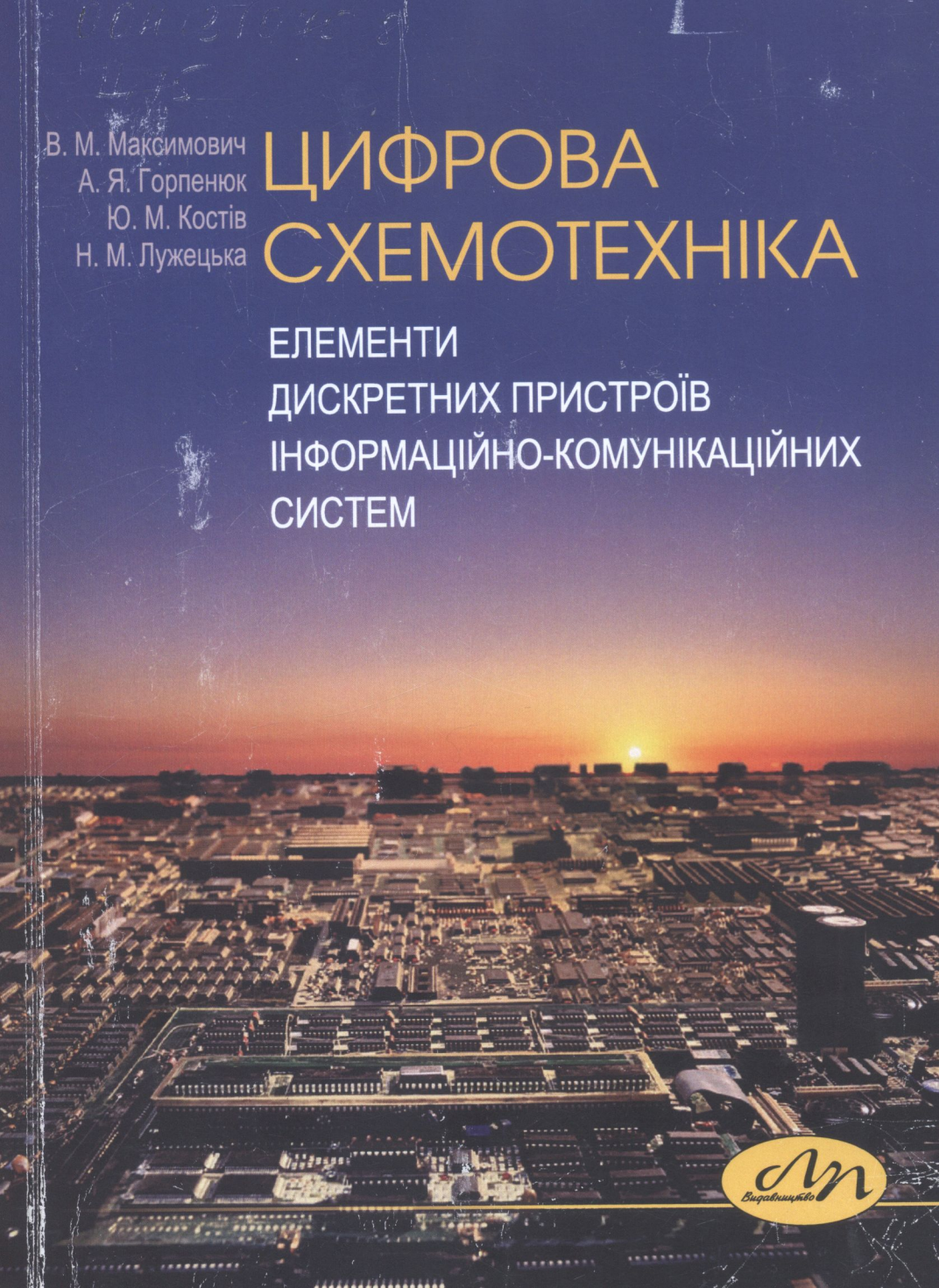
В. М. Максимович
А. Я. Горпенюк
Ю. М. Костів
Н. М. Лужецька

ЦИФРОВА СХЕМОТЕХНІКА

ЕЛЕМЕНТИ ДИСКРЕТНИХ ПРИСТРОЇВ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ



Видавництво Л

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ЦИФРОВА СХЕМОТЕХНІКА

ЕЛЕМЕНТИ ДИСКРЕТНИХ ПРИСТРОЇВ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ

Навчальний посібник

Рекомендувала Науково-методична рада
Національного університету "Львівська політехніка"

Львів
Видавництво Львівської політехніки
2015

УДК 004.35:316.77
ББК 22
Ц 752

Автори:

В. М. Максимович, д-р техн. наук, проф., зав. кафедри безпеки інф. технологій;
А. Я. Горпенюк, канд. техн. наук, доц. кафедри захисту інформації;
Ю. М. Костів, канд. техн. наук, доц. кафедри безпеки інф. технологій;
Н. М. Лужецька, асист. кафедри безпеки інф. технологій

Рецензенти:

Чекурін В. Ф., д-р фіз.-мат. наук, проф., завідувач відділу Інституту прикладних проблем механіки і математики ім. Я. С. Підстригача НАН України;

Марчук М. В., д-р фіз.-мат. наук, проф., завідувач відділу Інституту прикладних проблем механіки і математики ім. Я. С. Підстригача НАН України;

Хома В. В., д-р техн. наук, проф., Національний університет "Львівська політехніка"

Рекомендувала Науково-методична рада
Національного університету "Львівська політехніка"
як навчальний посібник для студентів галузі знань "Інформаційна безпека"
(протокол № 9 від 27.05.2015 р.)

Розглянуто основи алгебри логіки, принципи функціонування та побудови комбінаційних та послідовнісних цифрових пристроїв, особливості сучасної елементної бази цифрової схемотехніки, застосування цифрових пристроїв для захисту інформації. Значну увагу звернено на розгляд принципів побудови генераторів псевдовипадкових чисел на елементах цифрової техніки та аналіз особливостей сучасних програмованих логічних інтегральних схем.

Для студентів денної та заочної форм навчання, які навчаються за галуззю знань "Інформаційна безпека".

ISBN 978-617-607-870-8

ЗМІСТ

Вступ 5

1. ОСНОВИ БУЛЕВОЇ АЛГЕБРИ 7
1.1. Логічні висловлення, змінні та функції 7
1.2. Інверсія, диз'юнкція, кон'юнкція 8
1.3. Закони булевої алгебри 11
1.4. Вхідний набір. Повний перелік функцій однієї і двох змінних 12
1.5. Властивості функцій: сума за mod 2, рівнозначність, стрілка Пірса, штрих Шефера, імплікації 18
1.6. Функціонально повні набори функцій (базиси) 20
1.7. Форми зображення логічних функцій 23
1.8. Мінімізація булевих функцій 33
1.9. Запис логічних функцій у базисах функцій І-НЕ та АБО-НЕ 43
Питання і задачі для самоконтролю 45

2. ЦИФРОВІ ПРИСТРОЇ 46
2.1. Класифікація цифрових пристроїв 46
2.2. Комбінаційні пристрої 49
2.2.1. Принципи логічного проектування комбінаційних пристроїв 49
2.2.2. Негативні явища в комбінаційних пристроях 52
2.2.3. Синтез комбінаційних пристроїв із багатьма виходами 54
2.2.4. Дешифратори 59
2.2.5. Шифратори 62
2.2.6. Перетворювачі кодів 64
2.2.7. Мультиплексори 66
2.2.8. Демультиплексори 67
2.2.9. Програмовані логічні матриці 67
2.2.10. Комбінаційні суматори 69
2.2.11. Пристрої порівняння 73
Питання і задачі для самоконтролю 76
2.3. Послідовнісні пристрої 77

2.3.1. Принципи логічного проектування
послідовнісних пристроїв 77
2.3.2. Тригери 80
2.3.2.1. Тригерні системи, класифікація тригерів 80
2.3.2.2. RS-тригери 83
2.3.2.3. JK-тригери 87
2.3.2.4. D-тригери 89
2.3.2.5. T-тригери 92
2.3.2.6. Комбіновані тригери 92
2.3.3. Лічильники імпульсів 93
2.3.3.1. Призначення та класифікація
лічильників імпульсів 93
2.3.3.2. Синтез синхронних лічильників 95
2.3.3.3. Асинхронні лічильники 98
2.3.4. Регістри 98
2.3.4.1. Регістри пам'яті 101
2.3.4.2. Регістри зсуву 101
Питання і задачі для самоконтролю 104

3. ЗАСТОСУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ У ЗАСОБАХ ЗАХИСТУ ІНФОРМАЦІЇ 105
3.1. Генератори випадкових і псевдовипадкових чисел 105
3.2. Приклади побудови ГПВЧ 106
3.2.1. ГПВЧ на основі регістрів зсуву 106
3.2.2. ГПВЧ на основі регістрів пам'яті
та комбінаційного суматора 112
Питання і задачі для самоконтролю 116

4. ПРОГРАМОВАНІ ЛОГІЧНІ ІНТЕГРАЛЬНІ СХЕМИ (ПЛІС) 117
4.1. SPLD – прості програмовані логічні пристрої 118
4.2. CPLD – складні програмовані логічні пристрої 122
4.3. FPGA – програмовані користувачем вентильні
матриці 123
4.4. SOPC – системи на програмованому кристалі 128
Питання і задачі для самоконтролю 130

Список літератури 131

ВСТУП

У конспекті лекцій наведено матеріал, який є базовим для подальшого вивчення багатьох дисциплін, так чи інакше пов'язаних з цифровою технікою: обчислювальної техніки, архітектури комп'ютерних систем, мікропроцесорної техніки й інших.

У першому розділі подано основи алгебри логіки (булевої алгебри). Наведено відомості про основні логічні функції та їхні властивості, закони булевої алгебри і форми зображення логічних функцій. Особливо закцентовано на різних методах мінімізації логічних функцій.

У другому розділі розглянуто два основні типи цифрових пристроїв – комбінаційні пристрої і послідовнісні пристрої. В описі комбінаційних пристроїв особливу увагу звернуто на різні способи побудови комбінаційних суматорів – одного з основних елементів арифметично-логічних пристроїв. Під час розгляду послідовнісних пристроїв детально розкрито принцип роботи і внутрішню побудову різних типів тригерів.

У третьому розділі подано основи теорії побудови та аналізу, особливості застосування та приклади побудови генераторів псевдовипадкових чисел.

У четвертому розділі проаналізовано особливості структурної побудови різних типів програмованих логічних інтегральних схем.

В усіх розділах посібника є питання і задачі, за допомогою яких студенти зможуть перевіряти свої знання.

1. ОСНОВИ БУЛЕВОЇ АЛГЕБРИ

1.1. Логічні висловлення, змінні та функції

Принципи булевої алгебри (алгебри логіки) заклав у XIX ст. ірландський математик Джордж Буль (1815–1864). На початковому етапі застосування булевої алгебри її розглядали як алгебру висловлювань або алгебру подій. До уваги брали два стани: стан дійсного, правдивого чи наявного або протилежний стан – недійсного, неправдивого, відсутнього. Корисним також стало застосування булевої алгебри як алгебри множин, у якій розглядають стан належності або неналежності певного елемента до деякої множини.

У кінці 30-х років XX ст. К. Шеннон застосував булеву алгебру під час розгляду електричних схем, у яких використовують реле або інші елементи, що характеризуються тільки двома станами: короткозамкненим або розімкненим. У наступні роки булеву алгебру широко застосовували під час аналізу інших схем і пристроїв, побудованих на базі елементів із двома чітко визначеними станами, і передовсім під час аналізу цифрових пристроїв, оскільки більшість їх працює у двійковій системі числення.

Абстрагуючись від фізичного змісту того, що розглядається, булева алгебра, як математичний апарат – це алгебра змінного параметра, який може мати тільки два стани. Ці стани прийнято позначати символами 0 і 1. Символ 0, наприклад, позначає неправдиве висловлення, відсутність події, неналежність до множини, розімкнене реле, низький рівень напруги на виході логічного елемента, а символ 1 – правдиве висловлення, наявність події, належність до множини, короткозамкнене реле, високий рівень напруги на виході логічного елемента.

Треба особливо наголосити, що символи 0 і 1 в булевій алгебрі не є числами. Алгебра логіки – це алгебра станів, а не алгебра чисел. Вона має свої поняття і закони, які істотно відрізняються від понять і законів алгебри чисел.

Основним поняттям алгебри логіки є висловлення – будь-яке твердження, відносно якого можна говорити, що воно істинне чи хибне. У такому разі вважається, що кожне висловлення не може бути одночасно і істинним, і хибним. Висловлення можна позначити символом x і вважати, що $x = 1$, коли висловлення істинне, і $x = 0$, якщо висловлення хибне.

В алгебрі логіки висловлення можуть бути простими і складними. Висловлення, значення істинності яких не залежать від значень істинності інших висловлень, називаються простими. Під час аналізу і синтезу логічних схем просте висловлення розглядають як незалежну змінну, що набуває двох значень – 0 чи 1.

Висловлення, значення істинності яких залежать від значень істинності інших висловлень, що його складають, називаються складними і також можуть набувати двох значень – 0 чи 1. Складне висловлення можна розглядати як логічну функцію простих висловлень.

Так, булевими (логічними) функціями $y = f(x_1, x_2, ..., x_n)$ називаються функції скінченного числа аргументів $x_1, x_2, ..., x_n$, коли ці аргументи, а також і самі функції, можуть набувати тільки двох значень – 0 або 1.

1.2. Інверсія, диз'юнкція, кон'юнкція

Інверсія, диз'юнкція і кон'юнкція є базовими функціями, за допомогою яких можуть бути визначені і в зручній формі записані усі інші логічні функції.

Інверсія (інші назви – логічне заперечення, функція НЕ) є функцією одного аргументу. Вона звичайно позначається рискою над аргументом:

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

Запис $\bar{x}$ читається "не x".

Функція інверсії дорівнює 1, коли її аргумент дорівнює 0, і навпаки:

$$\bar{0} = 1, \quad \bar{1} = 0. \qquad (1.1)$$

Електронний логічний елемент, що реалізує функцію інверсії, називається інвертором (елементом НЕ, елементом логічного заперечення). Його умовне позначення наведено на рис. 1.1, *а*.

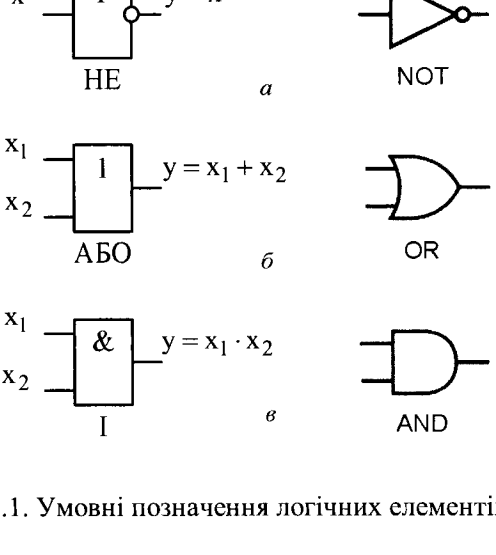


Рис. 1.1. Умовні позначення логічних елементів

Диз'юнкція (інші назви – логічне додавання, функція АБО) є функцією двох або більшого числа аргументів. Трапляються дві різні форми запису цієї функції:

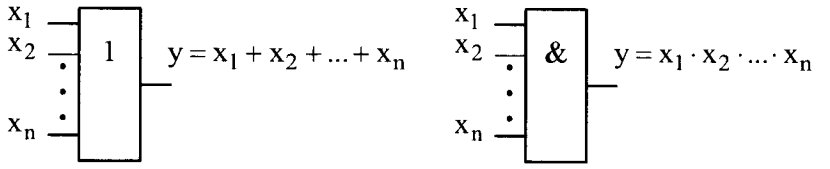
$$y = x_1 + x_2, \quad y = x_1 \vee x_2.$$

Далі використовуватимемо символ "+" і запис $x_1 + x_2$, читатимемо "x_1 плюс x_2" чи "x_1 або x_2".

Функція диз'юнкції дорівнює 1, коли хоча б один з її аргументів дорівнює 1. Для функції від двох аргументів це означає, що:

$$\begin{aligned} 0+0&=0, \\ 0+1&=1, \\ 1+0&=1, \\ 1+1&=1. \end{aligned} \qquad (1.2)$$

Логічний елемент, що реалізує функцію диз'юнкції, називається елементом АБО (елементом логічного додавання, диз'юнктором). Його умовне позначення для двох аргументів наведено на рис. 1.1, *б*, а для n аргументів – на рис. 1.2, *а*.



а *б*

Рис. 1.2

Кон'юнкція (інші назви – логічне множення, функція І) є функцією двох або більшого числа аргументів. Для її позначення використовуються різні символи:

$$y = x_1 \,\&\, x_2,\ y = x_1 \wedge x_2,\ y = x_1 \cdot x_2,\ y = x_1 x_2.$$

Далі використовуватимемо дві останні форми представлення, а записи $x_1 \cdot x_2$ і $x_1 x_2$ читатимемо "x_1 помножити на x_2" або просто "$x_1 x_2$".

Функція кон'юнкції дорівнює 1 тоді і лише тоді, коли усі її аргументи дорівнюють 1. Тобто для двох аргументів:

$$\begin{aligned} 0 \cdot 0 &= 0, \\ 0 \cdot 1 &= 0, \\ 1 \cdot 0 &= 0, \\ 1 \cdot 1 &= 1. \end{aligned} \qquad (1.3)$$

Логічний елемент, що реалізує функцію кон'юнкції, називають елементом І (елементом логічного множення, кон'юнктором). Його умовне зображення для двох аргументів наведено на рис. 1.1, *в*, а для n аргументів – на рис. 1.2, *б*.

1.3. Закони булевої алгебри

На основі постулатів (1.1)–(1.3) можна довести закони булевої алгебри, які наведено в табл. 1.1. Їхню справедливість можна перевірити прямою підстановкою усіх можливих комбінацій значень аргументів у відповідні рівняння і зведення їх, із використанням виразів (1.1)–(1.3), до тотожностей.

Таблиця 1.1

Закони булевої алгебри

Закони універсальної множини	$x+1=1$ $x\cdot 1=x$
Закони нульової множини	$x+0=x$ $x\cdot 0=0$
Закон подвійного заперечення	$=$ $x=x$
Закони доповняльності	$x+\overline{x}=1$ $x\cdot\overline{x}=0$
Закони тавтології	$x+x=x$ $x\cdot x=x$
Закони комутативності (переміщення)	$x_1+x_2=x_2+x_1$ $x_1\cdot x_2=x_2\cdot x_1$
Закони асоціативності (сполучення)	$(x_1+x_2)+x_3=x_1+(x_2+x_3)$ $(x_1\cdot x_2)\cdot x_3=x_1\cdot(x_2\cdot x_3)$
Закони дистрибутивності (розподілу)	$x_1\cdot(x_2+x_3)=x_1\cdot x_2+x_1\cdot x_3$ $x_1+(x_2\cdot x_3)=(x_1+x_2)\cdot(x_1+x_3)$
Закони поглинання (абсорбції)	$x_1+x_1\cdot x_2=x_1$ $x_1\cdot(x_1+x_2)=x_1$
Закони склеювання	$x_1\cdot x_2+x_1\cdot\overline{x_2}=x_1$ $(x_1+x_2)\cdot(x_1+\overline{x_2})=x_1$
Закони де Моргана	$\overline{x_1+x_2}=\overline{x_1}\cdot\overline{x_2}$ $\overline{x_1\cdot x_2}=\overline{x_1}+\overline{x_2}$

Дія деяких законів, наведених у табл. 1.1, може поширитися на довільне скінченне число аргументів. Так, наприклад, закони де Моргана для n аргументів мають вигляд:

$$\overline{x_1 + x_2 + ... + x_n} = \overline{x_1} \cdot \overline{x_2} \cdot ... \cdot \overline{x_n}, \qquad (1.4)$$

$$\overline{x_1 \cdot x_2 \cdot ... \cdot x_n} = \overline{x_1} + \overline{x_2} + ... + \overline{x_n}.$$

Закони де Моргана узагальнив Шеннон у теорему: під час застосування операції інвертування до довільної комбінації логічних змінних, сполучених знаками диз'юнкції та кон'юнкції $f(x_1,...,x_n,\text{“+”},\text{“·”})$, справедливе співвідношення

$$\overline{f(x_1,...,x_n,\text{“+”},\text{“·”})} = f(\overline{x_1},...,\overline{x_n},\text{“·”},\text{“+”}). \qquad (1.5)$$

Це означає, що операція інвертування довільного виразу згаданого вигляду еквівалентна заміні в ньому початкових значень логічних змінних їх інверсними значеннями за одночасної заміни диз'юнкцій на кон'юнкції і кон'юнкцій на диз'юнкції. Наприклад:

$$\overline{x_1 x_2 + x_3 x_4 \overline{x_5}} = (\overline{x_1} + \overline{x_2})(\overline{x_3} + \overline{x_4} + x_5).$$

Форма більшості законів, наведених у табл. 1.1, свідчить про те, що алгебра логіки дуальна відносно операцій логічного додавання і множення. Своєрідність цієї алгебри проявляється, зокрема, в дистрибутивних законах, другий з яких не має аналога в алгебрі чисел.

1.4. Вхідний набір. Повний перелік функцій однієї і двох змінних

Для логічної функції $y = f(x_1, x_2, ..., x_n)$ існує $m = 2^n$ різних вхідних наборів (комбінацій значень) аргументів $x_1, x_2, ..., x_n$, на яких вона може набувати значень 0 або 1. Так, кожній логічній функції можна увідповіднити 2^n розрядне двійкове число, а загальна кількість різних логічних функцій n змінних дорівнює:

$$M = 2^m = 2^{2^n}. \qquad (1.6)$$

Набори аргументів прийнято нумерувати. Звичайно, ці номери збігаються із значеннями двійкових чисел, розряди яких є аргументами $x_1, x_2, ..., x_n$. Зрозуміло, що номери наборів у такому разі змінюються від 0 до $2^n - 1$. Приклад множини вхідних наборів для функції від трьох аргументів наведений у табл. 1.2.

Таблиця 1.2

Номер набору	Аргументи		
	x_1	x_2	x_3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

У теорії логічних функцій особливе значення мають функції однієї і двох змінних. Із цих функцій за допомогою операції суперпозиції (підстановка одних логічних функцій замість змінних в інші логічні функції) може бути записана будь-яка логічна функція. Наприклад, якщо $y = f_1(x_1, x_2)$ і $x_2 = f_2(x_3, x_4)$, то $y = f_1[x_1, f_2(x_3, x_4)]$, тобто суперпозицією двох логічних функцій, кожна з яких залежить від двох змінних, отримана логічна функція, яка залежить від трьох змінних.

Для функцій від однієї логічної змінної згідно з (1.6) всього існує чотири логічні функції, наведені в табл. 1.3.

Аналіз цих функцій показує, що $f_0(x)$ і $f_3(x)$ є постійними величинами, незалежними від значень аргументу x: $f_0(x) = 0$, $f_3(x) = 1$. $f_1(x)$ повторює значення свого аргументу: $f_1(x) = x$. Практичний інтерес має лише функція $f_2(x) = \overline{x}$, яка розглянута в § 1.2.

Таблиця 1.3

Логічні функції однієї змінної

Значення аргументу x і функцій f_0 – f_3			Назва функції	Позначення
x	0	1		
f_0	0	0	Константа нуля	0
f_1	0	1	Повторення	x
f_2	1	0	Інверсія (логічне заперечення, функція НЕ)	$\overline{x}$
f_3	1	1	Константа 1	1

Для двох змінних згідно з (1.6) існує 16 різних логічних функцій. Перелік цих функцій наведено у табл. 1.4. Тут також наведені їх умовні позначення, еквівалентні вирази в базисі І, АБО, НЕ і умовні позначення відповідних логічних елементів.

Функції f_0 і f_{15} – постійні величини, незалежні від значень їх змінних x_1 і x_2:

$$f_0(x_1, x_2) = 0,\ f_{15}(x_1, x_2) = 1.$$

Функції f_3 і f_5 повторюють одну із змінних:

$$f_3(x_1, x_2) = x_1,\ f_5(x_1, x_2) = x_2.$$

Функції f_1, f_7, f_{10}, f_{12} не нові. Вони зображають уже відомі операції кон'юнкції, диз'юнкції, інверсії:

$$f_1(x_1, x_2) = x_1 \cdot x_2,\ f_7(x_1, x_2) = x_1 + x_2,\ f_{10}(x_1, x_2) = \overline{x_2},$$

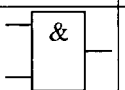
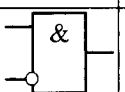
$$f_{12}(x_1, x_2) = \overline{x_1}.$$

Під час розгляду функцій з табл. 1.4 видно, що вони є попарно взаємоінверсними:

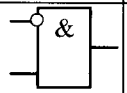
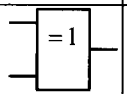
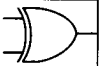
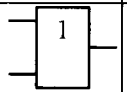
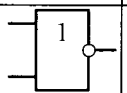
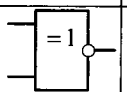
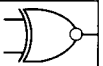
$$f_0 = \overline{f_{15}},\ f_1 = \overline{f_{14}},\ f_2 = \overline{f_{13}},\ f_3 = \overline{f_{12}},\ f_4 = \overline{f_{11}},\ f_5 = \overline{f_{10}},\ f_6 = \overline{f_9},\ f_7 = \overline{f_8}.$$

Таблиця 1.4

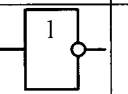
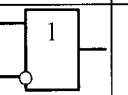
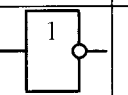
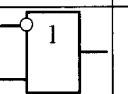
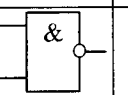
Логічні функції двох змінних

Значення аргументів x_1, x_2 і функцій $f_0 - f_{15}$					Назва функції	Умовне позначення	Еквівалентний вираз у базисі І, АБО, НЕ	Умовне позначення логічного елемента	
x_1	0	0	1	1					
x_2	0	1	0	1					
1	2	3	4	5	6	7	8	9	10
f_0	0	0	0	0	Константа нуля	0	0		
f_1	0	0	0	1	Кон'юнкція (логічне множення, функція І)	$x_1 \cdot x_2$	$x_1 \cdot x_2$		
f_2	0	0	1	0	Заборона по x_2	$x_1 \overline{\rightarrow} x_2$	$x_1 \cdot \overline{x_2}$		
f_3	0	0	1	1	Повторення x_1	x_1	x_1		

Продовження табл. 1.4

1	2	3	4	5	6	7	8	9	10
f_4	0	1	0	0	Заборона по x_1	$x_1 \overline{\leftarrow} x_2$	$\overline{x_1} \cdot x_2$		
f_5					Повторення x_2	x_2	x_2		
f_6	0	1	1	0	Сума за mod 2 (логічна нерівнозначність, виняткове АБО)	$x_1 \oplus x_2$	$x_1 \cdot \overline{x_2} + \overline{x_1}$		
f_7	0	1	1	1	Диз'юнкція (логічне додавання, функція АБО)	$x_1 + x_2$	$x_1 + x_2$		
f_8	1	0	0	0	Стрілка Пірса (функція Веба, функція АБО-НЕ)	$x_1 \downarrow x_2$	$\overline{x_1 + x_2}$		
f_9	1	0	0	1	Логічна рівнозначність (логічна еквівалентність, виняткове АБО-НЕ)	$x_1 \sim x_2$	$\overline{x_1} \cdot \overline{x_2} + x_1$		

Закінчення табл. 1.4

1	2	3	4	5	6	7	8	9	10
f_{10}	1	0	1	0	Інверсія x_2	$\overline{x_2}$	$\overline{x_2}$		
f_{11}	1	0	1	1	Імплікація від x_2 до x_1	$x_1 \leftarrow x_2$	$x_1 + \overline{x_2}$		
f_{12}	1	1	0	0	Інверсія x_1	$\overline{x_1}$	$\overline{x_1}$		
f_{13}	1	1	0	1	Імплікація від x_1 до x_2	$x_1 \rightarrow x_2$	$\overline{x_1} + x_2$		
f_{14}	1	1	1	0	Штрих Шефера (функція І-НЕ)	$x_1 \mid x_2$	$\overline{x_1 \cdot x_2}$		
f_{15}	1	1	1	1	Константа одиниці	1	1		

1.5. Властивості функцій: сума за mod 2, рівнозначність, стрілка Пірса, штрих Шефера, імплікації

Сума за mod 2

Для цієї функції справедливі закони комутативності і асоціативності:

$$x_1 \oplus x_2 = x_2 \oplus x_1,$$

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

Функція пов'язана з базовими логічними функціями (інверсія, диз'юнкція, кон'юнкція) рівняннями

$$x_1 \oplus x_2 = x_1\overline{x_2} + \overline{x_1}x_2 = (x_1 + x_2)(\overline{x_1} + \overline{x_2}).$$

Для неї справедливі також тотожності:

$$x \oplus 0 = x,\ x \oplus 1 = \overline{x},\ x \oplus \overline{x} = 1,\ x \oplus x = 0.$$

Логічна рівнозначність

Функція логічної рівнозначності є інверсною відносно попередньої:

$$x_1 \sim x_2 = \overline{x_1 \oplus x_2}.$$

Для неї також справедливі комутативний і асоціативний закони:

$$x_1 \sim x_2 = x_2 \sim x_1,$$

$$x_1 \sim (x_2 \sim x_3) = (x_1 \sim x_2) \sim x_3.$$

Функція пов'язана з базовими логічними функціями рівняннями

$$x_1 \sim x_2 = \overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2 = (\overline{x_1} + x_2)(x_1 + \overline{x_2}).$$

Для неї справедливі також тотожності:

$$x \sim 0 = \overline{x},\ x \sim 1 = x,\ x \sim \overline{x} = 0,\ x \sim x = 1.$$

Стрілка Пірса (функція АБО-НЕ)

Для цієї функції справедливий комутативний закон:

$$x_1 \downarrow x_2 = x_2 \downarrow x_1.$$

Асоціативний закон для неї не справджується.

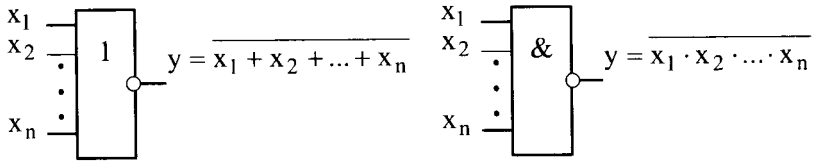
Функція пов'язана з базовими логічними функціями рівняннями

$$x_1 \downarrow x_2 = \overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}.$$

Для неї справедливі також тотожності:

$$x \downarrow 0 = \overline{x},\ x \downarrow 1 = 0,\ x \downarrow \overline{x} = 0,\ x \downarrow x = \overline{x}.$$

Функція "стрілка Пірса" може бути визначена для довільного числа аргументів: функція $f(x_1, x_2, ..., x_n) = \overline{x_1 + x_2 + ... + x_n}$ дорівнює 1 тоді і лише тоді, коли всі її аргументи дорівнюють 0. Логічний елемент, що реалізує функцію Пірса для n аргументів, зображений на рис. 1.3, *а*.



а *б*

Рис. 1.3

Штрих Шефера (функція І-НЕ)

Для цієї функції, так само як і для попередньої, є справедливим комутативний закон

$$x_1 \mid x_2 = x_2 \mid x_1,$$

а асоціативний закон не справджується.

З базовими логічними функціями функція "штрих Шефера" пов'язана рівняннями

$$x_1 \mid x_2 = \overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2}.$$

Функції "штрих Шефера" і "стрілка Пірса" пов'язані співвідношеннями, які аналогічні до законів де Моргана:

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

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

Для функції "штрих Шефера" справедливі такі тотожності:

$$x \mid 0 = 1,\ x \mid 1 = \overline{x},\ x \mid \overline{x} = 1,\ x \mid x = \overline{x}.$$

Функція "штрих Шефера" може бути визначена для довільного числа аргументів: функція $f(x_1, x_2, ..., x_n) = \overline{x_1 \cdot x_2 \cdot ... \cdot x_n}$ дорівнює 0 тоді і лише тоді, коли всі її аргументи дорівнюють 1. Логічний елемент, що реалізує функцію "штрих Шефера" для n аргументів, зображений на рис. 1.3, *б*.

Імплікація

Функція імплікації, на відміну від попередніх, несиметрична відносно своїх аргументів. Комутативний і асоціативний закони для неї не справджуються. З базовими логічними функціями вона пов'язана рівнянням

$$x_1 \to x_2 = \overline{x_1} + x_2.$$

Для неї виконуються такі тотожності:

$$x_1 \to x_2 = \overline{x_2} \to \overline{x_1},$$

$$(x_1 \to x_2) \to x_1 = x_1,$$

$$x \to 0 = \overline{x},\ x \to 1 = 1,\ x \to \overline{x} = \overline{x},\ x \to x = 1,$$

$$0 \to 1 = 1,\ 1 \to x = x,\ \overline{x} \to x = x.$$

1.6. Функціонально повні набори функцій (базиси)

Набір функцій алгебри логіки називається функціонально повним, якщо будь-яка функція алгебри логіки може бути утворена за допомогою суперпозиції функцій цього набору. Кажуть, що функціонально повний набір утворює базис.

Мінімальним базисом називають такий, у якому в разі вилучення хоча б однієї функції, що утворює базис, порушується його повнота.

Для визначення критеріїв повноти наборів логічних функцій необхідно класифікувати їх за певними властивостями.

Логічна функція $f(x_1, x_2, ..., x_n)$ називається такою, що зберігає константу нуль, якщо на вхідному наборі нульових значень її аргументів вона набуває значення нуля: $f(0,0,...,0) = 0$.

Логічна функція $f(x_1, x_2, ..., x_n)$ називається такою, що зберігає одиницю, якщо на вхідному наборі одиничних значень її аргументів вона набуває значення одиниці: $f(1,1,...,1) = 1$.

Логічна функція $f(x_1, x_2, ..., x_n)$ називається самодвоїстою, якщо інвертування усіх її аргументів приводить до інвертування функції: $f(\overline{x_1}, \overline{x_2}, ..., \overline{x_n}) = \overline{f(x_1, x_2, ..., x_n)}$.

Логічна функція $f(x_1, x_2, ..., x_n)$ називається монотонною, якщо для будь-яких двох наборів аргументів $(x_1', x_2', ..., x_n')$ і $(x_1'', x_2'', ..., x_n'')$, для яких $x_1' \le x_1''$, $x_2' \le x_2''$, ... , $x_n' \le x_n''$ виконується умова $f(x_1', x_2', ..., x_n') \le f(x_1'', x_2'', ..., x_n'')$.

Логічна функція $f(x_1, x_2, ..., x_n)$ називається лінійною, якщо її можна записати у такому вигляді: $f(x_1, x_2, ..., x_n) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n$, де a_i – константи, що мають значення нуля чи одиниці.

Наведені властивості дають змогу сформулювати теорему, яка визначає умови повноти будь-якого набору функцій алгебри логіки: для того, щоб набір функцій був повним, необхідно і достатньо, щоб він містив хоча б одну функцію, що не зберігає нуль, одну функцію, що не зберігає одиниці, одну нелінійну, одну немонотонну і одну несамодвоїсту функцію.

Із теореми випливає, що п'ять функцій утворюють повний набір, якщо кожна з них має одну із вказаних властивостей, яка відрізняється від властивостей інших функцій. Дійсне ж число функцій повного набору може бути меншим, оскільки та сама функція може мати одразу декілька властивостей.

У табл. 1.5 символом "х" позначені властивості логічних функцій двох змінних.

Таблиця 1.5

Властивості	Логічні функції відповідно до табл. 1.4															
	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}
Зберігає 0	x	x	x	x	x	x	x	x								
Зберігає 1		x		x		x		x		x		x		x		x
Самодвоїста				x		x					x		x			
Монотонна	x	x		x		x		x								x
Лінійна	x			x		x	x			x	x		x			x

На основі даних табл. 1.5 можна скласти різні мінімальні функціонально повні набори (мінімальні базиси), зокрема:

кон'юнкція, інверсія;

диз'юнкція, інверсія;

стрілка Пірса;

штрих Шефера.

Так, будь-яку логічну функцію можна утворити тільки за допомогою функцій одного мінімального базису, наприклад, винятково за допомогою функції "штрих Шефера".

Вибір мінімального базису пов'язаний з вибором набору логічних елементів, на яких буде побудований конкретний цифровий пристрій. Очевидно, що зменшення кількості функцій, що входять у базис, відповідає зменшенню кількості різних логічних елементів, що використовуються. Однак, необхідно враховувати, що в разі реалізації цифрового пристрою важлива не тільки кількість типів елементів, що використовуються, а їхня загальна кількість. Тому на практиці, звичайно, використовуються набори логічних елементів, які є значно ширшими від тих, що утворюють мінімальні базиси.

1.7. Форми зображення логічних функцій

Будь-яку логічну функцію можна задати і зобразити у різних формах, основними з яких є: словесна, таблична, числового запису, аналітична, координатна.

Словесна форма

Словесна форма – це логічне висловлення, яке обумовлює умови отримання одного з двох можливих значень функції і саме значення функції. Наприклад, функцію імплікації $f_{13} = \overline{x_1} + x_2$ словесно можна описати так: функція f_{13} дорівнює нулю тоді і лише тоді, коли x_1 дорівнює одиниці і x_2 дорівнює нулю.

Таблична форма

Таблична форма характеризується таблицею істинності, яка для логічної функції $f(x_1, x_2, ..., x_n)$ має n стовпців для запису значень аргументів, один стовпець для запису значень функції і 2^n рядків за числом вхідних наборів. Вхідні набори (рядки) можна пронумерувати за правилом, наведеним у § 1.4 (табл. 1.2). Приклад таблиці істинності функції трьох змінних $y = f(x_1, x_2, x_3)$ наведений у табл. 1.6.

Таблиця 1.6

Номер набору	x_1	x_2	x_3	y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Якщо для змінних $x_1, x_2, ..., x_n$ існує група логічних функцій $y_1, y, ..., y_s$, то їх можна зобразити у вигляді таблиці істинності так, як це показано в табл. 1.7 для функції трьох змінних.

Таблиця 1.7

Номер набору	x_1	x_2	x_3	y_1	y_2	...	y_s
0	0	0	0	1	0	...	1
1	0	0	1	1	1	...	1
2	0	1	0	1	1	...	0
3	0	1	1	0	0	...	0
4	1	0	0	1	0	...	0
5	1	0	1	0	1	...	1
6	1	1	0	0	0	...	0
7	1	1	1	1	1	...	0

Числовий запис

Числовий запис – це спосіб запису функції у вигляді номерів наборів аргументів, на яких ця функція набуває значення 1. Для функції, заданої табл. 1.6, можна записати $y = \{0,1,2,4,7\}_{x_1,x_2,x_3}$. Аналогічний числовий запис можна зробити, записавши номери наборів аргументів, на яких функція набуває значення 0.

Аналітичний запис

Аналітичний запис – це спосіб задання функції у вигляді алгебраїчного виразу, який отримують під час застосування логічних операцій до змінних алгебри логіки. Якщо попередні форми зображення логічних функцій лише визначають значення функції на конкретних наборах аргументів, то алгебраїчне зображення, крім цього, дає змогу виконувати різні аналітичні перетворення, що необхідні для аналізу і синтезу цифрового пристрою, що реалізує цю логічну функцію.

Залежно від вибраного базису, аналітичний запис тієї самої функції може бути різним. Окрім того, закони алгебри логіки дають змогу змінювати форму аналітичного запису, не змінюючи базис.

Найчастіше використовуються так звані нормальні (канонічні) форми аналітичного запису: диз'юнктивна нормальна форма і кон'юнктивна нормальна форма.

Для подання цих форм необхідно ввести низку нових понять.

Нехай на фіксованій множині змінних $X = \{x_1, x_2, ..., x_n\}$ задана логічна функція $f(x_1, x_2, ..., x_n)$.

Елементарною кон'юнкцією називається логічний добуток

$$k = \tilde{x}_{i_1} \tilde{x}_{i_2} ... \tilde{x}_{i_r}$$

змінних з множини X, у якому всі $\tilde{x}_{i_j}$ $(i = 1,2,...,n,\ j = 1,2,...,r)$ різні. Запис $\tilde{x}_{i_j}$ означає, що $\tilde{x}_{i_j} = x_{i_j}$ або $\tilde{x}_{i_j} = \overline{x}_{i_j}$. Число r називається рангом кон'юнкції.

Наприклад, елементарними кон'юнкціями множини $\{x_1, x_2, x_3\}$ є вирази $\overline{x}_1$, x_2, $x_1 x_2$, $\overline{x}_1 \overline{x}_2 x_3$, а вирази $x_1 x_2 x_1$, $\overline{x}_1 x_1$, $\overline{x_1 x_2}$ елементарними кон'юнкціями не є.

Елементарна кон'юнкція, що містить усі змінні з множини X, називається мінтермом або конституентою одиниці:

$$K = \tilde{x}_1 \tilde{x}_2 ... \tilde{x}_n .$$

Інакше кажучи, мінтерм – це елементарна кон'юнкція рангу n. Легко бачити, що для фіксованої множини змінних $x_1, x_2, ..., x_n$ буде стільки різних мінтермів, скільки є двійкових наборів з n компонентів, тобто 2^n.

Для двох змінних існує чотири мінтерми:

$$K_0 = \overline{x}_1 \overline{x}_2 ;$$

$$K_1 = \overline{x}_1 x_2 ;$$

$$K_2 = x_1 \overline{x}_2 ;$$

$$K_3 = x_1 x_2 ,$$

які також можна подати у вигляді таблиці (табл. 1.8).

Таблиця 1.8

x_1	x_2	K_0	K_1	K_2	K_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

Цей приклад дає змогу сформулювати таке визначення: мінтермом називають функцію, яка набуває одиничного значення на одному із усіх можливих наборів значень аргументів і нульового – у разі усіх інших наборів.

Диз'юнктивною нормальною формою (ДНФ) називають диз'юнкцію

$$k_1 + k_2 + ... + k_s$$

елементарних кон'юнкцій k_i $(i = 1,2,...,s)$, у якій усі k_i різні.

Існує алгоритм, який дає можливість перетворити будь-яку формулу булевої алгебри на ДНФ.

На першому етапі цього алгоритму формула перетворюється на рівносильну, побудовану із змінних та їх заперечень за допомогою самих лише кон'юнкцій та диз'юнкцій (тобто заперечення можуть стояти лише над окремими змінними). Для цього використовуються закони де Моргана та закон подвійного заперечення.

На другому етапі добиваються того, щоб усі кон'юнкції виконувались раніше від диз'юнкцій, для чого розкриваються дужки на основі дистрибутивного закону для кон'юнкції. Далі на основі співвідношень для констант (законів універсальної і нульової множин) і закону доповняльності проводять відповідні спрощення і на основі законів тавтології об'єднують однакові члени.

Приклад

$$\overline{x_1 + x_2}(x_1 \to x_3) = \overline{x_1 + x_2}(\bar{x}_1 + x_3) = \bar{x}_1\bar{x}_2(\bar{x}_1 + x_3) =$$
$$= \bar{x}_1\bar{x}_2\bar{x}_1 + \bar{x}_1\bar{x}_2x_3 = \bar{x}_1\bar{x}_2 + \bar{x}_1\bar{x}_2x_3.$$

Логічна функція може мати декілька ДНФ.

Приклад

$$x_1x_3 + x_2\bar{x}_3 + x_1x_2 = x_1x_3 + x_2\bar{x}_3,$$

що підтверджується такими перетвореннями:

$$x_1x_3 + x_2\bar{x}_3 + x_1x_2 = x_1x_3(x_2 + \bar{x}_2) + x_2\bar{x}_3(x_1 + \bar{x}_1) + x_1x_2(x_3 + \bar{x}_3) =$$

$$= x_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 + x_1x_2x_3 + x_1x_2\bar{x}_3 =$$

$$= x_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 =$$

$$= x_1x_3(x_2 + \bar{x}_2) + x_2\bar{x}_3(x_1 + \bar{x}_1) = x_1x_3 + x_2\bar{x}_3.$$

Досконалою диз'юнктивною нормальною формою (ДДНФ) називається ДНФ, в якій кожна елементарна кон'юнкція є мінтермом.

Довільна булева функція $f(x_1, x_2, ..., x_n) \neq 0$ має єдину ДДНФ.

ДДНФ функції збігається з диз'юнкцією мінтермів, які відповідають наборам аргументів, за яких ця функція має одиничні значення. Загалом ДДНФ можна записати так:

$$f(x_1, x_2, ..., x_n) = \sum_{i=0}^{2^n-1} f_i K_i, \qquad (1.7)$$

де f_i і K_i – відповідно значення функції і мінтерм на *i*-му наборі аргументів. Кожний член суми (1.7) набуває одиничного значення лише на наборі аргументів, якому відповідають значення $f_i = K_i = 1$.

ДДНФ може бути побудована на основі таблиці істинності під час використання таких правил. Записують ті набори аргументів, для яких функція дорівнює одиниці. Для кожного такого набору складають мінтерм. Змінні, що дорівнюють нулю, замінюють на інверсні значення. Одержані мінтерми з'єднують знаками диз'юнкції.

Приклад

ДДНФ для функції, що задана у табл. 1.6, має вигляд:

$$y = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 + \bar{x}_1x_2\bar{x}_3 + x_1\bar{x}_2\bar{x}_3 + x_1x_2x_3.$$

Будь-яку ДНФ можна звести до ДДНФ. Для цього використовують доповнення типу $x_i + \bar{x}_i = 1$.

Приклад

$$x_1 + \bar{x}_1 x_2 + x_1 \bar{x}_2 x_3 = x_1(x_2 + \bar{x}_2)(x_3 + \bar{x}_3) + \bar{x}_1 x_2 (x_3 + \bar{x}_3) + x_1 \bar{x}_2 x_3 =$$
$$= x_1 x_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 \bar{x}_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + \bar{x}_1 x_2 \bar{x}_3.$$

Введемо тепер низку понять, які характеризують кон'юнктивну нормальну форму.

Елементарною диз'юнкцією називається логічна сума

$$d = \tilde{x}_{i_1} + \tilde{x}_{i_2} + ... + \tilde{x}_{i_r}$$

змінних із множини $X = \{x_1, x_2, ..., x_n\}$, в якій усі $\tilde{x}_{i_j}$ ($i = 1,2,...,n$, $j = 1,2,...,r$) різні. Число r називається рангом диз'юнкції.

Наприклад, елементарними диз'юнкціями множини $\{x_1, x_2, x_3\}$ є вирази $x_1 + x_2 + \bar{x}_3$, $x_1 + \bar{x}_2$, x_3, а вирази $x_1 + \bar{x}_2 + x_1$ елементарними диз'юнкціями не є.

Елементарна диз'юнкція, що містить усі змінні з множини X, називається макстермом чи конституентою нуля:

$$D = \tilde{x}_1 + \tilde{x}_2 + ... + \tilde{x}_n.$$

Інакше кажучи, макстерм – це елементарна диз'юнкція рангу n. Максимальна кількість макстермів для множини $x_1, x_2, ..., x_n$ дорівнює 2^n.

Для двох змінних існує чотири макстерми:

$$D_0 = x_1 + x_2;$$
$$D_1 = x_1 + \bar{x}_2;$$
$$D_2 = \bar{x}_1 + x_2;$$
$$D_3 = \bar{x}_1 + \bar{x}_2,$$

які також можна подати у вигляді таблиці (табл. 1.9).

Таблиця 1.9

x_1	x_2	D_0	D_1	D_2	D_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

Цей приклад дає змогу сформулювати таке визначення: макстермом називається функція, яка набуває нульового значення за одного з усіх можливих наборів аргументів і одиничного – за всіх інших наборів.

Кон'юнктивною нормальною формою (КНФ) називається кон'юнкція

$$d_1 \cdot d_2 \cdot \ldots \cdot d_s$$

елементарних диз'юнкцій d_i $(i-1,2,\ldots,s)$, в якій всі d_i різні.

Існує алгоритм, який дає змогу будь-яку формулу булевої алгебри записати у КНФ. Перший етап цього алгоритму такий самий, як і для ДНФ.

На другому етапі добиваються, щоб усі диз'юнкції виконувались раніше від кон'юнкцій. Для цього використовують дистрибутивний закон

$$x_1 + x_2x_3 = (x_1 + x_2)(x_1 + x_3)$$

або його наслідок

$$x_1x_2 + x_3x_4 = (x_1 + x_3)(x_1 + x_4)(x_2 + x_3)(x_2 + x_4).$$

Далі, на основі співвідношень для констант (законів універсальної та нульової множин) і закону доповняльності для диз'юнкції проводять відповідні спрощення і на основі законів тавтології об'єднують рівні члени.

Приклад

$$\overline{x_1 + x_3} = (x_1 \to x_2) = \overline{x_1 + x_3}(\bar{x}_1 + x_2) = \bar{x}_1\bar{x}_3(\bar{x}_1 + x_2).$$

Логічна функція може мати декілька КНФ.

Приклад

$$(x_1 + x_2)(x_2 + x_3)(x_1 + \bar{x}_3) = (x_2 + x_3)(x_1 + \bar{x}_3).$$

Досконалою кон'юнктивною нормальною формою (ДКНФ) називається КНФ, у якій кожна елементарна диз'юнкція є макстермом.

Довільна булева функція $f(x_1, x_2, ..., x_n) \neq 1$ має єдину ДКНФ.

ДКНФ функції збігається з кон'юнкцією макстермів, що відповідають наборам аргументів, за яких ця функція має нульові значення. Загалом ДКНФ записується так:

$$f(x_1, x_2, ..., x_n) = \prod_{i=0}^{2^n-1} (f_i + D_i), \qquad (1.8)$$

де f_i і D_i – відповідно значення функції і макстерму на *i*-му наборі аргументів. Кожний член суми (1.8) набуває нульового значення лише під час набору аргументів, для яких $f_i = D_i = 0$.

ДКНФ може бути побудована на основі таблиці істинності в разі використання такого правила. Записуються ті набори змінних, для яких функція дорівнює нулю. Для кожного такого набору складається макстерм, при цьому змінні, що дорівнюють одиниці, замінюються на інверсні значення і одержані макстерми з'єднуються знаками кон'юнкції.

Приклад

ДКНФ для функції, що задана у табл. 1.6, має вигляд:

$$y = (x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + x_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + x_3).$$

Будь-яку КНФ можна звести до ДКНФ. Якщо у деяку елементарну диз'юнкцію не входить змінна x_i, потрібно записати рівносильний вираз

$$d_j = d_j + x_i \bar{x}_i$$

і скористатись дистрибутивним законом:

$$d_j + x_i \bar{x}_i = (d_j + x_i)(d_j + \bar{x}_i).$$

Приклад

$$(x_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2) = (x_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3\bar{x}_3) =$$
$$= (x_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3).$$

Канонічні форми запису функцій алгебри логіки у ДДНФ і ДКНФ еквівалентні. Використовуючи закони алгебри логіки, одну з цих форм завжди можна перетворити на іншу.

Приклад

Перетворення ДДНФ на ДКНФ:

$$y = \bar{x}_1\bar{x}_2 + x_1x_2 = \bar{x}_1\bar{x}_2 + \underbrace{x_1\bar{x}_1}_{0} + x_1x_2 + \underbrace{x_2\bar{x}_2}_{0} =$$

$$= \bar{x}_1(x_1 + \bar{x}_2) + x_2(x_1 + \bar{x}_2) = (x_1 + \bar{x}_2)(\bar{x}_1 + x_2).$$

Перетворення ДКНФ на ДДНФ:

$$y = (x_1 + x_2)(\bar{x}_1 + \bar{x}_2) = \underbrace{x_1\bar{x}_1}_{0} + x_1\bar{x}_2 + \bar{x}_1x_2 + \underbrace{x_2\bar{x}_2}_{0} = x_1\bar{x}_2 + \bar{x}_1x_2.$$

Координатна форма

Координатний спосіб зображення логічних функцій – це за своєю суттю певна форма таблиці істинності. Найпоширенішою координатною формою є так звані карти Карно.

Карта Карно – це прямокутна таблиця, що розділена вертикальними і горизонтальними лініями на клітинки, загальна кількість яких для функції від n аргументів дорівнює 2^n. Кожній клітинці відповідає певний мінтерм, розміщення останніх відбувається так, що у сусідніх клітинках знаходяться суміжні мінтерми. Суміжними називаються мінтерми, що відрізняються формою входження тільки однієї змінної: в один із них ця змінна входить у прямій формі, в інших – в інверсній. Сусідніми вважаються клітинки, що мають спільні сторони, а також клітинки, що розташовані на протилежних краях одних і тих самих рядків чи стовпців. Наведені правила є справедливими для логічних функцій, що мають не більше від чотирьох аргументів. Для функцій від більшого числа аргументів під час побудови карт Карно необхідно враховувати деякі доповнення до цих правил, про що йтиметься в підрозділі 1.8 – мінімізація булевих функцій.

Відповідність клітинок карти Карно певним мінтермам для логічної функції чотирьох змінних показана на рис. 1.4, *а*.

Звичайно мінтерми не записують у клітинки карти Карно, а по її краях здійснюють позначення рядків і стовпців, які відповідають прямим (неінвертованим) значенням аргументів так, як це показано на рис. 1.4, *б*.

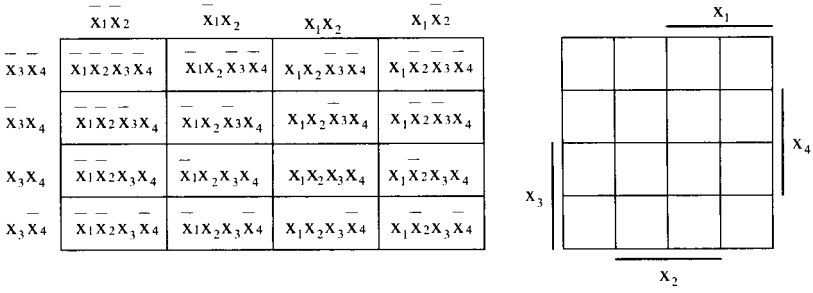


Рис. 1.4

На рис 1.5, *а–в* наведені карти Карно для логічних функцій відповідно двох, трьох і п'яти змінних.

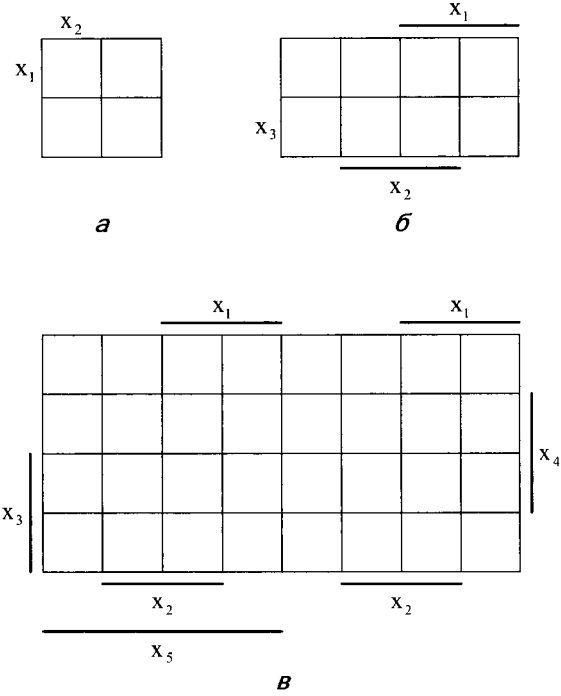


Рис. 1.5

Наведені на рис. 1.4 і 1.5 карти Карно не є єдино можливими. Можливі інші варіанти їхньої побудови, за інших позначень їхніх рядків і стовпців. Але для будь-яких варіантів потрібно виконувати правило: в сусідніх клітинках повинні знаходитись суміжні мінтерми.

У кожну клітинку карти Карно записуються значення функції на відповідному цій клітинці наборі значень аргументів. Як правило, записуються тільки одиничні значення, а нульові – не записуються.

Приклад

Карта Карно для функції, що задана в табл. 1.6, має вигляд, показаний на рис. 1.6.

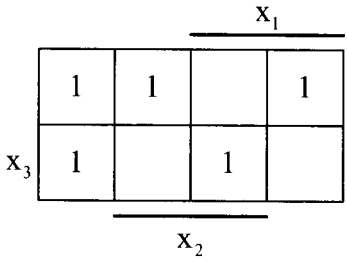


Рис. 1.6

Якщо карта Карно складається на основі аналітичного запису функції, тоді не обов'язково записувати цю функцію в ДДНФ, достатньо знати набори аргументів, яким відповідають одиничні значення функції. Якщо функція задана у вигляді якої-небудь формули, то ці набори визначаються методом перебору аргументів (карта Карно забезпечує, по суті, розгортання формули в ДДНФ).

1.8. Мінімізація булевих функцій

Спрощення формул алгебри логіки має на меті отримання такого їхнього вигляду, за якого побудована згідно з нею схема характеризувалась би мінімальною кількістю логічних елементів – була б найпростішою. Розв'язання такої задачі у найзагальнішому вигляді (для різних базисів і з урахуванням конкретних особливостей вико-

ристовуваних логічних елементів) є складною задачею, що потребує надзвичайно складних і громіздких розрахунків. Сьогодні найдетальніше розроблені методи пошуку мінімальної ДНФ, що у багатьох випадках є достатнім для оптимального спрощення цифрових пристроїв.

Мінімальною ДНФ булевої функції називається ДНФ цієї функції, що складається з найменш можливої кількості букв. За підрахунку кількості букв кожна буква враховується стільки разів, скільки вона трапляється в ДНФ. Наприклад, ДНФ $\bar{x}y + x\bar{y} + xz$ складається з шести букв.

Мінімізація – це знаходження мінімальної ДНФ.

Існує декілька методів мінімізації. Більшість з них описуються чіткими алгоритмами і піддаються програмуванню з використанням ЕОМ, що необхідно за великої кількості аргументів.

Найвідомішими методами мінімізації є:

– метод безпосередніх перетворень;

– метод Квайна;

– метод карт Карно.

Метод безпосередніх перетворень

Цей метод є аналітичним методом спрощення логічних функцій за допомогою законів булевої алгебри. У разі набуття певного досвіду цей метод є доволі ефективним для невеликої кількості змінних.

Приклад

$$y = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2x_3 =$$

$$= \bar{x}_2\bar{x}_3(\bar{x}_1 + x_1) + \bar{x}_2x_3(\bar{x}_1 + x_1) + x_2x_3(x_1 + \bar{x}_1) = \bar{x}_2\bar{x}_3 + \bar{x}_2x_3 + x_2x_3 =$$

$$= \bar{x}_2\bar{x}_3 + \bar{x}_2x_3 + \bar{x}_2x_3 + x_2x_3 = \bar{x}_2(\bar{x}_3 + x_3) + x_3(x_2 + \bar{x}_2) = \bar{x}_2 + x_3.$$

Метод Квайна

Вихідною (початковою) формою функції в разі мінімізації за методом Квайна є ДДНФ. Пошук мінімальної ДНФ відбувається в два етапи.

Результатом виконання першого етапу є одержання скороченої ДНФ функції. Основними поняттями, що використовуються

на цьому етапі, є поняття суміжних мінтермів і імплікант. Основною операцією спрощення є операція склеювання.

Нагадаємо, що суміжними називаються мінтерми, що відрізняються формою входження в них лише одного аргументу (в один з мінтермів цей аргумент входить в прямій формі, а в другий – в інверсній). Наприклад, суміжними є мінтерми $x_1x_2\bar{x}_3x_4$ і $\bar{x}_1x_2\bar{x}_3x_4$. Два суміжні мінтерми склеюються по аргументу, що відрізняється. Це приводить до заміни їх однією кон'юнкцією з числом аргументів на одиницю меншим, ніж у початкових мінтермів.

Наприклад:

$$x_1x_2\bar{x}_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 = x_2\bar{x}_3x_4;$$

$$x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 = x_1x_2\bar{x}_3;$$

$$\bar{x}_1x_2\bar{x}_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4 = \bar{x}_1x_2\bar{x}_3.$$

Кон'юнкції, що формуються в результаті склеювання суміжних мінтермів, називаються імплікантами.

Різні імпліканти з однаковим числом аргументів, своєю чергою, можуть бути суміжними, що дасть змогу виконувати їх склеювання між собою. У наведеному прикладі

$$x_1x_2\bar{x}_3 + \bar{x}_1x_2\bar{x}_3 = x_2\bar{x}_3.$$

Імпліканта $x_2\bar{x}_3$ еквівалентна чотирьом початковим мінтермам. Еквівалентність полягає в тому, що значення, яких набувають імпліканта $x_2\bar{x}_3$ і диз'юнкція чотирьох початкових мінтермів, на усіх можливих наборах аргументів $x_1,\ldots,x_4$, збігаються. Кажуть, що імпліканта покриває усі мінтерми, в результаті склеювання яких вона отримана. Для наочності зображення порядку покриття, мінтерми і імпліканти можна записувати так:

$$\left.\begin{array}{l}\left.\begin{array}{l}x_1x_2\bar{x}_3\bar{x}_4\\ x_1x_2\bar{x}_3x_4\end{array}\right\}x_1x_2\bar{x}_3\\ \left.\begin{array}{l}\bar{x}_1x_2\bar{x}_3\bar{x}_4\\ \bar{x}_1x_2\bar{x}_3x_4\end{array}\right\}\bar{x}_1x_2\bar{x}_3\end{array}\right\}x_2\bar{x}_3.$$

Процес багатоступеневого склеювання приводить до одержання імплікант, які не склеюються з іншими. Такі імпліканти називаються простими. Разом з початковими мінтермами, які не мали суміжних їм для склеювання, прості імпліканти входять до складу результуючої диз'юнктивної форми функції, яка називається скороченою ДНФ. Якщо початковий мінтерм, що не склеюється, так само вважати простою імплікантою, то можна сформулювати таке визначення: диз'юнкція усіх простих імплікант булевої функції називається скороченою ДНФ цієї функції.

Скорочена ДНФ булевої функції єдина, оскільки множина усіх простих імплікант булевої функції визначається однозначно.

Наведемо приклад отримання скороченої ДНФ:

$$y = x_1x_2x_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3$$

$$y = x_2x_3 + x_1x_3 + x_1\bar{x}_2$$

Тут стрілками позначені пари мінтермів, що склеюються.

Другий етап полягає в знаходженні мінімальної ДНФ на основі скороченої ДНФ заданої функції.

Спочатку знаходять так звані тупикові ДНФ, а потім з них вибирають мінімальну (мінімальні) ДНФ. Тупикові ДНФ булевої функції отримують із скороченої ДНФ цієї функції вилученням деяких елементарних кон'юнкцій.

ДНФ функції називається тупиковою, якщо:

– кожна її елементарна кон'юнкція є простою імплікантою;

– вилучення з неї будь-якої елементарної кон'юнкції приводить до ДНФ, яка вже не відповідає заданій функції.

Логічна функція може мати декілька тупикових і декілька мінімальних ДНФ.

Тупикову ДНФ можна знайти із скороченої ДНФ за допомогою імплікантної таблиці Квайна, що є прямокутною таблицею, в якій рядки позначаються різними простими імплікантами логічної

функції, а стовпці мінтермами ДДНФ цієї функції. На перетині відповідних рядків і стовпців ставиться позначка тоді, коли відповідна проста імпліканта покриває відповідний мінтерм.

Імплікантна таблиця для розглянутого вище прикладу наведена нижче (табл. 1.10).

Таблиця 1.10

Прості імпліканти	Мінтерми			
	$x_1x_2x_3$	$\bar{x}_1x_2x_3$	$x_1\bar{x}_2x_3$	$\bar{x}_1\bar{x}_2x_3$
x_2x_3	x	x		
x_1x_3	x		x	
$x_1\bar{x}_2$			x	x

Тупикову ДНФ знаходять з імплікантної таблиці за допомогою диз'юнктивного об'єднання тих простих імплікант, які забезпечують повноту покриття усіх мін-термів ДДНФ.

У прикладі, що розглядається, повне покриття можна забезпечити простими імплікантами x_2x_3 і $x_1\bar{x}_2$, а імпліканта x_1x_3 є зайвою. Так, тупиковою ДНФ є функція

$$y = x_2x_2 + x_1\bar{x}_2.$$

У цьому разі тупикова ДНФ є єдиною, і тому вона одночасно є мінімальною ДНФ. Якщо б тупикових ДНФ було декілька, потрібно було б серед них вибрати мінімальну (мінімальні) ДНФ.

Метод карт Карно

Метод мінімізації за допомогою карт Карно ґрунтується на поданні логічної функції у вигляді карти Карно.

Нагадаємо, що сусідні клітинки карт Карно відповідають суміжним мінтермам. Склеювання суміжних мінтермів, знаходження простих імплікант, пошук тупикових і мінімальних форм проводяться на основі аналізу розподілу одиничних значень мінтермів у клітинках карти.

Мінтерми, що склеюються, охоплюються контурами (незамкненими, якщо мінтерми знаходяться в крайніх клітинках). Інакше кажучи, клітинки, що містять 1, залучаються до певних об'єднань.

Об'єднання формуються за такими правилами:

– в об'єднання може входити 2^k клітинок з одиницями ($k = 0, 1, 2, 3, \ldots$);

– об'єднання повинні мати прямокутну форму, або можуть бути з розривами, з урахуванням того, що суміжним мінтермам відповідають також клітинки, що розташовані по краях одних і тих самих рядків чи стовпців;

– об'єднання можуть перетинатись, тобто ті самі клітинки можуть входити в різні об'єднання;

– об'єднаннями повинні бути охоплені всі клітинки з одиницями;

– не повинно бути об'єднань, у яких усі клітинки з одиницями входять в інші об'єднання;

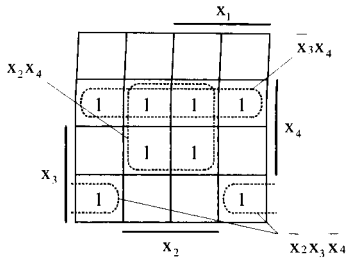
– об'єднань повинно бути якнайменше, а самі вони повинні бути якнайбільшими.

Для логічних функцій, що мають більше від чотирьох аргументів, існують певні доповнення до вказаних правил, які розглядатимуться нижче.

Деякі можливі випадки формування об'єднань наведені на рис. 1.7.

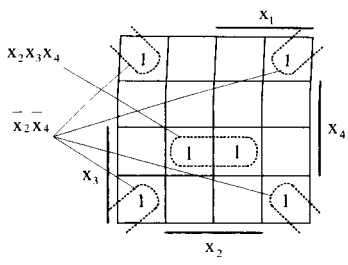
Зчитування спрощеної форми з карти Карно полягає у визначенні імплікант, що отримуються в результаті склеювання. Необхідно дотримуватись правила: кожному об'єднанню відповідає одна імпліканта, яка складається лише з тих аргументів, які для усіх клітинок об'єднання мають однакову форму – пряму чи інверсну (рис. 1.7). Отже, для отримання якнайпростішого результуючого виразу слід обов'язково дотримуватись останнього з вищенаведених правил формування об'єднань клітинок карт Карно.

На рис. 1.8 і 1.9 наведено приклади мінімізації логічних функцій від п'яти і шести змінних відповідно.



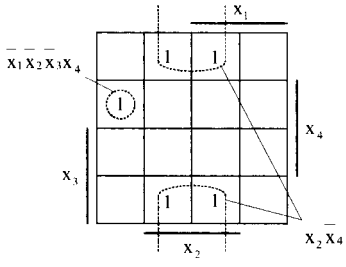
$y = x_2x_4 + \bar{x}_3x_4 + \bar{x}_2x_3\bar{x}_4$

а



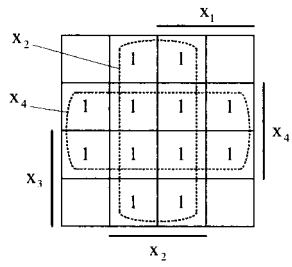
$y = \bar{x}_2\bar{x}_4 + x_2x_3x_4$

б



$y = x_2\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4$

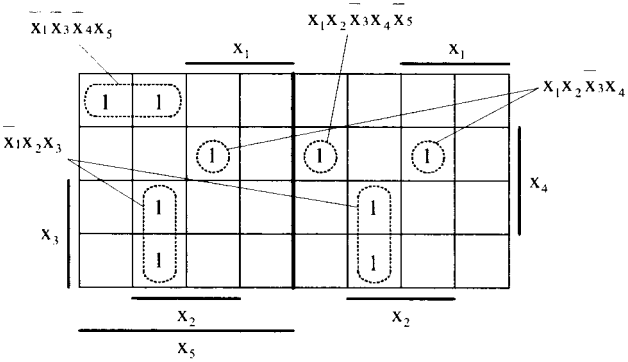
в



$y = x_2 + x_4$

г

Рис. 1.7



$y = x_1x_2\bar{x}_3x_4\bar{x}_5 + x_1x_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_3\bar{x}_4x_5 + \bar{x}_1x_2x_3$

Рис. 1.8

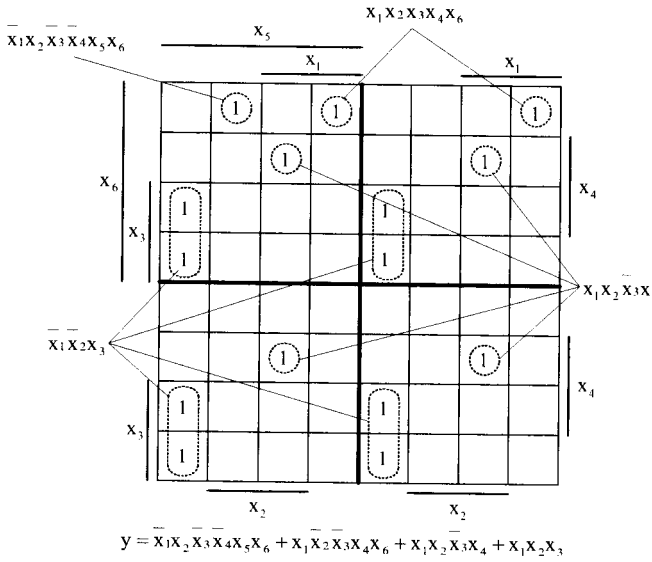
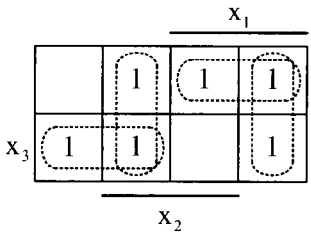


Рис. 1.9

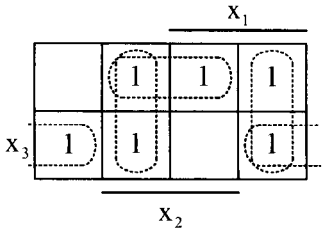
Карти Карно, що зображені на цих рисунках, можна вважати такими, що складаються з кількох часткових карт для чотирьох змінних (двох – для функції п'яти змінних і чотирьох – для функції шести змінних). У межах цих часткових карт правила формування об'єднань відповідають вищенаведеним. Окрім цього, об'єднання можуть формуватись з окремих частин, розташованих на однакових місцях часткових карт, оскільки їм відповідають суміжні мінтерми. Визначення імплікант відбувається так, як це було описано вище.

З використанням наведеного принципу можна також побудувати карти Карно для логічних функцій від більшого числа аргументів і мінімізувати ці функції.

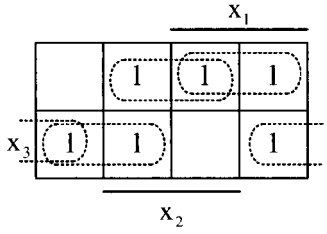
Мінімізація за допомогою карт Карно, як і з використанням інших методів, може приводити до різних варіантів відповіді. Це пояснюється, як тим, що логічна функція може мати кілька мінімальних ДНФ, так і не оптимальним способом розв'язання задачі. На рис. 1.10 наведено приклад різних варіантів мінімізації функції трьох змінних. Усі наведені форми ДНФ є тупиковими, а останні дві – мінімальними.



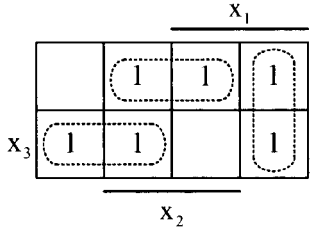
$$y = x_3\bar{x}_1 + x_2\bar{x}_1 + x_1\bar{x}_3 + x_1\bar{x}_2$$



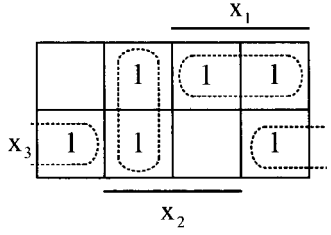
$$y = x_3\bar{x}_2 + x_2\bar{x}_1 + \bar{x}_3x_2 + x_1\bar{x}_2$$



$$y = x_3\bar{x}_2 + x_3\bar{x}_1 + x_2\bar{x}_3 + x_1\bar{x}_3$$



$$y = x_3\bar{x}_1 + x_2\bar{x}_3 + x_1\bar{x}_2$$



$$y = x_3\bar{x}_2 + x_2\bar{x}_1 + x_1\bar{x}_3$$

Рис. 1.10

Порівнюючи методи Квайна і Карно, можна зауважити, що останній не є систематичним. Процес мінімізації згідно з цим методом залежить від евристичних міркувань того, хто виконує спрощення.

Мінімізація частково визначених функцій

У практичних задачах синтезу цифрових пристроїв часто буває так, що значення логічної функції не визначені на деяких наборах аргументів і можуть довизначатись довільно. У цьому випадку довизначення функції доцільно виконувати так, щоб отримати найпростішу ДНФ з усіх можливих варіантів.

Мінімізацію розглянемо на прикладі функції, що задана таблицею істинності (табл. 1.11), де невизначені значення позначені символом $*$.

Таблиця 1.11

x_1	x_2	x_3	y	y′
0	0	0	*	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	*	1
1	1	0	*	1
1	1	1	*	1

На рис. 1.11, *а* наведено карту Карно заданої функції, а на рис. 1.11, *б* – карту Карно довизначеної функції y', в якій три її значення довизначені значенням 1, а одне – значенням 0 (довизначені значення також наведені в табл. 1.11). Відповідно до утворених об'єднань маємо:

$$y = x_1 + x_2 x_3 .$$

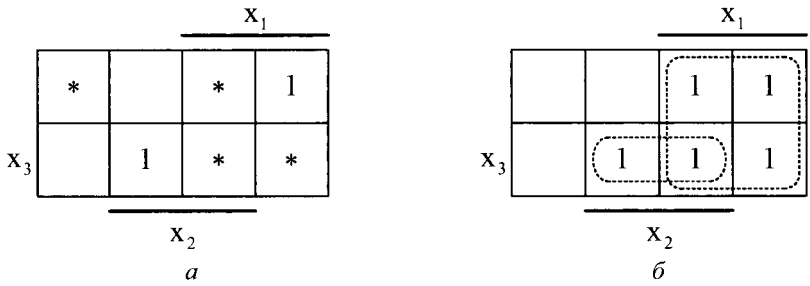


Рис. 1.11

1.9. Запис логічних функцій у базисах функцій І-НЕ та АБО-НЕ

Нехай логічна функція задана в диз'юнктивній нормальній формі:

$$y = k_1 + k_2 + ... + k_s,$$

де

$$k_i = \tilde{x}_{i_1} \tilde{x}_{i_2} ... \tilde{x}_{i_r} \quad (i = 1,2,...,s)$$

– елементарні кон'юнкції.

Використавши закон Моргана, отримаємо:

$$\bar{y} = \overline{k_1 + k_2 + ... + k_s} = \bar{k}_1 \cdot \bar{k}_2 \cdot ... \cdot \bar{k}_s,$$

$$= \quad = \quad = \quad =$$
$$y = y = \overline{k_1 \cdot k_2 \cdot ... \cdot k_s}.$$

Так, для того, щоб записати логічну функцію в базисі функції І-НЕ, потрібно два рази проінвертувати її диз'юнктивну нормальну форму.

Приклад

$$y = x_2 x_3 + x_1 \bar{x}_2,$$

$$= \quad = \quad = \quad =$$
$$y = y = \overline{x_2 x_3 \cdot x_1 \bar{x}_2}.$$

Для подання логічної функції в базисі функції АБО-НЕ введемо змінну

$$g_i = \overline{k_i} = \overline{\tilde{x}_{i_1}\tilde{x}_{i_2}...\tilde{x}_{i_r}} = \overline{\tilde{x}_{i_1}} + \overline{\tilde{x}_{i_2}} + ... + \overline{\tilde{x}_{i_r}} .$$

Виконавши нескладні перетворення з використанням законів алгебри логіки, маємо:

$$\equiv \quad \equiv \quad \equiv$$
$$y = y = \overline{k_1 \cdot k_2 \cdot ... \cdot k_s} = \overline{g_1 \cdot g_2 \cdot ... \cdot g_s} = \overline{g_1} + \overline{g_2} + ... + \overline{g_s} .$$

Отже, для того щоб записати логічну функцію в базисі функції АБО-НЕ, потрібно чотири рази проінвертувати її диз'юнктивну нормальну форму.

Приклад

$$y = x_2x_3 + x_1\overline{x_2},$$
$$\overline{y} = \overline{x_2x_3} \cdot \overline{x_1x_2} = (\overline{x_2} + \overline{x_3})(\overline{x_1} + x_2),$$
$$= \quad \equiv$$
$$y = \overline{(\overline{x_2} + \overline{x_3})} + \overline{(\overline{x_1} + x_2)},$$
$$\equiv \quad \equiv$$
$$y = y = \overline{(\overline{x_2} + \overline{x_3})} + \overline{(\overline{x_1} + x_2)}.$$

Питання і задачі для самоконтролю

1. Скільки значень можуть набувати логічні аргументи і функції?
2. Сформулюйте визначення логічним функціям інверсії, диз'юнкції та кон'юнкції.
3. На основі чого доводяться закони булевої алгебри?
4. Скільки існує різних логічних функцій двох змінних?
5. Сформулюйте визначення логічної функції І-НЕ для n змінних.
6. Сформулюйте визначення логічної функції АБО-НЕ для n змінних.
7. Наведіть таблицю істинності для логічної функції "диз'юнкція" від трьох змінних.
8. Наведіть таблицю істинності для логічної функції "кон'юнкція" від трьох змінних.
9. Наведіть таблицю істинності для логічної функції "сума за модулем 2".
10. Запишіть логічну функцію $y = x_1 + \overline{x_1}x_2x_3$ у досконалій диз'юнктивній нормальній формі.
11. Запишіть логічну функцію $y = x_1 + \overline{x_1}x_2x_3$ в досконалій кон'юнктивній нормальній формі.
12. Зобразіть функцію $y = x_1 + \overline{x_1}x_2x_3$ у формі карти Карно.
13. Мінімізуйте логічну функцію
$$y = \overline{x_1}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3 + x_1\overline{x_2}\,\overline{x_3} + x_1\overline{x_2}\,\overline{x_3} + x_1\overline{x_2}x_3.$$
14. Запишіть логічну функцію $y = x_1x_3 + \overline{x_1}x_2x_3 + \overline{x_1}\,\overline{x_2}$ в базисі функції І-НЕ.
15. Запишіть логічну функцію $y = x_1x_3 + \overline{x_1}x_2x_3 + \overline{x_1}\,\overline{x_2}$ в базисі функції АБО-НЕ.

2. ЦИФРОВІ ПРИСТРОЇ

2.1. Класифікація цифрових пристроїв

Пристрої цифрової техніки можна поділити на два класи:

– пристрої формування, генерування та перетворення сигналів з певними заданими параметрами;

– пристрої перетворення логічних сигналів і цифрових кодів.

До першого класу належать формувачі імпульсів (одновібратори, обмежувачі амплітуди, компаратори ...), тактові генератори (мультивібратори, релаксатори ...), аналого-цифрові перетворювачі, цифро-аналогові перетворювачі, перетворювачі частоти на код та інші.

До другого класу цифрових пристроїв, які називаються цифровими автоматами (скінченними цифровими автоматами), входять пристрої від найпростіших логічних елементів до мікропроцесорів. Тут перетворення інформації, що надходить у цифровій формі, здійснюється виконанням певної послідовності арифметичних та логічних операцій.

Треба зауважити, що перший клас пристроїв часто зараховують до імпульсної техніки і під цифровими пристроями розуміють тільки другий клас – цифрові автомати.

Розглянемо детальніше цифрові автомати. Розглядатимемо двійкові цифрові автомати, тобто такі, які оперують сигналами, що набувають тільки двох значень – 0 і 1.

На вхід цифрового автомата (ЦА) подається множина двійкових змінних X (елементи множини $x_1, x_2, ..., x_n$), з виходу знімається множина двійкових змінних Y (елементи множини $y_1, y_2, ..., y_s$) (рис. 2.1).

Рис. 2.1. Узагальнена схема цифрового автомата

Автомат реалізує деякий оператор перетворення вхідних змінних на вихідні

$$Y = \lambda(X). \qquad (2.1)$$

Вхідні, внутрішні та вихідні сигнали ЦА змінюються, як правило, не будь-коли, а в певні моменти часу, які визначають так звані такти роботи автомата. Дискретність тактів дають змогу вести їх нумерацію і використовувати номери тактів як абстрактний (автоматний, машинний) час. Абстрактний час позначатимемо символом t і вважатимемо, що $t = 0, 1, 2, 3, \ldots$.

Характер зміни вхідних і вихідних змінних, з урахуванням їх зміни в машинному часі, дає змогу поділити ЦА на дві групи:

- комбінаційні пристрої;
- послідовнісні пристрої.

У комбінаційних пристроях значення вихідних змінних у кожний момент машинного часу Y^t (протягом кожного такту t) однозначно визначаються значеннями вхідних змінних X^t у той самий момент:

$$Y^t = \lambda(X^t). \qquad (2.2)$$

У послідовнісних пристроях значення вихідних змінних протягом поточного такту t визначаються значеннями вхідних змінних протягом цього такту, а також низкою значень вхідних змінних протягом певної скінченної кількості попередніх тактів:

$$Y^t = \lambda(X^t, X^{t-1}, \ldots, X^{t-k}). \qquad (2.3)$$

Реалізація такого оператора означає, що послідовнісні пристрої мають властивість запам'ятовування вхідних змінних. Пам'ять автомата може охоплювати довільну, але обов'язково скінченну кількість тактів.

Зміна рівнів сигналів під час переходу від одного такту до іншого відбувається за скінченний час, у зв'язку з чим розрізняють динамічну і статичну частину тактів. На рис. 2.2 ці частини позначені буквами *д* і *с* відповідно, а символами s^0 і s^1 – низький і високий рівні сигналу. Враховуючи це, треба зауважити, що зв'язок

між вихідними і вхідними змінними ЦА, який описується залежностями (2.2) чи (2.3), виконується тільки для статичних частин тактів, тобто для усталених режимів роботи пристрою, і не виконується протягом перехідних процесів – динамічних частин тактів.

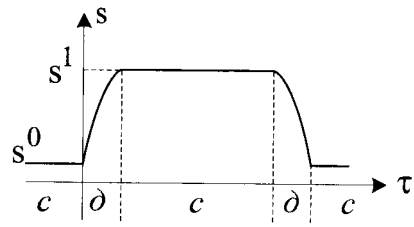


Рис. 2.2

ЦА, незалежно від складності функцій, які вони реалізують, виконуються на основі найпростіших (елементарних) комбінаційних і послідовнісних пристроїв – логічних елементів. Сукупність логічних елементів і взаємозв'язок між ними визначають структуру (структурну схему) автомата.

За характером передавання сигналів від одного цифрового пристрою до іншого розрізняють асинхронні та синхронні пристрої. В асинхронних пристроях зміна вхідних сигналів безпосередньо спричиняє зміну вихідних сигналів. У синхронних – вхідні сигнали сприймаються тільки за наявності додаткових тактових (синхронізуючих) сигналів (імпульсів). Комбінаційні пристрої, як правило, є асинхронними, а послідовнісні – можуть бути як асинхронними, так і синхронними.

Задача логічного проектування (синтезу) ЦА, що реалізує заданий оператор зв'язку вхідних і вихідних змінних, полягає у визначенні оптимальної структури пристрою за заданого переліку логічних елементів (заданої елементної бази).

До критерію оптимальності можуть входити такі показники: вартість, габарити, споживана потужність, швидкодія, надійність та інші.

Синтез ЦА полягає у побудові такого пристрою, який реалізує наперед задану довільно функцію "вхід–вихід". Синтез автомата можна поділити на декілька етапів:

– поблоковий синтез – передбачає поділ автомата на окремі блоки, для кожного з яких формується технічне завдання й установлюються функціональні зв'язки між блоками;

– абстрактний синтез – полягає у побудові абстрактного автомата, наприклад, його таблиці істинності, системи логічних рівнянь, графа, …;

– структурний синтез – полягає у розробленні структурної та принципової схем пристрою.

Цифрові пристрої слугують для оброблення інформації, що має форму цифрового коду. Цифровий код, що надходить на вхід ЦА є, як правило, результатом дискретизації та квантування неперервних сигналів, які здійснюються за допомогою аналого-цифрових перетворювачів. Вихідні коди цифрових пристроїв можуть перетворюватись на аналоговий сигнал за допомогою цифро-аналогових перетворювачів.

Цифрові пристрої, як правило, працюють з сигналами, що мають два рівні – високий і низький, які, своєю чергою, можуть бути позначені символами 1 і 0. Ці символи не можна розглядати як числа. Однак в обчислювальній техніці, де числову інформацію подають у двійковій системі числення, двійкові змінні (0 і 1) можна розглядати як елементи двійкового коду, тобто як цифри двійкової системи числення.

2.2. Комбінаційні пристрої

2.2.1. Принципи логічного проектування комбінаційних пристроїв

Під час абстрактного синтезу комбінаційного пристрою визначають перелік вхідних і вихідних двійкових (логічних) змінних і встановлюють однозначну відповідність між ними. Ця

відповідність відображається в табличній, аналітичній чи якійсь іншій формі подання логічних функцій. Таблиці, у яких відображаються зв'язки вхідних і вихідних змінних, називаються таблицями істинності. Аналітичну форму можна подати системою логічних функцій:

$$\begin{aligned} y_1 &= f_1(x_1, x_2, ..., x_n), \\ y_2 &= f_2(x_1, x_2, ..., x_n), \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ y_s &= f_s(x_1, x_2, ..., x_n), \end{aligned} \tag{2.4}$$

де $x_1, x_2, ..., x_n$ і $y_1, y_2, ..., y_s$ – вхідні і вихідні змінні комбінаційного пристрою відповідно.

Основні різновиди задач, що трапляються під час проектування комбінаційних пристроїв:

– проектування пристроїв з одним виходом;

– проектування пристроїв з багатьма виходами.

Типовий порядок проектування комбінаційних пристроїв з одним виходом містить такі етапи:

– кодування вхідних і вихідних змінних і перехід від словесного задання оператора до табличного;

– перехід від табличної форми оператора до аналітичної у вигляді ДДНФ;

– мінімізація ДДНФ;

– перехід від мінімальної ДНФ до мінімальної форми у заданому базисі;

– побудова схеми пристрою.

Абстрактний синтез пристрою вичерпується першим етапом, чотири наступні належать до структурного синтезу.

Як приклад, розглянемо задачу синтезу комбінаційного пристрою, що задається таким словесним описом. Маємо три сенсори, вихідні сигнали яких набувають двох значень – 0 чи 1 (наприклад, низький і високий рівень напруги). Необхідно, використовуючи логічні елементи І-НЕ (елементи "штрих Шефера"), забезпечити індикацію станів, коли одиничне значення набуває не менше від двох із трьох сигналів.

Послідовність розв'язання поставленої задачі є такою:

1. Вводимо позначення вхідних – x_1, x_2, x_3 і вихідної – у змінних. Будуємо таблицю істинності:

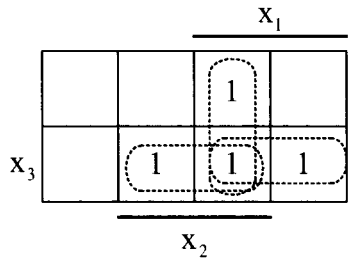
Таблиця 2.1

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

2. Записуємо ДДНФ:

$$y = \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3 .$$

3. Мінімізуємо ДДНФ методом карт Карно:

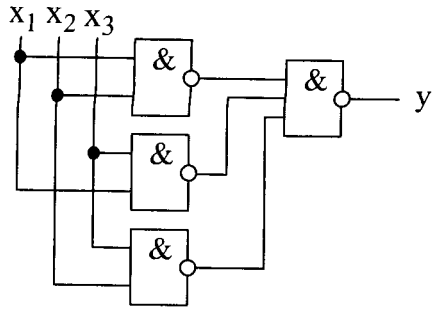


$$y = x_1 x_2 + x_1 x_3 + x_2 x_3 .$$

4. Виконавши подвійне інвертування, переходимо до запису логічної функції в базисі І-НЕ:

$$y = \overline{\overline{y}} = \overline{\overline{x_1 x_2} \cdot \overline{x_1 x_3} \cdot \overline{x_2 x_3}} .$$

5. Будуємо схему пристрою:



2.2.2. Негативні явища в комбінаційних пристроях

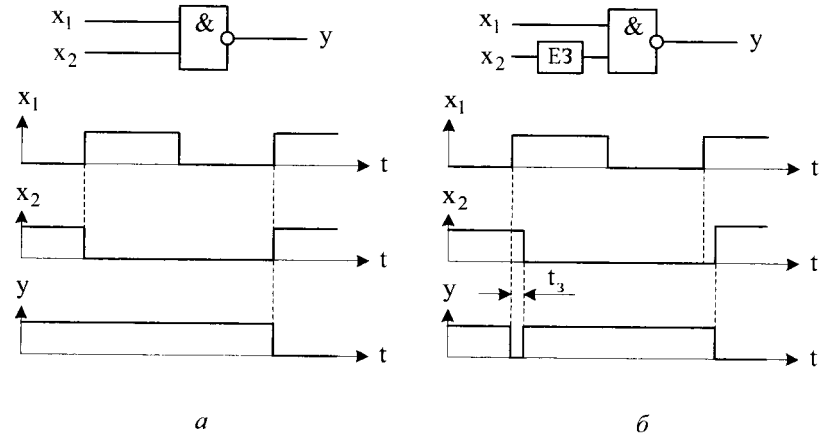
Під час проектування комбінаційних пристроїв потрібно враховувати певні особливості реальних комбінаційних пристроїв, які можуть спричинити негативні явища.

Якщо синтез комбінаційного пристрою здійснюється на функціонально-логічному рівні, можна припустити, що логічні елементи, інші функціональні вузли працюють без затримок і на їхніх виходах сигнали з'являються у той самий момент часу, що й вхідні сигнали, окрім того, – сигнали на виходах зберігаються доти, доки є сигнали на входах. Таке проектування, безумовно, є ідеалізованим і може бути корисним тільки для з'ясування принципових питань синтезу.

Під час проектування реальних цифрових пристроїв необхідно враховувати, що кожний логічний елемент чи функціональний вузол комбінаційного пристрою має власну затримку і тому сигнали на виходах схеми з'являються тільки через певний інтервал часу після подавання вхідних сигналів. Це, безперечно, негативно впливає на функціонування усього пристрою, оскільки затримки в колах логічних елементів не тільки зменшують швидкодію, але й можуть призвести до формування хибних сигналів. Явище, що спричиняє появу на виходах комбінаційних пристроїв хибних сигналів, називається гонками.

Хибні сигнали, як правило, короткотривалі, але вони можуть спричинити неправильну роботу наступних функціональних вузлів, особливо в тому випадку, коли ці вузли послідовнісного типу.

На рис. 2.3, *а*, *б* наведено приклад виникнення хибного сигналу.



а *б*

Рис. 2.3

Часові діаграми на рис. 2.3 відрізняються тим, що сигнал x_2 затриманий у часі відносно сингалу x_1 на величину $t_з$. Ця затримка може бути спричинена затримкою в попередніх функціональних вузлах (логічних елементах) чи лініях зв'язку. На рис. 2.3, *б* цей факт узагальнено відображений наявністю елемента затримки ЕЗ. Затримка x_2 відносно x_1 і є причиною формування хибного сигналу – від'ємного імпульсу (рис. 2.3).

Основними методами усунення хибних сигналів, що виникають внаслідок гонок, є:

- вирівнювання затримок;
- зміна схеми комбінаційного пристрою;
- синхронізація роботи вузлів цифрового пристрою.

Перші два методи використовуються рідко, оскільки вирівнювання затримок ускладнюється залежністю параметрів ліній затримки від зовнішніх факторів (передовсім температури навко-

лишнього середовища і напруги живлення), а зміна схеми пристрою в багатьох випадках неможлива. Третій із перелічених методів дає змогу кардинально вирішити цю проблему – унеможливити формування хибних сигналів на виходах цифрових пристроїв.

2.2.3. Синтез комбінаційних пристроїв із багатьма виходами

Синтез комбінаційних схем з багатьма виходами може бути зведений, у принципі, до синтезу схем з одним виходом. Для цього достатньо кожну із функцій системи, що описує логіку роботи пристрою, реалізувати окремо. У такому разі отримаємо схему з s виходами, що складається з s незалежних комбінаційних схем. Однак така схема в загальному випадку не оптимальна, оскільки деякі її частини можуть повторюватись.

Розв'язання цієї задачі в найзагальнішому вигляді (для будь-яких логічних функцій і для будь-яких логічних елементів) через велику кількість варіантів схем є достатньо громіздким завданням. Однак, існують методи, що дають змогу, в багатьох випадках, спростити загальну схему. Розглянемо один з таких методів, що основується на спільній мінімізації системи логічних функцій.

Простою імплікантою системи логічних функцій

$$f_1, f_2, ..., f_s \qquad (2.5)$$

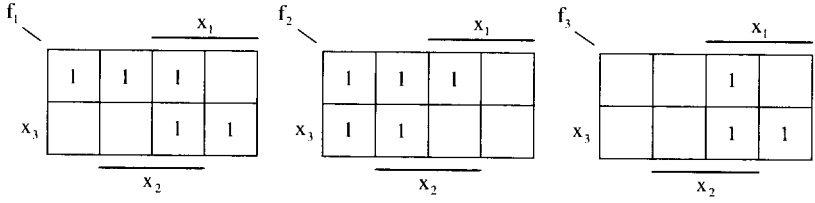
називають таку елементарну кон'юнкцію, яка є імплікантою кожної із функцій системи, але ніяка її власна частина не є імплікантою хоча б однієї з цих функцій.

На першому етапі спільної мінімізації знаходять усі прості імпліканти усіх можливих сукупностей функцій з цієї системи. Наприклад, якщо мінімізується система функцій (2.5), то спочатку знаходять усі прості імпліканти кожної функції системи (2.5). Потім із функцій системи утворюють усі можливі підсистеми, що складаються з двох функцій. Для кожної з отриманих підсистем функцій знаходять усі прості імпліканти. Потім утворюють усі можливі підсистеми з трьох функцій тощо. Зауважимо, що множина простих імплікант системи функцій $f_1, f_2, ..., f_s$ збігається з

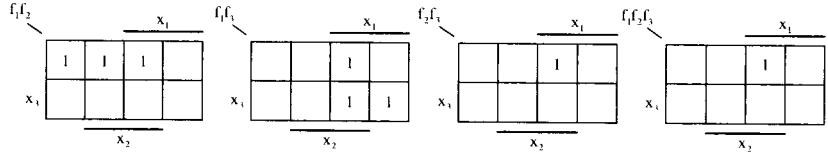
множиною простих імплікант функції вигляду $F = f_1 f_2 ... f_s$. Тому перший етап мінімізації системи функцій (2.5) зводиться до знаходження простих імплікант функцій f_1, f_2, … , f_s, $f_1 f_2$, $f_1 f_3$, … , $f_1 f_s$, $f_2 f_3$, … , $f_1, f_2, ..., f_s$, кількість яких дорівнює $2^s - 1$.

Приклад

Задана система функцій f_1, f_2, f_3:



Спочатку необхідно знайти усі прості імпліканти усіх можливих сукупностей функцій. Для цього утворимо усі можливі добутки функцій цієї системи:



Безпосередньо з одержаних карт Карно знаходимо прості імпліканти:

$$P_1^{(1)} = \bar{x}_1 \bar{x}_3,\ P_2^{(1)} = x_2 \bar{x}_3,\ P_3^{(1)} = x_1 x_2,\ P_4^{(1)} = x_1 x_3;$$

$$P_1^{(2)} = \bar{x}_1,\ P_2^{(2)} = x_2 \bar{x}_3;$$

$$P_1^{(3)} = x_1 x_2,\ P_2^{(3)} = x_1 x_3;$$

$$P_1^{(1,2)} = \bar{x}_1 \bar{x}_3,\ P_2^{(1,2)} = x_2 \bar{x}_3;$$

$$P_1^{(1,3)} = x_1 x_2,\ P_2^{(1,3)} = x_1 x_3;$$

$$P_1^{(2,3)} = x_1 x_2 \bar{x}_3;$$

$$P_1^{(1,2,3)} = x_1 x_2 \bar{x}_3.$$

На другому етапі мінімізації з отриманих простих імплікант за допомогою перебору різних варіантів знаходять найпростіші формули для подання функцій системи. Такі формули зручно знаходити за допомогою іиплікантних таблиць для систем функцій, які аналогічні відповідним таблицям для однієї функції.

Кількість стовпців у таблиці дорівнює сумарній кількості мінтермів у всіх функціях системи. Кожному стовпцю увідповіднюється один мінтерм певної функції системи. Для спрощення таблиці мінтерми зручно позначати номерами наборів, для яких мінтерми дорівнюють одиниці (табл. 1.2). Показник у дужках при цьому номері вказує на ту функцію вихідної системи, якій відповідає цей мінтерм. Усі стовпці, позначені однаковими мінтермами різних функцій, є сусідніми.

Кількість рядків в імплікантній таблиці дорівнює кількості простих імплікант різних сукупностей функцій цієї системи. Кожному рядку відповідає одна проста імпліканта. Якщо М є простою імплікантою, наприклад, функцій f_1 і f_2 і, одночасно з цим, системи функцій f_1, f_2, f_3, то ця імпліканта позначається в таблиці як імпліканта функцій f_1, f_2, f_3. У прикладі, що розглядається, таблиця має шість рядків, оскільки система функцій f_1, f_2, f_3 має шість імплікант (табл. 2.2).

Якщо який-небудь мінтерм функції f_i покривається простою імплікантою підсистеми, в яку входить функція f_i, то на перетині відповідних рядків і стовпців ставиться позначка. У такому разі необхідно враховувати, що прості імпліканти підсистеми функції можуть покривати тільки ті мінтерми, які входять до складу функції. Для того, щоб імпліканта $p^{(E)}$ покрила мінтерм $b^{(j)}$, необхідно, щоб $j \in E$, тобто j, належав множині Е показників системи функцій, для якої $p^{(E)}$ є простою імплікантою.

Таблиця 2.2

	$\bar{x}_1\bar{x}_2\bar{x}_3$ $0^{(1)}$	$\bar{x}_1\bar{x}_2\bar{x}_3$ $0^{(2)}$	$x_1x_2x_3$ $1^{(2)}$	$\bar{x}_1\bar{x}_2x_3$ $2^{(1)}$	$\bar{x}_1x_2\bar{x}_3$ $2^{(2)}$	$\bar{x}_1x_2x_3$ $3^{(2)}$	$x_1\bar{x}_2x_3$ $5^{(1)}$	$x_1\bar{x}_2x_3$ $5^{(3)}$	$x_1x_2\bar{x}_3$ $6^{(1)}$	$x_1x_2\bar{x}_3$ $6^{(2)}$	$x_1x_2\bar{x}_3$ $6^{(3)}$	$x_1x_2x_3$ $7^{(1)}$	$x_1x_2x_3$ $7^{(3)}$
$P_1^{(1,2,3)} = x_1x_2\bar{x}_3$									⊗	⊗	⊗		
$P_1^{(1,3)} = x_1x_2$									×		×	×	×
$P_2^{(1,3)} = x_1x_3$							⊗	⊗				⊗	⊗
$P_1^{(1,2)} = \bar{x}_1\bar{x}_3$	⊗	⊗		⊗	⊗								
$P_2^{(1,2)} = x_2\bar{x}_3$				×	×				×	×			
$P_1^{(2)} = \bar{x}_1$		⊗	⊗		⊗	⊗							

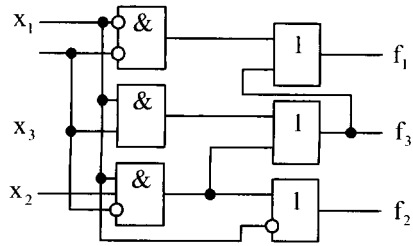
Після того, як усі необхідні позначки поставлені, за імплікантною таблицею вибирають систему простих імплікант, яка покриває усі мінтерми усіх функцій системи. Таку систему простих імплікант прийнято називати повною. З усіх повних систем імплікант вибирають ту, яка містить найменшу кількість букв, так звану мінімальну повну систему. З імплікант мінімальної повної системи формується ДНФ функції. Якщо в якому-небудь стовпці є тільки одна позначка, то відповідна імпліканта обов'язково повинна бути у будь-якій повній системі імплікант.

У прикладі, що розглядається, мінімальна повна система імплікант є такою:

$$x_1x_2\bar{x}_3,\ x_1x_3,\ \bar{x}_1\bar{x}_3,\ \bar{x}_1.$$

Звідси остаточно отримаємо результати спільної мінімізації системи логічних функцій:

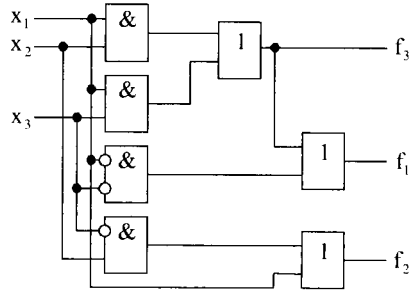
$$f_1 = \bar{x}_1\bar{x}_3 + x_1x_3 + x_1x_2\bar{x}_3,$$
$$f_2 = \bar{x}_1 + x_1x_2\bar{x}_3,$$
$$f_3 = x_1x_3 + x_1x_2\bar{x}_3.$$



Для реалізації цих функцій необхідно три логічні елементи *І* і три логічні елементи АБО із загальним числом входів 13.

Якщо ж мінімізувати задані логічні функції f_1, f_2, f_3, окремо для їх реалізації потрібно було б чотири логічні елементи *І* і три логічні елементи АБО із загальним числом входів 14:

$$f_1 = \bar{x}_1\bar{x}_3 + x_1x_2 + x_1x_3,$$
$$f_2 = \bar{x}_1 + x_2\bar{x}_3,$$
$$f_3 = x_1x_3 + x_1x_2.$$



2.2.4. Дешифратори

Дешифраторами називаються комбінаційні пристрої, що перетворюють двійковий код на унітарний позиційний код, тобто перетворення, в якому кожній комбінації двійкових вхідних сигналів відповідає активне значення тільки одного визначеного вихідного сигналу. Повний дешифратор з n входами має 2^n виходів.

Для неповного дешифратора, що має n входів, кількість виходів менша, ніж 2^n. У разі синтезу таких дешифраторів можна використати правила мінімізації частково визначених функцій.

До прикладу, наведемо таблицю істинності повного дешифратора, що має три входи (табл. 2.3).

Таблиця 2.3

x_3	x_2	x_1	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Логічні рівняння для вихідних змінних дешифратора з n входами записуються у вигляді мінтермів вхідних змінних:

$$y_1 = \bar{x}_n \bar{x}_{n-1} \ldots \bar{x}_2 \bar{x}_1,$$

$$y_2 = \bar{x}_n \bar{x}_{n-1} \ldots \bar{x}_2 x_1,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$y_{2^n} = x_n x_{x-1} \ldots x_2 x_1.$$

Умовне позначення дешифратора наведено на рис. 2.4, *а*.

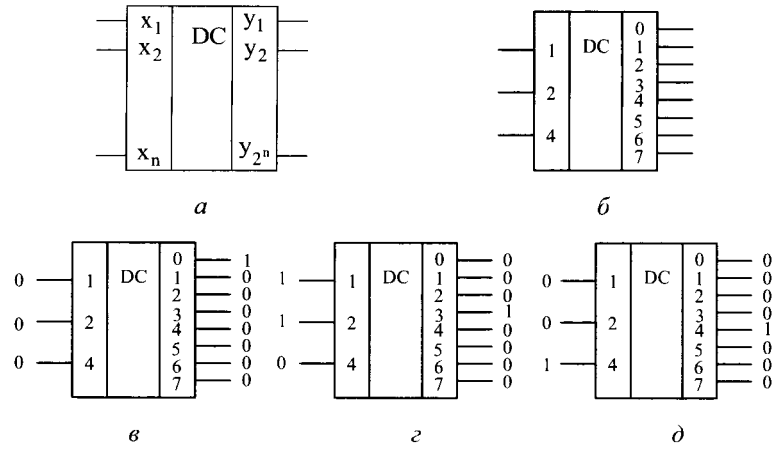


Рис. 2.4

На практиці часто використовується умовне зображення, що для дешифратора з трьома входами показано на рис. 2.4, *б*. Тут входи позначені вагами розрядів двійкового коду, а виходи пронумеровані від 0 до $2^n - 1$. Завдяки цьому легко визначити номер виходу, на якому формується сигнал відповідно до вхідних двійкових кодів (рис. 2.4, *в–д*).

Стосовно внутрішньої структури своєї побудови дешифратори поділяються на:

- матричні (лінійні, одноступеневі);
- пірамідальні;
- ступеневі (двоступеневі, триступеневі, …, багатоступеневі).

Схема матричного дешифратора, що має три входи, наведена на рис. 2.5.

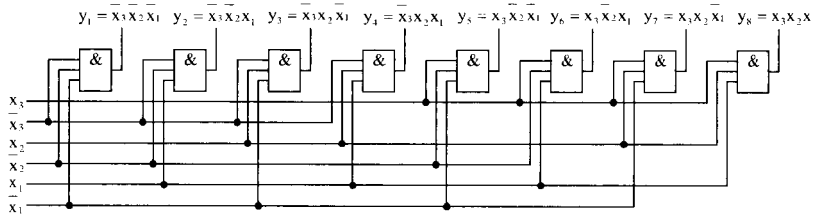


Рис. 2.5

Основною перевагою таких дешифраторів є висока швидкодія, зумовлена мінімальною затримкою формування вихідних сигналів під час зміни вхідних сигналів, оскільки між входами і виходами пристрою є тільки один логічний елемент. Недоліком матричних дешифраторів є те, що в разі збільшення кількості входів відповідно збільшується кількість входів логічних елементів, що має свої технологічні обмеження.

На рис. 2.4 наведено схему пірамідального дешифратора, що виконує ідентичну функцію.

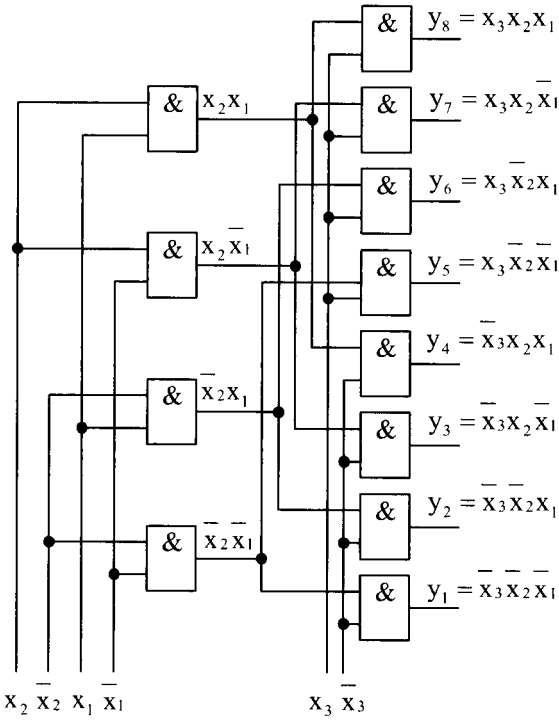


Рис. 2.6

Перевагою пірамідальних дешифраторів є те, що вони складаються з однотипних елементів, кількість входів яких не збільшується із збільшенням кількості входів пристрою. Їхні

недоліки полягають в тому, що, по-перше, за швидкодією вони поступаються матричним дешифраторам і, по-друге, в разі збільшення кількості входів дешифратора стрімко зростає кількість логічних елементів.

Структура ступеневих дешифраторів, яка в загальному вигляді подана на рис. 2.7, є певним компромісом між структурами матричних і пірамідальних пристроїв.

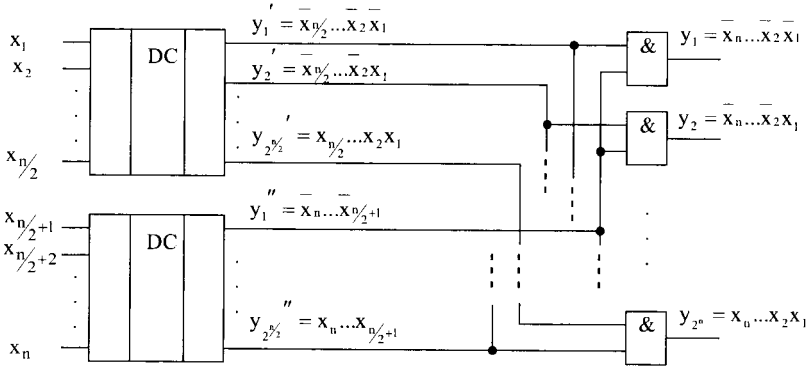


Рис. 2.7

Дешифратори, що зображені на рис. 2.7, своєю чергою, можуть бути матричними, пірамідальними чи ступеневими.

2.2.5. Шифратори

Шифратори виконують функцію, обернену до функції дешифратора, – перетворюють унітарний позиційний код на двійковий код. Повний шифратор має 2^n входів і n виходів. У неповному шифраторі кількість входів є меншою від 2^n.

Нижче наведена таблиця істинності повного шифратора, що має три виходи.

Таблиця 2.4

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y_3	y_2	y_1
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Умовне позначення шифратора наведено на рис. 2.8, *а*.

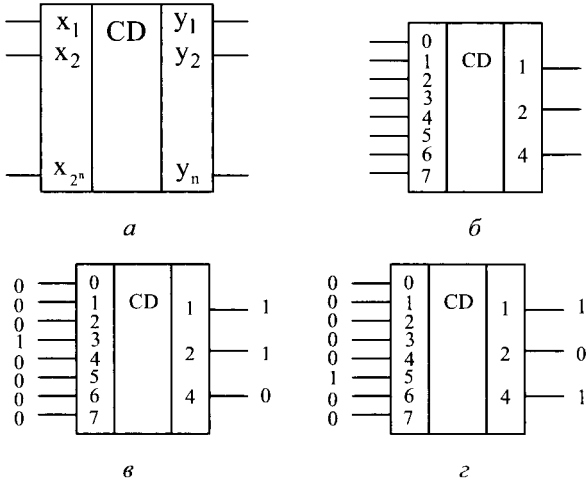


Рис. 2.8

На практиці використовується також умовне зображення, що для шифратора з трьома входами показано на рис. 2.8, *б*. Тут виходи позначені вагами розрядів двійкового коду, а входи пронумеровані від 0 до $2^n - 1$. Завдяки цьому можна просто визначити вихідний двійковий код залежно від того, на який вхід надходить одиничний сигнал (рис. 2.8, *в*, *г*).

Логічні рівняння для вихідних змінних і приклад реалізації повного шифратора наведені нижче для пристрою з трьома виходами.

$$y_1 = x_2 + x_4 + x_6 + x_8,$$

$$y_2 = x_3 + x_4 + x_7 + x_8,$$

$$y_3 = x_5 + x_6 + x_7 + x_8.$$

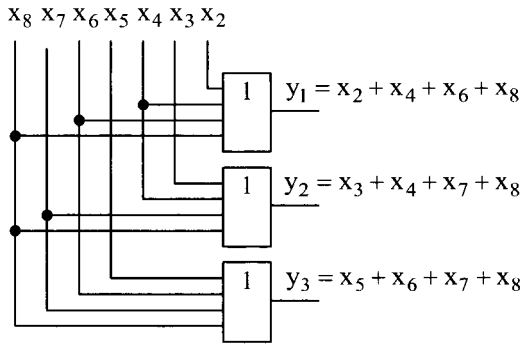


Рис. 2.9

2.2.6. Перетворювачі кодів

Перетворювачі кодів призначені для перетворення одного цифрового коду на інший за довільним законом, який можна задати таблично, аналітичними рівняннями чи в інший спосіб. Під час проектування перетворювачів можуть використовуватись різні підходи. Наведемо два з них на прикладі.

Нехай закон перетворення заданий таблицею істинності (табл. 2.5), де x_1, x_2 і y_1, y_2, y_3 – вхідні та вихідні двійкові змінні відповідно, X і Y – їх десяткові значення.

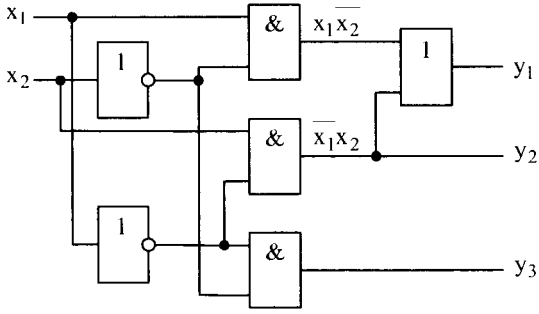
Таблиця 2.5

X	x_2	x_1	y_3	y_2	y_1	Y
0	0	0	1	0	0	4
1	0	1	0	0	1	1
2	1	0	0	1	1	3
3	1	1	0	0	0	0

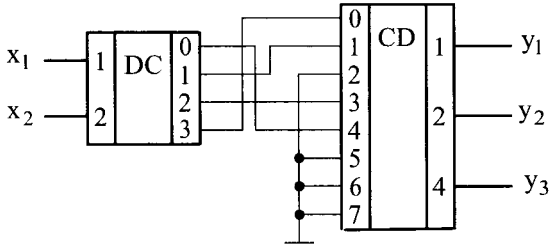
Робота такого перетворювача кодів може бути описана логічними рівняннями:

$$y_3 = \bar{x}_2\bar{x}_1,\ y_2 = x_2\bar{x}_1,\ y_1 = \bar{x}_2 x_1 + x_2\bar{x}_1.$$

Якщо таблиця істинності й відповідно логічні рівняння були б складнішими, необхідно було б провести мінімізацію логічних функцій, використовуючи, зокрема, правила спільної мінімізації системи логічних функцій. У прикладі, що розглядається, схему перетворювача кодів можна побудувати, безпосередньо враховуючи наведені рівняння:

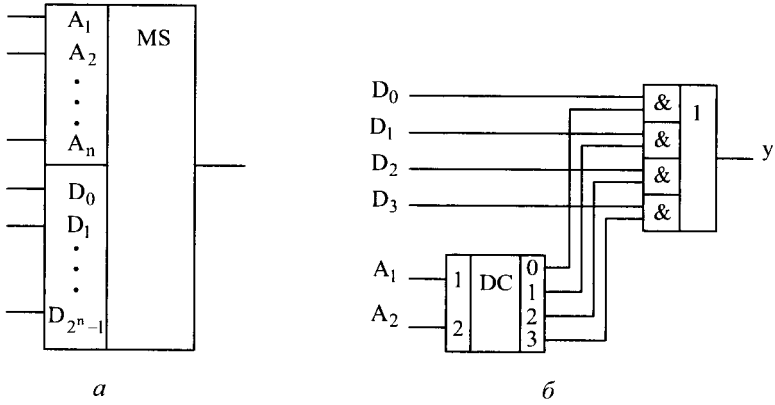


За другого підходу побудови перетворювачів кодів використовується пара "дешифратор–шифратор". Виходи дешифратора з'єднуються з входами шифратора відповідно до десяткових значень вхідних і вихідних кодів перетворювача (табл. 2.5):



2.2.7. Мультиплексори

Мультиплексором називається комбінаційний пристрій, який здійснює під'єднання (комутацію) одного з декількох входів даних на вихід. Загалом мультиплексор має n адресних входів, 2^n входів даних і один вихід. Умовне позначення мультиплексора і приклад його реалізації подані на рис. 2.10.



а *б*

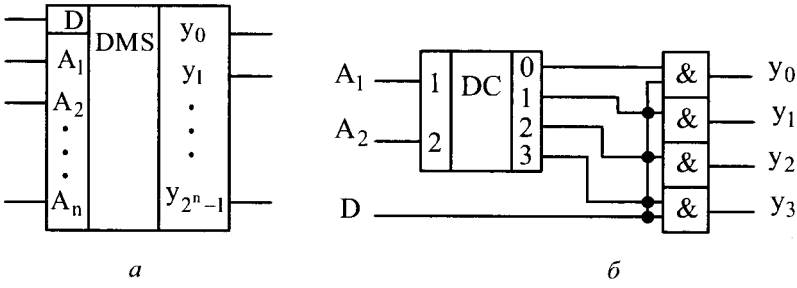
Рис. 2.10

Функціонування мультиплексора (рис. 2.10, *б*) відбувається відповідно до таблиці:

A_2	A_1	DC				y
		0	1	2	3	
0	0	1	0	0	0	D_0
0	1	0	1	0	0	D_1
1	0	0	0	1	0	D_2
1	1	0	0	0	1	D_3

2.2.8. Демультиплексори

Демультиплексором називається комбінаційний пристрій, який здійснює під'єднання (комутацію) входу даних до одного з виходів. Загалом демультиплексор має один вхід даних, n адресних входів і 2^n виходів. Умовне позначення демультиплексора і приклад його реалізації наведено на рис. 2.11.



а *б*

Рис. 2.11

Демультиплексор працює згідно з таблицею:

A_2	A_1	DC				y_0	y_1	y_2	y_3
		0	1	2	3				
0	0	1	0	0	0	D	0	0	0
0	1	0	1	0	0	0	D	0	0
1	0	0	0	1	0	0	0	D	0
1	1	0	0	0	1	0	0	0	D

2.2.9. Програмовані логічні матриці

Найпростіші програмовані логічні матриці – це комбінаційні пристрої, призначені для реалізації логічних функцій, заданих таблично чи аналітично. Їх можна програмувати (тобто формувати з'єднання між їхніми елементами) на фірмі, що їх виготовлює, або безпосередньо користувачем.

На рис. 2.12 показано приклад простої програмованої логічної матриці.

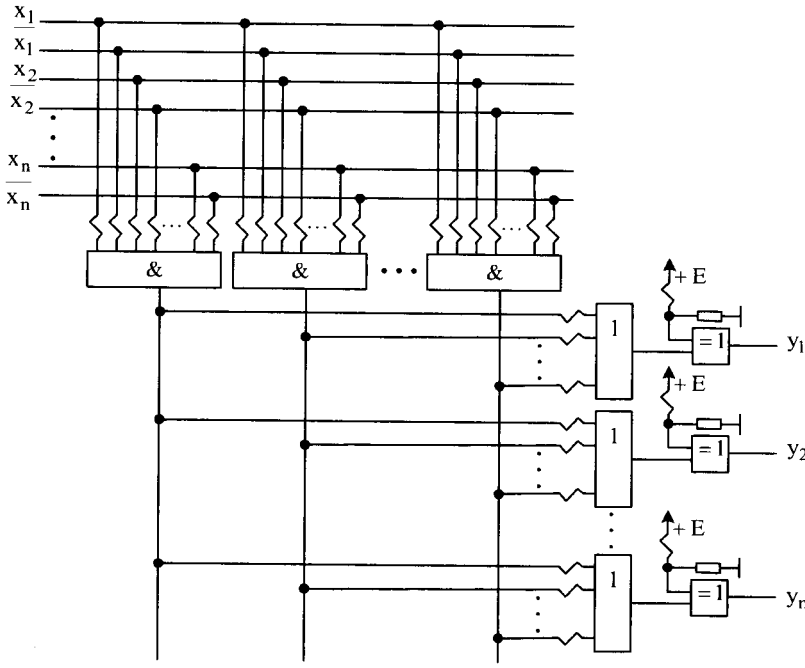


Рис. 2.12

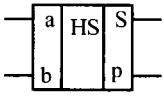
Вихідні сигнали $y_1, y_2, ..., y_m$ формуються відповідно до логічних рівнянь, поданих у диз'юнктивній нормальній формі. Кон'юнкції вхідних сигналів $x_1, x_2, ..., x_n$ і їх диз'юнкції формуються в спосіб руйнування або не руйнування відповідних перемичок на входах логічних елементів І і АБО. Суматори за модулем 2 використовуються для додаткового інвертування вихідних сигналів за необхідності.

2.2.10. Комбінаційні суматори

Комбінаційними суматорами називаються комбінаційні пристрої, що здійснюють арифметичне додавання чисел. У цифровій техніці додавання виконується переважно над двійковими (рідше двійково-десятковими) числами.

Комбінаційні суматори доволі часто будуються на основі однорозрядних напівсуматорів і однорозрядних повних суматорів.

Умовне позначення напівсуматора, його таблиця істинності, логічні рівняння, що описують його вихідні сигнали, і варіант його реалізації на логічних елементах подано на рис. 2.13. Тут a і b – входи доданків, S – вихід суми, p – вихід переносу.



a	b	S	p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = a\bar{b} + \bar{a}b = a \oplus b$$
$$p = ab$$

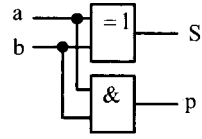
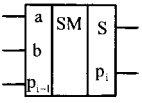


Рис. 2.13

На рис. 2.14 наведено умовне позначення однорозрядного повного суматора, його таблиця істинності, варіанти логічних рівнянь, що описують його роботу, і варіант реалізації на логічних елементах. Тут a і b – входи доданків, S – вихід суми, p_{i-1} – вхід переносу, p_i – вихід переносу.

Існує декілька типів багаторозрядних комбінаційних суматорів, тобто суматорів, що додають багаторозрядні числа. Вони різняться за формою вхідних кодів і способом побудови. Стосовно форми вхідних кодів суматори поділяються на послідовні та паралельні.



p_{i-1}	a	b	S	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

$$S = p_{i-1} \oplus a \oplus b$$
$$p_i = ab + p_{i-1}(a \oplus b)$$

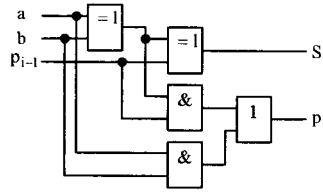


Рис. 2.14

На рис. 2.15 наведено схему послідовного багаторозрядного суматора, до складу якого входять однорозрядний повний суматор і елемент затримки ЕЗ.

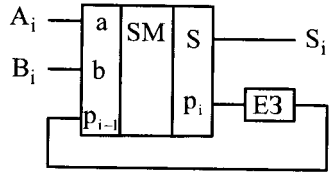


Рис. 2.15

Розряди вхідних двійкових кодів A_i і B_i послідовно надходять на входи доданків однорозрядного повного суматора, починаючи із молодших розрядів. ЕЗ затримує сигнал переносу до моменту надходження на входи пристрою наступних розрядів.

На рис. 2.16, *а* наведено схему паралельного багаторозрядного суматора з послідовним переносом, а на рис. 2.16, *б* – умовне позначення паралельного багаторозрядного суматора незалежно від типу переносу, що використовується.

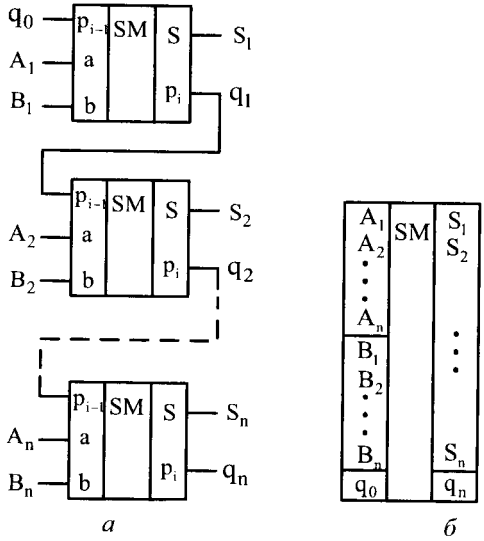


Рис. 2.16

У цьому випадку усі розряди доданків A_i і B_i надходять на входи суматора одночасно. Підсумовування відбувається з урахуванням міжрозрядних переносів q_i, які поширюються ланцюгом переносів, починаючи з виходу переносу однорозрядного повного суматора молодшого розряду. Перевагою такого типу багаторозрядного суматора є простота побудови, а недоліком – низька швидкодія, зумовлена часом поширенню сигналів переносу від молодшого до старшого розрядів.

Суттєве підвищення швидкодії досягається під час побудови багаторозрядних суматорів з використанням паралельного способу формування переносів. Для ілюстрації цього способу наведемо таблицю істинності однорозрядного повного суматора з використанням символів, використаних на рис. 2.16:

q_{i-1}	A_i	B_i	S_i	q_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

На основі цієї таблиці запишемо рівняння для сигналів виходу суми і виходу переносу для *i*-го розряду:

$$S_i = q_i \oplus A_i \oplus B_i, \tag{2.6}$$

$$q_i = A_i B_i + q_{i-1}(A_i + B_i) = \alpha_i + q_{i-1}\beta_i, \tag{2.7}$$

де $\alpha_i = A_i B_i$, $\beta_i = A_i + B_i$.

Використовуючи логічне рівняння (2.7), можна записати:

$$q_1 = \alpha_1 + q_0\beta_1,$$

$$q_2 = \alpha_2 + q_1\beta_2 = \alpha_2 + \alpha_1\beta_2 + q_0\beta_1\beta_2,$$

$$q_3 = \alpha_3 + q_2\beta_3 = \alpha_3 + \alpha_2\beta_3 + \alpha_1\beta_2\beta_3 + q_0\beta_1\beta_2\beta_3, \tag{2.8}$$

$$q_4 = \alpha_4 + q_3\beta_4 = \alpha_4 + \alpha_3\beta_4 + \alpha_2\beta_3\beta_4 + \alpha_1\beta_2\beta_3\beta_4 + q_0\beta_1\beta_2\beta_3\beta_4,$$

.....

На рис. 2.17 подано схему паралельного чотирирозрядного суматора з паралельним переносом, у частині, що забезпечує формування цих переносів, побудована у відповідності до логічних рівнянь (2.8).

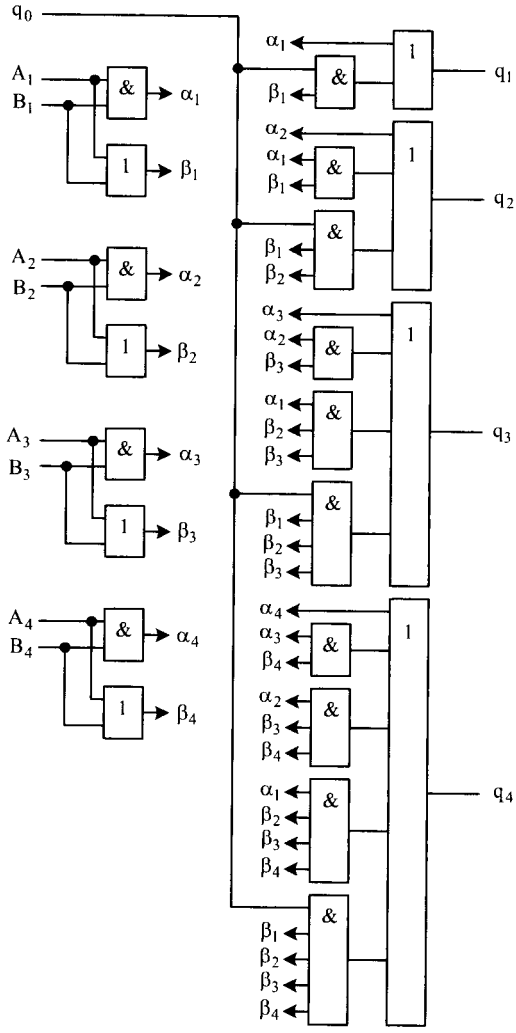


Рис. 2.17

У схемі немає ланцюга нагромадження затримок поширення сигналів переносів, за рахунок чого досягається істотне підвищення швидкодії пристрою. Однак під час збільшення кількості розрядів доданків стрімко зростатиме складність суматора.

На рис. 2.18 зображено структурну схему паралельного багаторозрядного суматора з паралельно-послідовним переносом. Такий суматор є компромісним варіантом відносно двох попередніх.

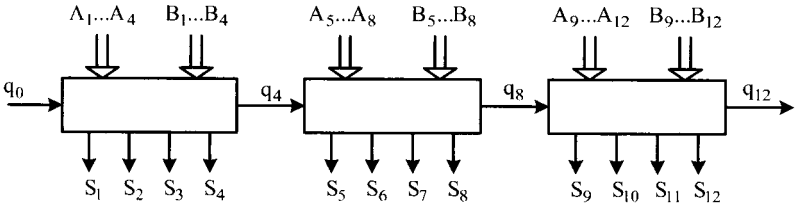


Рис. 2.18

У наведеній схемі дванадцятирозрядний суматор розділений на три чотирирозрядні блоки, кожен з яких побудований за принципом суматора з паралельним переносом, а між блоками реалізований послідовний принцип поширення переносу. Такий спосіб побудови суматорів дає змогу знайти компроміс між складністю їхньої побудови і швидкодією.

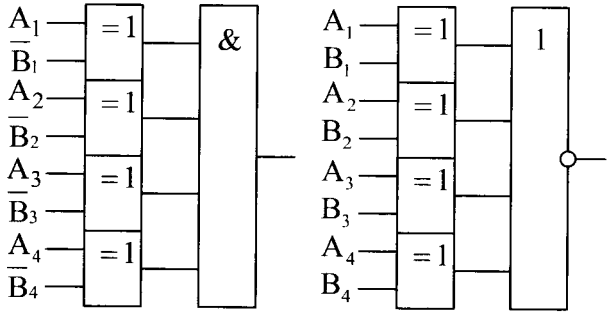
2.2.11. Пристрої порівняння

Пристрої порівняння (цифрові компаратори) є комбінаційними пристроями, що призначені для порівняння величин двох чисел, що подані у двійковому чи іншому коді.

Найпростіші компаратори виявляють лише факт рівності двох чисел. Два можливі варіанти реалізації таких компараторів наведено на рис. 2.19.

У схемі на рис. 2.19, *а* за умови рівності усіх розрядів вхідних двійкових чисел – $A_i = B_i$ $(i = 1,2,...,n)$ на виходах усіх суматорів за модулем 2 формуються логічні одиниці і, отже, тільки

в цьому випадку на виході компаратора формується логічна одиниця. У схемі на рис. 2.19, *б* за тих самих умов на виходах усіх суматорів за модулем 2 формуються нулі, і тільки в цьому випадку на виході формується одиниця.



а *б*

Рис. 2.19

Реалізація цифрового компаратора, що формує на своєму виході логічну одиницю у випадку, коли $A > B$, ґрунтується на такій логіці. Число $A > B$, коли: $A_n > B_n$; або $A_n = B_n$, $A_{n-1} > B_{n-1}$; або $A_n = B_n$, $A_{n-1} = B_{n-1}$, $A_{n-2} > B_{n-2}$ і так далі. Тут A_n і B_n – старші розряди двійкових чисел, що порівнюються. Цю логіку можна записати у вигляді логічного рівняння:

$$\begin{aligned} Y &= A_n\overline{B_n} + (A_nB_n + \overline{A_nB_n})A_{n-1}\overline{B_{n-1}} + \dots \\ &\dots + (A_nB_n + \overline{A_nB_n})\cdot\dots\cdot(A_2B_2 + \overline{A_2B_2})A_1\overline{B_1} = \\ &= A_n\overline{B_n} + F_nA_{n-1}\overline{B_{n-1}} + \dots + F_nF_{n-1}\cdot\dots\cdot F_2A_1\overline{B_1} = \\ &= A_n\overline{B_n} + F_n(A_{n-1}\overline{B_{n-1}} + F_{n-1}(\dots + F_3(A_2\overline{B_2} + F_2A_1\overline{B_1})\dots)), \end{aligned} \tag{2.9}$$

де Y – вихідний сигнал компаратора, що дорівнює одиниці у випадку $A > B$, $F_i = A_iB_i + \overline{A_iB_i}$ $(i = 2,3,\dots,n)$.

Функцію F_i можна реалізувати різними способами (рис. 2.20).

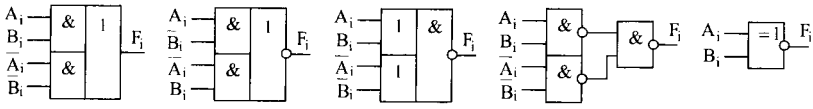


Рис. 2.20

Схему, що реалізує рівняння (2.9), у випадку порівняння чотирирозрядних двійкових чисел, наведено на рис. 2.21.

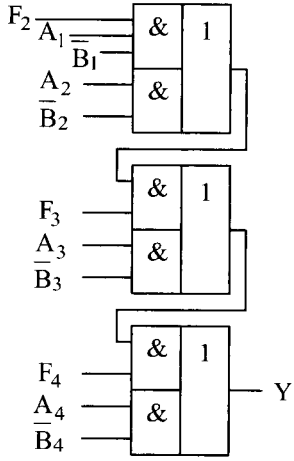


Рис. 2.21

Питання і задачі для самоконтролю

1. Чим відрізняються комбінаційні та послідовнісні цифрові пристрої?
2. Чим відрізняються асинхронні й синхронні цифрові пристрої?
3. За рахунок чого на виходах комбінаційних пристроїв можуть формуватись хибні сигнали?
4. Скільки виходів має повний дешифратор, що має n входів?
5. Який тип дешифраторів забезпечує найбільшу швидкодію?
6. Скільки виходів має повний шифратор, що має n входів?
7. Скільки входів даних має мультиплексор, що має n адресних входів?
8. Скільки виходів має демультиплексор, що має n адресних входів?
9. Яку основну функцію виконують комбінаційні суматори?
10. Який тип комбінаційних суматорів забезпечує максимальну швидкодію?

2.3. Послідовнісні пристрої

2.3.1. Принципи логічного проектування послідовнісних пристроїв

Значення вихідних змінних послідовнісного пристрою на будь-якому такті його роботи залежать від значень вхідних змінних на цьому такті, а також від значень вхідних змінних на певній скінченній кількості попередніх тактів. Отже, послідовнісні пристрої повинні мати внутрішню пам'ять. Властивість запам'ятовування інформації забезпечується існуванням низки стійких внутрішніх станів послідовнісного пристрою.

Позначимо кількість різних внутрішніх станів символом M. Повному набору внутрішніх станів можна поставити у відповідність набір комбінацій двійкових змінних $z_1, z_2, ..., z_m$, які називаються внутрішніми змінними послідовнісного пристрою. Позначимо через Z множину внутрішніх змінних. Кількість внутрішніх змінних m, що необхідна для кодування усіх станів M, визначається співвідношенням

$$m = [\log_2 M], \tag{2.10}$$

де символ [ ] означає, що береться найменше ціле число, рівне чи більше від символа в дужках.

Значення m називається об'ємом пам'яті послідовнісного пристрою. У разі дробового значення $\log_2 M$ деякі комбінації значень внутрішніх змінних будуть надлишковими.

Задання оператора, що реалізується послідовнісним пристроєм, передбачає, по-перше, формування зв'язку вихідних змінних Y^t із вхідними змінними X^t і внутрішніми змінними Z^t для одного і того самого такту t і, по-друге, формування зв'язку внутрішніх змінних Z^{t+1} для такту $t+1$ із значеннями X^t і Z^t. Основними формалізованими способами задання оператора, як і для комбінаційних пристроїв, є аналітичний і табличний.

Аналітичний спосіб характеризується двома системами рівнянь – рівняннями виходів (2.11) і рівняннями переходів (2.12):

$$\begin{cases} y_1^t = y_1(x_1^t, x_2^t, ..., x_n^t; z_1^t, z_2^t, ..., z_m^t), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_s^t = y_s(x_1^t, x_2^t, ..., x_n^t; z_1^t, z_2^t, ..., z_m^t). \end{cases} \quad (2.11)$$

$$\begin{cases} z_1^{t+1} = z_1(x_1^t, x_2^t, ..., x_n^t; z_1^t, z_2^t, ..., z_m^t), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ z_m^{t+1} = z_m(x_1^t, x_2^t, ..., x_n^t; z_1^t, z_2^t, ..., z_m^t). \end{cases} \quad (2.12)$$

Таблиці, що відповідають цим залежностям, називають таблицями виходів і переходів, а за їх об'єднання – таблицями станів. Тобто в таблицях станів кожному із можливих наборів X^t і Z^t увідповіднюють значення Y^t і Z^{t+1}.

Розглянемо узагальнену модель послідовнісного пристрою (рис. 2.22, *а*). Ця модель містить комбінаційний пристрій КП, який формує сигнал Y^t на основі значень X^t і $Z_{вх} = Z^t$, а також кола зворотного зв'язку КЗЗ, якими з виходу на вхід подаються сигнали внутрішніх змінних.

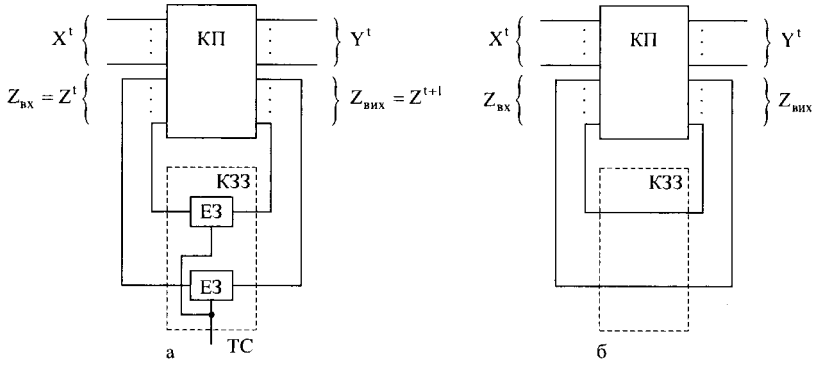


Рис. 2.22

Залежно від властивостей КЗЗ розрізняють два типи послідовнісних пристроїв – асинхронні та синхронні. В синхронних пристроях КЗЗ містять елементи затримки (запам'ятовування) ЕЗ.

Вони управляються зовнішніми тактовими сигналами ТС і виконують функцію запам'ятовування сигналів внутрішніх змінних на один такт. За відсутності тактових імпульсів передавання сигналів по КЗЗ не відбувається.

Модель асинхронного послідовнісного пристрою (рис. 2.22, *б*) характеризується характером кіл зворотного зв'язку. У них затримка сигналів внутрішніх змінних обумовлена інерційністю логічних елементів і скінченною швидкістю поширення електричних сигналів в електричних колах.

У комбінаційній частині послідовнісного пристрою можуть бути "гонки", що спричиняють формування хибних сигналів. У синхронних пристроях хибні значення $Z_{вих} = Z^{t+1}$ не впливають на роботу, оскільки тактові імпульси, що дають змогу передавати сигнали внутрішніх змінних з виходу на вхід, подаються після закінчення динамічної частини такту.

В асинхронних пристроях "гонки" мають першочергове значення – вони можуть спричиняти хибні спрацювання.

Асинхронні послідовнісні пристрої, що побудовані відповідно до основної моделі (рис. 2.22, *б*), містять лише комбінаційні елементи. Синхронні пристрої містять, окрім того, тактовані запам'ятовувальні елементи – елементарні автомати пам'яті (тригери). Менша вартість комбінаційних елементів порівняно із запам'ятовувальними на початковому етапі впровадження цифрової техніки визначала переваги асинхронних автоматів над синхронними (незважаючи на властиву першим небезпеку "гонок"). Впровадження мікроелектронної елементної бази привело до того, що вартість комбінаційних елементів і тактованих елементів затримки стали близькими. Відповідно сьогодні існує обґрунтована тенденція використання головно синхронних послідовнісних пристроїв.

В узагальненій моделі синхронного послідовнісного пристрою комбінаційні і запам'ятовувальні елементи розділені. Однак елементи пам'яті можуть виконувати і деякі логічні перетворення сигналів.

2.3.2. Тригери

Тригерами називають елементарні послідовнісні автомати, що характеризуються такими параметрами:

– невелика кількість вхідних змінних. Як правило, без урахування тактового сигналу, не більше від чотирьох;

– число внутрішніх станів дорівнює двом, що відповідає одній внутрішній змінній (останню прийнято позначати символом Q);

– одна вихідна змінна – у. Значення у збігається із значенням Q, тобто функція виходу для тригера має вигляд $y = Q$. Звичайно в тригерах є можливість разом із значенням Q отримати інверсну змінну $\overline{Q}$.

Функція переходів для тригерів називається характеристичним рівнянням і має вигляд:

$$Q^{t+1} = Q(X^t, Q^t).$$

2.3.2.1. Тригерні системи, класифікація тригерів

Запис інформації в тригер звичайно здійснюється не безпосередньо, а через схему управління. Сукупність тригера і схеми управління складають тригерну систему (рис. 2.23).

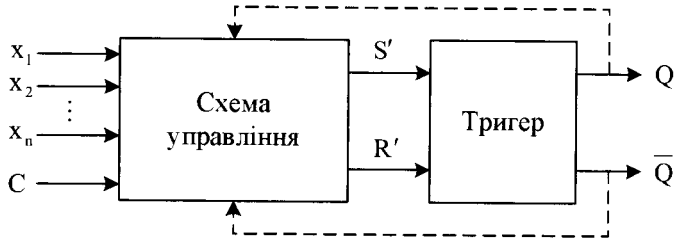


Рис. 2.23

У цій системі власне тригер виконує функцію елемента пам'яті для зберігання інформації. Схема управління передає з входів $x_1, x_2, ..., x_n$ необхідну інформацію на входи тригера S′ і R′, які

установлюють тригер у відповідний стан. Управління тригером може бути тактованим (синхронним) і асинхронним. За тактованого управління тригерна система має додатковий вхід тактових імпульсів С, які дають змогу схемі управління передавати інформацію в тригер. За асинхронного управління входу С немає. Виходи тригера Q і $\overline{Q}$ можуть бути з'єднані з входами схеми управління.

Більшість тригерів є тригерними системами, але далі, для спрощення, здебільшого використовуватимемо термін "тригер", маючи на увазі, що він, як правило, має у своєму складі схему управління.

Класифікацію тригерів наведено на рис. 2.24.

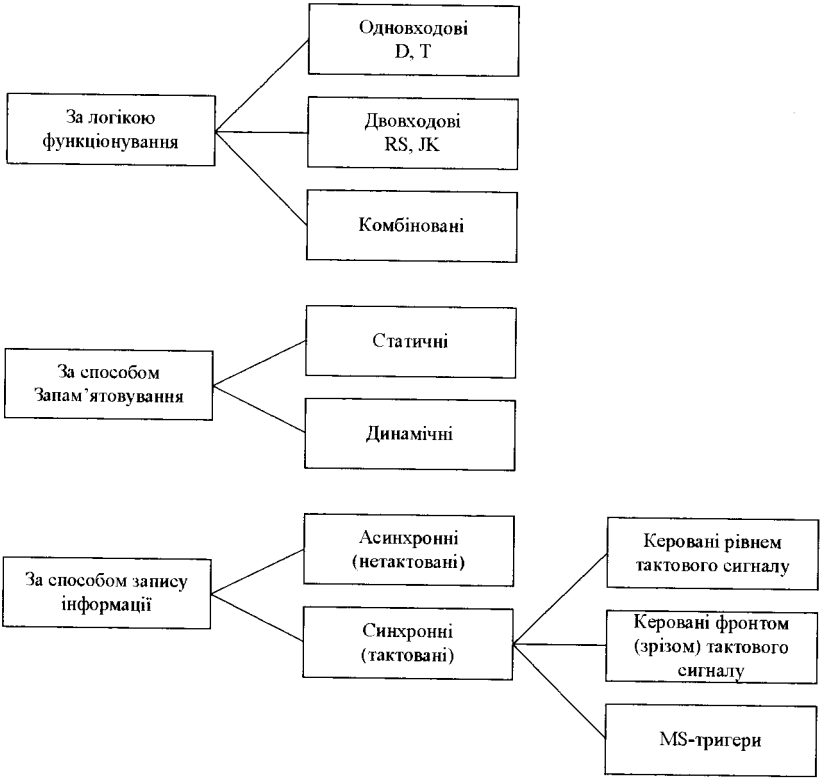


Рис. 2.24

Тригери, що відрізняються один від одного за логікою функціонування, мають різні характеристичні рівняння.

У статичних тригерах кожен із стійких станів характеризується різними значеннями вихідних напруг (струмів). У динамічних тригерах їхній стан характеризується наявністю чи відсутністю вихідних імпульсів із заданими параметрами. Такі тригери, що використовуються рідко, звичайно зараховують не до класу тригерів, а до керованих генераторів.

В асинхронних тригерах вхідна інформація сприймається безпосередньо, а в синхронних тільки за наявності додаткового тактового (синхронізувального) сигналу.

У синхронних тригерах, що керуються рівнем тактового сигналу, вхідна інформація сприймається під час усієї тривалості тактових імпульсів. У синхронних тригерах, що керуються фронтом чи зрізом тактових імпульсів, вхідна інформація сприймається тільки протягом коротких інтервалів часу, що відповідають фронтам чи зрізам цих імпульсів.

Завадостійкість тригерів, що керуються фронтом чи зрізом, більша, ніж у тригерів, що керуються рівнем тактових імпульсів, оскільки в перших завада може діяти на тригер тільки під час наростання чи падіння імпульсу, а в других – протягом усієї тривалості імпульсу.

До MS(master-slave)-тригерів належать тригери (тригерні системи), що складаються з головного і допоміжного тригерів. Кожен із цих тригерів має свою схему керування. Головний тригер виконує основну логічну функцію, а допоміжний тригер призначений для подальшого запам'ятовування стану тригерної системи, виходи якої є виходами допоміжного тригера. Обидва тригера керуються рівнем тактового імпульсу. Зв'язок між ними можна здійснювати трьома способами:

– з інвертуванням тактових імпульсів;

– з блокуванням входів допоміжного тригера сигналами із схеми управління головного тригера;

– з блокуванням входів головного тригера сигналами із схеми управління допоміжного тригера.

Як приклад на рис. 2.25 показано структурну схему MS-тригера з інвертуванням тактових імпульсів.

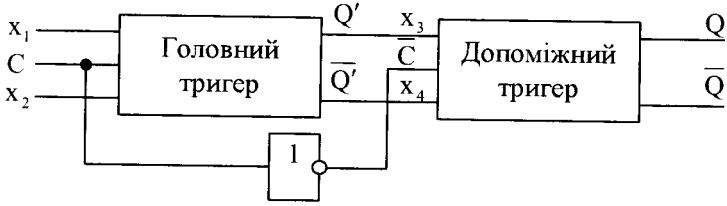


Рис. 2.25

Під час тривалості тактового імпульсу вхідна інформація сприймається головним тригером. У цей час схема управління допоміжного тригера блокується рівнем логічного 0 з виходу інвертора. Після закінчення тактового імпульсу блокується схема управління головного тригера. У цей час на виході інвертора формується логічна одиниця, і допоміжний тригер сприймає інформацію, що записана в головному тригері.

Синхронні тригерні системи мають складнішу схему управління, ніж асинхронні. Однак, синхронні тригери мають істотні переваги над асинхронними стосовно завадостійкості. Не менш важливою перевагою синхронних тригерів є можливість синхронізації роботи окремих вузлів цифрових пристроїв.

2.3.2.2. RS-тригери

Логіку роботи RS-тригера можна подати за допомогою таблиці, поданої на рис. 2.26, *а*. Такі таблиці називаються повними таблицями переходів, зміст яких полягає в тому, щоб визначити, в якому стані опиниться тригер за відомих вхідних сигналів і відомого попереднього стану. Тут R^t, S^t, Q^t – значення вхідних сигналів і попередній стан тригера, Q^{t+1} – наступний стан тригера. Отже, логіка роботи RS-тригера є такою:

– якщо $R^t = S^t = 0$, тригер зберігає попередній стан;

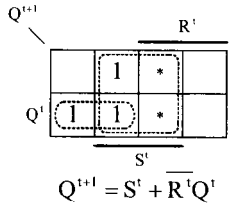
– якщо $R^t = 0$ і $S^t = 1$, тригер установлюється в стан логічної 1;

– якщо $R^t = 1$ і $S^t = 0$, тригер установлюється в стан логічного 0;

– значення $R^t = 1$ і $S^t = 1$ вважаються недозволеною комбінацією вхідних сигналів (недозволений стан – н/с).

Q^t	R^t	S^t	Q^{t+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	н/с
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	н/с

а



б

R^t	S^t	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	н/с

в

$Q^t \to Q^{t+1}$		R^t	S^t
0	0	*	0
0	1	0	1
1	0	1	0
1	1	0	*

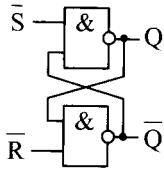
г

Рис. 2.26

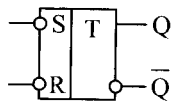
Мінімізувавши логічну функцію Q^{t+1}, задану табл. 2.26, *а*, за правилами мінімізації частково визначених функцій отримаємо характеристичне рівняння RS-тригера (рис. 2.26, *б*). На практиці використовують спрощену таблицю переходів RS-тригера (рис. 2.26, *в*). На рис. 2.26, *г* наведено характеристичну таблицю RS-тригера, яка дає змогу визначити значення вхідних сигналів, що забезпечують один з можливих переходів тригера. Тут символ * означає довільне значення змінної. Характеристичну таблицю можна сформувати (довести) на основі характеристичного рівняння, підставляючи в нього почергово усі можливі комбінації значень Q^t і Q^{t+1}.

Далі розглянемо схеми RS-тригерів різних типів, що побудовані на логічних елементах. Треба зауважити, що, по-перше, наведені схеми є одним із можливих варіантів (можливо одним із найчастіше використовуваних) реалізації тригерів, і, по-друге, існує однозначна відповідність між умовним позначенням тригера і його таблицею переходів незалежно від його внутрішньої побудови. Ці зауваження стосуються й усіх інших тригерів.

На рис. 2.27 наведено схему, умовне позначення і скорочену таблицю переходів асинхронного RS-тригера.



а



б

R^t	S^t	Q^{t+1}
0	0	н/с
0	1	1
1	0	0
1	1	Q^t

в

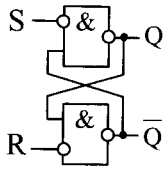
Рис. 2.27

Схема (рис. 2.27, *а*) побудована на основі логічного рівняння

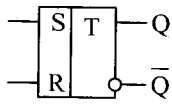
$$Q^{t+1} = \overline{\overline{S}^t \cdot \overline{\overline{R}^t Q^t}},$$

яке безпосередньо випливає з характеристичного рівняння (рис. 2.26, *б*) після його подвійного інвертування.

На рис. 2.28 наведено схему асинхронного RS-тригера, що відрізняється від попередньої іншою полярністю вхідних сигналів, його умовне позначення і скорочену таблицю переходів.



а



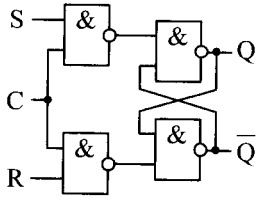
б

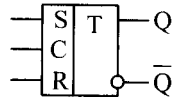
R^t	S^t	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	н/с

в

Рис. 2.28

Схему синхронного RS-тригера, що керується рівнем тактового сигналу, його умовне позначення і таблицю переходів подано на рис. 2.29. Тригер зберігає попередній стан за наявності логічного нуля на тактовому вході C, а за наявності логічної одиниці на цьому вході тригер працює як асинхронний RS-тригер.



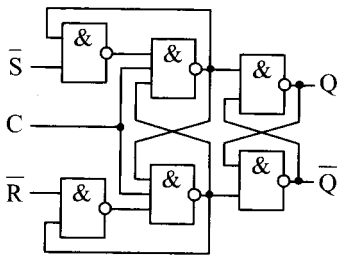


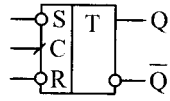
C^t	R^t	S^t	Q^{t+1}
0	0	0	Q^t
0	0	1	Q^t
0	1	0	Q^t
0	1	1	Q^t
1	0	0	Q^t
1	0	1	1
1	1	0	0
1	1	1	н/с

а *б* *в*

Рис. 2.29

Схему синхронного RS-тригера, що керується фронтом тактового сигналу, його умовне позначення і таблицю переходів наведено на рис. 2.30. Тригер працює за логікою роботи асинхронного RS-тригера за наявності на його тактовому вході С фронту імпульсу. В інших випадках він зберігає попередній стан.





C^t	R^t	S^t	Q^{t+1}
↗	0	0	н/с
↗	0	1	0
↗	1	0	1
↗	1	1	Q^t
0	0	0	Q^t
1	0	1	Q^t
↘	1	0	Q^t

а *б* *в*

Рис. 2.30

На рис. 2.31 зображено схему RS-тригера типу MS. Він складається з головного ГТ і допоміжного ДТ тригерів, які є синхронними тригерами, що керуються рівнем тактового сигналу. Тригер працює відповідно до загальної логіки роботи MS-тригера, наведеної під час опису схеми на рис. 2.25.

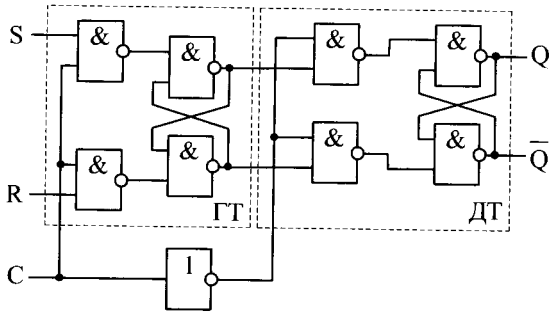


Рис. 2.31

2.3.2.3. JK-тригери

Логіку роботи JK-тригера можна подати за допомогою повної таблиці переходів (рис. 2.32, *а*). Тут J^t і K^t – значення вхідних сигналів. Словесно ця логіка описується так:

– якщо $J^t = K^t = 0$, тригер зберігає попередній стан;

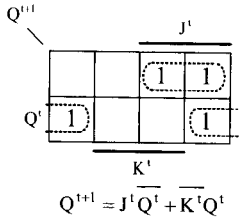
– якщо $K^t = 0$ і $J^t = 1$, тригер установлюється в стан логічної 1;

– якщо $K^t = 1$ і $J^t = 0$, тригер установлюється в стан логічного 0;

– якщо $K^t = 1$ і $J^t = 1$, тригер переходить у стан, протилежний до попереднього.

Q^t	J^t	K^t	Q^{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

а



б

J^t	K^t	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	$\overline{Q^t}$

в

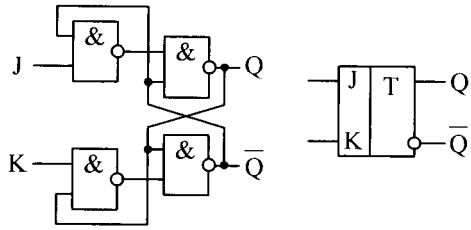
$Q^t \to Q^{t+1}$	J^t	K^t
0 0	0	*
0 1	1	*
1 0	*	1
1 1	*	0

г

Рис. 2.32

Мінімізувавши логічну функцію Q^{t+1}, задану табл. 2.32, *а*, за правилами мінімізації частково визначених функцій, отримаємо характеристичне рівняння JK-тригера (рис. 2.32, *б*). На рис. 2.32, *в* і *г* подано відповідно спрощену таблицю переходів і характеристичну таблицю тригера.

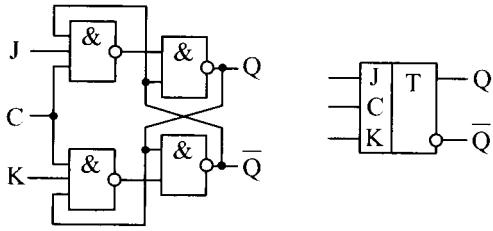
Схему асинхронного JK-тригера, що працює відповідно до таблиць на рис. 2.32, і його умовне позначення наведено на рис. 2.33.



а *б*

Рис. 2.33

Схему синхронного JK-тригера, що керується рівнем тактового сигналу, його умовне позначення і таблицю переходів зображено на рис. 2.34.

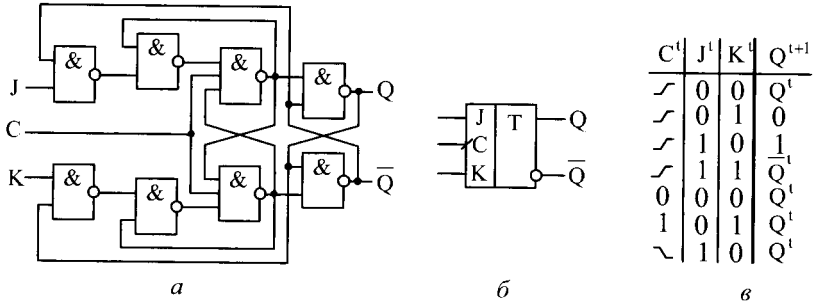


C^t	J^t	K^t	Q^{t+1}
0	0	0	Q^t
0	0	1	Q^t
0	1	0	Q^t
0	1	1	Q^t
1	0	0	Q^t
1	0	1	0
1	1	0	1
1	1	1	$\overline{Q}^t$

а *б* *в*

Рис. 2.34

Схему синхронного JK-тригера, що керується фронтом тактового сигналу, його умовне позначення і таблицю переходів наведено на рис. 2.35.



а *б* *в*

Рис. 2.35

На рис. 2.36 подано схему JK-тригера типу MS.

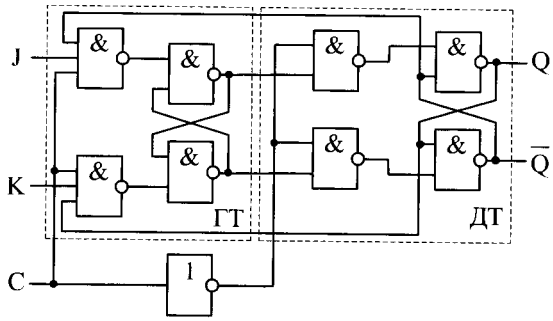


Рис. 2.36

2.3.2.4. D-тригери

Робота D-тригера визначається таблицею переходів (2.37, *а*), де D^t – значення вхідного сигналу, за такою логікою:

- якщо $D^t = 0$, тригер установлюється в стан логічного 0;
- якщо $D^t = 1$, тригер установлюється в стан логічної 1.

На рис. 2.37 також наведено характеристичне рівняння, скорочену таблицю переходів і характеристичну таблицю D-тригера.

Q^t	D^t	Q^{t+1}
0	0	0
0	1	1
1	0	0
1	1	1

а

$Q^{t+1} = D^t$

б

D^t	Q^{t+1}
0	0
1	1

в

$Q^t \to Q^{t+1}$		D^t
0	0	0
0	1	1
1	0	0
1	1	1

г

Рис. 2.37

Асинхронний D-тригер можна реалізувати так, як показано на рис. 2.38.

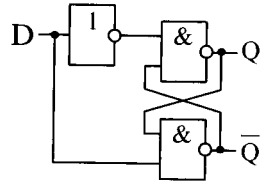
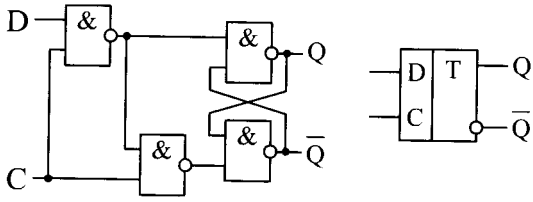


Рис. 2.38

На рис. 2.39 наведено схему, умовне зображення і таблицю переходів синхронного D-тригера, що керується рівнем тактового сигналу.



а *б*

C^t	D^t	Q^{t+1}
0	0	Q^t
0	1	Q^t
1	0	0
1	1	1

в

Рис. 2.39

Схема, умовне зображення і таблиця переходів для синхронного D-тригера, що керується фронтом тактового імпульсу, подані на рис. 2.40.

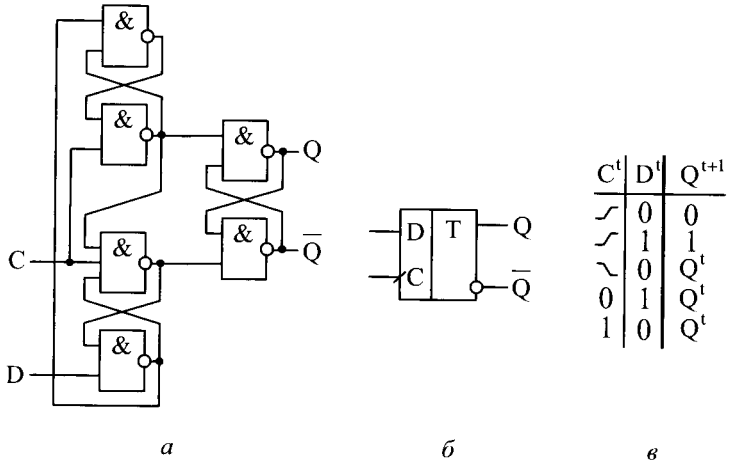


Рис. 2.40

Схему D-тригера типу MS наведено на рис. 2.41.

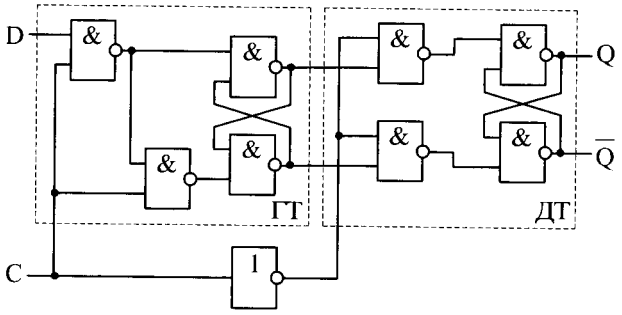


Рис. 2.41

2.3.2.5. Т-тригери

Логіка роботи Т-тригера визначається таблицею переходів (рис. 2.42, *а*), де T^t – значення вхідного сигналу, так:

– якщо $T^t = 0$, тригер зберігає попередній стан;

– якщо $T^t = 1$, тригер переходить у стан протилежний до попереднього.

На рис. 2.42 також наведено характеристичне рівняння, скорочену таблицю переходів і характеристичну таблицю Т-тригера.

Q^t	T^t	Q^{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

а

$$Q^{t+1} = Q^t\overline{T^t} + \overline{Q^t}T^t$$

б

T^t	Q^{t+1}
0	Q^t
1	$\overline{Q}^t$

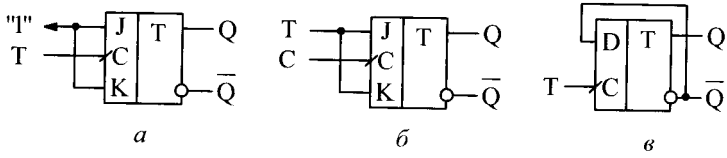
в

$Q^t \to Q^{t+1}$		T^t
0	0	0
0	1	1
1	0	1
1	1	0

г

Рис. 2.42

Т-тригери, як правило, будуються на основі JK і D тригерів. Приклади такої реалізації показано на рис. 2.43.



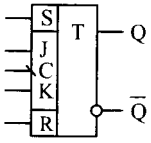
а *б* *в*

Рис. 2.43

2.3.2.6. Комбіновані тригери

Комбіновані тригери поєднують властивості кількох (звичайно двох) більш простих тригерів. У таких тригерах входи R і S, як правило, є асинхронними і такими, що мають пріоритет над

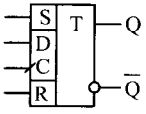
іншими входами. На рис. 2.44 і 2.45, для прикладу, наведено умовні позначення і таблиці переходів двох, із багатьох можливих, таких тригерів.



R^t	S^t	C^t	J^t	K^t	Q^{t+1}
1	1	*	*	*	н/с
1	0	*	*	*	0
0	1	*	*	*	1
0	0	↘	0	0	Q^t
0	0	↘	0	1	0
0	0	↘	1	0	1
0	0	↘	1	1	$\overline{Q}^t$
0	0	0	*	*	Q^t
0	0	1	*	*	Q^t
0	0	↗	*	*	Q^t

а *б*

Рис. 2.44



R^t	S^t	C^t	D^t	Q^{t+1}
1	1	*	*	н/с
1	0	*	*	0
0	1	*	*	1
0	0	↗	0	0
0	0	↗	1	1
0	0	0	*	Q^t
0	0	1	*	Q^t
0	0	↘	*	Q^t

а *б*

Рис. 2.45

2.3.3. Лічильники імпульсів

2.3.3.1. Призначення та класифікація лічильників імпульсів

Лічильниками називаються послідовнісні пристрої, що призначені для підрахунку кількості вхідних сигналів (імпульсів) і зберігання коду результату. Найпростішим лічильником є тригер, наприклад, Т-тригер чи JK-тригер, що працює у режимі лічби, – перемикається в стан, протилежний до попереднього, в разі надходження чергового вхідного імпульсу. Тригер може підраховувати не більше ніж два вхідні імпульси. З'єднавши певну кількість таких тригерів між собою, можна отримати лічильник, що підраховує $M_л$ вхідних імпульсів. Оскільки кожен тригер може перебувати у будь-якому з двох можливих станів, лічильник, що складається з m тригерів, може мати $M_л \le 2^m$ станів, кожен з яких визначається комбінацією станів усіх тригерів лічильника. Перехід лічильника з одного стану в інший відбувається під час надходження чергового вхідного імпульсу. Число $M_л$ називається

модулем лічби або ємністю лічильника. Якщо кількість імпульсів, що надійшли на вхід лічильника, дорівнює $M_л$, він повертається у початковий стан, у такому разі може формуватись сигнал переповнення.

Для заданого модуля лічби $M_л$ мінімальна кількість тригерів m, що необхідна для побудови лічильника, визначається виразом

$$m = [\log_2 M_л],$$

де квадратні дужки означають, що береться найменше ціле число, яке більше чи дорівнює значенню виразу в дужках.

За кодом (системою числення), в якому працює лічильник, вони поділяються на:

- двійкові;
- двійково-десяткові;
- лічильники, що працюють в інших кодах.

Залежно від напрямку підрахунку лічильники поділяються на:

- підсумовувальні;
- віднімаючі;
- реверсивні, що можуть працювати в двох режимах – додавання і віднімання.

За способом організації схеми лічильники бувають:

- асинхронними:
- синхронними.

Основними параметрами лічильників є ємність і швидкодія. Ємність характеризується модулем лічби $M_л$. Швидкодія характеризується розрізняльною здатністю і часом реєстрації.

Розрізняльна здатність лічильника – це мінімальний період надходження вхідних імпульсів, за якого забезпечується надійна робота лічильника. Інакше кажучи, розрізняльна здатність характеризується максимальною частотою надходження вхідних імпульсів, за якої лічильник надійно працює.

Час реєстрації характеризується максимальним часовим інтервалом між моментом надходження вхідного імпульсу, чи його робочого фронту, і моментом установлення лічильника в новий стійкий стан, тобто моментом завершення перехідних процесів.

2.3.3.2. Синтез синхронних лічильників

Синтез лічильників зводиться до визначення оптимальної, в певному розумінні, структури і побудови його принципової схеми. Під оптимальною розуміють структуру лічильника, що містить мінімальну кількість тригерів і зв'язків між ними, за якої забезпечується виконання лічильником заданих функцій із заданими параметрами.

Розглянемо, на прикладах, методику синтезу деяких типів синхронних лічильників.

Синтез синхронного двійкового підсумовувального лічильника

Нехай потрібно синтезувати лічильник з модулем лічби $M_л = 8$ на синхронних JK-тригерах. Мінімальна кількість тригерів необхідна для побудови такого лічильника – $m = [\log_2 M_л] = 3$.

На початку реалізації методики синтезу формуємо таблицю функціонування (рис. 2.46, *а*). Тут Q_1^t, Q_2^t, Q_3^t – стан тригерів лічильника в попередньому такті; Q_1^{t+1}, Q_2^{t+1}, Q_3^{t+1} – стан тригерів лічильника в наступному такті. Зміна номерів станів лічильника, наведена у таблиці, відповідає підсумовувальному режиму його роботи.

Далі на основі таблиці функціонування формуємо прикладні таблиці тригерів (рис. 2.46, *б–г*). На наступному кроці синтезу будуємо карти Карно для функцій J і K усіх тригерів лічильника (рис. 2.46, *д–к*). Карти Карно формуються на основі таблиці функціонування і характеристичної таблиці JK-тригера (рис. 2.32, *г*).

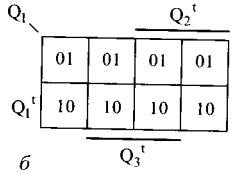
Мінімізувавши логічні функції, задані картами Карно за правилами мінімізації частково визначених функцій, отримуємо мінімальні логічні форми (рис. 2.46, *д–к*) і на їх основі будуємо схему лічильника (рис. 2.46, *л*).

Синтез синхронного двійкового віднімаючого лічильника

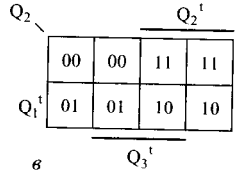
Синтез лічильника на синхронних JK-тригерах з модулем лічби $M_л = 8$ наведено на рис. 2.47.

	Номер стану							
	0	1	2	3	4	5	6	7
Q_1^t	0	1	0	1	0	1	0	1
Q_2^t	0	0	1	1	0	0	1	1
Q_3^t	0	0	0	0	1	1	1	1
Q_1^{t+1}	1	0	1	0	1	0	1	0
Q_2^{t+1}	0	1	1	0	0	1	1	0
Q_3^{t+1}	0	0	0	1	1	1	1	0

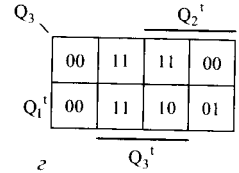
а



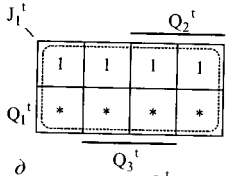
б



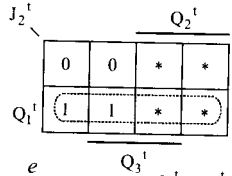
в



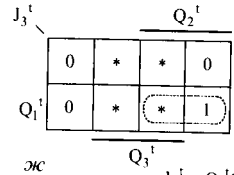
г



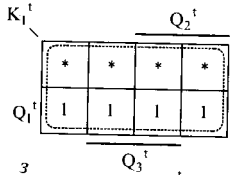
д $J_1^t = 1$



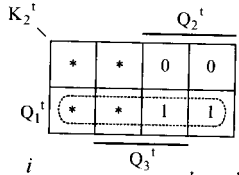
е $J_2^t = Q_1^t$



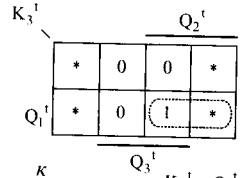
ж $J_3^t = Q_1^t Q_2^t$



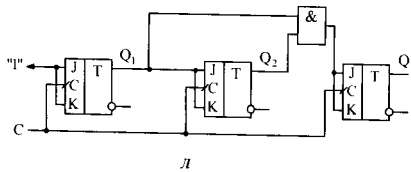
з $K_1^t = 1$



і $K_2^t = Q_1^t$



к $K_3^t = Q_1^t Q_2^t$



л

Рис. 2.46

а

	Номер стану							
	7	6	5	4	3	2	1	0
Q_1^t	1	0	1	0	1	0	1	0
Q_2^t	1	1	0	0	1	1	0	0
Q_3^t	1	1	1	1	0	0	0	0
Q_1^{t+1}	0	1	0	1	0	1	0	1
Q_2^{t+1}	1	0	0	1	1	0	0	1
Q_3^{t+1}	1	1	1	0	0	0	0	1

б

Q_1 \ Q_2^t	01	01	01	01
Q_1^t	10	10	10	10

Q_3^t

в

Q_2 \ Q_2^t	01	01	10	10
Q_1^t	00	00	11	11

Q_3^t

г

Q_3 \ Q_2^t	01	10	11	00
Q_1^t	00	11	11	00

Q_3^t

д

J_1^t \ Q_2^t	1	1	1	1
Q_1^t	*	*	*	*

Q_3^t

$J_1^t = 1$

е

J_2^t \ Q_2^t	1	1	*	*
Q_1^t	0	0	*	*

Q_3^t

$J_2^t = \overline{Q_1}^t$

ж

J_3^t \ Q_2^t	1	*	*	0
Q_1^t	0	*	*	0

Q_3^t

$J_3^t = \overline{Q_1}^t \overline{Q_2}^t$

з

K_1^t \ Q_2^t	*	*	*	*
Q_1^t	1	1	1	1

Q_3^t

$K_1^t = 1$

і

K_2^t \ Q_2^t	*	*	1	1
Q_1^t	*	*	0	0

Q_3^t

$K_2^t = \overline{Q_1}^t$

к

K_3^t \ Q_2^t	*	1	0	*
Q_1^t	*	0	0	*

Q_3^t

$K_3^t = \overline{Q_1}^t \overline{Q_2}^t$

л

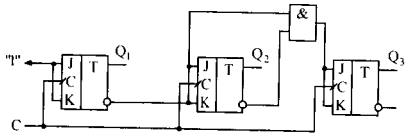


Рис. 2.47

Синтез синхронного лічильника з довільним модулем лічби

Якщо $M_л \neq 2^m$, тоді деякі стійкі стани лічильника вилучаються з його роботи. Покажемо це на прикладі синтезу лічильника з модулем лічби $M_л = 5$ на синхронних JK-тригерах (рис. 2.48). Специфіка такого синтезу полягає в тому, що в разі мінімізації логічних функцій незадіяні стани лічильника (позначені в прикладних таблицях тригерів і картах Карно рисками) і довільні значення логічних змінних (позначені зірочками) розглядаються такими, що їх можна рівноцінно використати (довизначені) за правилами мінімізації частково визначених функцій.

2.3.3.3. Асинхронні лічильники

Асинхронні лічильники характеризуються тим, що в них, на відміну від синхронних лічильників (у яких усі тригери перемикаються одночасно), тригери перемикаються один за одним в певній послідовності.

Приклади побудови таких лічильників і часові діаграми їхньої роботи наведено нижче: на рис. 2.49 – двійковий підсумовувальний лічильник; на рис. 2.50 – двійковий віднімаючий лічильник.

2.3.4. Регістри

Регістрами називаються послідовнісні пристрої, що призначені для приймання, запам'ятовування, перетворення та передавання цифрової інформації.

Залежно від введення інформації в регістр і зчитування інформації з нього регістри поділяються на:

– з послідовним входом і виходом (послідовні регістри, регістри зсуву);

– з паралельним входом і виходом (паралельні регістри, регістри пам'яті);

– комбіновані регістри, що поєднують властивості послідовних і паралельних регістрів.

	Номер стану				
	0	1	2	3	4
Q_1^t	0	1	0	1	0
Q_2^t	0	0	1	1	0
Q_3^t	0	0	0	0	1
Q_1^{t+1}	1	0	1	0	0
Q_2^{t+1}	0	1	1	0	0
Q_3^{t+1}	0	0	0	1	0

а

Q_1 \ Q_2^t				
	01	00	–	01
Q_1^t 10	–	–	10	

Q_3^t

б

Q_2 \ Q_2^t				
	00	00	–	11
Q_1^t 01	–	–	10	

Q_3^t

в

Q_3 \ Q_2^t				
	00	10	–	00
Q_1^t 00	–	–	01	

Q_3^t

г

J_1^t \ Q_2^t				
	1	0	–	1
Q_1^t *	–	–	*	

Q_3^t

$J_1^t = \overline{Q_3}^t$

д

J_2^t \ Q_2^t				
	0	0	–	*
Q_1^t 1	–	–	*	

Q_3^t

$J_2^t = Q_1^t$

е

J_3^t \ Q_2^t				
	0	*	–	0
Q_1^t 0	–	–	1	

Q_3^t

$J_3^t = Q_1^t Q_2^t$

ж

K_1^t \ Q_2^t				
	*	*	–	*
Q_1^t 1	–	–	1	

Q_3^t

$K_1^t = 1$

з

K_2^t \ Q_2^t				
	*	*	–	0
Q_1^t *	–	–	1	

Q_3^t

$K_2^t = Q_1^t$

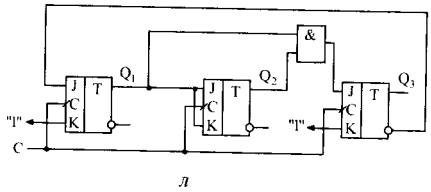
і

K_3^t \ Q_2^t				
	*	1	–	*
Q_1^t *	–	–	*	

Q_3^t

$K_3^t = 1$

к



л

Рис. 2.48

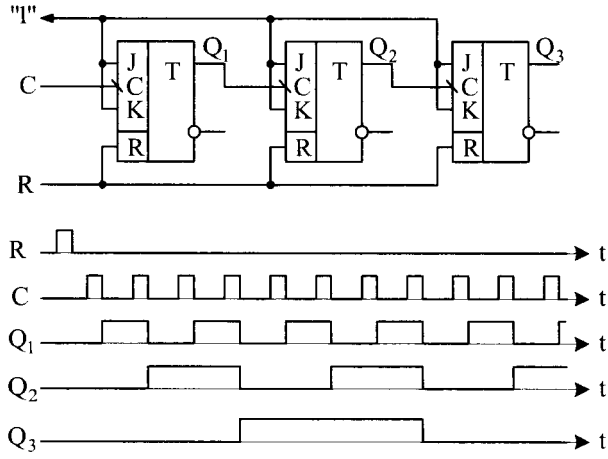


Рис. 2.49

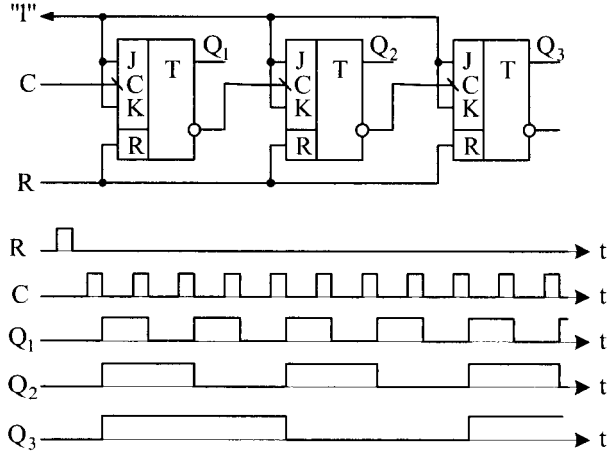
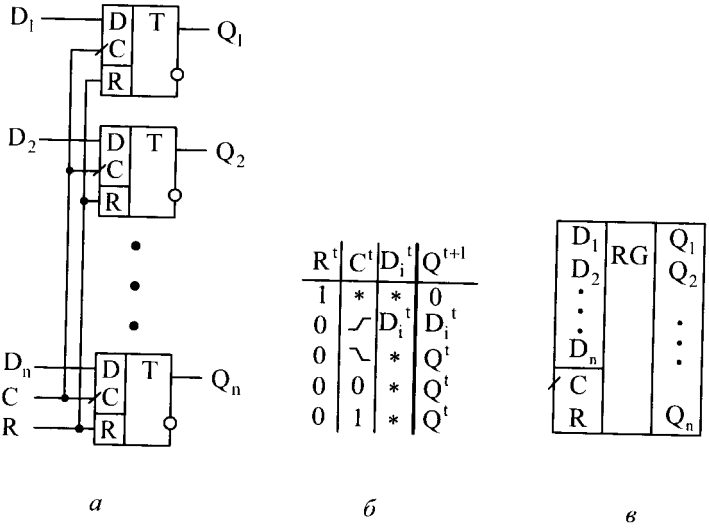


Рис. 2.50

2.3.4.1. Регістри пам'яті

Схему регістра пам'яті, таблицю, що пояснює його роботу, і умовне позначення наведено на рис. 2.51.



а *б* *в*

Рис. 2.51

Під час надходження на вхід R логічної одиниці (додатного імпульсу) усі тригери регістра встановлюються у нульовий стан – на прямих виходах тригерів формуються логічні одиниці. За наявності логічного нуля на вході R і надходження фронту тактового імпульсу на вхід C інформація з входів $D_1, D_2, ..., D_n$ записується в регістр, тобто тригери установлюються в стани, що відповідають вхідній інформації.

2.3.4.2. Регістри зсуву

Регістри зсуву можна умовно поділити на регістри зсуву вправо, регістри зсуву вліво, реверсивні регістри (що можуть працювати в двох режимах – зсув вправо і зсув вліво) і кільцеві регістри.

Приклад реалізації регістра зсуву вправо і часові діаграми його роботи наведені на рис. 2.52.

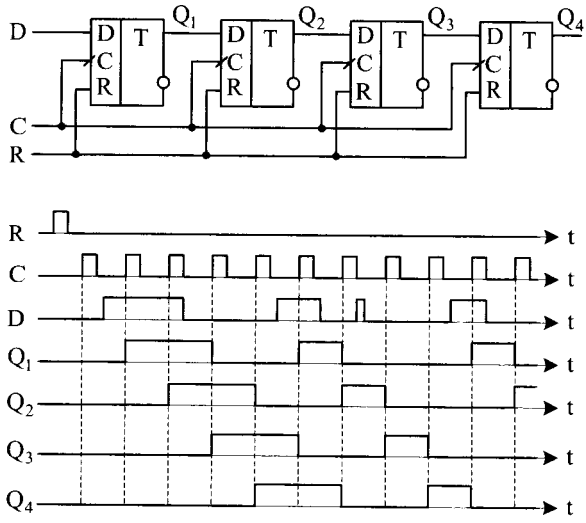
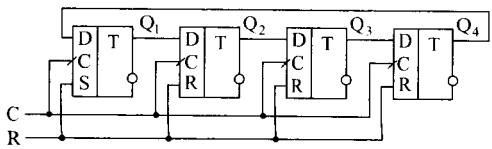


Рис. 2.52

Інформативний сигнал, що надходить на вхід D, прив'язується до фронтів татових імпульсів і з кожним тактом зсувається на один розряд регістра вправо.

Різновидом регістра зсуву є кільцевий регістр, один з можливих варіантів схеми якого і таблицю, що пояснює його роботу, наведено на рис. 2.53.



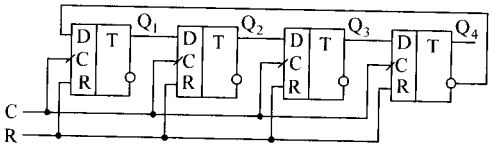
а

R^t	C^t	Q_1^{t+1}	Q_2^{t+1}	Q_3^{t+1}	Q_4^{t+1}
1	*	1	0	0	0
0	⌐	0	1	0	0
0	⌐	0	0	1	0
0	⌐	0	0	0	1
0	⌐	1	0	0	0
0	⌐	0	1	0	0
0	⌐	0	0	1	0

б

Рис. 2.53

Якщо в схему кільцевого регістру внести незначні зміни, а саме сигнал зворотного зв'язку знімати не з прямого, а з інверсного виходу тригера останнього справа розряду, отримаємо так званий лічильник Джонсона (рис. 2.54).



а

R^t	C^t	Q_1^{t+1}	Q_2^{t+1}	Q_3^{t+1}	Q_4^{t+1}
1	*	0	0	0	0
0	⌐	1	0	0	0
0	⌐	1	1	0	0
0	⌐	1	1	1	0
0	⌐	1	1	1	1
0	⌐	0	1	1	1
0	⌐	0	0	1	1
0	⌐	0	0	0	1
0	⌐	0	0	0	0

б

Рис. 2.54

Питання і задачі для самоконтролю

1. Залежності між якими змінними описують рівняння виходів і переходів послідовнісних пристроїв?
2. Чим відрізняються синхронні й асинхронні послідовнісні пристрої?
3. Скільки стійких внутрішніх станів має тригер?
4. Чим відрізняються синхронні й асинхронні тригери?
5. Сформулюйте логіку роботи RS-тригера.
6. Сформулюйте логіку роботи JK-тригера.
7. Сформулюйте логіку роботи D-тригера.
8. Сформулюйте логіку роботи T-тригера.
9. Назвіть основні типи лічильників.
10. Якими параметрами характеризується швидкодія лічильників?
11. Чим відрізняються синхронні й асинхронні лічильники?
12. Скільки тригерів необхідно для побудови лічильника з модулем лічби 18?
13. Синтезувати на базі JK-тригерів, з використанням логічних елементів І, АБО, НЕ, синхронний лічильник, що має таку послідовність зміни номерів станів: 0, 2, 1, 5, 3, 7, 6, 4, 0... .
14. Назвіть основні типи регістрів.

3. ЗАСТОСУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ У ЗАСОБАХ ЗАХИСТУ ІНФОРМАЦІЇ

Цифрові пристрої (ЦП) надзвичайно широко застосовують у різних технічних галузях, від побутових пристроїв до суперкомп'ютерів і космічної техніки. Завдяки своїм властивостям вони забезпечують високий рівень характеристик пристроїв: надійність і точність роботи в різних умовах, високі швидкодію і продуктивність, невеликі габарити, можливість оперативної зміни виконуваної функції тощо.

ЦП є також одним з основних елементів побудови систем захисту інформації і, зокрема, криптографічних пристроїв. Серед останніх на особливу увагу заслуговують генератори псевдовипадкових чисел і бітових послідовностей.

3.1. Генератори випадкових і псевдовипадкових чисел

У криптографії використовують як генератори випадкових чисел (ГВЧ), так і генератори псевдовипадкових чисел (ГПВЧ). Принциповою відмінністю ГПВЧ від ГВЧ є те, що псевдовипадкова послідовність чисел може бути відновлена у просторі та часі без попереднього її запису. Випадкова послідовність може бути відновлена тільки якщо її попередньо записати.

Робота ГВЧ ґрунтується на використанні джерел ентропії (невизначеності) – теплового шуму в електронних і напівпровідникових приладах, фотоелектричного ефекту, інших квантових явищ. Ці процеси, в теорії, абсолютно непередбачувані. Генератори основані на квантових процесах, зазвичай складаються із спеціального підсилювача і перетворювача. Підсилювач підсилює слабкі сигнали, одержувані в результаті певних фізичних явищ, до прийнятних розмірів, які перетворюються далі до цифрової форми. Отже, ГВЧ, принципово, не можуть бути побудовані винятково на ЦП.

ГПВЧ належать до детермінованих пристроїв. Послідовність чисел, яку вони генерують, можна передбачити і повторити. Однак, за виконання певних правил їхньої побудови, ця послідовність, за своїми статистичними характеристиками, практично не відрізняється від істинно випадкової. ГПВЧ можна реалізувати як програмними методами, так і за допомогою ЦП.

Окремо треба зазначити, що разом з генераторами чисел використовують генератори бітових послідовностей, у яких вихідний сигнал набуває тільки двох значень – 0 чи 1. Генератори псевдовипадкових бітових послідовностей, як правило, реалізуються на основі ГПВЧ.

У криптографічних системах ГВЧ і ГПВЧ використовують для:

- формування ключової інформації, на секретності і якості якої основується стійкість криптоалгоритмів;
- побудови потокових та інших шифрів;
- хешування інформації;
- формування цифрового підпису тощо.

3.2. Приклади побудови ГПВЧ

Існує багато різних типів структур ГПВЧ. Нижче наведено дві структури, які є, можливо, найпростішими, але такими, що їх достатньо часто використовують для побудови складніших і досконаліших генераторів.

3.2.1. ГПВЧ на основі регістрів зсуву

На рис. 3.1 наведено схему ГПВЧ на основі регістра зсуву. Їх подано в двох варіантах, що відрізняються способом зображення зв'язків між елементами схеми. Варіант на рис. 3.1, *б* є зручнішим для зображення складних, більш громіздких схем.

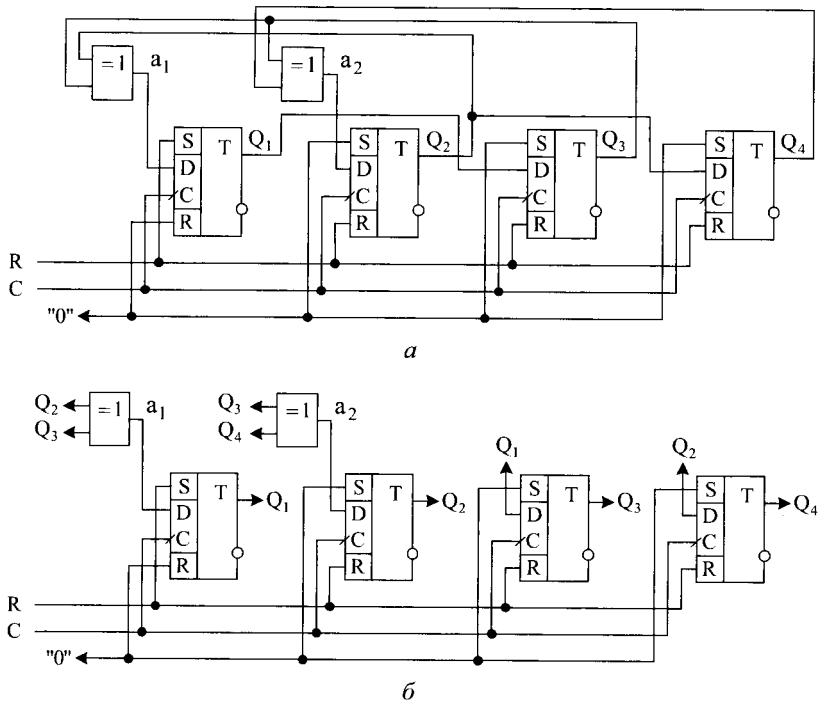
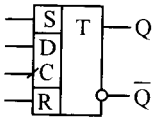


Рис. 3.1. ГПВЧ на основі регістра зсуву

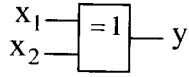
Аналіз роботи генератора здійснюється на основі таблиці переходів комбінованого синхронного RSD тригера, що працює по фронту тактового імпульсу (рис. 3.2, *а*) і таблиці істинності логічного елемента – суматора за модулем 2 (рис. 3.2, *б*).

На відміну від класичного регістра зсуву (п. 2.3.4.2), у якому тригери з'єднані послідовно (D-вхід кожного наступного тригера під'єднаний до виходу попереднього тригера), в наведеній схемі зв'язки між тригерами встановлюються за певними правилами. Існує теорія побудови таких схем, з якою можна ознайомитись, наприклад, у роботі [1].



R^t	S^t	C^t	D^t	Q^{t+1}
1	1	*	*	н/с
1	0	*	*	0
0	1	*	*	1
0	0	↗	0	0
0	0	↗	1	1
0	0	0	*	Q^t
0	0	1	*	Q^t
0	0	↘	*	Q^t

а



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

б

Рис. 3.2. Таблиця переходів RSD тригера
і таблиця істинності суматора за модулем 2

Функціонування генератора описується рівняннями:

$$\begin{aligned} Q_1(t+1) &= Q_2(t) \oplus Q_3(t), \\ Q_2(t+1) &= Q_3(t) \oplus Q_4(t), \\ Q_3(t+1) &= Q_1(t), \\ Q_4(t+1) &= Q_2(t), \end{aligned} \qquad (3.1)$$

де $Q_i(t)$ і $Q_{i+1}(t)$ – стани тригерів у поточному і наступному тактах роботи, що можуть набувати двох значень – 0 чи 1.

На рис. 3.3 наведено часові діаграми, що пояснюють роботу генератора, а в табл. 3.1 – відповідні числові дані.

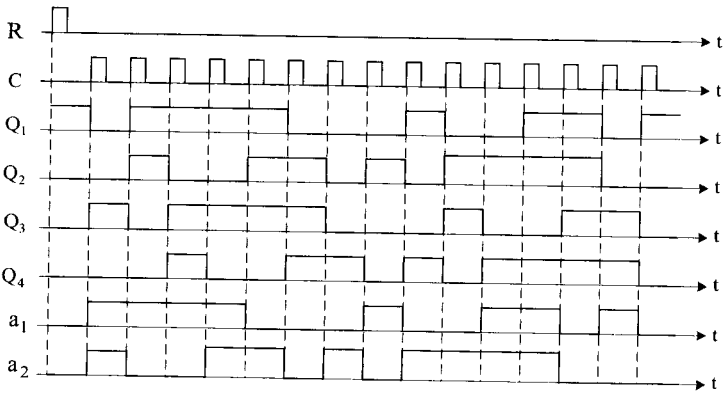


Рис. 3.3. Часові діаграми роботи ГПВЧ на основі регістра зсуву

Таблиця 3.1

Зміна станів ГПВЧ на основі регістра зсуву

i	A	Q_4	Q_3	Q_2	Q_1
		8	4	2	1
1	1	0	0	0	1
2	4	0	1	0	0
3	3	0	0	1	1
4	13	1	1	0	1
5	5	0	1	0	1
6	7	0	1	1	1
7	14	1	1	1	0
8	8	1	0	0	0
9	2	0	0	1	0
10	9	1	0	0	1
11	6	0	1	1	0
12	10	1	0	1	0
13	11	1	0	1	1
14	15	1	1	1	1
15	12	1	1	0	0

Спочатку на вхід R пристрою подається імпульс, який установлює тригери в початковий стан: перший тригер (Q_1) установлюється в одиничний стан, усі інші (Q_2 – Q_4) – в нульовий. Це пояснюється тим, що нульовий початковий стан усіх тригерів генератора заборонений, оскільки, в цьому разі, під час надходження на вхід C тактових імпульсів, стани тригерів не змінюватимуться.

У табл. 3.1 наведено номер такту "i" і стани усіх тригерів. Окрім цього, тут подано число A – число в десятковій системі числення, що відповідає двійковому коду $Q_4\,Q_3\,Q_2\,Q_1$ і визначається виразом

$$A = Q_4 \cdot 2^3 + Q_3 \cdot 2^2 + Q_2 \cdot 2^1 + Q_1 \cdot 2^0 . \qquad (3.2)$$

На рис. 3.4 наведено схему ГПВЧ на основі регістра зсуву, до складу якої входять 31 тригер і 11 суматорів за модулем 2.

Пристрій функціонує відповідно до рівнянь:

$$Q_1(t+1)=Q_8(t)\oplus Q_{21}(t),\ Q_2(t+1)=Q_9(t)\oplus Q_{22}(t),$$
$$Q_3(t+1)=Q_{10}(t)\oplus Q_{23}(t),\ Q_4(t+1)=Q_{11}(t)\oplus Q_{24}(t),$$
$$Q_5(t+1)=Q_{12}(t)\oplus Q_{25}(t),\ Q_6(t+1)=Q_{13}(t)\oplus Q_{26}(t),$$
$$Q_7(t+1)=Q_{14}(t)\oplus Q_{27}(t),\ Q_8(t+1)=Q_{15}(t)\oplus Q_{28}(t),$$
$$Q_9(t+1)=Q_{16}(t)\oplus Q_{29}(t),\ Q_{10}(t+1)=Q_{17}(t)\oplus Q_{30}(t),$$
$$Q_{11}(t+1)=Q_{18}(t)\oplus Q_{31}(t),\ Q_{12}(t+1)=Q_1(t),\ Q_{13}(t+1)=Q_2(t),$$
$$Q_{14}(t+1)=Q_3(t),\ Q_{15}(t+1)=Q_4(t),\ Q_{16}(t+1)=Q_5(t), \quad (3.3)$$
$$Q_{17}(t+1)=Q_6(t),\ Q_{18}(t+1)=Q_7(t),\ Q_{19}(t+1)=Q_8(t),$$
$$Q_{20}(t+1)=Q_9(t),\ Q_{21}(t+1)=Q_{10}(t),\ Q_{22}(t+1)=Q_{11}(t),$$
$$Q_{23}(t+1)=Q_{12}(t),\ Q_{24}(t+1)=Q_{13}(t),\ Q_{25}(t+1)=Q_{14}(t),$$
$$Q_{26}(t+1)=Q_{15}(t),\ Q_{27}(t+1)=Q_{16}(t),\ Q_{28}(t+1)=Q_{17}(t),$$
$$Q_{29}(t+1)=Q_{18}(t),\ Q_{30}(t+1)=Q_{19}(t),\ Q_{31}(t+1)=Q_{20}(t).$$

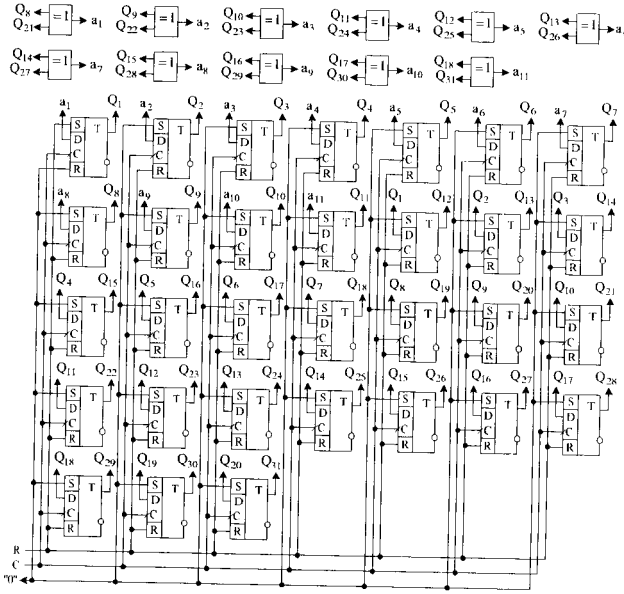


Рис. 3.4. ГПВЧ на основі регістра зсуву, що має 31 розряд

На рис. 3.5 наведено графік залежності числа A, що визначається виразом

$$A = \sum_{j=1}^{31} Q_j \cdot 2^{j-1}, \qquad (3.4)$$

від номеру такту "i" роботи генератора.

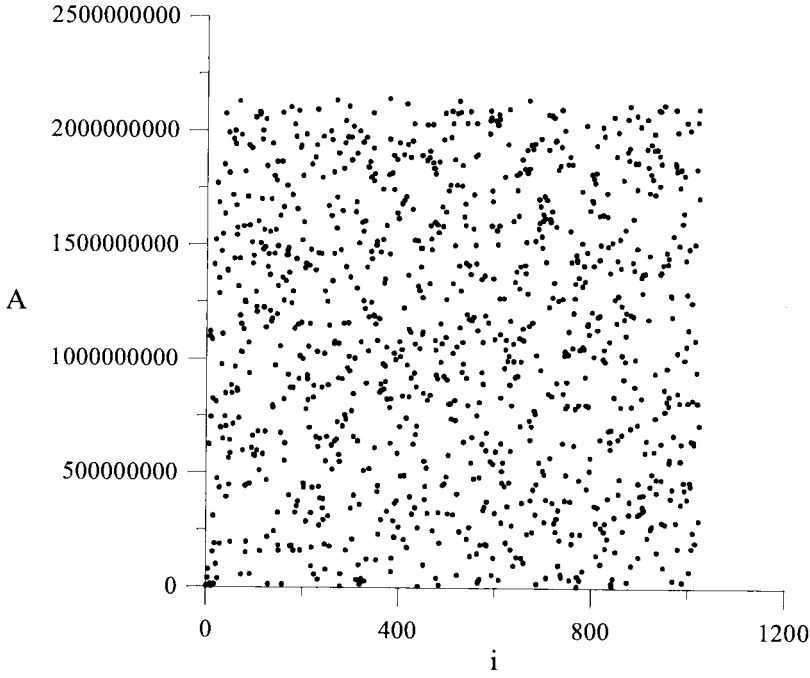


Рис. 3.5. Залежність псевдовипадкового числа A від номеру такту роботи генератора

Хаотичність розподілу послідовності чисел A на графіку (рис. 3.5) свідчить про те, що вона є наближеною до випадкової. Існують методики перевірки (тестування) послідовностей чисел на їх відповідність випадковому закону, з якими можна ознайомитись, зокрема, в роботі [1].

3.2.2. ГПВЧ на основі регістрів пам'яті та комбінаційного суматора

До складу генератора [2], схему якого наведено на рис. 3.6, входять логічна схема D1, комбінаційний суматор D2, логічні елементи D3 – D6 і регістри пам'яті D7 – D9.

Частина схеми, реалізована на логічних елементах D3 – D6, забезпечує запис в регістри D7, D8 і D9 початкових значень – $X=1$, $X1=0$, $X2=0$ відповідно, до приходу першого імпульсу на вхід C генератора. Це відбувається за допомогою вхідних сигналів R, C′, E відповідно до часової діаграми, зображеної на рис. 3.7.

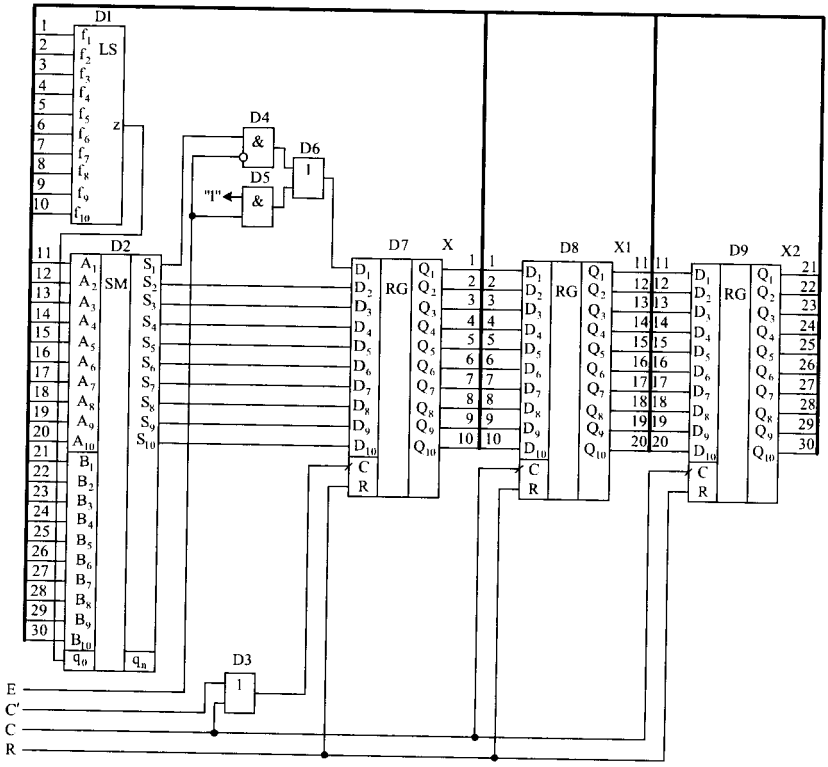


Рис. 3.6. Схема ГПВЧ на основі регістрів пам'яті та комбінаційного суматора

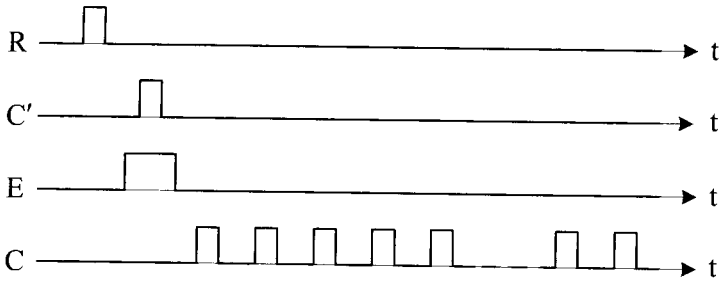


Рис. 3.7. Часова діаграма вхідних сигналів генератора

Роботу генератора можна описати рівняннями:

$$\begin{aligned} X1(t+1) &= X(t), \\ X2(t+1) &= X1(t), \\ X(t+1) &= [X1(t) + X2(t) + z] \bmod 2^n, \end{aligned} \qquad (3.5)$$

де n – кількість двійкових розрядів суматора D2 і регістрів D7 – D9; z – значення сигналу на виході логічної схеми D1. У випадку, що розглядається, $n = 10$. Тобто, з кожним імпульсом на вході C генератора число з регістра D8 переписується в регістр D9, з регістра D7 – в регістр D8, а в регістр D7 – з виходів комбінаційного суматора D2.

Вихідний сигнал логічної схеми D1 формується відповідно до логічного рівняння:

$$z = f_1 \oplus f_2 \oplus f_3 \oplus f_4 \oplus f_5 \oplus f_6 \oplus f_7 \oplus f_8 \oplus f_9 \oplus f_{10}. \qquad (3.6)$$

Для схеми, що розглядається, $f_i = Q_i$ $(i = 1,2,...,10)$, де Q_i – виходи розрядів числа X в регістрі D7.

Приклад реалізації логічної схеми LS – D1 наведено на рис. 3.8.

Початковий фрагмент функціонування ГВПЧ подано в табл. 3.2.

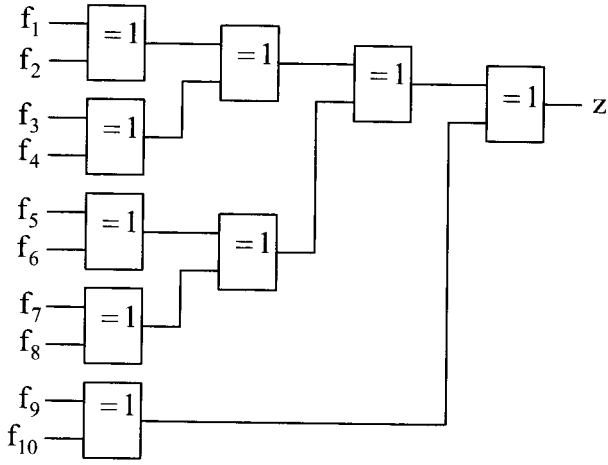


Рис. 3.8. Побудова схеми LS на суматорах за модулем 2

Таблиця 3.2

Зміна станів ГПВЧ на основі регістрів пам'яті та комбінаційного суматора

i	X	X1	X2	z	X									
					Q10	Q9	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1
					2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1
2	1	1	0	1	0	0	0	0	0	0	0	0	0	1
3	2	1	1	1	0	0	0	0	0	0	0	0	1	0
4	3	2	1	0	0	0	0	0	0	0	0	0	1	1
5	3	3	2	0	0	0	0	0	0	0	0	0	1	1
6	5	3	3	0	0	0	0	0	0	0	0	1	0	1
7	6	5	3	0	0	0	0	0	0	0	0	1	1	0
8	8	6	5	1	0	0	0	0	0	0	1	0	0	0
9	12	8	6	0	0	0	0	0	0	0	1	1	0	0
10	14	12	8	1	0	0	0	0	0	0	1	1	1	0
11	21	14	12	1	0	0	0	0	0	1	0	1	0	1
12	27	21	14	0	0	0	0	0	0	1	1	0	1	1
13	35	27	21	1	0	0	0	0	1	0	0	0	1	1
14	49	35	27	1	0	0	0	0	1	1	0	0	0	1

Продовження табл. 3.2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	63	49	35	0	0	0	0	0	1	1	1	1	1	1
16	84	63	49	1	0	0	0	1	0	1	0	1	0	0
17	113	84	63	0	0	0	0	1	1	1	0	0	0	1
18	147	113	84	0	0	0	1	0	0	1	0	0	1	1
19	197	147	113	0	0	0	1	1	0	0	0	1	0	1
20	260	197	147	0	0	1	0	0	0	0	0	1	0	0
21	344	260	197	0	0	1	0	1	0	1	1	0	0	0
22	457	344	260	1	0	1	1	1	0	0	1	0	0	1
23	605	457	344	0	1	0	0	1	0	1	1	1	0	1
24	801	605	457	0	1	1	0	0	1	0	0	0	0	1
25	38	801	605	1	0	0	0	0	1	0	0	1	1	0
26	383	38	801	0	0	1	0	1	1	1	1	1	1	1
27	839	383	38	0	1	1	0	1	0	0	0	1	1	1
28	421	839	383	1	0	1	1	0	1	0	0	1	0	1
29	199	421	839	1	0	0	1	1	0	0	0	1	1	1
30	237	199	421	0	0	0	1	1	1	0	1	1	0	1

На рис. 3.9 наведено графік залежності числа X від номеру такту "i" роботи генератора.

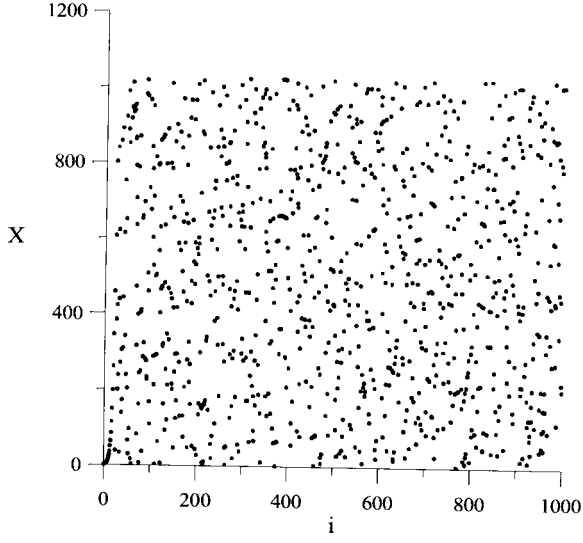


Рис. 3.9. Залежність псевдовипадкового числа X від номера такту роботи генератора

Питання і задачі для самоконтролю

1. Чим відрізняються генератори псевдовипадкових чисел від генераторів випадкових чисел?
2. Для чого в криптографічних системах використовують генератори випадкових і псевдовипадкових чисел?
3. Чим відрізняється генератор псевдовипадкових чисел на основі регістра зсуву від звичайного регістра зсуву?
4. Який стан тригерів генератора псевдовипадкових чисел на основі регістра зсуву є забороненим?
5. Графіками яких залежностей можна проілюструвати роботу генераторів псевдовипадкових чисел, підтвердивши, що їхні статистичні характеристики наближаються до випадкових?

Література

1. Иванов М. А. Криптографические методы защиты информации в компьютерных системах и сетях: учеб. пособ. / М. А. Иванов, И. В. Чугунков; под ред. М. А. Иванова. – М.: НИЯУ МИФИ, 2012. – 400 с.

2. Костів Ю. Апаратна реалізація і дослідження модифікованих генераторів Фібоначчі / Ю. Костів, В. Максимович, О. Гарасимчук, М. Мандрона // Вісник Української Академії Друкарства "Комп'ютерні технології друкарства". – 2013. – № 29. – С. 167–174.

4. ПРОГРАМОВАНІ ЛОГІЧНІ ІНТЕГРАЛЬНІ СХЕМИ (ПЛІС)

Розглянуті в другому розділі програмовані логічні матриці є раннім різновидом ширшого класу цифрових пристроїв – програмованих логічних інтегральних схем (ПЛІС). ПЛІС (Programmable Logic Integrated Circuits – PLIC) за останні десятиліття зпрогресували від програмованих постійних запам'ятовувальних пристроїв (ППЗП) (Programmable Read Only Memory – PROM) до систем на програмованому кристалі (СНПК) (System On Programmable Chip – SOPC).

ПЛІС класифікують за різними ознаками – архітектурою, технологією виготовлення, ступенем інтеграції, способами програмування, системними властивостями тощо. Обмежимося тут розглядом особливостей побудови і властивостей ПЛІС, класифікованих за архітектурою (класифікація на рис. 4.1).

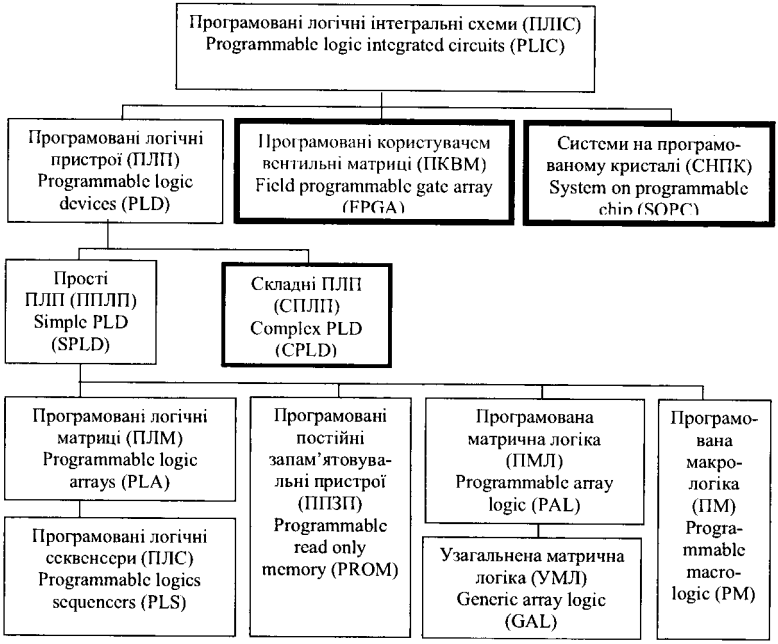


Рис. 4.1. Класифікація ПЛІС

До класу програмованих логічних інтегральних схем належать прості та складні програмовані логічні пристрої (ПЛП) – SPLD (Simple Programmable Logic Devices) і CPLD (Complex Programmable Logic Devices), а також програмовані користувачем вентильні матриці (FPGA – Field Programmable Gate Arrays) і вже згадані системи на програмованому кристалі SOPC.

Сьогодні ПЛІС є потужним і гнучким засобом побудови цифрових пристроїв найвищої складності. Ступінь інтеграції кристалів ПЛІС сягає мільйонів еквівалентних вентилів, працюють сучасні ПЛІС на гігагерцових частотах. Фірми-виробники супроводжують свої ПЛІС досконалими засобами розробки у вигляді САПР, які дають змогу: здійснювати опис пристрою різними способами (схемними, табличними, графовими, мовними (VHDL)); досліджувати розроблений пристрій на різних рівнях (логічному, функціональному і навіть фізичному); трасувати зв'язки ПЛІС і виготовляти розроблений пристрій за допомогою програмування ПЛІС. Причому для реалізації всіх цих етапів проектування та дослідження сьогодні достатньо звичайного домашнього персонального комп'ютера.

4.1. SPLD – прості програмовані логічні пристрої

Структуру більшості SPLD умовно можна подати у вигляді сукупності двох матриць взаємоортогональних провідників: матриці І (AND) і матриці АБО (OR). Така схема подібна до структури, поданої на рис. 2.12 (власне ПЛМ – один з видів ПЛІС типу SPLD). Вхідні сигнали зазвичай надходять на парафазні (пряме та інверсне значення вхідного сигналу) входи матриці І, яка на ортогональних шинах дає змогу реалізувати будь-які кон'юнкції вхідних змінних. Виходи матриці І з'єднані з входами матриці АБО, яка на своїх виходах реалізує диз'юнкції поданих кон'юнкцій. Сукупність вихідних шин матриці І утворює множину проміжних елементарних кон'юнкцій, які називають ще кон'юнктивними термами або просто термами (terms).

Так, структура SPLD програмуванням з'єднань між провідниками може програмуватися на реалізацію системи логічних функцій, заданих у диз'юнктивній нормальній формі. Різні типи SPLD відрізняються можливостями програмування матриці І та матриці АБО. Залежно від того, яка матриця програмується (матриця І, матриця АБО чи обидві), SPLD прийнято поділяти на чотири класи (рис. 4.1): програмовані логічні матриці ПЛМ (PLA); програмовані постійні запам'ятовувальні пристрої ППЗП (PROM); програмована матрична логіка ПМЛ (PAL) і програмована макрологіка ПМ (PM).

Ми вже знаємо, що в ПЛМ / PLA (спрощена структура на рис. 4.2) програмуються обидві матриці: І та АБО. Прикладами таких ПЛІС є вітчизняні мікросхеми К556РТ1, РТ2, РТ21. Недоліком ПЛМ / PLA є слабке використання ресурсів програмованої матриці АБО (адже логічні функції заданої системи часто містять небагато термів).

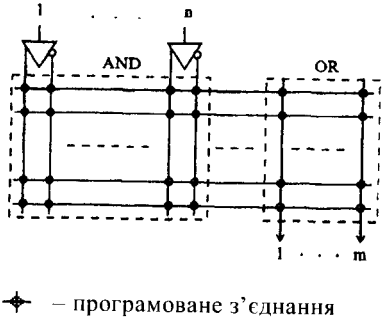


Рис. 4.2. Структура ПЛМ /PLA

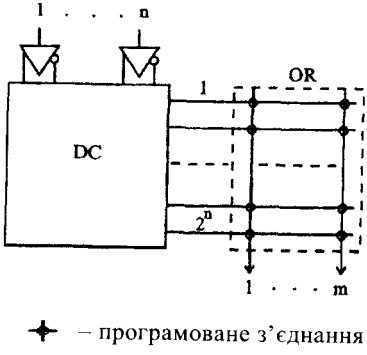


Рис. 4.3. Структура ППЗП / PROM

У ППЗП / PROM (рис. 4.3) матриця І постійно налаштована на функції повного дешифратора (DC), тобто формує всі мінтерми від вхідних змінних, а програмується тільки матриця АБО.

У структурі ПМЛ / PAL (рис. 4.4), навпаки, програмується тільки матриця І, а матриця АБО має фіксовану конфігурацію, за

якої q проміжних шин (кон'юнкцій) об'єднуються функцією АБО і зв'язуються з одним виходом. Це дає змогу матрицю АБО реалізувати у вигляді сукупності q – входових диз'юнкторів. На перший погляд, може видатися, що ПМЛ/PAL поступаються ПЛМ / PLA за своїми можливостями, оскільки вводиться обмеження на кількість термів у диз'юнктивній нормальній формі. Практика ж показує, що це обмеження в більшості застосувань не критичне. Адже за необхідності можна об'єднати кілька виходів ПМЛ, наприклад, диз'юнктором, збільшивши кількість термів понад q.

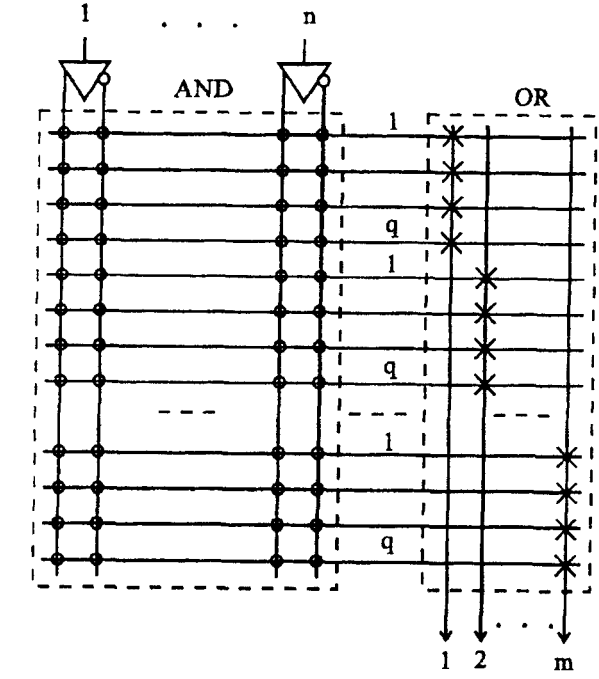


Рис. 4.4. Структура ПМЛ /PAL

З іншого боку, в ПМЛ / PAL програмується тільки одна матриця І, що значно спрощує структуру ПМЛ / PAL і, як наслідок, приводить до зниження вартості пристрою і підвищення його швидкодії. Крім того, спрощення матриці АБО дало змогу додати в структуру ПМЛ / PAL кола зворотного зв'язку і вихідні буфери, завдяки чому ПМЛ / PAL набула додаткових переваг.

ПМЛ / PAL мають архітектуру, дуже зручну для реалізації цифрових автоматів. До класу ПМЛ / PAL належить більшість ПЛІС невеликого ступеня інтеграції. Як приклад, можна навести вітчизняні інтегральні схеми КМ1556ХП4, ХП6, ХП8, ХЛ8, ранні розробки середини–кінця 80-х років ПЛІС фірм Intel, Altera, AMD, Lattice тощо. Різновидом класу ПМЛ / PAL є ПЛІС, що мають тільки одну (програмовану) матрицю І, наприклад, схема 85С508 фірми Intel.

Програмовані логічні секвенсери ПЛС / PLS (рис. 4.1) мають структуру, подібну до ПЛМ / PLA, а узагальнені матричні логіки УМЛ / GAL подібні до ПМЛ / PAL.

Програмована макрологіка ПМ / PM (рис. 4.1) містить єдину програмовану матрицю елементів Шефера або Пірса (логічних елементів І-НЕ чи АБО-НЕ), чого достатньо для реалізації будь-яких логічних функцій (див. § 1.6). Додатково, за рахунок численних інверсних зворотних зв'язків, ПМ здатна формувати складні логічні функції. До цього класу належать, наприклад, ПЛІС PLHS501 і PLHS502 фірми Signetics, що мають матрицю І-НЕ, а також ПЛІС XL78C800 фірми Exel, побудована на матриці АБО-НЕ.

Розглянуті архітектури ПЛІС типу SPLD містять невелику кількість еквівалентних вентилів, і на цей час вони морально застаріли, але далі застосовуються для реалізації порівняно простих пристроїв, для яких не існує готових ІС середнього ступеня інтеграції. Для реалізації сучасних алгоритмів, наприклад, алгоритмів цифрової обробки сигналів, вони не придатні. Сьогодні найпоширеніші сімейства ПЛІС мають архітектуру CPLD і FPGA.

4.2. CPLD – складні програмовані логічні пристрої

CPLD – це ПЛІС середнього ступеня інтеграції (до 50000 еквівалентних вентилів). Структурно CPLD будують на базі функціональних блоків (functional block – FB) з архітектурою ПМЛ/PAL. Функціональні блоки об'єднуються програмованою комутаційною матрицею (Switch Matrix – SM). Тобто ПЛІС CPLD-типу – це набір блоків ПМЛ (рис. 4.4), які можуть об'єднуватися в різній конфігурації за допомогою засобів комутаційної матриці.

Узагальнена структура CPLD подана на рис. 4.5. Кожен функціональний блок FB має свою множину двонапрямлених виводів, якими передаються оброблювані сигнали з комутаційної матриці і в неї. Основні логічні перетворення виконуються в FB, а комутаційна матриця слугує для передавання сигналів між FB. Крім того, структура CPLD має спеціалізовані входи, які пов'язані з комутаційною матрицею і з усіма FB. Ці входи зазвичай використовуються для глобальних сигналів управління пристроєм: d_C – для синхронізації; d_{OE} – для управління вихідними буферами і d_I, які можна використовувати як звичайні логічні входи.

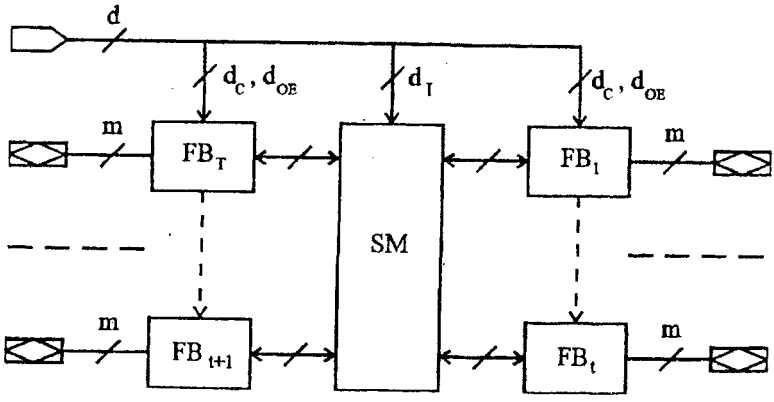


Рис. 4.5. Спрощена структура CPLD

Очевидно, що ПЛІС з архітектурою CPLD мають значно ширші функціональні можливості порівняно з SPLD за рахунок великої кількості функціональних блоків. Також перевагами CPLD є:

- висока швидкодія, оскільки оброблювані сигнали тільки один раз (під час введення) проходять через комутаційну матрицю;
- можливість для кожної вихідної макроячейки прямого управління своїм зовнішнім виводом.

Разом з тим архітектурі CPLD притаманні такі недоліки:

- передавання сигналів між FB неминуче пов'язане з використанням ресурсів комутаційної матриці;
- значне зростання площі комутаційної матриці в разі збільшення числа FB і зовнішніх виводів.

Прикладами мікросхем архітектури CPLD є багаторазові матричні таблиці (Multiple Array matriX – MAX) сімейств MAX3000, MAX7000 і MAX9000, MAX2, MAX5, які випускаються фірмою Altera. До класу CPLD також належать ПЛІС фірми Xilinx сімейств XC7000, XC9500, XC9500XL, XC9500XV, а також багато ПЛІС інших виробників.

4.3. FPGA – програмовані користувачем вентильні матриці

Це один з двох найпоширеніших типів архітектури сучасних ПЛІС. FPGA має більше логічних блоків і більш гнучку архітектуру, ніж CPLD. Наприклад, сучасні ПЛІС FPGA сімейства Virtex 7 фірми Xilinx мають більше від мільйона логічних блоків.

FPGA складається з матриці конфігурованих логічних блоків і комутувальних зв'язків – програмованих матриць з'єднань. Логічні блоки таких ПЛІС складаються з одного або декількох логічних елементів. У їхню основу покладено таблицю перекодування (ТП – Look – up table, LUT), програмований мультиплексор, D-тригер, а також кола управління. Також деякі сімейства FPGA мають вбудовані реконфігуровані модулі пам'яті (РМП – Embedded

Array Block, EAB), що значно розширює їхні функціональні можливості.

Прикладами ПЛІС архітектури FPGA є:

– мікросхеми фірми Xilinx сімейств: XC2000, XC3000, XC4000, XC5000, Spartan, Spartan-XL, Spartan-II, Spartan-IIE, Spartan-3, Spartan-3E, Spartan-3A, Spartan-3AN, Spartan-3ADSP, Spartan-6, Virtex, Virtex-E, Virtex-II, VirtexIIPro, Virtex4, Virtex5, Virtex6, Artix7, Kintex7,Virtex7;

– мікросхеми фірми Altera сімейств: FLEX6000, FLEX8000, FLEX10K, Stratix, StratixII, StratixII GX, StratixIII, StratixIV, Stratix5 GT, Stratix5 GX, Stratix5 GS, Stratix5 E, Cyclone 2, Cyclone 3, Cyclone 4, Cyclone 5.

Узагальнену труктуру ПЛІС з архітектурою FPGA подано на рис. 4.6. FPGA складається з прямокутної матриці конфігурованих логічних блоків (Configurable Logic Block – CLB – на рисунку LE), оточених блоками введення–виведення (Input / Output Block – IOB), які дають змогу реалізувати двонапрямлене введення–виведення, третій стан тощо. Між CLB розташовуються вертикальні та горизонтальні трасувальні лінії або міжз'єднання (Interconnection). На кристалах багатьох FPGA є внутрішній швидкодіючий інвертуючий підсилювач, що дає можливість за допомогою зовнішнього кварцового резонатора і двох резисторів створювати кварцовий генератор, який використовується в проектованому пристрої. Схема генератора активується на початку завантаження конфігурації, що дає змогу стабілізувати генератор.

Блоки введення–виведення БВВ сучасних ПЛІС, крім згаданих функцій, підтримують до 20 різних електричних інтерфейсів (TTL, CMOS та ін.) та високошвидкісних інтерфейсів обміну даними, що спрощує узгодження ПЛІС з іншими цифровими пристроями в складі різних проектів. Водночас виділення функцій введення–виведення для виконання окремими блоками унеможливлює 100 % використання логічних ресурсів ПЛІС архітектури FPGA.

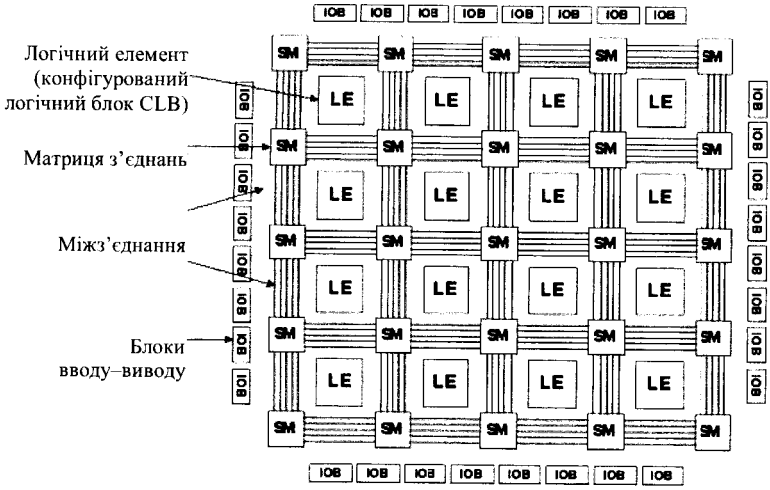


Рис. 4.6. Структура ПЛІС з архітектурою FPGA

Конфігуровані логічні блоки КЛБ (LE на рис. 4.6) призначені для виконання логічних функцій від декількох змінних, а також функцій пам'яті. Причому логічні функції в таких блоках реалізуються за допомогою програмованого мультиплексора, який вибирає значення функції для поданого набору значень логічних змінних. Значення функції для різних значень аргументів зберігаються в пам'яті блоку. Тобто FPGA, на відміну від CPLD, мають елементи пам'яті, а тому можуть застосовуватися для побудови не тільки комбінаційних, але й послідовнісних цифрових пристроїв.

Програмовані міжз'єднання в FPGA дають змогу об'єднувати входи і виходи будь-яких БВВ і КЛБ. Міжз'єднання – це мережа вертикальних і горизонтальних металевих сегментів, у місці перетину яких розташовані програмовані перемикаючі точки (транзистори – англ. *Phisical Interconnect Point* (PIP) – точка фізичного міжз'єднання) (матриці з'єднань SM на рис. 4.6). Це дає можливість реалізувати практично будь-який необхідний маршрут зв'язку і отримати для критичних зв'язків затримку менше від 0,1 нс. Для розведення по всьому кристалу сигналів з мінімальною затримкою слугують так звані довгі лінії (ДЛ – англ. *LL – Long Line*) і тактові буфери.

Кола міжз'єднань слугують у FPGA для формування складних логічних функцій і побудови вузлів, що складаються з багатьох КЛБ і БВВ. Логічні функції FPGA і міжз'єднання визначаються даними, що зберігаються у внутрішніх статичних запам'ятовувальних елементах (в "тіньовому" ЗП). Особливістю FPGA є можливість перепрограмування функцій КЛБ, БВВ і за допомогою міжз'єднань перезавантаження у внутрішній ("тіньовий") ЗП інформації про її конфігурацію. Це дає змогу отримувати різні пристрої на одному і тому самому кристалі FPGA навіть у динамічному режимі, тобто протягом малого часу і під час роботи мікросхеми у складі пристрою.

Основним функціональним елементом FPGA є конфігурований логічний блок. Структури таких блоків у ПЛІС різних сімейств та виробників можуть відрізнятися. Для пояснення принципів відтворення такими блоками логічних функцій розглянемо спрощений приклад структури КЛБ, зконфігурованого для відтворення функції нерівнозначності двох логічних змінних – A і B (рис. 4.7).

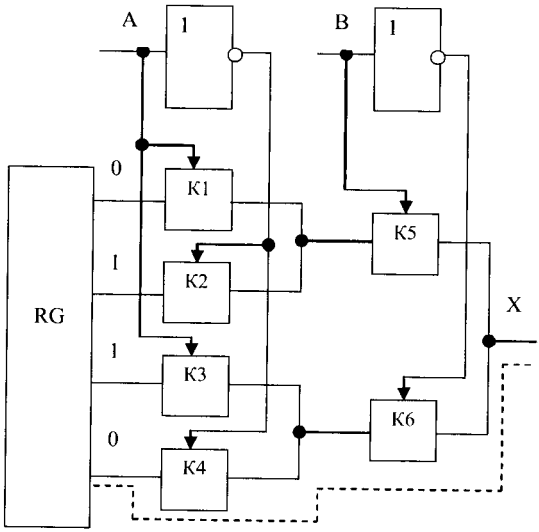


Рис. 4.7. Структура КЛБ
для відтворення функції нерівнозначності

Для реалізації необхідної функції в регістр RG (елементи пам'яті "тіньового" ЗП) записується потрібна інформація – значення функції для наборів значень вхідних змінних (таблиця істинності). У нашому прикладі (для функції нерівнозначності) – це 0110. Ключі К1-К6 (транзистори) відкриваються високим рівнем керуючого сигналу і під'єднують до виходу X відповідні розряди регістра. Наприклад, якщо A = 0 і B = 0, то відкриті ключі К2, К4 і К6. У цьому випадку сигнал на виході визначається станом четвертого розряду регістра (на рисунку цей шлях показаний пунктиром), тобто X = 0.

Повна таблиця істинності для цього стану регістра виглядає так (для кожного набору значень аргументів A і B вказано також шлях через відкриті ключі від розрядів регістра до виходу КЛБ):

B	A	X	
0	0	0	Через відкриті ключі К4,К6
0	1	1	Через відкриті ключі К3,К6
1	0	1	Через відкриті ключі К2,К5
1	1	0	Через відкриті ключі К1,К5

Тобто ключі й інвертори в структурі на рис. 4.7 фактично виконують функцію мультиплексора з двома адресними входами і чотирма входами даних. Відповідно на виході схеми КЛБ справді реалізується функція нерівнозначності:

$$X = \overline{A}B + A\overline{B}.$$

Для реалізації іншої логічної функції в регістр достатньо записати послідовність значень іншої функції.

Реальні КЛБ можуть подібно відтворювати функції більшого числа змінних – наприклад, функцію чотирьох змінних або одночасно дві функції чотирьох змінних. В останньому випадку з'являється можливість за допомогою застосування методу суперпозиції відтворювати також функції більшого числа змінних. КЛБ можуть містити тригери на виході, а вхідні регістри можуть конфігуруватися як пам'ять різного типу, регістри зсуву тощо.

Загалом FPGA мають більшу ємність і ширші можливості, ніж CPLD. Водночас FPGA застосовують для побудови доволі складні цифрові пристрої, через що часто ресурсів FPGA виявляється недостатньо для реалізації комбінаційної логіки таких пристроїв. Тому на практиці цифрові пристрої будують на парі ПЛІС – FPGA і CPLD.

4.4. SOPC – системи на програмованому кристалі

SOPC – це ПЛІС найвищого ступеня інтеграції, що містять мільйони еквівалентних вентилів. Такий високий рівень інтеграції досягається за допомогою найсучасніших технологічних процесів. На основі прогресивних технологічних процесів забезпечуються також одночасно високий рівень інтеграції і висока швидкодія. У результаті стає можливою інтеграція на одному кристалі цілої високопродуктивної системи.

За архітектурою SOPC належать до структур комбінованого типу, у яких поєднуються ознаки FPGA і CPLD з перевагою ознак FPGA. SOPC мають також додаткові блоки, часто аналогові, що істотно розширює їхні функціональні можливості. Однак не структурні особливості визначають основні переваги SOPC, а радше нові підходи до проектування пристроїв на таких ПЛІС. Через те, що SOPC мають дуже великий обсяг і широкі функціональні можливості, розробляти пристрій, орієнтований на таку ПЛІС, "з нуля" майже неможливо. Такі розробки здійснюють у межах нового підходу до проектування, який називають "проектуванням систем на кристалі". Спрощено суть цього підходу полягає в широкому застосуванні готових компонентів систем, які можна інтегрувати на кристал. Відповідно значна частина розробників працює тільки над наповненням бібліотек таких компонентів. Значну увагу звертають на параметричну гнучкість компонентів, їхні інтерфейси тощо. Такі компоненти називають одиницями інтелектуальної власності (Intellectual Properties) або

IP-ядрами. Розробники кінцевих пристроїв під час проектування користуються цими бібліотеками, вибираючи необхідні готові IP-блоки та інтегруючи їх у свій пристрій.

Залежно від типу застосованих IP-блоків клас SOPC поділяють на підкласи однорідних і блокових систем на кристалі. В однорідних SOPC застосовуються так звані soft-ядра (Softcores) – наприклад, VHDL – описи блоків з потрібними функціями. У разі інтеграції таких блоків на різні кристали апаратно їх можна реалізувати по-різному, наприклад, розташовувати в різних областях кристала.

У блокових SOPC застосовують апаратні ядра, тобто спеціалізовані області кристала, виділені для певних функцій. У цих областях створюються блоки незмінної структури, спроектовані за методологією ASIC (подібно до базових матричних кристалів), оптимізовані для заданої функції. Ці блоки не передбачають можливості їх перепрограмування. Такі блоки називають hard-ядрами (Hardcores). Реалізація функцій спеціалізованими апаратними ядрами вимагає значно меншої площі кристала порівняно з реалізаціями на єдиних програмованих засобах і покращує інші характеристики схеми, передовсім, швидкодію блоків, але зменшує універсальність ПЛІС.

Характерними прикладами hard-ядер, крім процесорів і мікроконтролерів, можуть слугувати блоки для реалізації інтерфейсів різних шин (PCI, VME), схеми підтримки інтерфейсу JTAG, пристрої перемноження для систем цифрової обробки сигналів. Під час розвитку блокових SOPC характер і складність ядер змінювалися від порівняно простих до складних ядер у вигляді процесора або мікроконтролера зі значною швидкодією.

Питання і задачі для самоконтролю

1. За якими ознаками класифікують ПЛІС? Наведіть класифікацію ПЛІС за архітектурою.
2. Вкажіть особливості структурної побудови SPLD – простих програмованих логічних пристроїв. Назвіть класи SPLD та їх структурні відмінності.
3. Наведіть особливості структурної побудови CPLD – складних програмованих логічних пристроїв. Які переваги CPLD перед SPLD?
4. Сформулюйте особливості структурної побудови FPGA – програмованих користувачем вентильних матриць. Який принцип роботи конфігурованого логічного блоку?
5. Наведіть особливості структурної побудови SOPC – систем на програмованому кристалі. Які особливості проектування пристроїв на SOPC? Чим відрізняються однорідні та блокові системи на кристалі?

Список літератури

1. Цифрова техніка : навч. посіб. / Б. Є. Рицар. – К. : НМК ВО, 1991. – 372 с.
2. Угрюмов Е. Цифровая схемотехника / Е. Угрюмов. – СПб.: БХВ-Петербург, 2004.
3. Фрике К. Вводный курс цифровой электроники / К. Фрике. – М.: Техносфера, 2003.
4. Цифровые электронные вычислительные машины / К. Г. Самофалов, В. И. Корнейчук, В. П. Тарасенко. – К.: Вища шк., 1983. – 455 с.
5. Прикладная теория цифровых автоматов / К. Г. Самофалов и др. – К. : Вища шк., 1987. – 375 с.
6. Бобало Ю. Я. Основи теорії електронних кіл / Ю. Я. Бобало, Б. А. Мандзій, П. Г. Стахів, Л. Д. Писаренко, Ю. І. Якименко; за ред. проф. Ю. Я. Бобала. – Львів : Вид-во Нац. ун-ту "Львівська політехніка", 2008. – 332 с.

Книги для навчання і роботи!

Дудикевич В. Б. та ін.
ЗАХИСТ ЗАСОБІВ І КАНАЛІВ ТЕЛЕФОННОГО ЗВ'ЯЗКУ
Навчальний посібник. – 2012. – 212 с.
ISBN 978-617-607-283-6

Викладено основи функціонування телефонних мереж загального користування, проаналізовано загрози інформаційній безпеці на об'єктах із телефонним зв'язком, детально описано методи і засоби захисту від витоку інформації абонентськими телефонними лініями, закриття мовних повідомлень під час їхньої передачі каналами зв'язку, запобігання несанкціонованому використанню засобів телефонії.

Призначено для студентів напряму "Інформаційна безпека".

Рекомендувало Міністерство освіти і науки, молоді та спорту України як навчальний посібник для студентів напряму "Інформаційна безпека".

Ромака В. А. та ін.
СИСТЕМИ МЕНЕДЖМЕНТУ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ
Навчальний посібник. – 2012. – 232 с.
ISBN 978-617-607-219-5

Висвітлено основні аспекти, які пов'язані з завданнями організації та функціонування системи менеджменту інформаційної безпеки. Наведено аналіз сімейства міжнародних стандартів у сфері менеджменту інформаційної безпеки (ISO/IEC 27000) та їх історію, розглянуто етапи організації захисту інформації. Детально проаналізовано аудит систем менеджменту інформаційної безпеки, ризик-та інцидент-менеджмент, планування управління інформаційною безпекою за вимогами міжнародних стандартів.

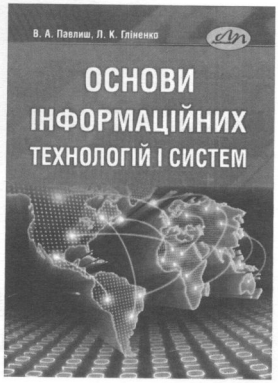
Для студентів денної та заочної форм навчання вищих навчальних закладів, які навчаються за галуззю знань "Інформаційна безпека", а також може бути корисним для аспірантів, молодих науковців та спеціалістів у галузі інформаційної безпеки.

Рекомендувало Міністерство освіти і науки, молоді та спорту України як навчальний посібник для студентів вищих навчальних закладів галузі знань "Інформаційна безпека".

Павлиш В. А., Гліненко Л. К.

ОСНОВИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

Навчальний посібник. – 2013. – 500 с.
ISBN 978-617-607-440-3



Розглянуто широке коло питань, що стосуються технологій формування і управління роботою з даними й інформацією у різних сферах діяльності. Наведено класифікацію інформаційних технологій і систем, розглянуто підходи та засоби моделювання предметної області, даних та знань під час їх створення. Викладено основи технології та систем баз даних; основи технології мультимедіа; принципи побудови та функціонування систем розосередженої обробки інформації, інформаційно-обчислювальних мереж; основи геоінформаційних технологій та систем; основи побудови та функціонування інтелектуальних інформаційних систем.

Посібник спрямований на формування у студентів загального системного уявлення про інформацію та сучасні інформаційні технології і системи. Посібник призначений для студентів вищих навчальних закладів, які вивчають інформаційні технології та системи, а також аспірантів та спеціалістів, які займаються цією проблемою.

Рекомендувало Міністерство освіти і науки України як навчальний посібник для студентів вищих навчальних закладів базового напряму 6.050902 "Радіоелектронні апарати".

Матвійків М. Д. та ін.

ЕЛЕМЕНТИ ТА КОМПОНЕНТИ ЕЛЕКТРОННИХ ПРИСТРОЇВ

Підручник. – 2015. – 496 с.
ISBN 978-617-607-794-7

Викладено основні відомості про сучасні та перспективні елементи і компоненти електронних пристроїв, зокрема наведено визначення різних видів елементів та компонентів, розглянуто їх призначення, класифікацію, умовні зображення і позначення, будову, роботу, властивості, застосування. Для студентів вищих навчальних закладів, які навчаються за напрямом "Радіоелектронні апарати", та фахівців, які проектують, виготовляють або обслуговують різноманітні електронні апарати, зокрема аудіо- та відеотехніку, електронні обчислювальні машини, мікропроцесори та персональні комп'ютери, медичні апарати, засоби зв'язку, контрольно-вимірювальні прилади, робототехніку, автоматизовані системи проектування та управління тощо.

Видавництво Львівської політехніки
вул. Ф. Колесси, 4, корп. 23А, м. Львів, 79013
тел. +380 32 2582146, факс +380 32 2582136, http://vlp.com.ua, vmr@vlp.com.ua

НАВЧАЛЬНЕ ВИДАННЯ

Максимович Володимир Миколайович
Горпенюк Андрій Ярославович
Костів Юрій Михайлович
Лужецька Наталія Миколаївна

ЦИФРОВА СХЕМОТЕХНІКА

ЕЛЕМЕНТИ ДИСКРЕТНИХ ПРИСТРОЇВ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ

Редактор *Олеся Пастущак*
Коректор *Наталія Колтун*
Технічний редактор *Лілія Саламін*
Комп'ютерне верстання *Олени Катачиної*
Художник-дизайнер *Уляна Келеман*

Здано у видавництво 12.11.2015. Підписано до друку 17.12.2015.
Формат 60×84 $^{1}/_{16}$. Папір офсетний. Друк на різографі.
Умовн. друк. арк. 7,9. Обл.-вид. арк. 5,6.
Наклад 150 прим. Зам. 150229/153505.

Видавець і виготівник: Видавництво Львівської політехніки
Свідоцтво суб'єкта видавничої справи ДК № 4459 від 27.12.2012 р.

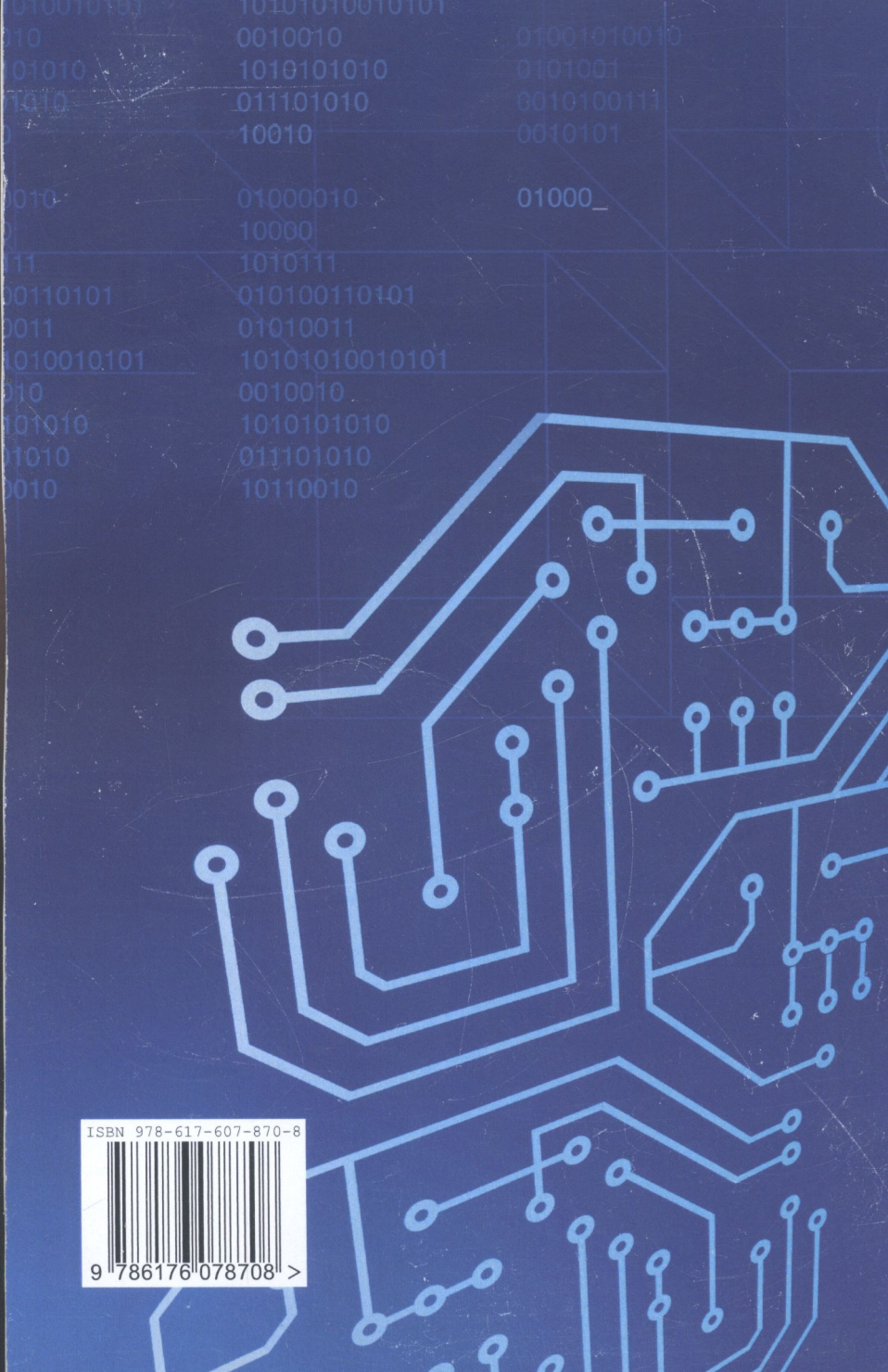
вул. Ф. Колесси, 4, Львів, 79013
тел. +380 32 2582146, факс +380 32 2582136
vlp.com.ua, ел. пошта: vmr@vlp.com.ua

Цифрова схемотехніка. Елементи дискретних пристроїв інфор-
Ц 752 маційно-комунікаційних систем : навч. посібник / В. М. Максимович, А. Я. Горпенюк, Ю. М. Костів, Н. М. Лужецька. – Львів : Видавництво Львівської політехніки, 2015. – 136 с.
ISBN 978-617-607-870-8

Розглянуто основи алгебри логіки, принципи функціонування та побудови комбінаційних та послідовнісних цифрових пристроїв, особливості сучасної елементної бази цифрової схемотехніки, застосування цифрових пристроїв для захисту інформації. Значну увагу звернено на розгляд принципів побудови генераторів псевдовипадкових чисел на елементах цифрової техніки та аналіз особливостей сучасних програмованих логічних інтегральних схем.

Для студентів денної та заочної форм навчання, які навчаються за галуззю знань "Інформаційна безпека".

УДК 004.35:316.77
ББК 22



ISBN 978-617-607-870-8