратор повинен мати сім виходів. Згідно з даними § 10.2, дешифратор за кількістю входів і виходів зв'язаний співвідношенням $m = 2^n$, де n — кількість входів, а m — кількість виходів. У такому випадку для побудови двійкового дешифратора на сім виходів необхідно мати $n = \lceil \log_2 7 \rceil = \lceil 2.807 \rceil = 3$.

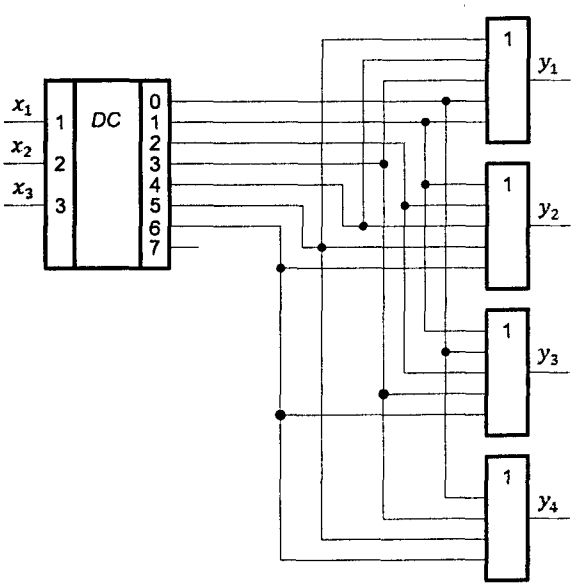


Рис. 13.4.1

Виходячи з цього, а також використовуючи дані § 10.1, будуємо чотирьохвихідну комбінаційну схему реалізацій функцій y_1, y_2, y_3, y_4 на дешифраторі і елементах «*АБО*», рис. 13.4.1.

Із рис. 13.4.1 випливає, що невикористаним набором є сьомий вихід дешифратора, який не бере участі в жодній із заданих за умовою функцій.

При збільшенні кількості змінних у функціях збільшують кількість дешифраторів, що приводить до утворення дешифраторного дерева.

13.5. Логіка побудови часових булевих схем

Робота часової булевої функції загалом описується рівнянням виду, § 7.1.1

$$y = \varphi(x_1, x_2, ...; x_n, t),\ t = 1, 2, ..., k.$$

Якщо маємо різні булеві функції в різні моменти часу, то їх можна записати так:

$$y_1 = \varphi_1 \cdot t_1 \vee \varphi_2 \cdot t_2 \vee ... \vee \varphi_n \cdot t_k, \qquad (13.5.1)$$

де φ_i — значення *i-ї* функції, яке визначає *i-y* компоненту в необхідній послідовності в момент часу t_i.

Структурна схема часової булевої функції, яка реалізує рівняння 13.5.1, наведена на рис. 13.5.1.

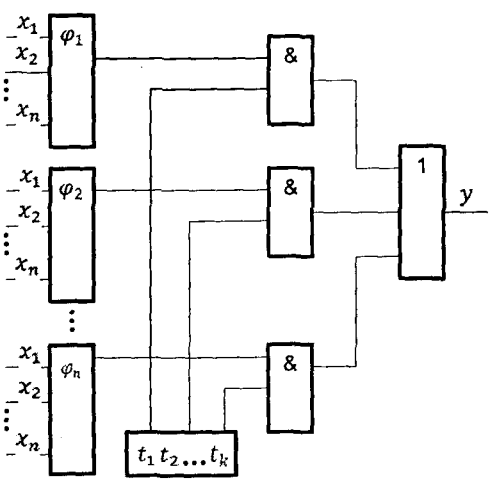


Рис. 135.1

Блоки $\varphi_1, \varphi_2, ..., \varphi_n$ реалізують функції $\varphi_1, \varphi_2, .., \varphi_k$, причому кожну з них реалізують звичайною вихідною комбінаційною схемою. Виходи блоків подаються на ключі, які реалізовані на елементах "*І*". Якщо на виході датчика часу буде організована послідовна видача сигналів $t_1, t_2, ..., t_n$, то на виходах схем "*І*" з'являтимуться послідовно в часі сигнали функцій $\varphi_1, \varphi_2, ..., \varphi_n$. При управлінні одним механізмом ці сигнали необхідно об'єднати елементом «*АБО*», але якщо необхідно здійснювати управління декількома механізмами, то цей елемент є зайвим.

Датчик часу може бути реалізований на дешифраторі та лічильнику, до якого підключають генератор *G*. Схема такого датчика часу на вісім затримок приведена на рис. 13.5.2.

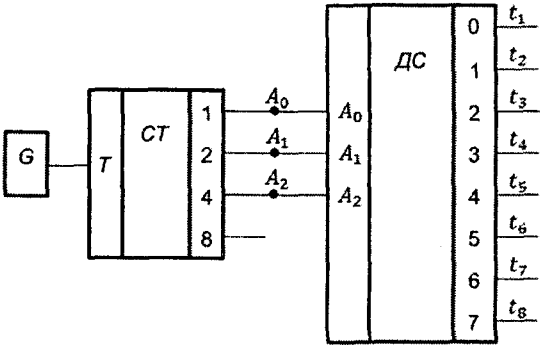


Рис. 13.5.2

В таких схемах дискретність часу $t_1 ... t_8$ визначається частотою роботи генератора *G*. Часова діаграма роботи датчика часу для рис. 13.5.2 наведена на рис. 13.5.3.

Із вищесказаного випливає, що логіка побудови часових логічних схем реалізується в декілька кроків. На першому кроці необхідно побудувати звичайні комбінаційні схеми за правилами, наведеними в § 13.1, ..., § 13.4. На другому кроці будують датчик часу, на третьому — ключі, а на четвертому — часову логічну схему, використовуючи перший, другий і третій кроки, а також елементи булевих функцій, наведених у § 10.1.

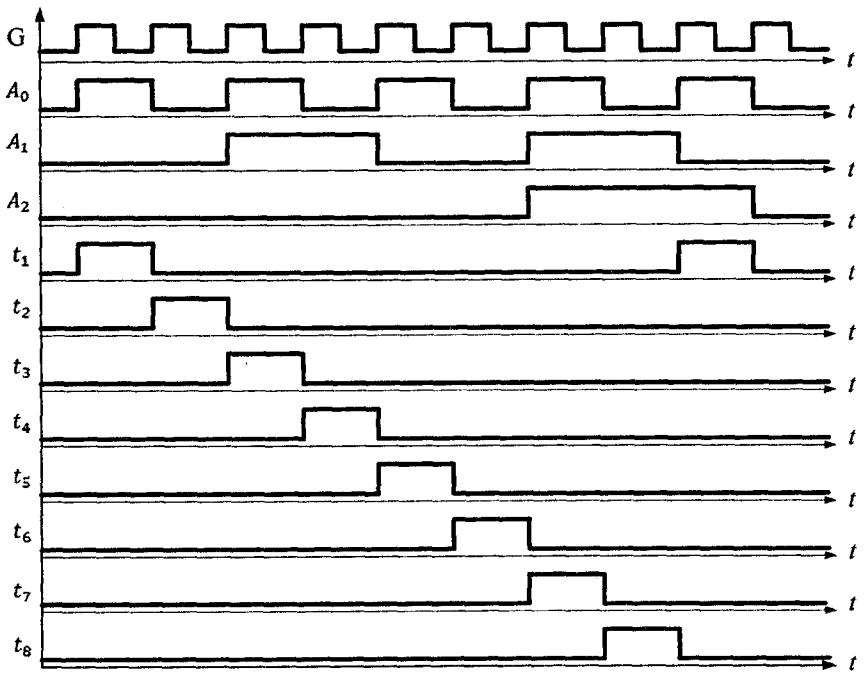


Рис. 13.5.3

Приклад 13.5.1. Побудувати часову булеву схему для функції

$$y_i = x_1 \cdot \overline{x_2} \cdot t_1 \vee \overline{x_1} \cdot x_2 \cdot t_2 \vee x_1 \cdot t_3 \vee x_2 \cdot t_4.$$

Розв'язання. Для реалізації заданого рівняння на першому кроці будуємо звичайні комбінаційні схеми мінтермів $x_1 \cdot \overline{x_2}$ і $\overline{x_1} \cdot x_2$ на елементах «*І*», «*НІ*» на другому — датчик часу на чотири затримки, на третьому — ключі з використанням елементів «*І*», а на четвертому — саму часову логічну схему, наведену на рис. 13.5.4.

Багатовихідні часові булеві схеми будуються аналогічно одновихідним, але тільки без елемента «*АБО*».

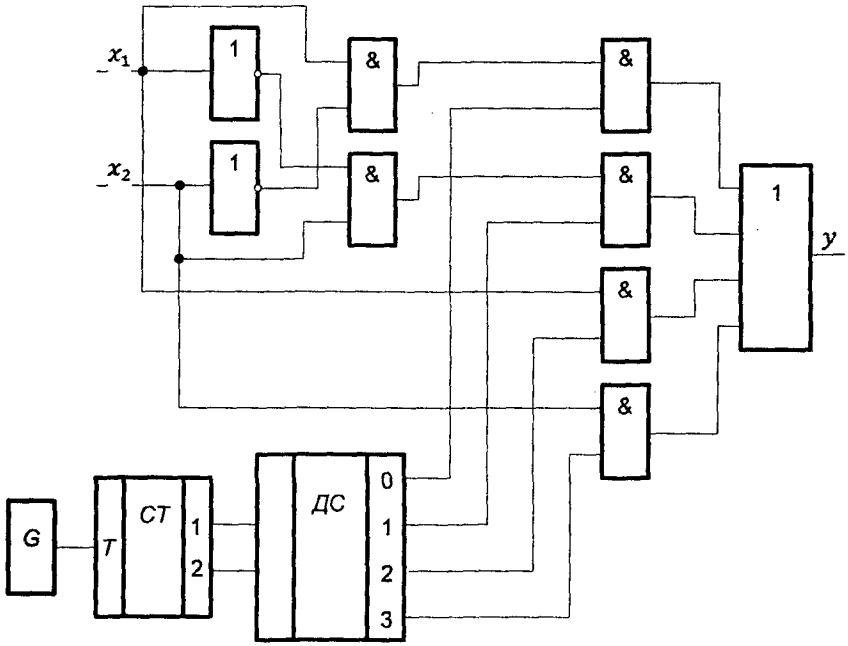


Рис. 13.5.4

13.6. Логіка побудови рекурентних булевих схем другого роду

Рекурентні булеві функції другого роду мають вигляд

$$y = \varphi\Big(x_{1t}, x_{2t}, \cdots, x_{nt}, x_{1(t-1)}, x_{2(t-1)}, \cdots, x_{n(t-1)}, \cdots,$$

$$x_{1(t-r)}, x_{2(t-r)}, \cdots, x_{n(t-r)}\Big).$$

Це функції, в яких між змінними, наприклад x_{1t} і $x_{1(t-1)}$ є часовий зсув.

Такі функції не мінімізують, тому що значення однієї і тієї ж змінної залежить від часу.

Логіка побудови рекурентних булевих схем другого роду має такі кроки. На першому кроці будують рекурентну булеву функ-

цію, яка описує роботу того чи іншого пристрою, а на другому — реалізують її, використовуючи § 13.1, ..., § 13.4, без урахування часових затримок. Побудова рекурентної булевої функції закінчується введенням між змінними часових затримок на третьому кроці. Часові затримки можна реалізувати на малогабаритних лініях затримки (МЛЗ) або D — тригерах, робота яких описана в § 12.2.

Приклад 13.6.1. За рекурентною булевою функцією

$$y = x_{1(t-1)} \vee x_{1t} \cdot \overline{x_{2t}} \vee x_{2(t-2)}$$

побудувати схему, використовуючи МЛЗ.

Розв'язання. В даній рекурентній булевій функції часові затримки введені між змінними x_{1t}, $x_{1(t-1)}$ і $\overline{x_{2t}}$, $x_{2(t-2)}$. Тому, використовуючи § 13.1, а також кроки два і три логіки побудови, отримаємо рекурентну булеву схему, наведену на рис. 13.6.1.

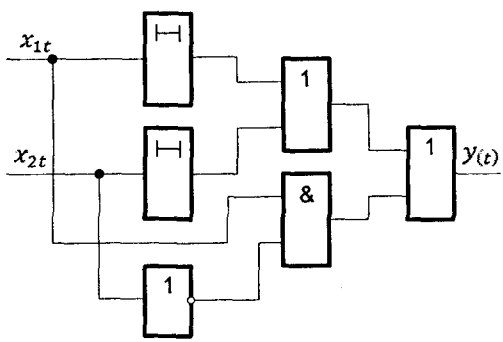


Рис. 13.6.1

Приклад 13.6.2. За рекурентною булевою функцією прикладу 13.6.1 побудувати схему, використовуючи в якості елементів часової затримки D–тригера.

Розв'язання. Побудова схеми відбувається аналогічно прикладу 13.6.1 з тою лише різницею, що замість МЛЗ вводимо синхронні D–тригери, § 12.2. Тоді отримаємо схему, яка реалізує рекурентну булеву функцію прикладу 13.6.1, рис. 13.6.2.

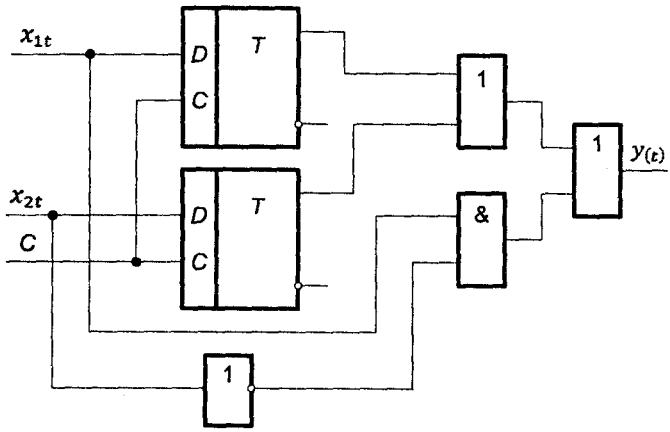


Рис. 13.6.2

13.7. Логіка побудови схем із застосуванням теорії автоматів

Логіка побудови схем із застосуванням теорії автоматів має такі кроки. На першому кроці на основі алгоритму роботи пристрою будують абстрактний автомат, який кодують на другому кроці двійковим нормальним кодом, отримуючи при цьому структурний автомат. На третьому кроці, використовуючи структурний автомат, будують таблиці переходів і виходів або відмічену таблицю переходів, за допомогою яких (якої) на четвертому кроці знаходять канонічні рівняння роботи пристрою, які на п'ятому кроці мінімізують, а на шостому за даними рівняннями будують схему пристрою.

Приклад 13.7.1. Згідно із алгоритмом роботи пристрою, який заданий абстрактним автоматом, рис. 13.7.1,

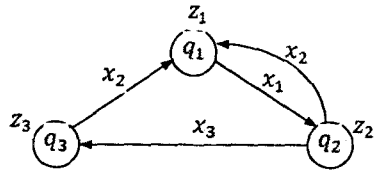


Рис. 13.7.1

побудувати схему управління.

Розв'язання. Для забезпечення реалізації трьох станів абстрактного автомата q_1, q_2, q_3 необхідно в структурному автоматі мати згідно з формулою $n=\lceil lod_2 3 \rceil = \lceil 1{,}807 \rceil = 2$ два елементи пам'яті, які можуть задовольнити реалізацію чотирьох станів: 00, 01, 10, 11. В нашому випадку для кодування станів абстрактного автомата використаємо кодові стани $q_1 \to 00$, $q_2 \to 01$, $q_3 \to 10$. В результаті цього структурний автомат матиме вигляд, рис. 13.7.2.

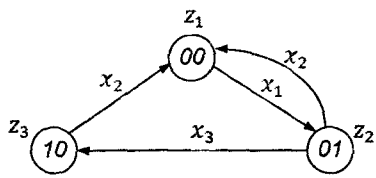


Рис. 13.7.2

Використовуючи відмічену для автомата Мура (§ 8.3) таблицю переходів, табл. 13.7.1,

Таблиця 13.7.1

q_j \ z_k / x_i \ q_j	z_1	z_2	z_3
	00	01	10
x_1	01	-	-
x_2	-	00	00
x_3	-	10	-

отримаємо канонічні рівняння роботи схеми пристрою, які матимуть такий вигляд:

$$z_1 = \overline{y_1} \cdot \overline{y_2};\ z_2 = \overline{y_1} \cdot y_2;\ z_3 = y_1 \cdot \overline{y_2};$$

$$\varphi_1^1 = x_3;\ \varphi_1^0 = x_2 \cdot \overline{y_2};\ \varphi_2^1 = x_1 \cdot \overline{y_1};\ \varphi_2^0 = x_3;$$

де φ_1^1, φ_2^3 і φ_1^0, φ_2^0 — функції включення і виключення відповідно першого і другого елементів пам'яті структурного автомата;

y_1, y_2 і $\overline{y_1}, \overline{y_2}$ — сигнали на виходах першого і другого елементів пам'яті, які відповідають логічним сигналам «1» і «0»відповідно;

z_1, z_2, z_3 — сигнали управління пристрою.

Функція φ_1 відповідає елементу кода, розміщеного зліва, а φ_2 — того, що справа. Рівняння включення і виключення першого і другого елементів пам'яті отримують у відповідності з принципом, описаним у § 8.10.

Як видно із отриманих рівнянь, їх мінімізація не потрібна, тому переходимо до шостого кроку, де будуємо схему пристрою. При побудові схеми використовуємо § 10.1, § 12.1, § 13.3. Наведена схема має вигляд, рис. 13.7.3,

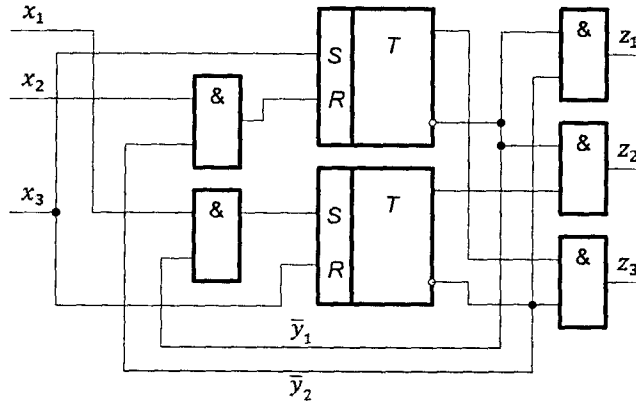


Рис. 13.7.3

13.8. Логіка побудови схем із застосуванням теорії автоматів і програмованих логічних матриць

Логіка побудови комбінаційних схем із застосуванням програмованих логічних матрицях (ПЛМ) була розглянута в § 11.5, а логіка побудови схем із застосуванням теорії автоматів — у § 13.7. Об'єднавши ці дві логіки, ми отримаємо логіку побудови комп'ютерних схем із застосуванням теорії автоматів і ПЛМ. Вона складається із семи кроків. На першому кроці на основі заданого алгоритму роботи будують абстрактний автомат, який на другому кроці кодують двійковим нормальним кодом. На третьому кроці, використовуючи структурний автомат, отриманий на другому кроці, будують таблиці переходів і виходів, за допомогою яких (якої) на четвертому кроці знаходять канонічні рівняння роботи. Мінімізація їх відбувається на п'ятому кроці, а на шостому – вибір необхідної ПЛМ та її програмування. На сьомому кроці за вибраною і запрограмованою

ПЛМ, а також за отриманими на другому кроці елементами пам'яті будують необхідну схему.

Приклад 13.8.1. За алгоритмом роботи пристрою, який заданий абстрактним автоматом, рис. 13.8.1,

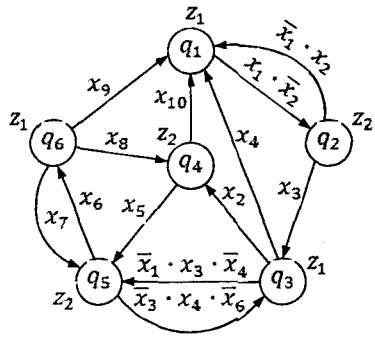


Рис. 13.8.1

побудувати схему управління з низьким рівнем активності для функції z і високим — для решти функцій.

Розв'язання. Щоб забезпечити реалізацію шести станів абстрактного автомата $q_1, ..., q_6$, необхідно в структурному автоматі мати згідно з формулою $n =]\log_2 6[=]2.807[= 3$ три елементи пам'яті, які можуть задовольнити реалізацію восьми станів: 000, 001, 010, 011, 100, 101, 111. В нашому випадку для кодування станів абстрактного автомата використаємо кодові стани: $q_1 \rightarrow 000$; $q_2 \rightarrow 001$; $q_3 \rightarrow 010$; $q_4 \rightarrow 011$; $q_5 \rightarrow 110$; $q_6 \rightarrow 101$. В результаті цього структурний автомат матиме вигляд, рис. 13.8.2,

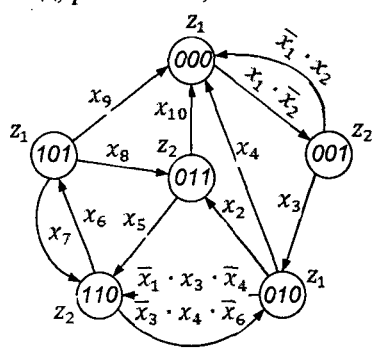


Рис. 13.8.2

Використовуючи відмічену для автомата Мура (§ 8.3) таблицю переходів, табл. 13.8.1,

Таблиця 13.8.1

z_k / q_j	z_1	z_2	z_1	z_2	z_2	z_1
q_j / x_i	000	001	010	011	110	101
$x_1 \cdot \overline{x_2}$	001	—	—	—	—	—
$\overline{x_1} \cdot x_2$	—	000	—	—	—	—
x_3	—	010	—	—	—	—
x_4	—	—	000	—	—	—
x_2	—	—	011	—	—	—
$\overline{x_1} \cdot x_3 \cdot \overline{x_4}$	—	—	110	—	—	—
$\overline{x_3} \cdot x_4 \cdot \overline{x_6}$	—	—	—	—	010	—
x_5	—	—	—	110	—	—
x_6	—	—	—	—	—	110
x_7	—	—	—	—	—	110
x_8	—	—	—	—	—	011
x_9	—	—	—	—	—	000
x_{10}	—	—	—	000	—	—

отримаємо канонічні рівняння роботи схеми, які матимуть такий вигляд:

$$z_1 = \overline{y_1} \cdot \overline{y_2} \cdot \overline{y_3} \vee \overline{y_1} \cdot y_2 \cdot \overline{y_3} \vee y_1 \cdot \overline{y_2} \cdot y_3;$$

$$z_2 = \overline{y_1} \cdot \overline{y_2} \cdot y_3 \vee \overline{y_1} \cdot y_2 \cdot y_3 \vee y_1 \cdot y_2 \cdot \overline{y_3};$$

$$\varphi_1^1 = \overline{x_1} \cdot x_3 \cdot \overline{x_4} \cdot y_2 \cdot \overline{y_3} \vee x_5 \cdot y_2;$$

$$\varphi_1^0 = \overline{x_3} \cdot x_4 \cdot \overline{x_6} \cdot y_2 \cdot \overline{y_3} \vee x_9 \cdot \overline{y_2} \vee x_8 \cdot y_3;$$

$$\varphi_2^1 = x_3 \cdot \overline{y_1} \vee x_6 \cdot y_1 \vee x_7 \cdot y_1 \vee x_8 \cdot y_3;$$

$$\varphi_2^0 = x_4 \cdot \overline{y_1} \cdot \overline{y_3} \vee x_{10} \cdot \overline{y_1};$$

$$\varphi_3^1 = x_1 \cdot \overline{x_2} \cdot \overline{y_1} \cdot \overline{y_2} \vee x_2 \cdot \overline{y_1} \cdot y_2;$$

$$\varphi_3^0 = \overline{x_1} \cdot x_2 \cdot \overline{y_1} \cdot \overline{y_2} \vee x_3 \cdot \overline{y_1} \vee x_5 \cdot y_2 \vee x_6 \cdot y_1 \vee x_7 \cdot y_1 \vee x_9 \cdot \overline{y_2} \vee x_{10} \cdot \overline{y_1};$$

де φ_1^1, φ_2^1, φ_3^1 і φ_1^0, φ_2^0, φ_3^0 — функції включення і виключення відповідно першого, другого і третього елементів пам'яті структурного автомата;

y_1, y_2, y_3 і $\overline{y_1}$, $\overline{y_2}$, $\overline{y_3}$ — сигнали на виходах першого, другого і третього елементів пам'яті, які аналогічні логічним сигналам "1" і "0" відповідно;

z_1, z_2 — сигнали управління пристрою.

Функція φ_1 відповідає елементу кода, розміщеного зліва, а φ_3 — справа. Рівняння включення і виключення елементів пам'яті отримують відповідно з принципом, описаним у § 8.10.

Як видно із отриманих рівнянь, їх мінімізація не потрібна, тому переходимо до шостого кроку, де необхідно вибрати відповідну ПЛМ.

Виходячи з канонічних рівнянь роботи, ПЛМ повинна відповідати таким даним. Кількість кон'юнкторів у ній повинна бути не менше 2, диз'юнкторів — не менше 8, вхідних змінних — не менше 13. Таким властивостям відповідає ПЛМ, мікросхема серії К556РТ1, яка має входи для 16 змінних, 8 виходів для реалізації восьми функцій і 48 кон'юнкторів.

Згідно з отриманими функціями, z_1, z_2, φ_1^1, φ_1^0, φ_2^1, φ_2^0, φ_3^1, φ_3^0 присвоюємо номери їх кон'юнкторам:

$$k_1 = \overline{y_1} \cdot \overline{y_2} \cdot \overline{y_3};\ k_2 = \overline{y_1} \cdot y_2 \cdot \overline{y_3};\ k_3 = y_1 \cdot \overline{y_2} \cdot y_3;\ k_4 = \overline{y_1} \cdot \overline{y_2} \cdot y_3;$$

$$k_5 = \overline{y_1} \cdot y_2 \cdot y_3;\ k_6 = y_1 \cdot y_2 \cdot \overline{y_3};\ k_7 = \overline{x_1} \cdot x_3 \cdot \overline{x_4} \cdot y_2 \cdot \overline{y_3};\ k_8 = x_5 \cdot y_2;$$

$$k_9 = \overline{x_3} \cdot x_4 \cdot \overline{x_6} \cdot y_2 \cdot \overline{y_3};\ k_{10} = x_9 \cdot \overline{y_2};\ k_{11} = x_3 \cdot \overline{y_1};\ k_{12} = x_6 \cdot y_1;$$

$$k_{13} = x_7 \cdot y_1;\ k_{14} = x_8 \cdot y_3;\ k_{15} = x_4 \cdot \overline{y_1} \cdot \overline{y_3};\ k_{16} = x_{10} \cdot \overline{y_1};$$

$$k_{17} = x_1 \cdot \overline{x_2} \cdot \overline{y_1} \cdot \overline{y_2};\ k_{18} = x_2 \cdot \overline{y_1} \cdot y_2;\ k_{19} = \overline{x_1} \cdot x_2 \cdot \overline{y_1} \cdot \overline{y_2};$$

$$k_{20} = x_3 \cdot \overline{y_1};\ k_{21} = x_5 \cdot y_2;\ k_{22} = x_6 \cdot y_1;\ k_{23} = x_7 \cdot y_1;\ k_{24} = x_9 \cdot \overline{y_2};$$

$$k_{25} = x_8 \cdot y_3;\ k_{26} = x_{10} \cdot \overline{y_1}.$$

Використовуючи рекомендації § 9.3 і дані § 9.4, програмуємо отримані функції і їх результати заносимо в табл. 13.8.2.

Таблиця 13.8.2

k_1	Кон'юнктори													Рівень активності							
	Вхідна змінна													0	0	1	1	1	1	1	1
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	y_1	y_2	y_3	Вихідна функція							
	Номер програмованого входу													z_1	z_2	φ_1^1	φ_1^0	φ_2^1	φ_2^0	φ_3^1	φ_3^0
	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	B1	B2	B3	B4	B5	B6	B7	B8
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
k_1	*	*	*	*	*	*	*	*	*	*	0	0	0	*A*	*	*	*	*	*	*	*
k_2	*	*	*	*	*	*	*	*	*	*	0	1	0	*A*	*	*	*	*	*	*	*
k_3	*	*	*	*	*	*	*	*	*	*	1	0	1	*A*	*	*	*	*	*	*	*
k_4	*	*	*	*	*	*	*	*	*	*	0	0	1	*	*A*	*	*	*	*	*	*
k_5	*	*	*	*	*	*	*	*	*	*	0	1	1	*	*A*	*	*	*	*	*	*
k_6	*	*	*	*	*	*	*	*	*	*	1	1	1	*	*A*	*	*	*	*	*	*
k_7	0	*	1	0	*	*	*	*	*	*	*	*	*	*	*	*A*	*	*	*	*	*
k_8	*	*	*	*	1	*	*	*	*	*	*	1	*	*	*	*A*	*	*	*	*	*
k_9	*	*	0	1	*	0	*	*	*	*	*	1	0	*	*	*	*A*	*	*	*	*
k_{10}	*	*	*	*	*	*	*	*	1	*	*	0	*	*	*	*	*A*	*	*	*	*
k_{11}	*	*	1	*	*	*	*	*	*	*	0	*	*	*	*	*	*	*A*	*	*	*
k_{12}	*	*	*	*	*	1	*	*	*	*	*	1	*	*	*	*	*	*A*	*	*	*
k_{13}	*	*	*	*	*	*	1	*	*	*	1	*	*	*	*	*	*	*A*	*	*	*
k_{14}	*	*	*	*	*	*	*	1	*	*	*	*	1	*	*	*	*	*A*	*	*	*
k_{15}	*	*	*	0	*	*	*	*	*	*	0	*	0	*	*	*	*	*	*A*	*	*
k_{16}	*	*	*	*	*	*	*	*	*	1	0	*	*	*	*	*	*	*	*A*	*	*
k_{17}	1	0	*	*	*	*	*	*	*	*	0	0	*	*	*	*	*	*	*	*A*	*
k_{18}	*	1	*	*	*	*	*	*	*	*	0	1	*	*	*	*	*	*	*	*A*	*
k_{19}	0	1	*	*	*	*	*	*	*	*	0	0	*	*	*	*	*	*	*	*	*A*
k_{20}	*	*	1	*	*	*	*	*	*	*	0	*	*	*	*	*	*	*	*	*	*A*
k_{21}	*	*	*	*	1	*	*	*	*	*	*	1	*	*	*	*	*	*	*	*	*A*
k_{22}	*	*	*	*	*	1	*	*	*	*	1	*	*	*	*	*	*	*	*	*	*A*

Закінчення табл. 13.8.2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
k_{23}	*	*	*	*	*	*	1	*	*	*	1	*	*	*	*	*	*	*	*	*	*A*
k_{24}	*	*	*	*	*	*	*	*	1	*	*	0	*	*	*	*	*	*	*	*	*A*
k_{25}	*	*	*	*	*	*	*	1	*	*	*	*	1	*	*	*	*A*	*	*	*	*
k_{26}	*	*	*	*	*	*	*	*	*	1	0	*	*	*	*	*	*	*	*	*	*A*

На невикористаних входах $A14$, ..., $A16$ з номерами кон'юнкторів k_1, ..., k_{26} перемички перепалюються.

На сьомому кроці будуємо схему з використанням ПЛМ, мікросхема серії К556РТ1. При побудові схеми використовуємо § 10.1, §12.1, § 13.3. Наведена схема має вигляд, рис. 13.8.3,

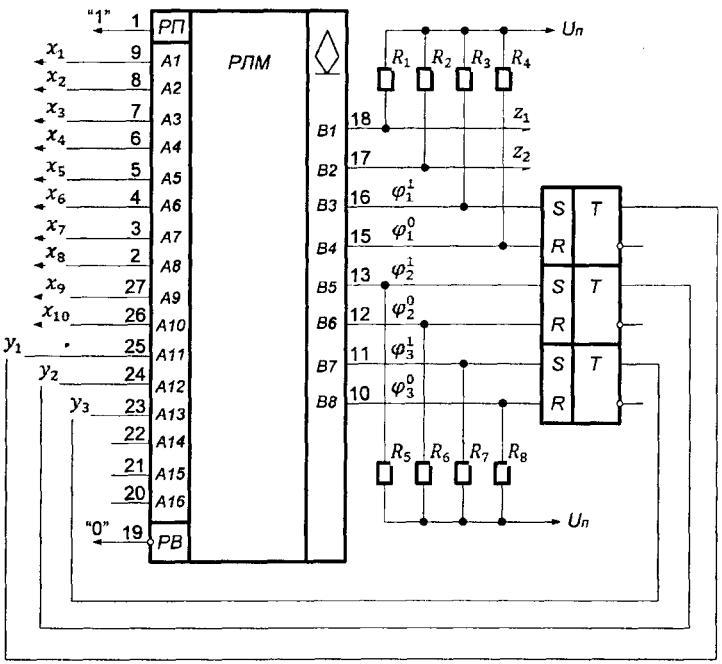


Рис. 13.8.3

де $R_1, ..., R_8$ — резистори величиною 2 кОм; $U_п$ — джерело живлення мікросхеми.

Із рис. 13.8.3 випливає, що схема реалізації пристрою із застосуванням ПЛМ має значно простіший вигляд, ніж із застосуванням звичайних елементів логіки Буля. Такі схеми доцільно використовувати при значній кількості канонічних рівнянь.

Контрольні запитання

1. Сформулюйте логіку побудови одновихідних комбінаційних схем на елементах логіки Буля.
2. Назвіть базиси побудови одновихідних комбінаційних схем.
3. Чим відрізняється базис Шеффера від базиса Пірса?
4. В якому базисі доцільно будувати одновихідні комбінаційні схеми?
5. Сформулюйте логіку побудови одновихідних комбінаційних схем на мультиплексорах.
6. Чим відрізняється логіка побудови одновихідних комбінаційних схем на мультиплексорах від логіки на елементах Буля ?
7. Що необхідно зробити в мультиплексорі, якщо кількість його управляючих входів менша за кількість змінних у функції ?
8. Чим відрізняється логіка побудови багатовихідних комбінаційних схем на елементах логіки Буля від логіки побудови на дешифраторах?
9. Яка із логік, наведених у п. 8, приводить до більш простих схем ?
10. Сформулюйте послідовність побудови часових булевих схем.
11. Який пристрій в часових булевих схемах необхідно додатково використовувати?
12. Сформулюйте послідовність побудови рекурентних булевих схем другого роду.
13. В чому полягає сутність логіки побудови схем із застосуванням теорії автоматів ?
14. Для чого застосовують кодування абстрактних автоматів ?
15. Що називають канонічними рівняннями роботи автомата ?
16. Де і в яких випадках доцільно застосовувати програмовані логічні матриці при побудові комп'ютерних схем ?

Задачі для самостійного розв'язування

1. Для логічної функції $y = x_1 \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot \overline{x_3}$ побудувати одновихідну комбінаційну схему на елементах логіки Буля.

2. Використовуючи умову задачі 1, побудувати одновихідну комбінаційну схему у базисі Шеффера і Пірса.

3. Використовуючи умову задачі 1, побудувати одновихідну комбінаційну схему на мультиплексорі.

4. Для логічної функції, заданої в десятковій системі числення

$$y = \vee(0, 1, 3, 4, 5, 7),$$

побудувати одновихідну комбінаційну схему на елементах логіки Буля, у базисі Шеффера і на мультиплексорі. Отримані схеми порівняти за складністю і зробити висновки.

5. Для заданої системи логічних функцій у десятковій системі числення

$$y_1 = \vee(0, 2, 4, 5, 7, 9, 10, 12, 13);$$
$$y_2 = \vee(0, 1, 3, 5, 6, 8, 9, 11, 12, 14)$$

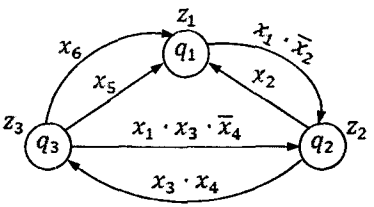
побудувати багатовихідну комбінаційну схему на елементах Буля.

6. Використовуючи умову задачі 5, побудувати багатовихідну комбінаційну схему на дешифраторі.

7. Для часової булевої функції $y_t = \overline{x_1} \cdot \overline{x_2} \cdot t_1 \vee x_1 \cdot \overline{x_3} \cdot t_2 \vee x_1 \cdot x_2 \cdot x_3 \cdot t_3$ побудувати схему.

8. Для рекурентної булевої функції другого роду $y_t = x_{2(t-2)} \vee \vee x_{2(t-1)} \cdot x_{1t} \cdot \overline{x}_{3t} \vee x_{1(t-1)} \vee x_{3(t-1)}$ побудувати схему із застосуванням D-тригерів.

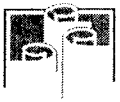
9. Алгоритм роботи електронного пристрою, заданий абстрактним автоматом.



Необхідно побудувати схему з використанням елементів логіки Буля і програмованої логічної матриці мікросхеми серії К556РТ1. Отримані схеми порівняти за складністю, зробити висновки.

Коментарі

Основні свідчення, які викладені в цьому розділі, за логікою побудови одновихідних і багатовихідних комбінаційних схем частково взяті з [1, 7, 19, 28, 29], логіка побудови часових рекурентних булевих схем випливає з [24, 27], а логіка побудови схем із застосуванням теорії автоматів, у тому числі із застосуванням програмованих логічних матриць, запропонована автором.



ЛІТЕРАТУРА

1. *Абутов Ю. О.* Микроэлектронные устройства программного и логического управления. Принципы построения. — Москва: Машиностроение, 1979. — 208 с.

2. *Айзерман М. А., Гусев Л. А., Розоноэр Л. И., Смирнова И. М., Таль А. А., Логіка. Автоматы. Алгоритмы.* — Москва: Физматгиз. — 1963. — 315 с.

3. *Бабич М. А., Жуков И. А.* Компьютерная схемотехника. — Киев; МК-Пресс, 2004. — 576 с.

4. *Баранов С. И.* Синтез микропрограммных автоматов. — Ленинград; Энергия,1979. — 232 с.

5. *Бардачов Ю. М., Соколова Н. А., Ходаков В. Є.* Дискретна математика. — Київ; Вища школа, 2007. — 383 с.

6. *Белоусов А. И., Ткачёв С. Б.* Дискретная математика. — Москва; Издательство МГТУ им. Н. Э. Баумана, 2001. — 774 с.

7. *Блейксли Т. Р.* Проектирование цифровых устройств с малыми и большими интегральными схемами. — Киев: Вища школа, 1981. — 336 с.

8. *Бондаренко М. Ф., Білоус Н. В., Руткас А. Г.* Комп'ютерна дискретна математика. — Харків: Компанія СМІТ, 2004. — 480 с.

9. *Гаазе-Раппопорт М. Г.* Автоматы и живые организмы. — Москва: Физматгиз, 1962. — 281 с.

10. *Гилл Артур.* Введение в теорию конечных автоматов. — Москва: Наука, 1966. — 272 с.

11. *Глушков В. М.* Синтез цифровых автоматов. — Москва: Физматгиз, 1962. — 476 с.

12. *Горбатов В. А.* Основы дискретной математики. — Москва: Высшая школа, 1986. — 312 с.

13. *Донской В. И.* Дискретная математика. — Симферополь: Сонат, 2000. — 360 с.

14. *Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В.* Прикладна теорія цифрових автоматів,-Київ: Видавництво НАУ, 2007. — 364 с.

15. *Жабін В.І., Ткаченко В.В.* Цифрові автомати. Практикум. — Київ: ВЕК+, 2004. — 160 с.

16. *Жабин В.И. и др.* Логические основы и схемотехника ЭВМ. Практикум. — Киев: ВЕК+, 1999. — 128 с.

17. *Захаров Б. Н., Поспелов Д. А., Хазицкий В. Е.* Системы управления. Задание. Проектирование. Реализация. — Москва: Энергия, 1977. — 420 с.

18. *Капітонова Ю. И., Кривий С. Л., Летичевський О. А., Луцкий Г. М., Печурін М. К.* Основи дискретної математики. — Київ: Наукова думка, 2002. — 579 с.

19. *Корнійчук А. І.* Проектування пристроїв та систем управління. — Житомир: ЖІТІ, 2000. — 276 с.

20. *Мелихов А.Н.* Ориентированые графы и конечные автоматы. — Москва: Наука, 1971. — 423 с.

21. *Нікольскій Ю. В., Пасічник В. В., Щербина Ю. М.* Дискретна математика. — Київ: Видавнича група BHV, 2007. — 368 с.

22. *Нешумова К. А.* Электронные вичислительные машины и системы. — Москва: Высшая школа, 1989. — 315 с.

23. Отраслевой стандарт. ОСТ 11.340.915-82. Микросхемы интеральные серии 556(556РТ1, 556РТ2), Р556(Р556РТ1, Р556РТ2). Руководство по применению ОКП. 623 000. — 51 с.

24. *Поспелов Д. А.* Логические методы анализа и синтеза схем. — Москва: Энергия, 1974. — 368 с.

25. *Рабинович З. А., Раманаускас В. А.* Типовые операции в вычислителных машинах. — Киев: Техника, 1980. — 264 с.

26. *Самофалов К.Г.* и др. Прикладная теорія цифрових автоматов. — Киев: Вища школа, 1987. — 375 с.

27. *Соломатин Н. М.* Логические элементы ЭВМ. — Москва: Висшая школа, 1990. — 160 с.

28. *Угрюмов Е. П.* Цифровая схемотехника – СПБ.: БХВ. — Петербург, 2001. — 528 с.

29. *Хоровиц П., Хилл В.* Искусство схемотехники. — Москва: Мир, 1993. — 367 с.

30. *Яблонский С. В.* Введение в дискретную математику. — Москва: Наука. 1986. — 384 с.

Наукове видання

МАТВІЄНКО Микола Павлович

КОМП'ЮТЕРНА ЛОГІКА

Навчальний посібник

Керівник видавничого проекту *Зарицкий В. І.*
Дизайн обкладинки *Седих О. О.*
Комп'ютерна верстка *Іваненко О. М.*

Підписано до друку 28.12.2011. Формат 60×84 1/16.
Папір офсетний. Друк офсетний. Гарнітура Times New Roman.
Умовн. друк. аркушів — 16,74. Обл.-вид. аркушів — 19,58. Тираж 500.
Зам. № 33/02.

«Видавництво Ліра-К»
Свідоцтво № 3981, серія ДК.
03067, м. Київ, вул. Прилужна 14, оф. 42
тел./факс (044) 247-93-37; 450-91-96
Сайт: lira-k.com.ua, відділ збуту: lira-k@ukr.net, редакція: zv_lira@ukr.net

Віддруковано з готових діапозитивів у ПП "Юнісофт"
61036, м. Харків, вул. Морозова, 13Б
www.uornado.com.ua; info@ttornado.com.ua для питань з приводу друку