В. І. Жабін, І. А. Жуков,
І. А. Клименко, В. В. Ткаченко

ПРИКЛАДНА ТЕОРІЯ ЦИФРОВИХ АВТОМАТІВ

2-ге видання,
доопрацьоване

Рекомендовано
Міністерством освіти і науки України
як навчальний посібник для студентів
вищих навчальних закладів

Київ
Видавництво Національного авіаційного університету
«НАУ – друк»
2009

УДК 004.31(075.8)
ББК з973.3я7
П 759

Рецензенти:
І. А. Дичка — д-р техн. наук, доц.
(Національний технічний університет України «КПІ»)
М. А. Виноградов — д-р техн. наук, проф.
(Національний авіаційний університет)

Гриф надано Міністерством освіти і науки України
(Лист № 14/18.2-425 від 21.02.2006)

Прикладна теорія цифрових автоматів : навч. посіб. /
П 759 В. І. Жабін, І. А. Жуков, І. А. Клименко, В. В. Ткаченко. — 2-ге вид., доопрац. — К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», 2009. — 360 с.
ISBN 978–966–598–608–9

Розглянуто прикладні питання теорії цифрових автоматів, методи аналізу і синтезу логічних схем у сучасному елементному базисі, способи подання інформації та реалізації арифметичних операцій в ЕОМ.

Запропоновано завдання та подано рекомендації з організації курсового проектування, лабораторних занять та контролю знань в умовах кредитно-модульної системи навчання.

Для студентів напряму «Комп'ютерна інженерія». Може бути корисний для спеціалістів, які працюють у галузі проектування цифрових систем.

УДК 004.31(075.8)
ББК з973я7

ISBN 978–966–598–608–9

Навчальне видання

ЖАБІН Валерій Іванович,
ЖУКОВ Ігор Анатолійович,
КЛИМЕНКО Ірина Анатоліївна,
ТКАЧЕНКО Валентина Василівна.

ПРИКЛАДНА ТЕОРІЯ ЦИФРОВИХ АВТОМАТІВ

Навчальний посібник

Коректор *А. Голуб*
Художник обкладинки *Т. Зябліцева*
Верстка *О. Іваненко, І. Триліського*

Підп. до друку .09. Формат 60×84/16. Папір офсет.
Ум. друк. арк. 20,92.Обл.-вид. арк. 22,5.
Тираж 1000 пр. Замовлення №

Видавництво Національного авіаційного університету «НАУ-друк»
03680, Київ – 58, просп. Космонавта Комарова, 1
Свідоцтво про внесення до Державного реєстру ДК, № 977 від 05.07.2002)
Тел. (044) 406-71-33. Тел./факс: (044) 406-78-33
E-mail: publish@nau.edu.ua

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Зміст

РОЗДІЛ Б

Модуль ІІ. Комп'ютерна арифметика

Б1. Теоретична частина

Передмова

У навчальному посібнику узагальнено матеріали наукових досліджень та методичних розробок, які виконувалися авторами в процесі викладання курсу «Прикладна теорія цифрових автоматів» навчального напряму «Комп'ютерна інженерія». Головна увага приділяється відповідності матеріалу новим формам організації навчального процесу в умовах кредитно-модульної системи навчання.

Розділи навчального посібника сформовано за принципом «один розділ — один модуль». Основний матеріал курсу подано двома модулями, а саме: «Комп'ютерна логіка» і «Комп'ютерна арифметика». Лекційний матеріал, що входить до складу теоретичного ядра кожного модуля, поділяється на аудиторний і призначений для самостійного вивчення.

Крім необхідного теоретичного матеріалу, в посібнику наведено завдання та рекомендації до виконання лабораторних робіт, приклади розв'язання задач за темами лекційного матеріалу, комплекти задач для самостійного розв'язання. Поданий матеріал сприятиме раціональній самостійній підготовці студента, звільненню його від непродуктивних витрат часу на пошуки навчально-методичної літератури.

Окремим навчальним модулем є курсова робота, що виконується студентом самостійно упродовж всього семестру і призначена для розширення, закріплення, узагальнення і практичного застосування знань, вмінь і навичок, отриманих студентом під час вивчення курсу.

Автори посібника вдячні рецензентам — професору кафедри комп'ютеризованих інформаційних технологій Національного авіаційного університету, доктору технічних наук, професору М. А. Віноградову та професору кафедри спеціалізованих комп'ютерних систем Національного технічного університету «Київський політехнічний інститут», доктору технічних наук, професору І. А. Дичці за слушні зауваження.

Розділ А

Модуль І. КОМП'ЮТЕРНА ЛОГІКА

А1. ТЕОРЕТИЧНА ЧАСТИНА

А1-1. Основи теорії перемикальних функцій

А1-1.1. Подання інформації в цифровій обчислювальній техніці

Цифрові електронні обчислювальні машини (ЕОМ), або комп'ютери — це системи з програмним керуванням, призначені для автоматизації оброблення цифрової інформації. Організація ЕОМ базується на певних принципах, які складають методологічну основу цифрової обчислювальної техніки (ЦОТ). ЕОМ належить до класу складних систем, аналіз і синтез котрих базується на ієрархічному підході до питань їх організації. Існують різні рівні опису ЕОМ, серед яких найпоширенішими є віртуальний, функціональний, логічний та фізичний. Кожний з них потребує певної деталізації компонентів ЕОМ. У навчальному посібнику застосовуються функціональний та логічний рівні опису, які базуються на теорії цифрових автоматів та є найефективнішими для розгляду принципів організації та функціонування вузлів та пристроїв ЕОМ, пов'язаних з обробленням дискретної (цифрової) інформації. При цьому не розглядаються фізичні процеси, що становлять предмет цифрової електроніки.

Матеріальні об'єкти характеризуються множиною властивих їм станів. Інформація несе в собі уявлення про стан об'єктів. Процес одержання інформації має на меті зняття невизначеності, у результаті чого з деякої сукупності можливих станів виділяється стан, що реально мав місце, тобто інформація сприяє збільшенню знань про об'єкт.

Різні застосування поняття інформації приводять до різних способів оцінки кількості інформації. Класичним є ймовірнісний підхід, пов'язаний з відомою формулою Шенона. Якщо об'єкт може перебувати в альтернативних станах $1, 2, ..., n$ з ймовірностями $p_1, p_2, ..., p_n$ відповідно, то кількість інформації, що несе представлення про стан об'єкта, визначається як

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i. \qquad \text{(А1-1.1)}$$

Якщо об'єкт може перебувати в кожному з n станів з однаковими ймовірностями, то кількість інформації, що виділяє стан об'єкта, оцінюється формулою $H = \log_2 n$, що випливає з виразу (А1-1.1).

Одиниця кількості інформації називається бітом. *Біт* — це кількість інформації, за допомогою якої виділяється одне з двох альтернативних і рівноймовірних станів.

Швидкість передачі інформації визначається кількістю інформації, що передана за одиницю часу (секунду), і вимірюється в *бодах*, де 1 *бод* = 1 *біт* / *сек*.

Існують два різних підходи до реалізації процесів передачі, збереження і перетворення інформації: *безперервний* та *дискретний*.

За безперервного підходу інформація подається у вигляді системи величин, які є дійсними числами і можуть змінюватися безперервно, приймаючи множину значень з деякого діапазону. У безперервній формі інформація відображується, наприклад, в аналогових обчислювальних пристроях.

За дискретного підходу довільна величина X може приймати значення з деякого обмеженого набору $\{x_1, x_2, ..., x_n\}$. Якщо в кожен момент часу величина X приймає тільки одне із рівноймовірних значень, то говорять, що X несе в собі $\log_2 n$ біт інформації. Дискретна форма подання інформації властива цифровим обчислювальним пристроям.

Дискретна інформація відображується у вигляді позначень, за допомогою яких із сукупності можливих об'єктів виділяється той, що позначається. Для побудови позначень використовується набір символів — *алфавіт*, на основі котрого створюють послідовності символів, що визначають різні числові й логічні значення, найменування величин, дії над величинами і таке інше. Так, наприклад, для числових значень у десятковій системі числення використовуються символи 0, 1, 2, ..., 9, «+», «–», «,».

Для позначення нечислової інформації використовуються букви різних алфавітів та математичні символи. Інформація, подана у вигляді послідовності символів певного алфавіту, називається *символьною інформацією*.

Оскільки набір символів і правила запису можуть обиратися довільно, то інформація може подаватися у вигляді, відмінному від звичного загальноприйнятого. Прикладом цього є різні коди. В ЕОМ найбільше поширення знайшли двійкові коди, що складаються із символів 0 та 1.

У цифрових пристроях інформація зберігається і перетворюється з використанням фізичних елементів. Найширше застосовують фізичні елементи з двома станами, що позначаються символами 0 і 1.

З цієї причини двійкові символи складають основу *структурних одиниць інформації*.

Для ЕОМ характерними є такі структурні одиниці інформації: біт, поле, байт, слово, масив. Біту інформації відповідає двійкова змінна *x* зі значеннями 0 або 1. Послідовність бітів, що має визначений зміст, називається *полем*. Поле, яке складається з 8 бітів, називається *байтом*. Зазвичай байт відображує двійковий код одного символу. Послідовність, котра складається зі строго визначеного числа бітів (байтів) і має певний зміст, називається *словом*. Послідовність полів, байтів або слів, що мають певний зміст, утворює *масив*.

Кількість бітів, байтів або слів у структурній одиниці інформації називається *довжиною одиниці інформації*. Порядок поділу символьної інформації на структурні одиниці зображений на рис. А1-1.1.

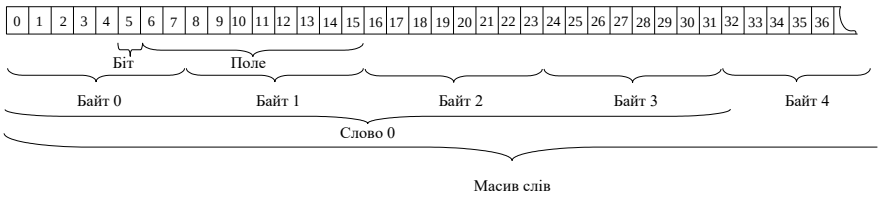


Рис. А1-1.1. Структурні одиниці інформації

У даному випадку біти пронумеровані числами, починаючи з нуля. Слово складається з 32 біт, тобто із 4 байтів.

Структурні одиниці інформації використовують як міру під час визначення кількості інформації.

Для подання бітів в ЕОМ використовуються сигнали. *Сигнал* — це характеристика процесу в певний момент часу. В ЕОМ використовують фізичні процеси, наприклад, електричний струм і магнітний потік. Фізичний процес характеризується деякою величиною, що визначає стан процесу. Так, електричний процес може характеризуватися величиною струму або напругою. Величина, яка характеризує сигнал, може змінюватися безперервно в часі, приймаючи будь-які значення в деякому діапазоні.

Процеси передачі, збереження і перетворення інформації реалізуються найпростіше, якщо розрізняти лише два рівні (дві величини) сигналу, що ототожнюються з символами 0 і 1. Сигнал, який подає одне з двох можливих значень, називається *двійковим*. Відповідність між фізичним процесом і двійковим сигналом встановлюється шляхом квантування процесу за рівнем.

Квантування за рівнем виконується в такий спосіб. Нехай сигнал характеризується напругою *U* і змінюється за часом, як показано на

рис. А1-1.2. З фізичних міркувань призначають два рівні сигналу: a і b (де $a < b$), що виділяють дві області $U \le a$ і $U \ge b$, які відповідають двом значенням логічного сигналу. В області $a < U < b$ значення логічного сигналу не визначено.

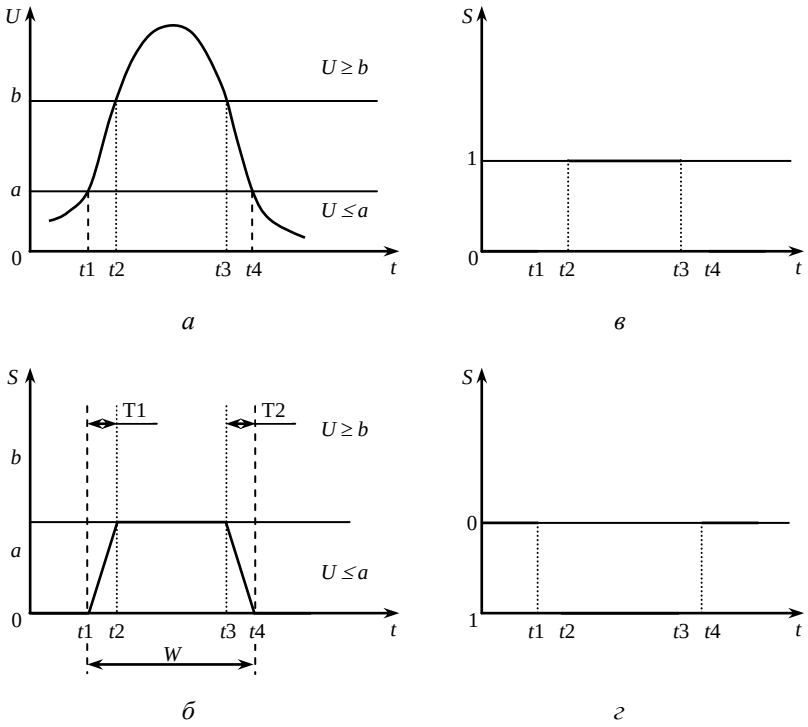


Рис. А1-1.2. Квантування безперервного сигналу:

а — фізичний сигнал, *б* — ідеалізований сигнал; *в* — кодування сигналів у системі високих потенціалів; *г* — кодування сигналів у системі низьких потенціалів

Визначення відповідності між рівнями фізичного і двійкового сигналу називається *кодуванням* сигналу. Можливі два способи кодування двійкового сигналу S: у системі високих потенціалів і в системі низьких потенціалів.

У системі високих потенціалів функція кодування має вигляд

$$S = \begin{cases} 1, \text{ якщо } U \ge b; \\ 0, \text{ якщо } U \le a. \end{cases}$$

У системі низьких потенціалів використовується функція кодування

$$S = \begin{cases} 0, \text{ якщо } U \geq b; \\ 1, \text{ якщо } U \leq a. \end{cases}$$

Двійкові сигнали, відповідні зазначеним способам кодування, наведено на рис. А1-1.2, *в* і *г*.

Проміжок T1 = t2 – t1 називається часом перемикання сигналу з 0 в 1, а проміжок T2 = t4 – t3 — часом перемикання сигналу з 1 в 0.

Для оцінки часу перемикання сигналів використовують максимальні, середні або середньоквадратичні значення. Під час проектування схем, як правило, використовують максимальні значення T_{max} = max(t2 – t1, t4 – t3).

Для правильного сприйняття інформації необхідно позбутися наявної невизначеності сигналів у моменти перемикання. З цією метою сигнали розглядають — як процеси в дискретному (автоматному) часі, який вимірюється в тактах T > T_{max}. При цьому передбачається, що сигнали переключаються тільки в межах інтервалу T. У такому випадку наприкінці такту значення сигналу (0 або 1) буде визначеним. Дискретний час вимірюється кількістю тактів і може бути застосований для відліку тривалості будь-якого процесу. Максимальна тривалість сигналів *W* може визначатися різним способом, наприклад, як показано на рис. А1-1.2.

Таким чином, процедура квантування сигналів за величиною та часом позбавляє необхідності постійних посилань на фізичну природу сигналів і дозволяє зосередити увагу на інформаційних аспектах роботи цифрових пристроїв.

А1-1.2. Перемикальні функції і логічні схеми

Функція $y = f(x_1, x_2, ..., x_n)$ називається *перемикальною*, або *логічною*, якщо сама функція *y* і кожен з її аргументів x_i, приймають значення тільки із множини {0, 1}.

Перемикальна функція може бути задана різними способами, наприклад:

— словесним описом;

— таблицею істинності;

— геометричним представленням;

— аналітичним виразом.

Перемикальну функцію, наприклад, можна описати таким чином.

Функція y від аргументів x_3, x_2, і x_1 приймає значення одиниці, якщо більшість її аргументів приймає одиничні значення.

Задану вище перемикальну функцію можна зобразити у вигляді таблиці істинності (табл. А1-1.1), де надано всі можливі двійкові набори та значення функції на цих наборах.

Набором називають упорядковану послідовність значень аргументів.

Таблиця А1-1.1

ТАБЛИЦЯ ІСТИННОСТІ

Номери наборів	Набори аргументів			Значення функції y
	x_3	x_2	x_1	
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Кожний набір має свій номер, який співпадає з кількісним еквівалентом двійкового числа, що відповідає набору. Наприклад, якщо набори впорядковуються у послідовності $x_n \ldots x_2\ x_1$, то номер набору визначається як

$$N = \sum_{i=1}^{n} x_i 2^{i-1}.$$

Таким чином, двійкові набори в таблиці істинності перемикальної функції можуть бути надані їх номерами.

Табличний спосіб подання перемикальної функції є наглядним і теоретично може бути застосований для запису функцій довільної кількості змінних.

Перемикальну функцію від n аргументів можна задати n-мірним кубом. Наприклад, для перемикальної функції трьох аргументів, заданої у табл. А1-1.1, геометричне зображення наведене на рис. А1-1.3, *в*. Кількість вершин дорівнює кількості наборів. Точкою позначені набори, на яких функція має одиничне значення. Більш детально графічний спосіб подання перемикальних функцій розглядається у розділі А1-2.3.

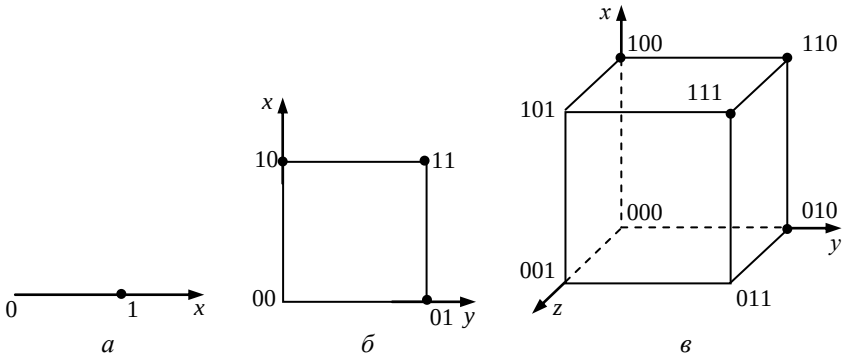


Рис. А1-1.3. Геометричне зображення перемикальних функцій:
а — однієї змінної; *б* — двох змінних; *в* — трьох змінних

Під час аналізу властивостей перемикальних функцій табличне та геометричне подання функцій не є компактними, а геометричне подання значно ускладнюється за збільшення кількості аргументів функції. Тому значно простіше виглядає аналітичний запис у вигляді формул.

Перемикальна функція може бути задана аналітичними виразами, побудованими за допомогою логічних операцій різних алгебр перемикальних функцій, наприклад:

$$y = \overline{x_3}\,\overline{x_2}x_1 \vee \overline{x_3}x_2x_1;$$

$$y = (\overline{x_3} / \overline{x_2} / x_1)/(\overline{x_3} / x_2 / x_1)/(x_3 / \overline{x_2} / x_1).$$

Одержання аналітичних форм перемикальних функцій у різних алгебрах розглядається в розділах А1-1.4 — А1-1.9.

За наявності n аргументів кількість різних наборів становить

$$N_{\text{Н}} = 2^n.$$

Врахувавши, що на кожному наборі перемикальна функція може приймати два значення (0 або 1), за правилами комбінаторики одержимо загальну кількість перемикальних функцій:

$$N_{\text{Ф}} = 2^{2^n}.$$

Таким чином, за одного аргументу існують 4 перемикальні функції, за двох аргументах — 16 функцій, за трьох — 256 функцій. Надалі кількість функцій швидко зростає.

Всі перемикальні функції одного і двох аргументів задані відповідно у табл. А1-1.2 — А1-1.3.

Таблиця А1-1.2

ФУНКЦІЇ ОДНОГО АРГУМЕНТУ

x	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Таблиця А1-1.3

ФУНКЦІЇ ДВОХ АРГУМЕНТІВ

x_2	x_1	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Серед наведених функцій є перемикальні, значення яких не залежать від значень аргументів. Такі функції називають *виродженими* перемикальними функціями (константами). У таблицях А1-1.2 і А1-1.3 виродженими є функції $f_0 = 0$, $F_0 = 0$, $f_3 = 1$, $F_{15} = 1$. В загальному випадку деякі функції залежать від $s < n$ аргументів.

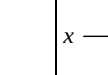
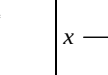
Логічний елемент — це електронна схема, що реалізує певну перемикальну функцію. На функціональних схемах логічний елемент зображується за допомогою умовного графічного позначення (УГП).

Розглянемо докладніше перемикальні функції та логічні елементи, які мають найбільше практичне значення.

Перемикальні функції одного аргументу $y = f(x)$ та УГП логічних елементів, що їх реалізують, наведені в табл. А1-1.4. Функції двох аргументів $y = f(x_2, x_1)$ та відповідні УГП логічних елементів надані у табл. А1-1.5.

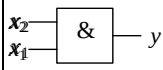
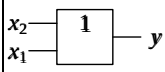
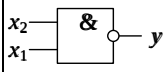
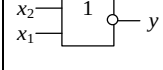
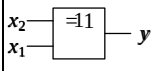
Таблиця А1-1.4

ФУНКЦІЇ ОДНОГО АРГУМЕНТУ

Таблиця істинності	Назва функції	Варіанти запису	УГП логічного елемента	Назва логічного елемента
x | y 0 | 0 1 | 1	Повторення, ТАК	$y = x$		Повторювач
x | y 0 | 1 1 | 0	Інверсія, заперечення, НЕ	$y = \bar{x}$ * $y = \neg x$		Інвертор

Таблиця А1-1.5

ФУНКЦІЇ ДВОХ АРГУМЕНТІВ

Таблиця істинності	Назва функції	Варіанти запису	УГП логічного елемента	Назва логічного елемента
x_2 x_1 y 0 0 0 0 1 0 1 0 0 1 1 1	Кон'юнкція, І, логічний добуток	$y = x_2\, x_1$ * $y = x_2 \cdot x_1$ $y = x_2 \,\&\, x_1$ $y = x_2 \wedge x_1$		Елемент І, кон'юнктор
x_2 x_1 y 0 0 0 0 1 1 1 0 1 1 1 1	Диз'юнкція, АБО, логічна сума	$y = x_2 \vee x_1$ * $y = x_2 + x_1$		Елемент АБО, диз'юнктор
x_2 x_1 y 0 0 1 0 1 1 1 0 1 1 1 0	І-НЕ, функція Шефера	$y = \overline{x_2 x_1}$ * $y = \overline{x_2 \cdot x_1}$ $y = \overline{x_2 \,\&\, x_1}$ $y = \overline{x_2 \wedge x_1}$ $y = x_2 / x_1$		Елемент І-НЕ, елемент Шефера
x_2 x_1 y 0 0 1 0 1 0 1 0 0 1 1 0	АБО-НЕ, функція Пірса (Веба)	$y = \overline{x_2 \vee x_1}$ * $y = \overline{x_2 + x_1}$ $y = x_2 \downarrow x_1$		Елемент АБО-НЕ, елемент Пірса
x_2 x_1 y 0 0 0 0 1 1 1 0 1 1 1 0	ВИКЛЮЧНЕ АБО, сума по модулю 2, нерівнозначність	$y = x_2 \oplus x_1$ * $y = x_2 + x_1 (\mathrm{mod}\, 2)$ $y = x_2 \neq x_1$		Елемент ВИКЛЮЧНЕ АБО, суматор по модулю 2

* — аналітичний запис, що використовується найчастіше

Розглянуті функції від двох аргументів існують і за довільною кількості аргументів. При цьому зберігаються форми запису функцій та УГП.

Наприклад, диз'юнкція чотирьох аргументів має такий вигляд

$$F(x_4, x_3, x_2, x_1) = x_4 \vee x_3 \vee x_2 \vee x_1 .$$

Логічний елемент, що реалізує наведену функцію, має чотири входи, УГП цього елементу зображене на рис. А1-1.4.

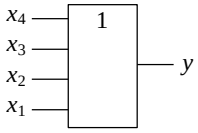


Рис. А1-1.4. УГП логічного елементу АБО
з чотирма входами

Як правило, на практиці використовують порівняно невеликий набір перемикальних функцій для побудови інших, більш складних функцій за допомогою методу суперпозиції функцій.

Під *суперпозицією* функцій розуміють підстановку замість аргументів однієї функції значень інших функцій.

Наприклад, якщо дано дві функції від двох аргументів $f(a,b)$ і $f_2(a,\ b),$ то можна одержати функцію від трьох аргументів $f_3(a,b,c)$ таким чином: $f_3 = f_1(c,\ f_2(a,\ b))$.

Сукупність взаємозв'язаних логічних елементів називають *логічною схемою*. Суперпозиції функцій відповідає певна логічна схема.

Існують два різновиди логічних схем:

— комбінаційні;

— послідовнісні.

Під *комбінаційною схемою* розуміють таку схему, функціонування якої може бути описане системою перемикальних функцій:

$$\begin{cases} y_1 = f(x_1, x_2, ..., x_n); \\ y_2 = f(x_1, x_2, ..., x_n); \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y_m = f(x_1, x_2, ..., x_n), \end{cases}$$

де n — кількість входів;

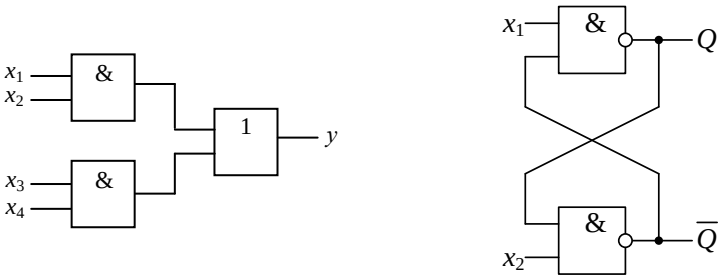
m — кількість виходів схеми.

Комбінаційна схема має тільки один стан. У зв'язку з цим вихідні сигнали комбінаційної схеми залежать тільки від значень вхідних логічних сигналів. Основною ознакою комбінаційних схем є відсутність петель.

Петля — це шлях від виходу логічного елемента до його входу, можливо, через інші логічні елементи.

Послідовнісні схеми мають більш ніж один стан. У зв'язку з чим вихідні сигнали схеми залежать не тільки від вхідних сигналів, а й від стану, в якому перебуває схема. Основною ознакою послідовнісних схем є наявність петель.

Приклади логічних схем показані на рис. А1-1.5, *а* і *б*.



а *б*

Рис. А1-1.5. Логічні схеми:
а — комбінаційна; *б* — послідовнісна

Як комбінаційні, так і послідовнісні логічні схеми називають *цифровими автоматами.* При цьому комбінаційні схеми мають назву *тривіальних автоматів*, або автоматів, що не мають пам'яті, а послідовнісні схеми називають *автоматами з пам'яттю.*

До параметрів, якими найчастіше характеризують логічні схеми належать складність схеми (структурний параметр) та затримка сигналів схемою (часовий параметр).

Існує декілька способів оцінки складності логічних схем. Найбільш універсальною є оцінка складності логічної схеми за Квайном, яка дорівнює

$$K = \sum_{i=1}^{s} n_i,$$

де s — кількість елементів;

n_i — кількість входів i-го елементу.

Таким чином, для визначення складності K необхідно знайти сумарну кількість входів усіх логічних елементів схеми.

Наприклад, для логічної схеми, зображеної на рис. А1-1.5, а складність за Квайном дорівнює $K = 6$, а для схеми на рис. А1-1.5, б — $K = 4$.

Складність логічної схеми можна також оцінити числом умовних логічних елементів, що визначається за формулою

$$N = \sum_{i=1}^{r} \frac{m_i \cdot n_i}{g},$$

де R — кількість типів елементів;
m_i — кількість елементів i-го типу;
n_i — кількість входів елементів i-го типу;
G — число входів умовного елементу.

Параметри складності K і N використовуються під час проектування інтегральних схем (як порівняна оцінка складності варіантів проектування).

Швидкодія логічних схем залежить від затримки сигналів логічними елементами, яка визначаються часовими параметрами логічних елементів t_{01} (перемикання з 0 в 1) і t_{10} (перемикання з 1 в 0).

Під *затримкою* сигналів елементом розуміють час переходу вихідного сигналу елементу від одного логічного рівня до іншого від моменту зміни значення вхідних сигналів, що викликають цей перехід. На практиці використовують усереднене значення затримки сигналів $t = (t_{01} + t_{10})/2$ або максимальне $t = \max(t_{01}, t_{10})$.

Для комбінаційних схем, побудованих на однотипних елементах, середній час затримки сигналів дорівнює

$$T = Lt,$$

де L — кількість логічних елементів, що входять в максимальний за довжиною ланцюжок елементів логічної схеми;
t — затримка сигналу логічним елементом.

Якщо використовуються елементи з різною затримкою, то в схемі визначається шлях, який вимагає максимального часу поширення сигналів від входів до виходу.

Мінімальний період зміни вхідних сигналів визначається з урахуванням максимальної затримки сигналів у схемі. Він не може бути менший максимальної затримки сигналів. Виходячи з цього визначається і максимальна частота зміни наборів вхідних сигналів.

А1-1.3. Алгебри перемикальних функцій

Аналітичний спосіб подання перемикальних функцій посідає особливе місце в теорії перемикальних функцій. Фактично всі перетворення над перемикальними функціями, необхідні для синтезу та аналізу цифрових автоматів, здійснюються на аналітичному рівні.

На практиці для побудови логічних схем може бути застосований певний функціональний елементній базис.

Функціональний елементний базис — це сукупність типів функціональних елементів, які можна використовувати для побудови схем.

Вміння представляти перемикальні функції у заданому елементному базисі має важливе практичне значення. Одна і та сама функція може бути представлена суперпозицією різних перемикальних функцій, а кожній суперпозиції, у свою чергу, відповідає комбінаційна схема з певною складністю.

Відповідні перетворення форм перемикальних функцій із одного елементного базису на інший виконують на основі співвідношень алгебр перемикальних функцій.

Під *алгеброю* розуміють сукупність змінних, на яких визначена система кінцевомісних функцій (операцій).

Кількість аргументів функцій називають її містністю.

Найбільший практичний інтерес для синтезу та аналізу схем мають алгебри Буля, Пірса, Шефера і Жегалкіна.

А1-1.4. Алгебра Буля

Алгебра визначена на $n \geq 2$ змінних. Для перетворення аргументів в алгебрі Буля використовуються функції І, АБО та НЕ. Система перемикальних функцій відповідно має вигляд

$$\begin{cases} f_1 = x_1 \cdot x_2 \cdot \ldots \cdot x_n; \\ f_2 = x_1 \vee x_2 \vee \ldots \vee x_n; \\ f_3 = \overline{x}_i \qquad (i = \overline{1, n}). \end{cases}$$

Аксіоми алгебри Буля:

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

Основні закони (властивості) алгебри Буля.

1. Закон комутативності: $x_2 x_1 = x_1 x_2$;
   $x_2 \vee x_1 = x_1 \vee x_2$.
2. Закон асоціативності: $x_3 x_2 x_1 = (x_3 x_2) x_1$;
   $x_3 \vee x_2 \vee x_1 = (x_3 \vee x_2) \vee x_1$.
3. Закон дистрибутивності: $x_3 (x_2 \vee x_1) = x_3 x_2 \vee x_3 x_1$;
   $x_3 \vee x_2 x_1 = (x_3 \vee x_2)(x_3 \vee x_1)$.

З практичного погляду закон комутативності означає, що всі входи елементів І та АБО мають однакові функціональні властивості. Різні вхідні сигнали можна подавати на будь-які входи елементів.

Властивість асоціативності дозволяє за рахунок декомпозиції (каскадування) елементів реалізувати функції І та АБО від будь-якої кількості аргументів на основі елементів з меншою кількістю входів.

Закон дистрибутивності визначає правило винесення за дужки перемінної. На практиці завдяки цьому у деяких випадках можна зменшувати складність логічних схем.

Зв'язок між функціями І, АБО, І-НЕ, АБО-НЕ встановлюється за допомогою *правил де Моргана*:

$$\begin{aligned} \overline{x \vee y} &= \bar{x} \cdot \bar{y}; \\ \overline{x \cdot y} &= \bar{x} \vee \bar{y}; \end{aligned} \qquad \text{(А1-1.2)}$$

або

$$\begin{aligned} x \vee y &= \overline{\bar{x} \cdot \bar{y}}; \\ x \cdot y &= \overline{\bar{x} \vee \bar{y}}. \end{aligned}$$

З наведених співвідношень видно, що функції І та АБО мають однакові властивості. Такі функції називають *дуальними*.

Для доведення логічних співвідношень використовують два способи: порівняння таблиць істинності лівої та правої частини співвідношення, а також застосування аксіом та теорем для приведення лівої та правої частини співвідношення до одного вигляду.

Приклад

Завдання. Довести справедливість властивості дистрибутивності в алгебрі Буля способом порівняння таблиць істинності лівої та правої частини співвідношення

$$x_3(x_2 \vee x_1) = x_3x_2 \vee x_3x_1.$$

Виконання завдання

Побудуємо таблицю, в якій поетапно визначимо значення всіх членів лівої та правої частини співвідношення дистрибутивності для всіх наборів аргументів (табл. А1-1.6). У лівій частині таблиці наведено всі можливі набори аргументів. Для зручності позначимо стовпці таблиці цифрами. Спочатку заповнюємо стовпці 2, 4 і 5, а далі — стовпці 3 і 6.

У стовпцях 3 і 6, що відповідають лівій і правій частині співвідношення дистрибутивності, значення співпадають для всіх наборів аргументів, що і треба було довести.

Таблиця А1-1.6

ДОВЕДЕННЯ ВЛАСТИВОСТІ ДИСТРИБУТИВНОСТІ В АЛГЕБРІ БУЛЯ

1			2	3	4	5	6
x_3	x_2	x_1	$x_2 \vee x_1$	$x_3(x_2 \vee x_1)$	x_3x_2	x_3x_1	$x_3x_2 \vee x_3x_1$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Приклад

Завдання. Довести справедливість співвідношення властивості дистрибутивності в алгебрі Буля способом застосування аксіом та теорем:

$$x_3 \vee x_2x_1 = (x_3 \vee x_2)(x_3 \vee x_1).$$

Виконання завдання

Перетворимо праву частину співвідношення:

$$(x_3 \vee x_2)(x_3 \vee x_1) = x_3x_3 \vee x_3x_1 \vee x_3x_2 \vee x_2x_1 =$$
$$= (x_3 \vee x_3x_1 \vee x_3x_2) \vee x_2x_1 = x_3(1 \vee x_1 \vee x_2) \vee x_2x_1 = x_3 \vee x_2x_1.$$

Вираз, отриманий у результаті перетворень, дорівнює лівій частині заданого співвідношення. Таким чином, справедливість співвідношень закону дистрибутивності доведено.

А1-1.5. Канонічні форми алгебри Буля

Оскільки в алгебрі Буля функції І та АБО є дуальними, існують дві канонічні форми булевих функцій:

— досконала диз'юнктивна нормальна форма (ДДНФ);

— досконала кон'юнктивна нормальна форма (ДКНФ).

Для визначення ДДНФ наведемо такі означення.

Будь-яка послідовність аргументів, об'єднаних однією операцією, називається *термом*.

Змінна із запереченням або без заперечення в термі називається *буквою*.

Кількість букв в термі називається *рангом* терма.

Функція, що приймає одиничне значення тільки на одному наборі аргументів, називається *конституентою одиниці*.

Кількість конституент одиниці дорівнює кількості наборів, тобто 2^n, де n — кількість аргументів.

Для перемикальних функцій від трьох аргументів $f = (x_3, x_2, x_1)$ конституенти одиниці наведені в табл. А1-1.7.

Таблиця А1-1.7

КОНСТИТУЕНТИ ОДИНИЦІ

x_3	x_2	x_1	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Аналітичний запис конституенти одиниці є *кон'юнктивним термом* n-го рангу. Відповідно до табл. А1-1.7 надамо аналітичний запис конституент одиниці:

$$C_0 = \overline{x}_3\overline{x}_2\overline{x}_1,\ \ C_1 = \overline{x}_3\overline{x}_2 x_1,\ C_2 = \overline{x}_3 x_2\overline{x}_1,\ \ C_3 = \overline{x}_3 x_2 x_1,$$
$$C_4 = x_3\overline{x}_2\overline{x}_1,\ \ C_5 = x_3\overline{x}_2 x_1,\ C_6 = x_3 x_2\overline{x}_1,\ \ C_7 = x_3 x_2 x_1.$$

Досконала диз'юнктивна нормальна форма (ДДНФ) — це диз'юнкція конституент одиниці, що відповідають наборам, на яких функція приймає одиничні значення, тобто

$$F_{\text{ДДНФ}} = \bigvee_{i=0}^{2^n-1} C_i \cdot \alpha_i, \qquad \text{(А1-1.3)}$$

де $\alpha_i \in \{0, 1\}$ — значення функції на i-му наборі.

Приклад

Завдання. Знайти ДДНФ для перемикальних функцій, заданих таблицею істинності (табл. А1-1.8).

Таблиця А1-1.8

ТАБЛИЦЯ ІСТИННОСТІ

x_3	x_2	x_1	y_1	y_2
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

Виконання завдання

Відповідно до виразу А1-1.3 одержимо ДДНФ функцій y_1 та y_2:

$$y_{1\text{ДДНФ}} = \overline{x_3}\,\overline{x_2}x_1 \vee \overline{x_3}x_2x_1 \vee x_3\overline{x_2}x_1 \vee x_3x_2\overline{x_1};$$

$$y_{2\text{ДДНФ}} = \overline{x_3}\,\overline{x_2}\,\overline{x_1} \vee \overline{x_3}\,\overline{x_2}x_1 \vee \overline{x_3}x_2\overline{x_1} \vee x_3\overline{x_2}\,\overline{x_1} \vee x_3x_2x_1.$$

Зауваження. Часто для спрощення запису функцій замість повного перечислення термів застосовують номери наборів, де функція приймає одиничне значення. Така форма запису називається *цифровою*.

Отримані ДДНФ можуть бути подані у цифровому вигляді, де конституента одиниці задається номером набору. Одержані аналітичні записи функцій y_1 та y_2 можна записати як

$$y_{1\text{ДДНФ}} = 1 \vee 3 \vee 5 \vee 6;$$

$$y_{2\text{ДДНФ}} = 0 \vee 1 \vee 2 \vee 6 \vee 7.$$

Базовим поняттям для побудови ДКНФ є конституента нуля.

Конституента нуля — це функція, що приймає нульове значення тільки на одному наборі.

Конституента нуля може бути записана у вигляді диз'юнктивного терму *n*-го рангу. Нульовому значенню аргументу в термі відповідає буква без заперечення, а одиничному — із запереченням.

Кількість конституент нуля, як і кількість конституент одиниці, дорівнює 2^n. Наприклад, для перемикальних функцій трьох аргументів x_3, x_2, x_1 конституенти нуля можуть мати вигляд:

$$S_0 = x_3 \vee x_2 \vee x_1,\ S_1 = x_3 \vee x_2 \vee \overline{x_1},\ S_2 = x_3 \vee \overline{x_2} \vee x_1,\ S_3 = x_3 \vee \overline{x_2} \vee \overline{x_1},$$

$$S_4 = \overline{x_3} \vee x_2 \vee x_1,\ S_5 = \overline{x_3} \vee x_2 \vee \overline{x_1},\ S_6 = \overline{x_3} \vee \overline{x_2} \vee x_1,\ S_7 = \overline{x_3} \vee \overline{x_2} \vee \overline{x_1}.$$

Досконала кон'юнктивна нормальна форма (ДКНФ) — це кон'юнкція конституент нуля, що відповідають наборам, на яких функція приймає значення нуля, тобто

$$F_{\text{ДКНФ}} = \mathop{\&}_{i=0}^{2^n-1} (S_i \vee \alpha_i), \qquad \text{(А1-1.4)}$$

де $\alpha_i \in \{0, 1\}$ — значення функції на *i*-му наборі.

Приклад

Завдання. Знайти ДКНФ перемикальних функції y_1 та y_2, заданих таблицею істинності (табл. А1-1.8).

Виконання завдання

Виходячи із виразу А1-1.4 одержимо ДКНФ функцій:

$$y_{1\text{ДКНФ}} = (x_3 \vee x_2 \vee x_1)(x_3 \vee \overline{x_2} \vee x_1)(\overline{x_3} \vee x_2 \vee x_1)(\overline{x_3} \vee \overline{x_2} \vee \overline{x_1});$$

$$y_{2\text{ДКНФ}} = (x_3 \vee \overline{x_2} \vee \overline{x_1})(\overline{x_3} \vee x_2 \vee x_1)(\overline{x_3} \vee x_2 \vee \overline{x_1}).$$

У цифровому вигляді одержані перемикальні функції можна записати так:

$$y_{1\text{ДКНФ}} = \overline{0} \cdot \overline{2} \cdot \overline{4} \cdot \overline{7};$$

$$y_{2\text{ДКНФ}} = \overline{3} \cdot \overline{4} \cdot \overline{5}.$$

Сумарна кількість букв у аналітичному запису перемикальної функції називається *ціною* форми. У загальному випадку ціна ДДНФ і ДКНФ може відрізнятися.

Якщо немає обмежень на кількість входів логічних елементів, то комбінаційні схеми, які реалізують нормальні форми будь-якої алгебри, завжди мають два рівні. Вхідний сигнал у таких схемах проходить до виходу через ланцюжок, що має два елементи.

Приклад

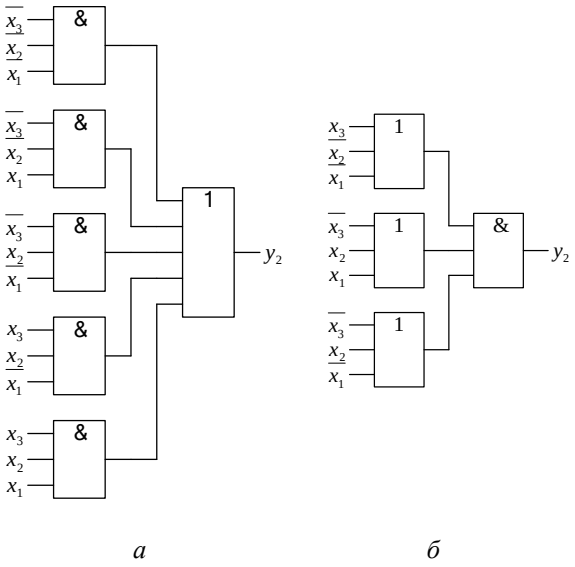
Завдання. Побудувати комбінаційну схему, що реалізує перемикальну функцію y_2 у ДДНФ і ДКНФ (табл. А1-1.8).

Виконання завдання

Комбінаційна схема, зображена на рис. А1-1.6, *а* реалізує ДДНФ перемикальної функції, а комбінаційна схема на рис. А1-1.6, *б* — ДКНФ перемикальної функції y_2.

Отримані комбінаційні схеми мають два логічних рівні. Складність за Квайном комбінаційних схем відрізняється. Складність комбінаційної схеми, що реалізує ДДНФ функції дорівнює $K = 20$ (рис. А1-1.6, *а*), а ДКНФ функції — $K = 12$ (рис. А1-1.6, *б*).

У кожній комбінаційній схемі сигнал розповсюджується через однакові ланцюжки, які містять один елемент І та один елемент АБО. Отже затримка сигналів дорівнює $T = t_{\text{І}} + t_{\text{АБО}}$.



а *б*

Рис. А1-1.6. Комбінаційні схеми:
а — для ДДНФ; *б* — для ДКНФ

А1-1.6. Алгебра Шефера

Алгебра Шефера визначена на $n \geq 2$ елементах і містить тільки одну функцію І-НЕ (функцію Шефера), яку в n-містному варіанті можна записати у вигляді

$$f = x_1 / x_2 / ... / x_n \quad = \overline{x_1 x_2 \cdots x_n}\ .$$

Аксіоми алгебри Шефера:

$$x/0 = \overline{x \cdot 0} = 1; \qquad x/x = \overline{x \cdot x} = \bar{x};$$
$$x/1 = \overline{x \cdot 1} = \bar{x}; \qquad x/\bar{x} = \overline{x \cdot \bar{x}} = 1.$$

У алгебрі Шефера виконується тільки закон (властивість) комутативності

$$x_2 / x_1 = x_1 / x_2.$$

Унаслідок того, що закон асоціативності не виконується, тобто

$$x_3 / x_2 / x_1 \neq (x_3 / x_2) / x_1,$$
$$\overline{x_3 x_2 x_1} \neq \overline{\overline{(x_3 x_2)} x_1},$$

ускладнюється порівняно із функцією І каскадування елементів для одержання функції І-НЕ з більшою кількістю аргументів на базі логічних елементів І-НЕ із кількістю входів, меншою необхідної.

Наприклад, якщо елементи мають два входи, а необхідно одержати функцію трьох аргументів, то еквівалентну форму в алгебрі Шефера можна одержати так:

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

Таким чином, схема, що реалізує праву частину співвідношення, повинна мати три рівні.

Канонічна нормальна форма в алгебрі Шефера містить терми Шефера *n*-го рангу й забезпечує дворівневу схему. Перемінні в термах і самі терми об'єднуються операцією І-НЕ.

Для одержання канонічної форми виписують терми Шефера для наборів, на яких функція приймає одиничні значення. Наприклад, якщо перемикальна функція на наборах з номерами 0, 1 і 7 приймає одиничні значення, то канонічна нормальна форма в алгебрі Шефера складатиметься з термів Шефера для відповідних наборів функції і матиме такий вигляд:

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

Враховуючи, що функція НЕ не міститься у складі функцій алгебри Шефера, заперечення над буквами означає, що перемінні можуть поступати на входи логічних схем із запереченнями. Якщо це неможливо, то у формі виконується заміна згідно з аксіомою $\bar{o} = x/x$. В цьому випадку одержана нормальна форма матиме вигляд

$$y = ((x_3 / x_3)/(x_2 / x_2)/(x_1 / x_1))/((x_3 / x_3)/(x_2 / x_2)/x_1)/(x_3 / x_2 / x_1).$$

Канонічна нормальна форма в алгебрі Шефера може також бути одержана на основі ДДНФ із застосуванням правила де Моргана (А1-1.2).

Для функції, що розглядається, можна записати

$$y = (\overline{x_3 x_2 x_1}) \vee (\overline{x_3 x_2} x_1) \vee (x_3 x_2 x_1) =$$

$$= \overline{(\overline{x_3 x_2 x_1}) \vee (\overline{x_3 x_2} x_1) \vee (x_3 x_2 x_1)} =$$

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

Приклад

Завдання. Одержати канонічну нормальну форму алгебри Шефера для функції, що задана у ДДНФ:

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

Побудувати комбінаційну схему реалізації функції на елементах І-НЕ, вважаючи, що на входи схем можуть подаватися аргументи і їх заперечення.

Виконання завдання

Із застосуванням правила де Моргана (див. розділ А1-1.4) отримаємо:

$$y_1 = (\overline{x_3} \cdot \overline{x_2} \cdot x_1) \vee (\overline{x_3} \cdot x_2 \cdot x_1) \vee (x_3 \cdot \overline{x_2} \cdot x_1) \vee (x_3 \cdot x_2 \cdot \overline{x_1}) =$$

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

$$= \overline{(\overline{x_3} \cdot \overline{x_2} \cdot x_1)} \cdot \overline{(\overline{x_3} \cdot x_2 \cdot x_1)} \cdot \overline{(x_3 \cdot \overline{x_2} \cdot x_1)} \cdot \overline{(x_3 \cdot x_2 \cdot \overline{x_1})} =$$

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

Комбінаційна схема, що реалізує задану перемикальну функ-

цію, зображена на рис. А1-1.7.

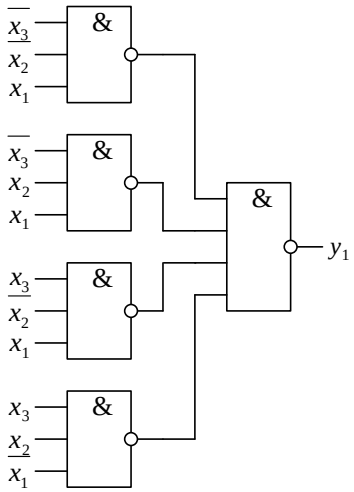


Рис. А1-1.7. Схема функції y_1 в елементному базисі алгебри Шефера

А1-1.7. Алгебра Пірса

Алгебра Пірса визначена на $n \geq 2$ елементах і містить тільки одну функцію АБО-НЕ (функцію Пірса або Веба), яку в n-містному варіанті можна записати у вигляді

$$y = x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n = \overline{x_1 \vee x_2 \vee \ldots \vee x_n}.$$

Основні аксіоми алгебри Пірса:

$$x \downarrow 0 = \overline{x \vee 0} = \bar{x}; \qquad x \downarrow x = \overline{x \vee x} = \bar{x};$$
$$x \downarrow 1 = \overline{x \vee 1} = 0; \qquad x \downarrow \bar{x} = \overline{x \vee \bar{x}} = 0.$$

Властивості алгебри Пірса і Шефера однакові. В алгебрі Пірса, як і в алгебрі Шефера, виконується тільки закон (властивість) комутативності

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

Канонічна нормальна форма в алгебрі Пірса забезпечує дворівневу комбінаційну схему і містить терми Пірса. Змінні в термах і самі терми об'єднуються операцією АБО-НЕ. Для одержання канонічної

нормальної форми виписують терми Пірса для наборів, на яких функція приймає нульові значення.

Наприклад, якщо перемикальна функція на наборах з номерами 0, 1 і 7 приймає нульові значення, то канонічна нормальна форма в алгебрі Пірса складатиметься з термів Пірса для відповідних наборів функції і матиме такий вигляд:

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

Якщо на входи комбінаційної схеми не можуть подаватися змінні із запереченням, то необхідно виконати заміну $\overline{x} = x \downarrow x$.

Канонічну форму можна також одержати із ДКНФ за допомогою правила де Моргана (А1-1.2).

Приклад

Завдання. Одержати канонічну нормальну форму алгебри Пірса для функції, що задана у ДКНФ:

$$y_2 = (x_3 \vee \overline{x_2} \vee \overline{x_1}) \cdot (\overline{x_3} \vee x_2 \vee x_1) \cdot (\overline{x_3} \vee x_2 \vee \overline{x_1}).$$

Побудувати комбінаційну схему реалізації функції на елементах АБО-НЕ з будь-якою кількістю входів, вважаючи, що на входи схем можуть подаватися тільки прямі значення аргументів.

Виконання завдання

Із застосуванням правила де Моргана (див. розділ А1-1.4) та аксіоми $\overline{o} = x \uparrow x$, отримаємо:

$$y_2 = (x_3 \vee \overline{x_2} \vee \overline{x_1}) \cdot (\overline{x_3} \vee x_2 \vee x_1) \cdot (\overline{x_3} \vee x_2 \vee \overline{x_1}) =$$

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

$$= \overline{\overline{(x_3 \vee \overline{x_2} \vee \overline{x_1})} \vee \overline{(\overline{x_3} \vee x_2 \vee x_1)} \vee \overline{(\overline{x_3} \vee x_2 \vee \overline{x_1})}} =$$

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

$$(x_3 \downarrow (x_2 \downarrow x_2) \downarrow (x_1 \downarrow x_1)) \downarrow ((x_3 \downarrow x_3) \downarrow x_2 \downarrow x_1) \downarrow$$

$$\downarrow ((x_3 \downarrow x_3) \downarrow x_2 \downarrow (x_1 \downarrow x_1)).$$

Комбінаційна схема, що реалізує задану функцію, зображена на рис. А1-1.8.

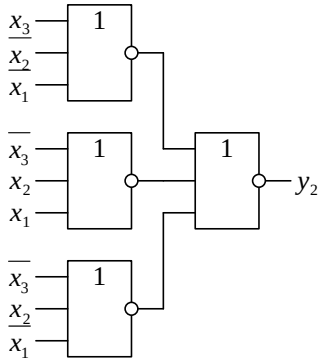


Рис. А1-1.8. Схема функції y_2 на елементах АБО-НЕ

А1-1.8. Алгебра Жегалкіна

Система функцій алгебри Жегалкіна містить двомісні функції І та ВИКЛЮЧНЕ АБО (сума по модулю 2), а також константу 1:

$$\begin{cases} f_1 = x_1 \cdot x_2; \\ f_2 = x_1 \oplus x_2\ ; \\ f_2 = 1. \end{cases}$$

Місткість функцій І та АБО може бути збільшена, але за непарного числа аргументів вводиться додатково константа 0. Це пояснюється необхідністю мати в системі функцію, що не зберігає 1, для забезпечення функціональної повноти системи функцій цієї алгебри (див. розділ А.1-1.10).

Аксіоми алгебри Жегалкіна:

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

Основні закони (властивості) алгебри Жегалкіна:

1. Властивість комутативності

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

2. Властивість асоціативності

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

3. Властивість дистрибутивності (виконується тільки в одному варіанті)

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

Канонічною нормальною формою алгебри Жегалкіна є поліном Жегалкіна, який можна одержати у такий спосіб.

1. Записати задану перемикальну функцію в ДДНФ.

2. Замінити знак операції АБО між термами на ВИКЛЮЧНЕ АБО ($\vee \to \oplus$).

Зауваження. Така заміна правомірна, якщо функція представлена у ДДНФ. Це пояснюється тим, що ДДНФ складається тільки із конституент одиниці. Отже, на будь-якому наборі аргументів тільки одна конституента приймає одиничне значення, а всі інші — нульові. Таким чином виконується рівність такого вигляду:

$$1 \vee 0 \vee \ldots \vee 0 = 1 \oplus 0 \oplus \ldots \oplus 0,$$

де положення одиниці у запису може бути довільним.

3. Кожний аргумент із запереченням замінити на суму по модулю два цього аргумента з одиницею згідно з аксіомою $\bar{x} = x_i \oplus 1$.

4. Розкрити дужки й спростити одержаний вираз шляхом викреслювання парних термів згідно з аксіомами $x \oplus x = 0$, $x \oplus 0 = x$.

Приклад

Завдання. Одержати канонічні нормальні форми в алгебрі Жегалкіна функцій y_1 і y_2, задані таблицею істинності (табл. А1-1.9).

Таблиця А1-1.9

ТАБЛИЦЯ ІСТИННОСТІ

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

Виконання завдання

Зауважимо, що функції y_1 і y_2 є відповідно функціями І та ВИКЛЮЧНЕ АБО, що очевидно із табл. А1-1.9. Представимо вказані функції у ДДНФ і виконаємо послідовно етапи перетворення форм перемикальних функцій.

$$y_1 = \overline{x_2}x_1 \vee x_2\overline{x_1} \vee x_2x_1 = \overline{x_2}x_1 \oplus x_2\overline{x_1} \oplus x_2x_1 =$$
$$= (x_2 \oplus 1)x_1 \oplus x_2(x_1 \oplus 1) \oplus x_2x_1 =$$
$$= x_2x_1 \oplus x_1 \oplus x_2x_1 \oplus x_2 \oplus x_2x_1 = x_2x_1 \oplus x_1 \oplus x_2;$$
$$y_2 = \overline{x_2}x_1 \vee x_2\overline{x_1} = \overline{x_2}x_1 \oplus x_2\overline{x_1} =$$
$$= (x_2 \oplus 1)x_1 \oplus x_2(x_1 \oplus 1) = x_2x_1 \oplus x_1 \oplus x_2x_1 \oplus x_2 = x_1 \oplus x_2.$$

З урахуванням одержаних у прикладі результатів і розглянутих аксіом можна визначити зв'язок між алгебрами Буля та Жегалкіна:

$$x_1 \vee x_2 = x_1x_2 \oplus x_1 \oplus x_2;$$
$$x_1 \oplus x_2 = x_1\overline{x_2} \vee \overline{x_1}x_2;$$
$$\overline{x} = x \oplus 1.$$

Відповідно до вигляду полінома Жегалкіна визначається лінійність перемикальної функції.

Функція називається *лінійною*, якщо її поліном містить кон'юнктивні терми тільки першого рангу. В іншому разі функція вважається *нелінійною*.

У розглянутому прикладі функція y_1 є нелінійною (її поліном містить терм другого рангу x_2x_1), а функція y_2 — лінійною.

А1-1.9. Перетворення нормальних форм перемикальних функцій

Одна і та сама перемикальна функція може задаватися різними аналітичними формами. Необхідність перетворення аналітичних форм функцій може бути обумовлена зміною елементного базису, пошуком форми з мінімальною ціною тощо.

На практиці під час побудови логічних схем можуть використовуватися логічні елементи, що реалізують функції різних алгебр. Найчастіше елементний базис, обумовлений мікроелектронною тех-

нологією, містить елементи з множини {І, АБО, НЕ, І-НЕ, АБО-НЕ}, тобто складається з елементів алгебр Буля, Шеффера і Пірса. Таку систему функцій можна розглядати в аспекті *розширеної практичної алгебри*, яка має сукупні властивості відповідних алгебр.

Розширена алгебра має *вісім нормальних форм* представлення перемикальних функцій, котрі забезпечують побудову дворівневих комбінаційних схем, якщо *на кількість входів логічних елементів немає обмежень.*

Чотири нормальні форми поширеної алгебри можна одержати виходячи із ДДНФ і ще чотири — виходячи із ДКНФ перемикальної функції або ДДНФ заперечення функції. Якщо немає обмежень на кількість входів логічних елементів, то такі форми перемикальних функцій забезпечують побудову дворівневих комбінаційних схем.

Нормальним формам зручно надати назву, що складається із назв внутрішньої та зовнішньої операції. Наприклад, ДДНФ може отримати назву форми І / АБО, ДКНФ — АБО / І і таке інше.

Розглянемо нормальні форми поширеної алгебри перемикальної функції двох аргументів $y=(x_2,\ x_1)$, заданої таблицею істинності (табл. А1-1.10).

Таблиця А1-1.10

ТАБЛИЦЯ ІСТИННОСТІ

x_2	x_1	Y
0	0	1
0	1	0
1	0	0
1	1	1

У формі ДДНФ і ДКНФ задана перемикальна функція має відповідно вигляд:

$$y_{\text{ДДНФ}} = \overline{x_2}\,\overline{x_1} \vee x_2 x_1,$$

$$y_{\text{ДКНФ}} = (x_2 \vee \overline{x_1})(\overline{x_2} \vee x_1).$$

Виходячи із ДДНФ з урахуванням аксіоми $\overline{\overline{x}} = x$ (див. розділ А1-1.4) та правила де Моргана (А1-1.2) одержимо перші чотири нормальні форми:

$$y = \overline{x_2}\,\overline{x_1} \vee x_2 x_1 = \qquad \text{(І / АБО)}$$

$$= \overline{\overline{\overline{x_2}\,\overline{x_1} \vee x_{21} x_1}} =$$

$$= \overline{\overline{x_2}\,\overline{x_1}} \cdot \overline{x_2 x_1} = \qquad \text{(І-НЕ / І-НЕ)}$$

$$= \overline{(x_2 \vee x_1)(\overline{x_2} \vee \overline{x_1})} = \qquad \text{(АБО / І-НЕ)}$$

$$= (\overline{x_2} \vee \overline{x_1}) \vee \overline{\overline{x_2} \vee \overline{x_1}}. \qquad \text{(АБО-НЕ / АБО)}$$

На базі ДКНФ аналогічним чином отримаємо ще чотири нормальні форми:

$$y = (x_2 \vee \overline{x_1})(\overline{x_2} \vee x_1) = \qquad \text{(АБО / І)}$$

$$= \overline{\overline{(x_2 \vee \overline{x_1})(\overline{x_2} \vee x_1)}} =$$

$$= \overline{\overline{(x_2 \vee \overline{x_1})} \vee \overline{(\overline{x_2} \vee x_1)}} = \qquad \text{(АБО-НЕ / АБО-НЕ)}$$

$$= \overline{x_2 x_1 \vee \overline{x_2}\,\overline{x_1}} = \qquad \text{(І / АБО-НЕ)}$$

$$= \overline{\overline{x_2 x_1}} \cdot \overline{\overline{x_2}\,\overline{x_1}}. \qquad \text{(І-НЕ / І)}$$

Останні чотири форми можна аналогічно одержати виходячи із *заперечення перемикальної функції*, що відповідає формі І / АБО-НЕ.

Для отримання заперечення функції виписують із запереченням кон'юнктивні терми наборів, на яких функція дорівнює нулю, поєднані операцією диз'юнкції.

Заперечення заданої функції має вигляд:

$$y_{\overline{\text{ДДНФ}}} = \overline{\overline{x_2}\,\overline{x_1} \vee x_2 x_1}.$$

Послідовно отримаємо:

$$y = \overline{\overline{x_2}\,\overline{x_1} \vee x_2 x_1} = \qquad \text{(І / АБО-НЕ)}$$

$$= \overline{\overline{x_2}\,\overline{x_1}} \cdot \overline{x_2 x_1} = \qquad \text{(І-НЕ / І)}$$

$$= (x_2 \vee \overline{x_1})(\overline{x_2} \vee x_1) = \qquad \text{(АБО / І)}$$

$$= \overline{\overline{(x_2 \vee \overline{x_1})(\overline{x_2} \vee x_1)}} =$$

$$= \overline{\overline{(x_2 \vee \overline{x_1})} \vee \overline{(\overline{x_2} \vee x_1)}} = \qquad \text{(АБО-НЕ / АБО-НЕ)}$$

Із числа восьми нормальних форм розширеної алгебри можна знайти ті форми, які дозволяють побудувати комбінаційні схеми з використанням певної підмножини логічних елементів. У результаті дослідження параметрів комбінаційних схем можна обрати з них такі, що відповідають заданій цільовій функції проектування,

наприклад, мають мінімальну складність або максимальну швидкодію.

Зауважимо, що отримання нормальних форм не вирішує завдання побудови комбінаційних схем в умовах обмеження кількості входів логічних елементів. У такому випадку необхідно одержати операторні форми функцій (див. розділ А1-2.11).

А1-1.10. Проблема функціональної повноти систем перемикальних функцій

Проблема функціональної повноти систем перемикальних функцій є однією з найважливіших проблем теорії цифрових автоматів. На практиці для побудови складних логічних схем використовують порівняно невеликий набір елементів, тобто функції від багатьох аргументів подаються за допомогою функцій від меншої кількості аргументів за допомогою принципу суперпозиції. Можливість такого перетворення пов'язана з поняттям функціональної повноти систем перемикальних функцій.

Будь-яку сукупність функцій можна вважати *класом*. Відрізняють функціонально замкнені та функціонально повні класи.

Клас функцій є функціонально повним, якщо за допомогою функцій даного класу можна представити будь-яку перемикальну функцію від будь-якої кількості аргументів методом суперпозиції.

Функціонально замкненим називають клас функцій, будь-яка суперпозиція функцій в якому не виводить за межі даного класу.

Серед функціонально замкнених класів існують особливі класи, які мають назву дивовижних і передповних.

Передповним називають такий клас функцій, додавання до якого будь-якої функції, яка не входить у даний клас, робить клас функцій функціонально повним.

Теорема про функціональну повноту систем функцій базується на приналежності функцій передповним класам.

Дослідження класів перемикальних функцій показало, що існують п'ять передповних класів:

— функцій що зберігають 0 ($К_0$);
— функцій що зберігають 1 ($К_1$);
— самодвоїстих функцій ($К_С$);
— монотонних функцій ($К_М$);
— лінійних функцій ($К_Л$).

Теорема Поста-Яблонського

Для того, щоб система функцій була функціонально повною, необхідно і достатньо, щоб до її складу входила хоча б одна функція,

що не зберігає 0, хоча б одна функція, що не зберігає 1, хоча б одна несамодвоїста функція, хоча б одна немонотонна функція і хоча б одна нелінійна функція.

Іншими словами, система функцій є функціонально повною тоді й тільки тоді, коли вона цілком не міститься в жодному із передповних класів.

Розглянемо передповні класи докладніше.

До перемикальних функцій класу K_0, що зберігають 0, належать усі функції $f(x_1, x_2, ..., x_n)$, для яких справедливе співвідношення

$$f(0, ..., 0) = 0.$$

До функцій класу K_1, що зберігають 1, належать усі функції, для яких справедливе співвідношення

$$f(1, 1, ..., 1) = 1.$$

Булева функція називається *самодвоїстою*, якщо на будь-яких двох протилежних наборах вона приймає протилежні значення, тобто

$$f(x_1, x_2, ..., x_n) = \overline{f}(\overline{x_1}, \overline{x_2}, ..., \overline{x_n}).$$

Для визначення класу K_M монотонних функцій дамо наступне визначення.

Двійковий набір $\alpha = <\alpha_1, \alpha_2, ..., \alpha_n>$ не менше двійкового набору $\beta = <\beta_1, \beta_2, ..., \beta_n>$ (тобто $\alpha \geq \beta$), якщо для кожної пари $(\alpha_i, \beta_i \mid i \in \overline{1, n})$ справедливе співвідношення $\alpha_i \geq \beta_i$. Такі набори називають *порівняними*.

Якщо для двох наборів не виконується ні співвідношення $\alpha \geq \beta$, ні співвідношення $\alpha \geq \beta$, то набори вважаються непорівняними. Наприклад, набори 1011 і 1010 порівняні (1011 ≥ 1010), а набори 1011 й 0100 є *непорівняними*.

Функція $f(x_1, x_2, ..., x_n)$ називається *монотонною*, якщо для двох будь-яких порівняних наборів $\alpha = <\alpha_1, \alpha_2, ..., \alpha_n>$ і $\beta = <\beta_1, \beta_2, ..., \beta_n>$, таких що $\alpha \geq \beta$, має місце нерівність

$$f(\alpha_1, \alpha_2, ..., \alpha_n) \geq f(\beta_1, \beta_2, ..., \beta_n).$$

Функція називається *лінійною* (належить класу $K_Л$), якщо є лінійним її поліном Жегалкіна, тобто

$$f(x_1, x_2, ..., x_n) = c_0 \oplus c_1 x_1 \oplus ... \oplus c_n x_n,$$

де $c_i \in \{0, 1\}$ — коефіцієнти.

Належність усіх перемикальних функцій двох аргументів передповним класам показана в табл. А1-1.11, де знак «+» відповідає належності, а знак «–» — неналежності до відповідного класу.

Усі розглянуті алгебри мають функціонально повні системи перемикальних функцій.

Таблиця А1-1.11

НАЛЕЖНІСТЬ ФУНКЦІЙ ДО ПЕРЕДПОВНИХ КЛАСІВ

x_2	x_1	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
K_0		+	+	+	+	+	+	+	+	–	–	–	–	–	–	–	–
K_1		–	+	–	+	–	+	–	+	–	+	–	+	–	+	–	+
K_C		–	–	–	+	–	+	–	–	–	–	+	–	+	–	–	–
K_M		+	+	–	+	–	+	–	+	–	–	–	–	–	–	–	+
$K_Л$		+	–	–	+	–	+	+	–	–	+	+	–	+	–	–	+

Як приклад у табл. А1-1.12 показано входження функцій булевої алгебри до передповних класів (знак «+» означає входження, а знак «–» означає невходження до відповідних класів).

Таблиця А1-1.12

НАЛЕЖНІСТЬ ФУНКЦІЙ АЛГЕБРИ БУЛЯ ДО ПЕРЕДПОВНИХ КЛАСІВ

Функція	Клас				
	K_0	K_1	K_C	K_M	$K_Л$
І	+	+	–	+	–
АБО	+	+	–	+	–
НЕ	–	–	+	–	+

Зауваження. Алгебра Буля має надлишкову систему функцій, що видно з табл. А1-1.12. З цієї системи можна вилучити функцію І або функцію АБО. Для забезпечення функціональної повноти достатньо залишити пару функцій: {І, НЕ} чи {АБО,

НЕ}. В кожній колонці таблиці при цьому залишиться знак «–», тобто буде виконана вимога теореми функціональної повноти

А1-2. Комбінаційні схеми

А1-2.1. Проблема мінімізації перемикальних функцій

Функції F і Ψ, подані у різних формах, називаються *еквівалентними*, якщо ці функції приймають однакові значення на всіх наборах аргументів, тобто їм відповідає одна таблиця істинності.

Еквівалентні форми перемикальних функцій можуть відрізнятися ціною (кількістю букв у аналітичному записі функції), а комбінаційні схеми, що реалізують еквівалентні форми функцій, можуть відрізнятися складністю.

Метою мінімізації перемикальних функцій є спрощення комбінаційних схем, які реалізують ці перемикальні функції. Проблема мінімізації зводиться до відшукання форми подання функції з мінімальною ціною. Мінімізацію можна виконувати у різних алгебрах.

Розглянемо методи мінімізації в диз'юнктивних формах булевої алгебри. Введемо деякі означення.

Перемикальна функції G називається *імплікантою* функції F, якщо функція G приймає значення одиниці тільки з числа тих наборів, на яких приймає значення одиниці функція F.

У загальному випадку імпліканта частково покриває функцію, тобто приймає значення одиниці не на всіх наборах, на котрих функція має значення одиниці. Імпліканта може бути подана кон'юнктивним термом r-го ранга, де $r \le n$ (n — кількість аргументів функції).

Імпліканта, ніяка частина якої не є імплікантою, називається *простою імплікантою*.

Диз'юнкція простих імплікант називається *скороченою ДНФ* (СДНФ).

Сукупність усіх простих імплікант в СДНФ завжди покриває всі одиничні значення функції, але може містити надлишкові (зайві) імпліканти, які повторно покривають функцію на деяких наборах. З метою зменшення ціни форми такі імпліканти можна вилучити із складу СДНФ.

СДНФ без надлишкових імплікант називають *тупиковою ДНФ* (ТДНФ).

ТДНФ з мінімальною ціною називають *мінімальною ДНФ* (МДНФ).

Функція може мати декілька ТДНФ і МДНФ.

Таким чином, формальні методі мінімізації функцій зводяться до знаходження МДНФ функції.

А1-2.2. Метод мінімізації Квайна

Вихідною формою подання перемикальної функції для виконання мінімізації за методом Квайна є ДДНФ.

Метод Квайна базується на використанні співвідношення неповного склеювання

$$Ax \vee A\bar{x} = Ax \vee A\bar{x} \vee A \qquad \text{(А1-2.1)}$$

і співвідношення поглинання

$$BC \vee C = C, \qquad \text{(А1-2.2)}$$

де A, B і C — довільні кон'юнктивні терми;
x — змінна.

Завдання мінімізації методом Квайна складається з находження СДНФ з використанням співвідношень неповного склеювання і поглинання. Далі за допомогою таблиці покриття, яка дозволяє позбавитися надлишкових простих імплікант, отримують ядро функції (якщо воно є), потім знаходять ТДНФ і обирають з них МДНФ.

Під *ядром функції* розуміють сукупність імплікант, які неможливо вилучити із СДНФ. Такі імпліканти покривають деякі конституанти тільки самостійно.

Виконання мінімізації перемикальної функції здійснюється за такими етапами.

1. Записати перемикальну функцію у вихідній формі, якою є ДДНФ.
2. Застосувати співвідношення неповного склеювання (А1-2.1) послідовно до конституент одиниці, потім до імплікант (n – 1)-го рангу, (n – 2)-го рангу і так далі, поки формування нових імплікант можливе.
3. Виконати всі можливі поглинання, використовуючи співвідношення (А1-2.2), в результаті чого визначаються всі прості імпліканти, які складають СДНФ.
4. Побудувати таблицю покриття (імплікантну матрицю) для подальшого спрощення запису функції.
5. Визначити ядро перемикальної функції та всі ТДНФ. З числа отриманих ТДНФ вибрати МДНФ.

Приклад

Завдання. Виконати мінімізацію перемикальної функції, заданої таблицею істинності (табл. А1-2.1), методом Квайна.

Таблиця А1-2.1

ТАБЛИЦЯ ІСТИННОСТІ

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

Виконання завдання

Задана у ДДНФ перемикальна функція має вигляд

$$y = \overline{x}_3\overline{x}_2x_1 \vee \overline{x}_3x_2\overline{x}_1 \vee \overline{x}_3x_2x_1 \vee x_3\overline{x}_2x_1 \vee x_3x_2\overline{x}_1.$$

Виконуємо попарне склеювання конституент одиниці відповідно до співвідношення неповного склеювання (А1-2.1):

$$\overline{x}_3\overline{x}_2x_1 \vee \overline{x}_3x_2x_1 = \overline{x}_3\overline{x}_2x_1 \vee \overline{x}_3x_2x_1 \vee \overline{x}_3x_1;$$
$$\overline{x}_3\overline{x}_2x_1 \vee x_3\overline{x}_2x_1 = \overline{x}_3\overline{x}_2x_1 \vee \overline{x}_3x_2x_1 \vee \overline{x}_2x_1;$$
$$\overline{x}_3x_2\overline{x}_1 \vee \overline{x}_3x_2x_1 = \overline{x}_3x_2\overline{x}_1 \vee \overline{x}_3x_2x_1 \vee \overline{x}_3x_2;$$
$$\overline{x}_3x_2\overline{x}_1 \vee x_3x_2\overline{x}_1 = \overline{x}_3x_2\overline{x}_1 \vee x_3x_2\overline{x}_1 \vee x_2\overline{x}_1.$$

Одержали множину імплікант 2-го рангу:

$$\overline{x}_3x_1,\ \overline{x}_2x_1,\ \overline{x}_3x_2,\ x_2\overline{x}_1.$$

Подальше склеювання імплікант неможливе. Тоді задана перемикальна функцію приймає такий вигляд

$$y = \overline{x}_3\overline{x}_2x_1 \vee \overline{x}_3x_2\overline{x}_1 \vee \overline{x}_3x_2x_1 \vee x_3\overline{x}_2x_1 \vee x_3x_2\overline{x}_1 \vee \overline{x}_3x_1 \vee \overline{x}_2x_1 \vee x_3\overline{x}_2 \vee x_2\overline{x}_1.$$

Далі, виконавши поглинання відповідно до співвідношення поглинання (А1-2.2), одержуємо СДНФ:

$$y = \overline{x}_3x_1 \vee \overline{x}_2x_1 \vee \overline{x}_3x_2 \vee x_2\overline{x}_1.$$

На наступному етапі мінімізації заданої перемикальної функції будуємо таблицю покриття (табл. А1-2.2).

Таблиця А1-2.2

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституанти				
	$\overline{x_3}\,\overline{x_2}x_1$	$\overline{x_3}x_2\overline{x_1}$	$\overline{x_3}x_2x_1$	$x_3\overline{x_2}x_1$	$x_3x_2\overline{x_1}$
$\overline{x_3}x_1$	Ⓥ		Ⓥ		
$\overline{x_2}x_1$	Ⓥ			Ⓥ	
$\overline{x_3}x_2$		∨	∨		
$x_2\overline{x_1}$		Ⓥ			Ⓥ

Знаходимо ядро функції — сукупність імплікант, що відповідають одноразово покритим конституентам. В даному прикладі ядро складають імпліканти $\overline{x_2}x_1$ і $x_2\overline{x_1}$.

Далі ядро необхідно доповнити імплікантами для одержання повного покриття всіх конституент вихідної перемикальної функції. У загальному випадку можна отримати різні варіанти покриття, які будуть являти собою ТДНФ. Серед отриманих ТДНФ обираємо форму з мінімальною складністю, тобто МДНФ.

Виходячи з таблиці покриття знаходимо дві рівноцінні ТДНФ:

$$y_{\text{ТДНФ1}} = \overline{x_3}x_1 \vee \overline{x_2}x_1 \vee x_2\overline{x_1};$$
$$y_{\text{ТДНФ2}} = \overline{x_2}x_1 \vee \overline{x_3}x_2 \vee x_2\overline{x_1}.$$

Одну з них обираємо як МДНФ, наприклад:

$$y_{\text{МДНФ}} = \overline{x_3}x_1 \vee \overline{x_2}x_1 \vee x_2\overline{x_1}.$$

А1-2.3. Метод мінімізації Квайна—Мак-Класкі

Метод Квайна—Мак-Класкі є модифікацією методу Квайна і також ґрунтується на співвідношеннях неповного склеювання (А1-2.1) і поглинання (А1-2.2). Особливістю методу є використання цифрової форми запису термів перемикальних функцій.

Наприклад, функція може бути подана у вигляді

$$f(x_3, x_2, x_1) = \overline{x}_3\overline{x}_2\overline{x}_1 \vee \overline{x}_3x_2\overline{x}_1 \vee x_3x_2x_1 = 000 \vee 010 \vee 111$$

У цьому випадку зменшується кількість символів для подання термів і кількість операцій у процесі мінімізації, що робить метод зручним під час програмної реалізації.

Мінімізацію перемикальних функцій методом Квайна—Мак-Класкі розглянемо на прикладі геометричної інтерпретації подання перемикальних функцій.

Кожен набір аргументів (x_n, …, x_2, x_1) є n-вимірним вектором (де n — кількість аргументів) і визначає точку n-вимірного простору. Сукупність усіх наборів, на яких визначена перемикальна функція n-аргументів, зображується n-вимірним кубом. Конституентам відповідають вершини куба, а імплікантам — ребра і грані. Кожній перемикальній функції відповідає певне просторове зображення.

Наприклад, для функції трьох змінних (рис. А1-2.1, *а*) конституентам відповідають вершини тривимірного куба, імплікантам 2–го рангу — ребра, а 1-го рангу — грані.

Можемо визначити *правило склеювання* для термів функції трьох змінних. Із геометричного подання функції виходить, що дві вершини, що належать одному й тому самому ребру і мають назву сусідніх вершин, склеюються за змінною, яка змінюється вздовж цього ребра.

Наприклад, для сусідніх вершин 000 і 001 (рис. А1-2.1, *а*) в цифровому вигляді можна записати (символом X позначаються змінні, по яких склеюються терми):

$$000 \vee 001 = 000 \vee 001 \vee 00\mathrm{X},$$

що відповідає

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

Після поглинання замість двох вершин одержимо ребро $\overline{x_3}\,\overline{x_2} \vee 00\mathrm{X}$.

Аналогічно, виконавши поетапно склеювання та поглинання, два ребра можна замінити на грань:

$$\mathrm{X01} \vee \mathrm{X11} = \mathrm{X01} \vee \mathrm{X11} \vee \mathrm{XX1};$$
$$\mathrm{X01} \vee \mathrm{X11} \vee \mathrm{XX1} = \mathrm{X11}.$$

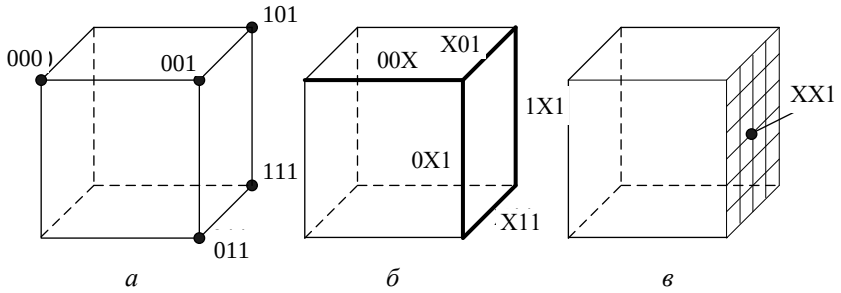


Рис. А1-2.1. Геометрична інтерпретація подання перемикальних функцій:
а — 0-куби; *б* — 1-куби; *в* — 2-куби

Терми максимального рангу (*n*-рангу) називають 0-кубами і позначають K^0, терми $(n-1)$-го рангу — 1-кубами (K^1), $(n-2)$-го рангу — 2-кубами (K^2) і т. д.

Якщо два 0-куби $K^0 = \{000\}$ і $K^0 = \{100\}$ розрізняються тільки однією координатою, вони утворюють 1-куб $K^1 = \{X00\}$, де X — незалежна змінна. Якщо два 1-куби $K^1 = \{X00\}$ і $K^1 = \{X10\}$ мають сумісну незалежну змінну і розрізняються однією координатою вони утворюють 2-куб — $K^2 = \{XX0\}$.

Сукупність *r*-кубів (де $r = \overline{0, m}$) називається *комплексом r*-кубів. Запишемо комплекси *r*-кубів для функції зображеної на рис. А1-2.1:

$$K^0 = \begin{cases} 000 \\ 001 \\ 011 \\ 101 \\ 111 \end{cases}; \; K^1 = \begin{cases} 00X \\ X01 \\ X11 \\ 1X1 \\ 0X1 \end{cases}; \; K^2 = \begin{cases} XX1 \\ XX1 \end{cases}.$$

Таким чином, під час застосування правил склеювання і поглинання символом X в *r*-кубах позначаються аргументи, по яких склеюються $(r+1)$-куби. Множина *r*-кубів *i*-го рангу (де $i = \overline{1, n}$) утворюють комплекс K^i.

Наведемо етапи мінімізації перемикальних функцій методом Квайна—Мак-Класкі.

1. Для заданої перемикальної функції виписують комплекс 0-кубів (K^0). Для зручності отримані куби упорядковують за кількістю одиниць. Формують групи кубів без одиниць, з однією одиницею, із двома одиницями і таке інше. В цьому випадку склеювання можливе тільки між кубами сусідніх груп.

2. Шляхом склеювання формують 1-куби, потім 2-куби і т. д. інше, поки можливе склеювання. Кожен черговий куб упорядковується таким чином. До однієї групи входять куби, що мають однакову кількість одиниць і загальну незалежну змінну X.

3. Виконують усі можливі поглинання і формують покриття *Z*, що відповідає скороченій ДНФ.

4. Будують таблицю покриття, на підставі якої визначають ядро перемикальної функції, можливі ТДНФ і МДНФ.

Приклад

Завдання. Виконати мінімізацію перемикальної функції *y*, заданої таблицею істинності (табл. А1-2.3), методом Квайна—Мак-Класкі.

Таблиця А1-2.3

ТАБЛИЦЯ ІСТИННОСТІ

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

Виконання завдання

Виходячи з таблиці істинності заданої перемикальної функції записуємо 0-куб K^0, поєднуючи набори у групи, що мають однакову кількість одиниць. Виконуючи склеювання, формуємо куб K^1 і потім куб K^2. Після виконання поглинань, одержуємо Z-покриття.

$$K^0 = \begin{cases} \cancel{000} \\ \cancel{010} \\ \cancel{100} \\ \cancel{011} \\ \cancel{101} \\ \cancel{110} \end{cases}; \qquad K^1 = \begin{cases} \cancel{X00} \\ \cancel{0X0} \\ \cancel{1X0} \\ \cancel{01X} \\ \cancel{10X} \end{cases}; \qquad K^2 = \{XX0; \qquad Z = \begin{cases} 01X \\ 10X \\ XX0 \end{cases}.$$

Куби в покритті Z відповідають простим імплікантам функції. Таким чином, отримана СДНФ:

$$y_{\text{СДНФ}} = \overline{x_3} x_2 \vee x_3 \overline{x_2} \vee \overline{x_1}.$$

Для видалення надлишкових імплікант і отримання МДНФ будуємо таблицю покриття (табл. А1-2.4).

Таблиця А1-2.4

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти					
	000	010	100	011	101	110
01X		Ⓥ		Ⓥ		
10X			Ⓥ		Ⓥ	
XX0	Ⓥ	Ⓥ	Ⓥ			Ⓥ

З таблиці покриття виходить, що всі отримані імпліканти входять до ядра перемикальної функції. Отже, МДНФ заданої перемикальної функції має такий вигляд:

$$y_{\text{МДНФ}} = \overline{x_3}x_2 \vee x_3\overline{x_2} \vee \overline{x_1}.$$

А1-2.4. Метод невизначених коефіцієнтів

Будь-яку перемикальну функцію можна подати у вигляді диз'юнкції всіх конституент і всіх можливих імплікант, помножених логічно на відповідний коефіцієнт, що може приймати значення 0 чи 1. Метод невизначених коефіцієнтів може бути використаний у будь-якій алгебрі перемикальних функцій. При цьому зазнають зміни тільки вихідні канонічні форми запису функцій і системи рівнянь для знаходження коефіцієнтів.

Наприклад, за $n = 2$ можна записати:

$$y = k_2^1 x_2 \vee k_2^0\overline{x_2} \vee k_1^1 x_1 \vee k_1^0\overline{x_1} \vee k_{21}^{00}\overline{x_2}\,\overline{x_1} \vee k_{21}^{01}\overline{x_2}x_1 \vee k_{21}^{10}x_2\overline{x_1} \vee k_{21}^{11}x_2x_1.$$

Кожна перемикальна функція визначається своїм набором значень коефіцієнтів. Для пошуку значень коефіцієнтів необхідно вирішити систему рівнянь:

$$\begin{cases} y(0,\ 0) = k_2^0 \vee k_1^0 \vee k_{21}^{00}; \\ y(0,\ 1) = k_2^0 \vee k_1^1 \vee k_{21}^{01}; \\ y(1,\ 0) = k_2^1 \vee k_1^0 \vee k_{21}^{10}; \\ y(1,\ 1) = k_2^1 \vee k_1^1 \vee k_{21}^{11}. \end{cases}$$

Усі ненульові коефіцієнти після процедури поглинань визначають сукупність простих імплікант, тобто дозволяють побудувати скорочену ДНФ. Мінімізацію зручно виконувати за допомогою спеціальної таблиці, що після знаходження простих імплікант розглядається як таблиця покриття. За допомогою цієї таблиці знаходять ТДНФ, а потім визначають МДНФ.

Етапи мінімізації методом невизначених коефіцієнтів:

1. Скласти таблицю коефіцієнтів.
2. Викреслити всі нульові коефіцієнти.
3. Виділити прості імпліканти.
4. Знайти покриття, що відповідають тупіковим ДНФ і обрати МДНФ.

Приклад

Завдання. Виконати мінімізацію перемикальної функції, заданої таблицею істинності (табл. А1-2.5), методом невизначених коефіцієнтів.

Таблиця А1-2.5

ТАБЛИЦЯ ІСТИННОСТІ

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

Виконання завдання

Складаємо таблицю коефіцієнтів (табл. А1-2.6). Викреслюємо в таблиці коефіцієнти, що знаходяться в рядках з нульовим значенням функції. Викреслені коефіцієнти мають нульові значення. Далі викреслюємо вже знайдені нульові коефіцієнти в інших рядках таблиці. Коефіцієнти, котрі залишилися, поглинають у рядку праворуч від себе всі інші коефіцієнти, в індекси яких входять індекси даного коефіцієнта. Наприклад, k_{32}^{01} поглинає k_{321}^{010}. Поглинені коефіцієнти в табл. А1-2.6 позначені зірочкою.

Коефіцієнти, що залишилися, $\{k_{32}^{01}, k_{32}^{10}, k_{31}^{01}, k_{31}^{10}, k_{21}^{01}, k_{21}^{10}\}$ визначають скорочену ДНФ:

$$y_{\text{СДНФ}} = \overline{x}_3 x_2 \vee x_3 \overline{x}_2 \vee \overline{x}_3 x_1 \vee x_3 \overline{x}_1 \vee \overline{x}_2 x_1 \vee x_2 \overline{x}_1.$$

Враховуємо в таблиці тільки ненульові коефіцієнти і розглядаємо її як таблицю покриття заданої перемикальної функції. Знаходимо дві ТДНФ, що містять по три імпліканти.

Отримані ТДНФ відповідають множині коефіцієнтів $\{k_{32}^{01}, k_{31}^{10}, k_{21}^{01}\}$ та $\{k_{32}^{10}, k_{31}^{01}, k_{21}^{10}\}$ і дорівнюють

$$y_{\text{ТДНФ}\,1} = \overline{x}_3 x_2 \vee x_3 \overline{x}_1 \vee \overline{x}_2 x_1;$$

$$y_{\text{ТДНФ}\,2} = x_3 \overline{x}_2 \vee \overline{x}_3 x_1 \vee x_2 \overline{x}_1.$$

Таблиця А1-2.6

ТАБЛИЦЯ КОЕФІЦІЄНТІВ

x_3	x_2	x_1	Коефіцієнти							y
0	0	0	k_3^0	k_2^0	k_1^0	k_{32}^{00}	k_{31}^{00}	k_{21}^{00}	k_{321}^{000}	0
0	0	1	k_3^0	k_2^0	k_1^1	k_{32}^{00}	k_{31}^{01}	k_{21}^{01}	k_{321}^{001*}	1
0	1	0	k_3^0	k_2^1	k_1^0	k_{32}^{01}	k_{31}^{00}	k_{21}^{10}	k_{321}^{010*}	1
0	1	1	k_3^0	k_2^1	k_1^1	k_{32}^{01}	k_{31}^{01}	k_{21}^{11}	k_{321}^{011*}	1
1	0	0	k_3^1	k_2^0	k_1^0	k_{32}^{10}	k_{31}^{10}	k_{21}^{00}	k_{321}^{100*}	1
1	0	1	k_3^1	k_2^0	k_1^1	k_{32}^{10}	k_{31}^{11}	k_{21}^{01}	k_{321}^{101*}	1
1	1	0	k_3^1	k_2^1	k_1^0	k_{32}^{11}	k_{31}^{10}	k_{21}^{10}	k_{321}^{110*}	1
1	1	1	k_3^1	k_2^1	k_1^1	k_{32}^{11}	k_{31}^{11}	k_{21}^{11}	k_{321}^{111}	0

Як МДНФ, наприклад, обираємо

$$y_{\text{МДНФ}} = x_3\overline{x_2} \vee \overline{x_3}x_1 \vee x_2\overline{x_1}.$$

А1-2.5. Графічний метод мінімізації функцій

Одержати МДНФ перемикальної функції, минаючи етапи формування скороченої і тупікової ДНФ, можна, застосувавши діаграми Вейча або карти Карно.

Діаграми Вейча і карти Карно для перемикальних функцій двох, трьох і чотирьох аргументів відповідно зображені на рис. А1-2.2, *а*, *б*. Усередині клітинок вказані номери наборів.

Графічні методи призначені для ручної мінімізації. Наочність цих методів зберігається за невеликої кількості аргументів. При цьому відшукання імплікант не формалізоване, і успіх мінімізації цілком визначається кваліфікацією оператора.

Кожна клітинка відповідає конституенті. Прямокутник, що містить 2^k клітинок ($k = \overline{1, (n-1)}$), відповідає імпліканті. Прямокутник максимального розміру відповідає простій імпліканті.

Обґрунтуванням графічного методу мінімізації є той факт, що розташовані поруч клітинки відповідають сусіднім наборам аргументів, що відрізняється значенням однієї змінної і, таким чином, можуть бути склеєні за методом Квайна (А1-2.1). За *n* перемінних кожна клітинка має *n* сусідніх клітинок.

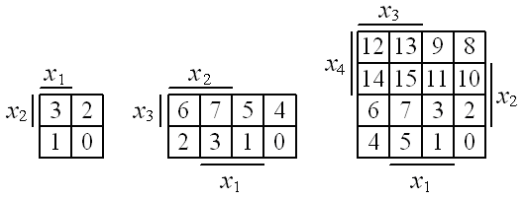
Чим більше клітинок містить прямокутник, тим менше букв входить у записі імпліканти. Імпліканта містить тільки ті змінні, котрі приймають однакові значення для всіх клітинок прямокутника.

Наведемо етапи мінімізації перемикальних функцій графічним методом.

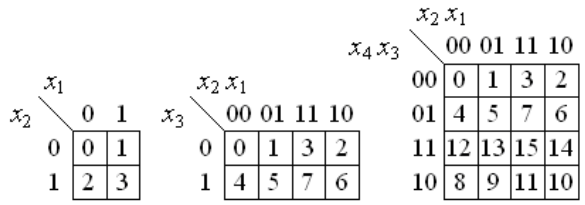
1. Заповнити діаграму Вейча або карту Карно. Значення функцій записують у клітинки, що відповідають номерам наборів.

2. Виконати об'єднання одиниць в прямокутники з максимально можливою кількістю клітинок (число клітинок, що можуть поєднуватися, має дорівнювати 2^k). При цьому кожна одиниця повинна входити як мінімум в один прямокутник. Прямокутник може містити й одну клітинку.

3. Визначити МДНФ. Сукупності простих імплікант, що входять у МДНФ, відповідає мінімальна множина прямокутників, що покривають усі одиниці.



а



б

Рис. А1-2.2. Графічний метод мінімізації перемикальних функцій:
а — діаграми Вейча; *б* — карти Карно

Приклад

Завдання. Виконати мінімізацію перемикальних функцій трьох аргументів y_1, y_2, y_3, заданих таблицею істинності (табл. А1-2.7), за допомогою діаграм Вейча.

Таблиця А1-2.7

ТАБЛИЦЯ ІСТИННОСТІ

x_3	x_2	x_1	y_1	y_2	y_3
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	1	1

Виконання завдання

Заповнимо діаграми Вейча і знайдемо мінімальну кількість прямокутників максимального розміру, що покривають усі одиниці (рис. А1-2.3).

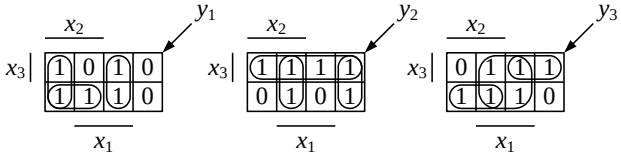


Рис. А1-2.3. Діаграми Вейча

У результаті отримаємо МДНФ функцій y_1, y_2, y_3:

$$y_{1\text{МДНФ}} = x_2\overline{x_1} \vee \overline{x_3}x_2 \vee \overline{x_2}x_1;$$

$$y_{2\text{МДНФ}} = x_3 \vee x_2x_1 \vee \overline{x_2}\,\overline{x_1};$$

$$y_{3\text{МДНФ}} = x_1 \vee \overline{x_3}x_2 \vee x_3\overline{x_2}.$$

Слід зазначити, що не обов'язково позначати всі можливі прямокутники в діаграмі Вейча. Достатньо тільки покрити всі одиниці хоча б один раз. Наприклад, у розглянутому прикладі в таблиці для функції y_1 можна ще виділити прямокутник, що відповідає імпліканті $\overline{x_3}x_1$, але ця імпліканта є зайвою, бо всі одиниці зазначенного прямокутника вже належать іншим прямокутникам.

Сусідні клітинки можуть бути розташовані у протилежних рядках (колонках) таблиці (див. наступний приклад).

Приклад

Завдання. За допомогою метода Вейча виконати мінімізацію перемикальної функції чотирьох аргументів, заданою ДНФ загального вигляду

$$F = x_4\overline{x_2}\overline{x_1} \vee \overline{x_3}x_2\overline{x_1} \vee x_4x_3x_2\overline{x_1} \vee \overline{x_4}x_1 \vee \overline{x_4}x_3\overline{x_2} =$$
$$= 8,\ 12 \vee 2,\ 10 \vee 14 \vee 1,\ 3,\ 5,\ 7 \vee 4,\ 5.$$

Числами визначені номери наборів, які відповідають одиничним значенням констітуенти та імплікант.

Виконання завдання

Заповнимо діаграму Вейча і знайдемо мінімальну кількість прямокутників максимального розміру, що покривають ядро функції (рис. А1-2.4).

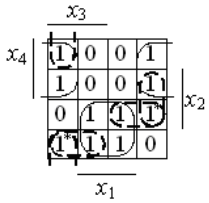


Рис. А1-2.4. Діаграма Вейча

Ядро функції, як видно із діаграми, складає множина імплікант $\left\{x_4\overline{x_1},\ \overline{x_4}x_1\right\}$, бо покрити ці імпліканти іншими прямокутниками на чотири клітинки неможливо. Кожну з одиниць, що залишилися непокритими (одиниці в клітинках з номерами 2 і 4 позначені зірочками), можна об'єднати з сусідніми клітинками двома різними способами, відміченими пунктиром. Таким чином можна обрати як МДНФ одну з чотирьох наступних ТДНФ:

$$F_{\text{ТДНФ1}} = x_4\overline{x_1} \vee \overline{x_4}x_1 \vee x_3\overline{x_2}\overline{x_1} \vee \overline{x_4}\overline{x_3}x_2;$$
$$F_{\text{ТДНФ2}} = x_4\overline{x_1} \vee \overline{x_4}x_1 \vee x_3\overline{x_2}\overline{x_1} \vee \overline{x_3}x_2\overline{x_1};$$
$$F_{\text{ТДНФ3}} = x_4\overline{x_1} \vee \overline{x_4}x_1 \vee \overline{x_4}x_3\overline{x_2} \vee \overline{x_4}\overline{x_3}x_2;$$
$$F_{\text{ТДНФ4}} = x_4\overline{x_1} \vee \overline{x_4}x_1 \vee \overline{x_4}x_3\overline{x_2} \vee \overline{x_3}x_2\overline{x_1}.$$

А1-2.6. Метод мінімізації Блейка—Порецького

Метод Блейка—Порецького дозволяє отримати СДНФ перемикальної функції з її ДНФ загального вигляду. Це дозволяє вилучити необхідність одержання ДДНФ функції, що може спростити процес мінімізації, особливо для функцій від великої кількості аргументів.

Метод базується на використанні формули *узагальненого* склеювання:

$$Ax \vee B\bar{x} = Ax \vee B\bar{x} \vee AB, \qquad \text{(А1-2.3)}$$

та поглинання (А1-2.2).

Справедливість (А1-2.3) можна довести таким чином. Виходячи з (А1-2.2) отримуємо:

$$Ax = Ax \vee BAx; \quad B\bar{x} = B\bar{x} \vee AB\bar{x}.$$

Отже,

$$Ax \vee B\bar{x} = Ax \vee ABx \vee B\bar{x} \vee AB\bar{x} =$$
$$= Ax \vee B\bar{x} \vee AB(x \vee \bar{x}) = Ax \vee B\bar{x} \vee AB.$$

В основі методу лежить таке твердження. Якщо в довільній ДНФ перемикальної функції зробити всі можливі узагальнені склеювання і виконати всі поглинання, то в результаті будуть отримані всі прості імпліканти, тобто СДНФ функції.

Етапи мінімізації можна сформулювати так:

1. Виконати всі можливі узагальнені склеювання імплікант.
2. Виконати всі можливі поглинання імплікант.
3. Порівняти множини імплікант до склеювання та одержані після поглинання.
4. Якщо множини імплікант не співпадають, то повторити п.1 та п. 2. Якщо множини імплікант співпадають, то перейти до п. 4.
5. Побудувати таблицю покриття. Заголовками рядків таблиці є одержані імпліканти, а заголовками колонок — імпліканти (констітуенти) заданої форми функції.
6. Знайти можливі покриття функції та обрати з них покриття з мінімальною ціною.

У результаті мінімізації може бути одержана як мінімальна, так і одна з тупикових форм функції. Для гарантованого одержання мінімальної форми (МДНФ) необхідно, щоб у заголовках колонок були всі констітуенти одиниці вихідної функції.

Приклад

Завдання. Методом Блейка—Порецького виконати мінімізацію перемикальної функції

$$f = ab\bar{d} \vee abc \vee a\bar{b}d \vee a\bar{b}\bar{c}\bar{d} \vee \bar{a}\bar{b}\bar{c}.$$

Виконання завдання

Робимо узагальнене склеювання вихідних імплікант:

$$ab\bar{d} \vee a\bar{b}\bar{c}\bar{d} = ab\bar{d} \vee a\bar{b}\bar{c}\bar{d} \vee a\bar{c}\bar{d};$$
$$abc \vee a\bar{b}d = abc \vee a\bar{b}d \vee acd;$$
$$a\bar{b}d \vee a\bar{b}\bar{c}\bar{d} = a\bar{b}d \vee a\bar{b}\bar{c}\bar{d} \vee a\bar{b}\bar{c};$$
$$a\bar{b}d \vee a\bar{b}\bar{c} = a\bar{b}d \vee a\bar{b}\bar{c} \vee b\bar{c}d;$$
$$a\bar{b}\bar{c}\bar{d} \vee a\bar{b}\bar{c} = a\bar{b}\bar{c}\bar{d} \vee a\bar{b}\bar{c} \vee b\bar{c}\bar{d}.$$

Склеюємо нові одержані імпліканти:

$$a\bar{c}\bar{d} \vee b\bar{c}d = a\bar{c}\bar{d} \vee b\bar{c}d \vee a\bar{b}\bar{c};$$
$$acd \vee a\bar{b}\bar{c} = acd \vee a\bar{b}\bar{c} \vee a\bar{b}d;$$
$$acd \vee b\bar{c}d = acd \vee b\bar{c}d \vee a\bar{b}d;$$
$$b\bar{c}d \vee b\bar{c}\bar{d} = b\bar{c}d \vee b\bar{c}\bar{d} \vee b\bar{c}.$$

Подальше склеювання неможливе.

Таким чином, після першого циклу узагальненого склеювання і поглинання одержали таку множину імплікант:

$$\{ab\bar{d}, abc, a\bar{b}d, a\bar{c}d, acd, b\bar{c}\}.$$

Другий цикл склеювання одержаних імплікант і їх поглинання приводить до такої самої множини імплікант, тобто можна переходити до побудови таблиці покриття.

Із табл. А1-2.8 знаходимо покриття з мінімальною ціною:

$$f_{\text{ТДНФ}} = ab\bar{d} \vee abc \vee a\bar{b}d \vee b\bar{c}.$$

Таблиця А1-2.8

ТАБЛИЦЯ ПОКРИТТЯ

Прості імпліканти	Вихідні імпліканти				
	$ab\bar{d}$	abc	$a\bar{b}d$	$a\bar{c}\bar{d}$	$\bar{a}\bar{b}c$
$ab\bar{d}$	Ⓥ				
abc		Ⓥ			
$a\bar{b}d$			Ⓥ		
$a\bar{c}\bar{d}$				∨	
acd					
$\bar{b}\bar{c}$				Ⓥ	Ⓥ

Одержана форма є однією з тупикових форм функції.

Можна показати, що для заданої функції МДНФ має вигляд

$$f_{\text{ТДНФ}} = ab\bar{d} \vee acd \vee \bar{b}\bar{c}.$$

Такий результат можна одержати з повної таблиці покриття, в якій заголовками колонок є всі конституенти одиниці заданої функції (табл. А1-2.9).

Треба зауважити, що побудова повної таблиці покриття може у деяких випадках звести нанівець усі переваги методу Блейка—Порецького відносно методів Квайна та Квайна—Мак-Класкі, бо це потребує виконання додаткових операцій.

Таблиця А1-2.9

ПОВНА ТАБЛИЦЯ ПОКРИТТЯ

Прості імпліканти	Конституенти							
	$abc\bar{d}$	$ab\bar{c}\bar{d}$	$abcd$	$a\bar{b}cd$	$a\bar{b}\bar{c}d$	$a\bar{b}\bar{c}\bar{d}$	$\bar{a}\bar{b}\bar{c}d$	$\bar{a}\bar{b}\bar{c}d$
$ab\bar{d}$	Ⓥ	Ⓥ						
abc	∨		∨					
$a\bar{b}d$				∨	∨			
$a\bar{c}\bar{d}$		∨				∨		
acd			Ⓥ	Ⓥ				
$\bar{b}\bar{c}$					Ⓥ	Ⓥ	Ⓥ	Ⓥ

А1-2.7. Знаходження покриття функцій методом Петрика

Метод Петрика використовується для знаходження можливих покриттів перемикальної функції з таблиці покриття, яка одержана будь-яким методом знаходження СДНФ. Метод Петрика дозволяє формалізувати процес знаходження мінімального покриття функції.

Сутність методу полягає в такому. Таблицю покриття функції подають у вигляді логічного виразу, спрощення якого приводить до визначення всіх можливих множин імплікант, що покривають функцію.

Будемо вважати, що, наприклад, за допомогою метода Квайна, одержана таблиця покриття, заголовками рядків якої є прості імпліканти, а заголовками стовпців — констітуенти одиниці заданої функції.

Можна запропонувати такі етапи реалізації методу.

1. Для зручності перетворення логічних виразів усі прості імпліканти, що входять у СДНФ, позначаються довільними символами (зазвичай прописними латинськими літерами).

2. Для кожного *i*-го стовпця таблиці покриття будується умова покриття відповідної костітуенти у вигляді диз'юнкції всіх імплікант, що покривають цю конституанту.

3. Записується умова покриття всієї функції, яка представляє собою кон'юнкцію одержаних в п. 2 диз'юнкцій (логічних умов покриття окремих констітуент функції).

4. Одержаний логічний вираз приводиться до бездужкової форми. Після спрощення виразу та виконання всіх можливих поглинань формується диз'юнкція кон'юнкцій. Кожна кон'юнкція в одержаній формі відповідає множині простих імплікант, що покривають функцію, тобто ТДНФ. Виходячи з одержаних тупікових форм обирають МДНФ функції.

Приклад

Завдання. Використовуючи задану таблицю покриття (табл. А1-2.10), знайти всі можливі ТДНФ перемикальної функції f за методом Петрика.

Таблиця А1-2.10

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти					
	$x_4\overline{x_3}\overline{x_2}\overline{x_1}$	$x_4x_3\overline{x_2}\overline{x_1}$	$x_4\overline{x_3}x_2\overline{x_1}$	$x_4x_3x_2\overline{x_1}$	$\overline{x_4}x_3x_2x_1$	$x_4x_3x_2x_1$
$x_4\overline{x_1}$	∨	∨	∨	∨		
$x_4x_3x_2$				∨		∨
$x_3x_2x_1$					∨	∨

Виконання завдання

Позначимо прості імпліканти так:

$$x_4\overline{x_1} = A;\ \ x_4x_3x_2 = B;\ \ x_3x_2x_1 = C.$$

Запишемо умову покриття окремо для кожної конституанти у формі «конституента → умова»:

$$x_4\overline{x_3}\,\overline{x_2}\,\overline{x_1} \to A;$$

$$x_4x_3\overline{x_2}\,\overline{x_1} \to A;$$

$$x_4\overline{x_3}x_2\overline{x_1} \to A;$$

$$x_4x_3x_2\overline{x_1} \to (A \vee B);$$

$$\overline{x_4}x_3x_2x_1 \to C;$$

$$x_4x_3x_2x_1 \to (B \vee C).$$

Сформуємо загальну умову покриття всіх констітуент:

$$A \cdot A \cdot A \cdot (A \vee B) \cdot C \cdot (B \vee C).$$

Застосовуючи правило поглинання вигляду $(A \vee B)A = A$, спростимо отриманий вираз:

$$A(A \vee B)C(B \vee C) = AC.$$

В результаті спрощення отримана одна кон'юнкція, що містить дві прості імпліканти $A = x_4\overline{x_1}$ і $C = x_3x_2x_1$, які складають ТДНФ. Отримана форма є також МДНФ:

$$f_{\text{МДНФ}} = x_4 \cdot \overline{x}_1 \vee x_3 \cdot x_2 \cdot x_1.$$

Завдання. Виходячи з таблиці коефіцієнтів (табл. А1-2.6) побудувати таблицю покриття функції $y(x_3, x_2, x_1)$ в цифровій формі і знайти ТДНФ та МДНФ функції методом Петрика.

Виконання завдання

Будуємо таблицю покриття (табл. А1-2.11).

Таблиця А1-2.11

ТАБЛИЦЯ ПОКРИТТЯ

Імлі канти (1-куби)	Конституенти (0-куби)					
	001	010	011	100	101	110
A = 01X		∨	∨			
B = 10X				∨	∨	
C = 0X1	∨		∨			
D = 1X0				∨		∨
E = X01	∨				∨	
G = X10		∨				∨

Позначимо в табл. А1-2.11 прості імпліканти літерами і запишемо умови покриття констітуент:

$$001 \to (C \vee E);$$
$$010 \to (A \vee G);$$
$$011 \to (A \vee C);$$
$$100 \to (B \vee D);$$
$$101 \to (B \vee E);$$
$$110 \to (D \vee G).$$

Загальна умова покриття всіх констітуент має вигляд

$$(C \vee E)(A \vee G)(A \vee C)(B \vee D)(B \vee E)(D \vee G).$$

Після спрощення отримаємо вираз

$$ADE \vee BCG,$$

який відповідає двом можливим покриттям функції:

$$y_{\text{ТДНФ1}} = A \vee D \vee E = 01X \vee 1X0 \vee X01;$$
$$y_{\text{ТДНФ2}} = B \vee C \vee G = 10X \vee 0X1 \vee X10.$$

Будь-яку з цих форм можна обрати у якості МДНФ, наприклад:

$$y_{\text{МДНФ}} = B \vee C \vee G = 10X \vee 0X1 \vee X10 = x_3\overline{x_2} \vee \overline{x_3}x_1 \vee x_2\overline{x_1}.$$

А1-2.8. Мінімізація систем перемикальних функцій

За спільної мінімізації декількох перемикальних функцій необхідно враховувати можливе дублювання однакових елементів з однаковими логічними сигналами на входах під час побудови комбінаційної схеми, що приводить до її ускладнення.

Розглянемо мінимізацію систем функцій у диз'юнктивних нормальних формах булевої алгебри. Для усунення повторення однакових елементів у схемі на етапі мінімізації системи функцій необхідно виявити однакові імпліканти у запису різних функцій.

Для мінімізації можуть бути використані різні методи, зокрема, методи Квайна і Квайна—Мак-Класкі.

Вихідною формою для мінімізації системи функцій методом Квайна та Квайна—Мак-Класкі є ДДНФ системи перемикальних функцій, для отримання якої необхідно подати в ДДНФ кожну функцію. При цьому кожній конституенті одиниці приписується множина міток, що визначають її приналежність до певної функцій.

Наведемо етапи мінімізації системи перемикальних функцій:

1. Записати ДДНФ системи перемикальних функцій.
2. Виконати склеювання термів.
3. Виконати всі можливі поглинання.
4. Скласти таблицю покриття.
5. Вибрати покриття для кожної функції.

Відмінності мінімізації систем перемикальних функцій від мінімізації окремих функцій полягають у такому.

Склеювання здійснюються тільки для тих термів, що мають хоча б одну однакову мітку. Отриманому в результаті склеювання терму присвоюється множина міток, що є перетинанням множин міток термів, які склеювались.

Поглинання одного терму іншим здійснюється тільки в тому випадку, коли множині міток двох термів цілком збігаються.

Під час вибору остаточної форми подання перемикальної функцій з використанням таблиці покриттів необхідно прагнути до того, щоб загальне число букв у покритті було мінімальним.

Приклад

Завдання. Виконати спільну мінімізацію системи перемикальних функцій y_1, y_2, y_3, заданих таблицею істинності (табл. А1-2.12).

Таблиця А1-2.12

ПАРАМЕТРИ ФУНКЦІЙ

x_3	x_2	x_1	y_1	y_2	y_3
0	0	0	1	0	1
0	0	1	1	1	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	1	0
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	0	0	1

Виконання завдання

Для вирішення задачі мінімізації системи перемикальних функцій скористаємося методом Квайна—Мак-Класкі.

Формуємо комплекси кубів K^0 і приписуємо їм відповідні множини міток, що визначають належність термів до певної функцій.

Виконуючи склеювання, послідовно одержимо комплекси кубів K^1 і K^2, де множини міток відповідають перетинанням множин міток термів, які склеювались:

$$K^0 = \begin{cases} \sout{000\{1, 3\}} \\ \text{- - - - - - -} \\ \sout{001\{1, 2, 3\}} \\ \sout{010\{2\}} \\ \sout{100\{1, 2\}} \\ \text{- - - - - - -} \\ \sout{011\{1, 2, 3\}} \\ \sout{101\{3\}} \\ \sout{110\{1, 2\}} \\ \text{- - - - - - -} \\ \sout{111\{3\}} \end{cases}; \quad K^1 = \begin{cases} X00\{1\} \\ \sout{X01\{3\}} \\ X10\{2\} \\ \sout{X11\{3\}} \\ \text{- - - - - - -} \\ 0X1\{1, 2, 3\} \\ 1X0\{1, 2\} \\ \sout{1X1\{3\}} \\ \text{- - - - - - -} \\ 00X\{1, 3\} \\ 01X\{2\} \end{cases}; \quad K^2 = \begin{cases} \sout{XX1\{3\}} \\ XX1\{3\} \end{cases}.$$

Виконавши поглинання, визначимо імпліканти, що відповідають СДНФ системи:

$$Z = \begin{cases} \text{X00}\{1\} \\ \text{X10}\{2\} \\ \text{0X1}\{1, 2, 3\} \\ \text{1X0}\{1, 2\} \\ \text{00X}\{1, 3\} \\ \text{01X}\{2\} \\ \text{XX1}\{3\}. \end{cases}$$

Складаємо таблицю покриття (табл. А1-2.13). В заголовках стовпців таблиці покриття записуємо конституанти окремо для кожної функції. Множина міток кожної імпліканти вказує на належність цієї імпліканти до відповідної функції, що необхідно враховувати під час заповнення таблиці.

Таблиця А1-2.13

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти														
	y_1					y_2					y_3				
	000	001	011	100	110	001	010	011	100	110	000	001	011	101	111
00X{1, 3}	Ⓥ	Ⓥ									Ⓥ	Ⓥ			
X00{1}	∨			∨											
0X1{1, 2, 3}		Ⓥ	Ⓥ			Ⓥ		Ⓥ				∨	∨		
01X{2}							Ⓥ	Ⓥ							
X10{2}							∨			∨					
1X0{1, 2}				Ⓥ	Ⓥ				Ⓥ	Ⓥ					
XX1{3}												Ⓥ	Ⓥ	Ⓥ	Ⓥ

На підставі таблиці покриття знаходимо остаточні форми зображення функцій:

$$y_1 = \overline{x}_3\overline{x}_2 \vee \overline{x}_3 x_1 \vee x_3\overline{x}_1;$$

$$y_2 = \overline{x}_3 x_1 \vee \overline{x}_3 x_2 \vee x_3\overline{x}_1;$$

$$y_3 = \overline{x}_3\overline{x}_2 \vee x_1.$$

Одержано загальні імпліканти $x_3\overline{x_1}$ і $\overline{x_3}x_1$ для функцій y_1, y_2 та $\overline{x_3}\,\overline{x_2}$ для функцій y_1 і y_3, що забезпечує зменшення складності комбінаційної схеми.

Комбінаційна схема, яка реалізує знайдені форми перемикальних функцій наведена на рис. А1-2.5.

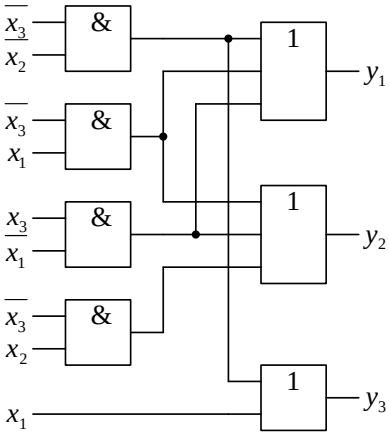


Рис. А1-2.5. Комбінаційна схема

Під час побудови комбінаційних схем, котрі відповідають системам перемикальних функцій, виникає проблема забезпечення заданого коефіцієнта розгалуження по виходу елементів, що реалізують спільні імпліканти.

Якщо число входів, до яких має бути підключений вихід елемента, перевищує коефіцієнт розгалуження, то застосовують способи дублювання елемента чи посилення сигналу.

У першому випадку в схему вводять необхідну кількість таких самих елементів, входи яких підключено паралельно, а в другому випадку вихідний сигнал підсилюється, наприклад, повторювачем з необхідним коефіцієнтом розгалуження.

А1-2.9. Мінімізація частково визначених функцій

У реальних системах можливі випадки, коли не всі набори змінних подаються на входи комбінаційної схеми, тобто існують заборонені вхідні комбінації змінних. На таких наборах перемикальна

функція вважається невизначеною, що дає додаткові можливості для спрощення комбінаційної схеми.

У таблиці істинності невизначені значення функції позначаються довільним символом, відмінним від 0 і 1, наприклад — прочерком.

Мінімізацію частково визначених функцій можна здійснювати будь-яким з наведених вище методів мінімізації. Для забезпечення найефективнішої мінімізації доцільно провести довизначення перемикальної функції на заборонених наборах таким чином, щоб спростити функцію за рахунок більших можливостей склеювання.

Мінімізація частково визначених функцій методом Вейча

Під час використання метода Вейча прочерки розглядають як одиниці в тих випадках, коли це приводить до збільшення розміру прямокутника, що відповідає імпліканті. В протилежному випадку прочерки розглядаються як нулі.

Приклад

Завдання. Виконати мінімізацію перемикальної функції, заданої діаграмою Вейча (рис. А1-2.6).

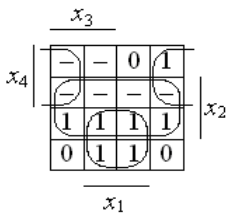


Рис. А1-2.6. Діаграма Вейча

Виконання завдання

Для визначення мінімальної форми запису функції замінюємо певні прочерки одиницями для збільшення розміру прямокутників, тобто отримання імплікант з меншою кількістю букв. Значення функції на наборі 1101, який не увійшов в прямокутники, вважаємо рівним нулю.

Диз'юнкція визначених простих імплікант дає МДНФ до визначеної перемикальної функції

$$y_{\text{МДНФ}} = x_4 \bar{x}_1 \vee x_2 \vee \bar{x}_4 x_1.$$

Мінімізація частково визначених функцій аналітичними методами

Під час використання аналітичних методів, таких як методи Квайна та Квайна—Мак-Класкі, виконуються ті самі етапи мінімізації, що й для повністю визначених функцій.

Особливість мінімізації частково визначених функцій в диз'юнктивних формах булевої алгебри складається з того, що перед початком мінімізації у ДДНФ перемикальної функції вводять усі конституенти заборонених наборів, тобто довизначають функцію на цих наборах одиничним значенням.

Далі виконують стандартні процедури мінімізації, але під час побудови таблиці покриття конституенти заборонених наборів не включають у заголовки стовпців таблиці покриття.

Приклад

Завдання. Виконати мінімізацію частково визначеної перемикальної функції, що задана таблицею істинності (табл. А1-2.14), методом Квайна.

Таблиця А1-2.14

ТАБЛИЦЯ ІСТИННОСТІ

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

Виконання завдання

Без урахування наборів, на яких функція невизначена, маємо

$$y = \overline{x_3}\,\overline{x_2}\,\overline{x_1} \vee \overline{x_3}\,\overline{x_2}x_1 \vee x_3x_2\overline{x_1}.$$

Після доповнення конституентами одиниці, що відповідають забороненим наборам (помічені зірочками), аналітичний запис функції приймає вигляд

$$y = \overline{x_3}\,\overline{x_2}\,\overline{x_1} \vee \overline{x_3}\,\overline{x_2}x_1 \vee \overline{x_3}x_2x_1^{*} \vee x_3\overline{x_2}\,\overline{x_1}^{*} \vee x_3\overline{x_2}x_1^{*} \vee x_3x_2\overline{x_1} \vee x_3x_2x_1^{*}.$$

Для наглядності процесу мінімізації конституенти та імпліканти записуємо стовпцями. У дужках для конституант вказуємо їх номери, а для імплікант — номери конституент, що склеювалися. Терми, що поглинаються, викреслюємо.

~~$\overline{x_3}\overline{x_2}\overline{x_1}(0)$~~	~~$\overline{x_3}\overline{x_2}(0,1)$~~	
~~$\overline{x_3}\overline{x_2}x_1(1)$~~	~~$\overline{x_2}\overline{x_1}(0,4)$~~	
~~$\overline{x_3}x_2\overline{x_1}(3)$~~	~~$\overline{x_3}x_1(1,3)$~~	
~~$x_3\overline{x_2}\overline{x_1}(4)$~~	~~$\overline{x_2}x_1(1,5)$~~	$\overline{x_2}(0,4,1,5)$
~~$\overline{x_3}x_2x_1(5)$~~	~~$x_2x_1(3,7)$~~	$x_1(1,5,3,7)$
~~$x_3\overline{x_2}x_1(6)$~~	~~$x_3\overline{x_2}(4,5)$~~	$x_3(4,5,6,7)$
~~$x_3x_2\overline{x_1}(7)$~~	~~$x_3\overline{x_1}(4,6)$~~	
	~~$x_3x_1(5,7)$~~	
	~~$x_3x_2(6,7)$~~	

Виконавши всі можливі склеювання та поглинання (див. розділ А1-2.2), одержуємо СДНФ заданої перемикальної функції, що довизначена на заборонених наборах:

$$y_{\text{СДНФ}} = \overline{x_2} \vee x_1 \vee x_3.$$

Але таке довизначення перемикальної функції на заборонених наборах можливо не є оптимальним. Для перевірки цього будуємо таблицю покриття (табл. А1-2.15), в яку включаємо тільки ті конституенти одиниці, котрі відповідають наперед визначеним одиничним значенням заданої перемикальної функції.

Таблиця А1-2.15

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти		
	$\overline{x_3}\overline{x_2}\overline{x_1}$	$\overline{x_3}\overline{x_2}x_1$	$x_3x_2\overline{x_1}$
$\overline{x_2}$	Ⓥ	Ⓥ	
x_1		∨	
x_3			Ⓥ

Виходячи з таблиці покриття знаходимо ТДНФ перемикальної функції, яка є її мінімальною МДНФ:

$$y_{\text{МДНФ}} = \overline{x}_2 \vee x_3.$$

Мінімізація систем частково визначених функцій

Під час мінімізації систем частково визначених функцій використовується такий саме підхід, як і під час мінімізації окремих частково визначених функцій. На заборонених наборах значення функцій вважається одиничним, але на етапі вибору покриття конституанти, що відповідають забороненим наборам, не включаються в таблицю покриття.

Приклад

Завдання. Виконати мінімізацію системи частково визначених перемикальних функцій, заданих таблицею істинності (табл. А1-2.16), методом Квайна—Мак-Класкі.

Таблиця А1-2.16

ПАРАМЕТРИ ФУНКЦІЙ

x_3	x_2	x_1	y_1	y_2	y_3
0	0	0	–	1	–
0	0	1	1	1	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	1	–	1
1	0	1	0	–	1
1	1	0	0	0	0
1	1	1	–	1	1

Виконання завдання

Виписуємо 0-куби, що відповідають забороненим наборам і наборам, на яких функції приймають одиничне значення. При цьому позначаємо належність конституент до заданих функцій.

Виконуємо всі можливі склеювання і поглинання (див. розділ А1-2.3):

$$K^0 = \begin{cases} \sout{000\{1,2,3\}} \\ \hline \sout{001\{1,2\}} \\ \sout{100\{1,2,3\}} \\ \hline \sout{011\{1,2,3\}} \\ \sout{101\{2,3\}} \\ \hline \sout{111\{1,2,3\}} \end{cases}; \qquad K^1 = \begin{cases} X00\{1,2,3\} \\ \sout{X01\{2\}} \\ X11\{1,2,3\} \\ \hline 0X1\{1,2\} \\ 1X1\{2,3\} \\ \hline 00X\{1,2\} \\ 10X\{2,3\} \end{cases}; \qquad K^2 = \begin{cases} X0X\{2\} \\ \sout{X0X\{2\}} \\ XX1\{2\} \\ \sout{XX1\{2\}} \end{cases};$$

$$Z = \begin{cases} \text{X00}\{1, 2, 3\} \\ \text{X11}\{1, 2, 3\} \\ \text{0X1}\{1, 2\} \\ \text{1X1}\{2, 3\} \\ \text{00X}\{1, 2\} \\ \text{10X}\{2, 3\} \\ \text{X0X}\{2\} \\ \text{XX1}\{2\} \end{cases}.$$

Таблиця А1-2.17

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти										
	y_1			y_2				y_3			
	001	011	100	000	001	011	111	011	100	101	111
00X{1, 2}	Ⓥ			Ⓥ	Ⓥ						
X00{1, 2, 3}			Ⓥ	∨					Ⓥ		
0X1{1, 2}	∨	∨			∨	∨					
X11{1, 2, 3}		Ⓥ				Ⓥ	Ⓥ	Ⓥ			∨
10X{2, 3}									∨	∨	
1X1{2, 3}							∨			Ⓥ	Ⓥ
X0X{2}				∨	∨						
XX1{2}					∨	∨	∨				

Складаємо таблицю покриття (табл. А1-2.17), на підставі якої знаходимо форми функцій вихідної системи:

$$y_1 = \overline{x}_3\overline{x}_2 \vee x_2x_1 \vee \overline{x}_2\overline{x}_1;$$
$$y_2 = \overline{x}_3\overline{x}_2 \vee x_2x_1;$$
$$y_3 = x_2x_1 \vee \overline{x}_2\overline{x}_1 \vee x_3x_1.$$

Зауважимо, що для визначення мінімального покриття можна використовувати метод Петрика (див. розділ А2-2.7).

А1-2.10. Декомпозиція перемикальних функцій

За збільшення числа аргументів перемикальної функції процес мінімізації значно ускладнюється. Методи декомпозиції дозволяють представити функцію як об'єднання кількох функцій, що залежать від меншого числа аргументів. Це спрощує процес мінімізації функції.

Декомпозиція перемикальних функцій методом Шенона дозволяє вилучити будь-яке число змінних, від яких залежить функція. Для виключення однієї змінної використовується таке розкладання:

$$\begin{aligned} y &= f(x_n, x_{n-1}, \ldots, x_i, \ldots, x_1) = \\ &= \overline{x_i} \cdot f_0(x_n, x_{n-1}, \ldots, x_{i+1}, x_{i-1}, \ldots, x_1) \vee \\ &\vee x_i \cdot f_1(x_n, x_{n-1}, \ldots, x_{i+1}, x_{i-1}, \ldots, x_1). \end{aligned} \qquad \text{(А1-2.4)}$$

Функції f_0 і f_1 називаються залишковими. Кожна з них залежить від $(n - 1)$ змінних. Схема, що реалізує таке перетворення, зображена на рис. А1-2.7. Для залишкових функцій може бути повторно застосоване розкладання (А1-2.4), що дозволяє виключити наступну змінну. Таким чином, може бути виключене будь-яке число змінних.

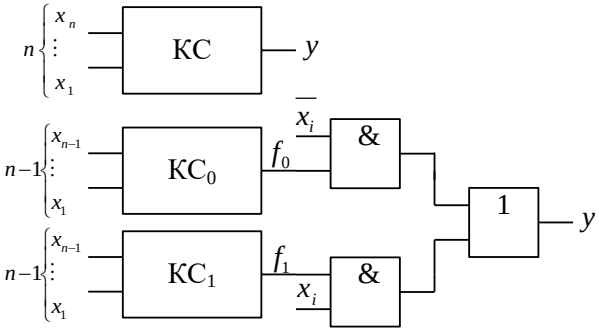


Рис. А1-2.7. Варіанти реалізації функції

Приклад

Завдання. Подати функцію φ, задану таблицею істинності (табл. А1-2.18), у вигляді декомпозиції функцій двох змінних.

Таблиця А1-2.18

ТАБЛИЦЯ ІСТИННОСТІ

x_4	x_3	x_2	x_1	φ	
0	0	0	0	0	f_0
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	1	
0	1	0	0	1	f_1
0	1	0	1	1	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	1	f_2
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	0	f_3
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

Виконання завдання

Можливі шість варіантів вилучення двох змінних. Нехай вилучаються змінні x_4 і x_3. Тоді отримуємо розклад

$$\varphi = \overline{x_4}\,\overline{x_3} f_0(x_2, x_1) \vee \overline{x_4} x_3 f_1(x_2, x_1) \vee x_4 \overline{x_3} f_2(x_2, x_1) \vee x_4 x_3 f_3(x_2, x_1).$$

Після мінімізації залишкових функцій одержимо:

$$f_0 = x_2; \quad f_1 = \overline{x_2}; \quad f_2 = \overline{x_2}\,\overline{x_1} \vee x_2 x_1; \quad f_3 = x_1 \vee x_2.$$

Об’єднання залишкових функцій зручно виконувати за допомогою мультиплексорів (див. розділ А1-4.3)

А1-2.11. Синтез комбінаційних схем

Мінімізацію функцій можна виконувати у різних алгебрах, з урахуванням чи без урахування на етапі мінімізації кількості входів елементів і тощо. У зв’язку з цим існують різні методики побудови комбінаційних схем. Розглянемо один з можливих підходів до побудови комбінаційних схем у елементному базисі трьох алгебр: Буля, Шефера і Пірса.

Синтез комбінаційних схем у заданому елементному базисі можна розбити на три етапи.

На першому етапі виконують мінімізацію перемикальних функцій обраним методом.

На другому етапі функції подають у можливих операторних формах, тобто у вигляді суперпозиції операторів заданих логічних елементів.

На третьому етапі з урахуванням цільової функції проектування обирають операторну форму перемикальної функції і будують комбінаційну схему.

Задана система логічних елементів у загальному випадку може забезпечити реалізацію кілька операторних форм подання перемикальних функцій. Це дозволяє обрати форму, яка найбільше відповідає цільовій функції проектування, тобто забезпечує побудову комбінаційної схеми з урахуванням заданих параметрів.

Розглянемо етапи синтезу комбінаційної схеми на прикладі перемикальної функції трьох аргументів, заданої у ДДНФ:

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

На першому етапі виконаємо мінімізацію заданої функції та її заперечення за допомогою діаграм Вейча (рис А1-2.8).

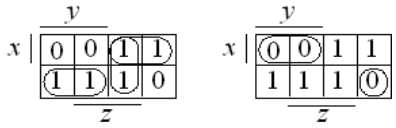


Рис. А1-2.8. Діаграми Вейча

Знаходимо МДНФ функції та її заперечення:

$$f_{\text{МДНФ}} = \bar{x}y \vee x\bar{y} \vee \bar{y}z;$$

$$f_{\overline{\text{МДНФ}}} = \overline{xy \vee \bar{x}yz}.$$

Другий етап починається з одержання восьми нормальних форм у базисі логічних елементів {І, АБО, НЕ, І-НЕ, АБО-НЕ} (див. розділ А1-1.9).

Виходячи із МДНФ функції за допомогою правила де Моргана послідовно одержимо чотири нормальні форми:

$$f(x,y,z) = \bar{x}y \vee x\bar{y} \vee \bar{y}z = \qquad \text{(форма І / АБО)}$$

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

$$= xy \vee x\bar{y} \vee yz =$$

$$= \overline{\overline{xy} \cdot \overline{x}\overline{y} \cdot \overline{yz}} = \quad \text{(форма І-НЕ / І-НЕ)}$$

$$= \overline{(x \vee \bar{y}) \cdot (\bar{x} \vee y) \cdot (y \vee \bar{z})} = \quad \text{(форма АБО / І-НЕ)}$$

$$= \overline{(x \vee \bar{y})} \vee \overline{(\bar{x} \vee y)} \vee \overline{(y \vee \bar{z})} \quad \text{(форма АБО-НЕ / АБО)}$$

На основі МДНФ заперечення заданої функції, отримаємо ще чотири нормальні форми:

$$f(x, y, z) = \overline{xy \vee \bar{x}\bar{y}\bar{z}} = \quad \text{(форма І / АБО-НЕ)}$$

$$= \overline{xy} \cdot \overline{\bar{x}yz} = \quad \text{(форма І-НЕ / І)}$$

$$= (\bar{x} \vee \bar{y}) \cdot (x \vee y \vee z) = \quad \text{(форма АБО / І)}$$

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

$$= \overline{\overline{\bar{x} \vee \bar{y}} \vee \overline{x \vee y \vee z}} \quad \text{(форма АБО-НЕ/АБО-НЕ)}$$

Для побудови схеми можуть використовуватися різні форми, які залежить від елементного базису. Наприклад, за наявності елементів І, АБО, І-НЕ можна вибрати одну з п'яти нормальних форм: І/АБО, І-НЕ/І-НЕ, АБО/І-НЕ, І-НЕ/І, АБО/І, якщо задані елементи І-НЕ, АБО-НЕ, то підходять форми І-НЕ/І-НЕ та АБО-НЕ/АБО-НЕ і т. д.

Якщо число входів логічних елементів достатнє для реалізації нормальної форми, то така форма водночас є операторною. У випадку, коли число входів p логічних елементів менше, ніж потрібно для реалізації нормальної форми, для одержання операторної форми змінні поєднують у групи, що містять не більше p елементів, і використовують співвідношення вигляду:

$$x_1 \cdot x_2 \cdot \ldots \cdot x_m = (x_1 \cdot \ldots \cdot x_g) \cdot \ldots \cdot (x_s \cdot \ldots \cdot x_m);$$

$$x_1 \vee x_2 \vee \ldots \vee x_m = (x_1 \vee \ldots \vee x_g) \vee \ldots \vee (x_s \vee \ldots \vee x_m);$$

$$\overline{x_1 \cdot x_2 \cdot \ldots \cdot x_m} = \overline{\overline{(x_1 \cdot \ldots \cdot x_g)} \cdot \ldots \cdot \overline{(x_s \cdot \ldots \cdot x_m)}};$$

$$\overline{x_1 \vee x_2 \vee \ldots \vee x_m} = \overline{\overline{(x_1 \vee \ldots \vee x_g)} \vee \ldots \vee \overline{(x_s \vee \ldots \vee x_m)}},$$

де $g \le p$ і $m - s + 1 \le p$.

Число груп змінних також не повинне перевищувати p. В іншому випадку зазначені перетворення виконують також стосовно груп змінних. Наведені перетворення дозволяють подати функції в операторній формі з урахуванням числа входів логічних елементів. Комбінаційна схема, отримана за такою операторною формою, може містити більше двох рівнів.

На третьому етапі треба вибрати одну схему з декількох можливих відповідно із заданими параметрами (зазвичай — за складністю та швидкодією).

Для порівняння складності схем доцільно визначати їх складність за Квайном або за числом умовних логічних елементів (див. розділ А1-1.2).

Швидкодія комбінаційних схем залежить від часових параметрів логічних елементів, що характеризують затримку сигналів елементами (див. розділ А1-1.2).

Таким чином, з декількох можливих обирають комбінаційну схему, що краще за інші задовольняє заданим параметрам.

Наприклад, за наявності двовходових елементів 2І-НЕ та 2АБО-НЕ (цифри показують кількість входів елемента) розглянуту функцію можна подати в нормальних формах І-НЕ / І-НЕ та АБО-НЕ / АБО-НЕ таким чином:

$$f(x, y, z) = \overline{\overline{xy} \cdot \overline{xy} \cdot \overline{yz}}; \qquad \text{(І-НЕ / І-НЕ)}$$

$$f(x,\ y,\ z) = \overline{\overline{x} \vee \overline{y} \vee \overline{x \vee y \vee z}}. \quad \text{(АБО-НЕ / АБО-НЕ)}$$

Виходячи з цього отримаємо операторні форми:

$$f(x, y, z) = \overline{((xy) \cdot (xy)) \cdot (yz)};$$

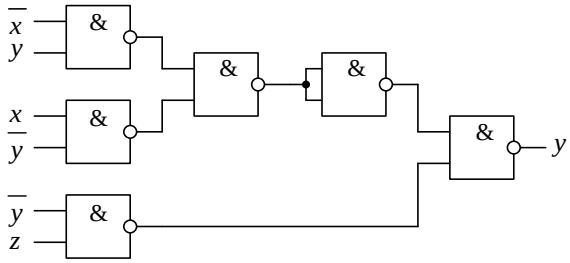
$$f(x, y, z) = \overline{(\overline{x} \vee \overline{y}) \vee (((x \vee y)) \vee z)}.$$

За отриманими операторними формами можна побудувати комбінаційні схеми на елементах 2І-НЕ та 2АБО-НЕ (рис. А1-2.9).

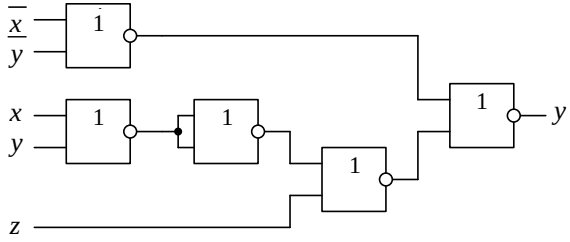
У кожній схемі максимальна затримка сигналів визначається ланцюжком із чотирьох елементів, тобто $T = 4t$, де t –затримка сигнала одним відповідним елементом. Якщо елементи І-НЕ мають менший час затримки сигналів, ніж елементи АБО-НЕ, то комбінаційна схема на рис. А1-2.9, *а* має більшу швидкодію, але вона програє комбінаційній схемі на рис. А1-2.9, *б* за складністю. Складність за Квайном комбінаційної схеми, побудованої на логічних елементах 2І-НЕ, дорівнює К = 12, а схеми на логічних елементах 2АБО-НЕ – К = 10.

Комбінаційні схеми за перехідного процесу можуть формувати короткочасні вихідні сигнали, не передбачені таблицею істинності. Якщо на наборах α та β в таблиці істинності задані однакові значен-

ня функції, то за зміни вказаних наборів на короткий час може виникати протилежний за значенням сигнал. Це обумовлено наявністю в схемі шляхів з різною тривалістю проходження сигналів від входів схеми до виходів. Наприклад, у схемах на рис А1-2.9 від входів до виходів сигнал може проходити два або чотири елемента. Ця проблема відома як проблема «гонок», яка обумовлює так званий «ризик збою» в комбінаційних схемах.



а

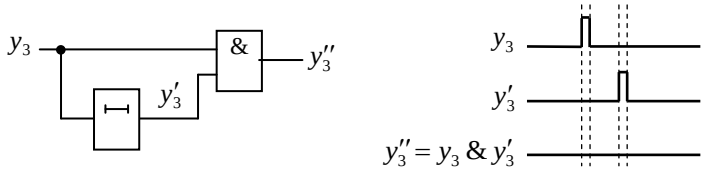


б

Рис. А1-2.9. Комбінаційні схеми:
а — на логічних елементах 2І-НЕ; *б* — на логічних елементах 2АБО-НЕ

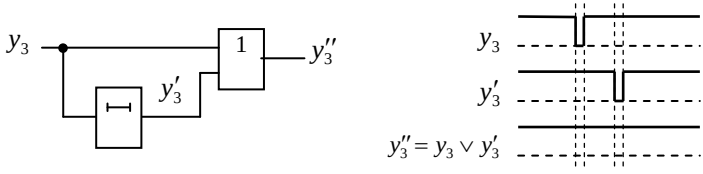
Для усунення збою використовується передусім синхронний принцип передачі сигналів від однієї схеми в іншу. При цьому інформаційний сигнал тактується синхросигналом, який забезпечує прийом інформаційного сигналу в подальших пристроях після закінчення перехідних процесів у схемі, що формує цей сигнал.

Другий спосіб пов’язаний із встановленням фільтрів для вихідних сигналів на тих виходах комбінаційної схеми, де можуть виникати помилкові сигнал (рис. А1-2.10 і А1-2.11).



а *б*

Рис. А1-2.10. Фільтр для усунення короткочасного одиничного вихідного сигналу:
а — логічна схема фільтру; *б* — часова діаграма роботи фільтру



а *б*

Рис. А1-2.11. Фільтр для усунення короткочасного нульового вихідного сигналу:
а — логічна схема фільтру; *б* — часова діаграма роботи фільтру

Короткочасні сигнали затримуються на елементі затримки перед поданням на один з входів вихідного елемента фільтра, наприклад, на повторювачі. За рахунок такого рознесення в часі короткочасних сигналів помилкові сигнали на виході фільтра відсутні.

А1-2.12. Аналіз комбінаційних схем

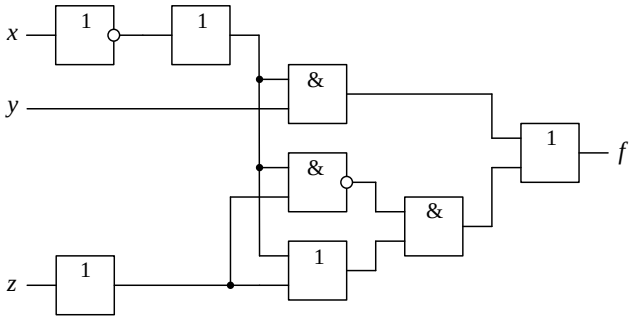
Задача аналізу комбінаційних схем є зворотною відносно задачі синтезу комбінаційних схем. Вихідними даними у цьому випадку є комбінаційна схема. Необхідно описати схему аналітичними формами функцій, що реалізує схема, у заданому функціональному базисі.

На першому етапі аналізу необхідно вилучити зі схеми елементи, що не несуть функціонального навантаження. Прикладом таких елементів є повторювачі, що використовуються як підсилювачі сигналів.

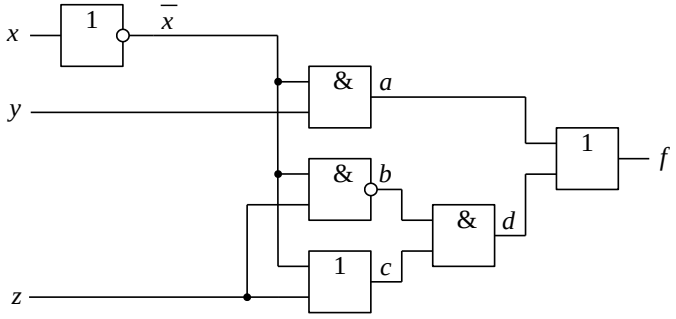
На другому етапі складають аналітичну форму функції, яку реалізує задана схема. Для цього виходи логічних елементів позначають різними символами і записують формулу у вигляді поетапної суперпозиції згідно з топологією схеми.

Необхідну аналітичну форму можна одержати після застосування операцій перетворення логічної інформації в заданій алгебрі.

Розглянемо процес аналізу комбінаційної схеми на прикладі комбінаційної схеми, зображеної на рис. А1-2.12, *а*. Будемо вважати, що в результаті перемикальну функцію необхідно подати у вигляді ДДНФ.



а



б

Рис. А1-2.12. Комбінаційні схеми:
а — вихідна комбінаційна схема перемикальної функції *f*; *б* — комбінаційна схема без повторювачів з позначеними виходами логічних елементів

Вилучаємо із схеми повторювач і позначаємо виходи всіх логічних елементів різними буквами (рис. А1-2.12, *б*).

Далі записуємо оператори для всіх елементів:

$$f = d \vee a;$$
$$d = b \cdot c;$$
$$a = \bar{x} \cdot y;$$

$$b = \overline{\bar{x} \cdot z};$$

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

Виконуємо перетворення, користуючись законами і аксіомами булевої алгебри:

$$f = d \vee a = bc \vee \bar{x}y = \overline{\bar{x}z}(\bar{x} \vee z) \vee \bar{x}y = (x \vee \bar{z})(\bar{x} \vee z) \vee \bar{x}y =$$

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

$$= xz(y \vee \bar{y}) \vee \bar{z}\bar{x}(y \vee \bar{y}) \vee \bar{x}y(\bar{z} \vee z) = xyz \vee x\bar{y}z \vee \bar{x}y\bar{z} \vee \bar{x}\bar{y}\bar{z} \vee \bar{x}yz.$$

У результаті одержали ДДНФ перемикальної функції:

$$f_{\text{ДДНФ}} = xyz \vee x\bar{y}z \vee \bar{x}y\bar{z} \vee \bar{x}\bar{y}\bar{z} \vee \bar{x}yz.$$

А1-3. Цифрові автомати із пам'яттю

А1-3.1. Абстрактні автомати

Цифровий автомат із пам'яттю — це послідовнісна схема, що здійснює перетворення двійкових перемінних. Такі схеми мають два і більше станів на відміну від комбінаційних схем, котрі мають тільки один стан (див. розділ А1-1.2). Цифрові автомати можна розглядати на абстрактному і структурному рівнях.

На *абстрактному рівні* розглядається взаємодія автомата з зовнішнім середовищем, при цьому не розглядаються його внутрішня організація. На *структурному рівні*, окрім взаємодії з зовнішнім середовищем, розглядається внутрішня організація автомата. Відносно рівня опису відрізняють абстрактні й структурні автомати.

У загальному виді абстрактний автомат може бути заданий сукупністю п'яти об'єктів:

$$A = \langle X, Y, Z, \delta, \lambda \rangle,$$

де $X = \{ X_i \mid i = \overline{1, n} \}$ — вхідний алфавіт;

$Y = \{ Y_i \mid i = \overline{1, k} \}$ — вихідний алфавіт;

$Z = \{ Z_i \mid i = \overline{1, s} \}$ — алфавіт внутрішніх станів;

δ — функція переходів;

λ — функція виходів.

Якщо множини *X*, *Y* і *Z* є кінцевими, то автомат називають кінцевим. Такий автомат перетворює вхідний алфавіт на вихідний за кінцеве число тактів. На абстрактному рівні алфавіти *X*, *Y*, *Z* складаються з абстрактних букв, а на структурному рівні кожній букві відповідає вектор (сукупність) структурних сигналів.

Функція переходів визначає спосіб переходу автомата із одного стану в інший, а *функція виходів* — формування вихідних сигналів.

Існують два основні різновиди автоматів, що відрізняються способом формування вихідних сигналів. Закон функціонування абстрактних автоматів може бути заданий функціями переходів і виходів.

Для автомата Мура зазначені функції можна записати у вигляді

$$\begin{cases} Z_i^{t+1} = \delta(X_j^t, Z_s^t); \\ Y_i^t = \lambda(Z_s^t), \end{cases}$$

а для автомата Мілі:

$$\begin{cases} Z_i^{t+1} = \delta(X_j^t, Z_s^t); \\ Y_i^t = \lambda(X_j^t, Z_s^t), \end{cases}$$

де верхні індекси відповідають моментам автоматного часу $t = 1, 2, 3, \ldots$

Як видно з наведених функцій, вихідні сигнали автомата Мура залежать тільки від його стану, а автомата Мілі — як від стану, так і від діючих вхідних сигналів. Для кожного типу автоматів перехід до іншого стану визначається попереднім станом і діючими вхідними сигналами.

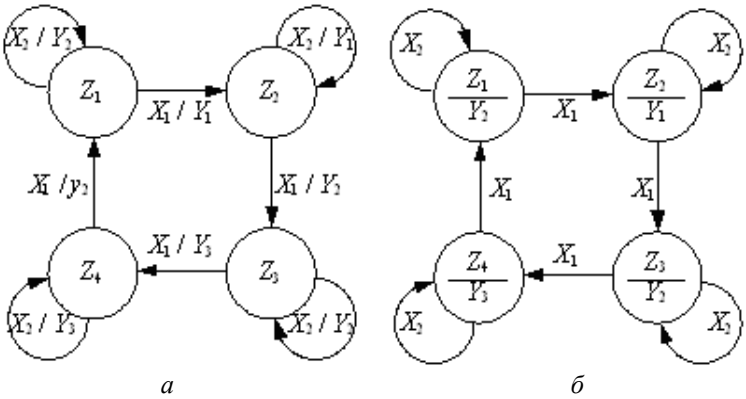
Функції переходів і виходів абстрактного автомата можуть бути задани за допомогою графа або таблиць переходів і виходів.

Граф автомату — це орієнтований граф, вершини якого відповідають станам автомату, а дуги — переходам між ними. Приклади графів автоматів Мілі і Мура зображені на рис. А1-3.1.

Дві вершини графу автомату Z_j і Z_i поєднуються дугою (Z_j, Z_i), якщо в автоматі можливий перехід від стану Z_j до стану Z_i. Дузі приписують букву вхідного алфавіту (умову переходу, сукупність вхідних сигналів).

У графі автомата Мура вихідні сигнали (букви вихідного алфавіту Y_k) записуються у вершинах графу, тому що вони визначаються тільки станом автомата і не залежать від вхідних сигналів. У графі автомата Мілі вихідні сигнали приписуються дугам, бо вони залежать від стану і вхідних сигналів.

За великої кількості станів автомата граф стає громіздким і втрачає наочність. У цьому випадку доцільно використовувати для формального опису автомата таблиці переходів і виходів.



а *б*

Рис. А1-3.1. Графи автоматів:
а — Мілі, *б* — Мура

У *таблицях переходів і виходів* заголовками рядків є букви вхідного алфавіту (вхідні сигнали), а стовпців — стани автомату. Графу автомата Мілі (рис. А1-3.1, *а*) відповідають табл. А1-3.1 і А1-3.2, а приклад табличного опису автомата Мура, граф якого подано на рис. А1-3.1, *б*, наведено у табл. А1-3.3.

Таблиця А1-3.1

ТАБЛИЦЯ ВИХОДІВ АВТОМАТА МІЛІ

λ	Z_1	Z_2	Z_3	Z_4
X_1	Y_1	Y_2	Y_3	Y_2
X_2	Y_2	Y_1	Y_2	Y_3

Таблиця А1-3.2

ТАБЛИЦЯ ПЕРЕХОДІВ АВТОМАТА МІЛІ

δ	Z_1	Z_2	Z_3	Z_4
X_1	Z_2	Z_3	Z_4	Z_1
X_2	Z_1	Z_2	Z_3	Z_4

Таблиця 1-3.3

ВІДМІЧЕНА ТАБЛИЦЯ ПЕРЕХОДІВ АВТОМАТА МУРА

(λ, δ)	Y_1	Y_2	Y_3	Y_4
	Z_1	Z_2	Z_3	Z_4
X_1	Z_2	Z_3	Z_4	Z_1
X_2	Z_1	Z_2	Z_3	Z_4

Для автомата Мілі у клітинках таблиці переходів на перетинанні рядків і стовпців вказується стан, у який переходить автомат, а в таблиці виходів — букву вихідного алфавіту (вихідний сигнал).

Оскільки в автоматі Мура вихідний сигнал залежить тільки від стану автомату, то цей автомат можна описати однією *відміченою таблицею переходів*, де у заголовку кожного стовпця, окрім стану автомату Z_i, записується вихідний сигнал Y_i, що відповідає цьому стану.

Таким чином, між графом автомата та таблицями переходів і виходів існує однозначна відповідність, тобто від графа можна перейти до таблиць і навпаки.

А1-3.2. Абстрактний синтез автоматів з пам'яттю

Синтез цифрових автоматів складається з етапів абстрактного та структурного синтезу.

Результатом абстрактного синтезу автомата є його формальний опис у вигляді п'ятірки $A = \langle X, Y, Z, \delta, \lambda \rangle$, а структурний синтез дозволяє одержати логічну схему автомата в заданому елементному базисі.

Абстрактний синтез охоплює такі етапи:

— формування вхідного та вихідного алфавітів;

— складання алгоритму функціонування автомата;

— формування алфавіту внутрішніх станів автомата;

— визначення функцій переходів і виходів за допомогою графа або відповідних таблиць.

Вхідний алфавіт формується на основі вивчення зовнішніх для автомата сигналів, яки можуть розглядати як логічні умови, що змінюють режим функціонування автомата. Різні комбінації вхідних сигналів розглядаються як букви X_i вхідного алфавіту.

Автомат, як правило, забезпечує управління деяким операційним пристроєм, що перетворює цифрову інформацію шляхом виконання певних операторів. Кожному оператору відповідає певний набір управляючих сигналів, який на абстрактному рівні розглядається як певна буква Y_i вихідного алфавіту управляючого автомата.

Отримання алгоритму функціонування автомата не є формалізованим процесом. Для цього необхідно чітко представляти спосіб перетворення інформації.

Для опису алгоритмів функціонування цифрового автомату зручно використовувати *графічні схеми алгоритмів* (ГСА) та *логічні* (*лінійні*) *схеми алгоритмів* (ЛСА).

ГСА складається з вершин чотирьох типів (рис. А1-3.2). Алгоритм починається з вершини «Початок» і закінчується вершиною «Кінець». Букви Y_i вихідного алфавіту (вихідні сигнали, оператори) записують в операторні вершини, а букви X_i вхідного алгоритму (вхідні сигнали, логічні умови) розміщують у логічні (умовні) вершини.

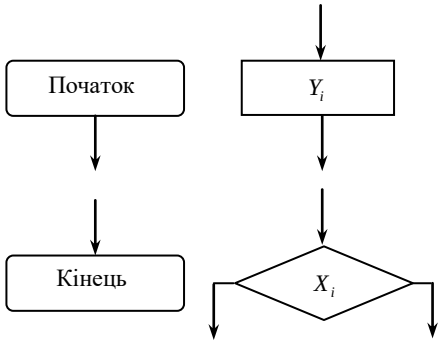
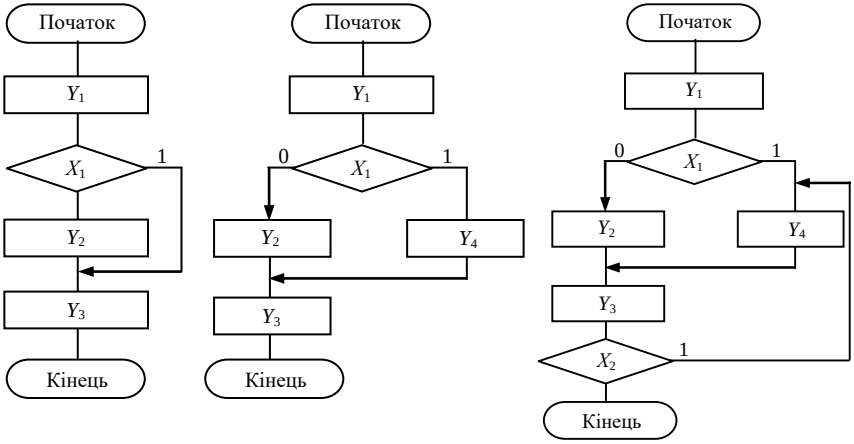


Рис. А1-3.2. Умовні позначення вершин у ГСА

Компактний запис алгоритму, хоча і з втратою наочності, забезпечує ЛСА, яка являє собою кінцевий рядок символів. Рядок починається символом П (початок) і закінчується символом К (кінець). Усередині рядка записують букви вхідного та вихідного алфавітів (X_i і Y_i), між якими можуть розміщуватись стрілки з верхніми індексами ($\overset{i}{\uparrow}$ і $\overset{i}{\downarrow}$). Стрілки з однаковими індексами, одна з яких спрямована вгору, а друга вниз, створюють нерозривну пару, що відповідає умовному (якщо стрілка вгору розташовані після X_i) або безумовному переходу (якщо така стрілка розташована після Y_i). За виконання умови перехід на наступний оператор відбувається за стрілкою. Якщо умова не виконується, то наступним є оператор, розташований праворуч від символу відповідної умови. Безумовний перехід виконується завжди за стрілками.

Приклади ГСА та відповідних їм ЛСА показані на рис. А1-3.3.



$$\text{П}\ \ Y_1 X_1 \overset{1}{\uparrow} Y_2 \overset{1}{\downarrow} Y_3\ \ \text{К}$$

$$\text{П}\ \ Y_1 X_1 \overset{1}{\uparrow} Y_2 \overset{2}{\uparrow} \overset{1}{\downarrow} Y_4 \overset{2}{\downarrow} Y_3\ \ \text{К}$$

$$\text{П}\ \ Y_1 X_1 \overset{1}{\uparrow} Y_2 \overset{2}{\uparrow} \overset{1}{\downarrow} \overset{3}{\downarrow} Y_4 \overset{2}{\downarrow} Y_3 X_1 \overset{3}{\uparrow}\ \ \text{К}$$

Рис. А1-3.3. Приклади ГСА та ЛСА

Після створення ГСА процес побудови графа автомата і визначення алфавіту внутрішніх станів можна формалізувати. У зв'язку з цим ГСА, як і граф, можна вважати формальним описом абстрактного автомата.

Для побудови графа автомата виконується розмітка станів автомата на ГСА. Способи розмітки автоматів Мура і Мілі відрізняються.

Розмітка станів автомата Мілі здійснюється таким чином:

— символом Z_1 позначається вхід наступної за початковою вершини (логічної або операторної), а також вхід кінцевої вершини (якщо автомат реалізує циклічний процес);

— входи всіх вершин, наступних за операторними, позначаються символами Z_i ($i = \overline{2,n}$).

Розмітка станів автомату Мура здійснюється за такими правилами:

— символом Z_1 позначаються початкова і кінцева вершини;

— всі операторні вершини позначаються різними символами Z_i ($i = \overline{2,n}$).

У результаті розмітки ГСА визначається алфавіт внутрішніх станів.

Побудова графа автомата виконується на основі розміченої ГСА. Число вершин графа дорівнює числу станів автомата. Кожному переходу автомата з одного стану в інший відповідає дуга графа. Дузі приписується умова, за якої здійснюється перехід автомата з одного стану в інший. Для автомата Мілі вихідні сигнали приписують дугам, а для автомата Мура — розміщують у вершинах графа. Приклади графів автоматів в абстрактних термінах наведені на рис. А1-3.1.

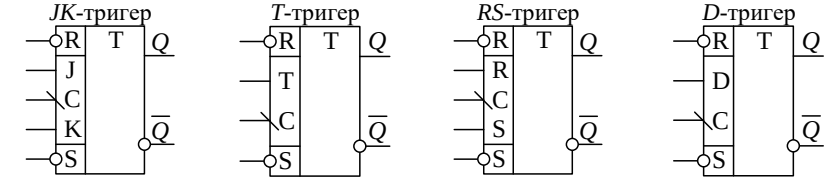
А1-3.3. Структурний синтез автомата методом композиції тригерів

Результати абстрактного синтезу автомата можуть розглядатися який вихідні дані для структурного синтезу. Якщо абстрактний автомат є математичною моделлю пристрою, що проектується, то структурний автомат визначає логічну організацію пристрою з урахуванням заданого елементного базису та подання вхідних і вихідних сигналів.

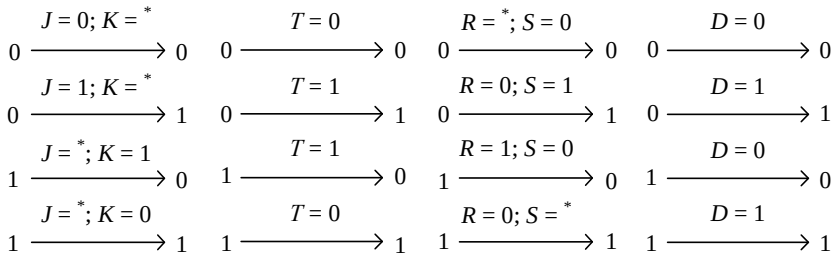
Слід зауважити, що під час проектування реальних систем на етапах структурного синтезу може виникнути необхідність корегування результатів абстрактного синтезу (наприклад, ГСА та графа автомата). Це може зумовлюватися необхідністю усунення ризиків збою, врахуванням тривалості структурних сигналів, елементної бази тощо. Таке корегування може привести до введення нових вершин і дуг в граф автомата. Таким чином, множина станів абстрактного автомата $\{Z_1, Z_2, ..., Z_n\}$ може відрізнятися від множини станів структурного автомата $\{z_1, z_2, ..., z_k\}$.

Один з підходів теорії цифрових автоматів до побудови структурних автоматів полягає в представленні будь-якого автомата у вигляді композиції елементарних автоматів Мура, що належать до скінченного числа типів таких автоматів і мають назву тригерів.

Існують чотири основні функціональні типи тригерів: *RS*-тригери, *JK*-тригери, *D*-тригери і *T*-тригери. Умовні графічні позначення синхронних тригерів і системи підграфів переходів, що пояснюють спосіб зміни стану тригерів, зображені на рис. А1-3.4. Тригери мають тільки два стани: нульовий стан — за $Q = 0$ і $\overline{Q} = 1$, та одиничний стан — за $Q = 1$ і $\overline{Q} = 0$. Перехід тригерів з одного стану в інший визначається інформаційними сигналами, а момент переходу — перепадом синхросигналу C (в даному випадку перепад з 1 в 0). Асинхронні входи тригерів R і S дозволяють встановлювати початковий стан тригерів. Внутрішня організація тригерів докладніше розглядається в розділі А1-4.7.



а



б

Рис. А1-3.4. Синхронні тригери:
а — умовні графічні означення; *б* — система підграфів переходів

Узагальнена структура автомата, побудованого за методом композиції елементарних автоматів (тригерів), показана на рис. А1-3.5.

Автомат містить комбінаційну схему (КС) і пам'ять (П), що складається з тригерів T_i. Входами КС є виходи Q_1, ..., Q_m тригерів і вхід-

ні структурні сигнали (логічні умови) x_1, ..., x_k, що формуються в операційному пристрої. КС виробляє структурні управляючі сигнали y_1, ..., y_p для операційного пристрою і функції збудження тригерів q_1, ..., q_m, що визначають перехід автомата з одного стану в інший. Кожному з множини станів $\{z_1, ..., z_m\}$ відповідає визначений набір значень Q_i. В синхронних автоматах перехід від одного стану до іншого відбувається під дією синхросигналів C, що виробляє генератор G.

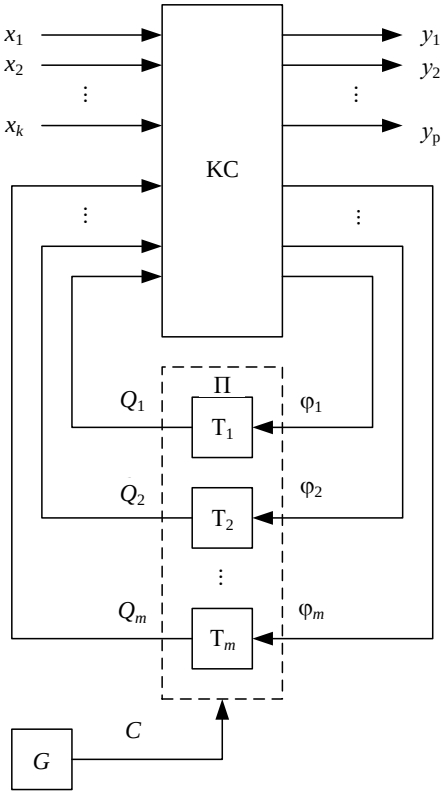


Рис. А1-3.5. Структурна схема цифрового автомата

Синтез синхронного автомата включає такі етапи:

1) складання списку структурних вихідних сигналів, що відповідають кожній букві вихідного алфавіту абстрактного автомата;

2) визначення тривалості кожного вихідного структурного сигналу (в числі тактів) і періоду тактуючих сигналів автомата;

3) одержання закодованої ГСА структурного автомата, тобто корегування ГСА абстрактного автомата, якщо сигнали мають різну тривалість;

4) розмітка ГСА структурного автомата;

5) складання графа структурного автомата;

6) кодування станів структурного автомата;

7) складання структурної таблиці автомата;

8) одержання МДНФ функцій збудження тригерів і керуючих сигналів;

9) представлення функцій збудження тригерів і керуючих сигналів в операторній формі;

10) побудова схеми керуючого автомата.

Якщо всі вихідні сигнали структурного автомата мають однакову тривалість, то після п. 1 на графі абстрактного автомата замінюють абстрактні терміни структурними та переходять до виконання п. 6.

Розглянемо *приклад синтезу автомата Мілі*. Будемо вважати, що управляючий автомат (УА) керує операційним автоматом (ОА) (рис. А1-3.6), який реалізує певну систему операцій. Кожній операції відповідає послідовність операторів (мікрооперацій), а кожному оператору відповідає сукупність структурних сигналів автомата із множини $\{y_i, y_2, ..., y_6\}$.

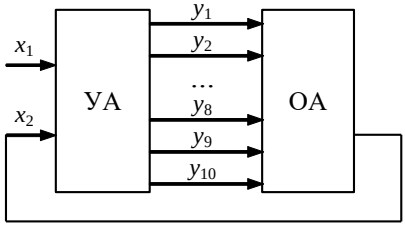


Рис. А1-3.6. Загальна структура пристрою

Нехай на етапі абстрактного синтезу одержана ГСА в абстрактних термінах (рис. А1-3.7).

Відповідність абстрактних та структурних сигналів визначається особливостями керування операційним пристроєм. Нехай за рахунок аналізу пристрою одержана відповідність зазначених сигналів (табл. А1-3.4).

На практиці необхідна тривалість керуючих сигналів визначається з урахуванням затримок в елементах операційного пристрою. Виходячи з цього обирається період t тактуючих сигналів. При цьому

величина t має бути не менше часу переключення автомата з одного стану в інший.

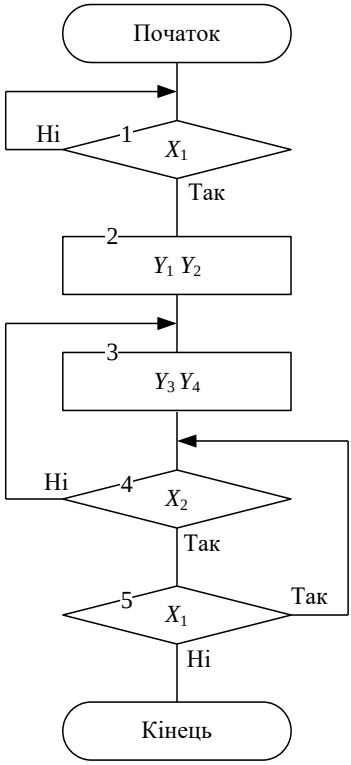


Рис. А1-3.7. ГСА абстрактного автомата

Як видно з табл. А1-3.4, керуючі сигнали y_3 і y_4 повинні мати тривалість $2t$, а інші — t. Подвійна тривалість сигналів може бути забезпечена, наприклад, введенням у ГСА додаткових операторних вершин з керуючими сигналами, тривалість яких перевищує t.

Таблиця А1-3.4

ТАБЛИЦЯ КЕРУЮЧИХ СИГНАЛІВ

Оператори	Керуючі сигнали	Тривалість керуючих сигналів
Y_1	y_1	t
Y_2	y_2	t
Y_3	y_3, y_4, y_5	$2t$, $2t$, t

Y_4	y_6	t

Для одержання ГСА автомата в структурних термінах (структурної ГСА) складаємо таблицю позначень логічних умов (табл. А1-3.5) і замінюємо абстрактні терміни в ГСА (рис. А1-3.7) на структурні. Оскільки управляючі сигнали y_3 та y_4 мають тривалість $2t$, то на структурній ГСА (рис. А1-3.8) вводимо додаткову операторну вершину 4, що відповідає цим управляючим сигналам.

Таблиця А1-3.5

ТАБЛИЦЯ ЛОГІЧНИХ УМОВ

Логічні умови	Позначення логічних умов
X_1	x_1
X_2	x_2

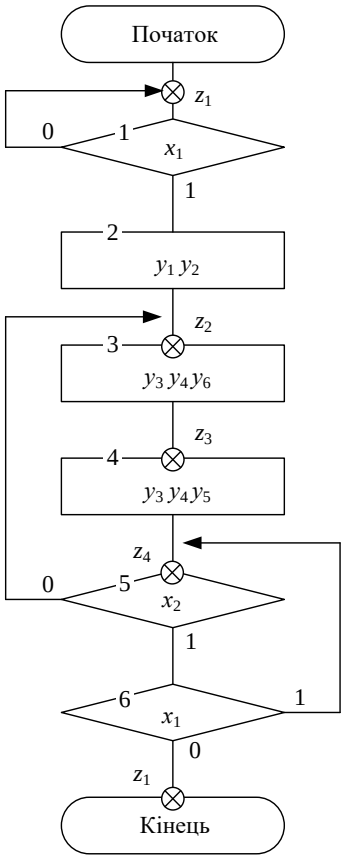


Рис. А1-3.8. ГСА структурного автомата Мілі

На рис. А1-3.8 ГСА відзначена чотирма різними станами (z_1, z_2, z_3, z_4). Таку саму кількість вершин має граф автомата Мілі, зображений на рис. А1-3.9.

Кожному переходові автомата з одного стану в інший відповідає дуга графа. Дузі приписується логічна умова, за якої здійснюється перехід автомата з одного стану в інший, а також набір управляючих сигналів, що відповідають даному переходові. Якщо перехід безумовний, то замість умови проставляють «—».

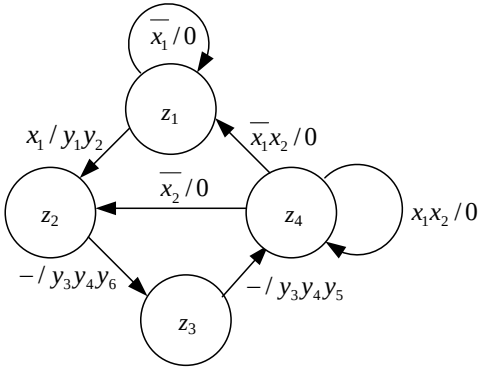


Рис. А1-3.9. Граф автомата Мілі

Кількість тригерів, необхідних для організації пам'яті автомата, визначається із співвідношення $m >]\log_2 M[$, де M — число станів автомата. Кожному станові z_i має відповідати одна визначена комбінація значень Q_1, ..., Q_m.

Для прикладу, що розглядається, вибираємо коди станів відповідно до табл. А1-3.6.

Таблиця А1-3.6

ТАБЛИЦЯ КОДУВАННЯ СТАНІВ АВТОМАТА МІЛІ

Стан	Код стану	
	Q_1	Q_2
z_1	0	0
z_2	0	1
z_3	1	1
z_4	1	0

Для організації пам'яті використовуватимемо *JK*-тригери, а для побудови комбінаційних схем — елементи булевого базису.

Відзначимо, що спосіб кодування станів впливає на правильність формування управляючих сигналів і складність автомата. (Це питання докладніше розглядатиметься в розділі А1-3.4).

Структурна таблиця автомата складається за його графом. Кожен рядок (табл. А1-3.7) відповідає визначеному переходові автомата з одного стану в інший. У рядку записують вихідний стан, стан переходу, коди цих станів, значення логічних умов, що забезпечують перехід, необхідні значення управляючих сигналів і функцій збудження тригерів. Значення функції збудження тригерів визначають згідно з таблицею переходів тригера відповідного типу. В кожному рядку для *i*-го тригера розглядаються переходи $Q_i^S \to Q_i^{S+1}$. Довільні значення управляючих сигналів (0 або 1) позначаються в структурній таблиці символом «–». Довільні значення управляючих сигналів визначаються особливостями операційного пристрою. В даному випадку як приклад такі значення прийняті для сигналів y_1, y_2, y_3.

Таблиця А1-3.7

СТРУКТУРНА ТАБЛИЦЯ АВТОМАТА МІЛІ

ПС*	Код ПС		СП*	Код СП		Логічна умова		Керуючі сигнали						Функції збудження тригерів			
	Q_1^S	Q_2^S		Q_1^{S+1}	Q_2^{S+1}	x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	J_1	K_1	J_2	K_2
z_1	0	0	z_1	0	0	0	–	0	–	0	0	0	0	0	–	0	–
z_1	0	0	z_2	0	1	1	–	1	1	0	0	0	0	0	–	1	–
z_2	0	1	z_3	1	1	–	–	0	0	1	1	0	1	1	–	–	0
z_3	1	1	z_4	1	0	–	–	–	0	1	1	1	0	–	0	–	1
z_4	1	0	z_4	1	1	1	1	0	0	0	0	0	0	–	0	0	–
z_4	1	0	z_2	0	1	–	0	0	0	–	0	0	0	–	1	1	–
z_4	1	0	z_1	0	0	0	1	0	0	–	0	0	0	–	1	0	–
z_1	0	0	z_1	0	0	0	–	0	–	0	0	0	0	0	–	0	–

* ПС — *початковий стан*, СП — *стан переходу*

На підставі структурної таблиці автомата визначаємо МДНФ функцій збудження тригерів і функцій управляючих сигналів. Аргументами функцій J_i, K_i та y_i є значення x_1, x_2 та Q_1, Q_2 у початковому стані.

Для отримання МДНФ функцій використаємо діаграми Вейча (рис. А1-3.10).

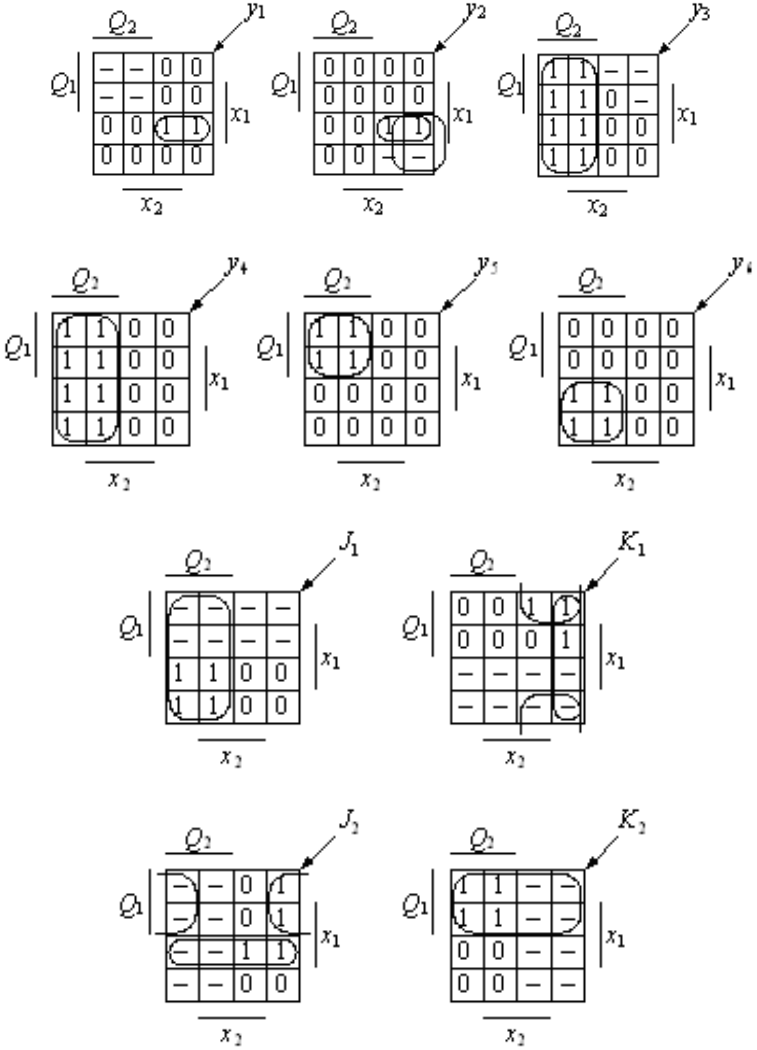


Рис. А1-3.10. Діаграми Вейча керуючих сигналів і функцій збудження тригерів

Після мінімізації одержимо функції:

$$y_1 = y_2 = \overline{Q_1}\,\overline{Q_2}x_1;$$
$$y_3 = y_4 = J_1 = Q_2;$$
$$y_5 = Q_1Q_2;$$
$$y_6 = \overline{Q_1}Q_2;$$
$$K_1 = \overline{Q_2}\,\overline{x_2} \vee \overline{Q_2}\,\overline{x_1} = \overline{Q_2}(\overline{x_2} \vee \overline{x_1});$$
$$J_2 = Q_1\overline{x_2} \vee \overline{Q_1}x_1;$$
$$K_2 = Q_1.$$

Функціональна схема автомата зображена на рис. А1-3.11, де УПС — установлення початкового стану (активний рівень сигналу низький), С — тактуючі сигнали.

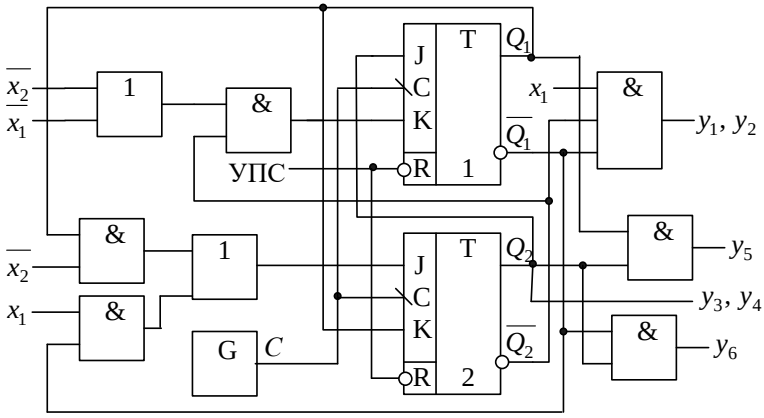


Рис. А1-3.11. Функціональна схема автомата Мілі

Розглянемо *приклад синтезу автомата Мура* на *D*-тригерах та елементах булевого базису.

Будемо вважати, що одержано таку саму ГСА структурного автомата, як і у попередньому прикладі (рис. А1-3.7). Після розмітки станів за правилами автомата Мура структурна ГСА приймає вигляд, наведений на рис. А1-3.12, а граф автомата Мура — як на рис. А1-3.13.

Кодування станів автомата Мура виконаємо аналогічно до розглянутого у прикладі автомату Мілі (табл. А1-3.7).

На основі графа складаємо структурну таблицю автомата Мура (табл. А1-3.8).

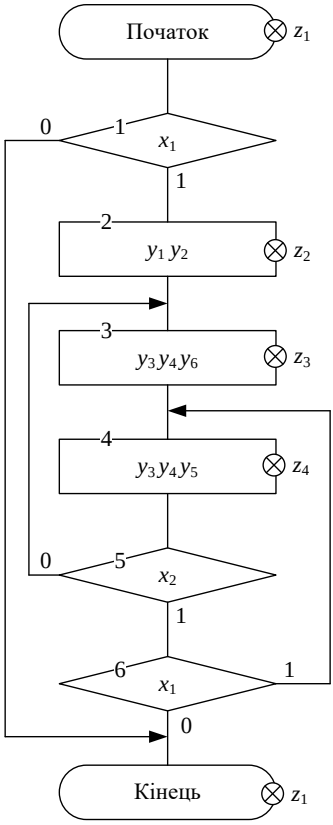


Рис. А1-3.12. ГСА структурного автомата Мура

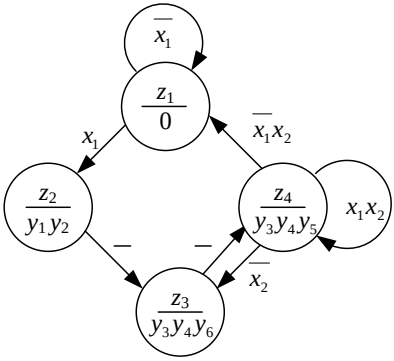


Рис. А1-3.13. Граф автомата Мура

Таблиця А1-3.8

СТРУКТУРНА ТАБЛИЦЯ АВТОМАТА МУРА

ПС*	Код ПС		СП*	Код СП		Логічна умова		Керуючі сигнали						Функції збудження тригерів	
	Q_1^S	Q_2^S		Q_1^{S+1}	Q_2^{S+1}	x_1	x_2	y_1	y_2	y_3	y_4	y_5	y_6	D_1	D_2
z_1	0	0	z_1	0	0	0	–	0	–	0	–	0	0	0	0
z_1	0	0	z_2	0	1	1	–	0	–	0	–	0	0	0	1
z_2	0	1	z_3	1	1	–	–	1	1	0	0	0	0	1	1
z_3	1	1	z_4	1	0	–	–	–	0	1	1	0	1	1	0
z_4	1	0	z_4	1	0	1	1	0	0	1	1	1	0	1	0
z_4	1	0	z_3	1	1	–	0	0	0	1	1	1	0	1	1
z_4	1	0	z_1	0	0	0	1	0	0	1	1	1	0	0	0

* ПС — *початковий стан,* СП — *стан переходу*

Вихідні сигнали в автоматі Мура залежать тільки від початкового стану Q_1 і Q_2. Функції збудження тригерів, як і для автомата Мілі, залежать від початкового стану автомата і вхідних сигналів.

На підставі структурної таблиці автомата визначаємо МДНФ функцій збудження тригерів і функцій управляючих сигналів, для чого використовуємо діаграми Вейча (рис. А1-3.14).

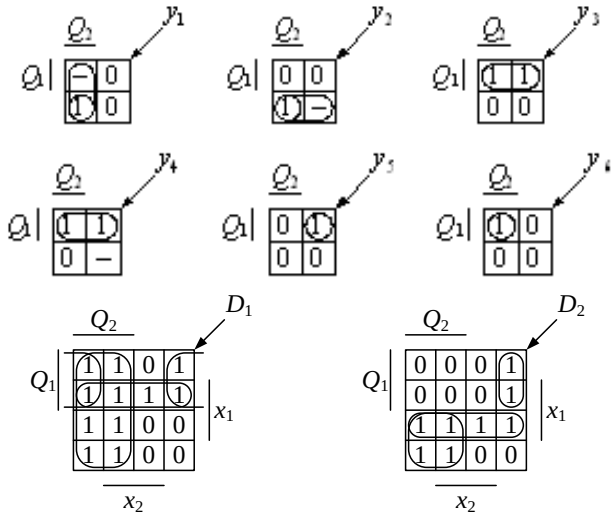


Рис. А1-3.14. Діаграми Вейча функцій керуючих сигналів і функцій збудження тригерів

Після мінімізації одержимо функції:

$$y_1 = Q_2;$$
$$y_2 = \overline{Q_1};$$
$$y_3 = y_4 = Q_1;$$
$$y_5 = Q_1\overline{Q_2};$$
$$y_6 = Q_1Q_2;$$
$$D_1 = Q_2 \vee Q_1x_1 \vee Q_1\overline{x_2} = Q_2 \vee Q_1(x_1 \vee \overline{x_2});$$
$$D_2 = Q_1\overline{Q_2}\,\overline{x_2} \vee \overline{Q_1}x_1 \vee \overline{Q_1}Q_2 = Q_1\overline{Q_2}\,\overline{x_2} \vee \overline{Q_1}(x_1 \vee Q_2).$$

Функціональна схема автомата подана на рис. А1-3.15.

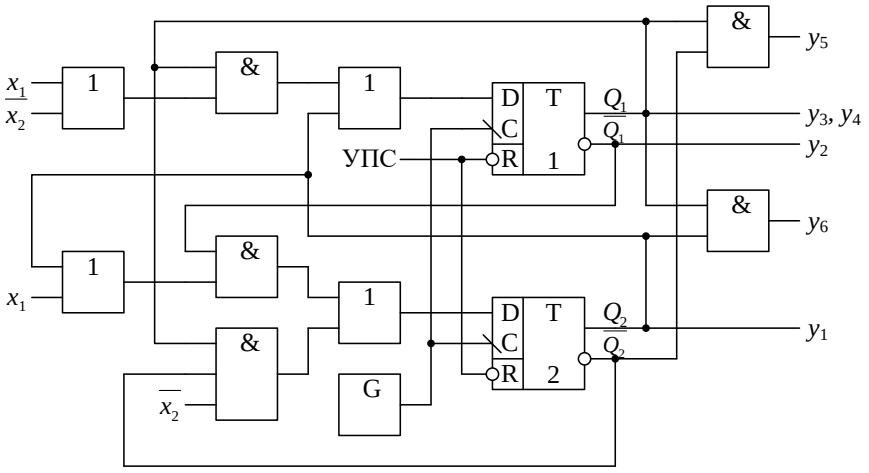


Рис. А1-3.15. Функціональна схема автомата Мура

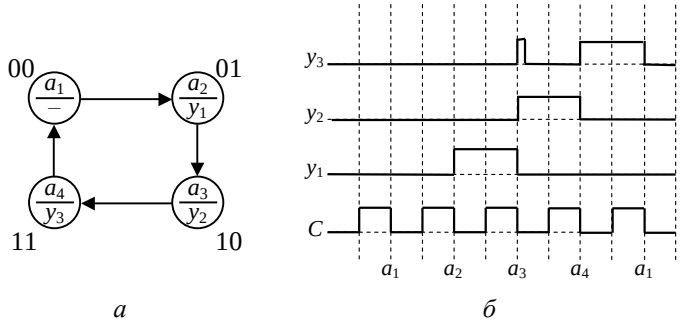
А1-3.4. Забезпечення стабільної роботи автоматів

Під час синтезу цифрових автоматів методом композиції тригерів виникають проблеми, пов'язані із забезпеченням коректності зміни станів, необхідної форми і тривалості вихідних сигналів, а також усуненням сигналів, непередбачених графом автомата.

На *коректність зміни станів* автомата не впливає вибір кодів станів, якщо для побудови автомата використовуються синхронні тригери, що спрацьовують за перепадом синхросигналу (див. розділ А1-4.7). В цьому випадку необхідно лише правильно обрати період синхросигналів. Мінімальний період синхросигналів не повинен бути менший за максимальну тривалість перехідних процесів у схемі автомату.

Хоча спосіб кодування станів синхронного автомату не впливає на правильність переходів з одного стану в інший, неоптимальне кодування станів обумовлює появу вихідних сигналів, непередбачених графом автомата. Це пов'язано з розкидом часу переключення окремих тригерів автомата (так званих «гонок»). Наприклад, під час переходу автомата зі стану 10 у стан 01, відбувається переключення двох тригерів. Унаслідок того, що один з тригерів може переключитися раніше ніж другий, можливе виникнення проміжних короткочасних станів 00 або 11 (залежно від того, який із тригерів раніше переключається). Ці проміжні стани можуть привести до появи *короткочасних помилкових керуючих сигналів*.

На рисунку А1-3.16, *а* зображений приклад графу автомату, коди станів якого наведені біля вершин графу. Наприклад, якщо під час переходу автомату із стану $a_2 = 01$ в стан $a_3 = 10$ виникне на короткий час стан $a_4 = 11$, то на виході автомату сформується короткочасний помилковий сигнал y_3, не відзначений на графі автомату (рис. А1-3.16, *б*). Цей сигнал може порушити роботу пристрою, яким керує автомат.



а *б*

Рис. А1-3.16. Приклад виникнення короткочасного помилкового сигналу: *а* — граф автомата; *б* — часова діаграма роботи автомата

Для усунення помилкових сигналів використовують різні методи. Один з підходів — це застосування сусіднього кодування станів, за якого не виникає одночасного перемикання декількох тригерів.

За *сусіднього кодування* дві поєднані однією дугою вершини кодуються так, щоб їхні коди відрізнялися значенням тільки одного розряду. Наглядним способом такого кодування є використання шаблонів, зображених на рис. А1-3.17.

Граф автомату накладається на шаблон відповідно до дуг, що поєднують вершини графу. У тому випадку, якщо на шаблоні немає відповідних графові зв'язків, вводяться додаткові вершини. Це дозволяє перенести вершину графа на таку позицію, що забезпечує необхідні зв'язки цієї вершини з іншими.

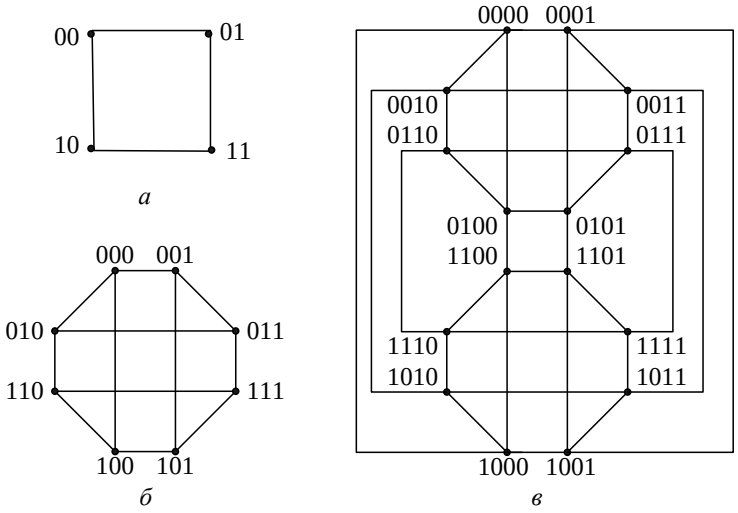


Рис. А1-3.17. Шаблони для виконання сусіднього кодування станів автомата:

а — для двохрозрядних кодів; *б* — для трьохрозрядних кодів; *в* — для чотирьохрозрядних кодів станів

Усунення короткочасних сигналів в автоматі Мура можна забезпечити також за рахунок спеціального кодування, що відрізняється від сусіднього кодування. За такого кодування управляючі сигнали можна знімати безпосередньо з виходів тригерів автомата Мура (комбінаційна схема для формування функцій y_j не потрібна). При цьому число тригерів має бути не менше числа керуючих сигналів, які не повторюють один одного і не є константами.

Код стану — це сукупність значень тригерів. Кожному вихідному сигналу відповідає один тригер. Наявність сигнала у вершині графа відповідає одиничному стану тригера, а відсутність — нулевому стану. Якщо у різних вершинах графа записані однакові сигнали, то коди станів будуть однаковими. Щоб усунути таку ситуацію, збільшують довжину кодів, тобто додають додаткові тригери, значення яких відрізняються для вершин з однаковими сигналами.

За такого кодування станів автомата Мура під час будь-якого переходу виключається можливість короткочасного формування помилкових управляючих сигналів. Крім того, відповідний рівень керуючих сигналів установлюється швидше, ніж в автомата Мілі, бо в автоматі відсутня комбінаційна схема для вихідних сигналів.

Приклад такого кодування станів автомата, що відповідає графові зображеному на рис. А1-3.18, наведений у табл. А1-3.9.

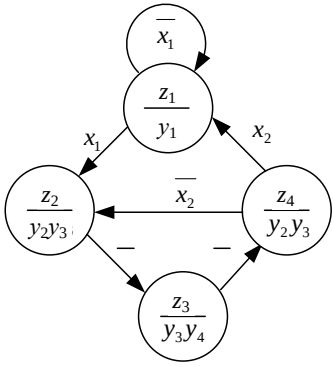


Рис. А1-3.18. Граф автомата Мура

Таблиця А1-3.9

ТАБЛИЦЯ КОДУВАННЯ СТАНІВ АВТОМАТА МУРА

Стан	Код стану				
	$y_1 = Q_1$	$y_2 = Q_2$	$y_3 = Q_3$	$y_4 = Q_4$	Q_5
z_1	1	0	0	0	0
z_2	0	1	1	0	0
z_3	0	0	1	1	0
z_4	0	1	1	0	1

У даному випадку автомат має п'ять тригерів, причому п'ятий тригер (вихід Q_5) забезпечує відмінність кодів для станів z_2 і z_4. Вихідні сигнали знімаються безпосередньо з виходів тригерів, тобто:

$$y_1 = Q_1,\ y_2 = Q_2,\ y_3 = Q_3,\ y_4 = Q_4.$$

Введення додаткових станів може знадобитися для забезпечення *необхідної форми і тривалості вихідних сигналів* автомата.

Якщо тривалість сигналу має дорівнювати декільком періодам синхросигналів, то сигнал повторюють у необхідній кількості додаткових вершин на ГСА автомата (так, на рис. А1-3.8 і рис. А1-3.12 введені додаткові вершини для сигналів y_3, y_4).

Додаткові вершини можуть бути також введені для забезпечення перепадів управляючих сигналів. Це необхідно, наприклад, якщо на ГСА є петля, яка охоплює операторну вершину з такими управляючими сигналами, або необхідно сформувати певну послідовність таких сигналів. (Звісно, що це не стосується сигналів, яки повинні мати тривалість у декілька тактів). Перепад сигналу забезпечується введенням у необхідному місці структурної ГСА порожньої операторної вершини, тобто вершини де цифровий автомат не виробляє управляючих сигналів (рис. А1-3.19).

На рисунку А-3.20 зображені часові діаграми, що відповідають фрагментам ГСА на рис. А1-3.19, *а* і *б*. Як видно з діаграм, за виконання циклічної гілки алгоритму без введення додаткового стану відсутні перепади управляючого сигналу y_5, а наявність додаткової вершини забезпечує перепади управляючого сигналу.

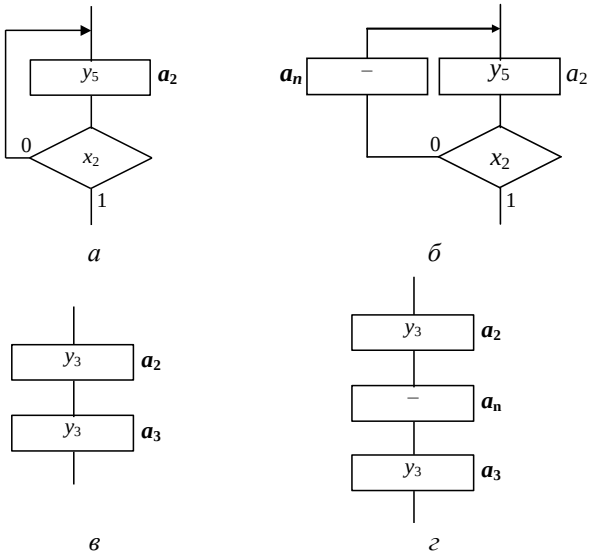
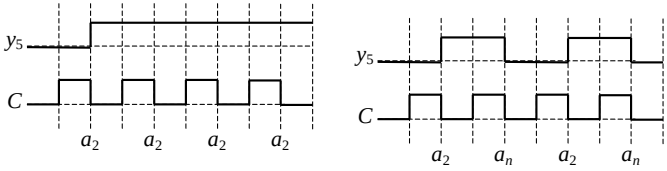


Рис. А1-3.19. Фрагменти алгоритмів, де необхідно введення додаткового стану для забезпечення перепаду сигналу:

а і *б* — відповідно вихідний та скорегований циклічний алгоритм; *б* і *г* — відповідно вихідний та скорегований лінійний алгоритм (a_n — додаткові вершини)



а *б*

Рис. А1-3.20. Часові діаграми перемикання керуючого сигналу:
а — без додаткового стану; *б* — з додатковим станом

А1-3.5. Структурний синтез синхронних автоматів з використанням апарата часових функцій

Автомат, що має два і більше станів, є послідовнісною схемою. Ознакою послідовнісної логічної схеми є наявність петель. Під петлею розуміють шлях з виходу логічного елемента на його вхід безпосередньо або через інші елементи. Метод структурного синтезу з використанням апарата часових функцій застосовується для прямого синтезу послідовнісних логічних схем і дозволяє побудувати цифровий автомат у будь-якому функціонально повному елементному базисі.

Часові перемикальні функції, на відміну від звичайних перемикальних функцій, як аргумент можуть використовувати власні значення, що обумовлено наявністю петлі з виходу елемента на його вхід.

Узагальнена структура автомата показана на рис. А1-3.21. Автомат містить логічну схему (ЛС), що складається з логічних елементів. Входами ЛС є зовнішні вхідні сигнали (логічні умови) x_1, ..., x_k, а також деякі виходи цієї ЛС, сигнали на яких розглядаються як часові функції Q_1^t, ..., Q_m^t (верхній індекс визначає момент автоматного часу). Логічна схема виробляє зовнішні вихідні управляючі сигнали y_1, ..., y_p.

Стани автомата кодуються функціями Q_i^t, кількість яких визначається співвідношенням $m \geq \left\lceil \log_2 s \right\rceil$. У загальному виді значення i-ої часової функції можна записати як

$$Q_i^{t+1} = f(x_1, ..., x_k, Q_1^t, ..., Q_m^t).$$

У процесі функціонування автомат переходить з одного стану в інший під дією вхідних сигналів. Кожному з множини станів $\{a_1, ..., a_s\}$ відповідає визначений набір значень часових функцій Q_i^t.

Розглянутий метод може бути використаний для синтезу автоматів Мілі і Мура, причому автомати можуть бути як синхронними, так і асинхронними.

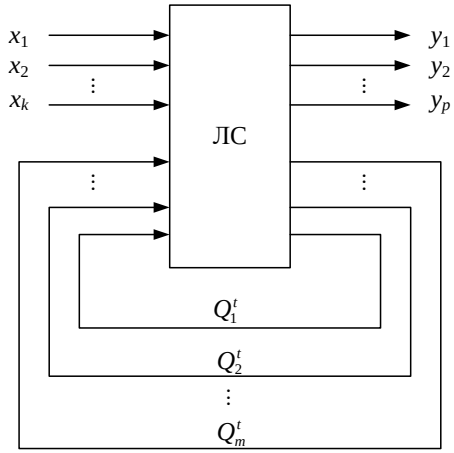


Рис. А1-3.21. Структура цифрового автомата

Автомат, як правило, забезпечує формування управляючих сигналів для функціонування деякого операційного пристрою за заданим алгоритмом. Таким чином, структурному синтезу автомата передує вивчення (побудова) операційного пристрою, системи операторів і розроблення алгоритму перетворення інформації, в результаті чого отримуємо ГСА абстрактного автомата, яку можна розглядати як вихідні дані для структурного синтезу.

Синтез автомата охоплює такі етапи:

1) складання для операційного пристрою списку управляючих сигналів, що забезпечують виконання кожного оператора;

2) визначення тривалості управляючих сигналів (у числі тактів) і періоду тактуючих сигналів автомата;

3) одержання ГСА структурного автомата;

4) розмітка станів автомата;

5) складання графа автомата;

6) «протигоночне» кодування станів автомата;

7) знаходження логічних виражень для часових функцій;

8) знаходження МДНФ функцій управляючих сигналів;

9) подання управляючих сигналів і часових функцій в операторній формі з урахуванням заданого елементного базису;

10) побудова і оптимізація схеми управляючого автомата.

Метод синтезу цифрових автоматів з використанням апарата часових функцій потребує обов'язкового використання сусіднього (протигоночного) кодування станів автомата (див. розділ А1-3.3).

На практиці переважно використовують синхронні автомати, що дозволяє забезпечити необхідну тривалість вихідних сигналів, пов'язану із періодом синхросигналів. Один такт роботи автомату відповідає напівперіоду синхросигналу. Для кожної вершини графу вхідні і вихідні дуги позначаються різними значеннями синхросигналів. Наприклад, якщо вхідні дуги позначаються C, то вихідні — $\overline{C}$ (або навпаки).

Для отримання часових змінних використовують такі вирази

$$Q_i^{t+1} = F_i \vee Q_i^t \overline{G_i}, \qquad \text{(А1-3.1)}$$

або

$$Q_i^{t+1} = (F_i \vee Q_i^t)\overline{G_i}, \qquad \text{(А1-3.2)}$$

де F_i — функція перемикання часової змінної з нульового стану в одиничний стан;

G_i — функція перемикання часової змінної з одиничного стану в нульовий стан.

Формула (А1-3.1) використовується для побудови логічних схем у елементному базисі Шефера, а формула (А1-3.2) — у елементному базисі Пірса. При цьому відповідне операторне подання часової змінної можна одержати із застосуванням правил де Моргана. Будь-який з наведених виразів може бути використаний для побудови логічних схем цифрових автоматів у булевому базисі.

Для отримання функцій F_i записують усі умови переключення значення Q_i^t з 0 у 1, поєднуючи їх логічною операцією АБО. Для отримання функцій G_i записують умови переключення змінної з 1 у 0.

Для визначення вказаних переходів доцільно побудувати структурну таблицю автомата за прикладом табл. А1-3.8, в якій замість функцій збудження тригерів відобразити часові функції.

Мінімізація функцій F_i і G_i виконується незалежно одна від одної. Якщо у кодуванні станів автомата використовуються не всі передбачені шаблоном коди, то невикористані коди розглядають як набори, на яких часові функції невизначені.

Для більш детального пояснення методу синтезу цифрових автоматів з використанням апарата часових функцій розглянемо структурний синтез автомата Мура, заданий графом на рис. А1-3.22, який

побудований на базі деякої структурної ГСА, складеної із урахуванням тривалості управляючих сигналів.

Граф на рис. А1-3.22 потребує доопрацювання, оскільки три пов'язані між собою вершини графу неможливо забезпечити сусіднім кодуванням.

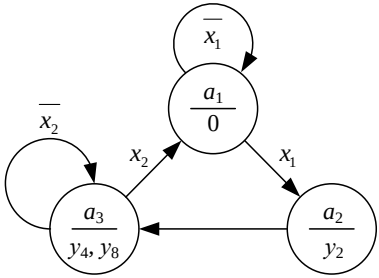


Рис. А1-3.22. Граф автомата Мура

Для виконання сусіднього кодування скористаємося шаблоном для двохрозрядних кодів станів автомата (рис. А1-3.17, *а*), в результаті одержимо граф, зображений на рис. А1-3.23.

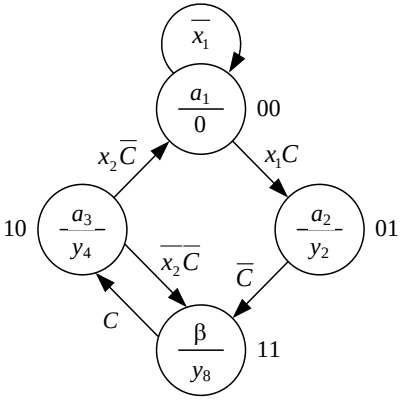


Рис. А1-3.23. Граф автомата із сусіднім кодуванням станів

Отримані коди станів автомату наведені в табл. А1-3.10. Дуги, що замикаються на власні вершини, на графі не відображені, тому що значення часових змінних визначаються тільки дугами між різними вершинами графу.

Таблиця А1-3.10

ТАБЛИЦЯ СУСІДНЬОГО КОДУВАННЯ СТАНІВ АВТОМАТА

Стан	Код стану

	Q_2^t	Q_1^t
a_1	0	0
a_2	0	1
a_3	1	0
β	1	1

Для графа автомата, зображеного на рис. А1-3.23, одержуємо систему функцій:

$$\begin{cases} F_1 = a_1 x_1 C \vee a_3 \overline{x_2}\,\overline{C} = \overline{Q_2}\,\overline{Q_1} x_1 C \vee Q_2 \overline{Q_1}\,\overline{x_2}\,\overline{C}; \\ G_1 = \beta C = Q_2 Q_1 C; \\ F_2 = a_2 \overline{C} = \overline{Q_2} Q_1 \overline{C}; \\ G_2 = a_3 x_2 \overline{C} = Q_2 \overline{Q_1} x_2 \overline{C}. \end{cases} \qquad \text{(А1-3.3)}$$

Реалізуємо отримані часові функції на логічних елементах І-НЕ, а вихідні сигнали на логічних елементах І. Відповідно до виразу (А1-3.1) і системи функцій (А1-3.3), застосовуючи правило де Моргана, одержимо операторні форми часових функцій:

$$Q_1^{t+1} = \overline{\overline{Q_2}\,\overline{Q_1} x_1 c \cdot \overline{Q_2}\,\overline{Q_1}\,\overline{x_2}\,\overline{c} \cdot \overline{R} \cdot \overline{Q_1^t \overline{R} \cdot \overline{Q_2 Q_1 c}}};$$

$$Q_2^{t+1} = \overline{\overline{\overline{Q_2}\,\overline{Q_1}\,\overline{c}\,\overline{R}} \cdot \overline{Q_2^t \overline{R} \cdot Q_2 \overline{Q_1}\,\overline{x_2}\,\overline{c}}},$$

де R — сигнал встановлення часових функцій у нульовий стан.

Вихідні сигнали в автоматі Мура цілком визначаються за кодами станів автомату, відповідно до графу на рис. А1-3.23 отримаємо:

$$y_2 = \overline{Q_2} Q_1;$$

$$y_4 = Q_2 \overline{Q_1};$$

$$y_8 = Q_2 Q_1.$$

Схема, отримана за знайденими операторними формами часових функцій і функцій управляючих сигналів, зображена на рис. А1-3.24.

Під час синтезу автомата Мілі часові функції визначаються аналогічним чином. Відмінність полягає в розмітці станів автомату, побудові графу автомата і отримані функцій виходів. У даному випадку функції виходів залежать від станів автомата і вхідних сигналів, синхросигнал як аргумент не враховується.

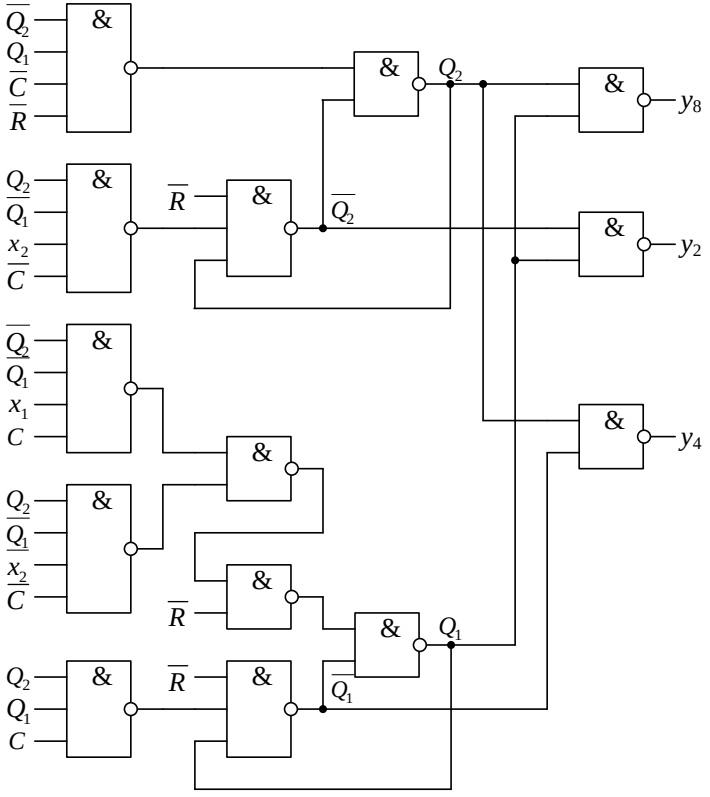


Рис. А.1-3.24. Функціональна схема автомата Мура

А1-4. Типові вузли цифрових ЕОМ

А1-4.1. Дешифратори

До типових вузлів належать логічні схеми, що використовуються в цифрових ЕОМ найчастіше. Серед них існують як комбінаційні, так і послідовнісні схеми з пам'яттю.

Дешифратор — це типова комбінаційна схема, призначена для реалізації конституент одиниці.

Розрізняють *повні* і *неповні* дешифратори. Повні дешифратори реалізують 2^n конституент, де n — це число інформаційних входів. Неповні дешифратори реалізують менш ніж 2^n конституент.

Приклади умовного графічного позначення трьохвходових дешифраторів показаний на рис. А1-4.1, *а, б*. Сигнали на виходах дешифраторів формуються відповідно до таблиці істинності (табл. А1-4.1). Неповний дешифратор (рис. А1-4.1, *б*) має тільки шість виходів C_0 — C_5, тому сигнали на виходах дешифратора формуються відповідно до виділеного фрагмента таблиці істинності (табл. А1-4.1) для наборів з номерами 0 — 5.

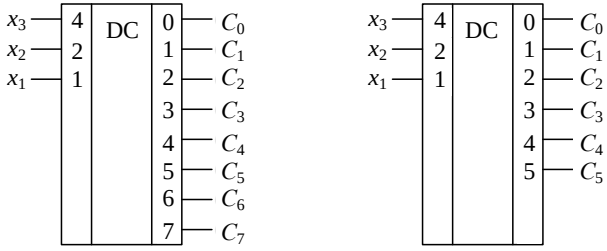


Рис. А1-4.1. Умовне графічне позначення дешифраторів:
а — повного дешифратора; *б* — неповного дешифратора

Повний дешифратор реалізує систему перемикальних функцій, що включає в себе всі конституенти одиниці для n аргументів:

$$C_0 = \overline{x}_3 \overline{x}_2 \overline{x}_1;$$
$$C_1 = \overline{x}_3 \overline{x}_2 x_1;$$
$$C_2 = \overline{x}_3 x_2 \overline{x}_1;$$
$$\ldots\ldots\ldots\ldots$$
$$C_7 = x_3 x_2 x_1.$$

Таблиця А1-4.1

ТАБЛИЦЯ ІСТИННОСТІ ФУНКЦІЙ ДЕШИФРАТОРА

	x_3	x_2	x_1	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Функціональна схема повного дешифратора зображена на рис. А1-4.2. Неповний дешифратор має на два елементи І меньше ніж повний (що реалізують конституенти одиниці 6 і 7).

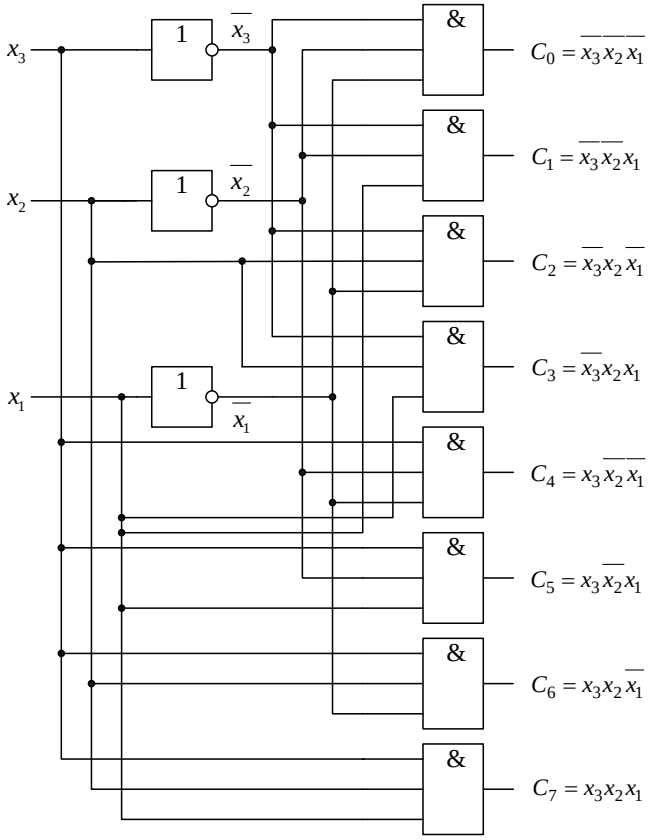


Рис. А1-4.2. Функціональна схема повного дешифратора

Під час побудови неповних дешифраторів можливе здійснення мінімізації функцій виходів за рахунок наборів, на яких функції не визначені. Наприклад, для неповного дешифратора, зображеного на рис. А1-4.1, *б*, функції виходів C_2, C_3, C_4, C_5 можна спростити із урахуванням заборонених наборів 6 та 7. Мінімізація визначених наборів за допомогою діаграм Вейча проілюстрована на рис. А1-4.3.

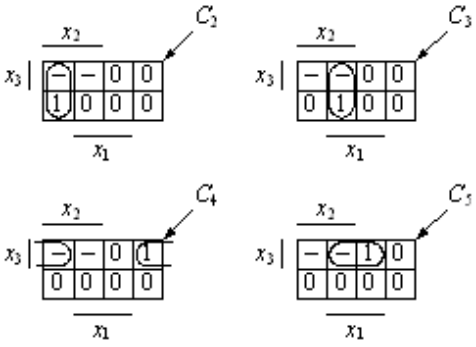


Рис. А1-4.3. Діаграми Вейча функцій виходів неповного дешифратора

У результаті мінімізації отримаємо функції виходів неповного дешифратора:

$$C_0 = \overline{x_3}\,\overline{x_2}\,\overline{x_1};$$
$$C_1 = \overline{x_3}\,\overline{x_2}\,x_1;$$
$$C_2 = x_2\,\overline{x_1};$$
$$C_3 = x_2 x_1;$$
$$C_4 = x_3\,\overline{x_1};$$
$$C_5 = x_3 x_1.$$

Функціональна схема неповного дешифратора зображена на рис. А1-4.4. Для цього дешифратора набори з номерами 6 і 7 вважаються забороненими. Якщо вказані набори подаватимуться на входи дешифратора, то на його виходах можуть виникати помилкові сигнали. Наприклад, набір $x_3x_2x_1$ викликає одиничні значення функцій $C_3 = x_2x_1$ і $C_5 = x_3x_1$.

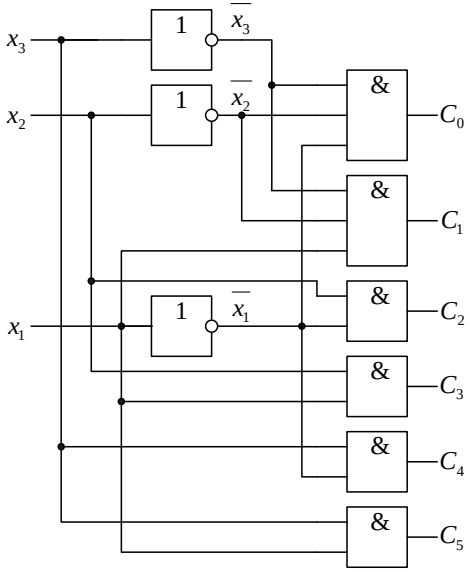


Рис. А1-4.4. Функціональна схема неповного дешифратора

А1-4.2. Шифратори

Шифратор — це типова комбінаційна схема, призначена для перетворення просторового унітарного коду в двійковий код з природним порядком вагів.

Унітарний код має в своєму записі одну одиницю. З урахуванням цього можна вважати, що шифратор виконує функцію, зворотну функції дешифратора, хоча в загальному випадку вихідний код може відрізнятися від коду з природним порядком вагів.

Таблиця істинності шифратора для трьохрозрядного вихідного коду з природнім порядком вагів наведена у табл. А1-4.2. Безпосередньо з таблиці одержимо:

$$y_1 = x_1 \vee x_3 \vee x_5 \vee x_7;$$
$$y_2 = x_2 \vee x_3 \vee x_6 \vee x_7;$$
$$y_3 = x_4 \vee x_5 \vee x_6 \vee x_7.$$

Функціональна схема шифратора, реалізована на елементах АБО, зображена на рис. А1-4.5, а умовне графічне позначення шифратора показано на рис. А1-4.6.

Таблиця А1-4.2

ТАБЛИЦЯ ІСТИННОСТІ ШИФРАТОРА

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

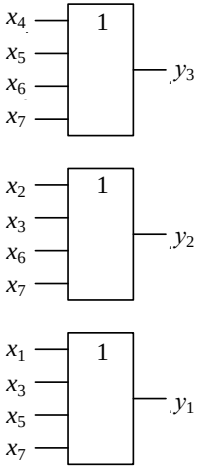


Рис. А1-4.5. Функціональна схема шифратора

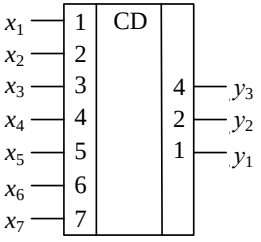


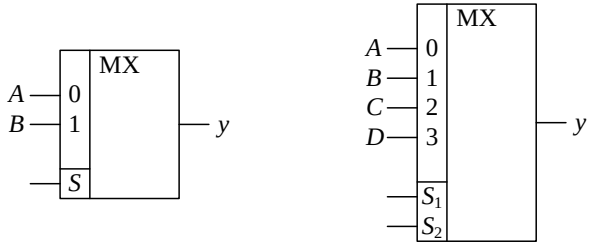
Рис. А1-4.6. Умовне графічне позначення шифратора

А1-4.3. Мультиплексори

Мультиплексор — це типова комбінаційна схема (комутатор), що підключає до свого виходу один із інформаційних входів у відповідно до сигналів, що подаються на управляючі входи.

Мультиплексор має $n = 2^s (s = 1, 2, 3, ...)$ інформаційних входів і $s = \log_2 n$ управляючих входів. Однорозрядний мультиплексор з n інформаційними входами характеризується парою чисел n та 1 і позначається як МХ(n – 1). Наприклад, МХ(4 – 1) — однорозрядний мультиплексор з чотирма інформаційними входами, МХ(2 – 1) — однорозрядний мультиплексор з двома інформаційними входами.

Умовні графічні позначення мультиплексорів МХ(2 – 1) і МХ(4 – 1) зображені на рис. А1-4.7, *а* і *б*, їх таблиці істинності –відповідно у табл. А1-4.3 і А1-4.4, а функціональні схеми — на рис. А1-4.8.



а *б*

Рис. А1-4.7. Умовні графічні позначення мультиплексорів:
а — МХ(2 – 1); *б* — МХ(4 – 1)

Таблиця А1-4.3

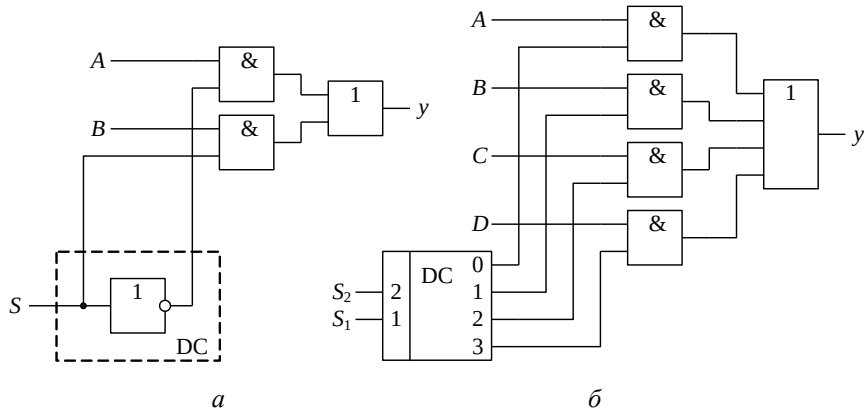
ТАБЛИЦЯ ІСТИННОСТІ МУЛЬТИПЛЕКСОРА МХ(2 – 1)

S	Y
0	A
1	B

Таблиця А1-4.4

ТАБЛИЦЯ ІСТИННОСТІ МУЛЬТИПЛЕКСОРА МХ(4 – 1)

S_2	S_1	Y
0	0	A
0	1	B
1	0	C
1	1	D



а *б*

Рис. А1-4.8. Функціональні схеми мультиплексорів:
а — МХ (2 – 1); *б* — МХ(4 – 1)

Виходом однорозрядного мультиплексора є вихід елемента АБО, до входів якого через елементи І підключаються інформаційні входи. Управління підключенням може забезпечити дешифратор.

Під час побудови багаторозрядних мультиплексорів для управлінням групами входів кожного розряду (виходу) використовується один дешифратор.

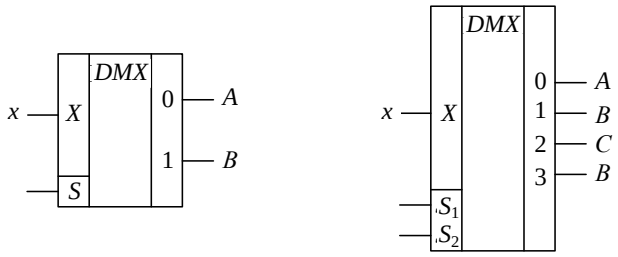
А1-4.4. Демультиплексори

Демультиплексор — це типова комбінаційна схема (комутатор), що підключає інформаційний вхід до одного з декількох інформаційних виходів відповідно до сигналів, що подаються на управляючі входи.

Демультиплексор виконує функцію, зворотну функції мультиплексора. Однорозрядний демультиплексор, що має один інформаційний вхід і *n* виходів, характеризується парою чисел 1 і *n*. Наприклад, DMX(1 – 2) — однорозрядний демультиплексор з двома виходами, DMX(1 – 4) — з чотирма виходами і таке інше. Кількість управляючих входів дорівнює $s = \log_2 n$.

Умовне графічне позначення демультиплексорів DMX(1 – 2) і DMX(1 – 4) зображено на рис. А1-4.9.

Демультиплексори на рис. А1-4.9 функціонують відповідно до табл. А1-4.5 і А1-4.6. Функціональні схеми демультиплексорів зображені на рис. А1-4.10.



а *б*

Рис. А1-4.9. Умовне графічне позначення демультиплексорів:
а — DMX(1 – 2); *б* — DMX(1 – 4)

Таблиця А1-4.5

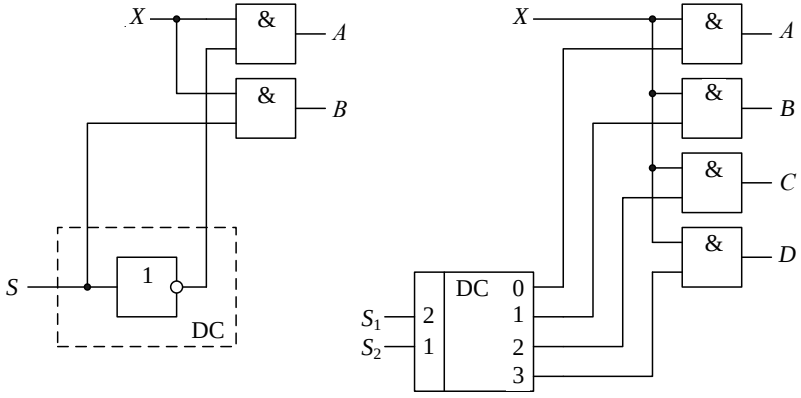
ТАБЛИЦЯ ІСТИННОСТІ ДЕМУЛЬТИПЛЕКСОРА DMX(1 – 2)

S	*A*	*B*
0	X	0
1	0	X

Таблиця А1-4.6

ТАБЛИЦЯ ІСТИННОСТІ ДЕМУЛЬТИПЛЕКСОРА DMX(1 – 4)

S_2	S_1	*A*	*B*	*C*	*D*
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X



а *б*

Рис. А1-4.10. Функціональні схеми демультиплексорів:
а — DMX(1 – 2); *б* — DMX(1 – 4)

А1-4.5. Комбінаційні суматори

Для додавання розрядів двійкових кодів з однаковою вагою використовують *двійкові комбінаційні суматори*.

Однорозрядним суматором називають комбінаційну схему, що за розрядним значенням x_i і y_i доданків і за значенням переносу z_i із сусіднього молодшого розряду формує значення розрядної суми S_i і переносу у наступний старший розряд P_i.

Робота однорозрядного суматора може бути описана таблицею істинності (табл. А1-4.7).

Таблиця А1-4.7

ТАБЛИЦЯ ІСТИННОСТІ КОМБІНАЦІЙНОГО СУМАТОРА

x	y_i	z_i	P_i	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Схемне рішення суматора залежить від системи елементів, що використовується, вимог до тривалості операції підсумовування і припустимих апаратурних витрат.

Розглянемо побудову суматора у булевому елементному базисі (рис. А1-4.11). Будемо вважати, що цільова функція проектування — мінімізація апаратурних витрат.

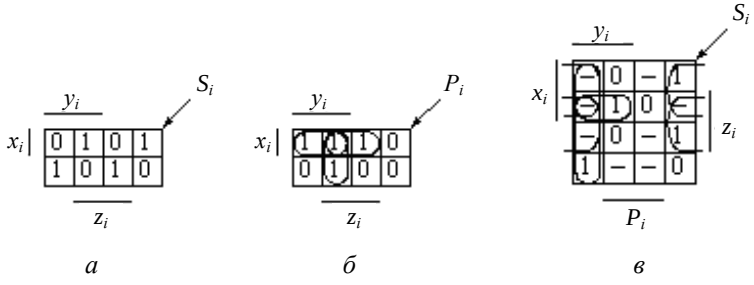


Рис. А1-4.11. Діаграми Вейча функцій:
а — суми, що залежать від трьох аргументів; *б* — переноса; *в* — суми, що залежать від чотирьох аргументів

Виходячи з рис. А1-4.11, *а* і А1-4.11, *б* МДНФ функцій S_i і P_i можна записати у вигляді:

$$S_i = \overline{x_i}\,\overline{y_i} z_i \vee \overline{x_i} y_i \overline{z_i} \vee x_i \overline{y_i}\,\overline{z_i} \vee x_i y_i z_i; \qquad P_i = x_i y_i \vee x_i z_i \vee y_i z_i. \tag{А1-4.1}$$

Аналогічно можна знайти МКНФ функцій S_i і P_i :

$$S_i = (x_i \vee y_i \vee z_i) \cdot (x_i \vee \overline{y_i} \vee \overline{z_i}) \cdot (\overline{x_i} \vee y_i \vee \overline{z_i}) \cdot (\overline{x_i} \vee \overline{y_i} \vee z_i); \qquad P_i = (x_i \vee y_i) \cdot (x_i \vee z_i) \cdot (y_i \vee z_i). \tag{А1-4.2}$$

Неважко підрахувати, що складність дворівневої комбінаційної схеми за Квайном у випадку використання форм (А1-4.2) і (А1-4.3) без урахування інверторів дорівнюватиме К = 25 (якщо немає обмежень кількості входів логічних елементів). Затримка сигналів у схемі дорівнюватиме сумарній затримці сигналів на елементах І і АБО, тобто $t = \tau_{\text{І}} + \tau_{\text{АБО}}$.

Можна спростити функцію S_i, якщо розглядати її, як залежну від чотирьох аргументів, а саме $S_i = f(x_i, y_i, z_i, P_i)$. В цьому випадку функція є визначеною тільки на восьми наборах із можливих шістнадцяти. Згідно з діаграмою Вейча (рис. А1-4.11, *в*), одержимо МДНФ функції у вигляді:

$$S_i = x_i y_i z_i \vee x_i \overline{P_i} \vee y_i \overline{P_i} \vee z_i \overline{P_i}\,.$$

Для зменшення складності комбінаційної схеми подамо функції у формі:

$$S_i = (x_i y_i) \cdot z_i \vee \overline{P_i}((x_i \vee y_i) \vee z_i); \qquad P_i = x_i y_i \vee z_i \cdot (x_i \vee y_i).$$

Функціональна схема комбінаційного суматора, побудованого за отриманими формами, зображена на рис. В1-4.12. Складність схеми за Квайном дорівнює К = 17. Така схема має меншу складність ніж схеми, побудовані за формами (А1-4.1) і (А1-4.2), але програє їм за швидкодією. Справді, для даної схеми затримка сигналів складає $t = 2\tau_{\text{І}} + 3\tau_{\text{АБО}} + \tau_{\text{НЕ}}$.

Умовне графічне позначення (УГП) комбінаційного суматора наведено на рис. А1-4.13, *а*.

На базі однорозрядних суматорів будують суматори для підсумування багаторозрядних чисел $X = x_n x_{n-1} \ldots x_1$ і $Y = y_n y_{n-1} \ldots y_1$. При цьому однорозрядні суматори пов'язані між собою ланцюгами поширення переносів.

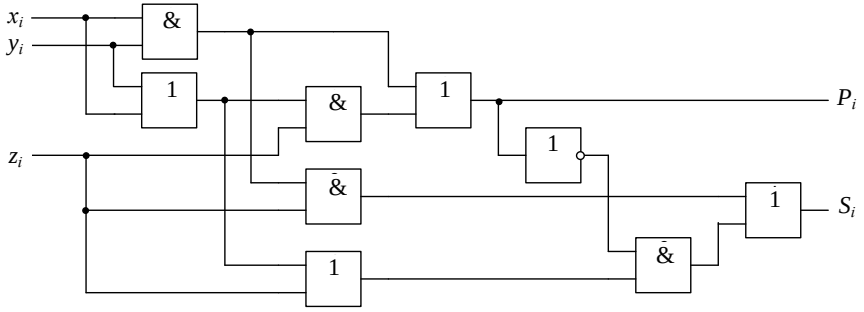
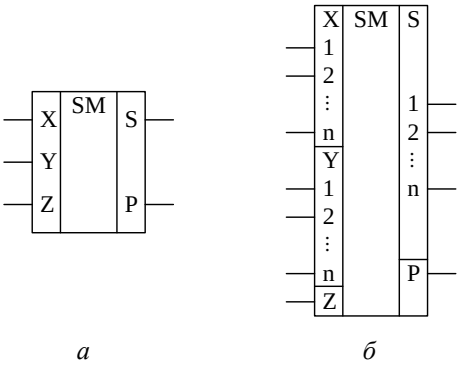


Рис. В1-4.12. Комбінаційний однорозрядний суматор

Багаторозрядний суматор з послідовним переносом зображений на рис. А1-4.14. Існують і складніші схеми розповсюдження переносів, які забезпечують більшу швидкодію суматорів. Умовне графічне позначення багаторозрядного суматора зображено на рис. А1-4.13, *б*.



а *б*

Рис. А1-4.13. Умовні графічні позначення суматорів:
а — для однорозрядного суматора; *б* — для багаторозрядного суматора

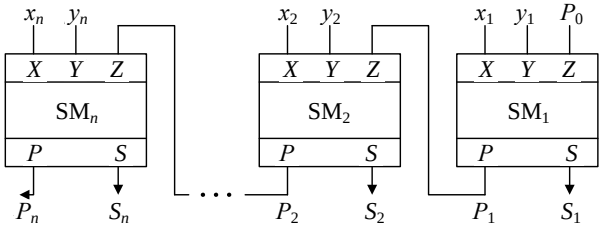


Рис. А1-4.14. Функціональна схема багаторозрядного суматора

Багаторозрядний суматор можна спрощено зобразити у вигляді операційного вузла (рис. А1-4.15). У цьому випадку окремі розряди двійкових кодів не зображують.

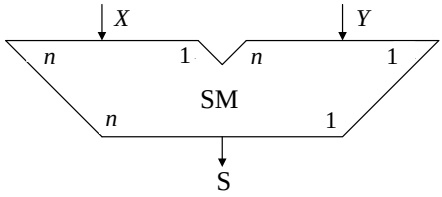


Рис. А1-4.15. Суматор для підсумовування *n*-розрядних чисел

А1-4.6. Програмовані логічні матриці

Досягнення в галузі технології інтегральних схем дозволяють в одному кристалі концентрувати велику кількість логічних елементів. Для розширення сфери застосування таких інтегральних схем, вони реалізуються у вигляді програмованих логічних матриць (ПЛМ).

ПЛМ характеризуються параметрами, які записуються у такому вигляді: ПЛМ(*n*, *p*, *m*), де *n* — число інформаційних входів, *p* — число проміжних внутрішніх шин, *m* — число інформаційних виходів. Логічну структуру ПЛМ наведено на рис. А1-4.16.

Програмування ПЛМ виконується на стадії виготовлення або програмується користувачем за допомогою спеціальних програматорів (перепрограмовувальних схем). Програмування зводиться до установки або вилучення зв'язків у точках, відзначених на рис. А1-4.16 хрестиками.

ПЛМ є типовою комбінаційною схемою і може реалізувати систему перемикальних функцій, поданих у нормальних формах І / АБО та І / АБО-НЕ, якщо система задовольняє таким умовам:

— кількість аргументів системи, від яких функції залежать суттєво, не перевищує кількості входів (*n*);

— кількість реалізовуваних функцій, що не є константами, не перевищує кількості виходів (*m*);

— сумарне число термів, що взаємно відрізняються один від одного, (імплікант і конституент), не перевищує кількості внутрішніх шин (*p*).

Для ілюстрації програмування матриць використовують спрощену *мнемонічну схему* ПЛМ. Приклад мнемонічної схеми ПЛМ(4, 6, 4) показаний на рис. А1-4.17, де точками відзначено зв'язки між шинами. Умовне графічне позначення ПЛМ(4, 6, 4) наведено на рис. А1-4.18.

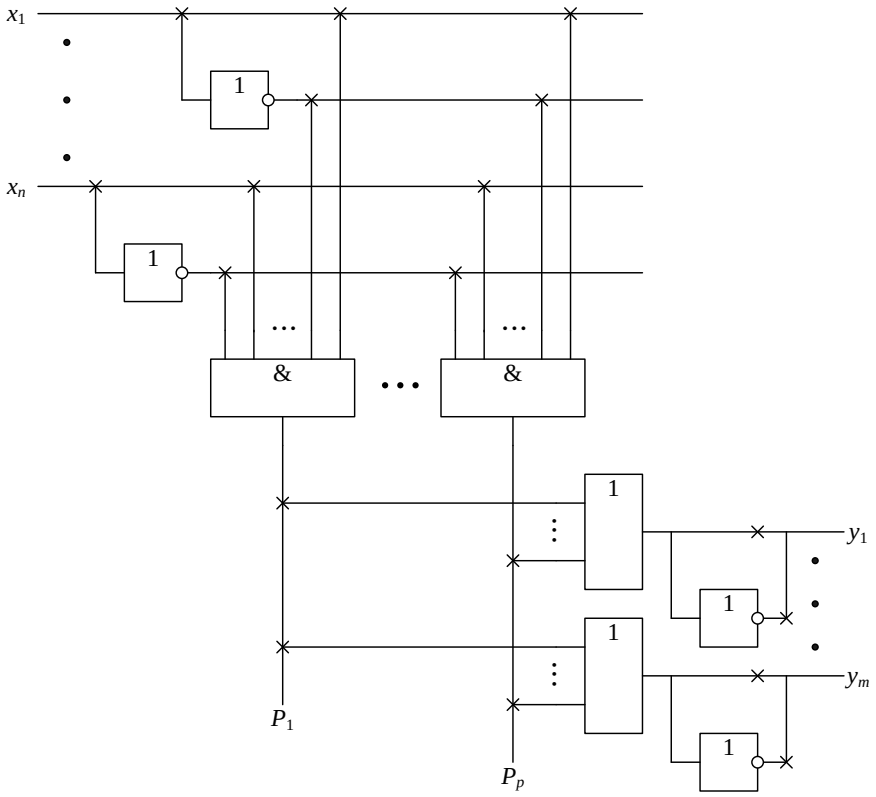


Рис. А1-4.16. Логічна структура ПЛМ

Як видно з рис. А1-4.17, логічні елементи І реалізують кон'юнкції:

$$P_1 = x_4 x_3 \overline{x_2};\ \ P_2 = x_4 \overline{x_2 x_1};\ \ P_3 = x_4 \overline{x_3} x_2 x_1;$$
$$P_4 = \overline{x_4} x_3 x_2 x_1;\ \ P_5 = \overline{x_4 x_3 x_2};\ \ P_6 = \overline{x_4 x_3 x_1}.$$

ПЛМ, що розглядається, реалізує систему функцій:

$$\begin{cases} y_1 = \overline{P_1 \vee P_2 \vee P_4} = \overline{x_4 x_3 \overline{x_2} \vee x_4 \overline{x_2 x_1} \vee \overline{x_4} x_3 x_2 x_1}; \\ y_2 = \overline{P_1 \vee P_3 \vee P_4 \vee P_6} = \overline{x_4 x_3 \overline{x_2} \vee x_4 \overline{x_3} x_2 x_1 \vee \overline{x_4} x_3 x_2 x_1 \vee \overline{x_4 x_3 x_1}}; \\ y_3 = P_2 \vee P_3 \vee P_4 = x_4 \overline{x_2 x_1} \vee x_4 \overline{x_3} x_2 \cdot x_1 \vee \overline{x_4} x_3 x_2 x_1; \\ y_4 = P_1 \vee P_5 \vee P_6 = x_4 x_3 \overline{x_2} \vee \overline{x_4 x_3 x_2} \vee \overline{x_4 x_3 x_1}. \end{cases}$$

ПЛМ може бути запрограмована відповідно до табл. А1-4.8.

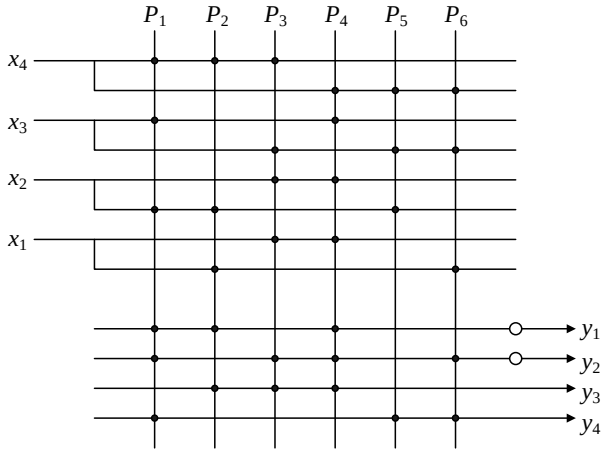


Рис. А1-4.17. Мнемонічна схема ПЛМ (4, 6, 4)

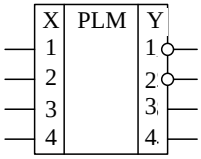


Рис. А1-4.18. Умовне графічне позначення ПЛМ (4, 6, 4)

Таблиця А1-4.8

КАРТА ПРОГРАМУВАННЯ ПЛМ(4, 6, 4)

x_4	x_3	x_2	x_1	P_i	$\overline{y_1}$	$\overline{y_2}$	y_3	y_4
1	1	0	–	P_1	1	1	0	1
1	–	0	0	P_2	1	0	1	0
1	0	1	1	P_3	0	1	1	0
0	1	1	1	P_4	1	1	1	0
0	0	0	–	P_5	0	0	0	1
0	0	–	0	P_6	0	1	0	1

Під час побудови комбінаційних схем на базі ПЛМ враховуються різні цільові функції проектування. Найскладнішим є завдання знаходження мінімальних параметрів ПЛМ за реалізації заданої системи функцій. Процес проектування комбінаційних схем у цьому випадку зводиться до сумісної мінімізації системи перемикальних

функції в базисі І, АБО та АБО-НЕ і отримання нормальних форм І / АБО та І / АБО-НЕ з мінімальним числом термів, що відрізняються один від одного.

А1-4.7. Тригери

У схемах цифрової обчислювальної техніки як запам'ятовуючі елементи широко використовують тригери. Вони призначені для зберігання двійкової змінної і являють собою пристрій із двома стійкими станами. Тригер містить запам'ятовуючий елемент (ЗЕ) і схему керування (СК) (рис. А1-4.19).

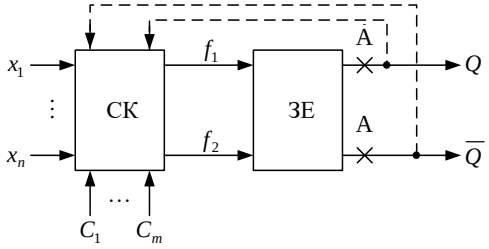


Рис. А1-4.19. Узагальнена структура тригера

На тригерній схемі (рис. А1-4.19) прийняті такі позначення:

$x_1, ..., x_n$ — інформаційні входи;

$C_1, ..., C_m$ — тактуючі входи;

f_1, f_2 — функції збудження ЗЕ;

Q, $\overline{Q}$ — відповідно прямий та інверсний виходи.

Можливі й простіші варіанти тригерних схем, наприклад, тригер може не мати тактуючих входів і навіть СК.

Тригери класифікують за функціональною ознакою і способом запису інформації.

Функціональна класифікація більш загальна і характеризує стани входів і виходів тригера в моменти часу до його перемикання і після нього. Наприклад, якщо тригер має один інформаційний вхід x, то на вході можливі стани $x = 0$ і $x = 1$. Тоді можливими станами тригера є: 0, 1, Q, $\overline{Q}$ та «*» (де «*» — невизначений стан□). У цьому випадку можна одержати 25 функціональних типів тригерів. У загальному випадку за наявності n інформаційних входів можна одержати 5^{2^n} типів функціональних типів тригерів.

На практиці застосовується невелика кількість типів тригерів. До них належать *RS*-, *D*-, *JK*-, *T*-, *E*-, *R*-, *S*-тригери. Частково інформація про *RS*-, *D*-, *JK*- і *T*-тригери надана в розділі А1-3.3 Розглянемо різні типи тригерів докладніше.

Спосіб функціонування тригерів описується *таблицею переходів* тригерів. З таблиці переходів *RS*-тригера (табл. А1-4.9) випливає, що тригер не змінює свого стану в момент $(t + 1)$ $(Q^{t+1} = Q^t)$, якщо в момент часу t має місце $R^t = 0$ і $S^t = 0$. За наявності сигналів $R^t = 0$, $S^t = 1$ тригер встановлюється в одиничний стан $Q^{t+1} = 1$, а за комбінації $R^t = 1$, $S^t = 0$ — у нульовий стан $Q^{t+1} = 0$. За $R^t = S^t = 1$ стан тригера не визначений $(Q^{t+1} = *)$. Така комбінація сигналів для *RS*-тригера є забороненою.

R-тригер відрізняється від *RS*-тригера тим, що за комбінації вхідних сигналів $S^t = R^t = 1$ він переходить у нульовий стан $(Q^{t+1} = 0)$ (табл. А1-4.10). *S*-тригер (табл. А1-4.11) у цьому випадку переходить в одиничний стан $(Q^{t+1} = 1)$, а *E*-тригер (табл. А1-4.12) не змінює свого стану $(Q^{t+1} = Q^t)$.

Таблиця А1-4.9

ТАБЛИЦЯ ПЕРЕХОДІВ *RS*-ТРИГЕРА

R^t	S^t	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	*

Таблиця А1-4.10

ТАБЛИЦЯ ПЕРЕХОДІВ *R*-ТРИГЕРА

R^t	S^t	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	0

Таблиця А1-4.11

ТАБЛИЦЯ ПЕРЕХОДІВ *S*-ТРИГЕРА

R^t	S^t	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	1

Таблиця А1-4.12

ТАБЛИЦЯ ПЕРЕХОДІВ *E*-ТРИГЕРА

R^t	S^t	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	Q^t

D-тригер називають *тригером затримки* (табл. А1-4.13). Він характеризується тим, що затримує вхідний сигнал. Для такого тригера справедливе рівняння $(Q^{t+1} = D^t)$.

T-тригер характеризується тим, що підраховує по модулю 2 одиниці, що надходять на його вхід *T* (табл. А1-4.14).

З таблиці переходів *JK*-тригера (табл. А1-4.15) випливає, що за комбінацій вхідних сигналів $J^t = K^t = 0$, $J^t = 0$ і $K^t = 1$, а також $J^t = 1$

і $K^t = 0$ він працює як *RS*-тригер (вхід *J* відповідає входові *S*, а *K* — входові *R*), а при $J^t = K^t = 1$ змінює свій стан на протилежний, тобто працює як *T*-тригер.

Таблиця А1-4.13

ТАБЛИЦЯ ПЕРЕХОДІВ *D*-ТРИГЕРА

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

Таблиця А1-4.14

ТАБЛИЦЯ ПЕРЕХОДІВ *T*-ТРИГЕРА

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

Таблиця А1-4.15

ТАБЛИЦЯ ПЕРЕХОДІВ *JK*-ТРИГЕРА

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

Класифікація тригерів за способом запису інформації характеризує хід процесу перемикання тригера. Відповідно до цієї класифікації тригери розділяють на *асинхронні* і *синхронні*. Запис інформації в *асинхронні тригери* здійснюється безпосередньо з надходженням інформаційних сигналів (такі тригери не мають тактуючих входів). *Синхронні тригери* мають тактуючі входи. Сьогодні найчастіше використовують однотактні тригери, що розглядаються далі.

Розрізняють синхронні тригери, *керовані рівнем тактового сигналу*, і *керовані перепадом тактового сигналу (тригери із внутрішньою затримкою)*.

Тригери першого типу з появою тактуючого сигналу на вході *C* (тобто за $C = 1$) перемикаються відповідно до таблиці переходів. Характерною рисою таких тригерів є те, що сигнали на виходах тригерів можуть неодноразово змінюватися (залежно від інформаційних сигналів), поки рівень тактуючого сигналу на вході *C* дорівнює рівневі логічної одиниці.

У тригерах другого типу вихідні сигнали, що відповідають новому станові тригера, з'являються тільки в момент переходу тактуючого сигналу з 0 в 1 або навпаки.

Тригер може бути синтезований як автомат Мура з використанням апарата часових функцій. Другий підхід до проектування тригерних пристроїв полягає у виборі ЗЕ (див. рис. А1-4.19) і синтезі СК, що реалізує функції збудження f_1 і f_2 для ЗЕ в заданому елементному базисі. Розглянемо цей підхід докладніше.

Якщо в стовпці Q^{S+1} таблиці переходів проектованого тригера є значення $\overline{Q^S}$ (див., наприклад, рис. А1-4.14 і А1-4.15), то сигнали на

виходах Q і $\overline{Q}$ тригера являють собою аргументи функцій f_1 і f_2. Для забезпечення правильного переключення тригера в точках А (див. рис. А1-4.19) необхідно включити елементи затримки. Для потенціальної системи сигналів, коли тривалість сигналів може бути різною, застосовують два основні способи побудови тригерів із внутрішньою затримкою, а саме: *MS-схеми* і *схеми трьох тригерів*.

Перший спосіб полягає у використанні для побудови тригера двох ЗЕ: основного (*M*-тригера) і допоміжного (*S*-тригера) (рис. А1-4.20).

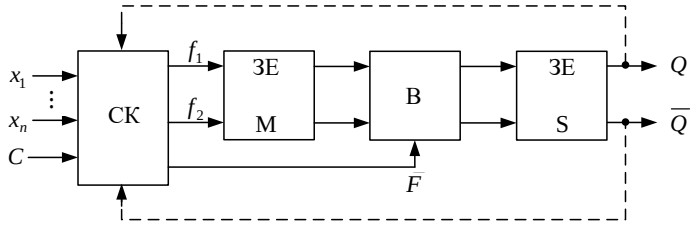


Рис. А1-4.20. Структура однотактного *MS*-тригера

Запис інформації в *M*-тригер тактується сигналом *C*, а в *S*-тригер — сигналом *F*. Інформація з *M*-тригера в *S*-тригер передається через вентилі В.

Найбільше поширення одержали *MS*-тригери з інвертором у ланцюзі *C*. Схеми таких тригерів на елементах І-НЕ показані на рис. А1-4.21. На цій схемі елементи 1, 2 утворюють *S*-тригер, а елементи 5, 6 — *M*-тригер.

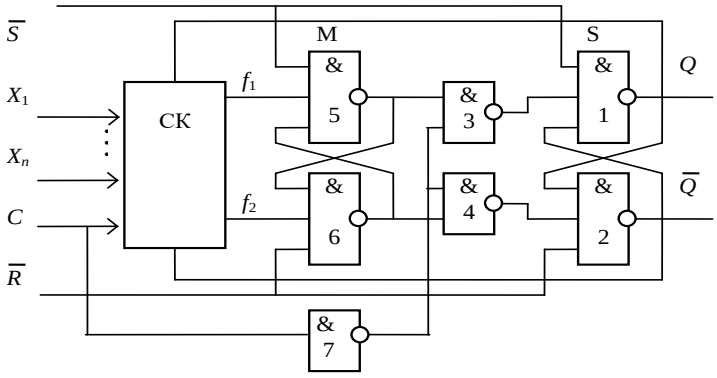


Рис. А1-4.21. Структура *MS*-тригера
з інвертором в ланцюзі *C*

Тригерний пристрій, побудований на елементах І-НЕ за схемою трьох тригерів, зображений на рис. А1-4.22.

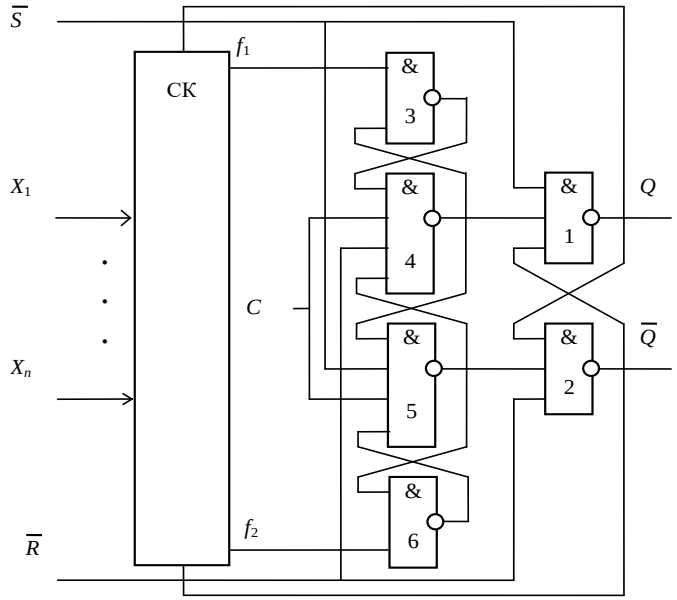


Рис. А1-4.22. Тригерний пристрій, реалізований за схемою трьох тригерів

У такому тригері сигнали, що відповідають новому стану, установлюються під час переходу тактуючого сигналу з 0 у 1.

Під час синтезу СК на підставі таблиці переходів тригера будується повна таблиця переходів, у якій відображають також значення Q^t у момент часу t і, за необхідності, — значення C. З повної таблиці переходів одержують вирази для f_1 і f_2, мінімізують отримані функції і реалізують їх на заданих елементах.

Як приклад розглянемо процес проектування *JK*-тригера на елементах І-НЕ.

Оскільки в табл. А1-4.15 є значення Q^t, то тригер повинний мати внутрішню затримку. Вибираємо структуру тригера, показану на рис. А1-4.21.

Порядок переходів тригера на вентилях І-НЕ (див. елементи 1 і 2 на рис. А1-4.21) залежно від значень f_1 і f_2 під час переходу сигналу на вході C з 1 у 0 можна відобразити у вигляді системи підграфів (рис. А1-4.23, *а*). У разі використання елементів АБО-НЕ застосовують систему рис. А1-4.23, *б*.

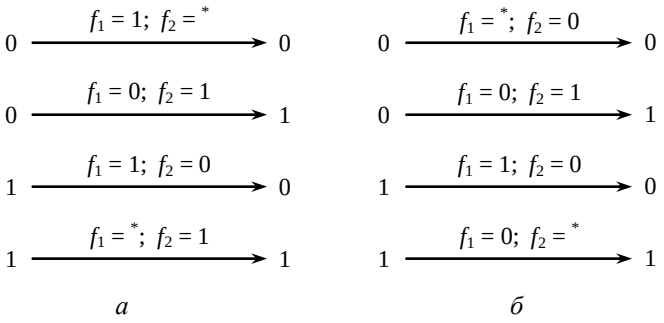


Рис. А1-4.23. Системи підграфів переходів ЗЕ:
а — для елементів І-НЕ; *б* — для елементів АБО-НЕ

Тут знаком * відзначено довільні значення функцій f_1 і f_2.

Для розглянутого тригера одержуємо повну таблицю переходів (табл. А1-4.16), побудовану відповідно до табл. А1-4.15.

На підставі рис. А1-4.23 заповнюємо в табл. А1-4.16 графи для f_1 і f_2, аналізуючи переходи $Q^t \to Q^{t+1}$ у кожному рядку таблиці.

Таблиця А1-4.16

ПОВНА ТАБЛИЦЯ ПЕРЕХОДІВ *JK*-ТРИГЕРА

C^t	J^t	K^t	Q^t	Q^{t+1}	f_1	f_2
0	0	0	0	0	1	*
0	0	0	1	1	*	1
0	0	1	0	0	1	*
0	0	1	1	1	*	1
0	1	0	0	0	1	*
0	1	0	1	1	*	1
0	1	1	0	0	1	*
0	1	1	1	1	*	1
1	0	0	0	0	1	*
1	0	0	1	1	*	1
1	0	1	0	0	1	*
1	0	1	1	0	1	0
1	1	0	0	1	0	1
1	1	0	1	1	*	1
1	1	1	0	1	*	1
1	1	1	1	0	1	0

За допомогою діаграм Вейча (рис. А1-4.24) знаходимо мінімальну диз'юнктивну нормальну форму функцій f_1 і f_2 (індекси t при цьому опускаємо):

$$f_1 = \overline{C} \vee Q \vee \overline{J}; \quad f_2 = \overline{C} \vee \overline{Q} \vee \overline{K}.$$

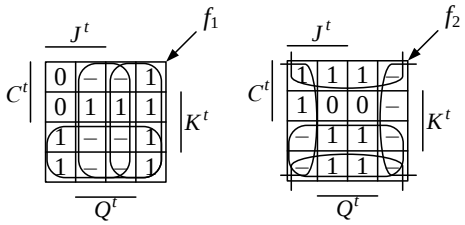


Рис. А1-4.24. Діаграми Вейча

Приводимо отримані функції до вигляду, зручного для реалізації на елементах І-НЕ:

$$f_1 = \overline{\overline{\overline{C} \vee Q \vee \overline{J}}} = \overline{C \cdot \overline{Q} \cdot J};$$

$$f_2 = \overline{\overline{\overline{C} \vee \overline{Q} \vee \overline{K}}} = \overline{C \cdot Q \cdot K}.$$

Отримана схема тригера показана на рис. А1-4.25, а умовне графічне позначення тригера — на рис. А1-4.26 (дві букви Т вказують на використання *MS*-схеми.).

Аналогічно виконується синтез тригерів на основі схеми трьох тригерів. Такі тригери називають тригерами з динамічним записом інформації. Стан тригера визначається інформаційними сигналами, що діють у момент перепаду сигналу на вході *C*. Рисочка на цьому вході вказує, за яким перепадом виконується запис. Приклад позначення на функціональних схемах *D*-тригера з динамічним записом інформації показаний на рис. А1-4.26, *б*.

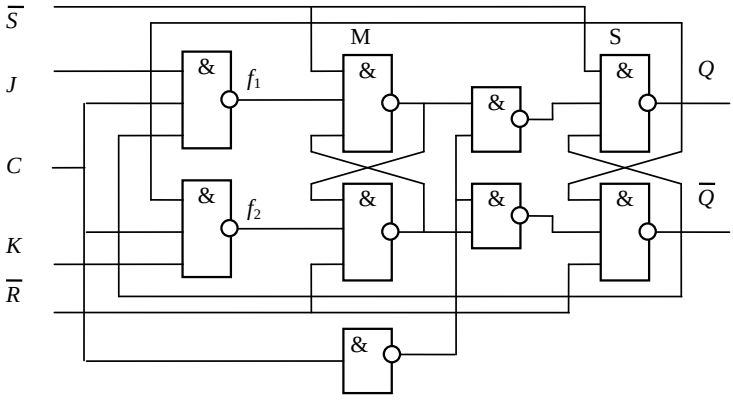


Рис. А1-4.25. Схема *JK*-тригера

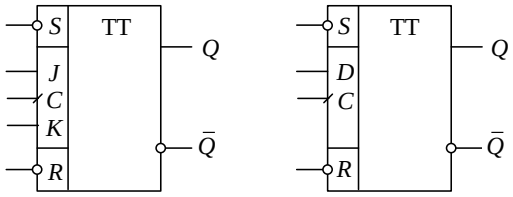


Рис. А1-4.26. Умовне графічне позначення тригерів:
а — *JK*-тригер, виконаний за *MS*-схемою; *б* — *D*-тригер, виконаний за схемою трьох тригерів

Синхронні тригери можуть мати асинхронні входи попередньої установки тригера в 0 (вхід $\overline{R}$) і в 1 (вхід $\overline{S}$). Нулеві сигнали, що надходять на ці входи, незалежно від стану інших входів тригера переключають його в новий стан, тобто мають пріоритет стосовно інших сигналів.

Під час графічного зображення тригерів на функціональних схемах логічні сигнали синхронних тригерів показують в одному полі з тактуючим сигналом *C*, а асинхронні входи — на окремих полях.

А1-4.8. Регістри

Регістром називається упорядкована послідовність запам'ятовуючих елементів (тригерів), призначена для збереження слів і виконання мікрооперацій над ними.

Під мікроопераціями розуміють елементарні дії, що виконуються на регістрах за один такт.

Число розрядів у регістрі називають його довжиною. У *n*-розрядному регістрі може бути записано 2^n різних слів, тобто регістр може перебувати в 2^n різних станах.

Найчастіше на регістрах виконують такі мікрооперації:

y_1 — встановлення вихідного стану (наприклад, нульового);

y_2 — прийом (запис) слова;

y_3 — зсув слова на і розрядів;

Узагальнена логічна структурна схема регістра показана на рис. А1-4.27.

На схемі (рис. А1-4.27) прийняті такі позначення:

КС — комбінаційна схема;

y_i — управляючі сигнали;

x_i — інформаційні входи регістра;

z_i — інформаційні виходи регістра;
A і B — інформаційні входи тригерів;
C — тактуючі входи.

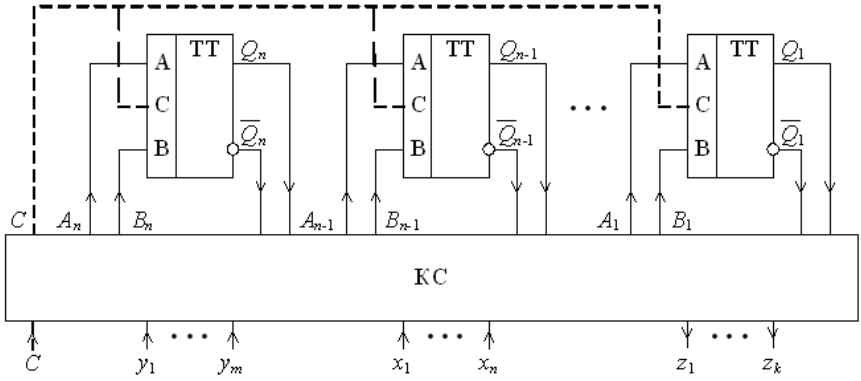


Рис. А1-4.27. Загальна структурна схема регістра

Складність КС залежить від типу мікрооперацій, що виконуються на регістрі, та від типу тригерів. Однак КС може бути і відсутня.

Регістри будуються на тригерах різного типу, наприклад, на *RS*-, *JK*-, *D*- і *T*-тригерах. Залежно від типу мікрооперацій, що виконуються на регістрі, для побудови регістрів використовують тригери з різною внутрішньою організацією (синхронні та асинхронні, із внутрішньою затримкою і без неї).

Регістри, на яких виконуються мікрооперації зсуву, називаються *зсувними*. Зсув слова може бути здійснений вліво (убік старших розрядів) або вправо (убік молодших розрядів) на i розрядів одночасно, де $i = \overline{1, n-1}$ (де n — довжина регістру). Регістри, що мають ланцюги як лівого, так і правого зсуву, називаються *реверсивними*.

Зазвичай під час виконання мікрооперацій усі розряди регістра працюють однаково.

Мікрооперація y_1 полягає в установці кожного розряду регістру в 0 або в 1. Для виконання цієї мікрооперації доцільно використовувати асинхронні входи тригерів R і S.

Під час виконання мікрооперації y_2 у i-й розряд регістра записується цифра x_i, тобто

$$Q_i^{t+1} = x_i^t,$$

де Q_i^{t+1} — стан i-го тригера в момент часу $(t + 1)$, що визначає момент часу після виконання мікрооперації; x_i — значення сигналу на

i-му вході регістра в момент часу *t*, що визначає момент часу перед виконанням мікрооперації.

Мікрооперація y_3 зсуву слова на *j* розрядів полягає в такому:

$$Q_i^{t+1} = Q_{i-j}^t \text{ — за зсуву вліво;}$$

$$Q_i^{t+1} = Q_{i+j}^t \text{ — за зсуву вправо.}$$

Проектування регістрів зводиться до вибору типу тригерів і синтезу КС, що формує функції збудження тригерів під час виконання заданих мікрооперацій. Враховуючи, що всі розряди регістру мають однакову схему, то синтез регістру зводиться до синтезу одного розряду, тобто тригера (див. розділ А1-4.7).

Регістр, що виконує три мікрооперації: встановлення нульового стану (y_1), запису даних (y_2) і зсуву на один розряд (y_3), показаний на рис. А1-4.28.

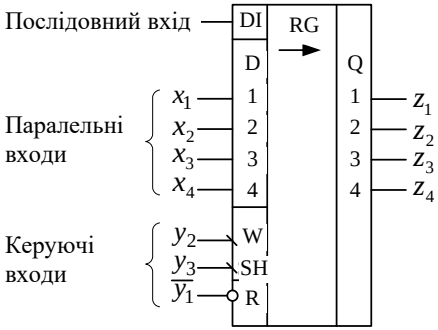


Рис. А1-4.28. Умовне графічне позначення регістру

Під час виконання зсуву в розряд, що звільняється, записуються цифри з послідовного входу *DI*. Запис даних і зсув керуються сигналами на динамічних входах відповідно *W* і *SH*.

А1-4.9. Лічильники

Лічильником називають послідовнісну схему (регістр), призначену для виконання мікрооперацій підрахунку сигналів.

Кількість дозволених станів лічильників називають його *періодом*, *модулем*, або *коефіцієнтом перерахування K*.

Лічильники можуть бути побудовані на основі тригерів зі спеціальними міжрозрядними зв'язками. За *характером мікрооперації ра-*

хунку лічильники поділяються на підсумовуючі, віднімаючі та реверсивні.

Під час надходження чергового вхідного сигналу x вміст підсумовуючого лічильника збільшується на 1, а віднімаючого — зменшується на 1. Реверсивний лічильник виконує мікрооперації підсумовування і віднімання залежно від значення сигналу на керуючому вході y (наприклад, при $y = 1$ виконується підсумовування, а при $y = 0$ — віднімання).

Лічильники класифікуються також *за схемними ознаками*. Для побудови лічильників у потенційній елементній базі застосовуються переважно тригери із внутрішньою затримкою, що дозволяє використовувати на один розряд двійкового лічильника один тригер.

Лічильники поділяються на асинхронні та синхронні.

В *асинхронних* лічильниках тригери спрацьовують не одночасно, оскільки переключення одних тригерів починається тільки після зміни стану інших, що обумовлює низьку швидкодію.

У *синхронних* лічильниках усі тригери переключаються одночасно під дією загального синхронізуючого сигналу, що надходить на тактуючі входи всіх тригерів одночасно.

Лічильники відрізняються способом побудови ланцюгів переносів. У лічильниках *з паралельним переносом* аргументами функцій переносів для кожного розряду є тільки сигнали на виходах тригерів відповідних розрядів. Переноси для всіх розрядів лічильника формуються одночасно, за умови, що всі логічні елементи в схемі мають однаковий час переключення.

Ланцюги *наскрізного переносу* організуються таким чином, щоб функція переносу i-го розряду лічильника була аргументом функції переносу $(i + 1)$-го розряду. У цьому випадку сигнали переносів для кожного розряду лічильника формуються почергово, починаючи з молодших. Лічильники з наскрізним переносом вимагають меншого числа логічних елементів організації ланцюгів переносу, але поступаються лічильникам з рівнобіжним переносом у швидкодії.

У лічильниках *із груповим переносом* розряди розбиваються на групи (наприклад, n розрядів розбиваються на m груп). У межах однієї групи зазвичай організується рівнобіжний перенос, а між групами — послідовний або наскрізний.

Якщо мікрооперація рахунку одиниць виконується у канонічній двійковій системі числення (в однорідній позиційній двійковій системі числення з природним порядком ваг), то він називається лічильником *із природним порядком рахунку*.

Стани чотирьохрозрядного лічильника з природним порядком рахунку наведено в табл. А1-4.17.

Якщо мікрооперація рахунку виконується в неканонічних системах (наприклад, символічних, зі штучним порядком ваг, тощо), порядок рахунку вважається *штучним*. Стани чотирьох розрядного лічильника зі штучним порядком рахунку, що виконує мікрооперацію рахунку в коді Грея, наведено в табл. А1-4.18.

Таблиця А1-4.17

ТАБЛИЦЯ СТАНІВ ЛІЧИЛЬНИКА З ПРИРОДНИМ ПОРЯДКОМ РАХУНКУ

Кількість рахункових сигналів	Стан лічильника	
	підсумовуючого	віднімаючого
0	0000	0000
1	0001	1111
2	0010	1110
3	0011	1101
4	0100	1100
5	0101	1011
6	0110	1010
7	0111	1001
8	1000	1000
9	1001	0111
10	1010	0101
11	1011	0101
12	1100	0100
13	1101	0011
14	1110	0010
15	1111	0001
16	0000	0000
...	...	...

Таблиця А1-4.18

ТАБЛИЦЯ СТАНІВ ЛІЧИЛЬНИКА ЗІ ШТУЧНИМ ПОРЯДКОМ РАХУНКУ

Кількість рахункових сигналів	Стан лічильника
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000
16	0000
...	...

З табл. А1-4.17 і А1-4.18 видно, що стани лічильників повторюються з періодом 2^n. Для скорочення таблиць станів у них заносять тільки стани для одного періоду.

Найпростіші схеми лічильників із природним порядком рахунку побудовані на основі тригерів з рахунковим входом (*T*-тригерів і *JK*-тригерів).

На рис. А1-4.29 зображено узагальнену структуру синхронного лічильника на *T*-тригерах. Комбінаційна схема КС формує сигнали переносів f_i, що надходять на рахункові входи *i*-х тригерів. На вхід

x поступають сигнали, що підраховуються, а вхід y налагоджує лічильних на додавання або віднімання одиниць.

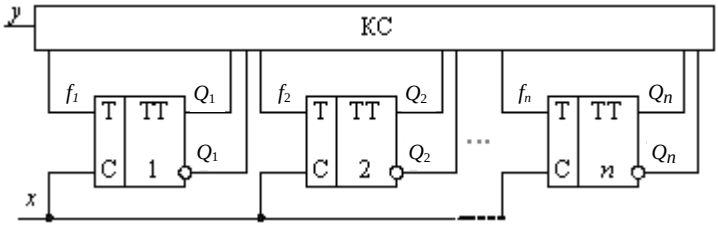


Рис. А1-4.29. Синхронний лічильник

У *JK*-тригерах рахунковий вхід організується з'єднанням входів *J* і *K*. З табл. А1-4.17 виходить, що переключення тригера молодшого розряду здійснюється з приходом кожного рахункового сигналу, а інших тригерів — тільки в тому випадку, якщо всі тригери молодших розрядів встановлені в 1 (підсумовуючий лічильник) або в 0 (віднімаючий лічильник).

Отже, для підсумовуючих лічильників із природним порядком рахунку по модулю 2^n, що має ланцюги паралельного переносу, перемикальні функції f_i мають вигляд:

$$f_i = Q_1 \cdot Q_2 \cdot \ldots \cdot Q_{i-1},\ (i = \overline{2, n}); \qquad \text{(А1-4.3)}$$

для віднімаючих лічильників:

$$fi = \overline{Q_1} \cdot \overline{Q}_2 \cdot \ldots \cdot \overline{Q}_{i-1},\ (i = \overline{2, n}); \qquad \text{(А1-4.4)}$$

а для реверсивних:

$$fi = Q_1 \cdot Q_2 \cdot \ldots \cdot Q_{i-1} \cdot Y \vee \overline{Q_1} \cdot \overline{Q}_2 \cdot \ldots \cdot \overline{Q}_{i-1} \cdot \overline{Y},\ (i = \overline{2, n}). \qquad \text{(А1-4.5)}$$

Для всіх типів лічильників $f_1 = 1$.

Функції (А1-4.3 — А1-4.5) для лічильників з наскрізним переносом можуть бути подані відповідно таким чином:

$$fi = f_{i-1} \cdot Q_{i-1}, \qquad (i = \overline{2, n});$$
$$fi = f_{i-1} \cdot \overline{Q}_{i-1}, \qquad (i = \overline{2, n});$$
$$fi = f_i' \vee f_i'', \qquad (i = \overline{2, n}),$$

де $f_i = 1,\ f_2' = Q_1 \cdot Y,\ f_2'' = \overline{Q_1} \cdot \overline{Y},\ f\dot{i} = f_{i-1}' \cdot Q_{i-1},\ f_i'' = f_{i-1}'' \cdot \overline{Q}_{i-1}, (i = \overline{3, n}).$

Як приклад на рис. А1-4.30 показана функціональна схема підсумовуючого лічильника з паралельним переносом на *Т*-тригерах, а на рис. А1-4.31– реверсивного лічильника з наскрізним переносом на *JK*-тригерах (для $n = 4$).

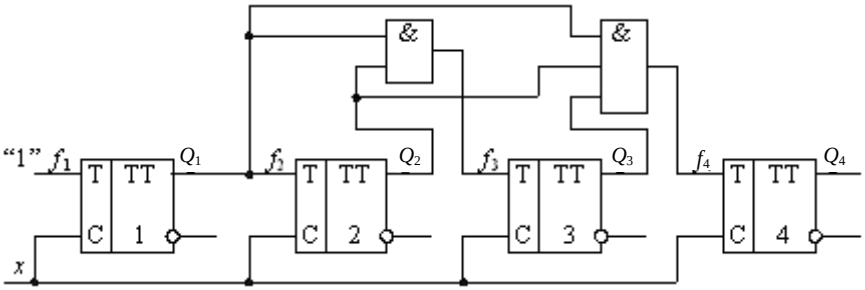


Рис. А1-4.30. Підсумовуючий лічильник з паралельним переносом

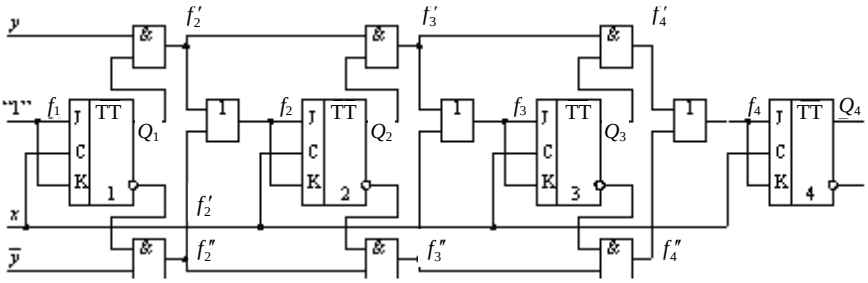


Рис. А1-4.31. Реверсивний лічильник з наскрізним переносом

Для побудови синхронних лічильників з довільним коефіцієнтом перерахування *K* і будь-яким порядком рахунку необхідно виконати такі дії.

1. Визначити число розрядів *n* лічильника

$$n =]\log_2 K[,$$

де $]\log_2 K[$ — ціле, не менше за $\log_2 K$.

2. Обрати тип тригерів.

3. Скласти таблицю переходів лічильника, загальний вигляд якої відповідає табл. А1-4.19. У відповідні стовпці таблиці переходів записують коди станів лічильника Q_i^t до надходження чергового рахункового сигналу і коди станів лічильника Q_i^{t+1} після надходження чергового сигналу.

4. Для кожного *i*-го розряду лічильника записати в кожному *j*-му рядку таблиці переходів лічильника значення сигналів на інформаційних входах f_i тригерів (функції збудження тригерів), що забезпечують переключення тригерів зі стану Q_i^t у стан Q_i^{t+1}, відповідно до таблиці переходів тригеру заданого типу.

5. Записати СДНФ функцій збудження тригерів f_i, аргументами яких є значення кодів стану лічильника $Q_n^t, Q_{n-1}^t, ..., Q_1^t$.

6. Одержати МДНФ функцій збудження тригерів f_i.

7. Отримати операторні форми подання функцій збудження тригерів f_i у заданому елементному базисі.

8. Побудувати функціональну схему лічильника.

Приклад УГП лічильника показано на рис. А1-4.32.

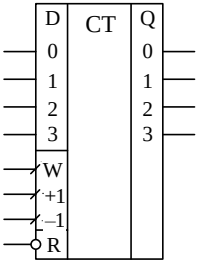


Рис. А1-4.32. Реверсивний 4-розрядний лічильник

Приклад

Завдання. Побудувати лічильник з коефіцієнтом перерахування $K = 6$. Для побудови лічильника застосувати *T*-тригери.

Виконання завдання

*В*изначаємо число розрядів *n* лічильника

$$n =]\log_2 6[= 3.$$

Обираємо тригери *T*-типу.

Вважаємо, що дозволеними для лічильника є стани:

000, 001, 010, 011, 110, 111.

Будуємо таблицю переходів станів лічильника (табл. А1-4.19). Відповідно до заданого коефіцієнту перерахування $K = 6$ заповнюємо шість рядків таблиці переходів лічильника.

Визначаємо функції збудження тригерів. Наприклад, для нульового рядка значення T_1, T_2 і T_3 визначені так. Переходи $Q_1^t \to Q_1^{t+1}$, $Q_2^t \to Q_2^{t+1}$, $Q_3^t \to Q_3^{t+1}$ мають відповідно вигляд $0 \to 1$, $0 \to 0$, $0 \to 0$.

Виходячи з графічної таблиці переходів *T*-тригера (рис. А1-3.4, *б*) одержуємо $T_1 = 1$, $T_2 = 0$ і $T_3 = 0$.

Таблиця А1-4.19

ТАБЛИЦЯ ПЕРЕХОДІВ ЛІЧИЛЬНИКА З КОЕФІЦІЄНТОМ ПЕРЕРАХУВАННЯ *K* = 6

Номер попереднього стану	Стан лічильника						Функції збудження тригерів		
	Попередній стан			Наступний стан					
j	Q_3^t	Q_2^t	Q_1^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	T_3	T_2	T_1
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
2	0	1	0	0	1	1	0	0	1
3	0	1	1	1	1	0	1	0	1
4	1	1	0	1	1	1	0	0	1
5	1	1	1	0	0	0	1	1	1

Запишемо СДНФ функцій збудження тригерів T_i:

$$T_1 = 1;$$
$$T_2 = \overline{Q_3^t}\,\overline{Q_2^t}\,\overline{Q_1^t} \vee Q_3^t Q_2^t Q_1^t;$$
$$T_3 = \overline{Q_3^t} Q_2^t Q_1^t \vee Q_3^t Q_2^t Q_1^t.$$

Після мінімізації функцій збудження тригерів (індекси *t* опускаємо) отримаємо:

$$T_1 = 1;$$
$$T_2 = Q_3 Q_1 \vee \overline{Q}_2 Q_1;$$
$$T_3 = Q_2 Q_1.$$

Функціональну схему лічильника зображено на рис. А1-4.33.

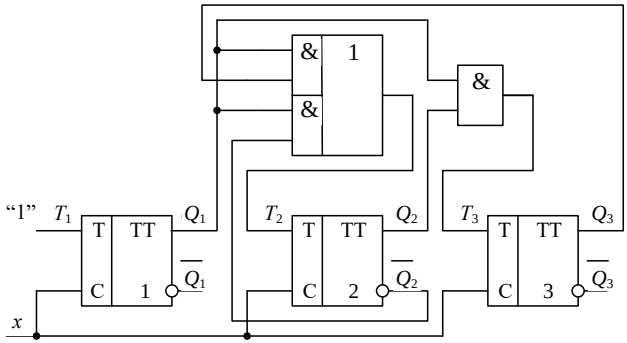


Рис. А1-4.33. Функціональна схема лічильника з коефіцієнтом перерахування *K* = 6

А2. ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

А2-1. Синтез комбінаційних схем

Приклад 2-1.1

Завдання. Побудувати комбінаційні схеми у елементному базисі 3І, 2АБО для функції, заданої таблицею істинності (табл. А2-2.1).

Таблиця А2-1.1

ТАБЛИЦЯ ІСТИННОСТІ

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

Виконання завдання

Використовуючи основні закони булевої алгебри та теорему де Моргана, знаходимо вісім нормальних форм заданої перемикальної функції:

$$y_{\text{ДДНФ}} = \overline{x}_3\overline{x}_2\overline{x}_1 \vee \overline{x}_3 x_2 x_1 \vee x_3\overline{x}_2 x_1 = \quad \text{(І / АБО)}$$

$$= \overline{\overline{x}_3\overline{x}_2\overline{x}_1 \vee \overline{x}_3 x_2 x_1 \vee x_3\overline{x}_2 x_1} = \overline{\overline{x}_3\overline{x}_2\overline{x}_1} \cdot \overline{\overline{x}_3 x_2 x_1} \cdot \overline{x_3\overline{x}_2 x_1} = \quad \text{(І-НЕ / І-НЕ)}$$

$$= \overline{(x_3 \vee x_2 \vee x_1)\cdot(x_3 \vee \overline{x}_2 \vee \overline{x}_1)\cdot(\overline{x}_3 \vee x_2 \vee \overline{x}_1)} = \quad \text{(АБО / І-НЕ)}$$

$$= \overline{x_3 \vee x_2 \vee x_1} \vee \overline{x_3 \vee \overline{x}_2 \vee \overline{x}_1} \vee \overline{\overline{x}_3 \vee x_2 \vee \overline{x}_1}. \quad \text{(АБО-НЕ / АБО)}$$

$$y_{\text{ДКНФ}} = (x_3 \vee x_2 \vee \overline{x}_1)\cdot(x_3 \vee \overline{x}_2 \vee x_1)\cdot(\overline{x}_3 \vee x_2 \vee x_1)\times$$

$$\times(\overline{x}_3 \vee \overline{x}_2 \vee x_1)\cdot(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_1) = \quad \text{(АБО / І)}$$

$$= \overline{(x_3 \vee x_2 \vee \overline{x}_1)\cdot(x_3 \vee \overline{x}_2 \vee x_1)\cdot(\overline{x}_3 \vee x_2 \vee x_1)\cdot(\overline{x}_3 \vee \overline{x}_2 \vee x_1)\cdot(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_1)} =$$

$$= \overline{(x_3 \vee x_2 \vee \overline{x}_1)} \vee \overline{(x_3 \vee \overline{x}_2 \vee x_1)} \vee \overline{(\overline{x}_3 \vee x_2 \vee x_1)} \vee$$

$$\vee \overline{\overline{x}_3 \vee \overline{x}_2 \vee x_1} \vee \overline{\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_1} \vee \overline{x_3 \vee x_2 \vee x_1} = \quad \text{(АБО-НЕ / АБО-НЕ)}$$

$$= \overline{\overline{x}_3\overline{x}_2\overline{x}_1 \vee \overline{x}_3 x_2 x_1 \vee x_3\overline{x}_2 x_1 \vee x_3 x_2\overline{x}_1 \vee x_3 x_2 x_1} = \quad \text{(І / АБО-НЕ)}$$

$$= \overline{\overline{x}_3\overline{x}_2\overline{x}_1} \cdot \overline{\overline{x}_3 x_2\overline{x}_1} \cdot \overline{x_3\overline{x}_2 x_1} \cdot \overline{x_3 x_2\overline{x}_1} \cdot \overline{x_3 x_2 x_1}. \quad \text{(І-НЕ / І)}$$

Заданому елементному базису задовольняють нормальні форми І / АБО та АБО / І. Запишемо їх в операторних формах:

$$y_{\text{І/АБО}} = (\overline{x_3}\,\overline{x_2}\,\overline{x_1}) \vee (\overline{x_3}x_2x_1)) \vee (x_3\overline{x_2}x_1);$$

$$y_{\text{АБО/І}} = (((x_3 \vee x_2) \vee \overline{x_1}) \cdot ((x_3 \vee \overline{x_2}) \vee x_1) \cdot ((\overline{x_3} \vee x_2) \vee x_1)) \cdot ((\overline{x_3} \vee \overline{x_2}) \vee x_1) \cdot ((\overline{x_3} \vee \overline{x_2}) \vee \overline{x_1}).$$

Побудуємо відповідні знайденим операторним формам комбінаційні схеми (рис. А2-1.1, *а*, *б*).

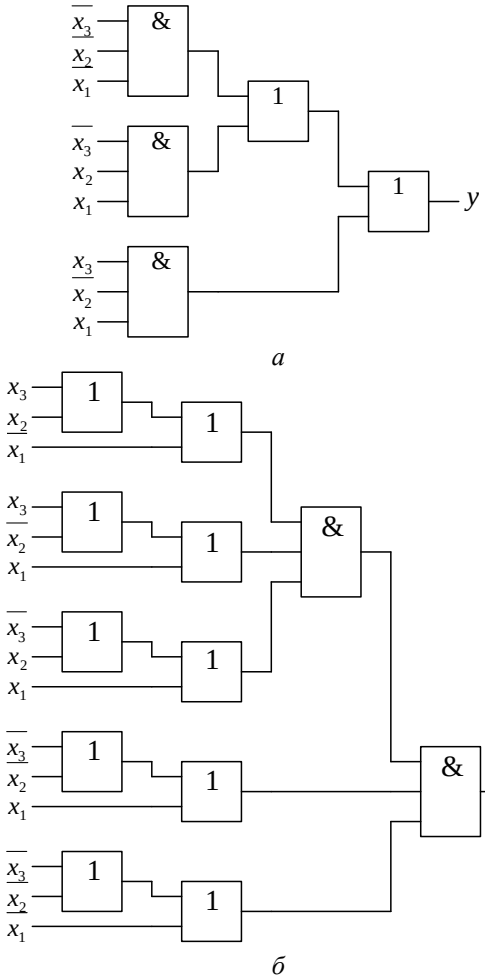


Рис. А2-1.1. Комбінаційні схеми:
а — на елементах І, АБО; *б* — на елементах АБО, І

Приклад 2-1.2

Завдання. Подати функцію *f* в нормальних формах алгебр Буля, Шеффера, Пірса, Жегалкіна. Показати належність функції *f* до п'яти передповних класів.

$$f = 1 \vee 2 \vee 3 \vee 7 \vee 9 \vee 10 \vee 12 \vee 13 \vee 15.$$

Виконання завдання

Подамо функцію *f* в ДДНФ і ДКНФ:

$$f_{\text{ДДНФ}} = (\overline{x}_4 \overline{x}_3 \overline{x}_2 x_1) \vee (\overline{x}_4 \overline{x}_3 x_2 \overline{x}_1) \vee (\overline{x}_4 \overline{x}_3 x_2 x_1) \vee$$
$$\vee (\overline{x}_4 x_3 x_2 x_1) \vee (x_4 \overline{x}_3 \overline{x}_2 x_1) \vee (x_4 \overline{x}_3 x_2 \overline{x}_1) \vee$$
$$\vee (x_4 x_3 \overline{x}_2 \overline{x}_1) \vee (x_4 x_3 \overline{x}_2 x_1) \vee (x_4 x_3 x_2 x_1);$$
$$f_{\text{ДКНФ}} = (x_4 \vee x_3 \vee x_2 \vee x_1) \cdot (x_4 \vee \overline{x}_3 \vee x_2 \vee x_1) \cdot (x_4 \vee \overline{x}_3 \vee x_2 \vee \overline{x}_1) \wedge$$
$$\wedge (x_4 \vee \overline{x}_3 \vee \overline{x}_2 \vee x_1) \cdot (\overline{x}_4 \vee x_3 \vee x_2 \vee x_1) \cdot (\overline{x}_4 \vee x_3 \vee \overline{x}_2 \vee \overline{x}_1) \wedge$$
$$\wedge (\overline{x}_4 \vee \overline{x}_3 \vee \overline{x}_2 \vee x_1).$$

Подамо функцію *f* в алгебрі Шеффера.

Вихідною у цьому випадку є ДДНФ перемикальної функції. Перетворення засновані на теоремі де Моргана. Отримаємо:

$$f = \overline{(\overline{x}_4 \overline{x}_3 \overline{x}_2 x_1) \vee (\overline{x}_4 \overline{x}_3 x_2 \overline{x}_1) \vee (\overline{x}_4 \overline{x}_3 x_2 x_1) \vee}$$
$$\overline{\vee (\overline{x}_4 x_3 x_2 x_1) \vee (x_4 \overline{x}_3 \overline{x}_2 x_1) \vee (x_4 \overline{x}_3 x_2 \overline{x}_1) \vee}$$
$$\overline{\vee (x_4 x_3 \overline{x}_2 \overline{x}_1) \vee (x_4 x_3 \overline{x}_2 x_1) \vee (x_4 x_3 x_2 x_1)} =$$
$$= \overline{(x_4 x_3 x_2 x_1)} \cdot \overline{(x_4 x_3 x_2 \overline{x}_1)} \cdot \overline{(x_4 x_3 \overline{x}_2 x_1)} \wedge$$
$$\wedge \overline{(x_4 \overline{x}_3 x_2 x_1)} \cdot \overline{(x_4 \overline{x}_3 \overline{x}_2 x_1)} \cdot \overline{(x_4 \overline{x}_3 x_2 \overline{x}_1)} \wedge$$
$$\wedge \overline{(x_4 x_3 \overline{x}_2 x_1)} \cdot \overline{(x_4 x_3 \overline{x}_2 x_1)} \cdot \overline{(x_4 x_3 x_2 x_1)} =$$
$$= (\overline{x}_4 / \overline{x}_3 / \overline{x}_2 / x_1) / (\overline{x}_4 / \overline{x}_3 / x_2 / \overline{x}_1) / (\overline{x}_4 / \overline{x}_3 / x_2 / x_1) /$$
$$/ (\overline{x}_4 / x_3 / x_2 / x_1) / (x_4 / \overline{x}_3 / \overline{x}_2 / x_1) / (x_4 / \overline{x}_3 / x_2 / \overline{x}_1) /$$
$$/ (x_4 / x_3 / \overline{x}_2 / \overline{x}_1) / (x_4 / x_3 / \overline{x}_2 / x_1) / (x_4 / x_3 / x_2 / x_1).$$

Подамо функцію f в алгебрі Пірса.

Вихідною у цьому випадку є ДКНФ перемикальної функції. Застосовуючи теорему де Моргана, отримаємо:

$$f = \overline{\overline{(x_4 \vee x_3 \vee x_2 \vee x_1)\cdot(x_4 \vee \overline{x_3} \vee x_2 \vee x_1)\cdot(x_4 \vee \overline{x_3} \vee x_2 \vee \overline{x_1})} \wedge$$
$$\wedge \overline{\overline{(x_4 \vee \overline{x_3} \vee \overline{x_2} \vee x_1)\cdot(\overline{x_4} \vee x_3 \vee x_2 \vee x_1)\cdot(\overline{x_4} \vee x_3 \vee \overline{x_2} \vee \overline{x_1})} \wedge$$
$$\wedge \overline{\overline{(\overline{x_4} \vee \overline{x_3} \vee \overline{x_2} \vee x_1)}} =$$
$$= \overline{(\overline{x_4 \vee x_3 \vee x_2 \vee x_1})} \vee \overline{(x_4 \vee \overline{x_3} \vee x_2 \vee x_1)} \vee \overline{(x_4 \vee \overline{x_3} \vee x_2 \vee \overline{x_1})} \vee$$
$$\vee \overline{(x_4 \vee \overline{x_3} \vee \overline{x_2} \vee x_1)} \vee \overline{\overline{(x_4 \vee x_3 \vee x_2 \vee x_1)}} \vee \overline{(x_4 \vee x_3 \vee \overline{x_2} \vee \overline{x_1})} \vee$$
$$\vee \overline{\overline{(\overline{x_4} \vee \overline{x_3} \vee \overline{x_2} \vee x_1)}} =$$
$$= (x_4 \downarrow x_3 \downarrow x_2 \downarrow x_1) \downarrow (x_4 \downarrow \overline{x_3} \downarrow x_2 \downarrow x_1) \downarrow (x_4 \downarrow \overline{x_3} \downarrow x_2 \downarrow \overline{x_1}) \downarrow$$
$$\downarrow (x_4 \downarrow \overline{x_3} \downarrow \overline{x_2} \downarrow x_1) \downarrow (\overline{x_4} \downarrow x_3 \downarrow x_2 \downarrow x_1) \downarrow (\overline{x_4} \downarrow x_3 \downarrow \overline{x_2} \downarrow \overline{x_1}) \downarrow$$
$$\downarrow (\overline{x_4} \downarrow \overline{x_3} \downarrow \overline{x_2} \downarrow x_1).$$

Подання функції f в алгебрі Жегалкіна.

Вихідною формою є ДДНФ заданої перемикальної функції. Виконуємо заміну $\vee$ на $\oplus$ та застосовуємо аксіоми алгебри Жегалкіна:

$$f = ((x_4 \oplus 1)\cdot(x_3 \oplus 1)\cdot(x_2 \oplus 1)\cdot x_1) \oplus ((x_4 \oplus 1)\cdot(x_3 \oplus 1)\cdot x_2 \cdot (x_1 \oplus 1)) \oplus$$
$$\oplus((x_4 \oplus 1)\cdot(x_3 \oplus 1)\cdot x_2 \cdot x_1) \oplus ((x_4 \oplus 1)\cdot x_3 \cdot x_2 \cdot x_1) \oplus$$
$$\oplus(x_4 \cdot (x_3 \oplus 1)\cdot(x_2 \oplus 1)\cdot x_1) \oplus (x_4 \cdot (x_3 \oplus 1)\cdot x_2 \cdot (x_1 \oplus 1)) \oplus$$
$$\oplus(x_4 \cdot x_3 \cdot (x_2 \oplus 1)\cdot(x_1 \oplus 1)) \oplus (x_4 \cdot x_3 \cdot (x_2 \oplus 1)\cdot x_1) \oplus (x_4 \cdot x_3 \cdot x_2 \cdot x_1) =$$
$$= \cancel{x_4x_3x_2x_1} \oplus \cancel{x_4x_3x_1} \oplus \cancel{x_4x_2x_1} \oplus \cancel{x_4x_1} \oplus \cancel{x_3x_2x_1} \oplus x_3x_1 \oplus \cancel{x_2x_1} \oplus x_1 \oplus$$
$$\oplus \cancel{x_4x_3x_2x_1} \oplus \cancel{x_4x_3x_2} \oplus \cancel{x_3x_2x_1} \oplus x_3x_2 \oplus \cancel{x_4x_2x_1} \oplus \cancel{x_4x_2} \oplus \cancel{x_2x_1} \oplus x_2 \oplus$$
$$\oplus \cancel{x_4x_3x_2x_1} \oplus \cancel{x_4x_2x_1} \oplus \cancel{x_3x_2x_1} \oplus x_2x_1 \oplus \cancel{x_4x_3x_2x_1} \oplus \cancel{x_3x_2x_1} \oplus$$
$$\oplus \cancel{x_4x_3x_2x_1} \oplus \cancel{x_4x_3x_1} \oplus \cancel{x_4x_2x_1} \oplus \cancel{x_4x_1} \oplus$$
$$\oplus \cancel{x_4x_3x_2x_1} \oplus \cancel{x_4x_3x_2} \oplus x_4x_2x_1 \oplus \cancel{x_4x_2} \oplus$$
$$\oplus \cancel{x_4x_3x_2x_1} \oplus \cancel{x_4x_3x_1} \oplus x_4x_3x_2 \oplus x_4x_3 \oplus \cancel{x_4x_3x_2x_1} \oplus \cancel{x_4x_3x_1} \oplus x_4x_3x_2x_1 =$$
$$= x_3x_1 \oplus x_1 \oplus x_3x_2 \oplus x_2 \oplus x_2x_1 \oplus x_4x_2x_1 \oplus x_4x_3x_2 \oplus x_4x_3 \oplus x_4x_3x_2x_1.$$

Визначемо приналежність перемикальної функції f до п'яти передповних класів:

K_1: $f(0, 0, 0, 0) = 0$ — зберігає 0;

K_2: $f(1, 1, 1, 1) = 1$ — зберігає 1;

K_3: $f(0, 0, 1, 1) = 1$, $f(1, 1, 0, 0) = 1$ — не самодвоїста;

K_4: $f(0,1,0,1) = 0$, $f(0, 0, 0, 1) = 1$, $f(0,1,0,1) < f(0, 0, 0, 1)$ — не монотонна;

K_5: f не виражається лінійним поліномом Жегалкіна — не лінійна.

Результати зведемо до таблиці:

f	K_1	K_2	K_3	K_4	K_5
	+	+	–	–	–

А2-2. Мінімізація перемикальних функцій

Приклад 2-2.1

Завдання. Виконати мінімізацію перемикальної функції f, заданої таблицею істинності (табл. А2-2.1), методом Квайна. Побудувати комбінаційну схему в елементному базисі 3І-НЕ.

Таблиця А2-2.1

ТАБЛИЦЯ ІСТИННОСТІ

x_4	x_3	x_2	x_1	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Виконання завдання

Запишемо задану перемикальну функцію у вигляді ДДНФ. Для зручності записуємо конституенти у стовпець. Виконуємо всі попарні склеювання згідно з теоремою неповного склеювання. У другому стовпці розміщуються терми третього рангу, далі терми другого рангу. Подальше склеювання неможливе. На наступному етапі виконуємо поглинання термів. Терми, що залишились не поглинутими, позначаємо овалами.

$\overline{x_4}\overline{x_3}\overline{x_2}\overline{x_1}$ (1) $\overline{x_4}\overline{x_3}\overline{x_2}x_1$ (2) $\overline{x_4}\overline{x_3}x_2\overline{x_1}$ (3) $\overline{x_4}\overline{x_3}x_2x_1$ (4) $\overline{x_4}x_3\overline{x_2}\overline{x_1}$ (5) $\overline{x_4}x_3\overline{x_2}x_1$ (6) $\overline{x_4}x_3x_2\overline{x_1}$ (7)

$x_4x_3x_2$ (1-2) $\overline{x_4}\overline{x_3}\overline{x_1}$ (1-3) $\overline{x_3}\overline{x_2}\overline{x_1}$ (1-5) $x_4x_2x_1$ (2-4) $\overline{x_3}\overline{x_2}\overline{x_1}$ (3-6) $x_3\overline{x_2}x_1$ (4-7) $\overline{x_4}\overline{x_3}\overline{x_1}$ (5-6)

$\overline{x_3}\overline{x_1}$ (1-3, 5-6) x_3x_1 (1-5, 3-6)

У результаті отримаємо скорочену ДНФ:

$$f = \overline{x_3}\overline{x_1} \vee \overline{x_4}\overline{x_3}\overline{x_2} \vee \overline{x_4}\overline{x_2}x_1 \vee x_3\overline{x_2}x_1.$$

Для визначення мінімальної ДНФ будуємо таблицю покриття (табл. А2-1.3).

Таблиця А2-2.2

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти						
	$\overline{x_4}\overline{x_3}\overline{x_2}\overline{x_1}$	$\overline{x_4}\overline{x_3}\overline{x_2}x_1$	$\overline{x_4}\overline{x_3}x_2\overline{x_1}$	$\overline{x_4}x_3\overline{x_2}x_1$	$x_4\overline{x_3}\overline{x_2}\overline{x_1}$	$x_4\overline{x_3}x_2\overline{x_1}$	$x_4x_3\overline{x_2}x_1$
$\overline{x_4}\overline{x_3}\overline{x_2}$	∨	∨					
$\overline{x_4}\overline{x_2}x_1$		Ⓥ					
$x_3\overline{x_2}x_1$				Ⓥ			Ⓥ
$\overline{x_3}\overline{x_1}$	Ⓥ		Ⓥ		Ⓥ	Ⓥ	

Знаходимо ядро функції — сукупність імплікант відповідних однократно покритим конституентам. У даному випадку ядро складають імпліканти $\{\overline{x_3}\,\overline{x_1},\ x_3\overline{x_2}x_1\}$.

Як МДНФ вибираємо

$$f_{\text{МДНФ}} = \overline{x_3}\,\overline{x_1} \vee \overline{x_4}\,\overline{x_2}x_1 \vee x_3\overline{x_2}x_1.$$

Для побудови комбінаційної схеми у заданому елементному базисі отримуємо нормальну форму І-НЕ / І-НЕ.

$$\begin{aligned} f &= \overline{x_3}\,\overline{x_1} \vee \overline{x_4}\,\overline{x_2}x_1 \vee x_3\overline{x_2}x_1 = \\ &= \overline{\overline{x_3}\,\overline{x_1}} \vee \overline{\overline{x_4}\,\overline{x_2}x_1} \vee \overline{x_3\overline{x_2}x_1} = \\ &= \overline{\overline{x_3}\,\overline{x_1}} \cdot \overline{\overline{x_4}\,\overline{x_2}x_1} \cdot \overline{x_3\overline{x_2}x_1}. \end{aligned} \qquad \text{(І-НЕ / І-НЕ).}$$

Отримана нормальна форма дозволяє побудувати схему на логічних елементах із заданою кількість входів, тому у даному випадку нормальна форма співпадає із операторною.

Комбінаційна схема, що реалізує задану функцію *f*, зображена на рис. А2-2.1.

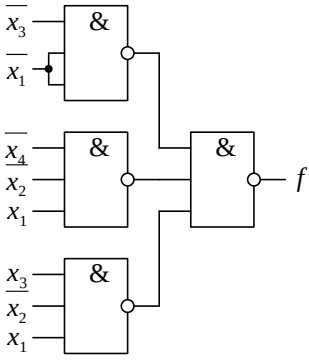


Рис. А2-2.1. Комбінаційна схема

Приклад 2-2.2

Завдання. Виконати мінімізацію перемикальної функції, заданої таблицею істинності (табл. А2-2.3), методом Квайна—Мак-Класкі. Реалізувати задану перемикальну функцію на логічних елементах І-НЕ.

Таблиця А2-2.3

ТАБЛИЦЯ ІСТИННОСТІ

x_4	x_3	x_2	x_1	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Виконання завдання

Виходячи з таблиці істинності функції записуємо комплекс кубів K^0. Конституенти одиниці поєднуємо у групи з однаковою кількістю одиниць.

Після виконання всіх можливих попарних склеювань, згідно з теоремою неповного склеювання, отримуємо комплекс кубів K^1, де терми згруповані за місцем розташування загальної змінної *X*. Подальше склеювання неможливе. Виконуємо поглинання термів, та формуємо *Z*-покриття, що відповідає СДНФ. Терми, що складають *Z*-покриття, позначимо овалами.

~~0000~~ (1) — 00X0 (0-1)

~~0010~~ (2) — 10X1 (4-5)

~~0011~~ (3) — 0X10 (1-3)

~~0110~~ (4) — X011 (2-5)

~~1001~~ (5) — X110 (3-6)

~~1011~~ (6) — 001X (1-2)

~~1110~~ (7)

Цифровий запис отриманої СДНФ заданої перемикальної функції має вигляд:

$$f_{\text{СДНФ}} = 00\text{X}0 \vee 10\text{X}1 \vee 0\text{X}10 \vee \text{X}011 \vee \text{X}110 \vee 001\text{X}.$$

Для видалення надлишкових імплікант будуємо таблицю покриття (табл. А2-2.4).

Таблиця А2-2.4

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти						
	0000	0010	0011	0110	1001	1011	1110
X011			Ⓥ			∨	
X110				Ⓥ			Ⓥ
0X10		∨		∨			
00X0	Ⓥ	Ⓥ					
10X1					Ⓥ	Ⓥ	
001X		∨	∨				

Ядро перемикальної функції складають імпліканти:

$$\{x_3 x_2 \overline{x_1},\ \overline{x_4}\,\overline{x_3}\,\overline{x_1},\ x_4 \overline{x_3} x_1\}.$$

Виходячи з таблиці як МДНФ функції f можна обрати

$$f_{\text{МДНФ}} = \overline{x_3} x_2 x_1 \vee x_3 x_2 \overline{x_1} \vee \overline{x_4}\,\overline{x_3}\,\overline{x_1} \vee x_4 \overline{x_3} x_1.$$

Виходячи з МДНФ отримуємо необхідну нормальну форму подання заданої перемикальної функції:

$$f = \overline{x_3} x_2 x_1 \vee x_3 x_2 \overline{x_1} \vee \overline{x_4}\,\overline{x_3}\,\overline{x_1} \vee x_4 \overline{x_3} x_1 =$$

$$= \overline{\overline{x_3} x_2 x_1 \vee x_3 x_2 \overline{x_1} \vee \overline{x_4}\,\overline{x_3}\,\overline{x_1} \vee x_4 \overline{x_3} x_1} =$$

$$= \overline{\overline{x_3} x_2 x_1} \cdot \overline{x_3 x_2 \overline{x_1}} \cdot \overline{\overline{x_4}\,\overline{x_3}\,\overline{x_1}} \cdot \overline{x_4 \overline{x_3} x_1}. \qquad \text{(І-НЕ / І-НЕ).}$$

Для побудови комбінаційної схеми подамо функцію f в операторній формі:

$$f = \overline{\overline{(\overline{x_3} x_2 x_1)} \cdot \overline{(x_3 x_2 \overline{x_1})} \cdot \overline{(\overline{x_4}\,\overline{x_3}\,\overline{x_1})}} \cdot \overline{(x_4 \overline{x_3} x_1)}.$$

Комбінаційна схема, що реалізує функцію f у заданому елементному базисі, зображена на рис. А2-2.2.

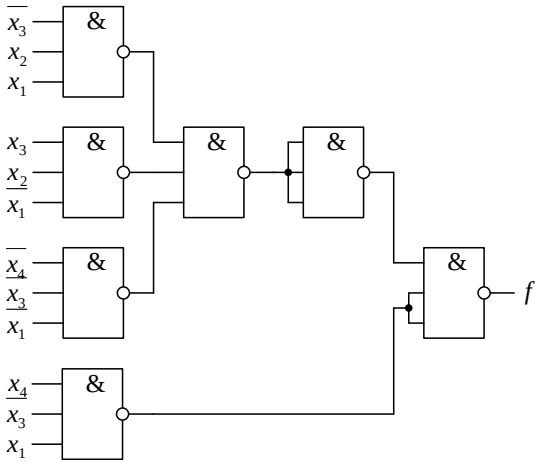


Рис. А2-2.2. Комбінаційна схема

Приклад 2-2.3

Завдання. Виконати мінімізацію перемикальної функції *f*, заданої таблицею істинності (табл. А2-2.5), методом невизначених коефіцієнтів. Реалізувати задану перемикальну функцію в елементному базисі ЗАБО-НЕ.

Таблиця А2-2.5

ТАБЛИЦЯ ІСТИННОСТІ

x_4	x_3	x_2	x_1	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Виконання завдання

Для реалізації перемикальної функцію в елементному базисі АБО-НЕ необхідно отримати МКНФ перемикальної функції.

Складаємо таблицю коефіцієнтів для заперечення функції (табл. А2-2.6).

Відповідно до нулевих значень функції викреслюємо коефіцієнти та виконуємо поглинання імплікант.

У даному випадку всі імпліканти входять в ядро функції. Отже, МКНФ має вигляд:

$$f_{\text{МКНФ}} = (\overline{x_4} \vee x_1) \cdot (x_3 \vee \overline{x_2}) \cdot (\overline{x_2} \vee x_1) \cdot (\overline{x_3} \vee x_2 \vee \overline{x_1}).$$

Таблиця А2-2.6

ТАБЛИЦЯ КОЕФІЦІЄНТІВ

f	x_4	x_3	x_2	x_1	x_4x_3	x_4x_2	x_4x_1	x_3x_2	x_3x_1	x_2x_1	$x_4x_3x_2$	$x_4x_3x_1$	$x_4x_2x_1$	$x_3x_2x_1$	$x_4x_3x_2x_1$
0	0	0	0	0	00	00	00	00	00	00	000	000	000	000	0000
0	0	0	0	1	00	00	01	00	01	01	000	001	001	001	0001
1	0	0	1	0	00	01	00	01	00	10	001	000	010	010	0010
1	0	0	1	1	00	01	01	01	01	11	001	001	011	011	0011
0	0	1	0	0	01	00	00	10	10	00	010	010	000	100	0100
1	0	1	0	1	01	00	01	10	11	01	010	011	001	101	0101
1	0	1	1	0	01	01	00	11	10	10	011	010	010	110	0110
0	0	1	1	1	01	01	01	11	11	11	011	011	011	111	0111
1	1	0	0	0	10	10	10	00	00	00	100	100	100	000	1000
0	1	0	0	1	10	10	11	00	01	01	100	101	101	001	1001
1	1	0	1	0	10	11	10	01	00	10	101	100	110	010	1010
1	1	0	1	1	10	11	11	01	01	11	101	101	111	011	1011
1	1	1	0	0	11	10	10	10	10	00	110	110	100	100	1100
1	1	1	0	1	11	10	11	10	11	01	110	111	101	101	1101
1	1	1	1	0	11	11	10	11	10	10	111	110	110	110	1110
0	1	1	1	1	11	11	11	11	11	11	111	111	111	111	1111

Отримаємо нормальну форму АБО-НЕ/АБО-НЕ:

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

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

$$= \overline{\overline{x_4} \vee x_1} \vee \overline{x_3 \vee \overline{x_2}} \vee \overline{\overline{x_2} \vee x_1} \vee \overline{\overline{x_3} \vee x_2 \vee \overline{x_1}}. \qquad \text{(АБО-НЕ / АБО-НЕ).}$$

Для побудови комбінаційної схеми на логічних елементах 3АБО-НЕ запишемо знайдену форму в операторному вигляді:

$$f = \overline{\overline{(\overline{x_4} \vee x_1) \vee (x_3 \vee \overline{x_2}) \vee (\overline{x_2} \vee x_1)} \vee \overline{(\overline{x_3} \vee x_2 \vee \overline{x_1})}}.$$

Будуємо комбінаційну схему (рис. А2-2.3).

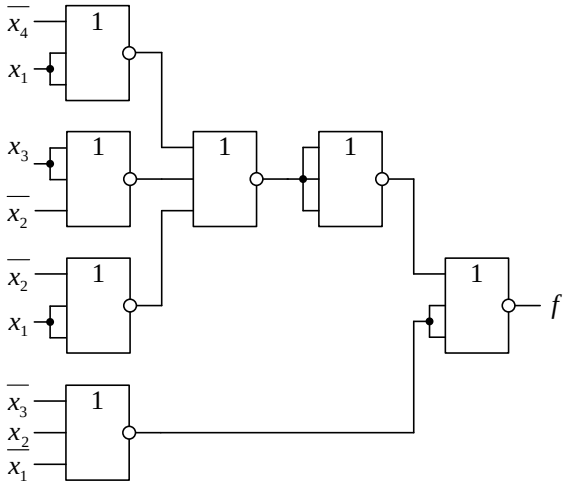


Рис. А2-2.3. Комбінаційна схема

Приклад 2-2.4

Завдання. Знайти МДНФ і МКНФ перемикальної функції за допомогою діаграм Вейча. Побудувати комбінаційну схему в елементному базисі ЗАБО-НЕ. Перемикальна функція задана таблицею істинності (табл. А2-2.7).

Таблиця А2-2.7

ТАБЛИЦЯ ІСТИННОСТІ

x_4	x_3	x_2	x_1	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Виконання завдання

Заповнюємо діаграми Вейча (рис. А2-2.4, *а*, *б*). Значення функції записуємо в клітинки, що відповідають номерам наборів.

а *б*

Рис. А2-2.4. Діаграми Вейча:
а — мінімізація за одиницями; *б* — мінімізація за нулями

На підставі діаграми Вейча (рис. А2-2.4, *а*) визначаємо МДНФ перемикальної функції:

$$f_{\text{МДНФ}} = (\overline{x_3}\,\overline{x_2}x_1) \vee (x_4x_3\overline{x_1}) \vee (x_4x_2x_1) \vee (x_3x_2x_1).$$

Виходячи з рис. А2-2.4, *б* отримуємо:

$$f_{\text{МКНФ}} = (\overline{x_3} \vee x_2 \vee \overline{x_1}) \cdot (x_4 \vee x_3 \vee \overline{x_2}) \cdot (x_3 \vee x_1) \cdot (x_4 \vee x_1).$$

Для побудови комбінаційної схеми із використанням логічних елементів 3АБО-НЕ отримуємо нормальну форму АБО-НЕ / АБО-НЕ на підставі МКНФ заданої функції:

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

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

$$= \overline{\overline{\overline{x_3} \vee x_2 \vee \overline{x_1}} \vee \overline{x_4 \vee x_3 \vee \overline{x_2}} \vee \overline{\overline{x_3 \vee x_1}} \vee \overline{\overline{x_4 \vee x_1}}}. \text{ (АБО-НЕ / АБО-НЕ).}$$

Подамо функцію f в операторній формі і побудуємо комбінаційну схему (рис. А2-2.5):

$$f = \overline{\overline{(\overline{x_3} \vee x_2 \vee \overline{x_1})} \vee \overline{(x_4 \vee x_3 \vee \overline{x_2})} \vee \overline{\overline{(x_3 \vee x_1)}} \vee \overline{\overline{(x_4 \vee x_1)}}}.$$

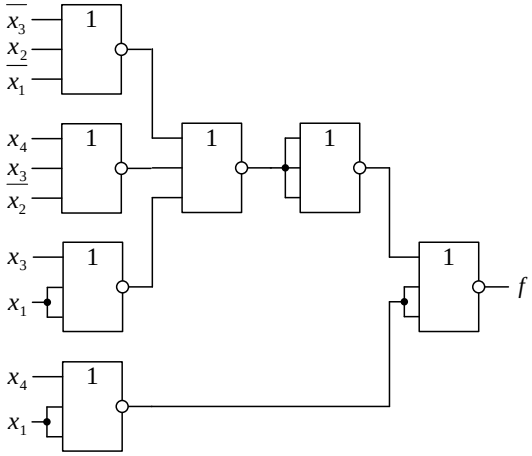


Рис. А2-2.5. Комбінаційна схема в елементному базисі ЗАБО-НЕ

Приклад 2-2.5

Завдання. Виконати мінімізацію системи частково визначених функцій методом Квайна. Зробити перевірку виконаної мінімізації методом Вейча. Система перемикальних функції задана таблицею істинності (табл. А2-2.8). Побудувати комбінаційну схему системи перемикальних функцій в елементному базисі І-НЕ, де кількість входів логічних елементів не перебільшує чотирьох

Таблиця А2-2.8

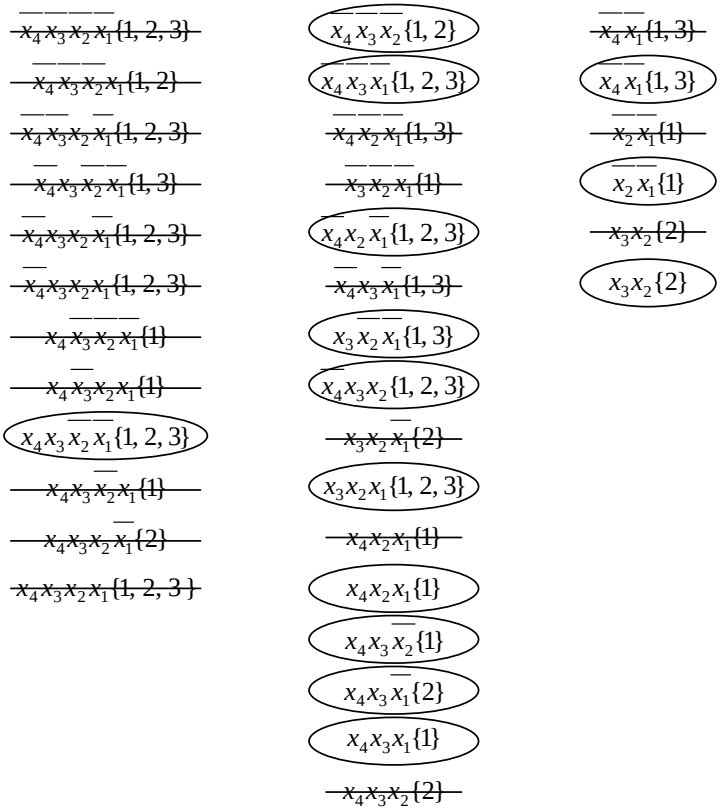
ТАБЛИЦЯ ІСТИННОСТІ

x_4	x_3	x_2	x_1	f_1	f_2	f_3
0	0	0	0	1	1	1
0	0	0	1	1	1	0
0	0	1	0	1	1	1
0	0	1	1	0	0	0
0	1	0	0	–	0	1
0	1	0	1	0	0	0
0	1	1	0	1	–	–
0	1	1	1	–	–	1
1	0	0	0	1	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	0	1	–	1
1	1	0	1	1	0	0
1	1	1	0	0	1	0
1	1	1	1	1	1	1

Виконання завдання

Виконуємо сумісну мінімізацію функцій методом Квайна. У перший стовпець виписуємо конституенти, що складають ДДНФ заданої системи перемикальних функцій. Кожній конституенті присвоюється множина міток, що визначають належність конституенти до відповідної функції.

Далі виконуємо всі попарні склеювання. Склеювання здійснюються тільки для тих термів, що мають спільні мітки. Отримані терми другого рангу розміщуємо у другому стовпці, терми третього рангу — у третьому стовпці. Здійснюємо поглинання термів, при цьому множини міток двох термів, один з яких поглинає другий, повинні повністю збігатися.



Складаємо таблицю покриття (табл. А2-2.9), на підставі якої знаходимо остаточні форми подання функцій, що забезпечують їх спільну реалізацію з мінімальними апаратурними витратами.

Таблиця А2-2.9

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	f_1									f_2					f_3					
	$\overline{x_4}\,\overline{x_3}\,\overline{x_2}\,x_1$	$\overline{x_4}\,\overline{x_3}\,x_2\,x_1$	$\overline{x_4}\,x_3\,\overline{x_2}\,x_1$	$\overline{x_4}\,x_3\,x_2\,x_1$	$x_4\,\overline{x_3}\,\overline{x_2}\,x_1$	$x_4\,\overline{x_3}\,x_2\,x_1$	$x_4\,x_3\,\overline{x_2}\,x_1$	$x_4\,x_3\,x_2\,\overline{x_1}$	$x_4\,x_3\,x_2\,x_1$	$\overline{x_4}\,\overline{x_3}\,\overline{x_2}\,x_1$	$\overline{x_4}\,\overline{x_3}\,x_2\,x_1$	$\overline{x_4}\,x_3\,\overline{x_2}\,x_1$	$x_4\,\overline{x_3}\,\overline{x_2}\,x_1$	$x_4\,x_3\,x_2\,x_1$	$\overline{x_4}\,\overline{x_3}\,\overline{x_2}\,x_1$	$\overline{x_4}\,\overline{x_3}\,x_2\,x_1$	$\overline{x_4}\,x_3\,\overline{x_2}\,x_1$	$x_4\,\overline{x_3}\,x_2\,x_1$	$\overline{x_4}\,x_3\,x_2\,x_1$	$x_4\,x_3\,x_2\,x_1$
$x_4\,x_3\,\overline{x_2}\,\overline{x_1}\{1, 2, 3\}$							∨						∨						∨	
$\overline{x_4}\,\overline{x_3}\,\overline{x_2}\{1, 2\}$	Ⓥ	Ⓥ								Ⓥ	Ⓥ									
$\overline{x_4}\,\overline{x_3}\,x_1\{1, 2, 3\}$	∨		∨							∨		∨			∨	∨				
$\overline{x_4}\,x_2\,\overline{x_1}\{1, 2, 3\}$			∨	∨								Ⓥ				∨				
$x_3\,\overline{x_2}\,\overline{x_1}\{1, 3\}$							∨										Ⓥ		Ⓥ	
$\overline{x_4}\,x_3\,x_2\{1, 2, 3\}$				∨														∨		
$x_3\,x_2\,x_1\{1, 2, 3\}$									∨					∨				Ⓥ		Ⓥ
$x_4\,x_2\,x_1\{1\}$						Ⓥ			Ⓥ											
$x_4\,x_3\,\overline{x_2}\{1\}$							∨	∨												
$x_4\,x_3\,\overline{x_1}\{2\}$													∨							
$x_4\,x_3\,x_1\{1\}$								Ⓥ	Ⓥ											
$\overline{x_4}\,\overline{x_1}\{1, 3\}$	Ⓥ		Ⓥ	Ⓥ											Ⓥ	Ⓥ	Ⓥ			
$\overline{x_2}\,\overline{x_1}\{1\}$	Ⓥ				Ⓥ		Ⓥ													
$x_3\,x_2\{2\}$													Ⓥ	Ⓥ						

$$f_{1\,\text{МДНФ}} = x_4x_2x_1 \vee x_4x_3x_1 \vee \overline{x_4}x_3x_2 \vee \overline{x_2}x_1 \vee \overline{x_4}x_1;$$

$$f_{2\,\text{МДНФ}} = \overline{x_4}x_2\overline{x_1} \vee \overline{x_4}x_3x_2 \vee x_3x_2; \quad f_{3\,\text{МДНФ}} = x_3\overline{x_2}x_1 \vee x_3x_2x_1 \vee \overline{x_4}x_1.$$

Для перевірки виконуємо окремо мінімізацію кожної функції за допомогою діаграм Вейча (рис. А2-2.6).

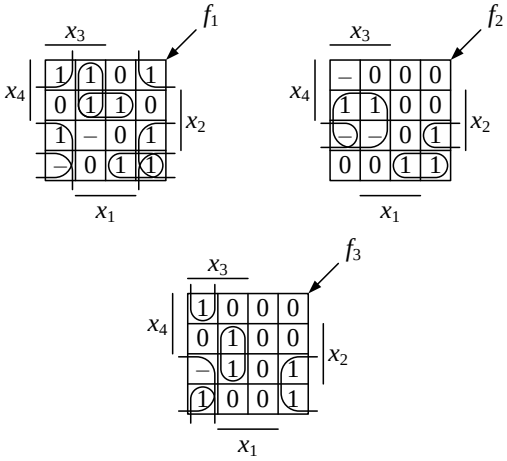


Рис. А2-2.6. Діаграми Вейча для функцій f_1, f_2, f_3

МДНФ, отримані під час мінімізації методом Квайна та методом Дейча, ідентичні.

Мінімальна ДНФ відповідає нормальній формі І/АБО, за допомогою теорем де Моргана запишемо нормальні форми І-НЕ / І-НЕ для функцій f_1, f_2, f_3:

$$f_1 = x_4x_2x_1 \vee x_4x_3x_1 \vee \overline{x_4}x_3x_2 \vee \overline{x_2}x_1 \vee \overline{x_4}x_1 =$$

$$= \overline{\overline{x_4x_2x_1 \vee x_4x_3x_1 \vee \overline{x_4}x_3x_2 \vee \overline{x_2}x_1 \vee \overline{x_4}x_1}} =$$

$$= \overline{x_4x_2x_1} \cdot \overline{x_4x_3x_1} \cdot \overline{\overline{x_4}x_3x_2} \cdot \overline{\overline{x_2}x_1} \cdot \overline{\overline{x_4}x_1}; \qquad \text{І-НЕ/І-НЕ}$$

$$f_2 = \overline{x_4}x_2\overline{x_1} \vee x_4x_3x_2 \vee x_3x_2 = \overline{\overline{x_4}x_2\overline{x_1} \vee \overline{x_4x_3x_2} \vee x_3x_2} =$$

$$= \overline{\overline{x_4}x_2\overline{x_1}} \cdot \overline{\overline{x_4x_3x_2}} \cdot \overline{x_3x_2}; \qquad \text{І-НЕ/І-НЕ}$$

$$f_3 = x_3\overline{x_2}x_1 \vee x_3x_2x_1 \vee \overline{x_4}x_1 = \overline{x_3\overline{x_2}x_1 \vee x_3x_2x_1 \vee \overline{x_4}x_1} =$$

$$= \overline{x_3\overline{x_2}x_1} \cdot \overline{x_3x_2x_1} \cdot \overline{\overline{x_4}x_1}. \qquad \text{І-НЕ/І-НЕ}$$

Нормальні форми функцій f_2, f_3 дозволяють побудувати комбінаційні схеми на логічних елементах І-НЕ з кількістю входів не більш ніж чотири, тому збігаються з операторними формами. Запишемо операторну форму для функції f_1:

$$f_1 = \overline{(\overline{x_4 x_2 x_1}) \cdot (\overline{x_4 x_3 x_1}) \cdot (\overline{\overline{x_4 x_3 x_2}}) \cdot (\overline{\overline{x_2 x_1}})} \cdot \overline{(\overline{x_4 x_1})};$$

За отриманими формами будуємо комбінаційну схему (рис. А2-2.7).

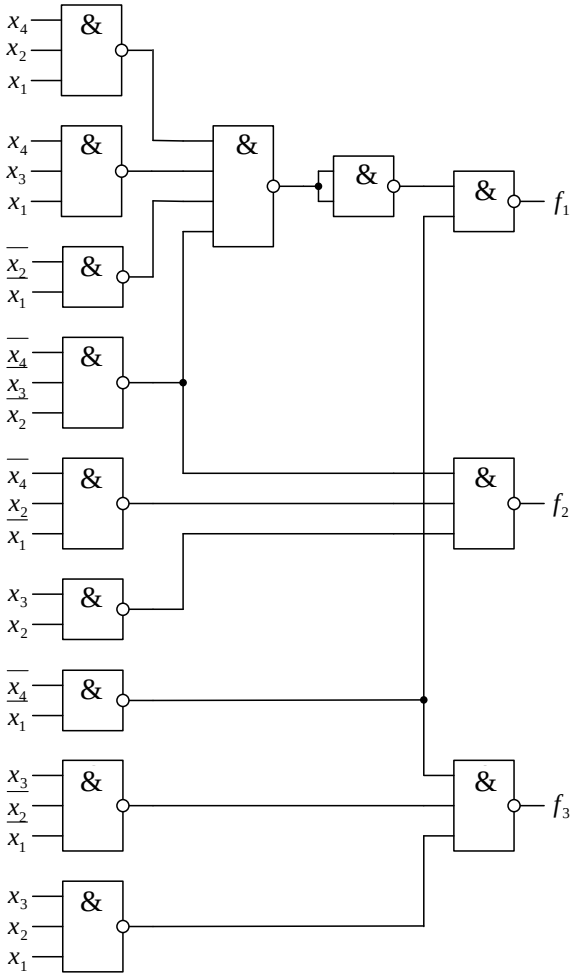


Рис. А2-2.7. Схема системи перемикальних функцій f_1 f_2 f_3

Приклад 2-2.6

Завдання. Виконати сумісну мінімізацію системи функцій та їх заперечень методом Квайна—Мак-Класкі (табл. А2-2.10). Побудувати комбінаційні схеми, використовуючи логічні елементи з кількістю входів не більш ніж чотири.

Таблиця А2-2.10

ТАБЛИЦЯ ІСТИННОСТІ

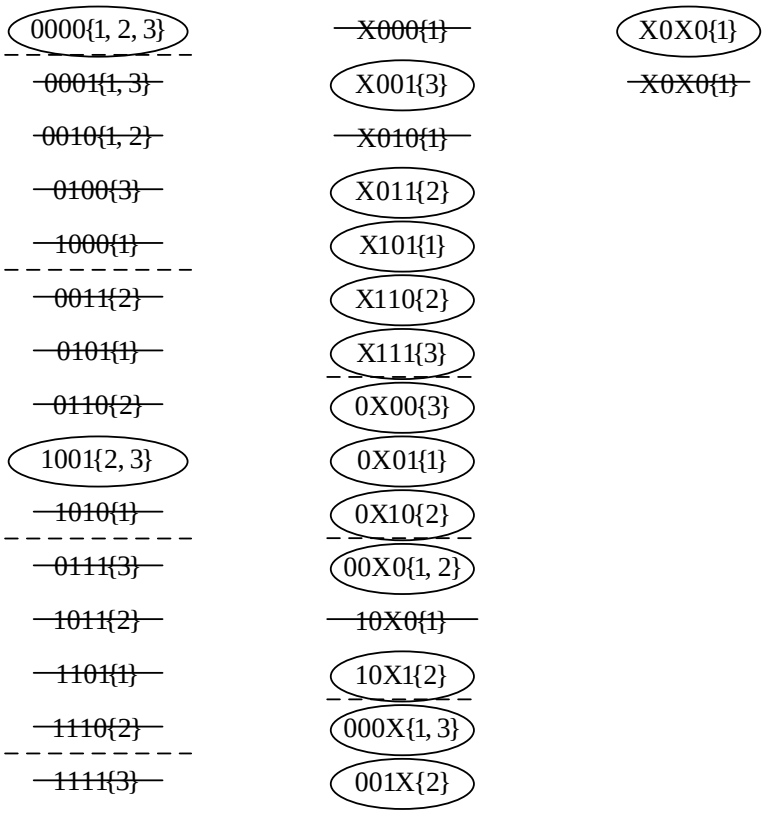
x_4	x_3	x_2	x_1	f_1	f_2	f_3
0	0	0	0	1	1	1
0	0	0	1	1	0	1
0	0	1	0	1	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	0	1	0
1	1	1	1	0	0	1

Виконання завдання

Для отримання МДНФ системи перемикальних функцій виконаємо мінімізацію прямих значень функцій.

Виходячи з таблиці істинності системи перемикальних функції записуємо комплекс кубів K^0. Для цього виписуємо набори, на яких хоча б одна із функцій системи приймає значення одиниці. Кожній конституенті ставиться у відповідність множина міток, що вказують на належність конституенти до певної функції системи. Виписані терми поєднуємо у групи за однаковою кількістю одиниць.

Виконуємо всі можливі попарні склеювання та отримуємо комплекси кубів K^1 і K^2. Шляхом поглинання термів формуємо Z-покриття, що відповідає СДНФ системи перемикальних функцій. Терми, що складають Z-покриття, позначені овалами.



Для видалення надлишкових імплікант будуємо таблицю покриття (табл. А2-2.11).

На підставі таблиці покриття одержуємо МДНФ перемикальних функцій:

$$f_1 = x_3\overline{x_2}x_1 \vee \overline{x_4x_3x_2} \vee \overline{x_3}\,\overline{x_1};$$

$$f_2 = x_4\overline{x_3}\,\overline{x_2}x_1 \vee \overline{x_3}x_2x_1 \vee x_3x_2\overline{x_1} \vee \overline{x_4x_3x_1};$$

$$f_3 = x_4\overline{x_3}\,\overline{x_2}x_1 \vee x_3x_2x_1 \vee \overline{x_4x_2x_1} \vee \overline{x_4x_3x_2}.$$

Отримані форми відповідають нормальній формі І / АБО, тому для побудови комбінаційної схеми застосуємо логічні елементи І, АБО.

Отримані нормальні форми збігаються з операторними, оскільки дозволяють реалізувати систему перемикальних функцій на логічних елементах з кількістю входів не більш ніж чотири.

Таблиця А2-2.11

ТАБЛИЦЯ ПОКРИТТЯ СИСТЕМИ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Імпліканти	f_1							f_2							f_3					
	0000	0001	0010	0101	1000	1010	1101	0000	0010	0011	0110	1001	1011	1110	0000	0001	0100	0111	1001	1111
0000{1, 2, 3}	∨							∨							∨					
1001{2, 3}												Ⓥ							Ⓥ	
X001{3}																∨			∨	
X011{2}										Ⓥ			Ⓥ							
X101{1}				Ⓥ			Ⓥ													
X110{2}											Ⓥ			Ⓥ						
X111{3}																		Ⓥ		Ⓥ
0X00{3}															Ⓥ		Ⓥ			
0X01{1}		∨		∨																
0X10{2}									∨		∨									
00X0{1, 2}	∨		∨					Ⓥ	Ⓥ											
10X1{2}												∨	∨							
000X{1, 3}	Ⓥ	Ⓥ													∨	Ⓥ				
001X{2}									∨	∨										
X0X0{1}	Ⓥ		Ⓥ		Ⓥ	Ⓥ														

За отриманими формами будуємо комбінаційну схему, зображену на рис. А2-2.8.

Виконаємо мінімізацію заперечень функцій.

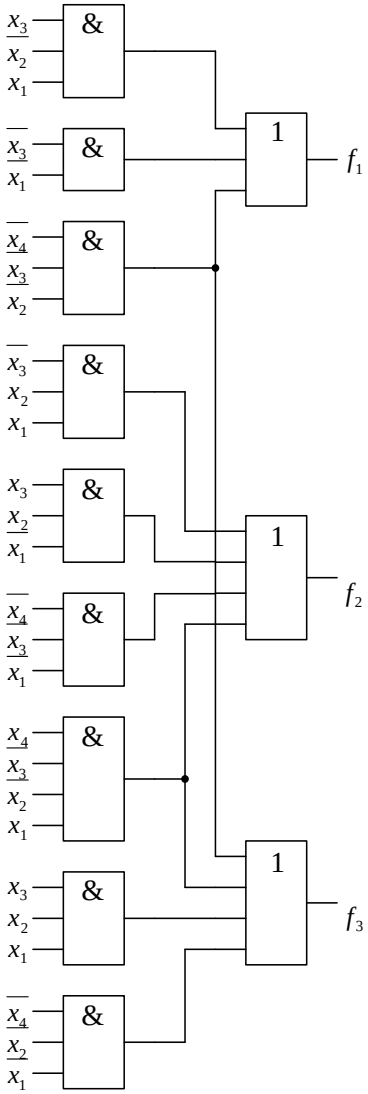
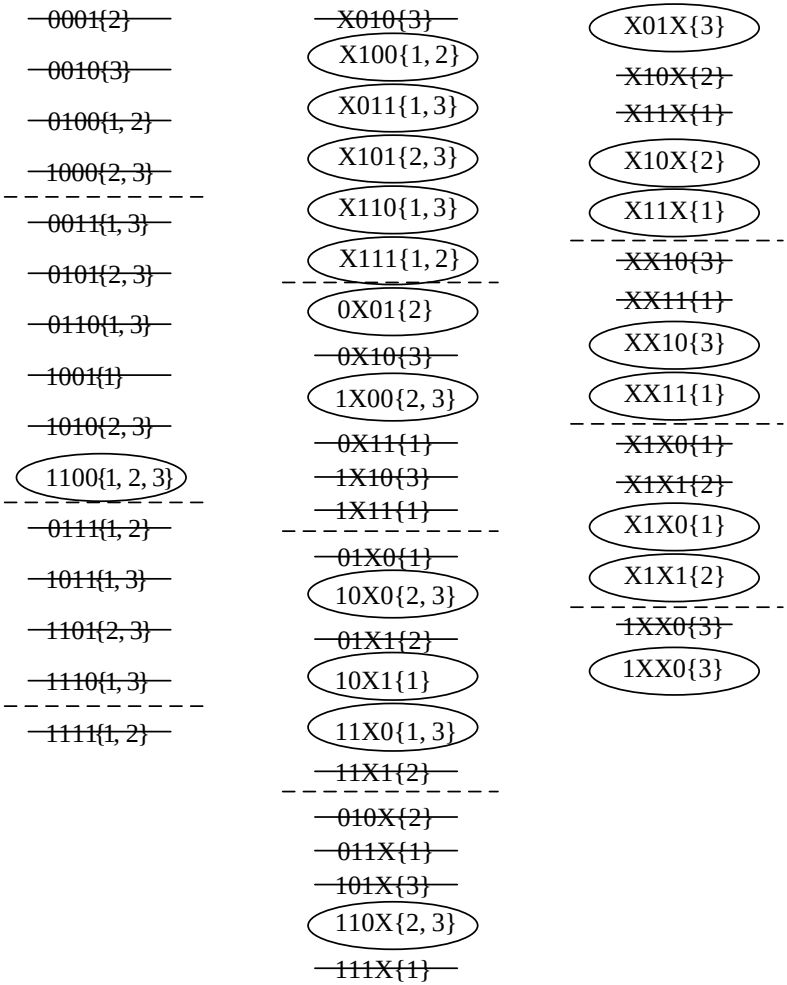


Рис. А2-2.8. Комбінаційна схема, що реалізує систему перемикальних функцій у елементному базисі І, АБО

Виходячи з таблиці істинності системи перемикальних функцій записуємо комплекс кубів K^0 для заперечень функцій. Для цього виписуємо набори, на яких хоча б одна із функцій системи приймає значення нуля. Кожному набору ставиться у відповідність множина міток.

У результаті склеювання і поглинання отримуємо *Z*-покриття, що відповідає СДНФ заперечення системи перемикальних функцій.



Будуємо таблицю покриття (табл. А2-2.12).

Таблиця А2-2.12

ТАБЛИЦЯ ПОКРИТТЯ СИСТЕМИ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Імпліканти	f_1^*									f_2^*									f_3^*									
	0011	0100	0110	0111	1001	1011	1100	1110	1111	0001	0100	0101	0111	1000	1010	1100	1101	1111	0010	0011	0101	0110	1000	1010	1011	1100	1101	1110
1100{1, 2, 3}							∨									∨										∨		
X100{1, 2}		∨					∨				∨					∨												
X011{1, 3}	∨					∨														∨					∨			
X101{2, 3}												∨					∨				Ⓥ						Ⓥ	
X110{1, 3}			∨					∨														∨						∨
X111{1, 2}				∨					∨				∨					∨										
0X01{2}										Ⓥ		Ⓥ																
1X00{2, 3}														∨		∨							∨			∨		
10X0{2, 3}														Ⓥ	Ⓥ								∨	∨				
10X1{1}					Ⓥ	Ⓥ																						
11X0{1, 3}							∨	∨																		∨		∨
110X{2, 3}																∨	∨									∨	∨	
X01X{3}																			Ⓥ	Ⓥ				Ⓥ	Ⓥ			
XX10{3}																			∨			∨		∨				Ⓥ
XX11{1}	Ⓥ			Ⓥ		Ⓥ			Ⓥ																			
X1X0{1}		Ⓥ	Ⓥ				Ⓥ	Ⓥ																				
1XX0{3}																							Ⓥ	Ⓥ		Ⓥ		Ⓥ
X1X1{2}												Ⓥ	Ⓥ				Ⓥ	Ⓥ										
X10X{2}											Ⓥ	Ⓥ				Ⓥ	Ⓥ											
X11X{1}			∨	∨				∨	∨																			

Виходячи з таблиці покриття одержуємо систему перемикальних функцій:

$$f_1^* = \overline{x_4\overline{x_3}x_1 \vee x_2x_1 \vee x_3\overline{x_1}};$$

$$f_2^* = \overline{\overline{x_4}\,\overline{x_2}x_1 \vee x_4\overline{x_3}\,\overline{x_1} \vee x_3x_1 \vee x_3\overline{x_2}};$$

$$f_3^* = \overline{\overline{x_3}\,\overline{x_2}x_1 \vee \overline{x_3}x_2 \vee x_2\overline{x_1} \vee x_4\overline{x_1}}.$$

Отримані форми відповідають нормальній формі І / АБО-НЕ, тому для побудови комбінаційної схеми використаємо елементний базис І, АБО-НЕ.

За отриманими формами будуємо комбінаційну схему, зображену на рис. А2-2.9.

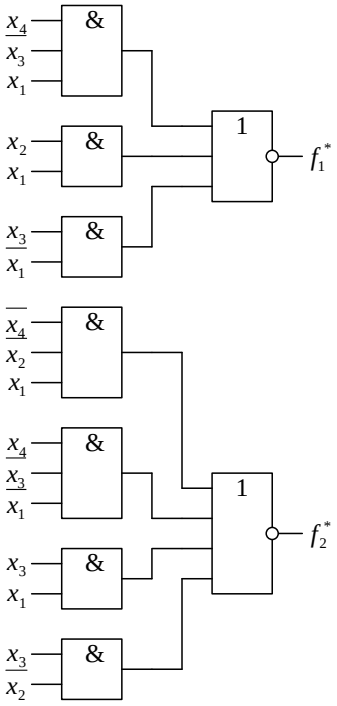
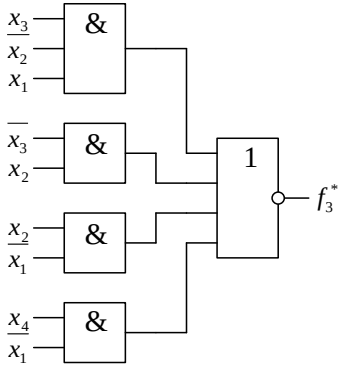


Рис. А2-2.9. Комбінаційна схема,
що реалізує систему перемикальних функцій
у елементному базисі І, АБО-НЕ



Закінчення рис. А2-2.9

Приклад 2-2.7

Завдання. За заданою таблицею покриття перемикальної функції f (табл. А2-2.13) знайти ТДНФ і МДНФ функції методом Петрика.

Таблиця А2-2.13

ТАБЛИЦЯ ПОКРИТТЯ

Імпліканти	Конституенти				
	$x_3x_2x_1$	$x_3\overline{x_2}x_1$	$x_3\overline{x_2}\,\overline{x_1}$	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}$	$\overline{x_3}x_2x_1$
x_3x_1	$\vee$	$\vee$			
x_2x_1	$\vee$				$\vee$
$x_3\overline{x_2}$		$\vee$	$\vee$		
$\overline{x_2}\,\overline{x_1}$			$\vee$	$\vee$	

Виконання завдання

Позначимо наявні у таблиці покриття прості імпліканти таким чином:

$$x_3x_1 = A;\ \ x_2x_1 = B;\ \ x_3\overline{x_2} = C;\ \ \overline{x_2}\,\overline{x_1} = D.$$

Сформуємо кон'юнктивне подання φ таблиці покриття:

$$\varphi = (A \vee B)\cdot(A \vee C)\cdot(C \vee D)\cdot D\cdot B.$$

Спростимо отриманий вираз:

$$\varphi = ABD \vee BCD.$$

У результаті спрощення отримана диз'юнкція двох кон'юнкцій, що відповідають двом ТДНФ:

$$f_{\text{ТДНФ 1}} = x_3x_1 \vee x_2x_1 \vee \overline{x}_2\overline{x}_1;$$

$$f_{\text{ТДНФ 2}} = x_2x_1 \vee x_3\overline{x}_2 \vee \overline{x}_2\overline{x}_1.$$

Усі отримані ТДНФ мають однакову ціну, тому як МДНФ можна обрати будь-яку з отриманих форм, наприклад

$$f_{\text{МДНФ}} = x_3x_1 \vee x_2x_1 \vee \overline{x}_2\overline{x}_1.$$

А2-3. Цифрові автомати з пам'яттю

Приклад 2-3.1

Завдання. Виконати синтез автомата Мура, структурну ГСА функціонування якого зображено на рис. А2-3.1. Побудувати функціональну схему автомата в елементному базисі І-НЕ, АБО-НЕ, де кількість входів логічних елементів не більш ніж чотири. Для організації пам'яті обрати тригери типу *T*.

Виконання завдання

В операторних вершинах ГСА записані вихідні структурні сигнали y_s, а в логічних вершинах — вхідні структурні сигнали x_k.

Виконаємо розмітку станів автомата Мура, при цьому кожну операторну вершину (рис. А2-3.1) позначимо символами z_i.

На підставі розміченої ГСА будуємо граф автомата, зображений на рис. А2-3.2.

Кількість вершин графа дорівнює числу станів автомата, де кожному стану відповідає набір управляючих сигналів. Кожному переходові автомата з одного стану в інший відповідає дуга графа, котрій приписується набір логічних умов, за яких здійснюються переходи автомата з одного стану в інший.

Виконаємо кодування станів автомату. На підставі того, що число станів автомата дорівнює восьми, знайдемо кількість тригерів необхідних для організації пам'яті (див. розділ А1-3.3):

$$n=3 >] \log_2 8 [.$$

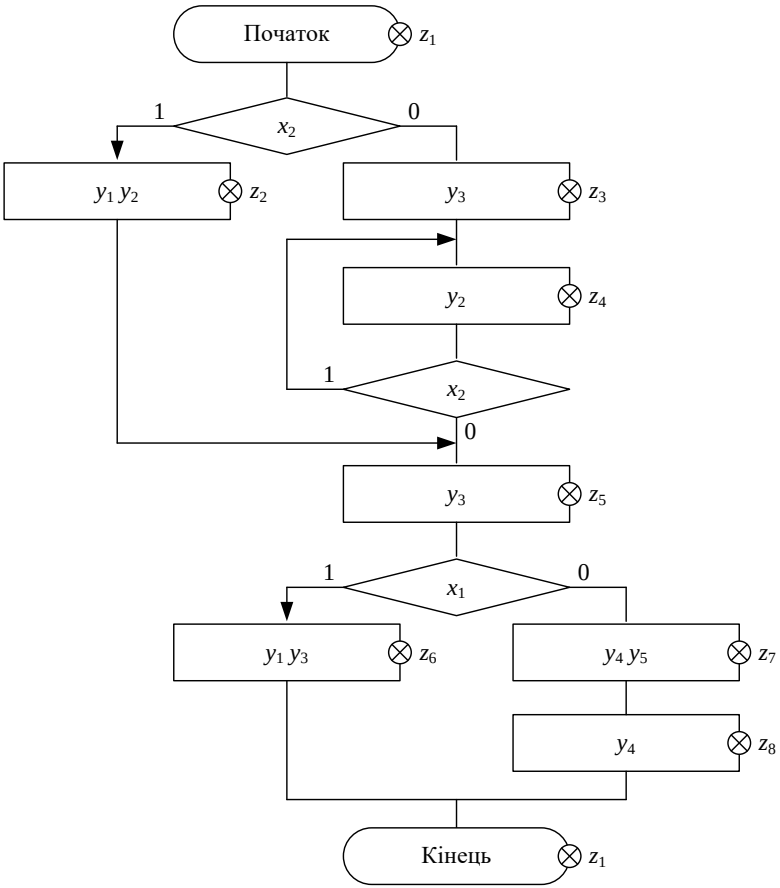


Рис. А2-3.1. Граф-схема автомата

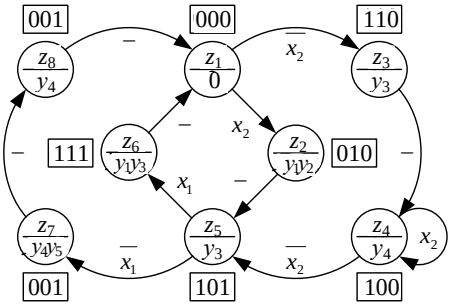


Рис. А2-3.2. Граф керуючого автомата Мура

Складаємо таблицю кодування станів (табл. А2-3.1).

Таблиця А2-3.1

ТАБЛИЦЯ КОДУВАННЯ СТАНІВ

Стан	Код стану		
	Q_3	Q_2	Q_1
z_1	0	0	0
z_2	0	1	0
z_3	1	1	0
z_4	1	0	0
z_5	1	0	1
z_6	1	1	1
z_7	0	1	1
z_8	0	0	1

За графом автомату складаємо відмічену таблицю переходів автомата Мура (табл. А2-3.2)

Таблиця А2-3.2

ТАБЛИЦЯ ПЕРЕХОДІВ АВТОМАТА МУРА

x_1	x_2	$\frac{z_1}{0}$	$\frac{z_2}{y_1\, y_2}$	$\frac{z_3}{y_3}$	$\frac{z_4}{y_2}$	$\frac{z_5}{y_3}$	$\frac{z_6}{y_1\, y_3}$	$\frac{z_7}{y_4\, y_5}$	$\frac{z_8}{y_4}$
0	0	z_3	z_5	z_4	z_5	z_7	z_1	z_8	z_1
0	1	z_2	z_5	z_4	z_4	z_7	z_1	z_8	z_1
1	0	z_3	z_5	z_4	z_5	z_6	z_1	z_8	z_1
1	1	z_2	z_5	z_4	z_4	z_6	z_1	z_8	z_1

За графом автомату складаємо структурну таблицю автомата (табл. А2-3.3). Кожен рядок таблиці відповідає визначеному переходові автомата з одного стану в наступний.

Для організації пам'яті за умовою завдання використовуємо *Т*-тригери. Значення функцій збудження тригерів визначаємо відповідно до системи переходів *Т*-тригера (рис. А1-3.4, *б*).

Таблиця А2-3.3

СТРУКТУРНА ТАБЛИЦЯ АВТОМАТА МУРА

Перехід ПС→СП	Код ПС			Код СП			Логічні умови		Керуючі сигнали					Функції збудження тригерів		
	Q_3^t	Q_2^t	Q_1^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	x_1^t	x_2^t	y_1	y_2	y_3	y_4	y_5	T_3	T_2	T_1
$z_1 \to z_2$	0	0	0	0	1	0	–	1	0	0	0	0	0	0	1	0
$z_2 \to z_5$	0	1	0	1	0	1	–	–	1	1	0	0	0	1	1	1
$z_1 \to z_3$	0	0	0	1	1	0	–	0	0	0	0	0	0	1	1	0
$z_3 \to z_4$	1	1	0	1	0	0	–	–	0	0	1	0	0	0	1	0
$z_4 \to z_4$	1	0	0	1	0	0	–	1	0	1	0	0	0	0	0	0
$z_4 \to z_5$	1	0	0	1	0	1	–	0	0	1	0	0	0	0	0	1
$z_5 \to z_6$	1	0	1	1	1	1	1	–	0	0	1	0	0	0	1	0
$z_5 \to z_7$	1	0	1	0	1	1	0	–	0	0	1	0	0	1	1	0
$z_6 \to z_1$	1	1	1	0	0	0	–	–	1	0	1	0	0	1	1	1
$z_7 \to z_8$	0	1	1	0	0	1	–	–	0	0	0	1	1	0	1	0
$z_8 \to z_1$	0	0	1	0	0	0	–	–	0	0	0	1	0	0	0	1

На основі структурної таблиці автомату визначаємо МДНФ функцій управляючих сигналів та функцій збудження тригерів. Мінімізацію виконуємо за допомогою діаграм Вейча, зображених на рис. А2-3.3.

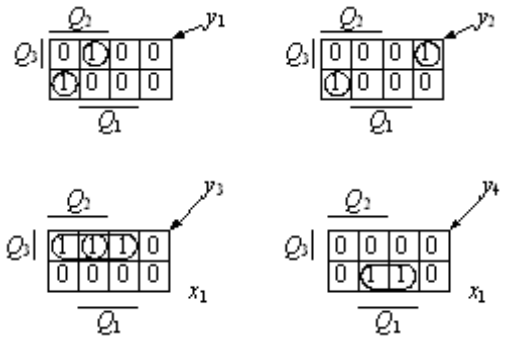
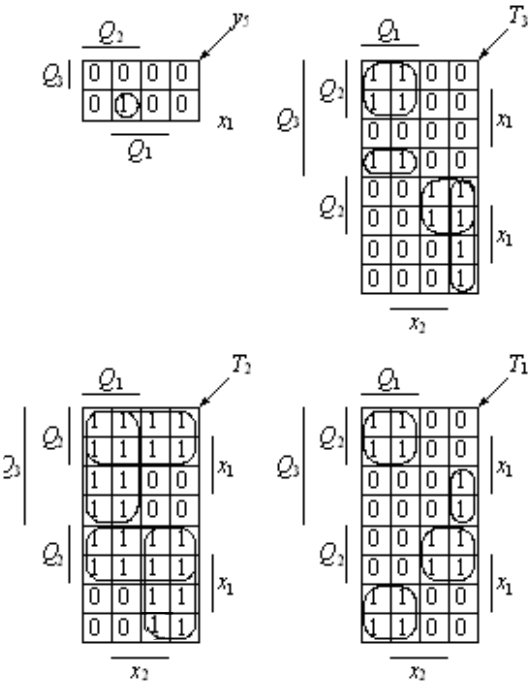


Рис. А2-3.3. Діаграми Вейча для функцій управляючих сигналів та функцій збудження тригерів



Закінчення рис. А2-3.3

Одержуємо МДНФ функцій управляючих сигналів та функцій збудження тригерів, які за допомогою правил де Моргана запишемо в нормальних формах І-НЕ/І-НЕ та АБО-НЕ/АБО-НЕ:

$$y_1 = Q_3Q_2Q_1 \vee \overline{Q_3}Q_2Q_1 = \overline{\overline{Q_3Q_2Q_1} \cdot \overline{\overline{Q_3}Q_2Q_1}};$$

$$y_2 = \overline{Q_3}Q_2\overline{Q_1} \vee Q_3\overline{Q_2}\overline{Q_1} = \overline{\overline{\overline{Q_3}Q_2\overline{Q_1}} \cdot \overline{Q_3\overline{Q_2}\overline{Q_1}}};$$

$$y_3 = Q_3Q_1 \vee Q_3Q_2 = \overline{\overline{Q_3Q_1} \cdot \overline{Q_3Q_2}};$$

$$y_4 = \overline{Q_3}Q_1 = \overline{Q_3 \vee \overline{Q_1}};$$

$$y_5 = \overline{Q_3}Q_2Q_1 = \overline{Q_3 \vee \overline{Q_2} \vee \overline{Q_1}}.$$

$$T_3 = \overline{Q_3}\overline{Q_1}\overline{x_2} \vee Q_3\overline{Q_2}Q_1\overline{x_1} \vee \overline{Q_3}Q_2\overline{Q_1} \vee Q_3Q_2Q_1 =$$

$$= \overline{\overline{Q_3}\overline{Q_1}\overline{x_2}} \cdot \overline{Q_3\overline{Q_2}Q_1\overline{x_1}} \cdot \overline{\overline{Q_3}Q_2\overline{Q_1}} \cdot \overline{Q_3Q_2Q_1};$$

$$T_2 = Q_3Q_1 \vee \overline{Q_3}\,\overline{Q_1} \vee Q_2 = \overline{\overline{Q_3Q_1} \cdot \overline{\overline{Q_3}\,\overline{Q_1}} \cdot \overline{Q_2}};$$

$$T_1 = Q_3Q_2Q_1 \vee \overline{Q_3}Q_2\overline{Q_1} \vee \overline{Q_3}\,\overline{Q_2}Q_1 \vee Q_3\overline{Q_2}\,\overline{Q_1}\,\overline{x_2} =$$

$$= \overline{\overline{Q_3Q_2Q_1} \cdot \overline{\overline{Q_3}Q_2\overline{Q_1}} \cdot \overline{\overline{Q_3}\,\overline{Q_2}Q_1} \cdot \overline{Q_3\overline{Q_2}\,\overline{Q_1}\,\overline{x_2}}};$$

Отримані нормальні форми дозволяють реалізувати функції на логічних елементах з кількістю входів не більш ніж чотири, тобто збігаються із операторними формами.

Застосовуючи отримані форми, будуємо функціональну схему автомата Мура, зображену на рис. А2-3.4.

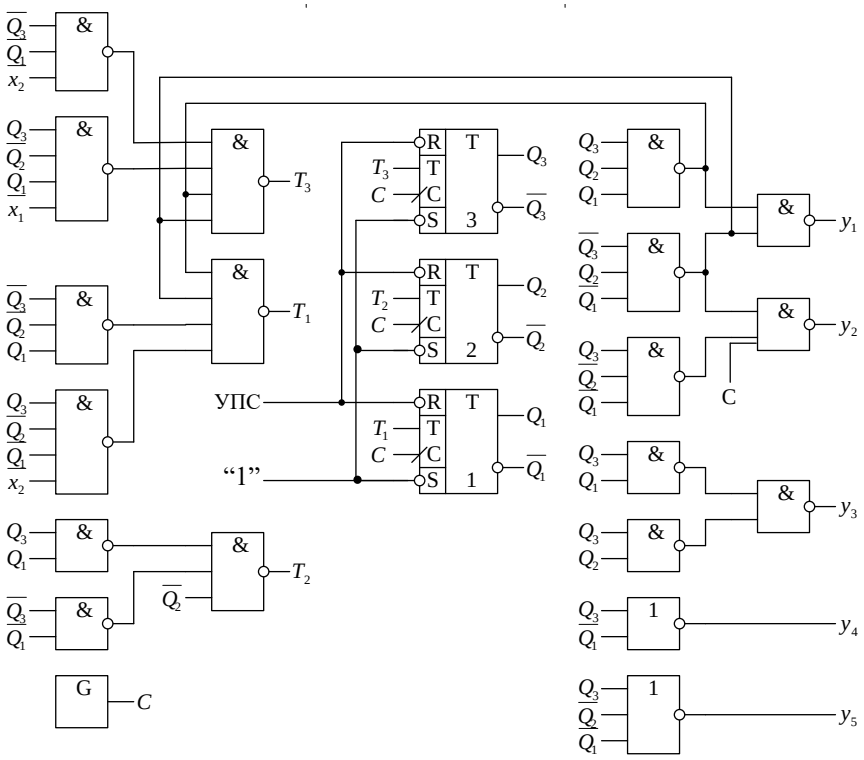


Рис. А2-3.4. Функціональна схема автомата Мура

Приклад 2-3.2

Завдання. Виконати синтез автомата Мілі методом композиції тригерів, структурна ГСА якого зображена на рис. А2-3.5. Побудувати функціональну схему управляючого автомата на базі *JK*-тригерів.

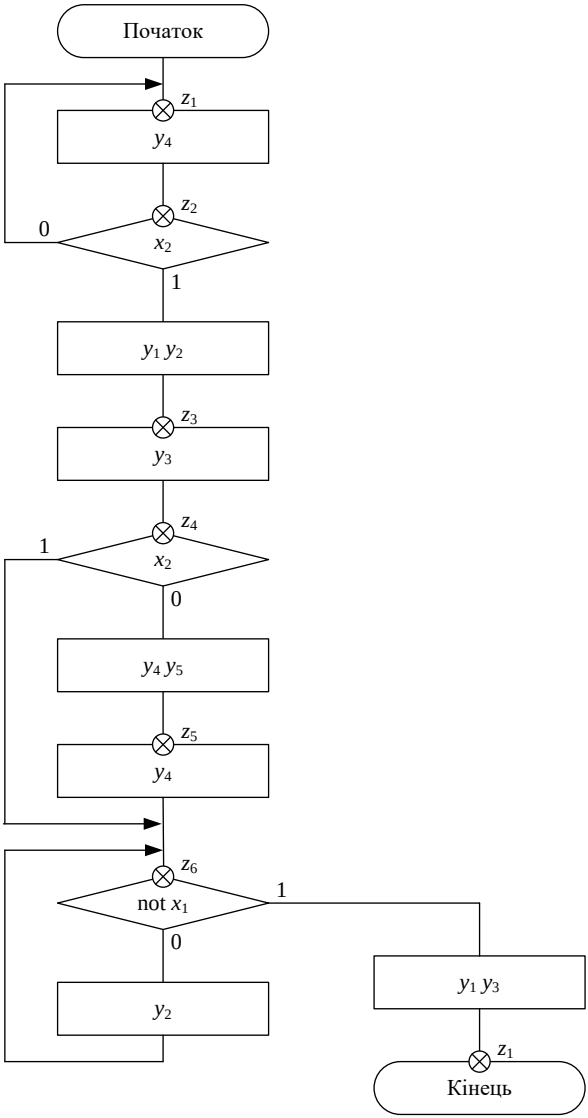


Рис. А2-3.5. ГСА автомата

Виконаємо розмітку станів автомата Мілі. При цьому входи всіх вершин заданого мікроалгоритму (рис. А2-3.5), наступних після операторних, позначимо символами z_i. Вхід вершини, наступної за початковою, та вхід кінцевої вершини позначимо символом z_1. Таким чином, отримаємо шість різних станів.

На підставі розміченої ГСА будуємо граф автомата Мілі (рис. А2-3.6).

Кожній вершині графу ставимо у відповідність стан z_i автомату. Кожному переходові автомата з попереднього стану в наступний відповідає дуга графа. Кожній дузі приписуємо набір логічних умов, за якого здійснюється перехід автомата з одного стану в інший, а також набір управляючих сигналів, що відповідають даному переходові.

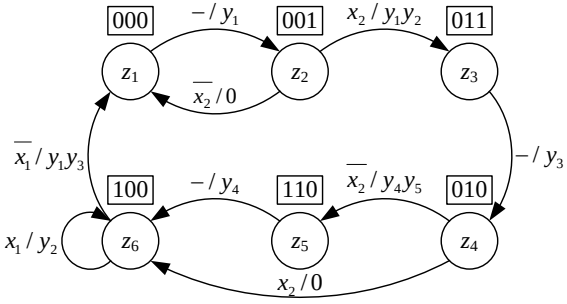


Рис. А2-3.6. Граф автомата Мілі

Виконаємо кодування станів автомату. Виходячи з того, що число станів автомату дорівнює шістьом, визначаємо кількість n тригерів необхідних для організації пам'яті:

$$n = 3 >] \log_2 6 [.$$

Складаємо таблицю кодування станів автомата (табл. А2-3.4).

Таблиця А2-3.4

ТАБЛИЦЯ КОДУВАННЯ СТАНІВ

Стан	Код стану		
	Q_3	Q_2	Q_1
z_1	0	0	0
z_2	0	0	1
z_3	0	1	1
z_4	0	1	0
z_5	1	1	0
z_6	1	0	0

Система підграфів переходів *JK*-тригера зображена на рис. А1-3.4, *б*. За графом автомату (рис. А2-3.6) складаємо структурну таблицю автомата Мілі (табл. А2-3.5).

Таблиця А2-3.5

ТАБЛИЦЯ СТРУКТУРНОГО СИНТЕЗУ АВТОМАТА МІЛІ

Перехід	Код ПС			Код СП			Входи		Виходи					Функції збудження					
	Q_3^t	Q_2^t	Q_1^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	x_1^t	x_2^t	y_1	y_2	y_3	y_4	y_5	J_3	K_3	J_2	K_2	J_1	K_1
$z_1 \to z_2$	0	0	0	0	0	1	–	–	1	0	0	0	0	0	–	0	–	1	–
$z_2 \to z_1$	0	0	1	0	0	0	–	0	0	0	0	0	0	0	–	0	–	–	1
$z_2 \to z_3$	0	0	1	0	1	1	–	1	1	1	0	0	0	0	–	1	–	–	0
$z_3 \to z_4$	0	1	1	0	1	0	–	–	0	0	1	0	0	0	–	–	0	–	1
$z_4 \to z_6$	0	1	0	1	0	0	–	1	0	0	0	0	0	1	–	–	1	0	–
$z_4 \to z_5$	0	1	0	1	1	0	–	0	0	0	0	1	1	1	–	–	0	0	–
$z_5 \to z_6$	1	1	0	1	0	0	–	–	0	0	0	1	0	–	0	–	1	0	–
$z_6 \to z_6$	1	0	0	1	0	0	1	–	0	1	0	0	0	–	0	0	–	0	–
$z_6 \to z_1$	1	0	0	0	0	0	0	–	1	0	1	0	0	–	1	0	–	0	–

Виконуємо мінімізацію функцій управляючих сигналів і функцій збудження тригерів методом діаграм Вейча, зображених на рис. А2-3.7.

Одержимо МДНФ функцій управляючих сигналів та функцій збудження тригерів.

$$y_1 = \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1} \vee Q_3\overline{Q_2}\,\overline{x_1} \vee \overline{Q_2}Q_1x_2;$$

$$y_2 = \overline{Q_2}Q_1x_2 \vee Q_3\overline{Q_2}x_1;$$

$$y_3 = Q_3\overline{Q_2}\,\overline{x_1} \vee Q_2Q_1;$$

$$y_4 = Q_2\overline{Q_1}\,\overline{x_2} \vee Q_3Q_2;$$

$$y_5 = \overline{Q_3}Q_2\,\overline{Q_1}\,\overline{x_2};$$

$$J_3 = Q_2\overline{Q_1}; \qquad K_3 = \overline{Q_2}\,\overline{x_1};$$

$$J_2 = Q_1x_2 \qquad K_2 = \overline{Q_1}x_2 \vee Q_3;$$

$$J_1 = \overline{Q_3}\,\overline{Q_2}; \qquad K_1 = Q_2 \vee \overline{x_2}.$$

Для побудови функціональної схеми управляючого автомату застосуємо елементи І, АБО, НЕ.

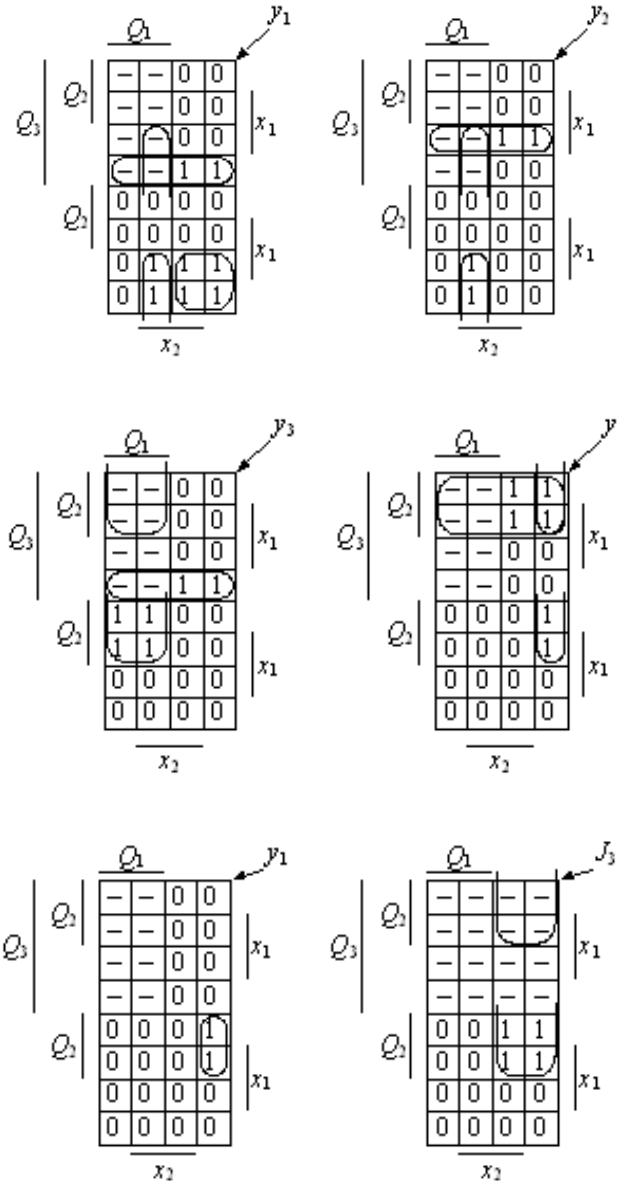
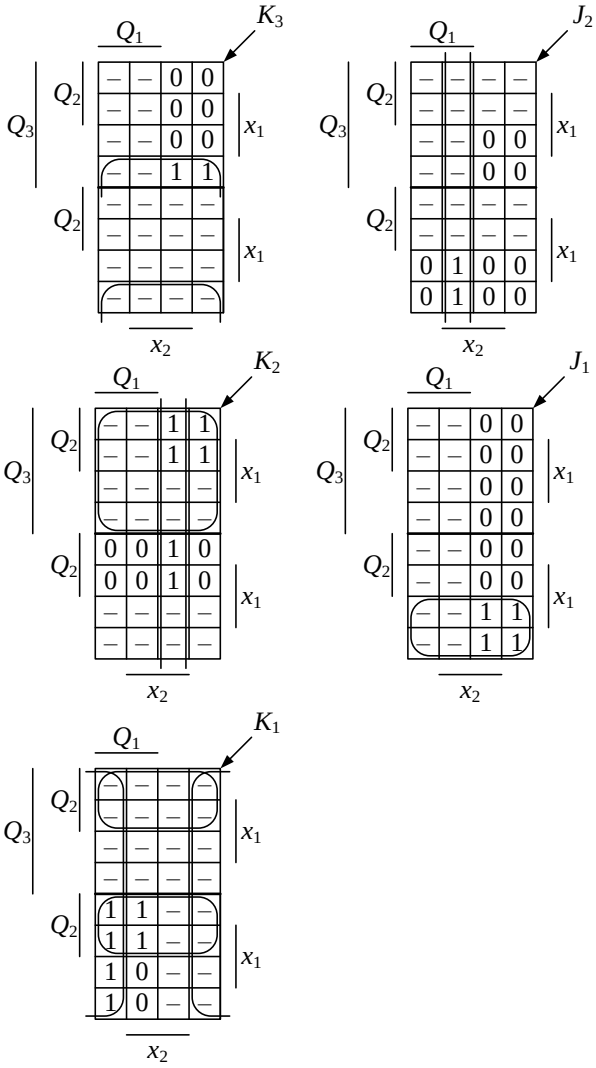


Рис. А2-3.7. Діаграми Вейча для функцій управляючих сигналів та функцій збудження тригерів



Закінчення рис. А2-3.7

Функціональну схему управляючого автомата Мілі (рис. А2-3.8) будуємо за отриманими формами функцій управляючих сигналів та функцій збудження тригерів.

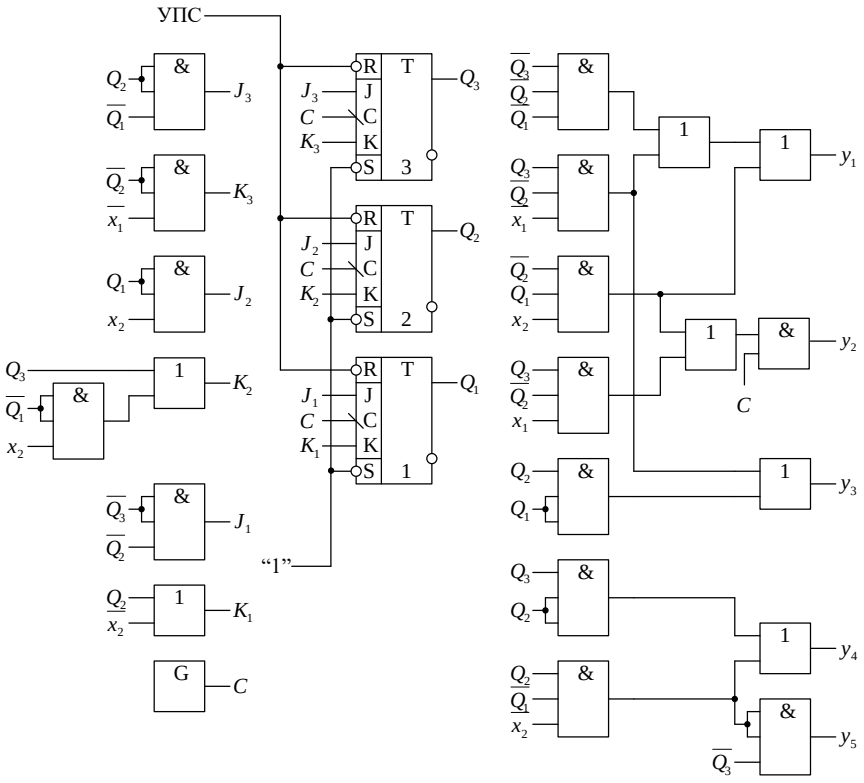


Рис. А2-3.8. Функціональна схема
управляючого автомата Мілі

Приклад 2-3.3

Завдання. Виконати синтез синхронного автомата Мілі з використанням апарата часових функцій. Структурна ГСА автомата зображена на рис. А2-3.9. Побудувати функціональну схему управляючого автомата.

Зробимо розмітку станів для автомата Мілі, помітивши стани автомату символами z_i. Отримаємо ГСА, позначену чотирма різними станами.

На підставі розміченої ГСА будуємо граф автомату (рис. А2-3.10).

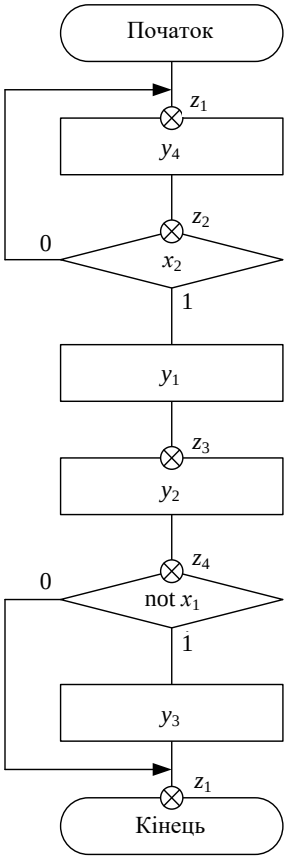


Рис. А2-3.9. ГСА автомата

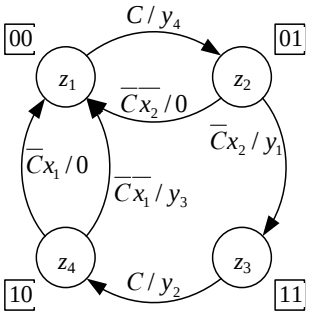


Рис. А2-3.10. Граф автомата Мілі з сусіднім кодуванням станів

Для виконання сусіднього кодування скористаємось шаблоном для двохрозрядних кодів станів автомату (рис. А1-3.17, *а*). Отримані коди станів автомату наведені в табл. А2-3.6.

Таблиця А2-3.6

ТАБЛИЦЯ СУСІДНЬОГО КОДУВАННЯ СТАНІВ АВТОМАТА

Стан	Код стану	
	Q_2	Q_1
z_1	0	0
z_2	0	1
z_3	1	1
z_4	1	0

Використовуючи граф автомата, будуємо структурну таблицю автомату Мілі (табл. А2-3.7), де ПС — початковий стан, СП — стан переходу.

Таблиця А2-3.7

ТАБЛИЦЯ СТРУКТУРНОГО СИНТЕЗУ АВТОМАТУ МІЛІ

Перехід	Код ПС	Код СП	Логічні умови	Керуючі сигнали	Часові функції			
ПС→ СП	$Q_2^t\ Q_1^t$	$Q_2^{t+1}\ Q_1^{t+1}$	$C\ x_2\ x_1$		F_2	φ_2	F_1	φ_1
$z_1 \to z_2$	0 0	0 1	$C - -$	y_4	0	0	1	0
$z_2 \to z_1$	0 1	0 0	$\overline{C}\ \overline{x_2} -$	0	0	0	0	1
$z_2 \to z_3$	0 1	1 1	$\overline{C}\ x_2 -$	y_1	1	0	0	0
$z_3 \to z_4$	1 1	1 0	$C - -$	y_2	0	0	0	1
$z_4 \to z_1$	1 0	0 0	$\overline{C} - x_1$	0	0	1	0	0
$z_4 \to z_1$	1 0	0 0	$\overline{C} - \overline{x_1}$	y_3	0	1	0	0

На основі таблиці структурного синтезу автомату виконаємо синтез часових функцій:

$$F_1 = \overline{Q_2}\,\overline{Q_1}C;$$

$$\varphi_1 = \overline{Q_2}Q_1\overline{C}x_2 \vee Q_2Q_1C;$$

$$Q_1^{t+1} = F_1 \vee Q_1^t\overline{\varphi_1} = \overline{Q_2}\,\overline{Q_1}C \vee Q_1^t\overline{\overline{Q_2}\,\overline{Q_1}\,\overline{x_2} \vee Q_2Q_1C};$$

$$F_2 = \overline{Q_2}Q_1\overline{C}x_2;$$

$$\varphi_2 = Q_2\overline{Q_1}\,\overline{C}x_1 \vee Q_2\overline{Q_1}\,\overline{C}\,\overline{x_1} = Q_2\overline{Q_1}\,\overline{C};$$

$$Q_2^{t+1} = F_2 \vee Q_2^t\overline{\varphi_2} = \overline{Q_2}Q_1\overline{C}x_2 \vee Q_2^t\overline{Q_2\overline{Q_1}\,\overline{C}}.$$

Застосовуючи таблицю, побудуємо діаграми Вейча для функцій управляючих сигналів (рис. А2-3.11).

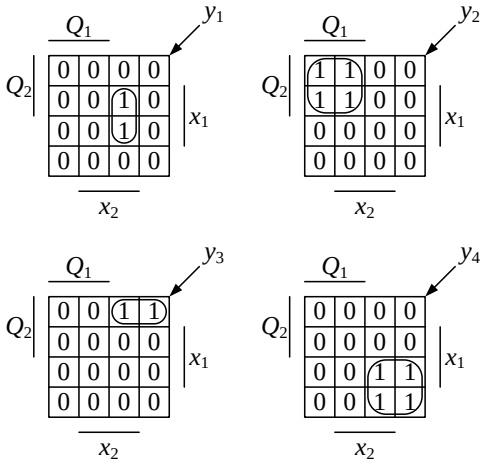


Рис. А2-3.11. Діаграми Дейча
для функцій управляючих сигналів

У результаті мінімізації одержуємо МДНФ функцій управляючих сигналів:

$$y_1 = \overline{Q_2}Q_1x_2;$$

$$y_2 = Q_1Q_2;$$

$$y_3 = Q_2\overline{Q_1}\,\overline{x_1};$$

$$y_4 = \overline{Q_2}\,\overline{Q_1}.$$

За отриманими виразами часових функцій та функцій управляючих сигналів побудуємо функціональну схему автомата Мілі (рис. А2-3.12).

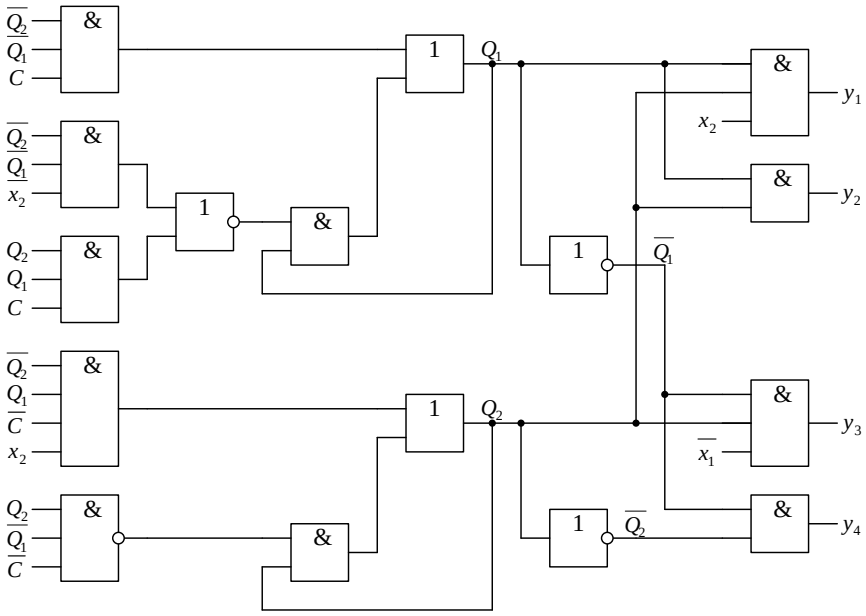


Рис. А2-3.12. Функціональна схема автомата Мілі

А2-4. Типові вузли

Приклад 2-4.1

Завдання. Реалізувати систему перемикальних функцій, розглянута у прикладі 2-2.6 на ПЛМ.

Виконання завдання

Для програмування ПЛМ використовують нормальні форми І /АБО та І /АБО-НЕ.

Розглянемо програмування ПЛМ для реалізації системи перемикальних функцій, що подана в нормальній формі І /АБО (див. приклад 2-2.6):

$$f_1 = x_3\overline{x_2}x_1 \vee \overline{x_4}\,\overline{x_3}\,\overline{x_2} \vee \overline{x_3}\,\overline{x_1}; \qquad \text{(І / АБО)}$$

$$f_2 = \overline{x_4}\,\overline{x_3}\,x_2x_1 \vee \overline{x_3}x_2x_1 \vee x_3x_2\overline{x_1} \vee \overline{x_4}\,\overline{x_3}\,\overline{x_1}; \qquad \text{(І / АБО)}$$

$$f_3 = x_4\overline{x_3}\,\overline{x_2}x_1 \vee x_3x_2x_1 \vee \overline{x_4}\,\overline{x_2}\,\overline{x_1} \vee \overline{x_4}\,\overline{x_3}\,\overline{x_2}. \qquad \text{(І / АБО)}$$

Позначимо терми системи перемикальних функцій:

$$P_1 = x_3\overline{x_2}x_1;\ \ P_2 = \overline{x_4}\,\overline{x_3}\,\overline{x_2};\ \ P_3 = \overline{x_3}\,\overline{x_1};\ \ P_4 = x_4\overline{x_3}x_2x_1;$$
$$P_5 = \overline{x_3}x_2x_1;\ \ P_6 = x_3x_2\overline{x_1};\ \ P_7 = \overline{x_4}\,\overline{x_3}\,\overline{x_1};\ \ P_8 = x_3x_2x_1;$$
$$P_9 = \overline{x_4}\,\overline{x_2}\,\overline{x_1}.$$

Тоді функції виходів описуються системою:

$$f_1 = x_3\overline{x_2}x_1 \vee \overline{x_4}\,\overline{x_3}\,\overline{x_2} \vee \overline{x_3}\,\overline{x_1} = P_1 \vee P_2 \vee P_3;$$
$$f_2 = x_4\overline{x_3}\,\overline{x_2}x_1 \vee \overline{x_3}x_2x_1 \vee x_3x_2\overline{x_1} \vee \overline{x_4}\,\overline{x_3}\,\overline{x_1} = P_4 \vee P_5 \vee P_6 \vee P_7;$$
$$f_3 = x_4\overline{x_3}\,\overline{x_2}x_1 \vee x_3x_2x_1 \vee \overline{x_4}\,\overline{x_2}\,\overline{x_1} \vee \overline{x_4}\,\overline{x_3}\,\overline{x_2} = P_4 \vee P_8 \vee P_9 \vee P_2.$$

Визначимо мінімальні параметри ПЛМ:

$n = 4$ — число інформаційних входів, що дорівнює кількості аргументів системи перемикальних функцій;

$p = 9$ — число проміжних внутрішніх шин, яке дорівнює кількості різних термів системи;

$m = 3$ — число інформаційних виходів, котре дорівнює кількості функцій виходів.

Побудуємо спрощену мнемонічну схему ПЛМ (4, 9, 3) (рис. А2-4.1).

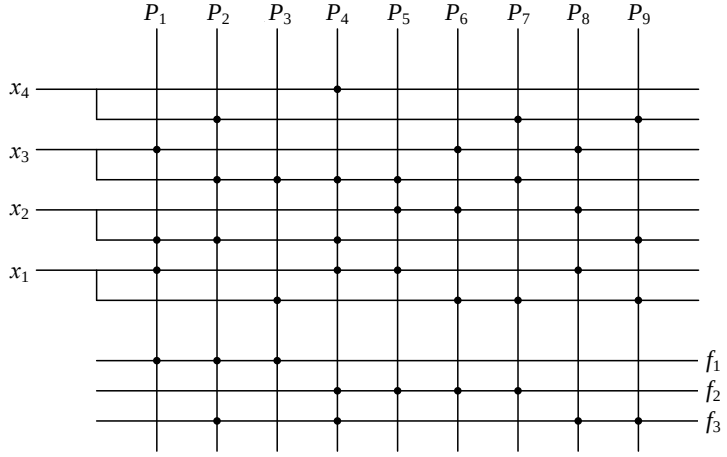


Рис. А2-4.1. Мнемонічна схема ПЛМ (4, 9, 3)

Складемо карту програмування ПЛМ (4, 9, 3) (табл. А2-4.1).

Таблиця А2-4.1

КАРТА ПРОГРАМУВАННЯ ПЛМ

№ шини	Входи				Виходи		
	x_4	x_3	x_2	x_1	f_1	f_2	f_3
1	–	1	0	1	1	–	–
2	0	0	0	–	1	–	1
3	–	0	–	0	1	–	–
4	1	0	0	1	–	1	1
5	–	0	1	1	–	1	–
6	–	1	1	0	–	1	–
7	0	0	–	0	–	1	–
8	–	1	1	1	–	–	1
9	0	–	0	0	–	–	1

Приклад 2-4.2

Завдання. Побудувати двійковий лічильник із коефіцієнтом перерахування $k = 10$. Для побудови лічильника застосувати *JK*-тригери.

Таблиця А2-4.2

ТАБЛИЦЯ ПЕРЕХОДІВ ЛІЧИЛЬНИКА

Стан лічильника								Функції збудження тригерів							
ПС				НС											
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	–	0	–	0	–	1	–
0	0	0	1	0	0	1	0	0	–	0	–	1	–	–	1
0	0	1	0	0	0	1	1	0	–	0	–	–	0	1	–
0	0	1	1	0	1	0	0	0	–	1	–	–	1	–	1
0	1	0	0	0	1	0	1	0	–	–	0	0	–	1	–
0	1	0	1	0	1	1	0	0	–	–	0	1	–	–	1
0	1	1	0	0	1	1	1	0	–	–	0	–	0	1	–
0	1	1	1	1	0	0	0	1	–	–	1	–	1	–	1
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
1	0	0	0	1	0	0	1	–	0	0	–	0	–	1	–
1	0	0	1	0	0	0	0	–	1	0	–	0	–	–	1

Виконання завдання

Для побудови лічильника будуємо таблицю переходів лічильника (табл. А2-4.2), де у відповідному стовпці кожного рядку записуємо коди станів лічильника до надходження чергового рахункового сигналу (ПС) (момент часу t) і після його надходження (СП) (момент часу (t +1)). Значення функцій збудження тригерів визначаються відповідно до системи переходів *JK*-тригера (рис. А1-3.4, *б*).

Значення функцій збудження тригерів одержуємо за допомогою діаграм Вейча (рис. А2-4.2).

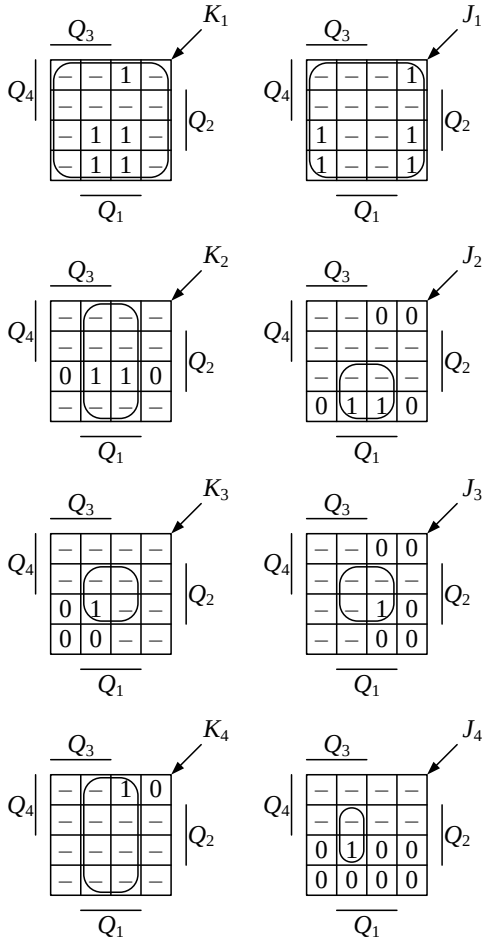


Рис. А2-4.2. Діаграми Вейча функцій збудження тригерів

Виконавши мінімізацію, отримаємо аналітичний запис функцій збудження тригерів:

$$J_1 = 1; \qquad K_1 = 1;$$
$$J_2 = \overline{Q_4}Q_1; \qquad K_2 = Q_1;$$
$$J_3 = Q_2Q_1; \qquad K_3 = Q_2Q_1;$$
$$J_4 = Q_3Q_2Q_1; \qquad K_4 = Q_1.$$

За отриманими формами будуємо функціональну схему лічильника (рис. А2-4.3), де РВ — рахунковий вхід.

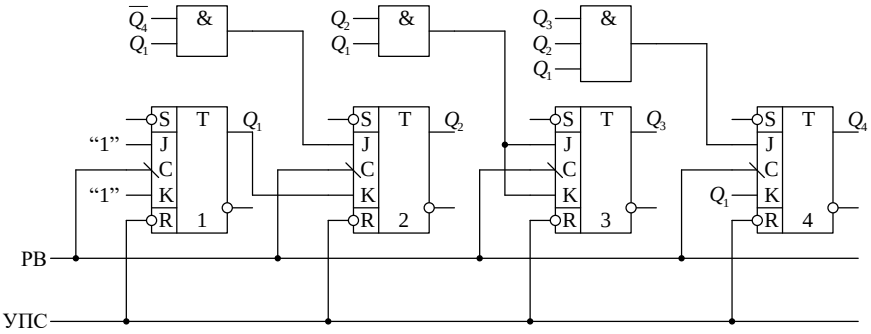


Рис. А2-4.3. Функціональна схема лічильника

А3. ЛАБОРАТОРНІ РОБОТИ

Кожній лабораторній роботі передує самостійна підготовка студентів, у процесі якої вони докладно вивчають опис лабораторної роботи, відповідні розділи посібника й додаткові літературні джерела. Розділи посібника для підготовки до роботи вказані в заголовках робіт. Додаткова література наведена в кінці опису кожної роботи.

У процесі підготовки складається звіт про лабораторну роботу, в якому відображуються всі пункти завдання, а також заготовлені необхідні для експериментальної частини лабораторної роботи таблиці, осі часових діаграм, комбінаційні схеми тощо.

Перед виконанням лабораторної роботи результати підготовки перевіряються викладачем. Студент має сформувати ціль і порядок виконання лабораторної роботи, представити підготовлений звіт і відповісти на контрольні питання. Непідготовлений студент до виконання лабораторної роботи не допускається.

Перед виконанням наступної лабораторної роботи студент повинен представити цілком оформлений звіт за попередню роботу. Звіт має містити короткі теоретичні відомості, необхідні для виконання завдання, відповіді на контрольні питання, усі схеми, формули, таблиці, діаграми, графіки, отримані під час виконання завдання та в процесі експериментального дослідження схем, а також висновки по роботі. Студент, який не представив звіт, не допускається до виконання наступної лабораторної роботи. Оцінку за виконання лабораторної роботи студент одержує після співбесіди з викладачем за темою лабораторної роботи.

Для проведення лабораторних робіт використовуються програмний комплекс моделювання цифрових схем ПРОГМОЛС 2.0 (див. додаток 1).

А3-1. Лабораторна робота 1

Проектування комбінаційних схем

[А1-1.5, А1-1.9**]**

Ціль роботи: оволодіти методами побудови комбінаційних схем у заданому елементному базисі, визначення складності і дослідження швидкодії комбінаційних схем.

Підготовка до роботи

1. Визначити свій варіант перемикальної функції. Для цього необхідно номер варіанта перевести в двійкову систему числення і за-

писати шість його молодших розрядів у вигляді слова $h_6\,h_5\,h_4\,h_3\,h_2\,h_1$. Значення h_i підставити в табл. А3-1.1 та А3-1.2. Наприклад, якщо номер варіанта 19 (у двійковій системі 010011), то $h_6 = 0$, $h_5 = 1$, $h_4 = 0$, $h_3 = 0$, $h_2 = 1$, $h_1 = 1$.

Таблиця А3-1.1

ПАРАМЕТРИ ПЕРЕМИКАЛЬНОЇ ФУНКЦІЇ

x_3	x_2	x_1	y
0	0	0	h_6
0	0	1	h_5
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	h_4

Таблиця А3-1.2

ХАРАКТЕРИСТИКИ ЕЛЕМЕНТІВ

h_3	h_2	h_1	Тип елементу	Час затримки, t
0	0	0	3І-НЕ, 3І	10 14
0	0	1	4І-НЕ, 2АБО	10 12
0	1	0	4І, 2АБО	14 12
0	1	1	3І, 2АБО	14 12
1	0	0	2АБО-НЕ, 4І	12 14
1	0	1	2І-НЕ, 2АБО	10 12
1	1	0	2АБО-НЕ, 3І	10 14
1	1	1	2І-НЕ, 2АБО-НЕ	10 12

2. Знайти ДДНФ функції і її заперечення. Подати функцію у восьми нормальних формах.

3. Одержати операторні представлення функції, що можуть бути реалізовані на елементах, заданих табл. А3-1.2.

4. Вибрати операторні форми, що забезпечують одержання комбінаційної схеми з мінімальною затримкою сигналів і комбінаційної схеми з мінімальною складністю за Квайном, тобто схему з мінімальними значенням T і K.

5. Побудувати зазначені комбінаційні схеми.

Порядок виконання роботи

1. Побудувати моделі заданих комбінаційних схем за допомогою програмного комплексу ПРОГМОЛС 2.0.

2. Переконатися в правильності функціонування моделей.

3. Визначити часові параметри комбінаційних схем за допомогою часових діаграм.

Контрольні питання

1. Сформулювати визначення перемикальної функції, логічного елемента, комбінаційної схеми.

2. Визначити основні властивості комбінаційних схем.

3. У чому сутність задач аналізу і синтезу комбінаційних схем?

4. Охарактеризувати основні етапи синтезу комбінаційних схем.

5. Що таке операторне представлення функції?

6. Як визначити складність і швидкодію комбінаційних схем?

7. Чим пояснюється можливість виникнення збоїв комбінаційних схем за збільшення частоти подачі змінних на їх входи?

8. Як за допомогою однотипних логічних елементів з фіксованою кількостью входів реалізувати функцію І (І-НЕ, АБО, АБО-НЕ), якщо кількість букв в термі і кількість входів логічних елементів не співпадають?

9. Покажіть, яким чином функцію, що реалізована на елементі 2АБО-НЕ, можна реалізувати на логічних елементах 2І-НЕ і навпаки.

10. За допомогою елементів І, АБО і НЕ розробити комбінаційну схему для кожного з виразів:

а) $x = \mathrm{AB}(\mathrm{C} \vee \mathrm{D})$;

б) $z = (\overline{\mathrm{A} \vee \mathrm{B} \vee \mathrm{CDE}}) \vee \overline{\mathrm{B}}\mathrm{C}\overline{\mathrm{D}}$;

в) $y = (\mathrm{A} \vee \mathrm{B})(\overline{\mathrm{A}} \vee \overline{\mathrm{B}})$.

11. Подати повну таблицю істинності для комбінаційних схем, отриманих в п.10.

Додаткова література: **[**6, 7, 15, 17, 19**]**

А3-2. Лабораторна робота 2

Мінімізація перемикальних функцій

[А1-2.2, А1-2.3, А1-2.5, А1-2.11**]**

Ціль роботи: оволодіти методами мінімізації перемикальних функцій, визначення операторних форм функцій, дослідження параметрів комбінаційних схем.

Підготовка до роботи

1. Визначити свій варіант перемикальної функції, заданої таблицею істинності (табл. А3-2.1). Для цього необхідно одержати дев'ять молодших розрядів номера варіанту, поданого в двійковій системі числення ($h_9\ h_8\ h_7$ ... h_1), а потім підставити h_i в табл. А3-2.1 та А3-2.2.

2. Виконати мінімізацію функції f_4 та її заперечення такими методами:

— Квайна;

— Квайна—Мак-Класкі;

— Діаграм Вейча.

Таблиця А3-2.1

ПЕРЕМИКАЛЬНА ФУНКЦІЯ

x_4	x_3	x_2	x_1	f_4
0	0	0	0	0
0	0	0	1	1
0	0	1	0	h_3
0	0	1	1	h_4
0	1	0	0	0
0	1	0	1	h_5
0	1	1	0	0
0	1	1	1	h_6
1	0	0	0	h_7
1	0	0	1	1
1	0	1	0	h_8
1	0	1	1	h_2
1	1	0	0	1
1	1	0	1	h_9
1	1	1	0	h_1
1	1	1	1	1

3. Одержати всі можливі операторні форми подання перемикальної функції за її реалізації в заданому елементному базисі (табл. А3-2.2).

4. Побудувати комбінаційні схеми, що відповідають отриманим операторним формам функцій.

5. Визначити параметри одержаних схем (складність і затримку сигналів). Час затримки на логічних елементах обрати з таблиці А3-1.2

Таблиця А3-2.2

ЕЛЕМЕНТНА БАЗА

h_{10}	h_9	h_8	Логічні елементи
0	0	0	3І-НЕ, 2І
0	0	1	3І, 4І-НЕ
0	1	0	3АБО, 4І, НЕ
0	1	1	3І, 2АБО, НЕ
1	0	0	2АБО-НЕ, 4І
1	0	1	2І-НЕ, 4АБО
1	1	0	3АБО-НЕ, 3І
1	1	1	3І-НЕ, 3АБО-НЕ

Порядок виконання роботи

1. Побудувати моделі комбінаційних схем, вказаних викладачем, за допомогою програмного комплексу ПРОГМОЛС 2.0.

2. Переконатися в правильності функціонування моделей, визначити практично параметри комбінаційних схем.

Контрольні питання

1. Сформулювати визначення перемикальної функції, конституенти, імпліканти і простої імпліканти перемикальної функції.
2. Що таке доконана, скорочена, тупікова і мінімальна ДНФ?
3. Дати визначення функціонально повної системи перемикальних функцій.
4. У чому сутність проблеми мінімізації перемикальних функцій?
5. Охарактеризувати основні етапи мінімізації перемикальних функцій різними методами.
6. Як побудувати операторні форми подання функцій?
7. Дайте порівняльну оцінку методів мінімізації функцій.

8. Побудуйте комбінаційну схему реалізації функції $z = \overline{(A \vee B)\overline{C}}$ на елементах одного типу.

9. Виконайте те саме для виразу $y = \overline{R\overline{S}T \vee \overline{Q}}$.

10. Побудуйте логічну схему для виразу $z = \overline{A}\overline{B}C$, використовуючи лише елементи АБО-НЕ.

11. Застосовуючи теореми де Моргана, спростіть вираз $y = \overline{A \vee \overline{B} \vee \overline{C}D}$.

12. Які з наведених виразів записані в нормальній формі булевої алгебри?

а) $AB \vee CD \vee E$;

б) $AB(C \vee D)$;

в) $(A \vee B)(C \vee D \vee F)$;

г) $\overline{MN} \vee PQ$.

Додаткова література: **[**5—9, 12, 14**]**

А3-3. Лабораторна робота 3

Мінімізація систем перемикальних функцій

[А1-2.2, А1-2.3, А1-2.8, А1-2.11**]**

Ціль роботи: оволодіння методом мінімізації систем перемикальних функцій, дослідження часових параметрів схем з багатьма виходами.

Підготовка до роботи

1. Визначити свій варіант системи перемикальних функцій (табл. А3-3.1). Для цього необхідно одержати дев'ять молодших розрядів номера варіанту студента, поданого в двійковій системі числення ($h_9\ h_8\ h_7$ ... h_1), а потім підставити h_i в табл. А3-3.1.

2. Виконати спільну мінімізацію функцій та їх заперечень методом Квайна або Квайна—Мак-Класкі.

3. Одержати операторні подання функцій для їх реалізації у формі І / АБО і у формі І / АБО-НЕ. Можна використовувати елементи з будь-яким числом входів, але не більш ніж за чотирма.

4. Побудувати комбінаційні схеми, що відповідають отриманим операторним формам. За необхідності виконати розв'язку схем з урахуванням коефіцієнта розгалуження елементів по виходу, що дорівнює двом.

5. Визначити та порівняти параметри одержаних схем.

Таблиця А3-3.1

ПЕРЕМИКАЛЬНА ФУНКЦІЯ

x_4	x_3	x_2	x_1	f_1	f_2	f_3
0	0	0	0	1	1	h_1
0	0	0	1	h_1	0	h_2
0	0	1	0	h_2	h_1	h_3
0	0	1	1	h_3	h_2	h_4
0	1	0	0	h_4	h_3	1
0	1	0	1	1	h_4	h_5
0	1	1	0	h_5	1	0
0	1	1	1	0	h_5	h_6
1	0	0	0	h_6	0	h_7
1	0	0	1	h_7	h_6	1
1	0	1	0	1	h_7	h_8
1	0	1	1	h_8	1	0
1	1	0	0	0	h_8	0
1	1	0	1	1	0	h_9
1	1	1	0	h_9	1	0
1	1	1	1	0	h_9	1

Порядок виконання роботи

1. Побудувати моделі комбінаційних схем, за допомогою програмного комплексу ПРОГМОЛС 2.0.
2. Переконатися в правильності функціонування моделей.
3. Визначити часові параметри комбінаційних схем за допомогою часових діаграм, порівняти їх з одержаними теоритично.

Контрольні питання

1. Сформулювати визначення ДДНФ системи перемикальних функцій.
2. Схарактеризувати етапи мінімізації системи функцій.
3. Як виконуються операції склеювання і поглинання за мінімізації систем перемикальних функцій?
4. У чому сутність проблеми мінімізації систем перемикальних функцій?
5. Схарактеризувати способи забезпечення заданого коефіцієнта розгалуження елементів по виходу. Вкажіть переваги і недоліки кожного із способів.

6. Як побудувати операторні форми подання перемикальних функцій?

Додаткова література: **[**6, 7, 15, 17, 19**]**

А3-4. Лабораторна робота 4

Мінімізація частково визначених функцій

[А1-2.8, А1-2.9, А1-2.11**]**

Ціль роботи: оволодіння методами мінімізації частково визначених функцій та усунення непередбачених короткочасних сигналів на виходах схем, дослідження параметрів схем.

Підготовка до роботи

1. Визначити свій варіант системи перемикальних функцій (табл. А3-4.1). Для цього необхідно одержати дев'ять молодших розрядів номера варіанта студента, поданого в двійковій системі числення ($h_9\ h_8\ h_7\ ...\ h_1$), а потім підставити h_i в табл. А3-4.1.

2. Виконати спільну мінімізацію заданих перемикальних функцій методом Квайна—Мак-Класкі.

3. Виконати спільну мінімізацію заперечень заданих перемикальних функцій методом Квайна—Мак-Класкі.

4. Одержати операторні подання перемикальних функцій для їх реалізації у формі І-НЕ / І-НЕ і у формі І-НЕ / І. Можна використовувати елементи з будь-яким числом входів, але не більш чотирьох.

5. Побудувати комбінаційні схеми, що відповідають отриманим операторним формам. За необхідності виконати розв'язку схем з урахуванням коефіцієнта розгалуження елементів по виходу, що дорівнює двом.

6. Оцінити можливість формування короткочасних помилкових сигналів в отриманих схемах. Показати способи усунення ризику збою в комбінаційних схемах.

Порядок виконання роботи

1. Побудувати моделі комбінаційних схем, вказаних викладачем, за допомогою програмного комплексу ПРОГМОЛС 2.0.

2. Переконатися в правильності функціонування моделей.

3. За необхідності побудувати фільтри для усунення короткочасних помилкових сигналів на виходах схем.

4. Визначити часові параметри комбінаційних схем за допомогою часових діаграм.

Контрольні питання

1. В яких випадках необхідно виконувати мінімізацію функцій в диз'юнктивних формах, а в яких — в кон'юнктивних формах для одержання дворівневої комбінаційної схеми?

2. Схарактеризувати особливість мінімізації частково визначених функцій.

3. Схарактеризувати етапи мінімізації частково визначених функцій різними методами мінімізації.

4. Схарактеризувати основні особливості спільної мінімізації систем частково визначених перемикальних функцій.

5. Як одержати операторні форми подання перемикальних функцій для певного елементного базису?

Таблиця А3-4.1

ПЕРЕМИКАЛЬНА ФУНКЦІЯ

x_4	x_3	x_2	x_1	f_1	f_2	f_3
0	0	0	0	1	1	1
0	0	0	1	1	1	0
0	0	1	0	1	1	1
0	0	1	1	0	0	0
0	1	0	0	–	0	1
0	1	0	1	0	0	0
0	1	1	0	1	–	–
0	1	1	1	–	–	1
1	0	0	0	1	h_4	h_7
1	0	0	1	0	0	h_8
1	0	1	0	0	0	h_9
1	0	1	1	h_1	0	0
1	1	0	0	1	–	1
1	1	0	1	h_2	h_5	0
1	1	1	0	h_3	h_6	h_{10}
1	1	1	1	1	1	1

6. Чим пояснюється можливість виникнення збоїв комбінаційних схем?

7. Як оцінити апаратурні витрати і швидкодію комбінаційних схем?

8. Як забезпечити заданий коефіцієнт розгалуження елементів по виходу під час побудови комбінаційних схем з багатьма виходами?

9. Як усунути короткочасні помилкові сигнали на виходах комбінаційної схеми за перехідних процесів?

10. Як вибрати тривалість затримки сигналу для фільтру на виході комбінаційної схеми?

Додаткова література: **[**6, 7, 15, 17, 19**]**

А3-5. Лабораторна робота 5

Проектування та дослідження автоматів на тригерах

[А1-3.2, А1.3.3, А1-3.4**]**

Ціль роботи: оволодіння методами синтезу керуючих автоматів з пам'яттю у заданому елементному базисі, одержання навичок в їх експериментальному дослідженні.

Підготовка до роботи

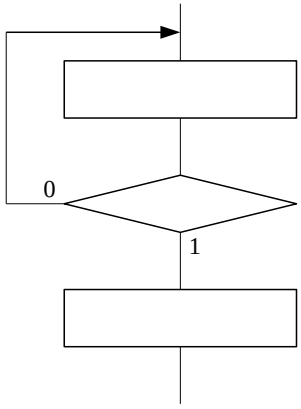
1. Виконати синтез і побудувати функціональну схему управляючого автомата за заданим алгоритмом у заданому елементному базисі, відповідно до варіантів завдання.

Варіант завдання визначається дев'ятьма молодшими розрядами номера варіанту студента, поданого в двійковій системі числення (h_9, h_8, h_7, ..., h_1).

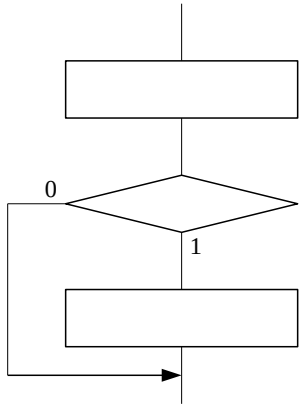
Для одержання алгоритму управління необхідно з'єднати послідовно зверху вниз фрагменти блок-схеми алгоритму (рис. А3-5.1) в порядку, зазначеному в табл. А3-5.1. В кожну логічну вершину отриманої схеми алгоритму, починаючи з верхньої, переписати з табл. А3-5.2 в зазначеному порядку по одному вхідному структурному сигналу. Потім відповідно до табл. А3-5.3 в порядку зверху вниз і зліва направо записати в операторні вершини структурні управляючі сигнали. Сигнали, зазначені в дужках, записують в одну вершину. Отримана блок-схема алгоритму корегується з урахуванням подвоєної тривалості сигналу, зазначеного в табл. А3-5.4 (інші сигнали мають тривалість t).

Тип тригерів і набір логічних елементів, які можна використовувати для побудови автомата, зазначено в табл. А3-5.5 і А3-5.6, а тип автомата визначено в табл. А3-5.7.

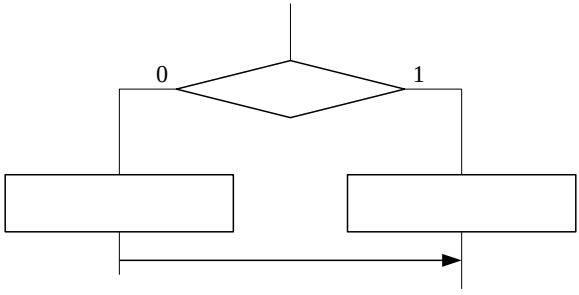
2. Побудувати часову діаграму роботи автомата для кожної комбінацій значень логічних умов.



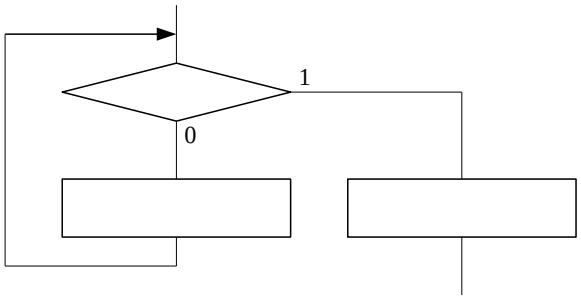
1



2



3



4

Рис. А3-5.1. Фрагменти блок-схеми алгоритму

Таблиця А3-5.1

ВАРІАНТИ ЗАВДАНЬ

h_8	h_4	h_2	Порядок з'єднання фрагментів
0	0	0	1, 4
0	0	1	2, 4
0	1	0	2, 3
0	1	1	4, 2
1	0	0	3, 1
1	0	1	3, 2
1	1	0	4, 1
1	1	1	1, 3

Таблиця А3-5.2

ВАРІАНТИ ЗАВДАНЬ

h_8	h_7	h_3	Логічні умови
0	0	0	$x_1,\ \overline{x_2}$
0	0	1	$x_1,\ x_2$
0	1	0	$\overline{x_1},\ x_2$
0	1	1	$x_1,\ x_2$
1	0	0	$\overline{x_1},\ \overline{x_2}$
1	0	1	$x_1,\ \overline{x_2}$
1	1	0	$\overline{x_1},\ x_2$
1	1	1	$\overline{x_1},\ \overline{x_2}$

Таблиця А3-5.3

ВАРІАНТИ ЗАВДАНЬ

h_9	h_4	h_1	Послідовність управляючих сигналів
0	0	0	$(y_1, y_2), y_3, (y_4, y_5), y_2$
0	0	1	$y_1, (y_1, y_2), y_3, (y_4, y_5)$
0	1	0	$(y_1, y_2), (y_4, y_5), y_2, y_3$
0	1	1	$(y_1, y_2), y_3, y_2, y_3$
1	0	0	$(y_1, y_2), y_3, (y_4, y_5), y_3$
1	0	1	$(y_1, y_2), (y_4, y_5), y_3, y_2$
1	1	0	$y_3, (y_4, y_5), y_2, y_3$
1	1	1	$y_3, (y_4, y_5), (y_1, y_2), y_2$

Таблиця А3-5.4

ВАРІАНТИ ЗАВДАНЬ

h_6	h_2	Сигнал, тривалістю $2t$
0	0	y_1
0	1	y_2
1	0	y_3
1	1	y_4

Таблиця А3-5.5

ВАРІАНТИ ЗАВДАНЬ

h_6	h_5	Тригери
0	0	*RS*
0	1	*D*
1	0	*JK*
1	1	*T*

Таблиця А3-5.6

ВАРІАНТИ ЗАВДАНЬ

h_3	h_2	h_1	Логічні елементи
0	0	0	3І-НЕ, 2І
0	0	1	3І, 4І-НЕ
0	1	0	3АБО, 4І, НЕ
0	1	1	3І, 2АБО, НЕ
1	0	0	2АБО-НЕ, 4І
1	0	1	2І-НЕ, 4АБО
1	1	0	3АБО-НЕ, 3І
1	1	1	3І-НЕ, 3АБО-НЕ

Таблиця А3-5.7

ВАРІАНТИ ЗАВДАНЬ

h_4	Тип автомата
0	Мілі
1	Мура

Порядок виконання роботи

1. Змоделювати та налагодити схему автомата, за допомогою програмного комплексу ПРОГМОЛС 2.0.
2. Дослідити часові параметри схеми автомата.

Контрольні питання

1. Подати узагальнену структурну схему автомата.
2. Написати вирази, що визначають закон функціонування автоматів Мілі і Мура.
3. У чому відмінність автоматів Мілі і Мура?
4. Схарактеризувати основні етапи проектування автомата.
5. Як побудувати граф автомата?
6. Як здійснюється розмітка станів автомата?
7. Від чого залежить кількість тригерів, необхідна для побудови автомата?
8. У чому сутність «протигоночного» кодування станів автомата?
9. Як скласти структурну таблицю автомата?
10. Як побудувати часову діаграму роботи автомата?
11. Скласти таблицю переходів для *JK*-, *RS*-, *T*- і *D*-тригерів.

Додаткова література: **[**6, 7, 13, 15, 17, 19**]**

А3-6. Лабораторна робота 6

Структурний синтез автоматів з використанням апарата часових функцій

[А1-3.2, А1-3.4, А1-3.5**]**

Ціль роботи: вивчити метод структурного синтезу управляючих автоматів із жорсткою логікою з використанням апарата часових функцій, одержати навички в їх налагодженні та експериментальному дослідженні.

Підготовка до роботи

1. Виконати синтез автомата за заданим алгоритмом і побудувати його функціональну схему.

Варіант завдання визначається дев'ятьма молодшими розрядами номера залікової книжки студента, представленого в двійковій системі числення (h_9, h_8, h_7, ..., h_1). Як вихідне десяткове число взяти три останні цифри десяткового номера залікової книжки.

Для одержання вихідного алгоритму керування необхідно з'єднати послідовно зверху вниз фрагменти блок-схеми алгоритму (рис. А3-6.1 в порядку, зазначеному в табл. А3-6.1).

У кожну логічну вершину отриманої схеми алгоритму, починаючи з верхньої, переписати з табл. А3-6.2 у зазначеному порядку по одній логічній умові (одному структурному вхідному сигналу). Згідно з табл. А3-6.4 записати в операторні вершини структурні управляючі сигнали. Тип автомату вибрати відповідно до табл. А3-6.5.

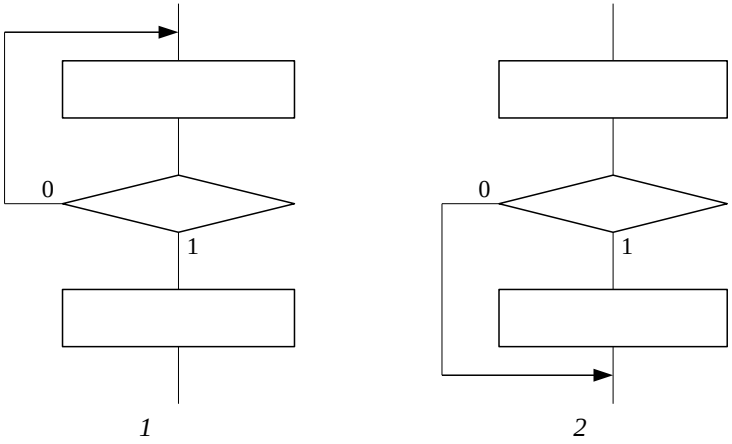
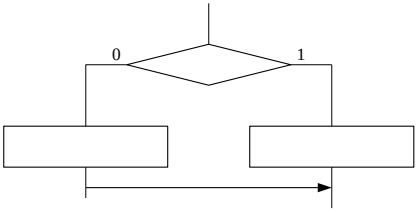
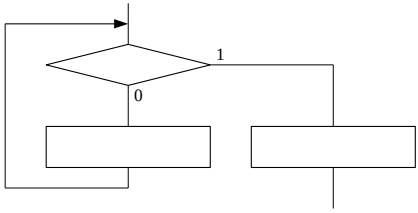


Рис. А3-6.1. Фрагменти блок-схеми алгоритму



3



4

Закінчення рис. А3-6.1

Таблиця А3-6.1

ВАРІАНТИ ЗАВДАНЬ

h_8	h_4	h_2	Порядок з'єднання фрагментів
0	0	0	1, 2
0	0	1	1, 3
0	1	0	2, 3
0	1	1	2, 1
1	0	0	3, 1
1	0	1	3, 2
1	1	0	4, 1
1	1	1	4, 2

Таблиця А3-6.2

ВАРІАНТИ ЗАВДАНЬ

h_7	h_3	Логічні умови
0	0	$x_2,\ \overline{x_1}$
0	1	$x_2,\ x_1$
1	0	$\overline{x_2},\ x_1$
1	1	$\overline{x_2},\ \overline{x_1}$

Таблиця А3-6.3

ВАРІАНТИ ЗАВДАНЬ

h_9	h_4	h_1	Послідовність управляючих сигналів
0	0	0	y_1, y_2, y_3, y_4
0	0	1	y_4, y_1, y_2, y_3
0	1	0	y_3, y_4, y_1, y_2
0	1	1	y_2, y_3, y_4, y_1
1	0	0	y_1, y_2, y_3, y_1
1	0	1	y_1, y_2, y_3, y_2
1	1	0	y_1, y_2, y_3, y_3
1	1	1	y_1, y_2, y_4, y_3

Таблиця А3-6.4

ВАРІАНТИ ЗАВДАНЬ

h_4	Тип автомата
0	Мілі
1	Мура

Порядок виконання роботи

1. Побудувати модель і перевірити правильність роботи автомата за допомогою моделюючого комплексу ПРОГМОЛС 2.0.
2. Визначити зазначені викладачем параметри схеми.

Контрольні питання

1. Дайте означення автомата.
2. Намалюйте узагальнену структурну схему управляючого автомата.
3. Напишіть вирази, що визначають закон функціонування автоматів Мілі і Мура.
4. У чому відмінність автоматів Мілі і Мура?
5. Схарактеризуйте основні етапи проектування цифрових автоматів.
6. Як перейти від змістовного алгоритму до закодованого алгоритму?
7. Як побудувати граф автомата?
8. Як здійснюється оцінка станів автомата?
9. Як визначити необхідну тривалість управляючих сигналів?
10. Від чого залежить кількість часових функцій, необхідних для кодування станів автомата?
11. У чому суть «протигоночного» кодування станів автомата?
12. Як одержати часові функції?
13. Як одержати операторну форму часових функцій?
14. Чи можливий перехід автомата в стан, непередбачений графом?
15. Коли можливе виникнення помилкових управляючих сигналів (непередбачених графом автомата) і чим визначається їх тривалість?
16. Як визначити час переходу автомата з одного стану в інший?

Додаткова література: **[**7, 15, 16**]**

А4. МОДУЛЬНИЙ КОНТРОЛЬ

А4-1. Задачі для самостійного розв'язування

А4-1.1. Перемикальні функції, логічні елементи, комбінаційні схеми

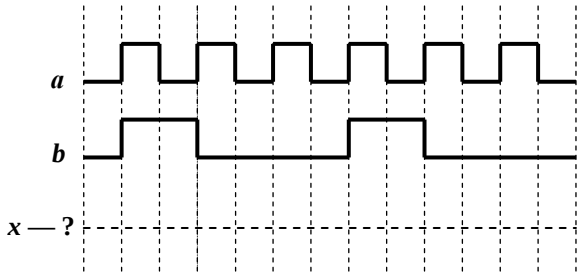
1.1. Побудувати таблицю істинності функції, що реалізує логічний елемент:

а) І,
б) АБО,
в) І-НЕ,
г) АБО-НЕ.

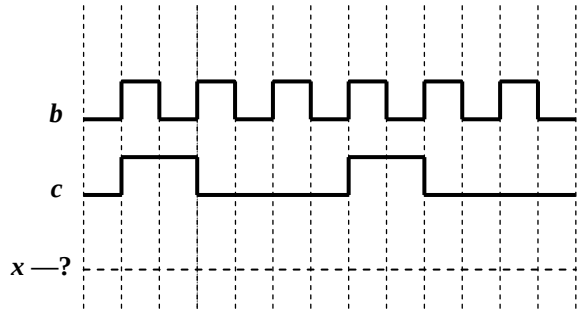
1.2. Подати часову діаграму вихідного сигналу *x* логічного елементу (без урахування затримки сигналів на елементі)

а) І,
б) АБО,
в) І-НЕ,
г) АБО-НЕ,

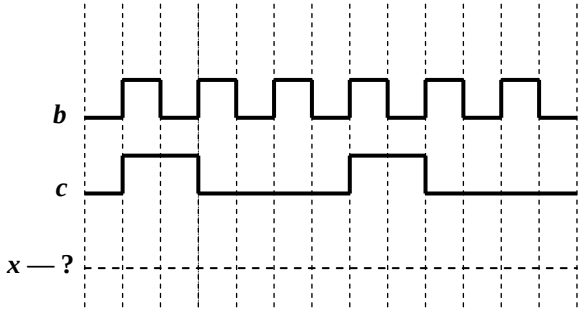
якщо діаграма вхідних сигналів ***a*** і ***b*** наведена на рисунку:



1.3. Визначити форму вихідного сигналу ***x***, якщо на вхід ***a*** логічного елементу **АБО**, що має три входи (***a, b, c***), постійно подавати нулевий сигнал. Діаграма вхідних сигналів наведена на рисунку:



1.4. Визначити форму вихідного сигналу ***x***, якщо на вхід ***a*** логічного елементу **АБО**, що має три входи (***a, b, c***), постійно подавати одиничний сигнал. Діаграма вхідних сигналів наведена на рисунку:



1.5. Вихід логічного елементу **І,** що має два входи, з'єднаний із входом інвертора. Побудувати таблицю істинності, що містить значення функції ***y*** для усіх наборів вхідних сигналів ***a*** і ***b***.

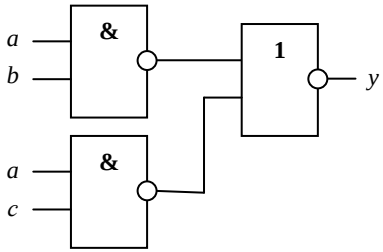
1.6. Яку комбінацію вхідних сигналів необхідно подати на вхід елемента для з'ясування його типу, якщо тип елементу може бути як **І**, так і **АБО**?

1.7. Перемикальна функція задана таким виразом:

$$y = ac \vee b\bar{c} \vee \bar{a}bc\ .$$

Побудувати таблицю істинності для заданої перемикальної функції.

1.8. Записати логічний вираз та побудувати таблицю істинності, що описують функціонування заданої комбінаційної схеми.



1.9. Застосовуючи логічні елементи І, АБО, НЕ, побудувати комбінаційні схеми, що реалізують такі перемикальні функції:

a) $y = ac \vee b\bar{c} \vee \bar{a}\bar{b}$;

b) $y = \overline{ab(c \vee d)}$;

c) $y = \overline{(a \vee b \vee \bar{c}d\bar{e})} \vee \bar{b}c\bar{d}$;

d) $y = \overline{a \vee b\bar{c}}$;

e) $y = \overline{a \vee b} \vee \bar{c}d$;

f) $y = ab(c \vee \bar{d})$;

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

1.10. Застосовуючи логічні елементи ЗАБО-НЕ реалізувати такі перемикальні функції:

a) $y = (d \vee \overline{(a \vee b) \cdot c}) \cdot d$;

b) $y = (d \vee \overline{(a \vee b) \cdot c}) \cdot e$;

c) $y = ac \vee b\bar{c} \vee \bar{a}bc$;

d) $y = \overline{(a \vee b \vee \bar{c}d\bar{e})} \vee \bar{b}c\bar{d}$.

Визначити кількість логічних елементів і складність комбінаційної схеми за Квайном.

1.11. Застосовуючи логічні елементи ЗІ-НЕ реалізувати такі перемикальні функції:

a) $y = (d \vee \overline{(a \vee b) \cdot c}) \cdot d$;

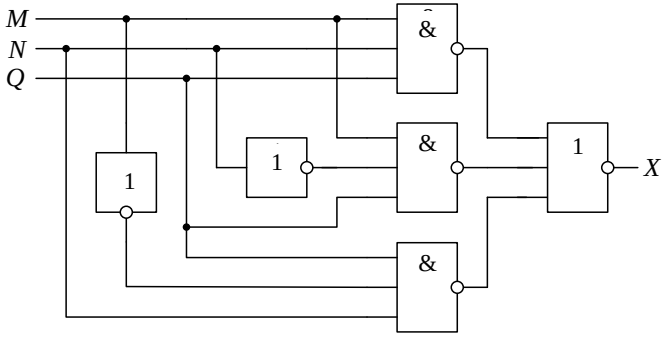
b) $y = (d \vee \overline{(a \vee b) \cdot c}) \cdot e$;

c) $y = ac \vee b\bar{c} \vee \bar{a}bc$;

d) $y = \overline{(a \vee b \vee \bar{c}d\bar{e})} \vee \bar{b}c\bar{d}$.

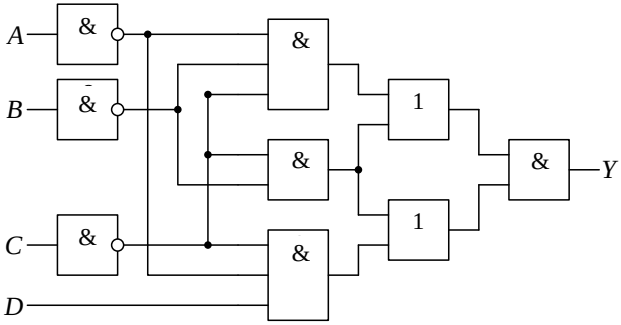
Визначити кількість логічних елементів і складність комбінаційної схеми за Квайном.

1.12. Спростити комбінаційну схему, використовуючи елементи алгебри Буля.

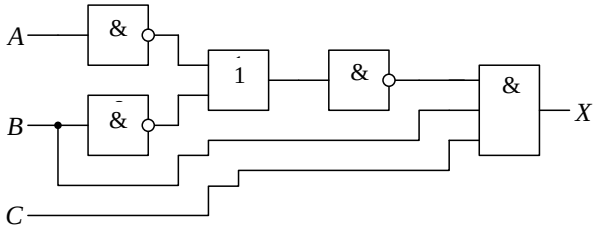


1.13. Задану комбінаційну схему побудовами з мінімальною складністю за Квайном на логічних елементах:

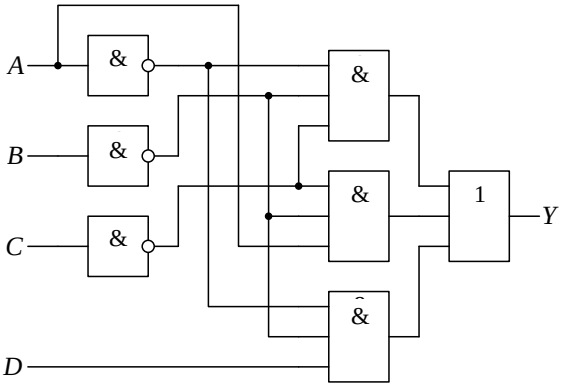
a) І-НЕ,
b) АБО-НЕ.



1.14. Записати аналітичний вираз для виходу X комбінаційної схеми. Визначити значення X для всіх можливих вхідних сигналів, тобто побудувати таблицю істинності.



1.15. Побудувати таблицю істинності функції, що реалізує комбінаційна схема.



1.16. Реалізувати перемикальну функцію y в базисі елементів АБО-НЕ і І-НЕ з мінімальною складністю за Квайном.

a) $y_{\text{ДДНФ}} = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6,$

b) $y_{\text{ДДНФ}} = 4 \vee 5 \vee 6,$

c) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 2 \vee 3 \vee 7,$

d) $y_{\text{ДДНФ}} = 1 \vee 2 \vee 3 \vee 5 \vee 7.$

1.17. Поясніть за допомогою часової діаграми можливість збою комбінаційної схеми під час реалізації функції, якщо на вхід схеми подавати тільки пряме значення x.

$$y = (A \vee x)(B \vee \bar{x}).$$

Побудувати схему, що усуває можливість збою.

1.18. У комбінаційній схемі, що реалізує функцію

$$y = (d \vee \overline{(a \vee b) \cdot c}) \cdot d$$

в булевому базисі, змінити логічні елементи І на І-НЕ та АБО на АБО-НЕ. Виконати аналіз отриманої комбінаційної схеми, подати функцію, що реалізує схема, в ДДНФ.

1.19. Подати перемикальні функції в нормальних формах алгебр Буля, Пірса, Шеффера, Жегалкіна.

a) $f = x_2 \overline{x_1} \vee \overline{x_3} x_2 x_1 \vee \overline{x_2} x_1,$

b) $f = x_3 x_1 \vee \overline{x_2} \overline{x_1} \vee \overline{x_3} \overline{x_2} \overline{x_1},$

c) $f = x_3 \overline{x_1} \vee \overline{x_3} x_2 \overline{x_1} \vee \overline{x_2} \overline{x_1},$

d) $f = x_3 \overline{x_2} \vee x_3 x_2 x_1 \vee \overline{x_3} \overline{x_1}.$

1.20. Визначити приналежність перемикальних функцій до п'яти передповних класів:

a) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 7,$

b) $y_{\text{ДДНФ}} = 1 \vee 2 \vee 3 \vee 4 \vee 7,$

c) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 11 \vee 12 \vee 13,$

d) $y_{\text{ДДНФ}} = 1 \vee 2 \vee 6 \vee 7 \vee 10 \vee 11 \vee 14.$

1.21. Довести, чи є функціонально повною система з двох функцій:

a) І та ВИКЛЮЧНЕ АБО,

b) І-НЕ та АБО,

c) І та АБО –НЕ,

d) АБО та НЕ,

e) НЕ та ВИКЛЮЧНЕ АБО.

1.22. Довести, справедливість співвідношення:

a) $x_3x_2\overline{x_1} \oplus x_1x_2\overline{x_3} = x_1x_2\overline{x_3} \vee x_3x_2\overline{x_1}$

b) $x_3x_2\overline{x_1} \vee \overline{x_3}x_2x_1 = x_3x_2\overline{x_1} \oplus \overline{x_3}x_2x_1$.

1.23. Довести, чи виконується властивість асоціативності в алгебрах

a) Буля,

b) Пірса,

c) Шеффера.

1.24. Довести, чи виконується властивість дистрибутивності в алгебрах

a) Буля,

b) Пірса,

c) Шеффера.

1.25. Використовуючи розкладення Шенона, виключити три змінні та реалізувати функцію на логічних елементах алгебри Буля у вигляді декомпозиції залишкових функцій. Набори, на яких функція приймає значення одиниці, задані їх номерами.

$$A = 3 \vee 8 \vee 9 \vee 11 \vee 19 \vee 24 \vee 25 \vee 27.$$

А4-1.2. Мінімізація перемикальних функцій

2.1. Довести справедливість співвідношення узагальненого склеювання

$$Ax \vee B\overline{x} = Ax \vee B\overline{x} \vee AB,$$

2.2. Перевірити справедливість співвідношення

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

2.3. Довести справедливість співвідношення неповного склеювання для кон'юнктивних форм

$$(A \vee \overline{x})\,(A \vee x) = (A \vee \overline{x})\,(A \vee x)\,A.$$

2.4. Довести справедливість співвідношення неповного склеювання для диз'юнктивних форм

$$Ax \vee A\overline{x} = Ax \vee A\overline{x} \vee A.$$

2.5. Одержати скорочену ДНФ перемикальної функції, заданої у вигляді

a) $f = x_2\overline{x_1} \vee \overline{x_3}x_2x_1 \vee \overline{x_2}x_1$,

b) $f = x_3x_1 \vee x_2\overline{x_1} \vee \overline{x_3}\,\overline{x_2}\,\overline{x_1}$,

c) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 7,$

d) $y_{\text{ДДНФ}} = 1 \vee 2 \vee 6 \vee 7 \vee 10 \vee 11 \vee 14.$

2.6. Скільки кон'юнктивних термів містять МКНФ перемикальних функції, задані у вигляді:

a) $f = x_3 \overline{x_1} \vee x_3 x_2 \overline{x_1} \vee \overline{x_2}\,\overline{x_1},$

b) $f = x_3 \overline{x_2} \vee x_3 x_2 x_1 \vee \overline{x_3}\,\overline{x_1},$

c) $y_{\text{ДДНФ}} = 1 \vee 2 \vee 3 \vee 4 \vee 7,$

d) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 11 \vee 12 \vee 13.$

2.7. Скільки диз'юнктивних термів містять ДКНФ перемикальних функції, задані у вигляді:

a) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 7,$

b) $y_{\text{ДДНФ}} = 1 \vee 2 \vee 3 \vee 4 \vee 6 \vee 7,$

c) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 11 \vee 12 \vee 13,$

d) $y_{\text{ДДНФ}} = 1 \vee 2 \vee 6 \vee 7 \vee 10 \vee 11 \vee 14.$

2.8. Скільки диз'юнктивних термів містять МДНФ перемикальних функції, задані у вигляді:

a) $f = x_3 \overline{x_1} \vee \overline{x_3}\,\overline{x_2}\,\overline{x_1} \vee \overline{x_2} x_1,$

b) $f = x_3 x_2 \vee x_3 \overline{x_2} \vee \overline{x_3} x_2,$

c) $y_{\text{ДДНФ}} = 1 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7,$

d) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 11 \vee 12 \vee 13 \vee 14 \vee 15.$

2.9. За допомогою діаграм Вейча отримати МДНФ функції, задану ДДНФ:

a) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 5 \vee 7,$

b) $y_{\text{ДДНФ}} = 1 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7,$

c) $y_{\text{ДДНФ}} = 1 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7 \vee 9 \vee 10,$

d) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 3 \vee 11 \vee 12 \vee 13 \vee 14 \vee 15.$

2.10. За допомогою діаграм Вейча отримати заперечення МДНФ функції, що задана ДДНФ:

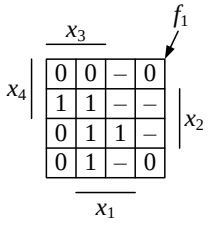
a) $y_{\text{ДДНФ}} = 1 \vee 3 \vee 4 \vee 5 \vee 7,$

b) $y_{\text{ДДНФ}} = 1 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7,$

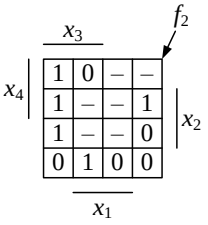
c) $y_{\text{ДДНФ}} = 0 \vee 4 \vee 6 \vee 8 \vee 12 \vee 14,$

d) $y_{\text{ДДНФ}} = 0 \vee 1 \vee 4 \vee 5 \vee 9 \vee 13.$

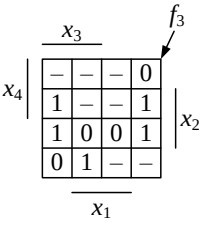
2.11. Отримати МДНФ перемикальних функцій, заданих діаграмами Вейча:



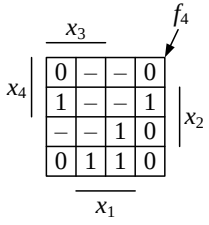
a)



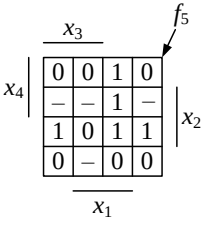
b)



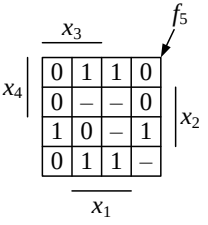
c)



d)



e)



f)

Застосувати такий метод мінімізації:

e) метод Квайна,

f) метод Квайна—Мак-Класкі,

g) метод невизначених коефіцієнтів.

Реалізувати перемикальну функцію у заданому елементному базисі:

h) І, АБО, НЕ, з мінімальною складністю за Квайном,

i) АБО-НЕ,

j) І-НЕ.

2.12. Отримати МДНФ перемикальної функції, заданої у вигляді:

a) $f = x_2\overline{x_1} \vee \overline{x_3}x_2x_1 \vee \overline{x_2}x_1 \vee \overline{x_3}\,\overline{x_1} \vee x_3\overline{x_2}\,\overline{x_1}$,

b) $f = x_2\overline{x_1} \vee \overline{x_3}x_2\overline{x_1} \vee \overline{x_2}x_1 \vee \overline{x_3}x_1 \vee x_3\overline{x_2}\,\overline{x_1}$,

c) $f = x_2x_1 \vee x_3\overline{x_2}\,\overline{x_1} \vee \overline{x_2}x_1 \vee \overline{x_3}x_2 \vee x_3\overline{x_2}x_1$,

d) $f = x_3x_2 \vee x_3\overline{x_2} \vee \overline{x_3}x_2x_1 \vee \overline{x_2}x_1 \vee x_3\overline{x_2}x_1$.

Застосувати такий метод мінімізації:

e) метод Квайна,

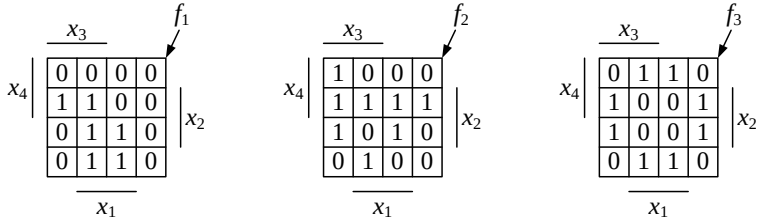
f) метод Квайна—Мак-Класкі,

g) діаграми Вейча,

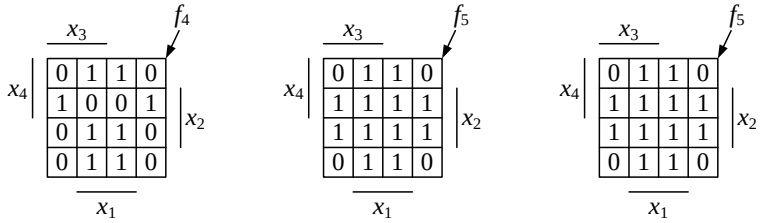
h) метод невизначених коефіцієнтів.

Реалізувати задані перемикальні функції на ПЛМ з мінімальними параметрами.

2.13. Знайти тупікові та мінімальні ДНФ перемикальних функцій, задані діаграмами Вейча:



a) *b*) *c*)



d) *e*) *f*)

2.14. Виконати мінімізацію перемикальної функції, заданої ДДНФ:

a) $y = 0 \vee 1 \vee 2 \vee 3 \vee 4 \vee 8 \vee 10 \vee 11 \vee 15$,

b) $y = 0 \vee 1 \vee 3 \vee 5 \vee 7 \vee 10 \vee 12$,

c) $y = 1 \vee 3 \vee 4 \vee 5 \vee 7 \vee 8 \vee 11$,

d) $y = 0 \vee 2 \vee 3 \vee 4 \vee 6 \vee 8 \vee 11 \vee 12$.

Застосувати такий метод мінімізації:

e) метод Квайна,

f) метод Квайна—Мак-Класкі,

g) діаграми Вейча,

h) метод невизначених коефіцієнтів.

Реалізувати перемикальну функцію у заданому елементному базисі

i) 2АБО-НЕ,

j) 3АБО-НЕ,

k) 2І-НЕ,

l) 3І-НЕ,

m) 2І/3АБО-НЕ,

n) 3І/2АБО-НЕ.

o) 2І/3АБО,

p) 3І/2АБО.

2.15. Отримати операторні форми для реалізації перемикальної функції, поданих у ДДНФ, у заданому елементному базисі з мінімальною складністю.

a) $y = 3 \vee 4 \vee 8 \vee 10 \vee 11 \vee 15$,

b) $y = 0 \vee 1 \vee 3 \vee 5 \vee 7 \vee 9 \vee 10 \vee 12$,

c) $y = 1 \vee 3 \vee 4 \vee 5 \vee 7 \vee 8 \vee 12 \vee 11$,

d) $y = 6 \vee 8 \vee 11 \vee 12 \vee 14 \vee 15$.

Реалізувати перемикальну функцію у заданому елементному базисі

e) 2АБО-НЕ
f) 3І-НЕ
g) 2І/3АБО-НЕ
h) 2І/2АБО
i) 2АБО/3І-НЕ
j) 2АБО-НЕ/2АБО
k) 3І/2АБО-НЕ
l) 2І-НЕ/2І

2.16. Реалізувати на ПЛМ з мінімальними параметрами перемикальну функцію:

a) $y = 3 \vee 4 \vee 8 \vee 10 \vee 11 \vee 15$,

b) $y = 0 \vee 1 \vee 3 \vee 5 \vee 7 \vee 9 \vee 10 \vee 12$,

c) $y = 1 \vee 3 \vee 4 \vee 5 \vee 7 \vee 8 \vee 12 \vee 11$,

d) $y = 6 \vee 8 \vee 11 \vee 12 \vee 14 \vee 15$.

Отримати та порівняти за ефективністю операторні форми І/АБО та І/АБО-НЕ для реалізації на ПЛМ.

2.17. Методом Петрика отримати мінімальну ДНФ перемикальної функції $y(x_3, x_2, x_1)$, якщо СДНФ заданої функції містить імпліканти (імпліканти задані номерами наборів термів, що склеювались):

a) 13, 23, 26, 15, 45, 46,
b) 23, 13, 40, 56, 57,
c) 10, 45, 46, 56, 57,
d) 13, 20, 23, 46, 54.

2.18. Отримати всі відмінні від нуля невизначені коефіцієнти перемикальної функції $f(x_3, x_2, x_1)$:

a) $f = x_2\overline{x_1} \vee \overline{x_3}x_2x_1 \vee \overline{x_2}x_1$,

b) $f = x_2\overline{x_1} \vee \overline{x_3}x_2x_1 \vee \overline{x_2}x_1 \vee \overline{x_3}\,\overline{x_2}x_1$,

c) $f = x_2x_1 \vee x_3x_2\overline{x_1} \vee \overline{x_2}x_1 \vee \overline{x_3}x_2 \vee x_3\overline{x_2}x_1$,

d) $f = x_3\overline{x_1} \vee \overline{x_3}\,\overline{x_2}\,\overline{x_1} \vee \overline{x_2}x_1$.

А4-1.3. Мінімізація систем перемикальних функцій

3.1. Виконати спільну мінімізацію системи перемикальних функцій:

a) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7; \\ B = 0 \vee 2 \vee 4 \vee 5 \vee 6 \vee 7 \vee 12 \vee 14; \end{cases}$

b) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 9 \vee 13; \\ B = 3 \vee 9 \vee 11 \vee 13 \vee 14 \vee 15; \end{cases}$

c) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7; \\ B = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 9; \end{cases}$

d) $\begin{cases} A = 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 10 \vee 12; \\ C = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 12 \vee 14. \end{cases}$

Застосувати такий метод мінімізації:

e) Квайна—Мак-Класкі;
f) Квайна.

Обрати елементний базис:

g) І, АБО, НЕ,
h) АБО-НЕ,
i) І-НЕ.

Визначити складність схеми за Квайном.

3.2. Виконати спільну мінімізацію системи перемикальних функцій

a) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7; \\ B = 0 \vee 2 \vee 4 \vee 5 \vee 6; \end{cases}$

b) $\begin{cases} A = 0 \vee 1 \vee 2 \vee 4 \vee 6; \\ B = 0 \vee 1 \vee 4 \vee 6 \vee 7; \end{cases}$

c) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5; \\ B = 0 \vee 1 \vee 3 \vee 4 \vee 7; \end{cases}$

d) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7; \\ B = 0 \vee 1 \vee 4 \vee 5 \vee 7. \end{cases}$

Застосувати такий метод мінімізації:

e) Квайна—Мак-Класкі,
f) Квайна,
g) діаграми Вейча.

Реалізувати систему перемикальних функцій у заданому елементному базисі:

h) 2АБО-НЕ,
i) 3І-НЕ,
j) 2І/3АБО-НЕ,
k) 2І/2АБО,
l) 2АБО/3І-НЕ,
m) 2АБО-НЕ/2АБО,
n) 3І/2АБО-НЕ,
o) 2І-НЕ/2І.

3.3. За допомогою діаграм Вейча виконати спільну мінімізацію перемикальних функцій:

a) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7; \\ B = 0 \vee 2 \vee 4 \vee 5 \vee 6 \vee 7 \vee 12 \vee 14; \end{cases}$

b) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 9 \vee 13; \\ B = 3 \vee 9 \vee 11 \vee 13 \vee 14 \vee 15; \end{cases}$

c) $\begin{cases} A = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7; \\ B = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 9; \end{cases}$

d) $\begin{cases} A = 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 10 \vee 12; \\ C = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 12 \vee 14. \end{cases}$

Скласти карту програмування ПЛМ для реалізації системи перемикальних функцій. Визначити мінімальні параметри ПЛМ.

А4-1.4. Синтез цифрових автоматів

4.1. За заданою ЛСА виконати абстрактний синтез автомата:

a) $\Pi \quad \overset{1}{x_1}\uparrow y_1 \overset{2}{y_2}\uparrow \overset{1}{\downarrow}\overset{3}{\downarrow} y_1 \overset{2}{\downarrow} x_2 \overset{3}{\uparrow} \quad К,$

b) $\Pi \quad \overset{1}{\downarrow} y_1 \overset{2}{\downarrow} x_1 \overset{1}{\uparrow} y_2 x_2 \overset{2}{\uparrow} \quad К,$

c) $\Pi \quad \overset{2}{\downarrow} y_1 \; x_1 \overset{1}{\uparrow} y_2 \overset{3}{\uparrow} \overset{1}{\downarrow} y_1 \; y_2 \overset{3}{\downarrow} x_2 \overset{2}{\uparrow} \quad К,$

d) $\Pi \quad y_2 \; x_1 \overset{1}{\uparrow} y_1 \; y_2 \overset{1}{\downarrow} x_2 \overset{2}{\uparrow} y_2 \overset{3}{\uparrow} \overset{2}{\downarrow} y_1 \overset{3}{\downarrow} \quad К.$

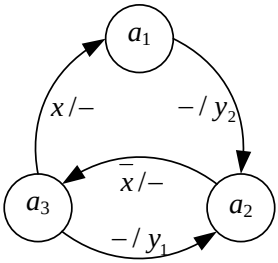
Обрати тип автомата:

e) автомат Мілі,
f) автомат Мура.

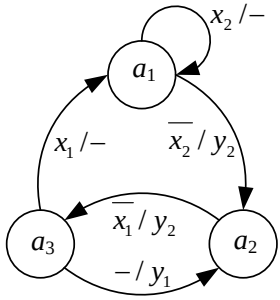
Згідно з результатами абстрактного синтезу виконати етап структурного синтезу, що полягає в отриманні

g) функцій виходів,
h) функції збудження тригерів D_1 та функції виходу y_2,
i) функцій збудження тригерів J_1 та J_2,
j) функцій збудження тригерів T_1, та функції виходів y_1,
k) функцій збудження тригерів R_1 та S_2,
l) функції збудження тригерів K_1 та функції виходу y_1.

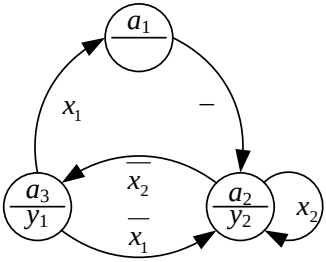
4.2. За заданим графом автомата виконати синтез автомата.



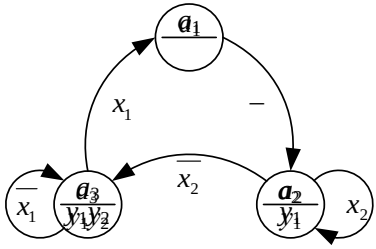
a)



b)



c)



d)

Для побудови функціональної схеми автомата застосувати тригери такого типу:

e) *T*-тригери,
f) *D*-тригери,
g) *RS*-тригери,
h) *JK*-тригери.

Обрати елементний базис:

i) І, АБО, НЕ,
j) АБО-НЕ,
k) І-НЕ.

4.3. За заданою ЛСА виконати абстрактний синтез автомата:

a) П $\overset{1}{x_1}\uparrow y_1y_2 \overset{213}{\uparrow\downarrow\downarrow} y_1\downarrow \overset{2}{x_2}\uparrow \overset{3}{}$ К,

b) П $\overset{1}{\downarrow} y_1 \overset{2}{\downarrow} x_1 \overset{1}{\uparrow} y_2x_2 \overset{2}{\uparrow}$ К,

c) П $\overset{2}{\downarrow}y_1\ x_1 \overset{1}{\uparrow} y_2 \overset{31}{\uparrow\downarrow} y_1\ y_2 \overset{3}{\downarrow} x_2 \overset{2}{\uparrow}$ К,

d) П $\overset{1}{y_2\ x_1}\uparrow y_1\ y_2 \overset{1}{\downarrow} x_2 \overset{2}{\uparrow} y_2 \overset{32}{\uparrow\downarrow} y_1 \overset{3}{\downarrow}$ К.

Обрати тип автомата:

e) автомат Мілі,

f) автомат Мура.

Згідно з результатами абстрактного синтезу виконати етап структурного синтезу синхронного автомата методом часових функцій, що полягає в отриманні

g) функцій Q_1 та Q_2,

h) функцій виходів,

i) функції Q_1 та функції виходу y_2,

j) функції Q_2 та функції виходу y_1.

4.4. Для абстрактного автомата, заданого відміченою таблицею переходів, побудувати ГСА, граф автомата та виконати протигоночне кодування станів.

	y_1	y_2	y_2	y_1	y_1	y_1
	a_1	a_2	a_3	a_4	a_5	a_6
x_1	a_5	a_6	a_4	a_5	a_4	a_4
x_2	a_4	a_3	a_3	a_1	a_2	a_6

4.5. Визначити можливі проміжні стани автомата, якщо він здійснює перехід

a) із стану 0001 в стан 1000,

b) із стану 0100 в стан 0010.

До чого приводить поява проміжних станів автомату, як їх позбутися?

4.6. Закодувати стани автомата методом протигоночного кодування, якщо граф автомата має вигляд чотирикутника з однією діагоналлю.

4.7. Скільки тригерів потрібно для побудови автомата, що має

a) 14 станів,

b) 21 стан.

А4-1.5. Типові вузли

5.1. На елементах булевого базису побудувати
a) мультиплексор МХ (2 — 1),
b) мультиплексор МХ (4 — 1),
c) демультиплексор DMX (1 — 2),
d) демультиплексор DMX (1 — 4).

5.2. Реалізувати на мультиплексорі МХ (4—1) та елементах булевого базису з використанням розкладу за Шеноном, перемикальну функцію, що задана ДДНФ:

a) $y = 1 \vee 3 \vee 4 \vee 5 \vee 7 \vee 8 \vee 11$,

b) $y = 0 \vee 1 \vee 3 \vee 5 \vee 11 \vee 12 \vee 13$,

c) $y = 0 \vee 1 \vee 2 \vee 3 \vee 4 \vee 8 \vee 10 \vee 11 \vee 15$,

d) $y = 0 \vee 1 \vee 3 \vee 5 \vee 7 \vee 10 \vee 12$.

5.3. На елементах булевого базису побудувати:

a) повний дешифратор на 3 входи,
b) шифратор на 3 виходи,
c) неповний дешифратор на 3 входи, де забороненими є набори 4 та 6,
d) неповний дешифратор на 3 входи, де забороненими є набори 5 та 7.

5.4. Побудувати лічильник з коефіцієнтом перерахування $k = 5$.

5.5. Побудувати лічильник з періодом

a) 0, 1, 2, 3,
b) 1, 2, 3, 4, 5,
c) 0, 1, 2, 4, 5, 6.
Застосувати такий тип тригерів:
d) *T*-тригери,
e) *D*-тригери,
f) *RS*-тригери,
g) *JK*-тригери.

5.6. Побудувати однорозрядний двійковий суматор на елементах

a) І-НЕ,
b) АБО-НЕ.

5.7. Виконати декомпозицію функції за Шеноном відносно змінної x_3.

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

Побудувати комбінаційну схему із застосуванням логічних елементів І, АБО, НЕ та мультиплексора МХ (2 — 1).

5.8. Виконати синтез синхронного *RS*-тригера по *MS*-схемі методом прямого синтезу автомата Мура з використанням часових функцій.

5.9. Виконати синтез синхронного *D*-тригера по *MS*-схемі методом прямого синтезу автомата Мура з використанням часових функцій.

5.10. Реалізувати на ПЛМ систему перемикальних функцій без їх мінімізації:

a) $\begin{cases} A = 2 \vee 3 \vee 4 \vee 5 \vee 7; \\ B = 0 \vee 2 \vee 4 \vee 5 \vee 7; \\ C = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7; \end{cases}$

b) $\begin{cases} A = 1 \vee 2 \vee 3 \vee 4 \vee 5; \\ B = 1 \vee 2 \vee 3 \vee 5 \vee 7; \\ C = 0 \vee 1 \vee 2 \vee 3 \vee 4 \vee 7; \end{cases}$

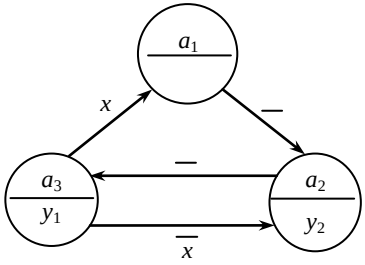
c) $\begin{cases} A = 1 \vee 2 \vee 3 \vee 5 \vee 6; \\ B = 0 \vee 1 \vee 3 \vee 4 \vee 5; \\ C = 0 \vee 1 \vee 2 \vee 3 \vee 6; \end{cases}$

d) $\begin{cases} A = 1 \vee 2 \vee 4 \vee 5 \vee 6; \\ B = 0 \vee 2 \vee 4 \vee 5 \vee 6 \vee 7; \\ C = 0 \vee 1 \vee 2 \vee 4 \vee 5 \vee 6. \end{cases}$

А4-2. Приклади завдань до модульного контролю

Варіант 1

1. За заданим графом автомата Мура визначити МДНФ функцій управління тригерами *D*1 та *D*2.



2. Виконати спільну мінімізацію системи перемикальних функцій трьох аргументів методом Квайна. Побудувати комбінаційну схему в елементному базисі 2АБО-НЕ.

$$A(a, b, c) = 2 \vee 3 \vee 4 \vee 5 \vee 7,$$
$$C(a, b, c) = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7.$$

3. Для перемикальної функції $F(x_3, x_2, x_1)$, що задана СДНФ, визначити МДНФ, використовуючи метод діаграм Вейча.

$$F = x_3 x_2 \overline{x_1} \vee \overline{x_3}\, x_2\, x_1 \vee \overline{x_2}\, x_1 \vee x_3 \overline{x_1}.$$

4. Довести, чи виконується властивість асоціативності в алгебрі Пірса.

Варіант 2

1. За заданою ЛСА побудувати граф автомата Мілі. Подати сусідні коди станів a_i

$$\Pi \quad X_1 \overset{1}{\uparrow} Y_2 \overset{2}{\uparrow} \overset{1}{\downarrow} \overset{3}{\downarrow} Y_1 \overset{2}{\downarrow} X_2 \overset{3}{\uparrow} K.$$

2. Виконати спільну мінімізацію системи перемикальних функцій чотирьох аргументів методом Квайна—Мак-Класкі. Побудувати комбінаційну схему з мінімальною складністю за Квайном.

$$A(a, b, c, d) = 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 10 \vee 12,$$
$$C(a, b, c) = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 12 \vee 14.$$

3. Скільки термів містить СДНФ перемикальної функції $F(x_3, x_2, x_1)$, задана у вигляді:

$$F = x_2 \overline{x_1} \vee \overline{x_3}\, x_2\, x_1 \vee \overline{x_2}\, x_1.$$

4. Довести чи є функціонально повною система перемикальних функцій І та ВИКЛЮЧНЕ АБО.

Варіант 3

1. По заданій ЛСА побудувати граф автомата Мілі. Знайти МДНФ функції управління Y_2.

$$\Pi \quad \overset{2}{\downarrow} Y_1 \quad X_1 \overset{1}{\uparrow} \quad Y_2 Y_3 \overset{1}{\downarrow} \quad X_2 \overset{2}{\uparrow} \quad K.$$

2. Виконати спільну мінімізацію методом Квайна—Мак-Класкі системи перемикальних функцій. Отримати нормальні форми для реалізації перемикальних функцій в елементному базисі 2І-НЕ.

$$A(a, b, c, d) = 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 10 \vee 12,$$
$$C(a, b, c) = 0 \vee 2 \vee 3 \vee 4 \vee 5 \vee 7 \vee 12 \vee 14.$$

3. Побудувати комбінаційну схему, що реалізує МДНФ перемикальної функції $F(x_3, x_2, x_1)$, в елементному базисі 2АБО-НЕ. Визначити складність схеми за Квайном.

$$F(x_3, x_2, x_1) = 0 \vee 2 \vee 4 \vee 5 \vee 7 \vee 8 \vee 10.$$

4. Довести справедливість співвідношення узагальненого склеювання.

Розділ Б

Модуль ІІ. КОМП'ЮТЕРНА АРИФМЕТИКА

Б1. ТЕОРЕТИЧНА ЧАСТИНА

Б1-1. Вступ у комп'ютерну арифметику

Б1-1.1. Системи числення

Системою числення (*численням, нумерацією*) називають сукупність прийомів і правил для позначення й найменування чисел.

У будь-якій системі числення число представляють сукупністю символів, які називають *цифрами*. Кожна цифра в запису числа однозначно подається певною кількістю, що відповідає цієї цифрі. Цю кількість називають *кількісним еквівалентом* даної цифри.

Розрізняють непозиційні й позиційні системи числення.

Система числення називається *непозиційною*, якщо кожній цифрі в будь-якому місці запису числа однозначно відповідає той самий кількісний еквівалент.

Історично більш ранніми є непозиційні системи числення, наприклад, загальновідома римська нумерація. Однак непозиційні системи числення практично не знаходять застосування в цифровій обчислювальній техніці (ОТ), тому що вони характеризуються дуже складними й громіздкими алгоритмами подання чисел і виконання арифметичних операцій.

Система числення називається *позиційною*, якщо одній і тій самій цифрі відповідають різні кількісні еквіваленти залежно від місця розташування цієї цифри у запису числа (номеру розряду).

Непозиційні й позиційні системи числення являють собою дві крайності. У принципі, можливі частково позиційні системи, у яких для однієї множини цифр кількісні еквіваленти постійні, а для іншої множини цифр вони залежать від їх місця розташування в запису числа.

Для визначення кількісного еквіваленту повного запису числа використовується деяка функція від кількісних еквівалентів сукупності цифр у його запису. Якщо використовується функція додавання, то систему числення називають *адитивною*, у випадку використання функції множення систему числення називають *мультиполікативною*. Для більшості існуючих систем числення зазначена фун-

кція є функцією десяткового додавання. У цьому випадку для знаходження кількісного еквівалента числа необхідно просумувати всі кількісні еквіваленти цифр цього числа за правилами десяткової арифметики.

Позиційну систему числення, де кожна цифра має свій певний символ, називають *системою з безпосереднім поданням цифр.* Існують *системи з кодованим поданням цифр*, де кожна цифра кодується певною комбінацією символів, які, як правило, являють собою цифри іншої системи числення.

Переважне поширення в ОТ одержали *однорідні позиційні системи числення*. У таких системах числення за безпосереднього подання цифр число записується у вигляді

$$X = x_s x_{s-1} \ldots x_1 x_0 , \;\; x_{-1} \ldots x_{-m+1} x_{-m} , \qquad \text{(Б1-1.1)}$$

кількісний еквівалент виразу (Б1-1.1) визначається як

$$X = k^s x_s + k^{s-1} x_{s-1} + \ldots + k^1 x_1 + k^0 x_0 + k^{-1} x_{-1} + \ldots + k^{-m} x_{-m} ,$$

у загальному вигляді

$$X = \sum_{i=-m}^{s} k^i x_i , \qquad \text{(Б1-1.2)}$$

де x_i – цифри i-го розряду запису числа, що приймають значення з певної множини,

k — основа системи числення.

Запис числа у певній системі числення називають *кодом* числа. Число подається у n-розрядному коді, відповідно до запису (Б1-1.1), причому $(s + 1)$ розрядів, що розташовані ліворуч від коми, подають цілу частину числа, а m розрядів, розташованих праворуч від коми — дробову частину числа. Відповідно до цього розрядність коду дорівнює $n = s + m + 1$. В розглянутому випадку вага i-го розряду в k раз більше ваги $(i – 1)$-го розряду. Таку систему числення називають системою *із природним порядком ваг (природною).*

Існують системи числення *зі штучним порядком ваг,* для яких зазначене співвідношення ваг сусідніх розрядів не є обов'язковим. Останні приймаються, наприклад, для створення завадостійких кодів і для кодування цифр в інших системах числення з більшою основою.

Системи числення із природним порядком ваг розрізняють за видом основи k. Відомі системи числення з натуральними, від'ємними, дробовими, комплексними основами. Найбільше поширення в ОТ одержали системи числення з *натуральними основами*. Однак інші

системи числення мають ряд особливостей, які у деяких випадках роблять їх більш ефективними, ніж системи з натуральними основами.

До систем числення висувають такі вимоги.

Однозначність. Кожному числу має відповідати єдиний код і навпаки.

Скінченність. Кожному цілому числу має відповідати код кінцевої довжини.

Ефективність. Повинен існувати алгоритм, за допомогою якого здійснюється перехід від коду числа кінцевої довжини до самого числа за кінцеве число кроків. При переході від числа до його коду має існувати алгоритм, який для цілого числа за кінцеве число кроків реалізує цей перехід, а для дробового числа за кінцеве число кроків дозволяє одержати код числа, кількісний еквівалент якого відрізняється від числа не більш ніж на задану величину похибки.

Системи числення з натуральною основою, що задовольняють вимогам однозначності, скінченності, ефективності, називають *канонічними*. Для таких систем цифра 0 (нуль) є обов'язковою, а кількість різних цифр дорівнює основі *k*.

Залежно від виду множин цифр, допустимих у кожному розряді, системи числення розділяють на *симетричні, зміщені* й *кососиметричні.*

Системи числення з непарними натуральними основами $k = 2r + 1$ і цифрами $x_i \in \{-r,\ -r+1,\ ...,\ 0,\ ...,\ r\}$ називають *симетричними*. Такі системи дозволяють подати будь-яке ціле число (додатне або від'ємне) у кінцевому вигляді. Якщо основа системи числення є парним числом, то побудова симетричної канонічної системи стає неможлива. Прикладом симетричної канонічної системи є трійкова система (k=3) із цифрами $\{-1, 0, 1\}$.

Якщо канонічні системи з натуральною основою *k* мають тільки цифри $x_i \in \{0, 1, ..., k-1\}$ або тільки цифри $x_i \in \{-k+1,\ -k+2,\ ...,\ 0\}$, то вони називаються *зміщеними*. За допомогою таких систем можна подати в кінцевому вигляді або тільки додатне, або тільки від'ємне ціле число. До зазначених систем належать десяткова система із цифрами $\{0, 1, ..., 9\}$ і двійкова система із цифрами $\{0, 1\}$.

Кососиметричними канонічними системами називають системи з натуральною основою *k* і цифрами $x_i \in \{-q, -q+1, ..., 0, ..., g\}$, причому $\{q + g + 1 = k\}$ та $q \neq g$. Такі системи за своїми властивостями посідають проміжне положення між симетричними й зміщеними системами.

У записі чисел у канонічних системах числення в кожному розряді може бути використана одна з *k* різних цифр. Оскільки загальна

кількість різних комбінацій в n розрядах дорівнює k^n і числа в таких системах подаються однозначно, то загальна кількість чисел, яку можна подати за допомогою n розрядів, також дорівнює k^n.

Підставляючи у вираз (Б1-1.2) замість x_i їх можливі максимальні й мінімальні значення для різних систем, можна визначити, що діапазони подання чисел для симетричних, зміщених із додатними цифрами, зміщених із від'ємними цифрами та кососиметричних систем числення відповідно визначаються інтервалами:

$$\left[-\frac{k^{s+1}-k^{-m}}{2},\ \frac{k^{s+1}-k^{-m}}{2}\right];$$
$$\left[0,\ k^{s+1}-k^{-m}\right];$$
$$\left[-k^{s+1}-k^{-m},\ 0\right];$$
$$\left[-q\frac{k^{s+1}-k^{-m}}{2},\ q\frac{k^{s+1}-k^{-m}}{2}\right], \qquad \text{(Б1-1.3)}$$

причому будь-які два найближчі за значенням числа відрізняються на одиницю молодшого розряду, тобто на величину k^{-m}.

У ряді випадків для зручності виконання арифметичних операцій і підвищення надійності подання інформації використовують позиційні системи числення із природним порядком ваг, у яких кількість різних припустимих для кожного розряду цифр перевищує основу системи числення. Такі системи числення задовольняють вимогам скінченності й ефективності, але не задовольняють вимозі однозначності й називаються *надлишковими*.

Надлишкові системи числення з натуральною основою $k = 2r$ і цифрами $x_i \in \{-r,\ -r+1,\ ...,\ 0,\ ...,\ r,\ r+1\}$ або з основою $k = 2r+1$ і цифрами $x_i \in \{-r-1, -r, ..., 0,\ ...,\ r,\ r+1\}$ називають *квазіканонічними*. Прикладами таких систем є десяткова система числення із цифрами $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6\}$ і трійкова система числення із цифрами $\{-2, -1, 0, 1, 2\}$. Загальна кількість різних цифр у таких системах дорівнює $k+2$.

Системи числення з натуральною основою $k = 2r$ і цифрами $x_i \in \{-r, -r+1, ..., 0, ..., r-1, r\}$ або з основою $k = 2r+1$ і цифрами $x_i \in \{-r, -r+1, ..., 0, ..., r, r+1\}$ називають *модифікованими квазіканонічними надлишковими* системами числення. У таких системах кількість різних цифр лише на одиницю більше основи системи чис-

лення. До модифікованих квазіканонічних систем належать, приміром, трійкова система із цифрами $\{-1, 0, 1, 2\}$ і двійкова система із цифрами $\{-1, 0, 1\}$. Зазначені системи числення ефективно використовуються, наприклад, під час виконання арифметичних операцій.

Поряд з однорідними позиційними системами числення існують *змішані* (*неоднорідні*) позиційні системи числення. Такі системи задаються за допомогою матриць вигляду:

$$\begin{pmatrix} t_m t_{m-1} \dots t_i \dots t_1 \\ k_m k_{m-1} \dots k_i \dots k_1 \end{pmatrix} = \begin{pmatrix} t_i \\ k_i \end{pmatrix}. \qquad \text{(Б1-1.4)}$$

У першому рядку матриці зазначене число розрядів t_i, що відводяться в i-ї групі розрядів $(i = 1,...,m)$ коду числа для запису цифр по основі k_i, що зазначено у відповідному стовпці другого рядка матриці.

Змішані системи так само, як й однорідні, можуть бути з *безпосереднім* і *кодованим* поданням цифр. Якщо кожна група розрядів змішаної системи числення подана цифрами однорідної системи із природним порядком ваг, що відповідає і змішаним і однорідним системам числення (як канонічним, так і надлишковим) можна сформувати в такий спосіб.

Змішана система числення називається системою із природним порядком ваг, якщо в цілій частині запису вага першого (молодшого) розряду дорівнює одиниці, а вага будь-якого іншого дорівнює добутку основ, що відповідають кожному з розрядів, розташованих у цілій частині правіше даного розряду; у дробовій частині запису числа вага будь-якого розряду дорівнює величині, зворотній добутку основ, що відповідають даному розряду й розрядам, розташованим у дробовій частині ліворуч від нього.

З наведеного визначення витікає, що ваги розрядів змішаних систем числення визначаються по мультиплікативній ознаці. Якщо задано матрицю (Б1-1.4), то можна обчислити вагу будь-якого розряду. Кількісний еквівалент запису числа визначається за адитивним принципом, тобто для знаходження кількісного еквівалента запису числа необхідно просумувати всі кількісні еквіваленти цифр за правилами десяткової арифметики.

Якщо групи розрядів, що відповідають матриці (Б1-1.4), повторюються в записі числа періодично, то таку систему числення можна розглядати як однорідну систему з натуральною основою $K = k_1^{t1}\, k_2^{t2} \dots\, k_m^{tm}$, у якій цифри кодуються цифрами системи числення з основами $k_i (i = 1 \dots m)$.

Практичний інтерес мають змішані системи, у котрих кожен розряд канонічної десяткової системи подається кількома двійковими розрядами. Побудова ЕОМ, у яких цифри відображені сигналами, квантованими по десятьом рівням, є практично доцільним тільки за наявності відповідних технічних засобів (схем). Однак, як правило, основні характеристики відомих схем для такого подання десяткових цифр поступається характеристикам двійкових схем. Тому під час побудови десяткових ЕОМ використовують кодування двійковими цифрами десяткових цифр.

Для кодування цифр системи числення з основою k двійковими цифрами необхідно мати не менше ніж $\rceil \log_2 k \lceil$ двійкових розрядів. Тут запис $\rceil t \lceil$ означає округлення t у бік найближчого цілого числа, якщо t дробове.

Очевидно, що будь-якій k-й цифрі можна поставити у відповідність кожне з 2^k двійкових чисел, записаних за допомогою h розрядів. Відповідно, число способів двійкового кодування k цифр дорівнює числу розміщень з 2^h по k, тобто $\frac{2^h!}{(2^h-k)!}$. Число способів дуже швидко зростає зі збільшенням k, що унеможливлює аналіз ручними методами всіх способів двійкового кодування навіть для практично найбільш важливого випадку $k = 10$. У зв'язку із цим до теперішнього часу досліджені далеко не всі способи кодування десяткових цифр.

Найширше поширення в ОТ одержали *двійково-десяткові системи числення*, у яких десяткові цифри подаються як чотирьохрозрядні двійкові числа — двійкові *тетради*. Іноді використовують також подання десяткових цифр за допомогою п'яти, шести або семи розрядів. Можна побудувати ЕОМ, що працює за будь-якого двійково-десятковому кодуванні. Як показують теоретичні дослідження й досвід розроблення десяткових ЕОМ, найкращі результати отримані за використання *двійково-десяткових кодів* (ДДК), що мають властивості *одиничності, упорядкованості, парності, доповнюваності* й *виваженості.*

Двійково-десяткові коди мають властивість *одиничності*, якщо між десятковою цифрою й комбінацією двійкових цифр встановлена взаємна однозначна відповідність.

Властивість *адитивності* полягає в тому, що ДДК суми десяткових цифр може бути отриманий як ДДК доданків.

Упорядкованість ДДК складається у виконанні однієї з умов:

$$0_{(2-10)} < 1_{(2-10)} < \ldots < 9_{(2-10)},$$

$$0_{(2-10)} > 1_{(2-10)} > \ldots > 9_{(2-10)},$$

для двійкових відображень десяткових цифр 0, 1, ..., 9. Наявність упорядкованості ДДК спрощує реалізацію логічних операцій. Тут і далі через $X_{(k)}$ будемо позначати число X в k-й системі числення, наприклад, $2_{(10)}$ — число 2 у десятковій системі числення.

Властивість *парності* ДДК виявляється в тому, що всім парним десятковим цифрам мають відповідати або тільки парні, або тільки непарні двійкові коди. Аналогічно, всім непарним десятковим цифрам мають відповідати або тільки непарні, або тільки парні двійкові коди. Ця властивість спрощує виконання операцій множення, ділення й округлення.

Сутність властивості *доповнюваності* ДДК полягає в тому, що, якщо сума двох десяткових цифр дорівнює 9, то перехід від двійкового коду однієї цифри до двійкового коду іншої має здійснюватися шляхом інвертування (тобто заміни 0 на 1 і навпаки) двійкових розрядів. Це полегшує формування зворотного й доповнювального кодів, які використовуються під час операцій в розділі Б1-1.2.

Двійково-десятковий код називають *зваженим*, якщо кожному з h розрядів двійкового подання $x_h\, x_{h-1} \ldots x_1$ десяткової цифри X поставлені у відповідність ваги $a_h\, a_{h-1}, \ldots, a_1$, причому

$$X = a_h x_h + a_{h-1} x_{h-1} + \ldots + a_1 h_1. \qquad \text{(Б1-1.5)}$$

Використання зважених ДДК полегшує виконання операцій і переклад чисел з однієї системи числення в іншу.

Двійково-десяткові коди, що виходять один із одного простою перестановкою ваг a_i, утворять *кодову групу*, або просто *групу*.

У кожному конкретному випадку застосування будь-якого ДДК обумовлюється певними його перевагами порівняно з іншими ДДК.

Найбільше розповсюдження дістали ДДК, у яких для подання десяткових цифр використовують двійкові тетради. Із категорії зважених кодів до таких відносять двійково-десяткові коди з *додатними вагами «2421»* та «*8421»*, а також незважений код *«8421 + 3»*, який називають *«код із надлишком три»* (табл. Б1-1.1).

Код «2421», який одержав назву *коду Емері*, має всі перераховані вище властивості, крім властивості одиничності. Особливістю ДДК «2421» є відсутність однозначного подання десяткових цифр, так деякі десяткові цифри можуть бути подані різними двійковими кодами. Наприклад, цифра 2 подається як 1000 або як 0010. Для виконання умови одиничності за такого кодування домовилися вважати «забороненими» деякі, хоча й правильні, двійкові коди десяткових цифр.

Таблиця Б1-1.1

ДВІЙКОВО-ДЕСЯТКОВІ КОДИ

Десяткова цифра	Код «2421»	Код «8421»	Код «8421+3»
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	1011	0101	1000
6	1100	0110	1001
7	1101	0111	1010
8	1110	1000	1011
9	1111	1001	1100

Під час додавання десяткових розрядів (тетрад) у даному коді перенос у старшу тетраду формується автоматично, але неприродний порядок вагів розрядів потребує корегування результату операції.

Приклад

Завдання. Записати десяткове число в коді «2421».

Виконання завдання

$$3259_{(10)} = 0011\ 0010\ 1011\ 1111_{(2-10)};$$
$$7705_{(10)} = 1101\ 1101\ 0000\ 1011_{(2-10)}.$$

Код «8421» називають також *кодом прямого заміщення*. Цей код створюється шляхом запису десяткових цифр у двійковій позиційній системі числення із природним порядком ваг і характеризується всіма перерахованими вище властивостями, окрім властивості доповнюваності. Останнє робить код незручним під час реалізації операції алгебраїчного додавання через труднощі формування переносів з молодшої тетради в старшу. Однак код «8421» зручний для машинного перетворення чисел з десяткової системи в двійкову й навпаки.

Приклад

Завдання. Записати десяткове число в коді «8421».

Виконання завдання

$$3259_{(10)} = 0011\ 0010\ 0101\ 1001_{(2-10)};$$
$$7705_{(10)} = 0111\ 0111\ 0000\ 0101_{(2-10)}.$$

Для запису десяткових цифр у коді «із надлишком три» необхідно кожну двійкову тетраду цієї цифри в коді «8421» скласти із двійковим кодом цифри 3. Код «із надлишком три» зручний для виконання операції алгебраїчного додавання завдяки автоматичному формуванню переносів під час складання тетрад.

Приклад

Завдання. Записати десяткове число в коді «8421+3».

Виконання завдання

$$3259_{(10)} = 0110\ 0101\ 1000\ 1100_{(2-10)};$$
$$7705_{(10)} = 1010\ 1010\ 0011\ 1000_{(2-10)}.$$

У програмуванні широко застосовують *двійковокодовані* системі числення з основами 8 та 16. Цифри таких систем кодуються двійковими розрядами відповідно до табл. Б1-1.2 і Б1-1.3.

Таблиця Б1-1.2

КОДУВАННЯ ЦИФР У СИСТЕМІ ЧИСЛЕННЯ З ОСНОВОЮ 8

Цифра	0	1	2	3	4	5	6	7
Код	000	001	010	011	100	101	110	111

Приклад

Завдання. Записати двійковокодоване вісімкове число.

Виконання завдання

$$3251_{(8)} = 011\ 010\ 101\ 001_{(2-8)};$$
$$7705_{(8)} = 111\ 111\ 000\ 101_{(2-8)}.$$

Таблиця Б1-1.3

КОДУВАННЯ ЦИФР У СИСТЕМІ ЧИСЛЕННЯ З ОСНОВОЮ 16

Цифра	0	1	2	3	4	5	6	7
Код	0000	0001	0010	0011	0100	0101	0110	0111
Цифра	8	9	A	B	C	D	E	F
Код	1000	1001	1010	1011	1100	1101	1110	1111

Приклад

Завдання. Записати двійковокодоване число з основою 16.

Виконання завдання

$3A2C_{(16)} = 0011\ 1010\ 0010\ 1100_{(2-16)}$;
$0D7F_{(16)} = 0000\ 1101\ 0111\ 1111_{(2-16)}$.

У позиційних системах числення кожен розряд у записі числа має свою вагу. Такі системи числення називають також *зваженими*. Поряд із цим можливі такі системи числення, у яких кожна окрема цифра ніяк не пов'язана з якимось кількісним еквівалентом і лише певним комбінаціям цифр ставиться у відповідність деякий кількісний еквівалент. Такі системи числення називають *символічними* (*незваженими*).

Головна перевага зважених систем числення — зручність подання чисел і простота виконання арифметичних операцій. Один з недоліків таких систем полягає в неможливості виконання арифметичних операцій як порозрядних.

Порозрядною операцією називають таку, результат котрої в будь-якому розряді не залежить від результату виконання цієї операції у всіх інших розрядах.

Прикладом системи числення, у якій арифметичні операції виконуються порозрядно, є *система залишкових класів* (СЗК), яка є *незваженою* системою.

У СЗК для подання цілих позитивних чисел обирається набір взаємно простих модулів (основ) $p_i (i = 1...m)$ таким чином, щоб виконувалася умова $X_{\max} < p_1\ p_2 \ldots p_m$, де $X_{\max}$ — максимальне із чисел, що подають у СЗК. Будь-яке ціле позитивне число X подається набором залишків від його розподілу на модулі p_i, тобто у вигляді $X = (x_m, x_{m-1}, ..., x_i, ..., x_1)$, де $x_i = restX (\mathrm{mod}\ p_i)$.

Максимальна кількість чисел, яку можна подати за допомогою m модулів, дорівнює $W = p_1\ p_2 \ldots p_m$. У таблиці Б1-1.4 показані зображення деяких десяткових чисел за використання системи модулів $p_1 = 5, p_2 = 3, p_3 = 2$.

Таблиця Б1-1.4

СИМВОЛІЧНІ КОДИ

Число ($k = 10$)	СЗК	Код Грея	Число ($k = 10$)	СЗК	Код Грея
0	000	0000	8	320	1100
1	111	0001	9	401	1101
2	220	0011	10	010	1111
3	301	0010	11	121	1110
4	410	0110	12	200	1010
5	021	0111	13	311	1011
6	100	0101	14	420	1001
7	211	0100	15	001	1000

До недоліків СЗК, що утруднює її застосування в ОТ, належать: відсутність зручного способу порівняння чисел; труднощі виконання операції ділення й округлення; відсутність зручного способу визначення виходу результату операції за межі діапазону припустимих чисел. Зазначені недоліки обмежують область застосування СЗК рамками спеціалізованих ЕОМ, для яких ці недоліки не є істотними.

Наданий у табл. Б1-1.4 символічний *рефлексний код*, котрий має назву *код Грея*, має таку властивість, що двом сусіднім десятковим цифрам відповідають кодові комбінації, які відрізняються одна від одної тільки в одному двійковому розряді (сусідні комбінації). Ця властивість активно використовується під час перетворення величини кутового й лінійного переміщення в цифровий еквівалент.

Вибір системи числення для подання інформації в ЕОМ впливає на її надійність й економічність. Від системи числення, що використовується, залежить структура апаратних засобів ЕОМ, зручність і швидкість виконання арифметичних і логічних операцій. Необхідно враховувати також простоту перекладу чисел у десяткову систему числення, тому що спілкування людини з машиною будується на основі десяткової системи числення незалежно від системи, що використовується для подання чисел у самій машині.

Практика розроблення й експлуатації ЕОМ показує, що обчислювальні машини, призначені для вирішення задач, у котрих кількість обчислювальних операцій на один символ інформації, що вво-

диться і виводиться, велике, використовують двійкової системи числення. До таких задач належить широке коло науково-технічних задач, задачі оптимального планування, транспортні, тощо. Коли час вводу-виводу даних значно перевищує час оброблення інформації, застосовується десяткова система числення.

Окрім числової інформації ЕОМ обробляють й текстову (алфавітно-цифрову) інформацію, задану символами: буквами, цифрами, різними умовними позначеннями, математичними знаками, розділовими знаками, тощо. Для цього символи кодуються двійковими цифрами. У сучасних ЕОМ кожен символ найчастіше подається восьмирозрядним кодом — байтом, що дозволяє закодувати $2^8 = 256$ різних символів.

Б1-1.2. Перевід чисел з однієї системи числення в іншу

У розділі розглянуті методи переводу чисел з зміщеної канонічної системи числення з основою k_1 у зміщену канонічну систему з основою k_2.

Правило переводу цілих чисел

Спочатку вихідне число в системі з основою k_1, а потім кожний черговий залишок *цілочислено* ділять на основу k_2 за правилами системи числення з основою k_1. Кожний отриманий залишок від ділення є черговою цифрою системи з основою k_2, починаючи з молодших розрядів. Ділення продовжується до тих пір, поки черговий залишок не стане менш ніж дільник. Цей залишок є старшою цифрою числа з основою k_2.

Правило переводу дробових чисел

Спочатку вихідне число в системі з основою k_1, а потім дробова частина кожного добутку, множиться на основу k_2 за правилами системи числення з основою k_1. Ціла частина кожного добутку є черговою цифрою числа в системі з основою k_2, починаючи зі старших розрядів. Множення триває до тих пір, поки не буде одержано необхідну кількість цифр числа з основою k_2, що визначається необхідною точністю одержання результату.

Якщо число має цілу й дробову частину, то кожна частина обробляється окремо, а одержані результати складаються.

Приклад

Завдання. Перевести у двійкову систему числення число

$$A = 238{,}87_{(10)}.$$

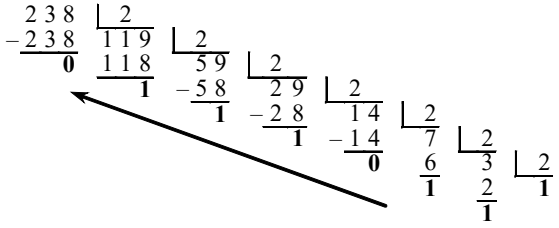
Виконання завдання

У даному випадку $k_1 > k_2$. Подамо число А у вигляді:

$$A = A_{ц} + A_{д} = 238_{(10)} + 0{,}87_{(10)}.$$

Цілу і дробову частину перекладемо у двійкову систему числення окремо.

Перевід цілого числа $A_{ц} = 238_{(10)}$ у двійкову систему числення.



Отримуємо: $A_{ц} = 11101110_{(2)}$.

Перевід дробового числа $A_{д}=0{,}87_{(10)}$ у двійкову систему числення (до 7 розрядів після коми).

$$\begin{aligned}
0{,}87 \times 2 &= \mathbf{1},74\\
0{,}74 \times 2 &= \mathbf{1},48\\
0{,}48 \times 2 &= \mathbf{0},96\\
0{,}96 \times 2 &= \mathbf{1},92\\
0{,}92 \times 2 &= \mathbf{1},84\\
0{,}84 \times 2 &= \mathbf{1},68\\
0{,}68 \times 2 &= \mathbf{1},36
\end{aligned}$$

Отримуємо: $A_{д} = 0{,}1101111_{(2)}$.

Відповідь: $A = 11101110{,}110111_{(2)}$.

Приклад

Завдання. Перевести у десяткову систему числення число

$$A = 11101110{,}110111_{(2)}.$$

Виконання завдання

У цьому випадку $k_1 < k_2$. Подамо число A у вигляді:

$$A = A_{ц} + A_{д} = 11101110_{(2)} + 0{,}1101111_{(2)}.$$

Цілу й дробову частину перекладаємо у десяткову систему числення окремо. Перевід цілого числа $A_{ц} = 11101110_{(2)}$ у десяткову систему.

```
  11101110         |1010
 –1010              10011        |1010
   10011           –1010          10
  –1010                11
    10011
   –1010
     10010
    –1010
      1000
```

Отримаємо: $A_ц = 238_{(10)}$.

Перевід дробового числа $A_д = 0{,}1101111_{(2)}$ у десяткову систему (до 3 розрядів після коми).

```
   0,1101111         0,1010110         0,1011100
      × 1010            × 1010            × 1010
    1101111           1010110           1011100
 + 1101111         + 1010110         + 1011100
 1000,1010110       110,1011100       111,0011000
```

Отримаємо: $A_д = 0{,}867_{(10)} \approx 0{,}87_{(10)}$.

Відповідь: $A = 238_{(10)} + 0{,}87_{(10)} = 238{,}87_{(10)}$.

Приклад

Завдання. Перевести із двійкової системи числення в трійкову число:

$$X = 10110110_{(2)}.$$

Виконання завдання

```
 10110110      |11
–11             111100     |11
  101          –11          10100     |11      |11
 –11              011      –11         110      10
   100           –11          100     –11
  –11               000      –11        00
     11                        10
   –11
    010
```

Відповідь: $X = 20202_{(3)}$.

Перевід чисел із симетричних і кососиметричних систем числення із цифрами, що мають різні знаки, у зміщені канонічні системи можна здійснити таким чином.

На першому етапі вихідне число з основою k_1 подають у вигляді алгебраїчної суми двох чисел. Перше додатне число складається з додатних цифр, які зберігають свої позиції в запису вихідного числа, і нулів у всіх інших розрядах. Аналогічним чином формується від'ємне число з від'ємних цифр і нулів у всіх інших розрядах.

На другому етапі, використовуючи розглянуті вище правила, здійснюється перетворення модулів одержаних чисел у зміщену систему числення з основою k_2.

На третьому етапі формується алгебраїчна сума одержаних чисел в k_2-й системі з урахуванням знаків чисел.

Приклад

Завдання. Перевести у двійкову канонічну зміщену систему числення число

$$X = \bar{1}\bar{1}25_{(10)}.$$

Виконання завдання

Подамо число X у вигляді:

$$X = \bar{1}\bar{1}\bar{2}\bar{5} = X_{\text{дод}} + X_{\text{від}} = 1020_{(10)} - 0105_{(10)}.$$

Після переводу отриманих чисел у двійкову систему числення одержимо $X_{\text{дод}} = 1111111100_{(2)}$ і $X_{\text{від}} = -1101001_{(2)}$. Результат перетворення знаходимо у вигляді

$$X = X_{\text{дод}} + X_{\text{від}} = 1111111100_{(2)} - 1101001_{(2)} = 1110010011_{(2)}.$$

Відповідь: $X = 1110010011_{(2)}$.

Перетворення чисел із надлишкової кососиметричної системи із цифрами одного знаку в канонічну зміщену систему відбувається таким чином.

Спочатку кожну цифру вихідної системи записують як число в канонічній системі за допомогою декількох розрядів, а потім складають одержані числа з урахуванням ваги кожного розряду.

Приклад

Завдання. Перевести із надлишкової системи числення з цифрами {0,1,2} в ненадлишкову двійкову канонічну систему числення з цифрами {0,1} двійкове число

$$X_{\text{зб}} = 212012_{(2)}.$$

Виконання завдання

Записуємо число $X_н = 210211_{(2)}$ у вигляді:

$$X_н = (10)(01)(00)(10)(01)(01)_{(2)}.$$

Підсумовуємо розряди з урахуванням ваги кожної двійкової цифри.

```
1   0
+   0   1
    +   1   0
        +   0   0
            +   0   1
                +   1   1
-------------------------
1   1   0   0   1   0   1
```

Відповідь: $X_к = 1100100_{(2)}$.

Алгоритми переводу чисел із систем з основами $2^3 = 8$ і $2^4 = 16$ дуже прості, чим і пояснюється застосування цих систем в ОТ.

Для переводу чисел із системи числення з основою 2^s у двійкову систему необхідно записати кожну цифру числа s-розрядним двійковим кодом.

Приклад

Завдання. Перевести у двійкову систему числення число

$$A = 762.15_{(8)}.$$

Виконання завдання

Виконаємо запис кожної двійкової цифри трирозрядним кодом, що відповідає вісімковій системі числення $2^3 = 8$:

```
7     6     2     ,     1     5
111   110   010   ,     001   101
```

Відповідь: $A = 111110010{,}001101_{(2)}$.

Для переведення двійкового числа в систему числення з основою 2^s необхідно, рухаючись від коми вліво й вправо, розбити двійкове число на групи, що містять по s розрядів, доповнюючи за необхідності нулями крайні ліву й праву групи. Потім кожну групу з s двійкових розрядів необхідно замінити цифрою системи числення з основою 2^s.

Приклад

Завдання. Перевести у шістнадцятирічну систему числення число

$$A = 11111101{,}1000001_{(2)}.$$

Виконання завдання

Розбиваємо задане число на групи, які містять по чотири розряди, що відповідає шістнадцятирічній системі числення $2^4 = 16$. Виконаємо заміну кожної групи відповідною цифрою системи числення з основою $2^4 = 16$:

1111	1101	,	1000	0010
F	*D*	,	8	2

Відповідь: $A = FD{,}82_{(16)}$.

Б1-1.3. Кодування від'ємних чисел у ЕОМ

За використання симетричних і кососиметричних систем числення проблеми кодування знака числа не виникає, тому що в цьому випадку знак числа визначається знаками цифр. Використання ж зміщених систем числення з додатними й від'ємними цифрами пов'язане з необхідністю рішення завдань кодування додатних і від'ємних чисел. Через повну аналогію методів рішення цих задач розглянемо лише способи кодування від'ємних чисел у зміщених системах числення з додатними цифрами.

В ЕОМ доцільно подавати знаки чисел за допомогою тих самих символів, що застосовуються для запису самого числа в *k*-й системі числення. Для цього використовують додатковий розряд, названий знаковим, який розташовується ліворуч від старшого розряду числа.

В ЕОМ для виконання операцій з числами, що мають знаки, використовують спеціальні коди:

— прямий код (ПК),

— обернений код (ОК),

— доповнювальний код (ДК).

Розглянемо прямий, обернений та доповнювальний коди для двійкової системи числення з цифрами {0,1}, саме яка має найбільше поширення в ОТ.

Під час запису числа вважаємо, що число має *n*-розрядну цілу частину і *k*-розрядну дробову частину. Крім цього, до числа додається ще знаковий розряд (ЗР). Відповідна *розрядна сітка* зображена на рис. Б1-1.1.

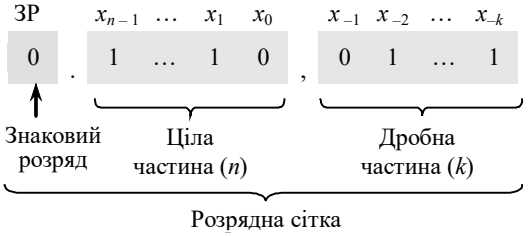


Рис. Б1-1.1. Приклад запису двійкового числа зі знаком

Старший розряд цілої частини має вагу 2^{n-1}, а молодший розряд дробної частини — вагу 2^{-k}.

Подання числа X у *прямому коді* визначається виразом.

$$[X]_{\text{ПК}} = \begin{cases} X, & \text{якщо } X \geq 0; \\ 2^n + |X|, & \text{якщо } X \leq 0. \end{cases} \qquad \text{(Б1-1.6)}$$

Під час утворення прямого коду знаковий розряд дорівнює 0, якщо число додатне, і 1, якщо число від'ємне.

Під час запису числа знаковий розряд відокремлюється від основних розрядів крапкою, а ціла частина числа від дробної — комою. У випадку, коли числа не мають цілої частини, знаковий розряд відокремлюється від основних розрядів комою.

Приклад

Завдання. Записати числа A = 10,101011; B = –10,0111010 у ПК.

Виконання завдання

$$[A]_{\text{ПК}} = 0.10,\ 101011;\ [B]_{\text{ПК}} = 1.10,\ 0111010$$

Приклад

Завдання. Записати числа A = 0,101011; B = – 0,0111010 у ПК.

Виконання завдання

$$[A]_{\text{ПК}} = 0,\ 101011;\ [B]_{\text{ПК}} = 1,\ 0111010$$

Прямий код застосовується для зберігання чисел в пам'яті комп'ютера, для операцій додавання і віднімання ПК не використовується.

Під час перетворення від'ємного числа на *обернений код*, у знаковий розряд записується 1, а значення основних розрядів інверту-

ються, тобто у кожному розряді 0 заміняється на 1, а 1 заміняється на 0. Додатне число у ОК збігається із числом у ПК, тобто основні розряди не інвертуються, у знаковий розряд записується 0.

Формула перетворення чисел у обернений код має вигляд:

$$[A]_{\text{ЗК}} = \begin{cases} A, \text{ якщо } A \geq 0; \\ 2^{n+1} - 2^{-k} - |A| = 2^{n+1} - 2^{k} + A, \text{ якщо } A \leq 0. \end{cases} \quad \text{(Б1-1.7)}$$

Приклад

Завдання. Записати числа A = 10,1011; B = – 01,11010 в ОК.

Виконання завдання

$$[A]_{\text{ок}} = 0.10{,}1011; \; [B]_{\text{ок}} = 1.10{,}00101.$$

Приклад

Завдання. Записати числа A = 101011; B = – 0111010 в ОК.

Виконання завдання

$$[A]_{\text{ок}} = 0.101011; \; [B]_{\text{ок}} = 1.1000101.$$

Приклад

Завдання. Записати числа A = 0,101011; B = – 0,0111010 в ОдоК.

Виконання завдання

$$[A]_{\text{ок}} = 0{,}101011; \; [B]_{\text{ок}} = 1{,}1000101.$$

Під час перетворення від'ємного числа на *доповнювальний код*, у знаковий розряд записується 1, а значення основних розрядів інвертуються, після чого до молодшого розряду додається 1 (з поширенням переносів між розрядами). Додатне число в ДК збігається з числами у ПК і ОК, тобто в знаковий розряд записується 0, а основні розряди не змінюються.

Формула перетворення чисел у доповнювальний код має вигляд:

$$[A]_{\text{ДК}} = \begin{cases} A, \text{ якщо } A \geq 0; \\ 2^{n+1} - |A| = 2^{n+1} + A, \text{ якщо } A < 0. \end{cases} \quad \text{(Б1-1.8)}$$

Приклад

Завдання. Записати числа A = – 011,100; B = 010,010 у ПК, ОК і ДК.

Виконання завдання

$$
\begin{array}{lll}
[A]_{\text{ПК}} = 1\ .\ 011,100 & [B]_{\text{ПК}} = 0\ .\ 010,010 \\
[A]_{\text{ОК}} = 1\ .\ 100,011 & [B]_{\text{ОК}} = 0\ .\ 010,010 \\
\qquad\quad + \qquad\quad 1 & [B]_{\text{ДК}} = 0\ .\ 010,010 \\
[A]_{\text{ДК}} = \overline{1\ .\ 100,100} &
\end{array}
$$

Приклад

Завдання. Записати числа $A = -0{,}011100$; $B = 0{,}010010$ у ПК, ОК і ДК.

Виконання завдання

$$
\begin{array}{lll}
[A]_{\text{ПК}} = 1\ ,\ 011100 & [B]_{\text{ПК}} = 0\ ,\ 010010 \\
[A]_{\text{ОК}} = 1\ ,\ 100011 & [B]_{\text{ОК}} = 0\ ,\ 010010 \\
\qquad\quad + \qquad\quad 1 & [B]_{\text{ДК}} = 0\ ,\ 010010 \\
[A]_{\text{ДК}} = \overline{1\ ,\ 100100} &
\end{array}
$$

B1-1.4. Форми подання чисел у ЕОМ

В ОТ переважно використовують дві форми подання чисел:

— подання чисел з фіксованою комою;

— подання чисел із плаваючою комою.

Будь-яке число X у позиційній системі числення з основою k можна записати як

$$X = k^P M,$$

де M — *мантиса* числа X, а P — *порядок* числа X.

Якщо порядок фіксований для всіх чисел, то говорять, що число подане у *формі з фіксованою комою*. В цьому випадку числа задаються тільки одним об'єктом — мантисою.

За подання чисел у формі з фіксованою комою положення коми залишається незмінним для всіх чисел, якими оперує машина, тобто спеціальні розряди для коми не виділяються. З метою спрощення обчислення масштабних коефіцієнтів кому звичайно фіксують перед старшим розрядом мантиси або після молодшого розряду.

За фіксації коми перед старшим розрядом мантиси ЕОМ оперує числами, які менше одиниці. Якщо число розрядів для запису мантис становить n, то мінімальне (за абсолютною величиною) число, що може бути подане в машині, дорівнює k^{-n}, а максимальне дорівнює $(1 - k^{-n})$.

У другому випадку, за фіксації коми після молодшого розряду, в машині виконуються операції над цілими числами, абсолютні величини яких перебувають у межах від 0 до ($k^{-n}-1$).

Розрядна сітка для подання чисел у ЕОМ з фіксованою комою складається із двох частин: один розряд для подання знака, інші розряди для подання мантиси.

Якщо кожне число задається парою P і M, то таке подання називають формою *із плаваючої комою*.

За подання чисел у формі із плаваючою комою порядок P може бути додатним або від'ємним цілим числом. Мантиса ж у більшості випадків є додатним або від'ємним правильним дробом, причому

$$k^{-1} \le M \le 1. \qquad \text{(Б1-1.9)}$$

Це означає, що старший розряд модуля мантиси завжди дорівнює 1 (мантиса нормалізована). Якщо в ЕОМ для запису порядку використається m розрядів, а для запису мантиси — n розрядів, то в цій машині може бути подане таке максимальне (за абсолютною величиною) число:

$$k^{k^m-1}(1-k^{-n}) = k^{k^m-1} - k^{k^m-n-1}.$$

У свою чергу, мінімальне за абсолютною величиною число, що відрізняється від нуля, у такій машині дорівнює

$$k^{-k^m+1}k^{-1} = k^{-k^m}.$$

Крім зазначених вище розрядів, для подання порядку й мантиси, у розрядній сітці ЕОМ із плаваючою комою є також розряди для подання знаків порядку й мантиси (рис. Б1-1.2).

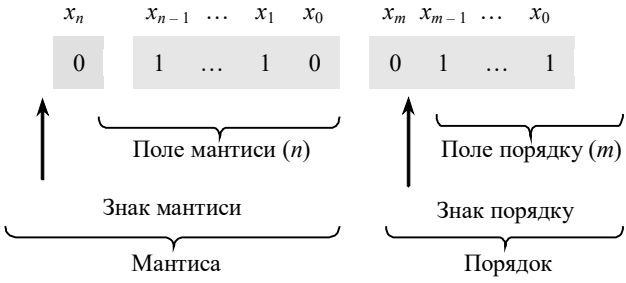


Рис. Б1-1.2. Приклад запису числа у формі з плаваючою комою

Приклад

Завдання. Записати у формі із плаваючою комою ($m = n = 4$) двійкові доповнювальні коди чисел:

$$X = 9_{(10)},\ Y = -9_{(10)},\ Z = -9/256_{(10)}.$$

Виконання завдання

$P_{(X)} = 0\ ,\ 0\,1\,0\,0$ $\quad$ $M_{(X)} = 0\ ,\ 1\,0\,0\,1$
$P_{(Y)} = 0\ ,\ 0\,1\,0\,0$ $\quad$ $M_{(Y)} = 1\ ,\ 0\,1\,1\,1$
$P_{(Z)} = 1\ ,\ 1\,1\,0\,0$ $\quad$ $M_{(Z)} = 1\ ,\ 0\,1\,1\,1$

У ЕОМ, де використовується подання чисел у формі із плаваючою комою, арифметичні операції виконуються як над мантисами чисел, так і над їхніми порядками. Часто також необхідно виконувати операцію нормалізації чисел, сутність якої полягає у виконанні умови $k^{-1} \le M \le 1$. Тому машини із плаваючою комою є більш складним і менш швидкодіючим, ніж машини з фіксованою комою. З іншого боку, у машинах з фіксованою комою через масштабування виникають труднощі під час програмування.

Для універсальних комп'ютерів розроблений стандарт ANSI/IEEE 754-1985 на подання чисел із плаваючою комою. У цьому стандарті визначені два основні базові формати: короткий та довгий. У кожному форматі використовують схований старший розряд мантиси з вагою 2^{-1}, який завжди дорівнює 1 для додатних чисел.

Короткий формат (рис. Б1-1.3, *а*) має 32 розряди, з яких 8 призначено для подання порядку, 23 розряди — для подання мантис й один розряд — для знака мантиси. Спеціального розряду для знаку порядку немає. Вводиться «зміщений порядок» $P_{зм} = P - 128$. Таким чином, від'ємні порядки P будуть подані «*зміщеними порядками*» $P_{зм}$, меншими 128, а додатні порядки — більшими 128. Введення «зміщеного порядку» дозволяє звести операції над порядками до арифметичної дії над цілими додатними числами.

Отже, максимальний додатний порядок числа в короткому форматі дорівнює

$$11111111_{(2)} - 10000000_{(2)} = 01111111_{(2)} = 127_{(10)},$$

а максимальний від'ємний порядок

$$00000000_{(2)} - 1000000_{(2)} = -128_{(10)}.$$

ЗМ	2^7	2^6	…	2^0	2^{-2}	2^{-3}	…	2^{-24}
Знак мантиси	Зміщений порядок (8 розрядів)				Мантиса (23 розряди)			

а

ЗМ	2^{10}	2^9	…	2^0	2^{-2}	2^{-3}	…	2^{-53}
Знак мантиси	Зміщений порядок (11 розрядів)				Мантиса (52 розряди)			

б

Рис. Б1-1.3. Формати чисел із плаваючою комою:
а — короткий формат; *б* — довгий формат

Довгий формат (рис. Б1-1.3, *б*), що використовується для обчислень із підвищеною точністю, має 64 розряди. Для порядку виділяється 11 розрядів, а для мантиси — 52 розряди.

Крім розглянутих вище найбільш поширених форм подання чисел у спеціалізованих ЕОМ можуть використатися також форми, одержані шляхом певного функціонального перетворення чисел. Прикладом такої форми є логарифмічна, коли числа представляються їхніми логарифмами по деякій основі. Це дає можливість замінити операції множення й ділення чисел операціями додавання й вирахування їхніх логарифмів.

Б1-1.5. Машинні алгоритми перетворення чисел

В ЕОМ оброблення даних, як правило, виконується у двійковій системі числення, а введення та виведення інформації – у двійково-десятковій системі числення, наприклад, в коді з вагами розрядів «8421». Тому виникає необхідність перетворення чисел з двійково-десяткової системи числення на двійкову і навпаки. Таке перетворення може бути виконано множенням та діленням чисел на 2 шляхом зсуву і корекції кодів.

Перехід від двійково-десяткової системи числення до двійкової відбувається шляхом ділення, а зворотний перехід — шляхом множення чисел на 2. У двійковій системі числення ці операції зводяться відповідно до звичайного зсуву на один розряд вправо або вліво. В двійково-десятковій системі, окрім зсуву, може бути необхідна корекція.

Розглянемо алгоритм перетворення *n*-розрядних двійково-десяткових чисел в коді «8421» на двійкову систему числення з природнім порядком вагів розрядів.

1. Коди всіх N тетрад і двійкове число зсуваються праворуч з переходом двійкових розрядів із кожної старшої тетради в сусідню, а із наймолодшої тетради в розряди двійкового числа.

2. Якщо із старшої тетради в молодшу перейшов нуль, то корекція не потрібна. Під час переходу в молодшу тетраду одиниці в цій тетраді виконується корекція, яка полягає у відніманні з коду тетради трійки.

3. Цифри двійкової системи формуються у процесі зсуву на виході молодшої тетради, починаючи із молодших двійкових розрядів. Перетворення закінчується, коли у всіх тетрадах двійково-десяткового числа будуть отримані нулі.

Значення корекції (–3) пояснюється так. Одиниця в тетраді двійково-десяткового числа дорівнює 10 одиницям сусідньої молодшої тетради. Коли під час зсуву вправо (діленні на 2) із старшої тетради в молодшу тетраду переходить одиниця, то вона має перенести половину ваги свого десяткового розряду, що складає 5 одиниць для молодшої тетради. Фактично одиниця потрапляє в двійковий розряд молодшої тетради, який має вагу 8. Звідси корекція дорівнює (–3).

Віднімання замінюють додаванням доповнювального коду числа три (1101) без розповсюдження переносу в старшу тетраду.

Приклад

Завдання. Перевести в двійкову систему числення число

$$A = 125_{10} = 0001\ 0010\ 0101_{2-10}.$$

Виконання завдання

Перетворення заданого числа на двійкову систему числення проілюстроване у табл. Б1-1.5.

Розглянемо алгоритм перетворення двійкових чисел на двійково-десяткові числа в коді «8421».

1. Коди всіх N тетрад зсуваються ліворуч на один розряд з переходом двійкових розрядів із кожної молодшої тетради в сусідню старшу тетраду. Розряди двійкового числа переходять у наймолодшу тетраду.

2. Якщо із тетради в тетраду не переходить одиниця або код у тетраді після зсуву (після множення на 2) менший за 10, то корекція не потрібна. Коли із тетради у сусідню старшу переходить одиниця або код в тетраді стає більше за 9, то потрібна корекція, яка полягає у додаванні до коду тетради числа 6 з розповсюдженням переносу.

Таблиця Б1-1.5

ПЕРЕТВОРЕННЯ ЧИСЛА НА ДВІЙКОВУ СИСТЕМУ ЧИСЛЕННЯ

Двійково-десяткове число			Двійкове число	Мікрооперація
0001	0010	0101		ПС
0000	1001 +1101 0110	0010	1	Зсув → Корекція (–3)
0000	0011	0001	01	Зсув →
0000	0001	1000 +1101 0101	101	Зсув → Корекція (–3)
0000	0000	1010 +1101 0111	1101	Зсув → Корекція (–3)
0000	0000	0011	11101	Зсув →
0000	0000	0001	111101	Зсув →
0000	0000	0000	1111101	Зсув →

Відповідь: $A = 1111101_2$.

3. Цифри двійково-десяткової системи формуються у процесі зсуву в тетрадах, починаючи із старших розрядів. Перетворення закінчується, коли всі двійкові розряди (включаючи нулі) переходять у двійково-десяткові тетради.

4. Необхідність корекції (+6) в тетрадах пояснюється так. Одиниця, що залишає тетраду, має забрати з неї 10 одиниць, а фактично забирає 16 (подвійну вагу старшого двійкового розряду тетради). Тому для компенсації до тетради додається число 6.

Приклад

Завдання. Перевести у двійково-десяткове число в коді «8421» двійкове число

$$A = 1111101_2.$$

Виконання завдання

Перетворення заданого числа на двійково-десяткову систему числення проілюстроване у табл. Б1-1.6.

Таблиця Б1-1.6

ПЕРЕТВОРЕННЯ ДВІЙКОВОГО ЧИСЛА НА ДВІЙКОВО-ДЕСЯТКОВУ СИСТЕМУ ЧИСЛЕННЯ

Двійково-десяткове число			Двійкове число	Мікрооперація
			1111101	ПС
0000	0000	0001	111101	Зсув ←
0000	0000	0011	11101	Зсув ←
0000	0000	0111	1101	Зсув ←
0000	0000 0001	1111 +0110 0101	101	Зсув ← Корекція (+6)
0000	0010 0011	1011 +0110 0001	01	Зсув ← Корекція (+6)
0000	0110	0010	1	Зсув ←
0000 0001	1100 +0110 0010	0101		Зсув ← Корекція (+6)

Відповідь: $A = 0001\ 0010\ 0101_{2-10} = 125_{10}$.

Б1-2. Додавання чисел

Б1-2.1. Операційні схеми та мікроалгоритми

Перетворення інформації в арифметико-логічних пристроях комп'ютерів провадиться шляхом послідовного виконання мікрооперацій над машинними словами (кодами чисел, символами та іншими об'єктами).

Під *мікрооперацією* розуміють елементарну дію, в результаті виконання якої можуть змінюватися значення машинних слів.

Машинне слово можна задати перерахуванням розрядів чи за допомогою ідентифікаторів. Наприклад, 01011001 — машинне слово, задане переліком усіх 8-ми розрядів. Ідентифікатори можуть подаватися рядком символів (букв та цифр), починаючи з букви. Доцільно як ідентифікатори використовувати позначення вузлів, на яких виконуються мікрооперації, наприклад: *RG1*, *CT*. Для визначення довжини слів та впорядкування номерів розрядів використовують додаткові дані у дужках, наприклад *RG*1(0…31), *CT*(7…0).

Алгебраїчні перетворення даних у цифрових пристроях виконуються за допомогою таких мікрооперацій як *пересилання*, *підсумовування*, *зсув*, *підрахування* (*декремент, інкремент*), *інвертування*.

Для позначення мікрооперації будемо використовувати оператор присвоювання (: =).

Пересилання слів здійснюється, як правило, між двома регістрами. Результатом пересилання є запис слова в регістр, що є приймачем слова. Джерелом інформації, окрім регістрів, можуть бути зовнішні входи пристрою, на які надходять дані, наприклад, з пам'яті, загальної шини системи, тощо.

Приклади операційних схем для виконання мікрооперацій пересилання *RG*1: = *DATA*, *RG*1: = *RG*2, *RG*1 [31…24]: = *RG*2 [7…0] відповідно показані на рис. Б1-2.1.

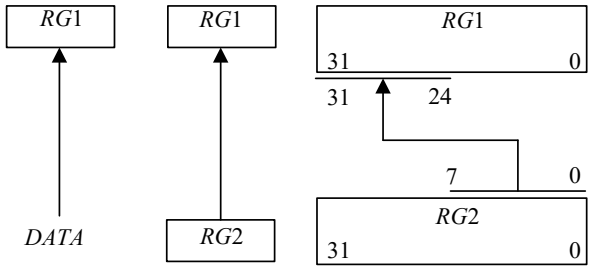


Рис. Б1-2.1. Операційні схеми для виконання мікрооперацій пересилання даних

Для підсумовування слів у схемі використовують суматор. Приклади операційних схем, що відповідають мікрооперaціям *RG*1: = *RG*1 + *RG*2 і *RG*1: = *RG*2 + *RG*3 + *CI* (де *CI* — перенос у молодший розряд суматора) відповідно показані на рис. Б1-2.2, *а*, *б*. У схемах рис. Б1-2.2, *а* і Б1-2.2, *б* використовуються комбінаційні суматори, а в схемі рис. Б1-2.2, *в* — накопичувальний суматор, який є композицією суматора і регістра.

Мікрооперації інкремента (*CT* : = *CT*+1) та декремента (*CT* : = = *CT* – 1) виконуються на лічильнику.

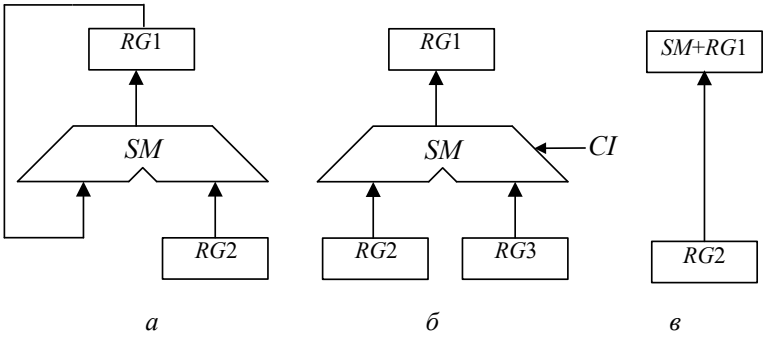


Рис. Б1-2.2. Операційні схеми для виконання мікрооперацій підсумовування чисел:

а, *б* — із застосуванням комбінаційних суматорів; *в* — з використанням накопичувального суматора

Мікрооперації зсуву слів можуть бути виконані на регістрі зсуву праворуч або ліворуч. Для означення напрямку зсуву використовують оператори зсуву *r* (*right*) та *l* (*left*). За допомогою складеного слова визначають значення розряду, який заповнюється внаслідок зсуву. Приклад операційної схеми, що відповідає мікрооперації зсуву *RG*1: = 0.*r*[*RG*1], зображений на рис Б1-2.3, *а*. Для реалізації зсуву, що відповідає операційній схемі на рис. Б1-2.3, *б*, необхідно виконати дві мікрооперайії в одному такті *RG*2: = *l*[*RG*2].0 та *RG*1: = = *l*[*RG*1].*RG*2(7).

Інвертування розрядів двійкових чисел можна забезпечити інвертуванням розрядів регістру, що зберігає це число, або використанням лінійки логічних елементів під час пересилання числа, наприклад, елементів ВИКЛЮЧНЕ АБО (рис. Б1-2.4).

Послідовність мікрооперацій, що забезпечує задане перетворення інформації, називається мікроалгоритмом.

Для опису мікроалгоритмів використовують *графічні схеми мікроалгоритмів*, які можуть бути описані мовами ГСА і ЛСА (див. розділ А1-3.2).

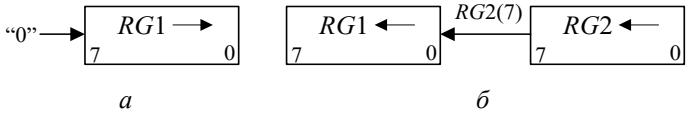


Рис. Б1-2.3. Операційні схеми
для виконання мікрооперацій зсуву

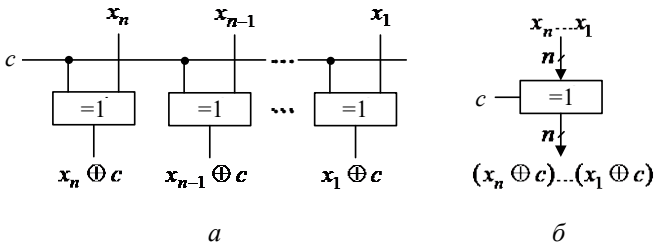


Рис. Б1-2.4. Інвертування розрядів x двійкового числа:
а — логічна схема; *б* — умовне позначення (*с* — сигнал управління інвертуванням)

Мікроалгоритми можуть складатися як у змістовній, так і в закодованій формі. Для складання змістовного мікроалгоритму мікрооперації записують у змістовній формі, наприклад, через оператори присвоювання або їх ідентифікаторів (рис. Б1-2.5). Для розроблення такого мікроалгоритму достатньо скласти операційну схему пристрою, який виконує задані мікрооперації.

Для складання закодованого мікроалгоритму необхідна докладніша схема (наприклад, функціональна), яка пояснює спосіб керування кожною мікрооперацією, включаючи означення керуючих сигналів. Змістовні мікрооперації у закодованому мікроалгоритмі замінюються на сукупність керуючих сигналів, які забезпечують виконання мікрооперацій.

Б1-2.1. Додавання чисел із знаками у машинних кодах

Операції алгебраїчного підсумовування і віднімання неможливо виконувати в прямому коді із використанням звичайного суматора, оскільки знакові розряди й основні розряди мають оброблятися порізному. Окрім того, операція віднімання повинна здійснюватися не на суматорі, а на спеціальній схемі.

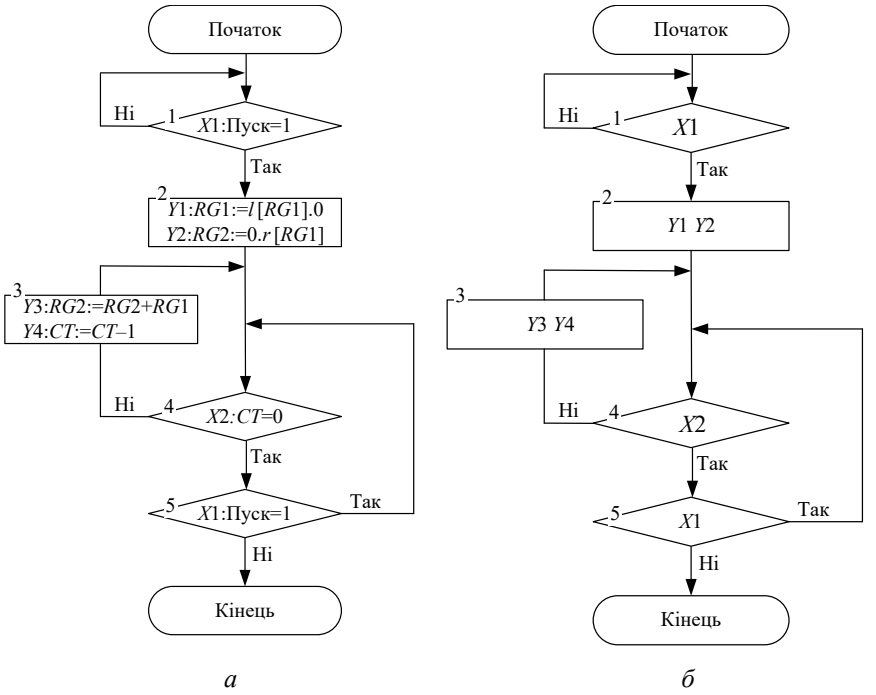


Рис. Б1-2.5. Приклад запису мікроалгоритму:

а — з розширеним змістовним описом мікрооперацій; *б* — із скороченим описом мікрооперацій у вигляді ідентифікаторів

З використанням зворотних та доповнювальних кодів операції додавання і віднімання можно виконувати за допомогою тільки суматорів, на яких оброблюються як основні, так й знакові розряди. В цьому випадку операція віднімання замінюється операцією додавання з числом, що має протилежний знак. Наприклад, операція $S = A - B$ виконується як $S = A + (-B)$.

Для виконання операції віднімання чисел у зворотних та доповнювальних кодах використовують багаторозрядні суматори, розглянуті у розділі А1-4.5. Спрощена схема багаторозрядного суматора зображена на рис. А1-4.14.

Під час додавання чисел із однаковими знаками може виникнути переповнення розрядної сітки, що приводить до втрати знака числа.

Ознакою переповнення є розбіжність значень вхідного переносу P_{in} у знаковий розряд і вихідного переносу P_{out} із знакового розряду. Отже, ознаку переповнення можна сформувати перемикальною функцією $\mathrm{OVR} = P_{in} \oplus P_{out}$.

Інший підхід для виявлення переносу складається у використанні другого допоміжного знакового розряду ліворуч від основного (першого) знакового розряду. Таке подання чисел називають *модифікованим машинним кодом*. Старший знаковий розряд при цьому завжди зберігає знак результату.

Формули кодування для модифікованих кодів мають вигляд:

$$[A]_{\text{ОК}} = \begin{cases} A, \text{ якщо } A \geq 0; \\ 2^{n+2} - 2^{-k} + A, \text{ якщо } A \leq 0. \end{cases} \quad \text{(Б1-2.1)}$$

$$[A]_{\text{ДК}} = \begin{cases} A, \text{ якщо } A \geq 0; \\ 2^{n+2} + A, \text{ якщо } A < 0. \end{cases} \quad \text{(Б1-2.2)}$$

Б1-2.3. Додавання і віднімання чисел в обернених кодах

У ОК операція віднімання замінюється операцією додавання, при цьому знаковий розряд і основні розряди обробляються як єдине ціле. Правильний знак суми утворюється автоматично в процесі підсумовування цифр знакових і основних розрядів з урахуванням переносів. Характерною рисою ОК є циклічний перенос зі знакового розряду в молодший розряд суми, завдяки якому здійснюється корекція результату.

Розглянемо чотири можливих випадки додавання чисел у модифікованих кодах.

Випадок 1. $A > 0$, $B > 0$, $A + B > 0$.

Під час підсумовування кодів відповідно до функції кодування (Б1-2.1) ми повинні одержати $[A]_{\text{ОК}} + [B]_{\text{ОК}} = [A + B]_{\text{ОК}} = A + B$, тому що доданки додатні, тобто $[A]_{\text{ОК}} = A$, $[B]_{\text{ОК}} = B$ і $[A]_{\text{ОК}} + [B]_{\text{ОК}} = A + B$.

У цьому випадку шуканий результат збігається з отриманим. Отже, корекція не потрібна. Помітимо, що під час підсумовування нульових знакових розрядів перенос не формується, тобто корекція за циклічним ланцюгом справді буде відсутня.

Випадок 2. $A > 0$, $B < 0$, $|A| > |B|$, $A + B > 0$.

Сума додатна, отже, результат повинний мати вигляд

$$[A]_{\text{ОК}} + [B]_{\text{ОК}} = [A + B]_{\text{ОК}} = A + B.$$

У підсумовуванні беруть участь коди $[A]_{\text{ОК}} = A$ і $[B]_{\text{ОК}} = 2^{n+2} - 2^{k} + B$. Сума буде мати вигляд

$$[A]_{\text{ОК}} + [B]_{\text{ОК}} = \text{A} + 2^{n+2} - 2^k + B = 2^{n+2} - 2^k + (A + B).$$

У результаті операції додавання виникла помилка $(2^{n+2} - 2^k)$. Тут 2^{n+2} — перенос зі знакового розряду за межі розрядної сітки, тобто на цю величину корекція не потрібна. Таким чином, результат отриманий з недоліком на величину 2^k. Корекція на $+2^k$ здійснюється додаванням у молодший розряд результату переносу за циклічним ланцюгом, що завжди виникає за зазначених значеннях операндів.

Приклад

Завдання. Задано A = 0,10101; B = – 0,01001. Знайти суму C.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ОК}} = 00,10101 \\ +\underline{[B]_{\text{ОК}} = 11,10110} \\ 00,01011 \\ + \qquad\qquad 1 \text{ (перенос)} \\ \hline [C]_{\text{ОК}} = 00,01100 \end{array}$$

Випадок 3. $A > 0$, $B < 0$, $|A| < |B|$, $A + B < 0$.

Сума від'ємна, виходить, результат відповідно до виразу (Б1-2.1) повинен мати вигляд

$$[A]_{\text{ОК}} + [B]_{\text{ОК}} = [A + B]_{\text{ОК}} = 2^{n+2} - 2^k + (A + B).$$

Сумуються коди $[A]_{\text{ОК}}$=A і $[B]_{\text{ОК}} = 2^{n+2} - 2^k + B$. У результаті одержимо

$$[A]_{\text{ОК}} + [\text{B}]_{\text{ОК}} = \text{A} + 2^{n+2} - 2^k + B = 2^{n+2} - 2^k + (A + B).$$

Шуканий результат збігається з отриманим, тобто корекція не потрібна.

Приклад

Завдання. Задано A = 0.01001; B = – 0.10101. Знайти суму C.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ОК}} = 00,01001 \\ +\underline{[B]_{\text{ОК}} = 11,01010} \\ [C]_{\text{ОК}} = 11,10011 \\ [C]_{\text{ПК}} = 11,01100 \end{array}$$

Випадок 4. $A < 0, B < 0, A + B < 0$.

Сума від'ємна, відповідно до цього результат повинен мати вигляд $[A]_{\text{ОК}} + [\text{B}]_{\text{ОК}} = [A + B]_{\text{ОК}} = 2^{n+2} - 2^k + (A + B)$.

Сумуються коди $[A]_{\text{ОК}} = 2^{n+2} - 2^k + A$ і $[\text{B}]_{\text{ОК}} = 2^{n+2} - 2^k + B$).

В результаті на суматорі одержимо

$$[A]_{\text{ОК}} + [\text{B}]_{\text{ОК}} = 2^{n+2} - 2^k + A + 2^{n+2} - 2^k + B =$$
$$= 2^{n+2} - 2^k + (A + B) + 2^{n+2} - 2^k.$$

Потрібна корекція результату, що здійснюється аналогічно другому випадкові.

Приклад

Завдання. Задано обернені коди чисел $A = -0.10101$ і $B = -0.01001$. Знайти обернений код суми C.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ОК}} = 11,01010 \\ +\underline{[B]_{\text{ОК}} = 11,10110} \\ 11,00000 \\ + \qquad\quad 1 \text{ (перенос)} \\ \hline [C]_{\text{ОК}} = 11,00001 \end{array}$$

Під час додавання чисел з однаковими знаками може виникнути переповнення розрядної сітки. Ознакою переповнення за використання модифікованих кодів є різні цифри в знакових розрядах. Функцію переповнення можна записати у вигляді $OVR = 3\text{P}_2 \oplus 3\text{P}_1$. Старший знаковий розряд завжди зберігає правильний знак результату.

Приклад

Завдання. Задано $A = 0{,}10101$ і $B = 0{,}01110$. Знайти суму C.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ОК}} = 00,10101 \\ +\underline{[B]_{\text{ОК}} = 00,01110} \\ [C]_{\text{ОК}} = 01,00011 \end{array} \text{ — додатне переповнення}$$

Приклад

Завдання. Задано $A = -0.10101$; $B = -0.01110$. Знайти суму C.

Виконання завдання

$$
\begin{array}{rl}
 & [A]_{\text{ОК}} = 11,01010 \\
+ & [B]_{\text{ОК}} = 11,10001 \\
\hline
 & 10,11011 \\
 & + \qquad\qquad 1 \text{ (перенос)} \\
\hline
 & [C]_{\text{ОК}} = 10,11100 \text{ — від'ємне переповнення}
\end{array}
$$

Функціональна схема пристрою, що реалізує операцію додавання і віднімання в обернених кодах наведена на рис. Б1-2.6.

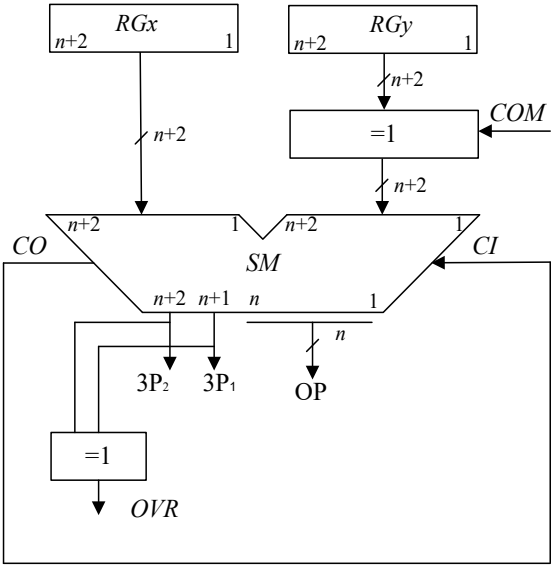


Рис. В1-2.6. Схема виконання операцій додавання і віднімання в обернених кодах

Операнди надходять на вхід суматора *SM* з регістрів *RGx* і *RGy*. Операція віднімання виконується шляхом додавання зменшуваного до від'ємника, який інвертується за допомогою елемента ВИКЛЮЧНЕ АБО. Для цього на вхід *COM* подається одиничний сигнал. Під час додавання на цей вхід потрібно подати 0. Суматор формує основні розряди суми (ОР) і два знакових розряди $ЗР_2$ і $ЗР_1$.

Корекція суми здійснюється за допомогою ланцюга циклічного переносу зі знакового розряду в молодший розряд суматора $CO \to CI$. Цей ланцюг завжди замкнутий, тому що перенос виникає тільки тоді, коли потрібно реалізувати корекцію суми. Щоб знайти переповнення розрядної сітки використовується ознака переповнення OVR ($OVR = 3P_2 \oplus 3P_1$). За значення $OVR = 0$, переповнення розрядної сітки відсутнє, у випадку, коли $OVR = 1$ наявне переповнення розрядної сітки.

Б1-2.4. Додавання і віднімання чисел у доповнювальних кодах

У ДК операція віднімання, як і в ОК, замінюється операцією додавання зменшуваного до від'ємного від'ємника. За підсумовування операндів у ДК автоматично отримуємо суму в ДК з урахуванням знаку. За застосування модифікованого коду корекції робити не треба при будь-якому сполученні знаків доданків. Факт переповнення розрядної сітки можна визначити за розбіжністю значень знакових розрядів. Старший знаковий розряд завжди зберігає знак результату.

Приклад

Завдання. Задано $A = 0{,}10101$; $B = 0{,}01001$. Знайти суму C в ДК.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ДК}} = 00,10101 \\ +\ [B]_{\text{ДК}} = 00,01001 \\ \hline [C]_{\text{ДК}} = 00,11110. \end{array}$$

Приклад

Завдання. Задано $A = 0.10101$; $B = -0.01001$. Знайти суму C в ДК.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ДК}} = 00,10101 \\ +\ [B]_{\text{ДК}} = 11,10111 \\ \hline [C]_{\text{ДК}} = 00,01100. \end{array}$$

Приклад

Завдання. Для $A = 0.01001$ і $B = -0.10101$ знайти суму C в ДК.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ДК}} = 00,01001 \\ +\ [B]_{\text{ДК}} = 11,01011 \\ \hline [C]_{\text{ДК}} = 11,10100. \end{array}$$

Приклад

Завдання. Задано $A = -0.10101$; $B = -0.01001$. Знайти C в ДК.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ДК}} = 11,01011 \\ +\ [B]_{\text{ДК}} = 11,10111 \\ \hline [C]_{\text{ДК}} = 11,00010. \end{array}$$

Приклад

Завдання. Задано $A = 0.10101$; $B = 0.01110$. Знайти суму C в ДК. Визначити знак результату.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ДК}} = 00,10101 \\ +\ [B]_{\text{ДК}} = 00,01110 \\ \hline [C]_{\text{ДК}} = 01,00011 \end{array}$$ — додатне переповнення.

У цьому випадку наявне додатне переповнення розрядної сітки та за застосування модифікованого коду старший розряд зберігає знак результату. Отже, отримане в результаті обчислень число є додатним.

Приклад

Завдання. Задано $A = -0.10101$ і $B = -0.01110$. Знайти суму C в ДК. Визначити знак результату.

Виконання завдання

$$\begin{array}{r} [A]_{\text{ДК}} = 11,01011 \\ +\ [B]_{\text{ДК}} = 11,10010 \\ \hline [C]_{\text{ДК}} = 10,11101 \end{array}$$ — від'ємне переповнення.

У цьому випадку наявне від'ємне переповнення розрядної сітки та за застосування модифікованого коду старший розряд зберігає знак результату. Отже, отримане в результаті обчислень число є від'ємним.

Схема, що реалізує операцію додавання і віднімання в ДК, аналогічна схемі для зворотних кодів (рис. В1-2.6), але в цьому випадку відсутній циклічний ланцюг корекції результату $CO \rightarrow CI$. За віднімання разом з одиничним сигналом на вхід *COM* подається одиниця на вхідний перенос *CI* суматора. Це необхідно для зміни знака від'ємника, яке здійснюється інвертуванням усіх розрядів від'ємника і додаванням 1 до його молодшого розряду. Під час додавання на входи *COM* і *CI* потрібно подати 0.

Б1-2.5. Зсуви машинних кодів

Існують два різновиди машинних зсувів:

— логічний зсув;

— арифметичний зсув.

Логічний зсув — це зміщення розрядів машинного слова у просторі із втратою розрядів, що виходять за межі розрядної сітки. Розряди, що звільняються, заповнюються нулями. Схема логічного зсуву чисел зображена на рис. Б1-2.7.

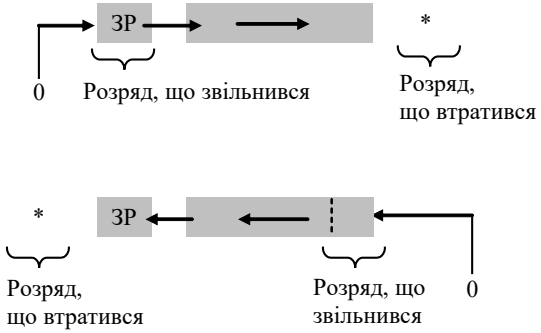


Рис. Б1-2.7. Операційна схема логічного зсуву чисел

Приклад

Завдання. Виконати логічний зсув двійкового числа вліво і вправо на один розряд.

$$A_{(2)} = 0.0101{,}1101.$$

Виконання завдання

$$\begin{array}{ll} A & |1.0101,1101| \\ A\rightarrow & |0.1010,1110|* \end{array}$$

$$\begin{array}{ll} A & |1.0101,1101| \\ \leftarrow A & *|0.1011\ \ 1010| \end{array}$$

Арифметичний зсув виконується з урахуванням знакового розряду. Правила зсуву чисел, поданих у ПК, ОК та ДК, відрізняються. Арифметичний зсув ліворуч означає множення числа на 2 (тобто на основу системи числення), а зсув праворуч — ділення числа на 2.

Арифметичний зсув чисел, поданих у ПК

При цьому типі зсуву знаковий розряд не зсувається. Основні розряди зсуваються ліворуч або праворуч із заповненням нулями розрядів, що звільнилися під час зсуву. Розряди, що вийшли за межі розрядної сітки втрачаються. За арифметичного зсуву вліво можлива втрата значимості числа. За зсуву праворуч виникає похибка. Схема арифметичного зсуву чисел, поданих у ПК, зображена на рис. Б1-2.8.

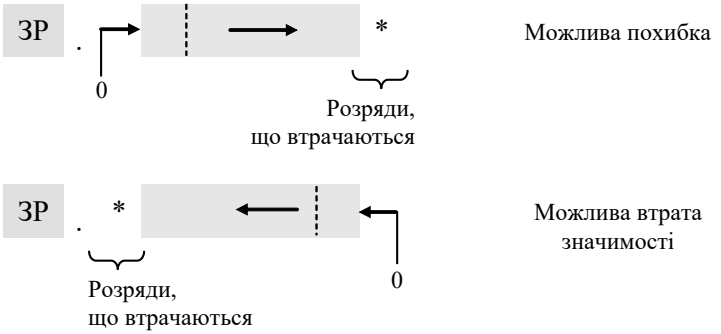


Рис. Б1-2.8. Операційна схема арифметичного зсуву чисел у ПК

Приклад

Завдання. Виконати арифметичний зсув вліво і вправо на один розряд двійкового числа, поданого у ПК.

$$A_{(2)} = 1.\ 10101;$$
$$B_{(2)} = 0.\ 11011.$$

Виконання завдання

A	1 . | 10101 |	
$A\rightarrow$	1 . | 01010 |*	*Похибка*
$\leftarrow A$	1 .* | 01010 |	*Втрата значимості*
B	0 . | 11011 |	
$B\rightarrow$	0 . | 01101 |*	*Похибка*
$\leftarrow B$	0 .* | 10110 |	*Втрата значимості*

Арифметичний зсув ліворуч чисел, поданих в ОК

Правило. Якщо під час зсуву від'ємного числа ліворуч за розрядну сітку виходить одиниця із знакового розряду, то необхідно виконати корекцію $К = +2^{-k}$.

На рис. Б1-2.9 зображена операційна схема реалізації арифметичного зсуву ліворуч від'ємних чисел, поданих в ОК.

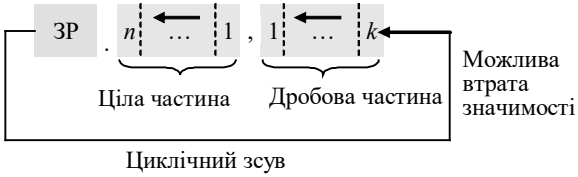


Рис. Б1-2.9. Операційна схема арифметичного зсуву ліворуч від'ємних чисел в ОК

Приклад

Завдання. Виконати арифметичний зсув ліворуч на один розряд двійкових чисел, поданих в ОК.

$$A_{(2)} = 1.\ 1101,\ 1010;$$
$$B_{(2)} = 1.\ 0011,\ 0011.$$

Виконання завдання

A	1.1101,0101	
$\leftarrow A$	1.1010,1011	
B	1.0011,0011	
$\leftarrow B$	0.0110,0111	*Втрата значимості*

Арифметичний зсув праворуч чисел, поданих в ОК

Правило. За зсуву праворуч від'ємного числа знаковий розряд переходить у поле основних розрядів і знову заповнюється тим самим значенням.

На рис. Б1-2.10 зображена операційна схема реалізації арифметичного зсуву праворуч від'ємних чисел, поданих в ОК.

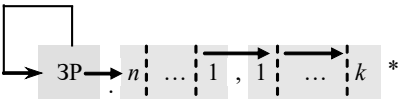


Рис. Б1-2.10. Операційна схема арифметичного зсуву праворуч від'ємних чисел в ОК

Приклад

Завдання. Виконати арифметичний зсув праворуч на один розряд двійкових чисел, поданих в ОК.

$$A_{(2)} = 0.\ 1011,\ 1001;$$
$$B_{(2)} = 1.\ 1010,\ 0011.$$

Виконання завдання

$$\begin{array}{ll} A & 0.1011,1001 \\ \rightarrow A & 0.0101,1100^{*} \end{array}$$

$$\begin{array}{ll} B & 1.1010,0011 \\ \rightarrow B & 1.1101,0001^{*} \end{array}$$

Арифметичний зсув ліворуч чисел, поданих у ДК

Правило. За зсуву ліворуч числа, поданого у ДК розряди, що звільнилися, заповнюються нулями. Якщо знаковий розряд змінює значущість виникає втрата значимості числа.

На рис. Б1-2.11 зображена операційна схема реалізації арифметичного зсуву ліворуч чисел, поданих у ДК.

Рис. Б1-2.11. Операційна схема арифметичного зсуву ліворуч від'ємних чисел у ДК

Арифметичний зсув праворуч чисел, поданих у ДК

Правило. За зсуву праворуч числа, поданого у ДК, знаковий розряд розповсюджується у поле основних розрядів і знову заповнюється тим самим значенням. У результаті може виникнути похибка.

Операційна схема реалізації арифметичного зсуву праворуч чисел, поданих у ДК, зображена на рис. Б1-2.12.

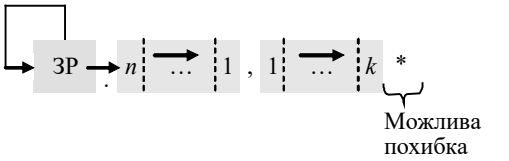


Рис. Б1-2.12. Операційна схема арифметичного зсуву праворуч від'ємних чисел у ДК

Приклад

Завдання. Виконати арифметичний зсув праворуч і ліворуч на один розряд двійкових чисел, поданих у ДК.

$$A_{(2)} = 0.\ 1011,\ 0111;$$
$$B_{(2)} = 1.\ 1011,\ 1001.$$

Виконання завдання

$$\begin{array}{ll} A & 0.1011,0111 \\ \leftarrow A & *1.0110,1110 \\ A \rightarrow & 0.0101,1011* \end{array}$$

$$\begin{array}{ll} B & 1.1011,1001 \\ \leftarrow B & *1.0111,0010 \\ \rightarrow B & 1.1101,1100* \end{array}$$

Б1-3. Множення чисел

Б1-3.1. Способи множення чисел, поданих паралельним кодом

Під час множення чисел у прямих кодах знакові та основні розряди обробляються окремо. Для визначення знака добутку здійснюють підсумовування по модулю 2 цифр, що розміщуються в знакових розрядах співмножників.

Будемо вважати, що множене Y та множник X — правильні двійкові дроби вигляду $X = 0, x_1, x_2, ..., x_n$, $Y = 0, y_1, y_2, ..., y_n$, де $x_i, y_i \in \{0,1\}$. Тоді добуток Z абсолютних величин чисел Y та X дорівнює

$$Z = YX = Yx_1 2^{-1} + Yx_2 2^{-2} + ... + Yx_i 2^{-i} + ... + Yx_n 2^{-n}. \quad \text{(Б1-3.1)}$$

Множення двох чисел Y та X може бути реалізоване шляхом виконання визначеного циклічного процесу, характер якого залежить від конкретної форми виразу (Б1-3.1). Один цикл множення складається з додавання чергового часткового добутку, що являє собою добуток множеного на одну цифру множника, до суми часткових добутків. Розрізняють чотири способи множення.

Перший спосіб

Вираз (Б1-3.1) можна подати у вигляді:

$$Z = YX = ((...((0 + Yx_n)2^{-1} + Yx_{n-1})2^{-1} + ... + Yx_i)2^{-1} + ... + Yx_1)2^{-1},$$

з отриманого виразу виходить, що отримання суми часткових добутків в i-му циклі, де $i = \overline{1, n}$, зводиться до обчислення виразу

$$Z_i = (Z_{i-1} + Yx_{n-i+1})2^{-1}$$

з початковими значеннями $i = 1, Z_0 = 0$, причому $Z_n = Z = Y \cdot X$.

У розглянутому способі множення здійснюється з молодших розрядів множника, сума часткових добутків зсувається вправо, а множене залишається нерухомим.

Другий спосіб

Подамо вираз (Б1-3.1) у такому вигляді:

$$Z = ((...((0 + Y2^{-n}x_n) + Y2^{-n+1}x_{n-1}) + ... + Y2^{-1}x_1.$$

Очевидно, що процес множення може бути зведений до n-кратного виконання циклу

$$Z_i = Z_{i-1} + Y_i x_{n-i+1}, \; Y_i = 2Y_{i-1}.$$

З початковими значеннями $i=1,\ Y_0 = Y2^{-n},\ Z_0 = 0.$

У розглянутому способі множення здійснюється з молодших розрядів, множене зсувається вліво, а сума часткових добутків залишається нерухомою.

Третій спосіб

Подамо вираз (Б1-3.1) у такому вигляді:

$$Z = ((...((0 + Y2^{-n}x_1)2 + Y2^{-n}x_2)2 + ... + Y2^{-n}x_i)2 + ... + Y2^{-n}x_n.$$

Суму часткових добутків у *i*-му циклі $i=\overline{1,n}$ можна одержати за виразом

$$Z_i = 2Z_{i-1} + Y2^{-n}x_i,$$

з початковими значеннями $i=1,\ Z_0 = 0.$

У розглянутому способі множення здійснюється зі старших розрядів множника, сума часткових добутків зсувається вліво, а множене нерухоме.

Четвертий спосіб

Подамо вираз (Б1-3.1) так:

$$Z = ((...((0 + Y2^{-1}x_1) + Y2^{-2}x_2) + ... + Y2^{-i}x_i) + ... + Y2^{-n}x_n.$$

У цьому випадку процес множення може бути зведений до *n*-кратного виконання циклу

$$Z_i = Z_{i-1} + Y_{i-1}x_i, \quad Y_i = Y_{i-1}2^{-1}$$

з початковими значеннями $i=1,\ Y_0 = Y2^{-1},\ Z_0 = 0.$

У розглянутому способі множення виконується зі старших розрядів множника, сума часткових добутків залишається нерухомою, а множене зсувається вправо.

Для формування і накопичення суми часткових добутків можна використовувати або комбінаційний суматор (*SM*) і регістр добутку, або тільки накопичувальний суматор, який у функціональному відношенні можна розглядати як композицію комбінаційного суматора і регістра.

Принцип побудови пристроїв, що реалізують різні способи множення, показаний на рис. Б1-3.1, де *RG*3 — регістр множеного, *RG*1 — регістр добутку, *RG*2 — регістр множника. Цифрами зазначені номери розрядів *SM* і регістрів, а стрілками показаний напрямок зсуву кодів у регістрах.

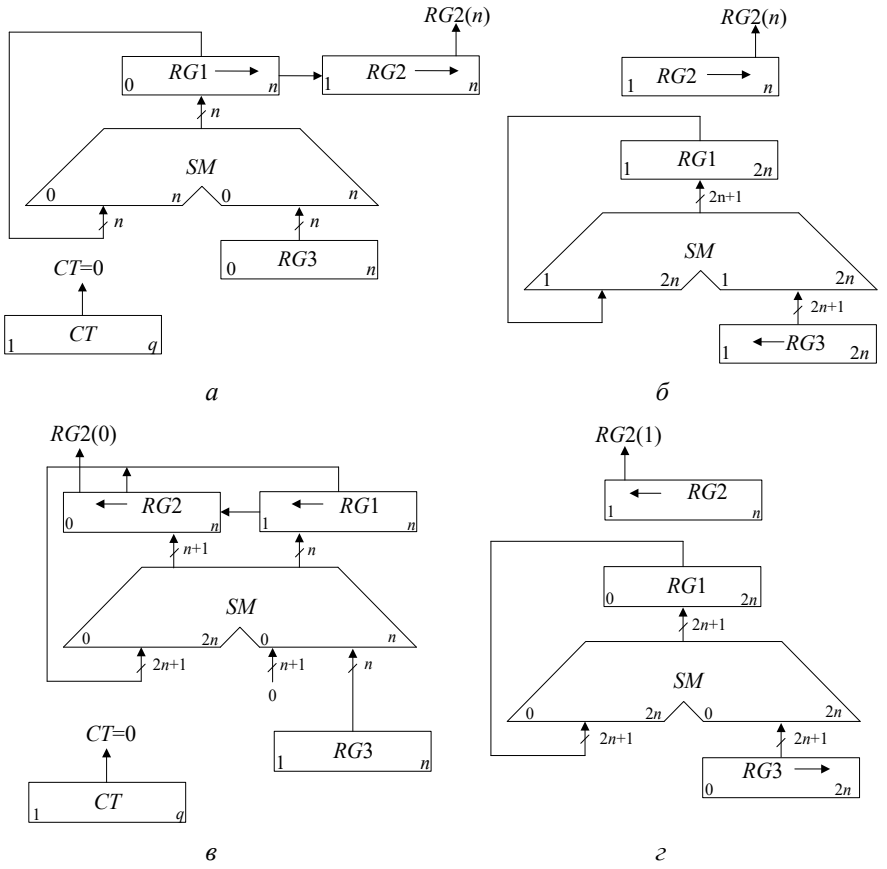


Рис. Б1-3.1. Операційні схеми пристроїв для множення чисел:
а — перший спосіб; *б* — другий спосіб; *в* — третій спосіб; *г* — четвертий спосіб

Цифри, записані в молодших розрядах регістрів *RG*3 і *RG*1, за реалізації першого способу мають вагу 2^{-n}, а за реалізації інших способів — 2^{-2n}. Перед початком множення будь-яким способом регістр *RG1* встановлюється в нульовий стан. Підрахунок кількості циклів множення забезпечують лічильники *CT*, відповідно до чого обирається його розрядність *q*.

Під час множення *першим способом* (рис. Б1-3.1, *а*) в першому такті *i*-го циклу аналізується значення *RG2(n)* — молодшого *n*-го розряду регістру *RG*2, в якому знаходиться чергова цифра множника. Вміст *RG*3 додається до суми часткових добутків, що знаходяться

в регістрі RG1, якщо *RG*2(*n*) = 1, або не додається, якщо *RG*2(*n*) = 0. В другому такті здійснюється правий зсув у регістрах *RG*1 і *RG*2, що еквівалентно множенню їх вмісту на 2^{-1}. За зсуву цифра молодшого розряду регістру *RG*1 записується у вивільнюваний старший розряд регістру *RG*2. Після виконання *n* циклів молодші розряди 2*n*-розрядного добутку будуть записані в регістр *RG*2, а старші — у *RG*1.

Час множення, якщо не застосовуються методи прискорення операції (див. розділ Б1-3.2), визначається виразом $t_{\text{м}} = n(t_{\text{п}} + t_{\text{з}})$, де $t_{\text{п}}$ і $t_{\text{з}}$ — тривалості тактів підсумовування і зсуву відповідно.

Перед початком множення *другим способом* (рис. Б1-3.1, *б*) множник *X* записують у регістр *RG*2, а множене *Y* — в молодші розряди регістру *RG*3 (тобто в регістрі *RG*3 установлюють $Y_0 = Y2^{-n}$). В кожному *i*-му циклі множення додаванням кодів *RG*3 і *RG*1 керує цифра *RG*2(*n*), а в регістрі *RG*3 здійснюється зсув вліво на один розряд, у результаті чого формується величина $Y_i = 2Y_{i-1}$. Оскільки сума часткових добутків у процесі множення нерухома, зсув у регістрі *RG3* можна сполучити в часі з підсумовуванням (як правило, $t_{\text{п}} \geq t_{\text{з}}$). В цьому випадку $t_{\text{і}} = nt_{\text{ї}}$. Завершення операції множення визначається за нульовим вмістом регістру *RG*2, що також приводить до збільшення швидкодії, якщо множник ненормалізований.

Під час множення *третім способом* (рис. Б1-3.1, *в*) вага молодшого розряду *RG*3 дорівнює 2^{-2n}, тому код у регістрі *RG3* являє собою значення $Y2^{-n}$. На початку кожного циклу множення здійснюється лівий зсув у регістрах *RG*1 і *RG*2, а потім виконується додавання, яким керує *RG*2(1). У результаті підсумовування вмісту *RG*3 і *RG*1 може виникнути перенос у молодший розряд регістру *RG*2. У старшій частині суматора, на якому здійснюється підсумовування коду *RG*2 з нулями, відбувається поширення переносу. Збільшення довжини *RG*2 на один розряд усуває можливість поширення переносу в розряди множника. Після виконання *n* циклів молодші розряди добутку будуть знаходитися в регістрі *RG*1, а старші — в регістрі *RG*2. Час множення третім способом визначається аналогічно першому способу і дорівнює $t_{\text{і}} = n(t_{\text{ї}} + t_{\text{ç}})$.

Перед множенням *четвертим способом* (рис. Б1-3.1, *г*) множник записують у регістр *RG*2, а множене — в старші розряди регістру *RG*3 (тобто в *RG*3 установлюють $Y_0 = Y2^{-1}$). У кожному циклі цифра *RG*2(1), що знаходиться в старшому розряді регістру *RG*2, керує підсумовуванням, а в *RG*3 здійснюється правий зсув на один розряд, що еквівалентно множенню вмісту цього регістра на 2^{-1}. Час виконання множення четвертим способом складає $t_{\text{і}} = nt_{\text{ї}}$, визначається аналогічно другому способу.

В ЕОМ під час роботи з дробовими числами часто потрібно обчислювати не $2n$, а тільки $(n+1)$ цифр добутку й округляти його до n розрядів. У цьому випадку під час реалізації другого способу можна зменшити довжину *SM* і *RG*1, а під час реалізації четвертого — зменшити довжину *SM*, *RG*1 і *RG*3. Для того, щоб похибка від відкидання молодших розрядів не перевищила половини ваги n-го розряду результату, в перерахованих вузлах досить мати тільки по l додаткових молодших розрядів, де l вибирається з умови

$$l \geq 1 + \log_2(n - l - 1).$$

Операція округлення здійснюється зазвичай шляхом додавання одиниці до n + 1-го розряду результату і відкидання всіх розрядів, розташованих правіше n-го. При цьому похибка стає знакозмінною, а максимальне абсолютне її значення не перевищує половини ваги молодшого розряду. Додаткового такту підсумовування для округлення не потрібно. Досить записати одиницю перед початком множення в той розряд регістру *RG*1, що після виконання множення залишається старшим розрядом, який відкидається.

У процесі формування суми часткових добутків код з регістру *RG*1 видається на суматор *SM*, а з виходів *SM* знову записується в регістр *RG*1. У зв'язку з цим під час використання потенційних елементів регістр *RG*1 будують на тригерах із внутрішньою затримкою. Характер керуючих сигналів і ланцюга, на який вони впливають, визначається конкретною теоретичною реалізацією вузлів і використовуваною елементною базою.

В операційних пристроях, що реалізують другий і четвертий способи множення, можна без пересилань кодів між регістрами обчислювати вирази вигляду $\sum X_i Y_i$, де $(i = \overline{1, n},)$, для чого достатньо черговий результат операції залишати в регістрі *RG*2, який в цьому випадку повинен мати додаткові старші розряди.

В операційному пристрої, що реалізує третій спосіб, можна без пересилань обчислювати, наприклад, функції вигляду X_i. Для цього множник X перед початком обчислення записується в регістр *RG*3 і в молодші розряди регістру *RG*2, а потім $(i - 1)$ разів виконується операція множення з округленням проміжних результатів до n розрядів. Після кожної чергової операції регістр *RG*1 встановлюється в нульовий стан. Остаточний результат знаходитиметься в n молодших розрядах регістру *RG*2. Найпростішими є пристрої, що реалізують перший спосіб, а найбільш швидкодіючими — другий і четвертий. Однак другий спосіб не має особливих переваг порівняно з четвертим і, крім того, вимагає великих апаратурних витрат при реалізації.

Б1-3.2. Способи прискорення множення чисел, поданих паралельним кодом

Для виконання операції множення з використанням основних чотирьох методів (див. розділ Б1-3.1) необхідно зробити n операцій зсуву й до n операцій додавання (де n — розрядність операндів). За рівноймовірних значень 0 та 1 цифр множника середнє число підсумовувань дорівнює $n/2$.

Зменшення часу обчислень досягається застосуванням логічних та апаратурних методів прискорення. *Логічні методи* прискорення характеризуються ускладненням пристрою керування практично за незмінної операційної частини пристрою множення (регістрів, суматорів, тощо). Такі методи засновані на одночасному перегляді декількох цифр множника. Логічні методи дозволяють одержати в середньому $n/3$ додавань, причому максимальна кількість додавань становить $n/2$.

Більш ефективними методами прискорення операції множення є *апаратні методи*, під час реалізації яких ускладнюється операційна частина пристрою. Найбільше прискорення операції множення досягається із застосуванням методів другого порядку (апаратурні витрати зростають пропорційно n^2), до яких належать матричні методи множення.

Матричні методи множення засновані на застосуванні комбінаційного множного пристрою, у якому множене помножується на всі розряди множника одночасно. За допомогою матриці елементів І формується матриця часткових добутків, після чого цифри матриці підсумовуються відповідно до ваг розрядів за допомогою однорозрядних суматорів (рис. Б1-3.2). Матриця елементів І містить n^2 елементів І з двома входами, що формують n^2 цифр $z_{ij} = y_i \cdot x_j$ (де i = 1, 2, …, n, j = 1, 2, .., n), де y_i, x_j — розряди співмножників Y і X, n — розрядність співмножників. Час формування матриці часткових добутків визначається затримкою сигналу в одному логічному елементі І.

Варіанти побудови матричних пристроїв для множення відрізняються способом розповсюдження переносів у матрицях однорозрядних суматорів.

Розглянемо два типи матричних пристроїв для множення, що відрізняються організацією матриць суматорів.

Спосіб побудови таких матриць за n = 8 показаний на рис. Б1-3.3 і рис. Б1-3.4. Кожний прямокутник відповідає однорозрядному двійковому суматору. Знизу суматора показаний вихід розрядної суми, а зліва — перенос у сусідній розряд.

Рис. Б1-3.2. Матрична схема множення

Матриця на рис. Б1-3.3 складається з $n(n-1)$ однорозрядних суматорів, які поєднані в $(n-1)$ рядків і $2n$ стовпців. Кожен рядок містить n однорозрядних суматорів, пов'язаних між собою ланцюгами переносів. Суматори першого ряду формують суму часткового добутку, множеного на перший розряд і зсунутого вліво на один розряд часткового добутку, множеного на другий розряд множника. Отримана перша сума часткових добутків підсумовується на суматорах другого ряду зі зсунутим вліво на два розряди частковим добутком, множеного на третій розряд множника й так далі. Кожен частковий добуток, що надходить на входи суматорів, формується за допомогою логічних елементів І (які не показані на рисунку). У розглянутій схемі переноси розповсюджуються справа наліво, а суми зверху вниз.

Час множення двох n-розрядних чисел оцінюється виразом

$$t_{\text{ì}} = (3n-4)t_{+} + t_{\text{å}},$$

де t_{+} — затримка сигналів на суматорі, t_{e} — затримка сигналів на логічному елементі І.

Пристрій на рис. Б1-3.4 характеризується тим, що переноси розповсюджуються по рядках справа наліво тільки в останньому рядку. В інших рядках переноси розповсюджуються по діагоналі (справа вниз наліво), а сигнали сум — зверху вниз.

Час множення двох n-розрядних чисел оцінюється виразом

$$t_{\text{ì}} = (2n-2)t_{+} + t_{\text{Ä}}.$$

Таким чином, за використання однорозрядних суматорів, всі розглянуті пристрої вимагають однакових апаратурних витрат, але найбільш швидкодіючим є пристрій на рис. Б1-3.4.

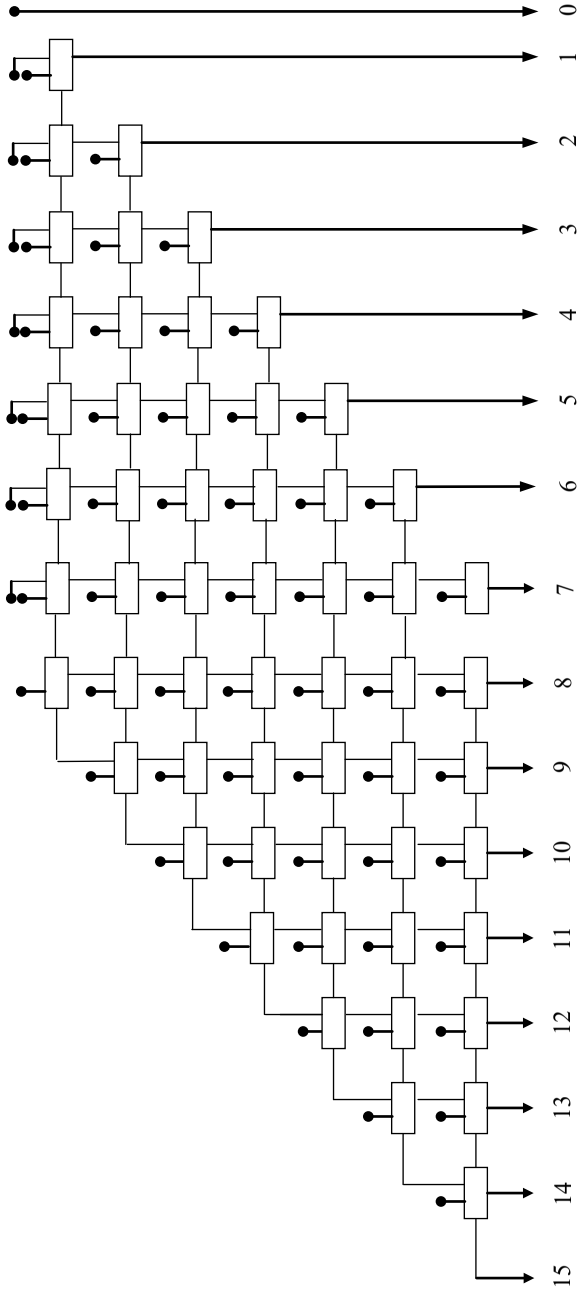


Рис. Б1-3.3. Матриця суматорів
із розповсюдженням переносів по рядках

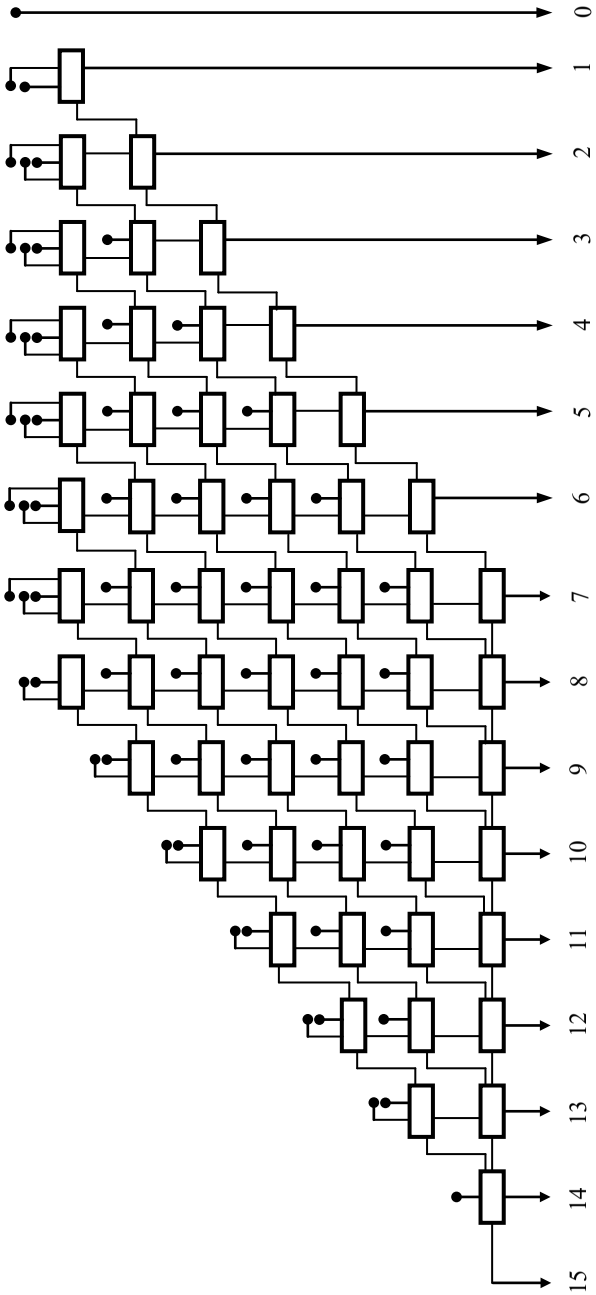


Рис. Б1-3.4. Матриця суматорів
із розповсюдженням переносів по діагоналях

Б1-3.3. Способи множення чисел, поданих послідовним кодом

Розглянемо методи множення, які забезпечують суміщення в часі процесів порозрядного введення операндів й обчислення результату.

Будемо вважати, що під час множення двійкових чисел операнди приймаються в операційний пристрій порозрядно (послідовним кодом), причому період t_n надходження чергових розрядів операндів визначаються зовнішніми стосовно операційного пристрою факторами.

Суміщення в часі процесів порозрядного введення операндів й їх оброблення не може бути досягнуто застосуванням пристроїв паралельної дії, тому що в цих пристроях операнди мають бути подані паралельним кодом. Таким чином, за множення операндів, що надходять порозрядно із періодом t_n, у випадку застосування пристроїв паралельної дії час T, необхідний для одержання результату, визначається виразом:

$$T = T_1 + T_2, \qquad \text{(Б1-3.2)}$$

де T_1 — час порозрядного введення операндів,

T_2 — час виконання операції множення в пристрої паралельної дії.

Застосовуючи логічні або апаратні операції множення, можна зменшити складову T_2 у виразі (Б1-3.2), однак її значення завжди буде відмінне від нуля.

Нижче розглядаються алгоритми виконання операції множення, що дозволяють усунути складову T_2 у виразі (Б1-3.2).

Нехай множене A и множник B є двійковими дробами вигляду:

$$A = 0, a_1 a_2 \cdots a_i \cdots a_{n-1} a_n; \;\; B = 0, b_1 b_2 \cdots b_i \cdots b_{n-1} b_n,$$

де a_i й b_i — двійкові цифри множеного й множника.

Розглянемо спочатку випадок, коли співмножники вводяться, починаючи з молодших розрядів. При цьому в кожному i-му циклі обчислень є можливість аналізувати тільки i молодших розрядів співмножників.

Введемо такі позначення: A_i і B_i — коди, кожний із котрих містить по i дійсних молодших розрядів відповідно множеного й множника й нулі у всіх $(n\text{-}i)$ старших розрядах.

Тоді коди A_i і B_i мають вигляд:

$$A_i = 0, \underbrace{00...00}_{n-1} a_{n-i+1} ... a_{n-1} a_n = \sum_{j=n-i+1}^{n} a_j 2^{-j}, \qquad \text{(Б1-3.3)}$$

$$B_i = 0, \underbrace{00...00}_{n-1} b_{n-i+1} ... b_{n-1} b_n = \sum_{j=n-i+1}^{n} b_j 2^{-j}, \qquad \text{(Б1-3.4)}$$

причому $B_0 = 0$.

Покажемо, що добуток C співмножників A и B може бути обчислений за такою формулою:

$$C = \sum_{i=1}^{n} A_i b_{n-i+1} 2^{-(n-i+1)} + \sum_{i=0}^{n-1} B_i a_{n-i} 2^{-(n-i)}. \qquad \text{(Б1-3.5)}$$

Справді, виходячи з того що $B_0 = 0$, другу суму у виразі (Б1-3.5) з урахуванням виразу (Б1-3.4) можна подати у вигляді:

$$\sum_{i=1}^{n-1} \left(a_{n-i} 2^{-(n-1)} \sum_{j=n-i+1}^{n} b_j 2^{-j} \right),$$

або

$$a_{n-1} 2^{-(n-1)} b_n 2^{-n} +$$

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

$$+a_{n-2} 2^{-(n-2)} b_n 2^{-n} + a_{n-2} 2^{-(n-2)} b_{n-1} 2^{-(n-1)} +$$

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

$$+a_2 2^{-2} b_n 2^{-n} + a_2 2^{-2} b_{n-1} 2^{-(n-1)} + \ldots + a_2 2^{-2} b_3 2^{-3} +$$

$$+a_1 2^{-1} b_n 2^{-n} + a_1 2^{-1} b_{n-1} 2^{-(n-1)} + \ldots + a_1 2^{-1} b_3 2^{-3} + a_1 2^{-1} b_2 2^{-2}.$$

Підсумовуючи отриманий ряд по стовпцях, знаходимо:

$$\sum_{i=1}^{n-1} \left(b_{n-i+1} 2^{-(n-i+1)} \sum_{j=1}^{n-i} a_j 2^{-j} \right). \qquad \text{(Б1-3.6)}$$

Підставимо у вираз (Б1-3.5) значення A_i з виразу (Б1-3.3) та з урахуванням виразу (Б1-3.6) отримаємо:

$$C = \sum_{i=1}^{n} \left(b_{n-i+1} 2^{-(n-i+1)} \sum_{j=n-i+1}^{n} a_j 2^{-j} \right) + \sum_{i=1}^{n-1} \left(b_{n-i+1} 2^{-(n-i+1)} \sum_{j=1}^{n-i} a_j 2^{-j} \right).$$

Поєднуючи обидва члени отриманого виразу й взявши до уваги, те, що другий член виразу дорівнює нулю за ($i = n$), прийдемо до наступного виразу:

$$C = \sum_{i=1}^{n} \left(b_{n-i+1} 2^{-(n-i+1)} \sum_{j=1}^{n} a_j 2^{-j} \right).$$

Сума в дужках не залежить від індексу i, але в цьому випадку

$$C = \sum_{j=1}^{n} a_j 2^{-j} \sum_{i=1}^{n} b_{n-i+1} 2^{-(n-i+1)} = AB,$$

отже, обчислення добутку чисел A и B за формулою (Б1-3.5) приводить до правильного результату.

Якщо співмножники A и B вводяться, починаючи зі старших розрядів, то їх добуток C можна одержати за формулою:

$$C = \sum_{i=1}^{n} A_i^* b_i 2^{-i} + \sum_{i=0}^{n-1} B_i^* a_{i+1} 2^{-(i+1)}, \qquad \text{(Б1-3.7)}$$

де

$$A_i^* = 0,\, a_1 a_2 ... a_i \underbrace{00...00}_{n-i} = \sum_{j=1}^{i} a_j 2^{-j};$$

$$B_i^* = 0,\, b_1 b_2 ... b_i \underbrace{00...00}_{n-i} = \sum_{j=1}^{i} b_j 2^{-j}.$$

На відміну від кодів A_i і B_i коди A^*_i й B^*_i містять дійсні цифри відповідних операндів в i старших розрядах.

Формула (Б1-3.7) дає правильне значення результату, у чому нескладно переконатися, зробивши перетворення, аналогічні викладеним вище.

Під час обчислення добутку за формулами (Б1-3.5) і (Б1-3.7) у кожному i-му циклі обчислень використовується інформація тільки про i розрядів співмножників, які на той час виявляються прийнятими.

Розглянемо декілька способів множення чисел із порозрядним введенням операндів.

Перший спосіб

Якщо подати вираз (Б1-3.5) у вигляді:

$$C = ((...(((0 + A_1 b_n + B_0 a_n)2^{-1} + A_2 b_{n-1} + B_1 a_{n-1})2^{-1} + ...$$
$$... + A_i b_{n-i+1} + B_{i-1} a_{n-i+1})2^{-1} + ... + A_{n-1} b_2 + B_{n-2} a_2)2^{-1} +$$
$$+ A_n b_1 + B_{n-1} a_1)2^{-1},$$

нескладно помітити, що одержання суми часткових добутків в i-му циклі (де i=1,2,...n) зводиться до обчислення виразу

$$C_i = (C_{i-1} + A_i b_{n-i+1} + B_{i-1} a_{n-i+1})2^{-1}$$

з початковими значеннями $C_0 = 0$, $B_0 = 0$.

Під час реалізації цього способу співмножники вводяться, починаючи з молодших розрядів, а сума часткових добутків зсувається вправо.

Другий спосіб

Запишемо вираз (Б1-3.6) у вигляді:

$$C = ((...((0 + A_1 b_n 2^{-n} + B_0 a_n 2^{-n}) + A_2 b_{n-1} 2^{-(n-1)} + B_1 a_{n-1} 2^{-(n-1)}) + ...$$
$$... + A_i b_{n-i+1} 2^{-(n-i+1)} + B_{i-1} a_{n-i+1} 2^{-(n-i+1)}) + ...$$
$$... + A_{n-1} b_2 2^{-2} + B_{n-2} a_2 2^{-2}) + A_n b_1 2^{-1} + B_{n-1} a_1 2^{-1}.$$

Очевидно, що суму часткових добутків в i-му циклі (де i=1, 2, …, n) можна одержати за формулою:

$$C_i = C_{i-1} + A_i b_{n-i+1} 2^{-(n-i+1)} + B_{i-1} a_{n-i+1} 2^{-(n-i+1)},$$

де $C_0 = 0$, $B_0 = 0$.

У цьому випадку співмножники вводяться, починаючи з молодших розрядів, сума часткових добутків залишається нерухомою, а A_i і B_i зсуваються вліво.

Третій спосіб

Перетворимо вираз (Б1-3.7) до вигляду:

$$C = ((...((0 + A_1^* b_1 2^{-n} + B_0^* a_1 2^{-n})2 + A_2^* b_2 2^{-n} + B_1^* a_2 2^{-n})2 + ...$$
$$... + A_i^* b_i 2^{-n} + B_{i-1}^* a_i 2^{-n)})2 + ... + A_{n-1}^* b_{n-1} 2^{-n} +$$
$$+ B_{n-2}^* a_{n-1} 2^{-n})2 + A_n^* b_n 2^{-n} + B_{n-1}^* a_n 2^{-n}.$$

З отриманого виразу випливає, що процес множення може бути зведений до n-разового виконання циклу

$$C_i = 2C_{i-1} + A_i^* b_i 2^{-n} + B_{i-1}^* a_i 2^{-n}$$

з початковими значеннями $C_0 = 0$, $B^*_0 = 0$.

Під час реалізації даного способу співмножники вводяться, починаючи зі старших розрядів, а сума часткових добутків зсувається вліво.

Четвертий спосіб

Подамо вираз (Б1-3.7) у вигляді:

$$C = ((...((0 + A_1^* b_1 2^{-1} + B_0^* a_1 2^{-1}) + A_2^* b_2 2^{-2} + B_1^* a_2 2^{-2}) + ...$$
$$... + A_i^* b_i 2^{-i} + B_{i-1}^* a_i 2^{-i)}) + ... + A_{n-1}^* b_{n-1} 2^{-(n-1)} +$$
$$+ B_{n-2}^* a_{n-1} 2^{-(n-1)}) + A_n^* b_n 2^{-n} + B_{n-1}^* a_n 2^{-n}.$$

Очевидно, що процес множення може бути зведений до n-разового виконання циклу

$$C_i = C_{i-1} + A_i^* b_i 2^{-i} + B_{i-1}^* a_i 2^{-i}$$

з початковими значеннями $C_0 = 0$, $B^*_0 = 0$.

У цьому випадку співмножники вводяться, починаючи зі старших розрядів, сума часткових добутків залишається нерухомою, а коди A^*_i й B^*_i зсуваються вправо.

Розглянемо можливу операційну схему реалізації першого способа (рис. Б1-3.5).

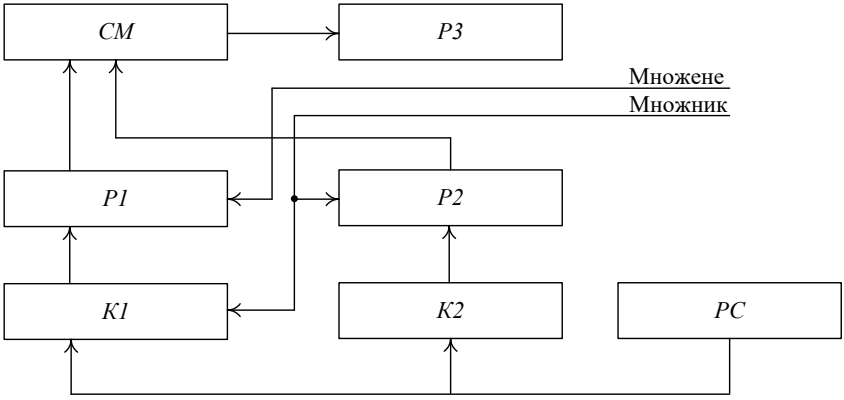


Рис. Б1-3.5. Операційна схема, для реалізації першого способу множення чисел із порозрядним введенням операндів

Пристрій містить $(n-1)$-розрядний регістр множника ($P1$), n-розрядні регістри множеного ($P2$) і молодших розрядів добутку ($P3$), $(n+1)$-розрядний накопичуючий суматор (CM), комутатори ($K1$, $K2$) і розподільник сигналів PC. Суматор CM і регістр $P3$ мають ланцюги правого зсуву на один розряд. Як розподільник сигналів PC використаємо регістр зі зсувом або лічильник з дешифратором. Функції комутаторів $K1$ і $K2$ виконують ланцюги прийому коду $P1$ і $P2$. у кожному i-му циклі коди A_i і B_i формуються відповідно в регістрах $P2$ і $P1$ шляхом запису цифр $a_{(n-i+1)}$ та $b_{(n-i+1)}$ у відповідні розряди регістрів через комутатори $K1$ і $K2$, якими керує вузол PC. Першою на вхід пристрою приймається цифра $a_{(n-i+1)}$, що керує передачею в суматор CM коду $B_{(i-1)}$, сформованого в регістрі $P1$ в $(i-1)$-му циклі. Одночасно із цим у регістрі $P2$ формується код A_i. Після чого приймається черговий розряд $b_{(n-i+1)}$ множника, що керує передачею в суматор CM коду A_i, а в регістрі $P1$ одночасно із цим форму-

ється код B_i. Далі в суматорі *СМ* та у регістрі *Р*3 здійснюється правий зсув на один розряд, що еквівалентно множенню на 2^{-1}. Одночасно зі зсувом виконується перебудова комутаторів *К*1 і *К*2 для прийому чергових розрядів співмножників.

Час $t_{\text{ц}}$ виконання одного циклу обчислень у такому пристрої дорівнює $t_{\text{ö}} = 2t_{\text{ä}} + t_{\text{ç}}$, де $t_{\text{д}}$ — час додавання кодів у суматорі, а $t_{\text{з}}$ — час зсуву.

Варто відзначити, що застосування другого та четвертого способів дозволяє зменшити час виконання одного циклу. У цьому випадку, як і у пристроях паралельної дії, де сума часткових добутків залишається нерухомою, процес зсуву може бути суміщений у часі із процесом підсумовування.

Таким чином, за множення чисел за кожним із наведених способів процес обчислення закінчується одночасно із прийомом на вхід пристрою останніх розрядів операндів. Якщо виконується співвідношення $t_u \le t_n$, то процеси порозрядного введення операндів й обчислення результату можуть бути суміщені у часі, тобто складова часу T_2 у виразі (Б1-3.2) у цьому випадку буде відсутня.

Наведені способи можуть знайти застосування в пристроях, що виготовляють у вигляді великих інтегральних схем, тому що вони, завдяки порозрядному введенню операндів, мають оптимальну кількість зовнішніх зв'язків, маючи при цьому таку саму швидкодію, як пристрої паралельного типу.

Б1-4. Ділення чисел

Б1-4.1. Методи ділення чисел

Існують два основних методи ділення чисел:

— з відновленням від'ємного залишку;

— без відновлення від'ємного залишку.

Реалізація цих методів вимагає приблизно однакового обсягу устаткування, але під час ділення першим методом потрібно більше часу для виконання операції. Тому метод ділення чисел без відновлення залишку є більш ефективним, тому надалі розглядатимемо саме його.

Нехай ділене X і дільник Y є n-розрядними правильними дробами, поданими в прямому коді. В цьому випадку знакові й основні розряди операндів обробляються окремо. Знак результату визначається шляхом підсумовування за модулем два цифр, записаних у знакових розрядах.

Алгоритм ділення чисел без відновлення залишку зводиться до виконання таких дій.

1. Одержати різницю $R_0 = X - Y$. Якщо $R_0 \geq 0$, то цифра Z_0 частки, що має вагу 2^0, дорівнює 1, а за $R_0 < 0$ — дорівнює 0. Різниця R_0 є залишком.

2. Подвоїти залишок (тобто одержати значення $2R_i$).

3. За $2R_i < 0$ додати Y, в зворотному випадку, якщо $2R_0 \geq 0$, відняти Y. Якщо знову отриманий залишок $R_{i+1} \geq 0$, то $Z_{i+1} = 1$, інакше $Z_{i+1} = 0$.

4. Повторити дії, описані в пунктах 2 та 3, $(n - 1)$ раз.

Пункт 2 алгоритму можна замінити пунктом «зменшити вдвічі дільник». Наявність двох інтерпретацій другого пункту дає два основних способи реалізації ділення.

Перший спосіб

Під час реалізації ділення за першим способом здійснюється зсув вліво залишку при нерухомому дільнику, такий спосіб називається *діленням із зсувом залишку*. На рис. Б1-4.1 показаний принцип побудови пристрою для ділення чисел. Черговий залишок формується в регістрі $RG2$ (у вихідному стані в цьому регістрі записане ділене X). Дільник Y знаходиться в регістрі $RG1$. Максимальний час одержання цифри результату визначається виразом $t = t_Д + t_З$, де $t_Д$ — тривалість виконання мікрооперації додавання-віднімання; $t_З$ — тривалість виконання мікрооперації зсуву. Результат формується в регістрі $RG3$ за $(n + 1)$ циклів.

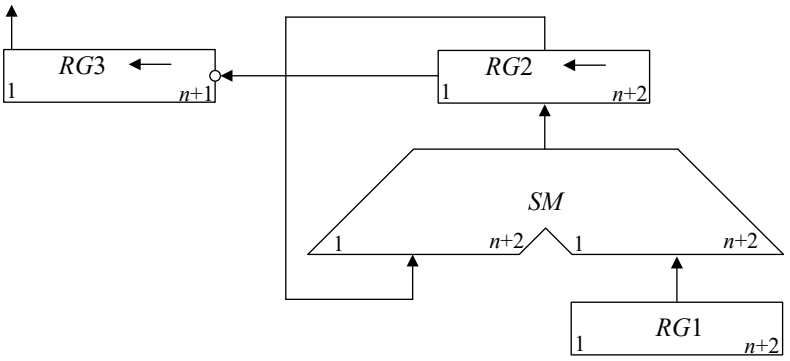


Рис. Б1-4.1. Операційна схема пристрою ділення із зсувом залишку

Другий спосіб

Під час реалізації ділення другим способом, який називається *діленням із зсувом дільника*, збільшується розрядність регістрів *RG*1, *RG*3 і суматора *SM*. Принцип побудови пристрою для ділення чисел за другим способом показаний на рис. Б1-4.2. В цьому випадку процеси додавання-віднімання і зсуву можуть бути сполучені в часі. Отже, для ділення за другим способом час одержання цифри результату дорівнює $t = t_Д$. Цифра результату формується на виході переносу суматора *SM*(*p*).

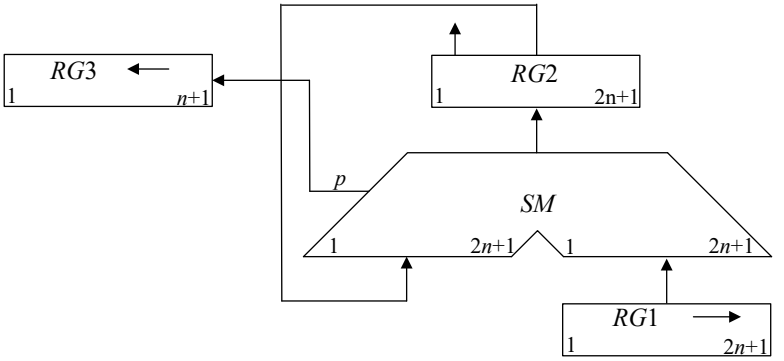


Рис. Б1-4.2. Операційна схема пристрою ділення із зсувом дільника

Під час ділення чисел з фіксованою комою має бути передбачена можливість фіксації переповнення розрядної сітки. Ознака переповнення виробляється, якщо в першому циклі обчислення отримана цифра результату із значенням 1.

Б1-5. Обчислення функцій

Б1-5.1. Метод обчислення квадратного кореня

Розглянемо метод обчислення функції $Y = \sqrt{X}$ за послідовного введення коду аргументу X зі старших розрядів. Такий метод дозволяє виконувати процеси введення аргументу та обчислювання функції у режимі суміщення.

Будемо вважати, що значення аргументу подане нормалізованим n-розрядним числом у позиційній системі числення з основою 2.

Якщо для подання значення функції використовувати надлишкову позиційну систему числення із цифрами $\{0, 1, 2\}$ і основою 2, то i-ту цифру значення функції можна визначати раніше вводу $(i+1)$-ої цифри значення аргументу. Доведемо це, використовуючи метод математичної індукції.

Значення функції, що подане n розрядами, запишемо у вигляді:

$$Y = \sum_{i=1}^{n} y_i 2^{-i},$$

де $\frac{1}{2} < Y < 1,\ \ y_i \in \{0,\ 1,\ 2\}$ — цифри значення функції.

Введемо такі позначення:

X_i — код аргументу, поданий i старшими розрядами;

Y_i — наближене значення функції $\sqrt{X_i}$, подане i старшими розрядами.

Легко помітити, що $X_0 = Y_0 = 0,\ \ X_n = X$ і $Y_n = Y$.

Надалі будуть також корисні очевидні рівності:

$$X_i = X_{i-1} + x_i 2^{-i}; \qquad \text{(Б1-5.1)}$$

$$Y_i = Y_{i-1} + y_i 2^{-i}. \qquad \text{(Б1-5.2)}$$

Якщо на кожному i-му кроці (де $i = \overline{1, n}$) обчислення цифру y_i обирати таким чином, що буде виконуватись умова

$$Y_i^2 \le X_i < (Y_i + 2^{-i})^2, \qquad \text{(Б1-5.3)}$$

то після n-го кроку буде отримане значення функції $Y_n = Y$, оскільки $X_n = X$. З виразу (Б1-5.3) випливає, що

$$0 \le X_i - Y_i^2 < Y_i 2^{-i+1} + 2^{-2i}. \qquad \text{(Б1-5.4)}$$

Далі введемо позначення:

$$R_i = (X_i - Y_i^2)2^{i-1}, \qquad \text{(Б1-5.5)}$$

та подамо вираз (Б1-5.4) у вигляді:

$$0 \le R_i < Y_i + 2^{-i-1}, \qquad \text{(Б1-5.6)}$$

З чого виходить, що перед початком обчислення $R_0 = Y_0 = 0$ отримана умова завжди виконується.

Припустимо, що вираз (Б1-5.6) виконується на (i – 1)-му кроці, тобто:

$$0 \le R_{i-1} < Y_{i-1} + 2^{-i}. \qquad \text{(Б1-5.7)}$$

Покажемо, що в цьому випадку вибором відповідного значення цифри y_i з множини значень {0, 1, 2} завжди можна досягти виконання умови (Б1-5.6) на i-му кроці. Беручи до уваги вираз (Б1-5.1) і (Б1-5.2), подамо значення (Б1-5.5) у вигляді:

$$\begin{aligned} R_i &= [X_{i-1} + x_i 2^{-i} - (Y_{i-1} + y_i 2^{-i})^2]2^{i-1} = \\ &= 2(X_{i-1} - Y_{i-1}^2)2^{i-2} + (x_i 2^{-i} - Y_{i-1} y_i 2^{-i+1} - y_i^2 2^{-2i})2^{i-1} = \\ &= 2R_{i-1} + x_i 2^{-1} - Y_{i-1} y_i - y_i^2 2^{-i-1}. \end{aligned} \qquad \text{(Б1-5.8)}$$

З урахуванням виразів (Б1-5.2) і (Б1-5.8) вираз (Б1-5.6) можна привести до вигляду:

$$\begin{aligned} Y_{i-1} y_i 2^{-1} + y_i^2 2^{-i-2} + y_i^2 2^{-i-2} - x_i 2^{-2} \le R_{i-1} < Y_{i-1} y_i 2^{-1} + \\ + y_i^2 2^{-i-2} + Y_{i-1} 2^{-1} + y_i 2^{-i-1} + 2^{-i-2} - x_i 2^{-2}. \end{aligned} \qquad \text{(Б1-5.9)}$$

Підставляючи в вираз (Б1-5.9) послідовно y_i = 0, 1 і 2, одержимо відповідно:

$$-x_i 2^{-2} \le R_{i-1} < Y_{i-1} 2^{-1} + 2^{-i-2} - x_i 2^{-2}; \qquad \text{(Б1-5.10)}$$

$$Y_{i-1} 2^{-1} + 2^{-i-2} - x_i 2^{-2} \le R_{i-1} < Y_{i-1} + 2^{-i} - x_i 2^{-2}; \qquad \text{(Б1-5.11)}$$

$$Y_{i-1} + 2^{-i} - x_i 2^{-2} \le R_{i-1} < 3Y_{i-1} 2^{-1} + 2^{-i+1} + 2^{-i-2} - x_i 2^{-2}. \qquad \text{(Б1-5.12)}$$

Ліве співвідношення в (Б1-5.10) завжди виконується, що очевидно з виразу (Б1-5.7), отже, $y_i \ge 0$. Завжди виконується й праве співвідношення у виразі (Б1-5.12), тобто $y_i \le 2$. Справді, для виконання зазначеного співвідношення досить, щоб виконувалася нерівність

$$Y_{i-1} + 2^{-i} < 3Y_{i-1} 2^{-1} + 2^{-i+1} + 2^{-i-2} - x_i 2^{-2}, \qquad \text{(Б1-5.13)}$$

складена з урахуванням виразу (Б1-5.7). Підставляючи у вираз (Б1-5.13) значення $x_i = 1$, після виконання перетворень одержимо:

$$Y_{i-1} > 2^{-1} - 2^{-i+1} - 2^{-i-1}. \qquad \text{(Б1-5.14)}$$

За $i = 1$ ліва частина виразу (Б1-5.14) дорівнює $Y_0 = 0$, а права $-\frac{3}{4}$. Окрім того, оскільки $Y > 2^{-1}$, то $Y_1 = 2^{-1}$. Тоді для будь-якого ($i > 1$) справедливо $Y_i \geq 2^{-1}$. Отже, нерівність (Б1-5.14) завжди виконується. Це свідчить про те, що умова (Б1-5.6) виконується на i-му кроці, якщо $y_i \in \{0, 1, 2\}$.

Позначивши праві частини виразів (Б1-5.10) і (Б1-5.11) відповідно як N_1 та N_2, одержимо остаточні умови для вибору y_i:

$$y_i = \begin{cases} 0, \text{ якщо } R_{i-1} < N \\ 1, \text{ якщо } N_1 \leq R_{i-1} < N_2 \\ 2, \text{ якщо } R_{i-1} \geq N_2. \end{cases} \qquad \text{(Б1-5.15)}$$

Таким чином, процес обчислення квадратного кореня може бути зведений до послідовного виконання n кроків. Перед початком обчислення $X_0 = Y_0 = R_0 = 0$. На кожному i-му кроці $(i = \overline{1, n})$ цифру y_i визначають відповідно до виразу (Б1-5.15), а значення Y_i та R_i — за формулами (Б1-5.2) і (Б1-5.8). При цьому формула (Б1-5.2) забезпечує автоматичне перетворення коду результату з надлишкової системи числення в ненадлишкову, якщо операцію додавання виконувати в ненадлишковій системі числення.

Процес обчислення квадратного кореня за запропонованим алгоритмом закінчується безпосередньо після введення останнього розряду коду аргументу. Це дозволяє зменшити час обчислення в тих випадках, коли швидкість надходження чергових розрядів аргументу визначається зовнішніми стосовно операційного блоку факторами.

Б1-5.2. Метод обчислення зворотної величини

Розглянемо метод обчислення функції $Z = \frac{1}{x}$, який дозволяє процеси вводу аргументу та обчислювання функції виконувати у режимі суміщення.

Будемо вважати, що код аргументу X вводиться послідовним кодом зі старших розрядів, а для подання значень аргументу й функції використається надлишкова симетрична двійкова позиційна система

числення із цифрами $\{\bar{1},\ 0,\ 1\}$, що належить до модифікованих квазіканонічних систем числення. Крім того, будемо вважати, що значення аргументу X є дробовим числом, тобто

$$X = \sum_{i=1}^{n} x_i 2^{-i},\ \text{де}\ \ x_i \in \{\bar{1},\ 0,\ 1\}.$$

Якщо розглядати функцію вигляду $Y = 2^{-p} f(X),$ де p — додатне ціле, то за певного значення p значення функції $3 \le -2 - 2\dfrac{Y_{(i-1)\max}}{X_{i\min}}$ також може бути дробовим числом вигляду

$$Y = \sum_{i=1}^{n} y_i 2^{-i},\ \text{де}\ \ y \in \{\bar{1},\ 0,\ 1\}.$$

Введемо позначення:

X_i — код аргументу X, що містить лише i старших розрядів;

Y_i — код значення $2^{-p} f(X_i)$ що містить тільки i старших розрядів.

Очевидно, що $X_n = X,\ Y_n = Y$ та крім того,

$$X_i = X_{i-1} + x_i 2^{-i},\ Y_i = Y_{i-1} + y_i 2^{-i}. \qquad \text{(Б1-5.16)}$$

Розглянемо метод обчислення функції $Y = 2^{-p}\dfrac{1}{X}$. Будемо вважати, що $2^{-1} \le X < 1,$ тоді $2^{-p} < Y \le 2^{-p+1}$.

Якщо на i-му кроці цифру y_i обирати таким чином, щоб виконувалася умова

$$Y_i - 2^{-(i+1)} \le 2^{-p}\frac{1}{X_i} < Y_i + 2^{-(i+1)}, \qquad \text{(Б1-5.17)}$$

то після виконання n кроків буде отримане значення $Y_n = Y$ із точністю до $2^{-(n+1)}$. Дійсно, за ($i = n$) вираз (Б1-5.17) є умовою симетричного округлення значення Y_n.

Ведемо позначення:

$$R_i = (2^{-p} - Y_i X_i)2^{i+1}, \qquad \text{(Б1-5.18)}$$

тоді умову (Б1-5.17) можна подати як

$$-X_i \le R_i < X_i. \qquad \text{(Б1-5.19)}$$

Припустимо, що на (i–1)-му кроці мають місце нерівності

$$-X_{i-1} \le R_{i-1} < X_{i-1} \qquad \text{(Б1-5.20)}$$

та

$$Y_{i-1\max} \le 2^{-p+1}. \qquad \text{(Б1-5.21)}$$

Визначимо, за якого значення p у цьому випадку виконуватиметься умова (Б1-5.19). Відповідно до виразів (Б1-5.16) і (Б1-5.18) для визначення R_i можна записати:

$$R_i = [2^{-p} - (Y_{i-1} + y_i 2^{-i})(X_{i-1} + x_i 2^{-i})]2^{i+1} =$$
$$= (2^{-p} - Y_{i-1}X_{i-1})2^{i+1} - (Y_{i-1}x_i 2^{-i} + X_{i-1}y_i 2^{-i} + x_i y_i 2^{-2i})2^{i+1} =$$
$$= 2R_{i-1} - 2Y_{i-1}x_i - 2X_i y_i. \qquad \text{(Б1-5.22)}$$

Позначивши

$$N_i = 2R_{i-1} - 2Y_{i-1}x_i, \qquad \text{(Б1-5.23)}$$

подамо вираз (Б1-5.22) у вигляді

$$R_i = N_i - 2X_i y_i. \qquad \text{(Б1-5.24)}$$

Тоді вираз (Б1-5.19) з урахуванням виразу (Б1-5.24) можна подати у вигляді

$$(2y_i - 1)X_i \le N_i < (2y_i + 1)X_i. \qquad \text{(Б1-5.25)}$$

Оскільки $X_i > 0$ за $i \ge 1$, то ліве співвідношення виразу (Б1-5.25) можна подати у вигляді $2y_i - 1 \le \dfrac{N_i}{X_i}$. Для виконання цієї нерівності досить, щоб виконувалася умова $3 \le -2 - 2\dfrac{Y_{(i-1)\max}}{X_{i\min}}$, отримана з урахуванням виразів (Б1-5.20) і (Б1-5.23). Підставивши в цю умову $Y_{(i-1)\max} = 2^{-p+1}$ і $X_{i\min} = 2^{-1}$, можна встановити, що вона виконується за $p = 3$. Можна переконатися також, що за такого значення p виконується праве співвідношення у виразі (Б1-5.25) за $y_i = 1$ та $N_i = N_{i\max}$, а отже, виконується й умова (Б1-5.19) на i-му кроці.

Підставляючи у виразах (Б1-5.25) значення $y \in \{\bar{1}, 0, 1\}$, одержуємо правило вибору цифри функції на i-му кроці:

$$y_i = \begin{cases} \bar{1}, & \text{якщо } N_i < -X_i \\ 0, & \text{якщо } -X_i \le N_i < X_i \\ 1, & \text{якщо } X_i \le N_i \end{cases}. \qquad \text{(Б1-5.26)}$$

Виконання одного кроку обчислення зводиться до такого. За формулами (Б1-5.16) і (Б1-5.23) одержуємо значення X_i та N_i. Відповідно до виразу (Б1-5.26) визначаємо значення y_i, а необхідні для виконання наступного кроку значення R_i та Y_i знаходимо відповідно за формулами (Б1-5.24) і (Б1-5.16).

Врахувавши що $X_0 = Y_0 = 0$, за виразом (Б1-5.18) за p = 3 одержимо $R_0 = 2^{-2}$. Виконавши два кроки обчислення функції з початковими значеннями X_0, Y_0 та R_0 для всіх можливих комбінацій значень розрядів коду X_2, можна показати, що отримані в цьому випадку значення X_2, Y_2 та R_2 задовольняють умовам (Б1-5.20) та (Б1-5.21), виконання яких передбачалося раніше. Варто враховувати, що можливими є тільки комбінації 10 й 11 значень відповідно x_1 и x_2, тому що в іншому випадку не буде виконуватиметься умова $2^{-1} \le X$. Таким чином, процес обчислення функції є збіжним, якщо $X_0 = Y_0 = 0$, $R_0 = 2^{-2}$ та p = 3.

Значення X_i та Y_i можна формувати безпосередньо в ненадлишковому коді за формулами (Б1-5.16). У цьому випадку всі проміжні операції на кожному кроці, які зводяться до виконання алгебраїчного додавання й зсуву, також можна виконувати в ненадлишковому коді.

Б1-6. Операції з числами у форматі з плаваючою комою

Б1-6.1. Додавання чисел із плаваючою комою

Суму двох чисел $X = 2^{P_X} M_X$ і $Y = 2^{P_Y} M_Y$, поданих у форматі із плаваючою комою, можна записати у вигляді

$$2^{P_X} M_X + 2^{P_Y} M_Y = 2^{P_Z} M_Z.$$

У загальному випадку дійсні ваги однакових розрядів мантиси різних чисел із урахуванням порядків чисел можуть відрізнятися. Тому для додавання чисел із плаваючою комою необхідно привести їх до загального порядку $P_{\text{заг}}$, яким зручно обрати більший порядок з двох доданків

$$P_{\text{çàã}} = \max(P_X, P_Y).$$

При цьому зменшенню за рахунок зсуву праворуч підлягає мантиса числа з меншим порядком. У протилежному випадку виникає переповнення розрядної сітки мантиси числа, що перетворюється. Після цього суму двох чисел можна подати у вигляді

$$2^{P_{\text{çàã}}} M_X + 2^{P_{\text{çàã}}} M'_Y = 2^{P_{\text{çàã}}} (M_X + M'_Y),$$

де за M'_Y прийнято перетворену мантису числа з меншим порядком.

Можна визначити такі етапи додавання чисел із плаваючою комою. Будемо вважати, що мантиси чисел зберігаються у пам'яті у прямих кодах.

1. *Зрівняння порядків.* На цьому етапі виконується віднімання порядків доданків з метою визначення доданку з меншим порядком. Отримана різність порядків і її знак вказують, порядок якого доданку менше і на скільки розрядів необхідно зсунути його мантису праворуч. Далі як загальний обирається більший порядок, а мантиса числа з меншим порядком зсувається вправо на різність порядків. При цьому молодші розряди мантиси можуть втрачатися, вслід чого у доданок вноситься похибка.

2. *Додавання мантис.* Виконується додавання мантис доданків за правилами складання чисел із знаками в модифікованих зворотних або доповнювальних кодах.

3. *Нормалізація результату.* Мантиса результату може виявитися відразу нормалізованою або подана з порушенням нормалізації вліво чи вправо. В двох останніх випадках проводиться нормалізація

результату, тобто приведення мантиси до вигляду, що відповідає виразу (Б1-1.9).

Якщо за додавання мантис виникає переповнення розрядної сітки, то таке переповнення говорить про *порушення нормалізації вліво*. Ознакою порушення нормалізації вліво є розбіжність у знакових розрядах модифікованих кодів чисел. Для приведення результату до нормалізованої форми необхідно зсунути мантису результату вправо на один розряд, а загальний порядок збільшити на одиницю.

Нормалізовані мантиси додатних чисел у розряді з вагою 2^{-1} завжди мають одиницю. Нормалізовані мантиси від'ємних чисел, що подані зворотнім або доповнювальним кодом, мають у цьому розряді нуль. Збіжність цифр у знаковому і старшому розряді мантиси у додатних і від'ємних чисел говорить про *порушення нормалізації вправо*. Для приведення до нормалізованого вигляду мантиса отриманого результату зсувається вліво до появи одиниці або нуля (залежно від використовуваних кодів) у старшому розряді, а із загального порядку результату віднімається число одиниць, що дорівнює кількості зсувів мантиси.

Слід зазначити, якщо у результаті додавання отриманий нульовий результат, процес зсуву мантиси може виявитися нескінченним, тому кількість зсувів обмежується кількістю розрядів мантиси і результат операції подається *машинним нулем*.

4. *Формування результату.* На останньому етапі мантиса результату, отримана у доповнювальному або зворотному коді, подається у прямому коді. Далі до мантиси приписується загальний порядок доданків.

Унаслідок зсуву мантиси на етапі зрівняння порядків частина розрядів виходить з меж розрядної сітки і втрачається, тому після підсумовування може бути зроблено округлення результату, для цього до розряду, що вийшов із розрядної сітки останнім, додається одиниця з розповсюдженням переносив у старші розряди, що представляють результат у межах розрядної сітки. Округлення може визвати порушення нормалізації вліво, тобто повторну нормалізацію.

Приклад

Завдання. Виконати додавання чисел $Z = X + Y$, подані у формі із плаваючою комою.

$$X = 0\ 1000\ 0\ 10111,$$

$$Y = 0\ 1010\ 0\ 11100.$$

Виконання завдання

$$P_{(X)} = 0\ ,\ 1000 \qquad M_{(X)} = 0\ ,\ 10111$$
$$P_{(Y)} = 0\ ,\ 1010 \qquad M_{(Y)} = 0\ ,\ 11100$$

1. Зрівняння порядків.

Виконаємо віднімання порядків доданків у вигляді $\Delta = P_{(X)} + {} + (-P_{(Y)})$ з використанням доповнювального коду.

$$\begin{array}{rl} P_{(Y)\,[\text{ДК}]} & +\,1\ ,\ 0110 \\ P_{(X)} & \quad 0\ ,\ 1000 \\ \hline (P_{(X)} - P_{(Y)})_{[\text{ДК}]} & \quad 1\ ,\ 1110 \end{array}$$

$$(P_{(X)} - P_{(Y)})_{[\text{ПК}]} \quad 1\ ,\ 0010.$$

Отримана в результаті від'ємна різність порядків ($\Delta = -2_{(10)}$) вказує на те, що порядок числа *Y* більший за порядок числа *X*. Тому порядок цього числа обираємо як загальний порядок:

$$P_{(\text{заг})} = 0\ ,\ 1010.$$

Приводимо порядок числа *X* до загального порядку, для чого збільшуємо його на різність порядків і зсуваємо мантису числа *X* на два розряди вправо:

$$P_{(\text{заг})} = 0\ ,\ 1010 \qquad M_{(X)} = 0\ ,\ 0010111.$$

2. Додавання мантис:

Мантиси є додатними числами, тому підсумовування виконуємо за правилами додавання звичайних додатних чисел, причому застосовуємо модифікований код подання чисел:

$$\begin{array}{rl} M_{(X)} & +\,00\ ,\ 0010111 \\ M_{(Y)} & \quad 00\ ,\ 1110000 \\ \hline M_{(Z)} & \quad 01\ ,\ 0000111. \end{array}$$

3. Нормалізація результату.

Отримана мантиса результату $M_{(Z)}$ не є нормалізованою, оскільки присутня розбіжність у знакових розрядах. Це говорить про переповнення розрядної сітки, тобто відбулося порушення нормалізації вліво.

Для приведення результату до нормалізованої форми зсунемо мантису результату на один розряд вправо, збільшимо загальний порядок на одиницю.

$$P'_{(\text{заг})} = P_{(\text{заг})} + 1$$

$$P'_{(\text{заг})} = 0\ ,\ 1011 \qquad M'_{(Z)} = 0\ ,\ 10000111$$

4. Формування результату.

Округляємо мантису результату, для чого до розряду, що вийшов із розрядної сітки останнім, додається одиниця:

$$\begin{array}{rl} M'_{(Z)} & 0\ ,\ 10000\,|\,111 \\ + & \qquad\qquad\quad\ \ |\,1 \\ \hline M_{(Z)} & 0\ ,\ 10001\,| \end{array}.$$

До отриманої мантиси результату приписуємо загальний порядок:

$$P'_{(\text{заг})} = P_{(Z)};$$

$$P_{(Z)} = 0\ ,\ 1011 \qquad M_{(Z)} = 0\ ,\ 10001$$

Відповідь: $Z = 0\ 1011\ 0\ 10001$.

Б1-6.2. Множення чисел із плаваючою комою

Множення двох чисел $X = 2^{P_X} M_X$ і $Y = 2^{P_Y} M_Y$, заданих у форматі із плаваючою комою, можна подати у вигляді

$$2^{P_Z} M_Z = 2^{P_X} M_X \cdot 2^{P_Y} M_Y = 2^{(P_X + P_Y)} \cdot (M_X \cdot M_Y).$$

Під час виконання операції множення мантис можна отримати результат із порушенням нормалізації вправо. Справді, якщо мантиси множників подані у нормалізованій формі

$$2^{-1} \le M_X, M_Y < 1,$$

то результат може знаходитися у межах

$$2^{-2} \le M_Z < 1.$$

Етапи множення чисел із плаваючою комою:

1. Одержання порядку результату у вигляді суму порядків множників

$$P_Z = P_X + P_Y.$$

2. Знаходження мантиси результату

$$M_Z = M_X \cdot M_Y.$$

3. Нормалізація результату, тобто приведення мантиси результату до вигляду

$$2^{-1} \le M_Z < 1.$$

Приклад

Завдання. Подати числа у формі з плаваючою комою и знайти добуток $Z = X \cdot Y$, якщо:

$$X = 10{,}101,$$

$$Y = 0{,}1011.$$

Виконання завдання

Приведемо задані числа до форми:

$$X = 2^{P_X} M_X = 2^2 \cdot 0{,}10101,$$

$$Y = 2^{P_Y} M_Y = 2^0 \cdot 0{,}1011.$$

Отримали порядки і мантиси:

$$P_{(X)} = 0\ ,\ 10 \qquad M_{(X)} = 0\ ,\ 10101;$$
$$P_{(Y)} = 0\ ,\ 00 \qquad M_{(Y)} = 0\ ,\ 10110.$$

Визначимо порядок результату, виконавши додавання порядків множників $P_Z = P_X + P_Y = 2 + 0 = 2$. Після множення мантис отримаємо:

$$P_Z = 0\ ,\ 10 \qquad M_Z = 0\ ,\ 0111001110.$$

Мантиса результату не є нормалізованою. Для виконання нормалізації мантису результату зсуваємо ліворуч на один розряд і виконуємо корекцію порядку $P_Z = P_Z - 1 = 1$. Отримаємо:

$$P_{(Z)} = 0\ ,\ 01 \qquad M_Z = 0\ ,\ 11100111.$$

Після округлення мантиси отримаємо остаточний результат.

Відповідь: $Z = 0\ 01\ 0\ 11101$.

Б1-6.3. Ділення чисел із плаваючою комою

Результат ділення двох чисел $X = 2^{P_X} M_X$ і $Y = 2^{P_Y} M_Y$, заданих у форматі із плаваючою комою, можна записати у вигляді

$$2^{P_Z} M_Z = \frac{2^{P_X} M_X}{2^{P_Y} M_Y} = 2^{(P_X - P_Y)} \cdot \frac{M_X}{M_Y}$$

Ділення мантис має виконуватися за виконання умови

$$M_X < M_Y, \qquad \text{(Б1-6.1)}$$

яка не завжди виконується за подання мантис у нормалізованій формі. Тому перед началом ділення мантису діленого завжди зсувають вправо, чим забезпечують зменшення її у два рази. Відповідно до порядку діленого додається одиниця, тобто

$$2^{P_X} M_X = 2^{P_X + 1} \cdot M_X \cdot 2^{-1}. \qquad \text{(Б1-6.2)}$$

Після цього маємо

$$2^{-2} \le M_X < 2^{-1},$$
$$2^{-1} \le M_Y < 1,$$
$$2^{-2} \le M_Z < 1.$$

Етапи ділення чисел із плаваючою комою:

1. Одержання порядку результату із урахуванням виконання умови (Б1-6.1) шляхом віднімання

$$P_Z = P_X - P_Y.$$

2. Одержання мантиси результату. На цьому етапі виконується ділення мантис:

$$M_Z = \frac{M_X}{M_Y}.$$

3. Нормалізація результату. Виконується аналогічно нормалізації при множенні чисел із плаваючою комою.

Приклад

Завдання. Виконати ділення чисел $Z = X / Y$, що подані у формі із плаваючою комою.

$$X = 0\ 10\ 0\ 1011,$$
$$Y = 0\ 01\ 0\ 1001.$$

Виконання завдання

Маємо ділене і дільник, задані у формі:

$$P_X = 0,10 \qquad M_X = 0,1011,$$
$$P_Y = 0,01 \qquad M_Y = 0,1001.$$

Застосуємо перетворення (Б1-6.2) для забезпечення умови (Б1-6.1). Отримаємо змінену мантису і порядок діленого X:

$$P_X = 0,11 \qquad M_{(X)} = 0,01011.$$

Визначимо порядок результату:

$$P_Z = P_X - P_Y = 3 - 1 = 2, \text{ тобто}$$

$$P_Z = 0,10.$$

Визначимо мантису результату діленням мантис і одержуємо результат:

$$M_Z = 0,11001.$$

Отриману в результаті мантису частки округляємо і усікаємо до розмірів розрядної сітки:

$$M_Z = 0,1101.$$

Мантиса результату є нормалізованою, тому сформуємо результат обчислення:

$$P_Z = 0,10 \qquad M_Z = 0,1101.$$

Відповідь: $Z = 0\ 01\ 0\ 1101$.

Б1-7. Синтез операційних пристроїв з розподіленою логікою

В операційних пристроях (ОП) *з розподіленою логікою* (інакше їх називають операційні пристрої *з закріпленими мікрооперація*ми), кожний вузол (регістр, лічильник) використовує власні логічні ланцюги для виконання мікрооперацій. Структура ОП з розподіленою логікою залежить від операцій, які вони реалізують.

Функціонально ОП поділяються на дві частини:

— управляючий автомат (УА), що забезпечує формування всіх управляючих сигналів;

— операційний автомат (ОА), що забезпечує перетворення інформації за заданим мікроалгоритмом.

Наведемо етапи розроблення мікроалгоритмів для ОП з розподіленою логікою.

1. Для кожної операції будується операційна схема, за якою складається змістовний мікроалгоритм. Рекомендується обирати мікроалгоритми (МА), такі що краще сполучаються під час виконання різних операцій, тобто вимагають однакового напрямку зсуву в регістрах, однакових джерел для операндів суматорів тощо.

2. Обирається розрядність регістрів, лічильників. Виконується логічне моделювання роботи ОП, наприклад, із застосуванням діаграми стану регістрів під час виконання мікроалгоритму з критичним значенням операндів.

3. Розробляється функціональна схема ОП із вказанням управляючих сигналів для кожного вузла пристрою.

4. Складається закодований структурний мікроалгоритм виконання заданої операції.

5. Виконується синтез пристрою управління (управляючого автомата).

6. Будується логічна схема в заданому елементному базисі.

Для більш детального розгляду етапів розроблення ОП розглянемо декілька прикладів.

Розроблення операційного пристрою з розподіленою логікою для обчислення функції

$$D = 2A^2 + 0{,}5B.$$

Операційна схема для виконання зазначеної операції зображена на рис. Б1-7.1, де *RG*1, *RG*2 — регістри, *SM* — суматор, *CT* — лічильник. Будемо вважати, що операнди і результат операції додатні і мають тільки дробову частину, що досягається масштабуванням операндів перед початком операції. Змістовний мікроалгоритм зображен на рис Б1-2.5, *а*. В операторних вершинах розміщені мікрооперації *Yi*. При цьому в кожній операторній вершині розміщуються

мікрооперації, що виконуються водночас, тобто в одному такті. У логічних вершинах розміщено логічні умови *Xi*.

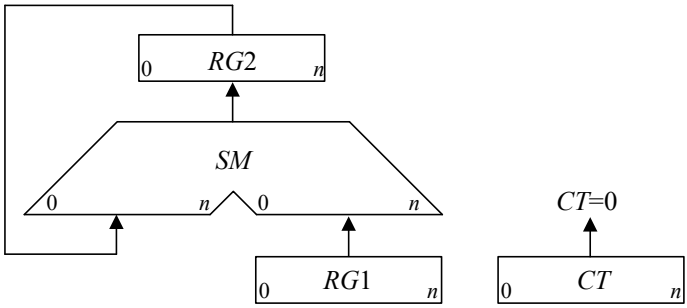


Рис. Б1-7.1. Операційна схема пристрою
для обчислення функції

У вихідному стані операнд *B* записаний в регістр *RG*2, а операнд *A* — в регістр *RG*1 і в лічильник *CT*. У першому такті шляхом зсуву вліво операнда *A* в регістрі *RG*1 здійснюється подвоєння цього операнда і шляхом зсуву вправо операнда *B* в *RG*2 здійснюється ділення операнда *B* на 2. Далі до вмісту *RG*2 *A* раз додається слово, записане в *RG*1. Після кожного додатку вміст *CT* зменшується на 1. Обчислення закінчуються за виконання умови *CT.* = 0. Відповідний цьому сигнал можна одержати, наприклад, дешифруванням нульового стану *C*. Результат операції формується в регістрі *RG*2.

На основі операційної схеми і змістовного мікроалгоритму розробляємо функціональну схему пристрою, яка зображена на рис. Б1-7.2. На функціональній схемі указані способи формування умов і управляючі сигнали для усіх вузлів пристрою обчислення функції:

W — запис інформації в регістри;
SR — зсув вправо вмісту регістрів;
SL — зсув вліво вмісту регістрів;
CLR— обнулення;
dec — декремент лічильника;
inc — інкремент лічильника;
d — сигнал, що дозволяє одержати доповнювальний код числа під час реалізації операції віднімання, подається на управляючий вхід пристрою інвертування та вхід *CI* суматора (рис. Б1-7.7).

Для виконання мікрооперації на регістрі необхідно подати одиничний сигнал на відповідний управляючий вхід. На всі інші управляючі входи цього регістру має подаватися нульовий сигнал.

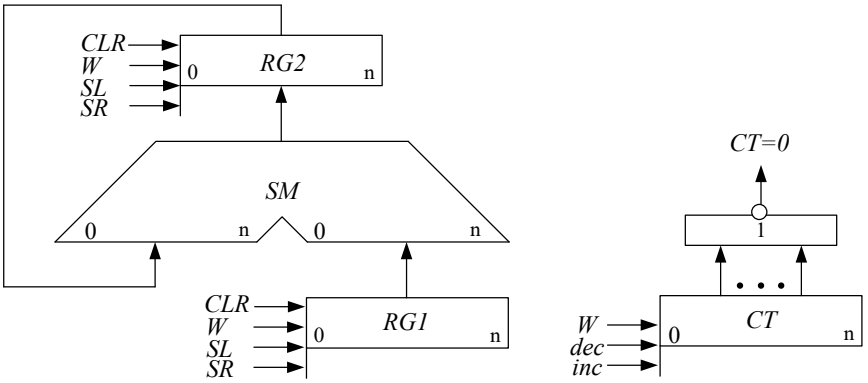


Рис. Б1-7.2. Функціональна схема операційного автомата

Визначимо сукупність управляючих сигналів необхідних для виконання кожної мікрооперації. Мікрооперації *Y*1, *Y*2, *Y*3 та *Y*4 кодуються відповідно управляючими сигналами *SL*1, *SR*2, *W*2, та *dec*, що подаються на управляючі входи вузлів пристрою відповідно до функціональної схеми (Б1-7.2).

Необхідна тривалість управляючих сигналів визначається з урахуванням затримок в елементах операційного пристрою. Період *t* тактуючих сигналів зазвичай обирається або рівним максимальній тривалості управляючих сигналів, або мінімальнім. При цьому величина *t* повинна бути не менше часу переключення автомата з одного стану в інший. У першому випадку всі мікрооперації виконуються в синхронному режимі (за однаковий проміжок часу), а в другому — в асинхронному, причому тривалість управляючих сигналів кратна величині *t*.

Асинхронний режим можна забезпечити, наприклад, введенням у мікроалгоритм додаткових операторних вершин з управляючими сигналами, тривалості яких перевищують *t*. Будемо вважати, що з урахуванням швидкодії елементів для розглянутого приклада управляючі сигнали повинні мати тривалість *t*.

Для одержання закодованого мікроалгоритму узагальнимо результати перших двох етапів в табл. Б1-7.1, де описи мікрооперацій подамо відповідними управляючими сигналами із указанням їх тривалості, описи логічних умов в змістовному мікроалгоритмі подамо їх позначеннями (табл. Б1-7.2). Відповідно до отриманих таблиць виконаємо заміну мікрооперацій і логічних умов у змістовному мікроалгоритмі. Отримаємо закодований мікроалгоритм, що приведений на рис. Б1.7-3.

Таблиця Б1-7.1

ТАБЛИЦЯ КОДУВАННЯ МІКРООПЕРАЦІЙ

Мікрооперації	Управляючі сигнали	Тривалість управляючих сигналів
$Y1 : RG1 := l[RG1].0$	$SL1$	t
$Y2 : RG2 := 0.r[RG2]$	$SR2$	t
$Y3 : RG2 := RG2 + RG1$	$W1$	t
$Y4 : CT := CT - 1$	dec	t

Таблиця Б1-7.2

ТАБЛИЦЯ КОДУВАННЯ ЛОГІЧНИХ УМОВ

Логічні умови	Позначення логічних умов
$X1$: Пуск = 1	x_1
$X2 : CT = 1$	x_2

Управляючі сигнали $SL1$ і $SR2$ генеруються управляючим автоматом в один і той самий такт автоматного часу, отже, можуть бути замінені одним управляючим сигналом y_1, аналогічним чином сигнали $W1$, *inc* замінимо сигналом y_2. Отримаємо спрощений закодований мікроалгоритм, зображений на рис. Б1.7-4.

Закодований мікроалгоритм (рис. Б1.7-4) є вихідним для побудови управляючого автомата.

Будемо вважати, що управляючий автомат необхідно побудувати у вигляді автомата Мілі. Виконаємо розмітку закодованого мікроалгоритму, наведеного на рис. Б1-7.4. Далі необхідно виконати синтез автомата Мілі, етапи якого детально подані у розділі А1-3.3. У результаті синтезу буде отримана функціональна схема управляючого автомата, побудова якої у даному прикладі не розглядається. Управляючі сигнали з виходів управляючого автомату подаються до відповідних входів операційного автомату. Загальний вигляд ОП для обчислення заданої функції поданий на рис. Б1-7.5.

Якщо під час виконання обчислень необхідно виконувати мікрооперацію віднімання, то до складу операційної схеми, зображеної на рис. Б1-7.1 необхідно ввести вузол інвертування для отримання доповнювального коду операнда.

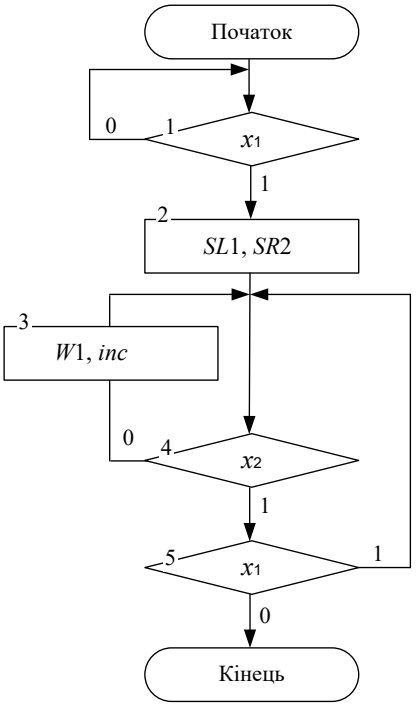


Рис. Б1-7.3. Закодований мікроалгоритм автомата

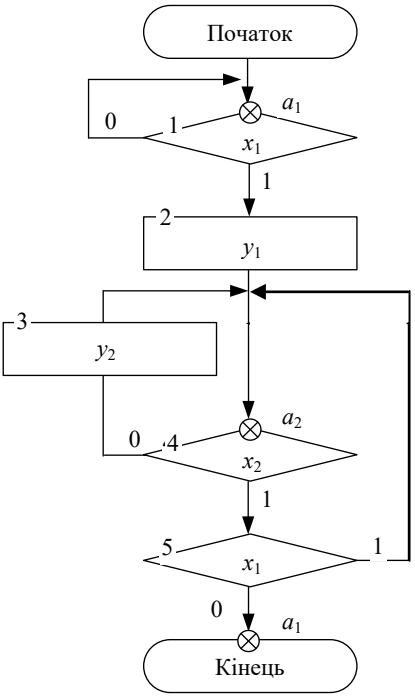


Рис. Б1-7.4. Розмічений мікроалгоритм автомата

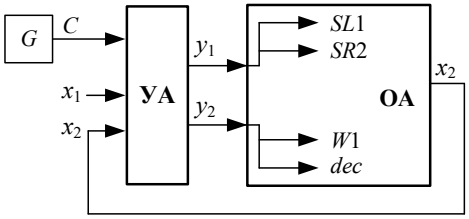


Рис. Б1-7.5. Структурна схема ОП для обчислення функції

Розроблення операційного пристрою з розподіленою логікою для обчислення функції

$$D = 0{,}5C - 2AB.$$

На підставі операційної схеми (рис. Б1-7.1) та змістовного мікроалгоритму обчислення заданої функції, зображеного на рис. Б1-7.6, розробляємо функціональну схему операційного автомата для обчислення функції (рис. Б1-7.7). Будемо вважати, що операнди і результат операції мають тільки дробову частину, що досягається масштабуванням операндів перед початком операції.

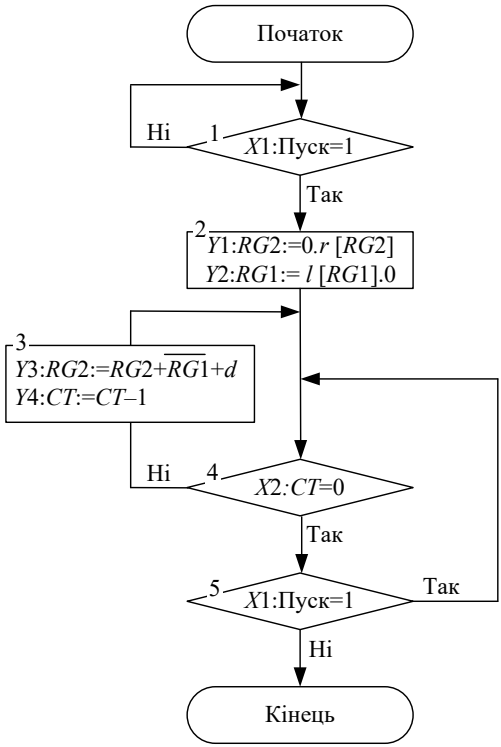


Рис. Б1-7.6. Змістовний мікроалгоритм обчислення функції

У вихідному стані операнд *C* записаний в регістр *RG*2, операнд *A* — в регістр *RG*1, операнд *B* — в лічильник *CT*. У першому такті шляхом зсуву вправо операнда в регістрі *RG*2 здійснюється ділення операнда *C* на 2 і шляхом зсуву вліво операнда *A* в регістрі *RG*1 здійснюється подвоєння цього операнда. Далі із вмісту регістра *RG*2 *B* раз віднімається слово, записане в регістрі *RG*1. Операція віднімання реалізується в доповнювальних кодах. Операнд А є додатнім числом, тому реалізація доповнювального коду не потребує вводу до-

даткових вузлів до операційного пристрою. Перетворення вмісту регістру *RG*2 у доповнювальний код, оскільки операнд С, записаний у цьому регістрі, є зворотнім числом, відбувається за допомогою вузла інвертування і додавання одиниці на вхід *CI* суматора. Для реалізації операції віднімання на вхід *d* операційного пристрою необхідно подати одиничний сигнал. Після кожного віднімання вміст *CT* зменшується на 1. Обчислення закінчуються за виконання умови *CT* = 0. Результат обчислення функції формується в регістрі *RG*2.

Подальші етапи розроблення операційного пристрою виконуються аналогічно попередньому прикладу.

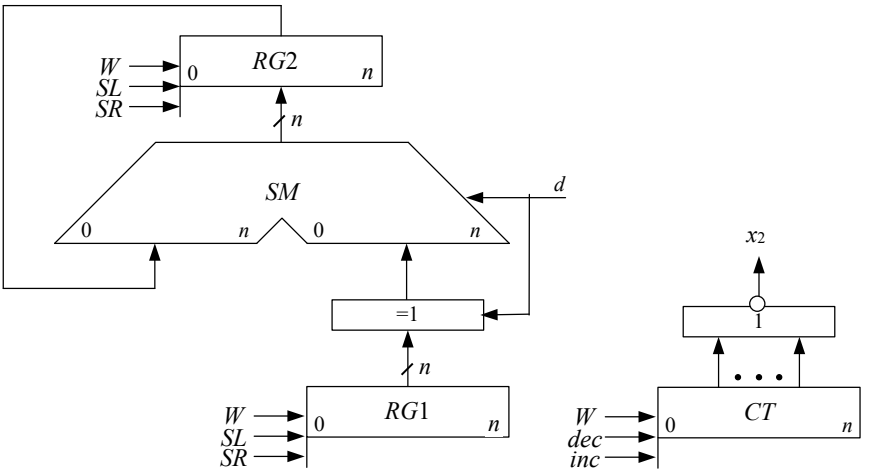


Рис. Б1-7.7. Функціональна схема операційного автомата

Розроблення операційного пристрою для виконання операції множення першим способом

Операційна схема пристрою для реалізації операції множення першим способом наведена на рис. Б1-3.1, *а*. Будемо вважати, що операнди і результат є дробовими числами, поданими у прямому коді. Оброблення знаків мантис виконується окремо і в прикладі не розглядається. Виконаємо множення $C = A \times B$ мантис:

$$B = 0{,}1111;\ \ A = 0{,}1011.$$

У початковому стані (ПС) в регістр *RG*2 записуємо множник *A*, в *RG*3 — нерухоме множене *B*, в *RG*1 — нулі, в $CT - n = 5$, що дорівнює розрядності множника. В регістрі *RG*1 накопичуються часткові

добутки. Після закінчення операції множення добуток буде розміщений у регістрах *RG*1 і *RG*2. Закінчення операції множення визначає лічильник циклів *CT*. В даному прикладі $n = 5$, тому знадобиться 5 циклів, тобто $q = 3$.

Цифрова діаграма стану регістрів під час виконання операції множення без округлення результату наведена на рис. Б1-7.8.

	*RG*1 0 → *n*		*RG*2 0 → *n*		*RG*3 0 *n*		CT
	0	0000	0	1011	0	1111	101
1	0 0 0 0	0000 1111 1111 0111	 1	 0101	0	1111	 100
2	0 0 1 0	0111 1111 0110 1011	 0	 1010	0	1111	 011
3	 0	 0101	 1	 0101	0	1111	 010
4	0 0 1 0	0101 1111 0100 1010	 0	 1010	0	1111	 001
5	 0	 0101	 0	 0101	0	1111	 000

Рис. Б1-7.8. Цифрова діаграма стану регістрів під час виконання операції множення першим способом

За операційною схемою пристрою множення першим способом (рис. Б1-3.1, *а*) та цифровою діаграмою (рис. Б1-7.8) складаємо змістовний мікроалгоритм виконання операції множення першим способом, що наведений на рис. Б1-7.9.

Далі на основі узагальненої операційної схеми одержуємо функціональну схему пристрою для виконання операції множення першим способом, в якій вказуються управляючі сигнали для всіх вузлів, і способи формування логічних умов (рис. Б1-7.10).

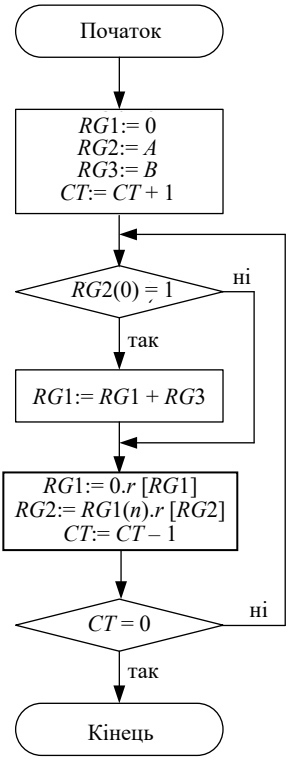


Рис. Б1-7.9. Змістовний алгоритм виконання операції множення першим способом

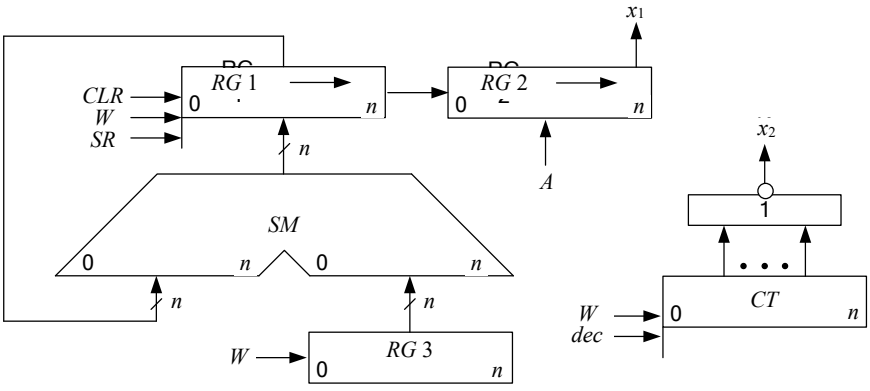


Рис. Б1-7.10. Функціональна схема пристрою для виконання операції множення першим способом

Далі слідує етап структурного синтезу управляючого автомата, вихідним для якого є отриманий змістовний мікроалгоритм виконання операції множення першим способом. Детально етапи структурного синтезу управляючих автоматів розглянуті у розділі А1-3.3.

Розроблення операційного пристрою для виконання операції ділення із зсувом дільника

Виконаємо ділення мантис $Z = Y / X$ двох додатних чисел:

$$Y = 0{,}1001;\ X = 0{,}1100;\ 0 < Y, X < 1;\ Y < X.$$

Операційна схема пристрою для реалізації операції ділення із зсувом дільника наведена на рис. Б1-4.2. Цифрова діаграма стану вузлів представлена на рис. Б1-7.11.

№ такту	*RG*3		*RG*2		*RG*1		Логічна умова
	1	1111	0	10010000	0	11000000	
1			0	10010000			*RG*1(0)=0
			1	01000000			
			1	11010000			
	1	1110			0	01100000	*RG*2(0)=1
2	1	1110	1	11010000	0	01100000	*RG*1(0)=1
			0	01100000			
			0	00110000			
	1	1101			0	00110000	*RG*2(0)=1
3	1	1101	0	00110000	0	00110000	*RG*1(0)=0
			1	11010000			
			0	00000000			
	1	1011			0	00011000	*RG*2(0)=1
4	1	1011	0	00000000	0	00011000	*RG*1(0)=0
			1	11101000			
			1	11101000			
	1	0110			0	00001100	*RG*2(0)=1
5	1	0110	1	11101000	0	00001100	*RG*1(0)=1
			0	00001100			
			1	11110100			
	0	1100			0	00000110	*RG*2(0)=0

Рис. Б1-7.11. Цифрова діаграма виконання операції ділення із зсувом дільника

Для підрахунку кількості циклів застосовані маркерні одиниці у регістрі *RG*2, що дозволяє усунути лічильник і зменшує кількість устаткування у пристрої. Змістовний алгоритм виконання операції ділення наведений на рис. Б1-7.12.

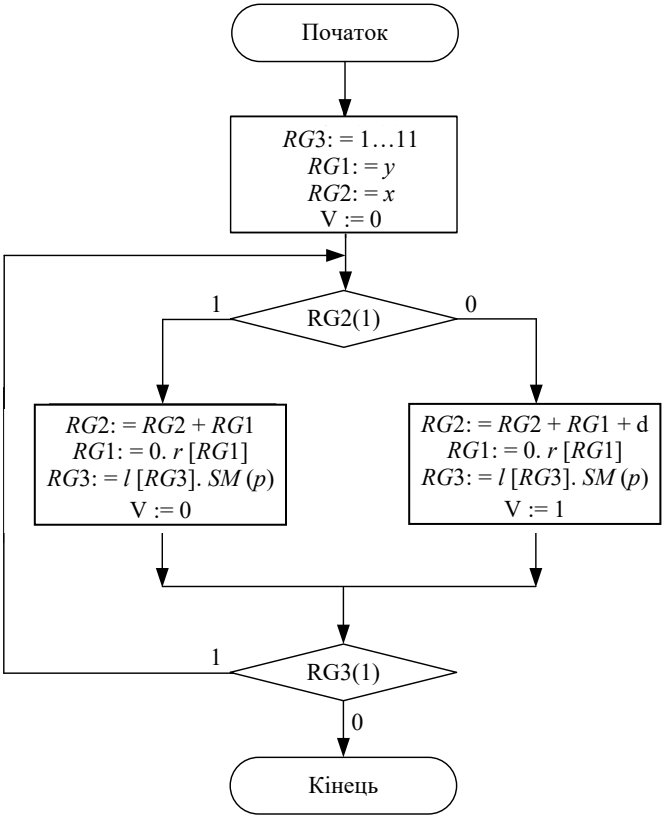


Рис. Б1-7.12. Змістовний мікроалгоритм виконання операції ділення із зсувом дільника

На рис. Б1-7.13 наведена функціональна схема пристрою для виконання операції ділення із зсувом дільника. На схемі зображені сигнали управління:

W — запис інформації в регістри;
SR — зсув вправо умісту регістрів;
SL — зсув вліво умісту регістрів;
CLR— обнулення;

dec — декримент лічильника;
inc — інкремент лічильника;
d — сигнал, що дозволяє одержати доповнювальний код числа під час реалізації операції віднімання, подається на керуючий вхід пристрою інвертування та вхід *CI* суматора;
V — сигнал керування мультиплексором;
та логічні умови:
M — маркер кінця операції;
N — знак залишку.

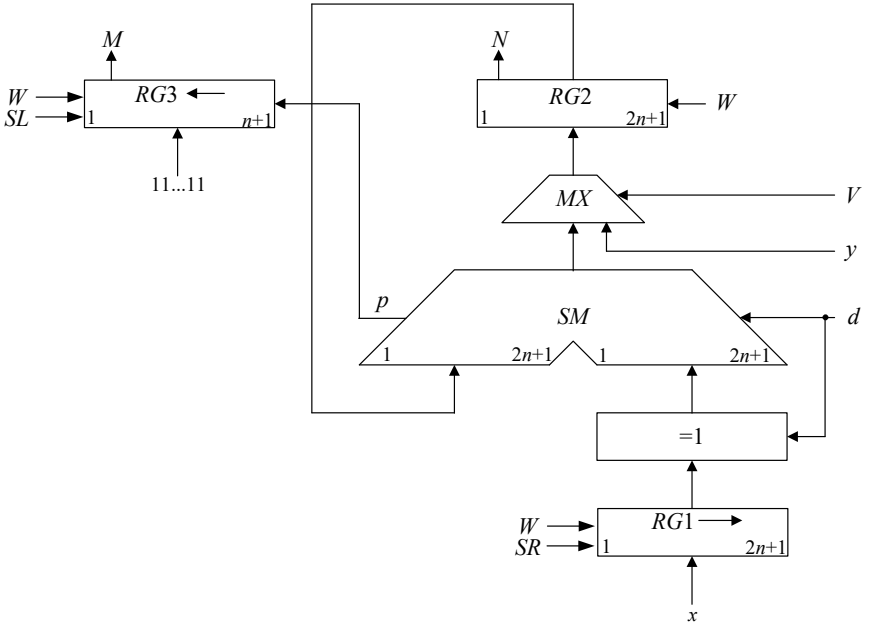


Рис. Б1-7.13. Функціональна схема операційного автомата ділення із зсувом дільника

Далі виконується етап структурного синтезу управляючого автомата, вихідним для якого є отриманий змістовний мікроалгоритм виконання операції ділення із зсувом дільника (див. розділ А1-3.3).

Б2. ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

Б2-1. Подання чисел у різних системах числення

Приклад Б2-1.1

Завдання. Перевести у двійкову систему числення числа

$$A = 398_{(10)} \text{ та } B = 0{,}647_{(10)}.$$

Виконати округлення дробової частини числа $B_{(2)}$ до 10 розрядів.

Виконання завдання

Виконаємо перевід цілого числа $A = 398_{(10)}$ у двійкову систему числення:

398	2							
398	199	2						
0	198	99	2					
	1	98	49	2				
		1	48	24	2			
			1	24	12	2		
				0	12	6	2	
					0	6	3	2
						0	2	**1**
							1	

Виконаємо перевід дробового числа $B = 0{,}647_{(10)}$ у двійкову систему числення:

0	647	
×	2	
1	294	
×	2	
0	588	
×	2	
1	176	
×	2	
0	352	
×	2	
0	704	
×	2	
1	408	
×	2	
0	816	
×	2	
1	632	
×	2	
1	264	
×	2	
0	528	
×	2	
1	056	

У результаті подання дробового числа $B_{(10)}$ у двійковій системі числення отримаємо одинадцять розрядів після коми:

$$B = 0{,}647_{(10)} \approx 0{,}1010010111_{(2)}.$$

Виконаємо округлення результату:

$$\begin{array}{r} 0{,}10100101 \\ 101 \\ \hline +1 \\ \hline 0{,}10100101 \\ 110 \end{array} \quad \begin{array}{l} (\text{о} \\ \text{кр.}) \end{array}$$

Відповідь: $A = 398_{(10)} = 110001110_{(2)}$;
$B = 0{,}647_{(10)} = 0,\ 1010010111_{(2)}$.

Приклад Б2-1.2

Завдання. Перевести у вісімкову, а потім з вісімкової у двійкову систему числення числа

$$A = 398_{(10)} \text{ та } B = 0{,}647_{(10)}.$$

Виконати округлення дробової частини числа $B_{(2)}$ до 10 розрядів.

Виконання завдання

Виконаємо перевід цілого числа $A = 398_{(10)}$ у вісімкову систему числення:

398	8	
392	49	8
6	48	**6**
	1	

Виконаємо перевід дробового числа $B = 0{,}647_{(10)}$ у вісімкову систему числення:

0	647
×	8
5	176
×	8
1	408
×	8
3	264
×	8
2	112

У результаті отримали:

$$A = 398_{(10)} = 616_{(8)},$$

$$B = 0{,}647_{(10)} = 0{,}5132_{(8)}.$$

Запишемо кожну цифру отриманих вісімкових чисел три-розрядними двійковими кодами (що відповідає вісімковій системі числення $2^3 = 8$), отримаємо:

$$A = 616_{(8)} = (110\ 001\ 110)_{(2-8)} = 110001110_{(2)};$$

$$B = 0{,}5132_{(8)} = (0{,}101\ 001\ 011\ 010)_{(2-8)} \approx 0{,}\ 101001011010_{(2)}.$$

Виконаємо округлення числа $B_{(2)}$ до десяти двійкових розрядів після коми:

$$\begin{array}{r} 0{,}101001011 \\ 010 \\ \hline +1 \\ \hline 0{,}101001011 \\ 1 \end{array} \quad \begin{array}{l} \\ \\ (\text{о} \\ \text{кр.}) \end{array}$$

Відповідь: $A = 398_{(10)} = 616_{(8)} = 110001110_{(2)}$;
$B = 0{,}647_{(10)} = 0{,}5132_{(8)} = 0{,}1010010111_{(2)}$.

Приклад Б2-1.3

Завдання. Записати у формі із плаваючою комою двійкові числа (для запису порядку використати $m = 5$ розрядів):

$$X = -\,11010{,}1101_{(2)},\ Y = 0{,}000110101110_{(2)}.$$

Виконання завдання

Подамо задані двійкові числа в нормалізованій формі, тобто з виконанням для мантис умови (В1-1.9):

$$X = -\,0{,}110101101 \cdot 2^{5},\quad Y = 0{,}110101110 \cdot 2^{-3}.$$

Порядки чисел запишемо у двійковій системі числення, для їх зображення застосуємо п'ять розрядів і один розряд для подання знаку.
Отримаємо:

$P_{(X)} = 0\ 00101$; $\quad P_{(Y)} = 1\ 00011$;
$M_{(X)} = 1\ 110101110$; $\quad M_{(Y)} = 0\ 110101110$.

Відповідь: $X = 0\ 00101\ 0\ 110101101$,
$Y = 1\ 00011\ 0\ 110101110$.

Приклад Б2-1.4

Завдання. Методом зсуву-корекції перевести у двійкову і вісімкову систему числення числа

$$A = 398_{(10)} \text{ та } B = 0{,}647_{(10)}.$$

Виконання завдання

Для переводу чисел у двійкову систему числення методом зсуву-корекції, подамо вихідні числа у коді «8421», відповідно до табл. В1-1.1.

$$A = 398_{(10)} = 0011\ 1001\ 1000_{(2-10)},$$

$$B = 0{,}647_{(10)} = 0,\ 0110\ 0100\ 0111_{(2-10)}.$$

Виконаємо перевід отриманого двійково-десяткового числа $A_{(2-10)}$ у двійкову систему числення (ПС — початковий стан):

398	0011	1001	1000		ПС
	0001	1100	1100	0	Зсув →
		+1101	+1101		Корекція (–3)
199	0001	1001	1001	0	
	0000	1100	1100	10	Зсув →
		+1101	+1101		Корекція (–3)
99	0000	1001	1001	10	
	0000	0100	1100	110	Зсув →
			+1101		Корекція (–3)
49	0000	0100	1001	110	
24	0000	0010	0100	1110	Зсув →
12	0000	0001	0010	01110	Зсув →
	0000	0000	1001	001110	Зсув →
			+1101		Корекція (–3)
6	0000	0000	0110	001110	
3	0000	0000	0011	0001110	Зсув →
1	0000	0000	0001	10001110	Зсув →
0	0000	0000	0000	**110001110**	Зсув →

Виконаємо перевід отриманого двійково-десяткового числа $B_{(2-10)}$ у двійкову систему числення:

0,647	0110	0100	0111	0,	ПС
	1100	1000	1110	0,0	Зсув ←
	+0110		+0110		Корекція (+6)
0,294	0010	1001	0100	0,1	
	0101	0010	1000	0,10	Зсув ←

		+0110			Корекція (+6)
0,588	0101 1011 +0110	1000 0001 +0110	1000 0000 +0110	0,10 0,100	 Зсув ← Корекція (+6)
0,176	0001 0010	0111 1110 +0110	0110 1100 +0110	0,101 0,1010	 Зсув ← Корекція (+6)
0,352	0011 0110	0101 1010 +0110	0010 0100	0,1010 0,10100	 Зсув ← Корекція (+6)
0,704	0111 1110 +0110	0000 0000	0100 1000	0,10100 0,101000	 Зсув ← Корекція (+6)
0,408	0100 1000	0000 0001	1000 0000 +0110	0,101001 0,1010010	 Зсув ← Корекція (+6)
0,816	1000 0000 +0110	0001 0010	0110 1100 +0110	0,1010010 0,10100101	 Зсув ← Корекція (+6)
0,632	0110 1100 +0110	0011 0110	0010 0100	0,10100101 0,101001010	 Зсув ← Корекція (+6)
0,264	0010 0100	0110 1100 +0110	0100 1000	0,101001011 0,1010010110	 Зсув ← Корекція (+6)
0,528	0101 1010 +0110	0010 0101	1000 0000 +0110	0,1010010110 0,10100101100	 Зсув ← Корекція (+6)
0,056	0000	0101	0110	0,10100101101 +1 **0,10100101110**	 Округлення

Відповідь:
$A = 398_{(10)} = 110001110_{(2)} = 110\ 001\ 110_{(2)} = 616_{(8)}$,
$B = 0{,}647_{(10)} = 0,\ 1010010111_{(2)} = 0,\ 101\ 001\ 011\ 100_{(2)} = 0{,}5134_{(8)}$.

Б2-2. Операції додавання і віднімання у машинних кодах

Приклад Б2-2.1

Завдання. Подати в прямому, зворотному та доповнювальному кодах двійкові дроби:

$$A_1 = +0{,}0110001110; \quad B_1 = +0{,}1010010111;$$

$$A_2 = -0{,}0110001110; \quad B_2 = -0{,}1010010111.$$

Виконання завдання

Подамо задані двійкові дроби у прямому, оберненому та доповнювальному кодах:

A_1	+0,0110001110	A_2	– 0,0110001110
$[A_1]_{ПК}$	0,0110001110	$[A_2]_{ПК}$	1,0110001110
$[A_1]_{ОК}$	0,0110001110	$[A_2]_{ОК}$	1,1001110001
$[A_1]_{ДК}$	0,0110001110		+ 1
		$[A_2]_{ДК}$	1,1001110010

B_1	+0,1010010111	B_2	– 0,1010010111
$[B_1]_{ПК}$	0,1010010111	$[B_2]_{ПК}$	1,1010010111
$[B_1]_{ОК}$	0,1010010111	$[B_2]_{ОК}$	1,0101101000
$[B_1]_{ДК}$	0,1010010111		+ 1
		$[B_2]_{ДК}$	1,0101101001

Приклад Б2-2.2

Завдання. Знайти суму S та різницю D двох операндів A_i та B_j. Операції машинного додавання та віднімання виконати в оберненому та доповнювальному кодах.

Знайти $S_1=A_1+B_1$ $D_1=A_1-B_1$, якщо $A_1 = 0110001110$
$S_2=A_1+B_2$ $D_2=A_1-B_2$ $A_2 = -0110001110$
$S_3=A_2+B_1$ $D_3=A_2-B_1$ $B_1 = 1010010111$
$S_4=A_2+B_2$ $D_4=A_2-B_2$ $B_2 = -1010010111$

Виконання завдання

Операнди A_i та B_j у вихідному стані розміщуються у регістрах *RGA* та *RGB* відповідно. Результат операції машинного додавання (віднімання) формується на виході комбінаційного суматора *SM*. Для відображення знаку операндів використовуємо модифікований код, для подання якого у регістрах відведено два знакових розряди.

Виконаємо операцію машинного додавання в оберненому коді.

$S_1 = A_1 + B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	00	1010010111	$[B_1]_{ок}$
SM	01	0000100101	$[S_1]_{ок}$

$S_2 = A_1 + B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	11	0101101000	$[B_2]_{ок}$
SM	11	1011110110	$[S_2]_{ок}$

Додатне переповнення

$S_3 = A_2 + B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	00	1010010111	$[B_1]_{ок}$
SM	00	0100001000	
		+1	
SM	00	0100001001	$[S_3]_{ок}$

$S_4 = A_2 + B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	11	0101101000	$[B_2]_{ок}$
SM	10	1111011001	
		+1	
SM	10	1111011010	$[S_4]_{ок}$

Від'ємне переповнення

Виконаємо операцію машинного віднімання в оберненому коді:

$D_1 = A_1 - B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	11	0101101000	$[-B_1]_{ок}$
SM	11	1011110110	$[D_1]_{ок}$

$D_2 = A_1 - B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	00	1010010111	$[-B_2]_{ок}$
SM	01	0000100101	$[D_2]_{ок}$

Додатне переповнення

$D_3 = A_2 - B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	11	0101101000	$[-B_1]_{ок}$
SM	10	1111011001	
		+1	
SM	10	1111011010	$[D_3]_{ок}$

Від'ємне переповнення

$D_4 = A_2 - B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	00	1010010111	$[-B_2]_{ок}$
SM	00	0100001000	
		+1	
SM	00	0100001001	$[D_4]_{ок}$

Виконаємо операцію машинного додавання в доповнювальному коді:

$S_1 = A_1 + B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	00	1010010111	$[B_1]_{дк}$
SM	01	0000100101	$[S_1]_{дк}$

Додатне переповнення

$S_2 = A_1 + B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	11	0101101001	$[B_2]_{дк}$
SM	11	1011110111	$[S_2]_{дк}$

$S_3 = A_2 + B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	00	1010010111	$[B_1]_{дк}$
SM	00	0100001001	$[S_3]_{дк}$

$S_4 = A_2 + B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	11	0101101001	$[B_2]_{дк}$
SM	10	1111011011	$[S_4]_{дк}$

Від'ємне переповнення

Виконаємо операцію машинного віднімання в доповнювальному коді:

$D_1 = A_1 - B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	11	0101101001	$[-B_1]_{дк}$
SM	11	1011110111	$[D_1]_{дк}$

$D_2 = A_1 - B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	00	1010010111	$[-B_2]_{дк}$
SM	01	0000100101	$[D_2]_{дк}$

Додатне переповнення

$D_3 = A_2 - B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	11	0101101001	$[-B_1]_{дк}$
SM	10	1111011011	$[D_3]_{дк}$

Від'ємне переповнення

$D_4 = A_2 - B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	00	1010010111	$[-B_2]_{дк}$
SM	00	0100001001	$[D_4]_{дк}$

Приклад Б2-2.3

Завдання. Одержати суму чисел A і B із плаваючою комою, поданих у прямому коді:

$A = 0\ 011\ 1\ 10101$, $B = 0\ 101\ 0\ 11001$;
$A = 0\ 101\ 1\ 10101$, $B = 0\ 011\ 0\ 11001$;
$A = 0\ 101\ 1\ 10101$, $B = 0\ 100\ 1\ 11001$.

Виконання завдання

1) Знаходимо різницю порядків $B - A = 101 - 011 = 010$. Виконуємо зрівняння порядків заданих чисел, при цьому менший порядок числа A (011) збільшується до більшого порядку числа B (101), а мантиса числа A зсувається на два розряди вправо. Одержуємо:

$$\begin{aligned} A &= 0\ 101\ 1\ 0010101, \\ B &= 0\ 101\ 0\ 11001. \end{aligned}$$

Подамо мантиси чисел у доповнювальному коді і виконаємо додавання мантис:

$$\begin{array}{rll} & \mathrm{M}_{[A']\text{ДК}} & 11,\ 1101011 \\ + & \mathrm{M}_{[B]\text{ДК}} & 00,\ 1100100 \\ \hline & \mathrm{M}_{[C]\text{ДК}} & 00,\ 1001111\ . \end{array}$$

Отриманий результат є нормалізованим, тому лише подамо отримане число у прямому коді і виконаємо його округлення до заданої кількості розрядів мантиси.

Відповідь: $C = 0\ 101\ 0\ 10100$.

2) Знаходимо різницю порядків $B - A = 011 - 101 = -010$. Виконуємо зрівняння порядків заданих чисел, при цьому менший порядок числа B (011) збільшується до більшого порядку числа A (101), а мантиса числа B зсувається на два розряди вправо. Одержуємо:

$$\begin{aligned} A &= 0\ 101\ 1\ 10101, \\ B &= 0\ 101\ 0\ 0011001. \end{aligned}$$

Подамо мантиси чисел у доповнювальному коді і виконаємо додавання мантис:

$$\begin{array}{rll} & \mathrm{M}_{[A]\text{ДК}} & 11,\ 0101100 \\ + & \mathrm{M}_{[B']\text{ДК}} & 00,\ 0011001 \\ \hline & \mathrm{M}_{[C]\text{ДК}} & 11,\ 1000101\ . \end{array}$$

Отриманий результат є денормалізованим вправо, про що свідчить збіжність цифр знакового і старшого розряду мантиси.

Нормалізація результату. Зсуваємо мантису вліво на один розряд (11,000101) і виконуємо корекцію порядку (101 – 1 = 100). Подаємо

результат у прямому коді (11,111011) і виконуємо його округлення (11,11110).

Відповідь: $C = 0\ 100\ 1\ 11101$.

3) Знаходимо різницю порядків $B - A = 100 - 101 = -001$. Виконуємо зрівняння порядків заданих чисел, при цьому менший порядок числа B (100) збільшується до більшого порядку числа A (101), а мантиса числа B зсувається на один розряд вправо. Одержуємо:

$$A = \ 0\ 100\ 1\ 10101,$$
$$B = \ 0\ 101\ 1\ 011001.$$

Подамо мантиси чисел у доповнювальному коді і виконаємо додавання мантис:

$$\begin{array}{rll} & \text{М}_{[A]\text{ДК}} & 11,\,010110 \\ + & \text{М}_{[B']\text{ДК}} & 11,\,100111 \\ \hline & \text{М}_{[C]\text{ДК}} & 10,\,111101\,. \end{array}$$

Отриманий результат є денормалізованим вліво, про що свідчить незбіжність цифр у знакових розрядах мантиси.

Нормалізація результату. Зсуваємо мантису вправо на один розряд (11, 0111101) і виконуємо корекцію порядку (101+1=110). Подаємо мантису у прямому коді і виконуємо його округлення (1,10000).

Відповідь: $C = 0\ 110\ 1\ 10000$.

Б2-3. Реалізація арифметичних операцій

Приклад Б2-3.1

Завдання. Побудувати управляючий автомат для операційного пристрою з розподіленою логікою для обчислення функції

$$D = 0{,}5C - 4A(B + 1).$$

Для реалізації автомата застосувати автомат Мура. Як елементну базу застосувати логічні елементи І, АБО, НЕ та *JK*-тригери.

Виконання завдання

Розробимо спочатку операційну схему для обчислення заданої функції. В узагальненому вигляді пристрій складається з двох регістрів *RG*1 і *RG*2, суматора *SM* та лічильника *CT*. Операційна схема зображена на рис. Б2-3.1.

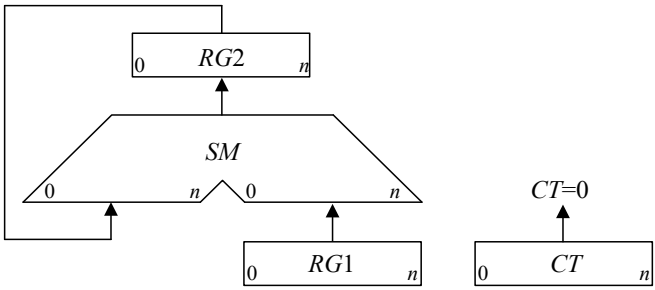


Рис. Б2-3.1. Операційна схема пристрою
для обчислення функції

Розробимо змістовний мікроалгоритм обчислення функції.

У вихідному стані в регістрах *RG*1 і *RG*2 та лічильнику *CT* знаходяться відповідно операнди *A*, *C* та *B*.

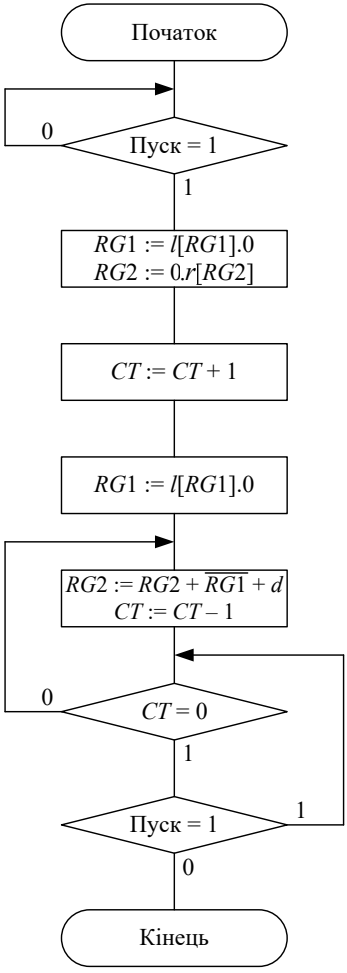


Рис. Б2-3.2. ГСА змістовного мікроалгоритму обчислення функції

Для реалізації множення операнду *A* на чотири необхідно виконати дві мікрооперації зсуву вліво вмісту регістру *RG*1. Для ділення операнда *C* на два необхідно виконати мікрооперацію зсуву вправо вмісту регістру *RG*2. Для збільшення на одиницю операнда *B* виконується інкремент лічильника *CT*. На етапі обчислення функції у циклі до *RG*2 додається *(B* + 1) раз вміст регістру *RG*1, поданий у доповнювальному коді, тобто реалізуємо мікрооперацію віднімання. Після чого зменшується на одиницю вміст лічильника *CT*. Обчис-

лення закінчується за виконання умови ($CT = 0$). Результат обчислення функції формується в регістрі $RG2$.

Змістовний мікроалгоритм має вигляд, зображений на рис. Б2-3.2.

На підставі операційної схеми та змістовного мікроалгоритму розробимо функціональну схему пристрою, наведену на рис. Б2-3.3. Розрядність вузлів пристрою залежить від діапазону подання операндів і не впливає на побудову управляючого автомата.

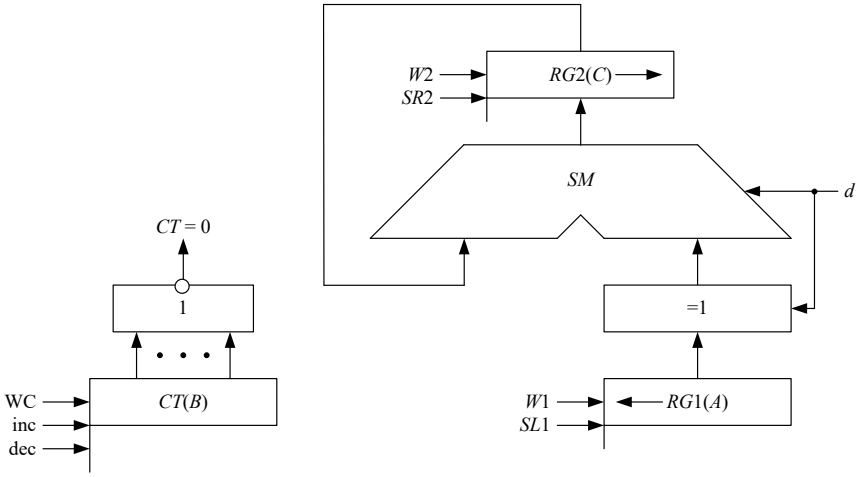


Рис. Б2-3.3. Функціональна схема пристрою для обчислення функції

Для одержання закодованого структурного мікроалгоритму складаємо таблицю кодування мікрооперацій (табл. Б2-3.1) та таблицю кодування логічних умов (табл. Б2-3.2), на підставі яких виконаємо заміну описів мікрооперацій та логічних умов у змістовному мікроалгоритмі відповідними позначеннями управляючих та логічних сигналів.

Теоретично мікрооперації віднімання та декримента лічильника можуть бути виконані в одному такті, бо вони виконуються на різних вузлах пристрою, але в цьому випадку не буде забезпечений необхідний перепад управляючих сигналів. Тому ці мікрооперації треба виконувати в різних тактах, що відображено у таблицях кодування мікрооперацій та на структурному мікроалгоритмі. Закодований структурний мікроалгоритм зображений на рис. Б2-3.4.

Таблиця Б2-3.1

ТАБЛИЦЯ КОДУВАННЯ МІКРООПЕРАЦІЙ

Мікрооперації	Управляючі сигнали	Позначення управляючих сигналів
$RG1 := l\,[RG1].0$	$SL1$	y_1
$RG2 := 0.r\,[RG2]$	$SR2$	y_2
$CT := CT + 1$	inc	y_3
$CT := CT - 1$	dec	y_4
$RG2 := RG2 + \overline{RG1} + d$	$W2, d$	y_5

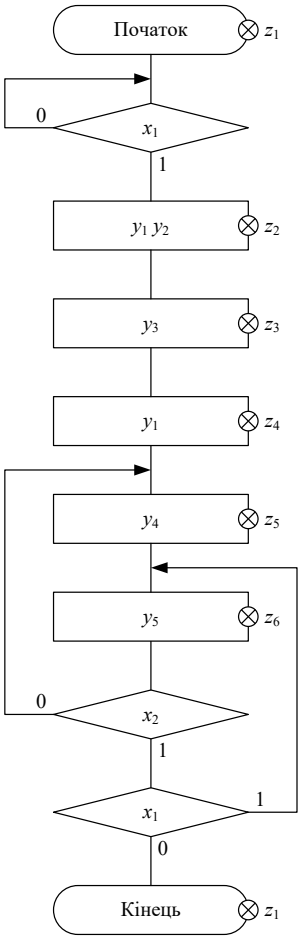


Рис. Б2-3.4. ГСА закодованого мікроалгоритму автомата Мура

Таблиця Б2-3.2

ТАБЛИЦЯ КОДУВАННЯ ЛОГІЧНИХ УМОВ

Логічні умови	Позначення логічних умов
Пуск = 1	x_1
$CT = 0$	x_2

Отриманий закодований мікроалгоритм (рис. Б2-3.4) є вихідним для побудови пристрою управління.

Виконаємо розмітку станів автомата Мура на закодованому мікроалгоритмі (рис. Б2-3.4), стани автомату позначимо z_i. В результаті отримуємо мікро алгоритм, позначений шістьма станами, за яким будуємо граф автомата Мура (рис. Б2-3.5).

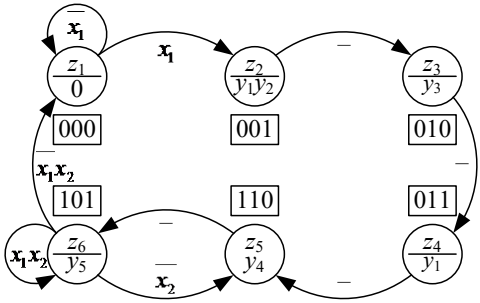


Рис. Б2-3.5. Граф автомата Мура

Для побудови функціональної схеми управляючого автомату у цьому випадку знадобиться три тригери і таблиця кодування станів буде мати вигляд наведений у таблиці Б2-3.3.

Таблиця Б2-3.3

ТАБЛИЦЯ КОДУВАННЯ СТАНІВ АВТОМАТА МУРА

Стан	Код стану		
	Q_3	Q_2	Q_1
z_1	0	0	0
z_2	0	0	1
z_3	0	1	0
z_4	0	1	1
z_5	1	1	0
z_6	1	0	1

За графом автомата Мура складаємо структурну таблицю автомата (табл. Б2-3.4). Значення функцій збудження тригерів визначаються відповідно до графічної схеми переходів *JK*-тригера (рис. А1-4.3).

Таблиця Б2-3.4

СТРУКТУРНА ТАБЛИЦЯ АВТОМАТА МУРА

ПС	Код ПС			СП	Код СП			Логічні умови		Управляючі сигнали					Функції збудження тригерів					
	Q_3^t	Q_2^t	Q_1^t		Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	x_1	x_2	y_1	y_2	y_3	y_4	y_5	J_3	K_3	J_2	K_2	J_1	K_1
z_1	0	0	0	z_1	0	0	0	0	–	0	0	0	0	0	0	–	0	–	0	–
z_1	0	0	0	z_2	0	0	1	1	–	0	0	0	0	0	0	–	0	–	1	–
z_2	0	0	1	z_3	0	1	0	–	–	1	1	0	0	0	0	–	1	–	–	1
z_3	0	1	0	z_4	0	1	1	–	–	0	0	1	0	0	0	–	–	0	1	–
z_4	0	1	1	z_5	1	1	0	–	–	1	0	0	0	0	1	–	–	0	–	1
z_5	1	1	0	z_6	1	0	1	–	–	0	0	0	1	0	–	0	–	1	1	–
z_6	1	0	1	z_5	1	1	0	–	0	0	0	0	0	1	–	0	1	–	–	1
z_6	1	0	1	z_6	1	0	1	1	1	0	0	0	0	1	–	0	0	–	–	0
z_6	1	0	1	z_1	0	0	0	0	1	0	0	0	0	1	–	1	0	–	–	1

На підставі структурної таблиці автомата визначаємо МДНФ функцій управляючих сигналів y_i та J_j і K_j. Аргументами функцій збудження тригерів є значення коду початкового стану автомату Q_3, Q_2, Q_1 та значення логічних умов x_1, x_2, що забезпечують відповідний перехід автомату у наступний стан. Аргументами перемикальних функцій управляючих сигналів є значення Q_3, Q_2, Q_1, це пов'язане з тим, що вихідні сигнали автомата Мура не залежать від вхідних сигналів. Тому МДНФ функцій управляючих сигналів отримаємо безпосередньо із графа автомата Мура (рис. Б2-3.5), а для одержання МДНФ функцій збудження тригерів використаємо діаграми Вейча, наведені на рис. Б2-3.6.

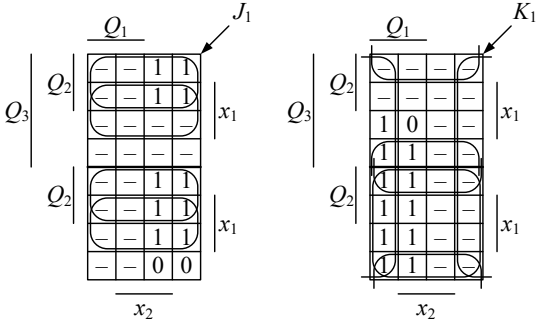
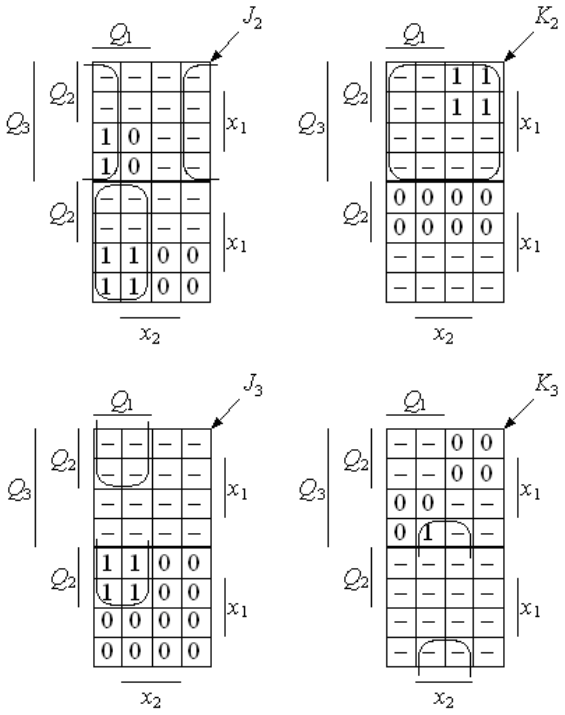


Рис. Б2-3.6. Діаграми Вейча функцій збудження тригерів



Закінчення рис. Б2-3.6.

Після виконання мінімізації отримаємо:

$$J_1 = Q_2 \vee x_1;$$
$$K_1 = \overline{Q_3} \vee \overline{x_2} \vee \overline{x_1};$$
$$J_2 = Q_3 \cdot \overline{x_2} \vee \overline{Q_3} \cdot Q_1;$$
$$K_2 = Q_3;$$
$$J_3 = Q_2 \cdot Q_1;$$
$$K_3 = \overline{Q_2} \cdot x_2 \cdot \overline{x_1};$$

$$y_1 = \overline{Q_3} \cdot \overline{Q_2} \cdot Q_1 \vee \overline{Q_3} \cdot Q_2 \cdot Q_1 = \overline{Q_3} \cdot Q_1;$$
$$y_2 = \overline{Q_3} \cdot \overline{Q_2} \cdot Q_1;$$
$$y_3 = \overline{Q_3} \cdot Q_2 \cdot \overline{Q_1};$$
$$y_4 = Q_3 \cdot Q_2 \cdot \overline{Q_1};$$
$$y_5 = Q_3 \cdot \overline{Q_2} \cdot Q_1.$$

За отриманими формами будуємо функціональну схему управляючого автомата в елементному базисі І, АБО, НЕ (рис. Б2-3.7).

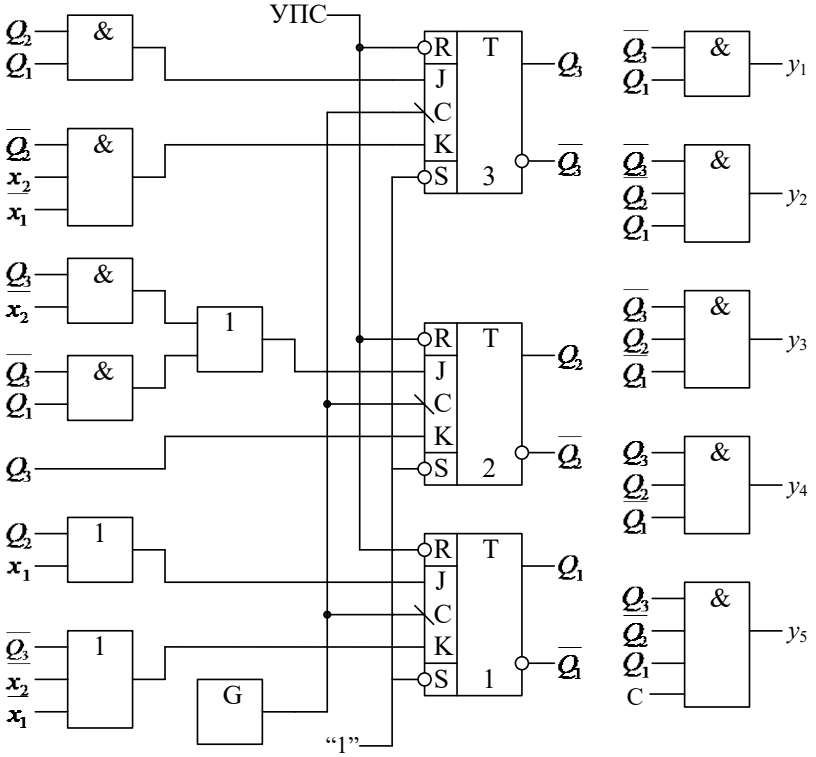


Рис. Б2-3.7. Функціональна схема управляючого автомата для обчислення функції

Приклад Б2-3.2

Завдання. За умови попереднього прикладу побудувати управляючий автомат Мілі.

Виконання завдання

Функціональна схема пристрою для обчислення заданої функції наведена на рис. Б2-3.3.

Як вихідний для побудови управляючого автомата застосуємо розроблений закодований мікроалгоритм обчислення заданої функції (рис. Б2-3.4).

Виконуємо розмітку станів автомата Мілі і отримуємо мікроалгоритм, позначений шістьма станами (рис. Б2-3.8), за яким будуємо граф автомата Мілі (рис. Б2-3.9).

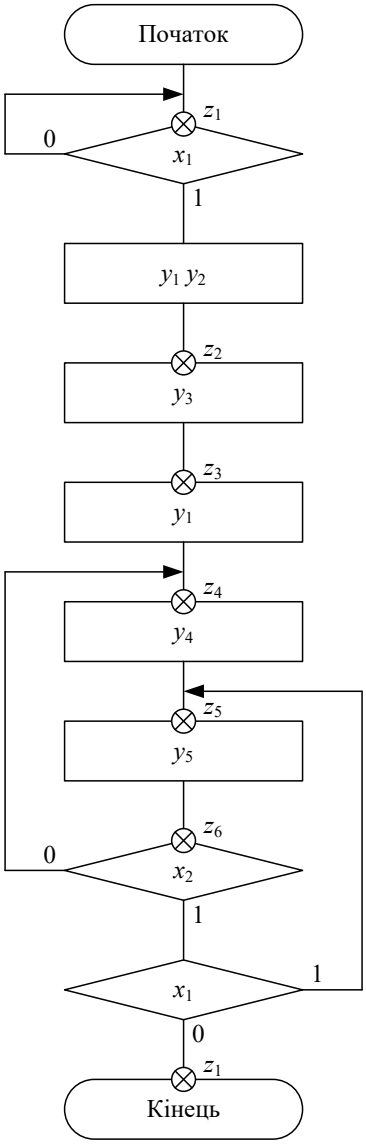


Рис. Б2-3.8. ГСА закодованого мікроалгоритму автомата Мілі

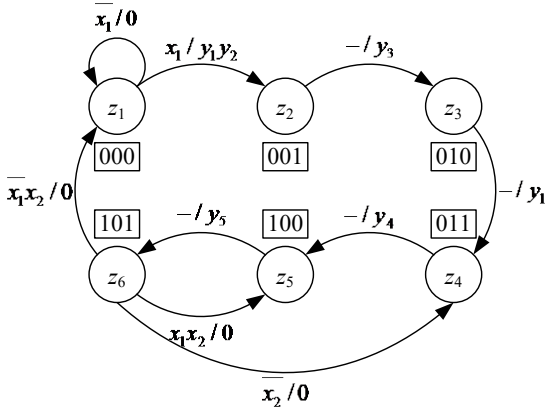


Рис. Б2-3.9. Граф автомата Мілі

Таблиця кодування станів має вигляд, наведений у табл. Б2-3.5.

Таблиця Б2-3.5

ТАБЛИЦЯ КОДУВАННЯ СТАНІВ АВТОМАТА МІЛІ

Стан	Код стану		
	Q_3	Q_2	Q_1
z_1	0	0	0
z_2	0	0	1
z_3	0	1	0
z_4	0	1	1
z_5	1	0	0
z_6	1	0	1

За графом автомата Мура складаємо структурну таблицю автомата (табл. Б2-3.6).

Аргументами перемикальних функцій y_i, J_j та K_j є значення коду початкового стану автомату Q_3, Q_2, Q_1 та значення логічних умов x_1 і x_2. Мінімальні ДНФ функцій управляючих сигналів отримаємо безпосередньо зі структурної таблиці автомату, для одержання МДНФ функцій збудження тригерів використаємо діаграми Вейча, наведені на рис. Б2-3.10.

Таблиця Б2-3.6

СТРУКТУРНА ТАБЛИЦЯ АВТОМАТА МІЛІ

ПС	Код ПС			СП	Код СП			Логічна умова		Управляючі сигнали					Функції збудження тригерів					
	Q_3^t	Q_2^t	Q_1^t		Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	x_1	x_2	y_1	y_2	y_3	y_4	y_5	J_3	K_3	J_2	K_2	J_1	K_1
z_1	0	0	0	z_1	0	0	0	0	–	0	0	0	0	0	0	–	0	–	0	–
z_1	0	0	0	z_2	0	0	1	1	–	1	1	0	0	0	0	–	0	–	1	–
z_2	0	0	1	z_3	0	1	0	–	–	0	0	1	0	0	0	–	1	–	–	1
z_3	0	1	0	z_4	0	1	1	–	–	1	0	0	0	0	0	–	–	0	1	–
z_4	0	1	1	z_5	1	0	0	–	–	0	0	0	1	0	1	–	–	1	–	1
z_5	1	0	0	z_6	1	0	1	–	–	0	0	0	0	1	–	0	0	–	1	–
z_6	1	0	1	z_5	1	0	0	1	1	0	0	0	0	0	–	0	0	–	–	1
z_6	1	0	1	z_4	0	1	1	–	0	0	0	0	0	0	–	1	1	–	–	0
z_6	1	0	1	z_1	0	0	0	0	1	0	0	0	0	0	–	1	0	–	–	1

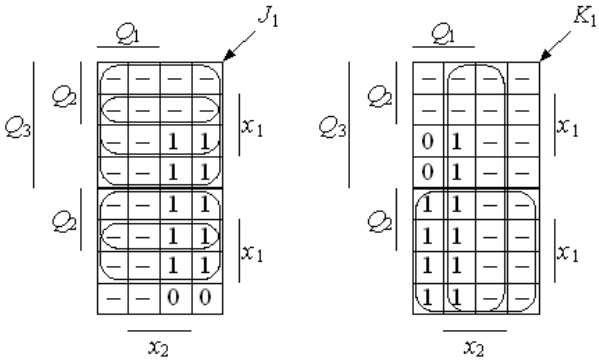
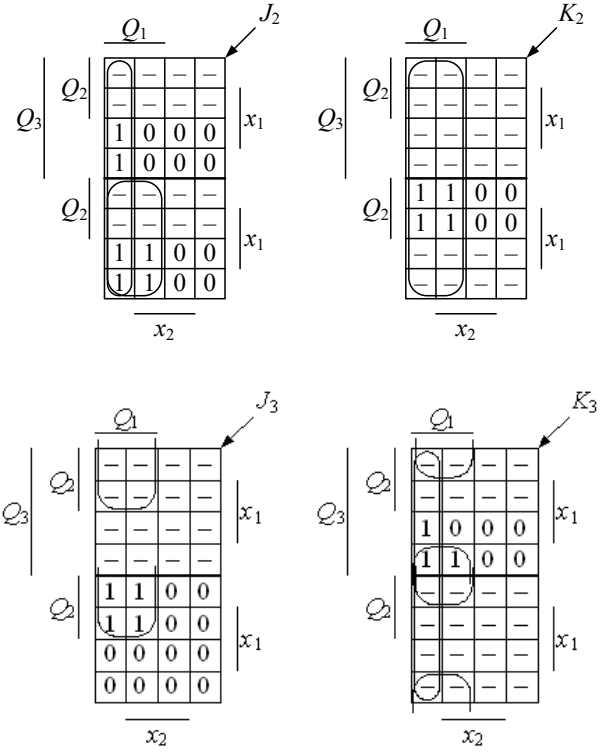


Рис. Б2-3.10. Діаграми Вейча функцій
управляючих сигналів
та функцій збудження тригерів



Закінчення рис. Б2-3.10

У результаті мінімізації отримуємо:

$$J_1 = Q_3 \vee Q_2 \vee x_1;$$
$$K_1 = \overline{Q_3} \vee x_2;$$
$$J_2 = \overline{Q_3} \cdot Q_1 \vee Q_1 \cdot \overline{x_2};$$
$$K_2 = Q_1;$$
$$J_3 = Q_2 \cdot \overline{Q_1};$$
$$K_3 = Q_1 \cdot \overline{x_2} \vee Q_1 \cdot \overline{x_1};$$

$$y_1 = \overline{Q_3} \cdot \overline{Q_2} \cdot \overline{Q_1} \cdot x_1 \vee \overline{Q_3} \cdot Q_2 \cdot \overline{Q_1};$$
$$y_2 = \overline{Q_3} \cdot \overline{Q_2} \cdot \overline{Q_1} \cdot x_1;$$
$$y_3 = \overline{Q_3} \cdot \overline{Q_2} \cdot Q_1;$$
$$y_4 = \overline{Q_3} \cdot Q_2 \cdot \overline{Q_1};$$
$$y_5 = Q_3 \cdot \overline{Q_2} \cdot \overline{Q_1}.$$

За отриманими нормальними формами будуємо функціональну схему управляючого автомата в елементному базисі І, АБО, НЕ (рис. Б2-3.11).

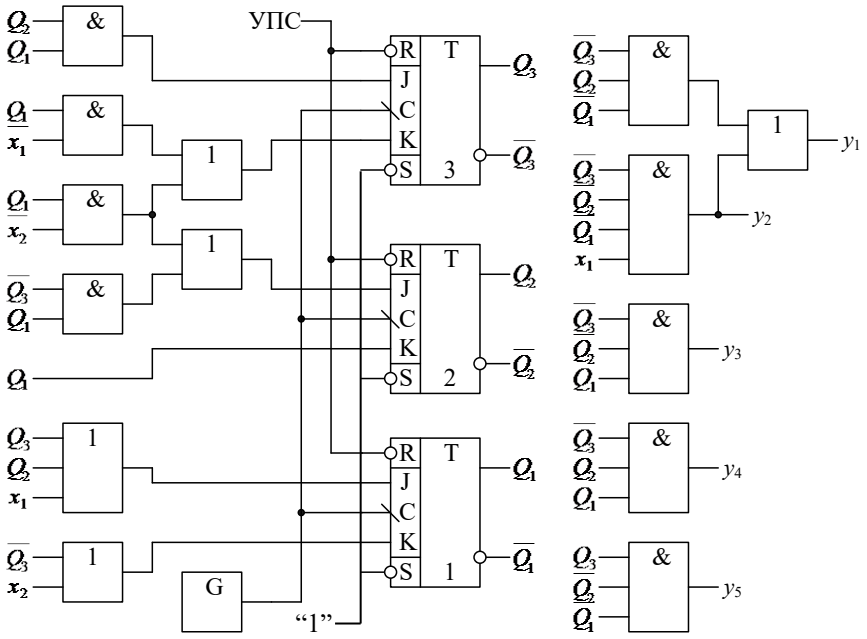


Рис. Б2-3.11. Функціональна схема управляючого автомата для обчислення функції

Приклад Б2-3.3

Завдання. Побудувати таблицю станів регістрів арифметичного пристрою для множення чисел B і A за другим способом, якщо:

$$A = 0{,}1011,\ B = 0{,}1001.$$

Виконання завдання

За другим способом множення здійснюється з молодших розрядів множника, множник зсувається вправо, множене вліво, сума часткових добутків залишається нерухомою.

Операційна схема пристрою для виконання операції множення за другим способом наведена на рис. Б1-3.1, *б*.

У регістрі $RG2$ знаходиться множник A, в молодших розрядах регістру $RG3$ — множене B, в регістрі $RG1$ у вихідному стані записані нулі, у цьому регістрі накопичується сума часткових добутків, та наприкінці виконання множення формується результат операції. За нульовим вмістом лічильнику циклів CT визначається закінчення операції множення. В цьому прикладі $n = 4$.

Змістовний мікроалгоритм виконання операції множення зображено на рис. Б2-3.12.

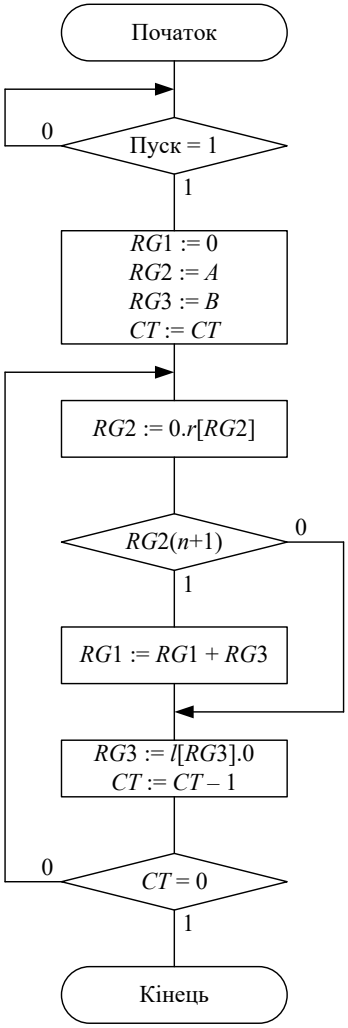


Рис. Б2-3.12. ГСА змістовного мікроалгоритму виконання операції множення другим способом

Наведемо діаграму стану регістрів під час виконання операції множення за другим способом (рис. Б2-3.13).

	*RG*1(0...0) [1…2*n*]	← *RG*3 (*B*) [1…2*n*]	*RG*2 (*A*) → [1…*n*]	*n*+1	СТ (*n*) [1…*q*]	Мікрооперація
ПС	0000 0000	0000 1001	1011		100	
1	+0000 1001 0000 1001	0001 0010	0101	1	011	Зсув *RG*2 → *C* = 1 *RG*1:= *RG*1+ *RG*3 Зсув ← *RG*3 *CT*:= *CT*–1
2	+0001 0010 0001 1011	0010 0100	0010	1	010	Зсув *RG*2 → *C* = 1 *RG*1:= *RG*1+ *RG*3 Зсув ← *RG*3 *CT*:= *CT*–1
3		0100 1000	0001	0	001	Зсув *RG*2 → *C* = 1 Зсув ← *RG*3 *CT*:= *CT*–1
4	+0100 1000 **0110 0011** Результат	1001 0000	0000	1	000	Зсув *RG*2 → *C* = 1 *RG*1:= *RG*1+ *RG*3 Зсув ← *RG*3 *CT*:= *CT*–1

Рис. Б2-3.13. Таблиця стану регістрів під час виконання операції множення другим способом

Приклад Б2-3.4

Завдання. Для арифметичного пристрою для ділення чисел із зсувом залишку побудувати таблицю станів регістрів для таких значень аргументів: $Y = 0{,}1101$, $X = 0{,}1000$.

Виконання завдання

Під час реалізації способу, що розглядається, здійснюється зсув вліво залишку за нерухомого дільника. Операційна схема пристрою для виконання операції ділення із зсувом залишку наведена на рис. Б1-4.1. Змістовний мікроалгоритм виконання операції представлено на рис. Б2-3.14.

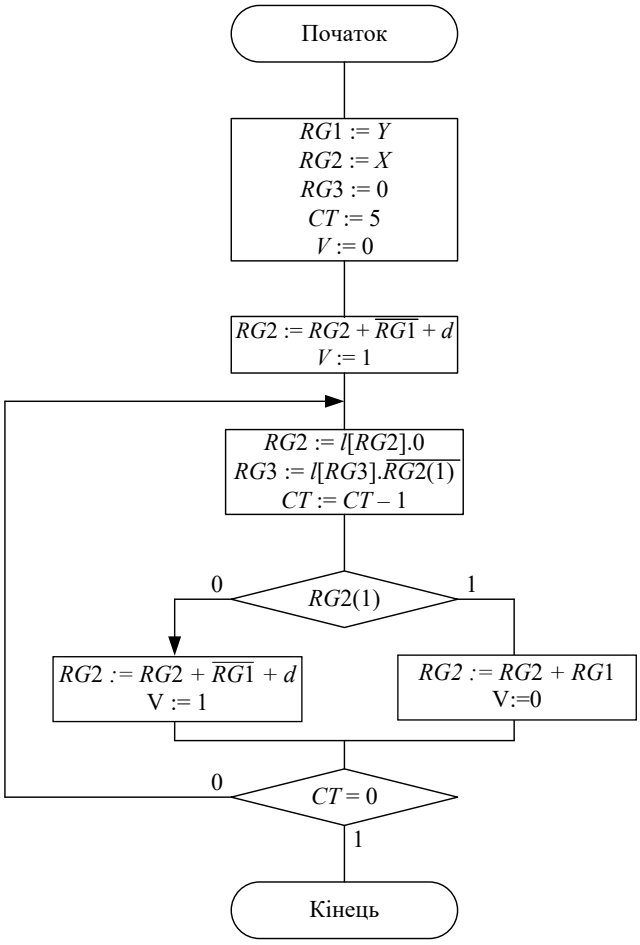


Рис. Б2-3.14. ГСА змістовного мікроалгоритму виконання операції ділення із зсувом залишку

У початковому стані в регістрі *RG*2 розміщується ділене *X*. Дільник *Y* знаходиться в регістрі *RG*1. Черговий залишок формується в регістрі *RG*2. Розрядність регістрів *RG*1 і *RG*2 дорівнює $(n + 2)$, де два розряди відводяться для подання знаку операндів. У лічильник СТ записується кількість циклів обчислень, необхідна для отримання результату певної розрядності. За нульовим вмістом лічильнику циклів *CT* визначається закінчення операції ділення.

Наведемо таблицю стану регістрів під час виконання операції (рис. Б2-3.15).

	$\leftarrow RG3(Z)$ [1...(*n*+1)]	$\leftarrow RG2(X)$ [1...(*n*+2)]	$RG1(Y)$ [1...(*n*+2)]	$CT(n)$ [1...q]	Мікрооперації
ПС	00000	00,1000	$00,1101_{пк}$ $11,0011_{дк}$	101	
1		00,1000 +11,0011 11,1011			$RG2 := RG2 - RG1$
2	$\bar{1}$ 00000	11,0110 11,0110 +00,1101 00,0011		100	$\leftarrow RG2$ $\leftarrow RG3$ $CT := CT - 1$ $RG2(1) = 1$ $RG2 := RG2 + RG1$ $CT \neq 0$
3	$\bar{0}$ 00001	00,0110 00,0110 +11,0011 11,1001		011	$\leftarrow RG2$ $\leftarrow RG3$ $CT := CT - 1$ $RG2(1) = 0$ $RG2 := RG2 - RG1$ $CT \neq 0$
4	$\bar{1}$ 00010	11,0010 11,0010 +00,1101 11,1111		010	$\leftarrow RG2$ $\leftarrow RG3$ $CT := CT - 1$ $RG2(1) = 1$ $RG2 := RG2 + RG1$ $CT \neq 0$
5	$\bar{1}$ 00100	11,1110 11,1110 +00,1101 00,1011		001	$\leftarrow RG2$ $\leftarrow RG3$ $CT := CT - 1$ $RG2(1) = 1$ $RG2 := RG2 + RG1$ $CT \neq 0$
6	$\bar{0}$ 01001	01,0110 01,0110 +11,0011 00,1001		000	$\leftarrow RG2$ $\leftarrow RG3$ $CT := CT - 1$ $RG2(1) = 0$ $RG2 := RG2 - RG1$ $CT \neq 0$

Рис. Б2-3.15. Таблиця стану регістрів під час виконання операції ділення зі зсувом залишку

Б3. ЛАБОРАТОРНІ РОБОТИ

Б3.1. Лабораторна робота 1

Додавання чисел у машинних кодах

[Б1-2.2, Б1-2.3, Б1-2.4]

Ціль роботи: вивчити правила перетворення додатних і від'ємних двійкових чисел у машинні коди. Дослідити способи реалізації операцій додавання і віднімання.

Підготовка до роботи

1. Визначити свій варіант завдання за табл. Б3-1.1. Для цього необхідно одержати шість молодших розрядів номера залікової книжки студента, подати їх у двійковій системі числення (h_6, ..., h_1), після чого підставити значення h_i у табл. Б3-1.1.

2. Перевести зазначені в завданні числа в двійкову систему числення і визначити довжину розрядної сітки.

3. Подати кожне з отриманих двійкових чисел у вигляді ПК, ОК, ДК.

4. Виконати операції $S1$, $S2$, $S3$, відповідно до варіанта завдання в прямому, зворотному і доповнювальному кодах.

5. Подати отримані результати у вигляді двійкових та десяткових чисел зі знаком.

Таблиця Б3-1.1

ВАРІАНТИ ЗАВДАНЬ

h_3	h_2	h_1	A	B	C	D	h_6	h_5	h_4	$S1$	$S2$	$S3$	Код
0	0	0	31	– 12	52	– 63	0	0	0	$A + C$	$D + B$	$C – B$	ДК
0	0	1	– 8	43	– 26	17	0	0	1	$B + C$	$C + D$	$D – A$	ДК
0	1	0	64	– 13	27	5	0	1	0	$C + B$	$A – C$	$D + C$	ОК
0	1	1	34	– 6	10	52	0	1	1	$A + D$	$C + A$	$B – C$	ДК
1	0	0	27	73	– 2	– 6	1	0	0	$C – A$	$A + C$	$D + B$	ОК
1	0	1	– 24	– 7	62	39	1	0	1	$B + D$	$C – B$	$A + B$	ДК
1	1	0	43	– 28	7	14	1	1	0	$A – C$	$A – D$	$D + B$	ОК
1	1	1	52	– 18	3	– 36	1	1	1	$A + C$	$B – D$	$C + B$	ОК

6. Побудувати операційну схему пристрою для виконання операцій додавання і віднімання в машинному коді, зазначеному в завданні (ДК або ОК).

7. Довести, що за будь-якого сполучення знаків операндів результат додавання (віднімання) у ДК не вимагає корекції.

Порядок виконання роботи

1. Побудувати і налагодити операційний пристрій для реалізації операцій додавання і віднімання в модифікованому машинному коді, зазначеному в завданні (ДК або ОК). Приклад операційного пристрою для додавання чисел наведений у додатку 2.1.

2. Виконати операції додавання і віднімання на операційному пристрої з різними наборами чисел.

3. Дослідити випадки, коли може бути переповнення розрядної сітки, зробити висновок.

Контрольні питання

1. Сформулюйте правило перетворення від'ємних і додатних чисел у прямий код.

2. Сформулюйте правило перетворення від'ємних і додатних чисел в обернений код.

3. Сформулюйте правило перетворення від'ємних і додатних чисел у доповнювальний код.

4. Як реалізується операція віднімання в цифрових машинах?

5. У яких випадках може бути втрата значимості результату при додаванні й відніманні чисел у машинних кодах?

6. Як визначити переповнення розрядної сітки?

7. Для чого застосовується модифікований код?

8. При якому сполученні знаків операндів модифікований код не потрібен для правильного формування знака результату.

9. Побудуйте схему пристрою тільки для додавання чисел в ОК (ДК).

Додаткова література: **[**9, 10, 13, 16, 17, 19**]**

Б3.2. Лабораторна робота 2

Розроблення мікроалгоритмів та операційних пристоїв

[Б1-2.1, Б1-2.3, Б1-2.4, Б1-2.5]

Ціль роботи: ознайомитися зі способами побудови операційних схем та мікроалгоритмів на прикладі реалізації алгебраїчних обчислень, одержати навички управління мікроопераціями.

Підготовка до роботи

1. Скласти змістовний мікроалгоритм та операційну схему пристрою із розподіленою логікою для обчислення функції *D*. Для визначення варіанту застосовуються п'ять молодших розрядів номеру залікової книжки студента, поданого у двійковій системі числення — $h_5\ h_4\ h_3\ h_2\ h_1$. Вид функції *D* визначається за табл. Б3-2.1 відповідно до значень розрядів h_5 і h_4 отриманого двійкового числа.

2. Вважати, що у вихідному стані операнди *A*, *B* та *C* записані в регістри *RG*1, *RG*2 та лічильник *CT*.

3. Побудувати функціональну схему обчислювального пристрою та скласти закодований мікроалгоритм обчислення функції.

4. Виконати структурний синтез управління автомата. Тип тригерів визначається за табл. Б3-2.1 залежно від значень розрядів h_3 та h_2, тип автомата визначається значенням h_1. Для побудови комбінаційної схеми використовувати логічні елементи І-НЕ.

Таблиця Б3-2.1

ВАРІАНТИ ЗАВДАННЯ

h_5	h_4	Функція	h_3	h_2	Тип тригера	h_1	Тип автомата
0	0	$D = 2C - 4AB$	0	0	*JK*	0	Мілі
0	1	$D = A\ (B - 1) + 0{,}5C$	0	1	*T*		
1	0	$D = 2A(B + 1) + 0{,}5C$	1	0	*RS*	1	Мура
1	1	$D = A(B + 1) + 2C$	1	1	*D*		

5. Побудувати часову діаграму роботи автомата для кожної комбінації значень логічних умов.

6. Проілюструвати прикладом обчислення результату *D* для одного довільного набору значень операндів *A*, *B* та *C*, поданих трьома двійковими розрядами. Змоделювати роботу пристрою за допомогою таблиці станів вузлів у кожному такті роботи пристрою.

Порядок виконання роботи

1. Викликати з відповідного каталогу схему операційного пристрою, зберегти її з новим ім'ям у власній папці студента. Схема операційного пристрою для обчислення функції наведена у додатку 2.2. Відкрити операційну схему пристрою у моделюючій програмі ПРОГМОЛС 2.0 та доповнити її схемою управляючого автомата.

2. Виконати налагодження схеми управляючого автомата в синхронному режимі, при цьому виходи автомата до входів операційного пристрою не підключати.

3. Підключити до управляючих входів операційного пристрою виходи автомата. Записати в регістри *RG*1, *RG*2 та лічильник *CT* вихідні операнди, відповідно до завдання. Зробити комплексне налагодження схеми в синхронному режимі і переконатися в правильності одержання результату *D* у регістрі *RG*2.

4. Перейти до асинхронного моделювання. Дослідити зазначені викладачем часові параметри схеми.

Контрольні питання

1. Дайте визначення таким поняттям: операція, мікрооперація, мікроалгоритм, змістовний мікроалгоритм, закодований мікроалгоритм, операційна схема, структурна схема, функціональна схема, принципова схема.

2. Складіть операційну схему та змістовний мікроалгоритм заданого викладачем алгебраїчного перетворення даних.

3. Як перейти від змістовного мікроалгоритму до закодованого мікроалгоритму?

4. Як побудувати граф управляючого автомата за закодованим алгоритмом?

5. Як розрахувати розрядність регістрів для операційного пристрою?

6. Як визначити необхідну тривалість управляючих сигналів?

7. Як визначити час переходу автомата з одного стану в інший?

8. Поясніть необхідність введення додаткових станів автомата.

9. Чи можливий перехід автомата в стан, непередбачений графом, за використання тригерів із внутрішньою затримкою (тригерів, керованих рівнем сигналів)?

10. Коли можливе виникнення помилкових управляючих сигналів (непередбачених графом автомата) і чим визначається їх тривалість?

11. Як визначити час виконання мікрооперацій?

12. Наведіть таблиці переходів для *JK*-, *RS*-, *T*- і *D*-тригерів.

Додаткова література: **[**9, 10, 13, 16, 17, 19**]**

Б3-3. Лабораторна робота 3

Реалізація операції множення чисел

[Б1-2.1, Б1-3.1]

Ціль роботи: вивчити методи реалізації операції множення, одержати навички в проектуванні й налагодженні операційних і управляючих пристроїв.

Підготовка до роботи

1. Для визначення варіанту застосовуються шість молодших розрядів номеру залікової книжки студента, поданого у двійковій системі числення — $h_6\ h_5\ h_4\ h_3\ h_2\ h_1$.

2. Відповідно до завдання (табл. Б3-3.1) розробити операційну схему і змістовний мікроалгоритм множення позитивних чисел. Для побудови операційної схеми використати суматор, лічильник циклів і асинхронні регістри, що мають входи управління зсувом і занесенням інформації. На схемі має бути зазначена розрядність регістрів.

Таблиця Б3-3.1

ВАРІАНТИ ЗАВДАННЯ

h_6	h_5	h_4	Спосіб множення	Розрядність операндів	h_3	h_2	Тип тригера	h_1	Тип автомата
0	0	0	1-й	4	0	0	*JK*	1	Мілі
0	0	1	2-й	4					
0	1	0	3-й	4	0	1	*T*		
0	1	1	4-й	4					
1	0	0	1-й	8	1	0	*RS*	0	Мура
1	0	1	2-й	3					
1	1	0	3-й	6	1	1	*D*		
1	1	1	4-й	3					

3. Побудувати функціональну схему пристрою для множення чисел заданим способом.

4. Здійснити синтез управляючого автомата. Враховувати, що мікрооперації на регістрах виконуються по негативному перепаду управляючих сигналів.

5. Побудувати часову діаграму роботи автомата для кожної комбінації значень логічних умов.

6. Виконати числовий приклад множення з довільними значеннями операндів, розрядність операндів зазначена у табл. Б3-3.1. Змоделювати роботу пристрою за допомогою таблиці станів вузлів пристрою з обраними значеннями операндів. Наведений приклад використати при налагодженні пристрою.

Порядок виконання роботи

1. Викликати з відповідного каталогу схему операційного пристрою, зберегти її з новим ім'ям у власній папці студента. Схема операційного пристрою для виконання операції множення наведена у додатку 2.3. Відкрити операційну схему пристрою у моделюючій програмі ПРОГМОЛС 2.0 та доповнити її схемою управляючого автомата.

2. Виконати налагодження схеми автомата в синхронному режимі, при цьому виходи автомата до входів операційного пристрою не підключати.

3. Підключити до управляючих входів операційного пристрою виходи автомата. Записати у відповідні регістри пристрою вихідні операнди. Зробити комплексне налагодження схеми в синхронному режимі і переконатися в правильності одержання результату виконання операції.

4. Перейти до асинхронного моделювання. Досліджувати зазначені викладачем часові параметри схеми.

Контрольні питання

1. Охарактеризуйте чотири основних методи множення чисел.
2. Як розрахувати розрядність вузлів операційного пристрою?
3. Що таке мікроалгоритм операції?
4. Нарисуйте узагальнену структурну схему управляючого автомата.
5. Наведіть вирази функцій, що визначають закон функціонування автоматів Мілі і Мура.
6. У чому відмінність автоматів Мілі та Мура?
7. Охарактеризуйте основні етапи проектування автомата.
8. Як перейти від змістовного мікроалгоритму до закодованого мікроалгоритму?
9. Як побудувати граф автомата?
10. Як визначити необхідну тривалість управляючих сигналів?
11. Від чого залежить кількість тригерів, необхідна для побудови автомата?
12. Як скласти структурну таблицю автомата?

Додаткова література: **[**9, 10, 13, 16, 17, 19**]**

Б3-4. Лабораторна робота 4

Дослідження способів ділення чисел

[Б1-2.1, Б1-4.1]

Ціль роботи: оволодіти методами ділення чисел в прямих кодах і способами їх апаратурної реалізації, одержати навички в налагодженні та дослідженні операційних і управляючих автоматів.

Підготовка до роботи

1. Для визначення варіанту застосовуються чотири молодших розрядів номеру залікової книжки студента, поданого у двійковій системі числення — $h_4\ h_3\ h_2\ h_1$.

2. Відповідно до завдання (табл. Б3-4.1) розробити операційну схему і змістовний мікроалгоритм операції ділення позитивних чисел. Для побудови операційної схеми використати суматор, лічильник циклів і асинхронні регістри, що мають входи управління зсувом і занесенням інформації. На схемі має бути зазначена розрядність регістрів.

3. Побудувати функціональну схему пристрою для ділення чисел заданим способом.

4. Здійснити синтез управляючого автомата. Враховувати, що мікрооперації на регістрах виконуються по негативному перепаду управляючих сигналів.

5. Побудувати часову діаграму роботи автомата для кожної комбінації значень логічних умов.

6. Виконати числовий приклад ділення з довільними значеннями операндів, розрядність операндів зазначена у табл. Б3-4.1. Змоделювати роботу пристрою за допомогою таблиці станів вузлів пристрою з обраними значеннями операндів. Наведений приклад використати при налагодженні пристрою.

Таблиця Б3-4.1

ВАРІАНТИ ЗАВДАННЯ

h_5	h_4	Спосіб ділення	Розрядність операндів	h_3	h_2	Тип тригера	h_1	Тип автомата
0	0	1-й	7	0	0	*JK*	1	Мілі
0	1	2-й	6	0	1	*T*		
1	0	1-й	5	1	0	*RS*	0	Мура
1	1	2-й	4	1	1	*D*		

Порядок виконання роботи

1. Викликати з відповідного каталогу схему операційного пристрою, зберегти її з новим ім'ям у власній папці студента. Схема операційного пристрою для виконання операції ділення наведена у додатку 2.4. Відкрити операційну схему пристрою у моделюючій програмі ПРОГМОЛС 2.0 та доповнити її схемою управляючого автомата.

2. Виконати налагодження схеми автомата в синхронному режимі, при цьому виходи автомата до входів операційного пристрою не підключати.

3. Підключити до управляючих входів операційного пристрою виходи автомата. Записати у відповідні регістри пристрою вихідні операнди. Зробити комплексне налагодження схеми в синхронному режимі і переконатися в правильності одержання результату виконання операції.

4. Перейти до асинхронного моделювання. Досліджувати зазначені викладачем часові параметри схеми.

Контрольні питання

1. Опишіть алгоритми ділення чисел в прямих кодах.
2. Вкажіть переваги і недоліки реалізації різних варіантів ділення.
3. Як визначити тривалість виконання цифрових операцій?
4. В яких пристроях для ділення можна сполучати мікрооперації додавання, віднімання і зсуву? Чому можливе сполучення цих цифрових операцій?
5. Чи можна зменшити довжину регістрів операційного пристрою при реалізації ділення чисел за другим варіантом, якщо результат має бути представлений q розрядами ($q < n$)?
6. Як здійснюється округлення результату ділення?
7. Схарактеризуйте етапи синтезу операційних пристроїв.
8. Які тригери (з погляду внутрішньої організації) можна використовувати для побудови кожного з регістрів ділильного пристрою?
9. Як одержати *T*- і *RS*-тригери на основі *JK*-тригерів?
10. Складіть таблиці переходів *T*-, *D*-, *RS*- і *JK*-тригерів.
11. В яких випадках управляючий автомат може виробляти сигнали, непередбачені його графом? Яка максимальна тривалість таких сигналів?
12. Нарисуйте операційні схеми пристроїв для ділення чисел.

Додаткова література: **[**9, 10, 13, 16, 17, 19**]**

Б4. МОДУЛЬНИЙ КОНТРОЛЬ

Б4-1. Задачі для самостійного розв'язування

Б4-1.1. Подання чисел у ЕОМ

1.1. Перевести двійкове число X у десяткову систему числення:

a) $X_{(2)} = 101011{,}11001$,
b) $X_{(2)} = 110010{,}00101$,
c) $X_{(2)} = 100011{,}10101$,
d) $X_{(2)} = 11111111$,
e) $X_{(2)} = 10111111$,
f) $X_{(2)} = 11101010001$.

1.2. Перевести десяткове число B у двійкову систему числення:

a) $B_{(10)} = 325{,}257$,
b) $B_{(10)} = 165{,}257$,
c) $B_{(10)} = 624{,}325$,
d) $B_{(10)} = 2313$,
e) $B_{(10)} = 3151$,
f) $B_{(10)} = 6627$.

1.3. Перевести десяткове число A в шістнадцятирічну систему числення:

a) $A_{(10)} = 7511$,
b) $A_{(10)} = 2048$,
c) $A_{(10)} = 6727$,
d) $A_{(10)} = 8885$,
e) $A_{(10)} = 2689$,
f) $A_{(10)} = 1204$.

1.4. Перевести число Z, що задане шістнадцятирічним кодом, в його десятковий еквівалент:

a) $Z_{(16)} = 1A64$,
b) $Z_{(16)} = 3FD7$,
c) $Z_{(16)} = 20F1$,
d) $Z_{(16)} = B165$,
e) $Z_{(16)} = CC27$,
f) $Z_{(16)} = CB13$.

1.5. Перевести число C, задане двійково-десятковим кодом, в його десятковий еквівалент:

a) $C_{(2\text{-}10)} = 100101110101$,
b) $C_{(2\text{-}10)} = 000110000100$,
c) $C_{(2\text{-}10)} = 011010010101$,
d) $C_{(2\text{-}10)} = 011101110111$,
e) $C_{(2\text{-}10)} = 010010010010$,
f) $C_{(2\text{-}10)} = 010101010101$.

1.6. Використовуючи метод зсуву-корекції, подати двійково-десяткове число B, задане в коді «8421», у двійковій системі числення:

a) $B_{(8421)} = 275$,
b) $B_{(8421)} = 133$,
c) $B_{(8421)} = 117$,
d) $B_{(8421)} = 151$.

1.7. Використовуючи метод зсуву-корекції, подати двійкове число C у двійково-десятковій системі числення у коді «8421»:

a) $C_{(2)} = 111000111$,
b) $C_{(2)} = 101011011$,
c) $C_{(2)} = 100111001$,
d) $C_{(2)} = 100111101$.

1.8. Подати задане число A у прямому, оберненому і доповнювальному двійкових кодах в 11-розрядній сітці (з урахуванням знакового розряду):

a) $A_{(10)} = -19/32$,
b) $A_{(10)} = -17/32$,
c) $A_{(10)} = -12/16$,
d) $A_{(10)} = -15/16$.

1.9. Задане двійкове число B

a) $B = -0{,}1010001$,
b) $B = -0{,}1100101$,
c) $B = -0{,}1010101$,
d) $B = -0{,}1011101$,

подати в двійкових кодах:

e) прямому,
f) оберненому,
g) доповнювальному.

Виконати арифметичний зсув отриманого двійкового числа:

h) вліво на 2 розряди і вправо на 3 розряди,
i) вліво на 3 розряди і вправо на 2 розряди,
j) вліво на 3 розряди і вправо на 1 розряд.

1.10. Виконати арифметичний зсув

a) вліво на три розряди і вправо на один розряд,
b) вліво і вправо на один розряд,
c) вліво на три розряди і вправо на один розряд,

мантиси двійкового числа B, поданої у оберненому та доповнювальному двійкових кодах:

d) $B = -0{,}10100$,
e) $B = -0{,}11011$,
f) $B = -0{,}10111$,
g) $B = -0{,}10101$.

1.11. Подати двійкове число X у формі з плаваючою комою без погрішності, використовувати зміщений порядок і схований розряд мантиси. Вибрати й обґрунтувати розрядність порядку і мантиси.

a) $X = 1011{,}10101001$,
b) $X = 101{,}11010101$,
c) $X = 10011{,}10101001$,
d) $X = 101{,}110001$,
e) $X = 11001{,}001001$,
f) $X = 1001{,}101001$.

Б4-1.2. Додавання чисел у машинних кодах

2.1. Виконати додавання двійкових чисел A і B у оберненому та доповнювальному двійкових кодах:

a) $A = 0{,}110101$, $B = -0{,}111011$,
b) $A = -0{,}101011$, $B = 0{,}110001$,
c) $A = -0{,}110011$, $B = -0{,}101011$,
d) $A = 0{,}1111$, $B = -0{,}1101$.

2.2. Проілюструвати етапи додавання двійкових чисел A і B, поданих у формі з плаваючою комою:

a) $A = 2^2\ 0{,}110101$, $B = 2^{-2}\ (-0{,}111011)$,
b) $A = 2^{-2}\ (-0{,}101011)$, $B = 2^3\ 0{,}110001$,
c) $A = 2^{-1}\ (-0{,}110011)$, $B = 2^{-2}\ (-0{,}101011)$,
d) $A = 2^{-1}\ 0{,}1111$, $B = 2^2\ (-0{,}1101)$.

2.3. Подати операційну схему, змістовний мікроалгоритм і цифрову діаграму стану регістрів при обчисленні функції

a) $D = 0{,}5C - 2AB$, якщо $A = 3$, $B = 4$ і $C = 42$,
b) $D = 2AB + 2C$, якщо $A = 13$, $B = 2$ і $C = 5$,
c) $D = 4AB + 0{,}5C$, якщо A = 5, $B = 4$ і $C = 5$.

Б4-1.3. Операції множення та ділення двійкових чисел

3.1. Подати операційну схему, змістовний мікроалгоритм і цифрову діаграму стану регістрів при множенні двійкових чисел A і B:

a) $A = 1011$, $B = 1101$,
b) $A = 1001$, $B = 1101$,
c) $A = 1011$, $B = 1110$,
d) $A = 11001$, $B = 10101$,
e) $A = 11100$, $B = 10111$,
f) $A = 10001$, $B = 11011$.

Обрати спосіб множення чисел:

g) перший спосіб,
h) другий спосіб,
i) третій спосіб,
j) четвертий спосіб.

2.17. Представити операційну схему, змістовний мікроалгоритм і цифрову діаграму стану регістрів при діленні числа X на Y:

a) $X = 0{,}1011$, $Y = 0{,}1101$,
b) $X = 0{,}1001$, $Y = 0{,}1101$,
c) $X = 0{,}1001$, $Y = 0{,}1011$,
d) $X = 0{,}1010$, $Y = 0{,}1011$,
e) $X = 0{,}1100$, $Y = 0{,}1001$,
f) $X = 0{,}1110$, $Y = 0{,}1010$.

Обрати спосіб ділення чисел:

g) спосіб ділення зі зсувом дільника,
h) спосіб ділення зі зсувом залишку.

Б4-2. Приклади завдань до модульного контролю

Варіант 1

1. Подати двійкове число $X = 101011{,}11001$ у десятковій системі числення.

2. Подати число $A = -12/32$ у двійковому вигляді в прямому, оберненому і доповнювальному машинних кодах в 10-розрядній сітці (з урахуванням знакового розряду).

3. Використовуючи метод зсуву-корекції, перетворити двійково-десяткове число $A = 117$, подане в коді «8421», на двійкове число.

4. Розробити операційну схему пристрою для множення чисел першим способом та змістовний мікроалгоритм виконання операції. Подати цифрову діаграму стану регістрів для значень аргументів $X = 0.1001$ та $Y = 0{,}1101$.

5. Подати двійкове число $B = 1011{,}10010011$ у формі з плаваючою комою без похибки. Обрати й обґрунтувати розрядність порядку і мантиси.

Варіант 2

1. Подати десяткове число B = 325,257 у двійковій системі числення.

2. Виконати арифметичний зсув вліво на 2 розряди і вправо на 3 розряди двійкового числа –0,10101, поданого в доповнювальному коді.

3. Виконати додавання чисел 0,101101 та –0,110001 в оберненому та доповнювальному кодах.

4. Розробити операційну схему пристрою для обчислення функції $D = 2C + 4AB$ та змістовний мікроалгоритм обчислення. Подати цифрову діаграму стану регістрів для значень аргументів $A = 1011$, $C = 1011$, $B = 0100$.

5. Проілюструвати етапи додавання чисел, поданих у форматі із плаваючою комою, якщо $X = 2^3 \cdot 0{,}10011$ і $X = 2^{-1} \cdot 0{,}10111$.

Варіант 3

1. Виконати операції машинного додавання і віднімання в оберненому і доповнювальному кодах для двох чисел A = – 127 і B = 234.

2. Проілюструвати етапи додавання чисел, поданих у форматі із плаваючою комою, якщо $X = 2^2 \cdot 0{,}11011$ і $Y = 2^{-2} \cdot (-0{,}10101)$.

3. Розробити операційну схему пристрою для виконання операції ділення способом із зсувом дільника та змістовний мікроалгоритм операції. Подати цифрову діаграму стану регістрів для значень аргументів $X = 0.1001$, $Y = 0{,}1101$.

4. Виконати арифметичний зсув вліво і вправо мантиси 0,10101, поданої в доповнювальному і прямому коді.

5. Використовуючи метод зсуву-корекції перетворити двійкове число $A_2 = 11100011$ на двійково-десяткове в коді «8421».

Варіант 4

1. Виконати операції машинного додавання і віднімання в оберненому і доповнювальному кодах для двох чисел $A = -127$ і $B = 234$. Визначити мінімальну довжину розрядної сітки для правильного одержання результату.

2. Проілюструвати етапи додавання чисел, поданих у форматі із плаваючою комою, якщо $X = 2^{1} \cdot 0{,}11011$ і $Y = 2^{-2} \cdot (-0{,}10101)$.

3. Розробити операційну схему пристрою для виконання операції ділення способом із зсувом дільника та змістовний мікроалгоритм операції. Подати цифрову діаграму стану регістрів для значень аргументів $X = 0.1011$, $Y = 0{,}1101$.

4. Виконати арифметичний зсув вліво і вправо мантиси $-0{,}10101$, поданої в додатковому і прямому коді.

5. Використовуючи метод зсуву-корекції, перетворити двійкове число $A = 11100011$ на двійково-десяткове в коді «8421».

Розділ В

Модуль ІІІ. ПРОЕКТУВАННЯ ЦИФРОВИХ АВТОМАТІВ

В1. ВИКОНАННЯ КУРСОВОЇ РОБОТИ

В1-1. Загальні положення

Курсова робота з дисципліни «Прикладна теорія цифрових автоматів» виконується за індивідуальним завданням і є самостійною роботою студента. Вона призначена для розширення, закріплення, узагальнення і практичного застосування знань, умінь і навичок, отриманих студентом під час вивчення курсу. Виконання курсової роботи є важливим етапом у підготовці до виконання дипломного проекту (роботи) майбутнього фахівця з комп'ютерної інженерії.

Курсова робота має дві структурні частини, що відповідають матеріалам першого і другого модулів дисципліни. Перша частина курсової роботи стосується синтезу комбінаційних схем, метою виконання другої частини є побудова операційного пристрою, що складається з операційного та керуючого автомату.

Виконання курсової роботи рекомендується здійснювати студентом упродовж усього семестру. Контроль за виконанням першої і другої частини курсової роботи виконується наприкінці відповідного модуля, оформлення і захист курсової роботи здійснюється наприкінці семестру.

Оформлення курсової роботи здійснюється відповідно до вимог і правил, установлених Єдиною системою конструкторської документації (ЄСКД).

У процесі виконання курсової роботи студент повинен вивчити принципи проектування керуючих автоматів та методи виконання основних арифметичних операцій в ЕОМ.

У результаті виконання курсової роботи студент має вміти виконувати структурний синтез керуючих автоматів методом декомпозиції тригерів, мінімізувати системи перемикальних функцій, будувати комбінаційні схеми у заданому елементному базисі. Окрім того студент повинен навчитися використовувати довідкову літературу і вивчити процес створення проектно-конструкторської документації відповідно до чинних стандартів.

B1-2. Завдання до виконання курсової роботи

1. Синтез комбінаційних схем

1.1. Побудувати комбінаційні схеми, що реалізують перемикальні функції в заданому елементному базисі.

2. Синтез операційного та управляючого автомату

2.1. Розробити операційну схему обчислювального пристрою, що реалізує задану арифметичну операцію.

2.2. Виконати синтез і побудувати функціональну схему операційного і управляючого автомата, що забезпечує управління обчислювальним пристроєм.

Варіант завдання визначається десятьма молодшими розрядами номера залікової книжки студента, представленого в двійковій системі числення (*h*10, *h*9, ..., *h*1).

Наприклад: номер залікової книжки студента 435206 перекладаємо у двійкову систему числення і отримуємо двійкове число:

$$435206_{(10)} = 1101010010000000110_{(2)}.$$

Десять молодших розрядів отриманого двійкового числа, поданих у табл. B1-2.1, застосовуємо для визначення варіанта завдання в табл. B1-2.2 — табл. B1-2.6.

Таблиця B1-2.1

ВИЗНАЧЕННЯ ВАРІАНТА ЗАВДАННЯ

0	0	0	0	0	0	0	1	1	0
*h*10	*h*9	*h*8	*h*7	*h*6	*h*5	*h*4	*h*3	*h*2	*h*1

Відповідно до визначеного варіанта та таблиці істинності (табл. B1-2.2) визначається вихідна система з чотирьох перемикальних функцій. Набір логічних елементів, які можна використовувати для побудови комбінаційних схем, надані в табл. B1-2.3.

Тип арифметичної операції (множення, ділення або обчислювання функції), що реалізує обчислювальний пристрій, визначається відповідно до табл. B1-2.4. Під час проектування вважати, що операнди під час виконання операцій множення та ділення мають 8 основних розрядів, а під час обчислення функцій — 4 розряди.

Тип керуючого автомата і тип тригерів, які застосовуються під час синтезу автомата визначаються відповідно до табл. B1-2.5 і табл. B1-2.6. Для побудови комбінаційних схем, що входять до складу автомата, можна використовувати будь-яки елементи.

Таблиця В1-1.2

ТАБЛИЦЯ ІСТИННОСТІ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

x_4	x_3	x_2	x_1	f_1	f_2	f_3	f_4
0	0	0	0	1	1	1	0
0	0	0	1	1	1	0	1
0	0	1	0	1	1	1	*h*3
0	0	1	1	0	0	0	*h*4
0	1	0	0	–	0	1	0
0	1	0	1	0	0	0	*h*5
0	1	1	0	1	–	–	0
0	1	1	1	–	–	1	*h*6
1	0	0	0	1	*h*4	*h*7	*h*7
1	0	0	1	0	0	*h*8	1
1	0	1	0	0	0	*h*9	*h*8
1	0	1	1	*h*1	0	0	*h*2
1	1	0	0	1	–	1	1
1	1	0	1	*h*2	*h*5	0	*h*9
1	1	1	0	*h*3	*h*6	*h*10	*h*1
1	1	1	1	1	1	1	1

Таблиця В1-2.3

ЕЛЕМЕНТНИЙ БАЗИС

*h*10	*h*9	*h*8	Логічні елементи
0	0	0	3І-НЕ, 2І
0	0	1	3І, 4І-НЕ
0	1	0	3АБО, 4І, НЕ
0	1	1	3І, 2АБО, НЕ
1	0	0	2АБО-НЕ, 4І
1	0	1	2І-НЕ, 4АБО
1	1	0	3АБО-НЕ, 3І
1	1	1	3І-НЕ, 3АБО-НЕ

Таблиця B1-2.4

ВИБІР ТИПУ КЕРУЮЧОГО АВТОМАТА

*h*7	Тип автомата
0	Мілі
1	Мура

Таблиця B1-2.5

ВИБІР АРИФМЕТИЧНОЇ ОПЕРАЦІЇ

*h*4	*h*3	*h*2	*h*1	Множення
0	0	0	0	1-м способом
0	0	0	1	2-м способом
0	0	1	0	3-м способом
0	0	1	1	4-м способом
*h*4	*h*3	*h*2	*h*1	Ділення
0	1	0	0	1-м способом
0	1	0	1	2-м способом
*h*4	*h*3	*h*2	*h*1	Обчислення функції
0	1	1	0	$D = 2C - 4AB$
0	1	1	1	$D = A(B - 1) + 0{,}5C$
1	0	0	0	$D = 2A(B + 1) + 0{,}5C$
1	0	0	1	$D = A(B + 1) + 2C$
1	0	1	0	$D = C + 2AB$
1	0	1	1	$D = AB + 0{,}5C$
1	1	0	0	$D = 2A(B + 1) + C$
1	1	0	1	$D = A(B - 1) + 2C$
1	1	1	0	$D = A(B + 1) + 0{,}5C$
1	1	1	1	$D = 2A(B - 1) + C$

Таблиця B1-2.6

ВИБІР ТИПУ ТРИГЕРУ

*h*6	*h*5	Тип тригеру
0	0	*RS*
0	1	*D*
1	0	*JK*
1	1	*T*

В1-3. Зміст курсової роботи

Курсова робота оформлюється у вигляді комплекту текстових і графічних документів, що відповідає вимогам чинних стандартів щодо правил оформлення конструкторської документації.

За чинними стандартами курсова робота має містити такі текстові й графічні документи (документи наведені у порядку їх комплектування):

— титульний аркуш;

— опис альбому;

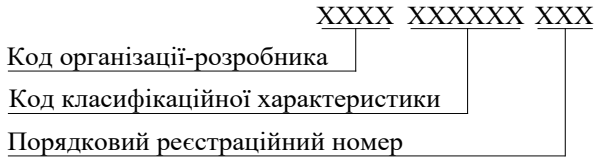
— технічне завдання;

— керуючий автомат. Схема електрична функціональна;

— пояснювальна записка.

Укомплектовані документи слід виконати на аркушах формату А4 і скріпити в один альбом.

Уся технічна документація повинна мати позначення. Структуру позначення для основних конструкторських документів встановлено відповідно до чинних стандартів. Під час виконання курсової роботи прийнято умовну структуру позначення основних конструкторських документів:



Код організації розробника для студентів, котрі виконують курсову роботу, визначається скороченою назвою університету.

Перші дві цифри коду класифікаційної характеристики відображують рік виконання курсової роботи, наступні — номер залікової книжки студенту.

Порядковий реєстраційний номер присвоюють документам у межах одного альбому.

Позначення конструкторських документів, складається із позначення основного документа і коду документа, встановленого відповідними стандартами. Конструкторські документи, з яких складається курсова робота, мають такі коди: опис альбому — ОА, технічне завдання — ТЗ, пояснювальна записка — ПЗ та схема електрична функціональна — Е2.

Приклади позначень, прийняті для позначення конструкторських документів за виконання курсової роботи:

— позначення технічного завдання

НАУ 06 4512 002 ТЗ,

КПІ 06 3510 002 ТЗ;

— позначення схеми електричної функціональної

НАУ 06 4512 003 Е2,

КПІ 06 3510 003 Е2.

Зразок оформлення *титульного аркуша* наведений у додатку 3.1.

Опис альбому містить перелік усіх документів альбому. Приклад оформлення опису альбому наведений у додатку 3.2.

Технічне завдання розробляється студентом на підставі вихідних даних відповідно до чинного стандарту. У технічному завданні мають бути відображені такі розділи.

1. Вступ.
2. Призначення розроблювального об'єкта, у якому розкриваються його області застосування.
3. Вихідні дані для розроблення проектованого пристрою.
4. Етапи проектування і терміни їх виконання.
5. Перелік текстової і графічної документації.

Технічне завдання має бути підписане виконавцем і керівником проекту.

Пояснювальна записка має містити такі розділи.

Вступ.

1. Синтез комбінаційних схем.
2. Розроблення операційної схеми обчислювального пристрою.
3. Синтез керуючого автомата.

Висновки.

Перелік літератури.

У *вступі* вказується, на підставі яких документів (вихідних даних) здійснюється розроблення.

Перший розділ присвячений синтезу комбінаційних схем. У цьому розділі необхідно виконати такі завдання:

1.1. Функцію f_4 зобразити в канонічних формах алгебри Буля, Жегалкіна, Пірса і Шеффера.

1.2. Визначити належність функції f_4 до п'яти передповних класів.

1.3. Виконати мінімізацію функції f_4 методами:

— Квайна (Квайна — Мак-Класки);

— Діаграм Вейча.

1.4. Виконати спільну мінімізацію перемикальних функцій f_1, f_2, f_3 та окремо їх заперечень методом Квайна (Квайна—Мак-Класкі).

1.5. Отримати вісім нормальних форм подання заданої системи перемикальних функцій.

1.6. Одержати операторні подання перемикальних функцій для реалізації системи функцій у заданому елементному базисі.

1.7. Побудувати комбінаційні схеми системи перемикальних функцій.

1.8. Визначити параметри комбінаційних схем (складність і швидкодію).

У *другому розділі* необхідно виконати:

2.1. Розробити операційну схему обчислювального пристрою, що реалізує задану арифметичну операцію.

2.2. Розробити змістовний мікроалгоритм виконання заданої арифметичної операції.

2.3. Подати цифрову діаграму стану вузлів операційного пристрою в кожному такті для конкретних значень операндів.

У *третьому розділі* необхідно виконати:

3.1. Зобразити закодовану графічну схему алгоритму (ГСА).

3.2. Виконати розмітку станів автомата.

3.3. Виконати синтез автомата.

3.4. Виконати спільну мінімізацію функцій збудження тригерів і функцій вихідних сигналів автомата.

3.5. Побудувати функціональну схему управляючого автомата.

У *висновку* необхідно узагальнити основні результати роботи.

В1-4. Правила виконання функціональних схем

Функціональна схема належить до графічної документації.

Функціональну схему зображують на окремому аркуші (формату А3 або А4) за правилами виконання схем електричних функціональних.

Усі графічні документи повинні мати відповідні позначення.

Графічна документація супроводжується основним написом. Основні написи розміщують у правому нижньому куті документа, а на аркушах формату А4 тільки по короткій стороні аркуша, тобто формат А4 має завжди вертикальне положення.

Приклади основних написів перших та наступних аркушів схем та креслень, якими супроводжуються графічна документація за виконання курсової роботи, наведені у додатку 3.3.

Функціональна схема визначає основні функціональні частини виробу, їхнє призначення і взаємозв'язок, роз'ясняє визначені процеси, що протікають в окремих функціональних частинах чи у виробі в цілому. Приклад функціональної схеми керуючого автомату наведений у додатку 4.

Функціональні схеми можуть виконуватися як на весь виріб, так і на його окремі функціональні частини. Функціональні частини таких схем зображуються, як правило, у вигляді прямокутників.

Допускається використання у функціональних схемах умовних графічних позначень (УГП) деяких функціональних частин, наприклад, комбінаційних елементів, суматорів, дешифраторів, елементів пам'яті і таке інше.

Умовне графічне позначення елемента має форму прямокутника, до якого підведені лінії виводів. У загальному випадку УГП елемента містить три поля: одне основне і два додаткових, що розташовані ліворуч і праворуч від основного. В основному полі УГП розміщують позначення функції, що виконується елементом. У додаткових полях наводять інформацію про призначення входів та виходів елемента. В різних випадках УГП може складатися тільки з основного поля або з основного та одного додаткового полів, розташованих праворуч або ліворуч від основного поля.

Додаткові поля допускається розділяти на зони, що відокремлюються горизонтальною рискою.

Входи елемента зображують з лівої сторони УГП, виходи — з правої сторони.

При підведенні ліній до контуру УГП *не допускається* проводити лінії на рівні сторін прямокутника та вказувати на них стрілками у контурі УГП напрямок передачі інформації.

Розміри сторін УГП мають бути кратні множникові M, тоді відстані між виводами кратні — $2M$. Діаметр покажчика інверсного виводу має дорівнювати M. Значення множника M обирається виходячи з вимог до мікрофільмування. Для виконання курсової роботи прийняти $M = 2{,}5$ мм.

Позначення функцій, які виконує елемент, утворюють із прописних букв латинського алфавіту, арабських цифр і спеціальних знаків, записаних без пробілів. Кількість знаків у позначенні функції не обмежено, однак варто прагнути до їхньої мінімальної кількості за збереження однозначності розуміння кожного позначення.

Стандартні позначення деяких функцій елементів, яких варто дотримуватись під час виконання курсової роботи, наведені у табл. В1-4.1.

Виводи елементів, що несуть логічну інформацію, підрозділяють на статичні та динамічні, а також на прямі та інверсні. Функціона-

льне призначення виводів елемента позначають за допомогою міток виводів (табл. В1-4.2).

Таблиця В1-4.1

ПОЗНАЧЕННЯ ФУНКЦІЙ ЕЛЕМЕНТІВ

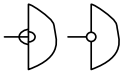
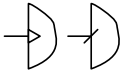
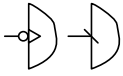
Найменування	Позначення
Тригер	*T*
Двохрівневий тригер	*TT*
Логічне «І»	&
Логічне «АБО»	1
Виключне «АБО»	= 1
Генератор імпульсів	*G*

Мітки виводів елементів утворюють із прописних букв латинського алфавіту, арабських цифр і спеціальних знаків, записаних в одному рядку без пробілів.

Приклади УГП деяких елементів наведені в табл. В1-4.3. Приклади УГП логічних елементів представлені у табл. А1-1.4 — А1-1.5, приклади УГП типових вузлів наведені у відповідних пунктах розділу А1-4.

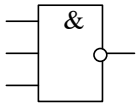
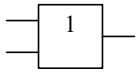
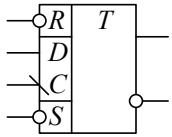
Таблиця В1-4.2

ПОЗНАЧЕННЯ ВИВОДІВ ЕЛЕМЕНТІВ

Найменування	Позначення
Прямий статичний вхід	
Інверсний статичний вхід	
Прямий динамічний вхід	
Інверсний динамічний вхід	

Таблиця В1-4.3

ПРИКЛАДИ УМОВНИХ ГРАФІЧНИХ ПОЗНАЧЕНЬ ЕЛЕМЕНТІВ

Найменування елемента	УГП елемента
Елемент 3І-НЕ	
Елемент 2АБО	
D-тригер	

В1-5. Вимоги до оформлення текстових документів

Пояснювальна записка та технічне завдання належать до текстової конструкторської документації.

Порядок побудови розділів і підрозділів текстових документів, правила викладу тексту, розрахунків, а також побудови таблиць мають повністю відповідати вимогам чинних стандартів щодо правил оформлення конструкторської документації.

Текст виконується авторським рукописом чорним чорнилом чи роздруковується на принтері.

Усі текстові документи повинні мати відповідні позначення.

Текстова документація супроводжується основним написом. Основні написи розміщують у правому нижньому куті документа.

Приклади основних написів для перших та наступних аркушів текстових документів, якими супроводжуються текстова документація під час виконання курсової роботи, наведені у додатку 3.3.

Перед комплектацією всі документи курсової роботи повинні бути підписані виконавцем і керівником на титульному аркуші та в основних написах текстових і графічних документів.

Підпис керівника про допуск до захисту курсової роботи ставиться після остаточного оформлення альбому.

В1-6. Захист курсових робіт

Захист курсових робіт є особливою формою перевірки якості виконання роботи і знань студента в цій області.

Захист проводиться перед комісією з викладачів. Захист складається з короткої доповіді (5—8 хв.) студента за результатами виконаної роботи і відповідей на запитання.

Якщо в конструкторській документації проекту будуть виявлені грубі порушення стандартів чи виявиться, що спроектований виріб принципово непрацездатний, робота повертається на доопрацювання.

Студент, котрий не виконав у встановлений термін курсової роботи, має академічну заборгованість.

Перелік літератури

1. *Бабич М.П., Жуков І.А.* Атестаційні роботи магістрів і спеціалістів: Навч.-метод. посіб. — К.: НАУ, 2004. — 216 с.

2. *Бабич М.П., Жуков І.А.* Комп'ютерна схемотехніка: Навч. посіб. — К.: МК-Прес, 2004. — 412 с.

3. *Букреев И.Н., Мансуров В.М., Горячев В.И.* Микроэлектронные схемы цифровых устройств.– М.: Сов. радио, 1975. — 368 с.

4. *Жабин В.И.* Алгоритм неавтономного вычисления квадратного корня на цифровом устройстве, работающем в реальном времени // Автоматика и вычислительная техника. — 1977. — № 1. — С. 83—84.

5. *Жабин В.И., Корнейчук В.И., Тарасенко В.П.* Методы вычисления некоторых функций при поразрядном вводе и выводе информации // Известия ВУЗов. Машиностроение — 1978. — №2. — С. 64—69.

6. *Жабін В.І., Клименко І.А., Ткаченко В.В.* Прикладна теорія цифрових автоматів: Метод. вказівки до виконання курсової роботи. — К.: НАУ, 2004. — 54 с.

7. *Жабін В.І., Ткаченко В.В.* Цифрові автомати: Практикум. — К.: ВЕК +, 2004. — 160 с.

8. *Зиссос Д.* Проектирование систем на микропроцессорах / Пер. с англ. под ред. А.И. Петренко. — К.: Техніка, 1982. — 176 с.

9. *Карцев М.А.* Арифметика цифровых машин. — М.: Наука, 1969. — 576 с.

10. *Корнейчук В.И., Тарасенко В.П.* Основы компьютерной арифметики. — К.: Корнійчук, 2003. — 176 с.

11. *Корнейчук В.И., Тарасенко В.П., Жабин В.И.* Способ быстрого умножения чисел, представленных последовательным кодом // Автоматика и вычислительная техника. — 1975. — № 5. — С. 82—86.

12. *Майоров С.А., Новиков Г.И.* Принципы организации цифровых машин. — Л.: Машиностроение, 1977. — 432 с.

13. *Папернов А.А.* Логические основы цифровых машин и программирования. — М.: Наука, 1968. — 591 с.

14. *Поспелов Д.А.* Логические методы анализа и синтеза схем. — М.: Энергия, 1974. — 367 с.

15. *Прикладна* теорія цифрових автоматів: Метод. вказівки до виконання лабораторних робіт / І.А. Жуков, В.І. Жабін., І.А. Клименко, В.В. Ткаченко. — К.: НАУ, 2005. — 56 с.

16. *Прикладна* теорія цифрових автоматів: Метод. вказівки до виконання лабораторних робіт / І.А. Жуков, В.І. Жабін., І.А. Клименко, В.В. Ткаченко. — К.: НАУ, 2005. — 54 с.

17. *Прикладная* теория цифровых автоматов / К.Г. Самофалов, А.М. Романкевич, В.Н. Валуйский и др. — К.: Вища шк., 1987. — 375 с.

18. *Проектирование* цифровых вычислительных машин: Учеб. пособие для студентов вузов / Под ред. С.А. Майорова. — М.: Высшая шк., 1972. — 344 с.

19. *Савельев А.Я.* Арифметические и логические основы цифровых автоматов: Учебник. — М.: Высш. шк., 1980. — 255 с.

20. *Самофалов К.Г., Корнейчук В.И., Тарасенко В.П.* Цифровые ЭВМ: Теория и проектирование. — К.: Вища шк., 1989. — 427 с.

21. *Цифровые* ЭВМ: Практикум / К.Г. Самофалов, В.И. Корнейчук, В.П. Тарасенко, В.И. Жабин. — К.: Вища шк., 1990. — 215 с.

Додаток 1

Опис програмного комплексу для моделювання логічних схем

Для проведення лабораторних робіт використовуються програмні засоби моделювання цифрових схем.

Програмний комплекс ПРОГМОЛС 2.0 призначений для моделювання процесів у комбінаційних і послідовних схемах. Він дозволяє створювати і редагувати логічні схеми, здійснювати моделювання у синхронному (без урахування затримок сигналів в елементах схеми) і в асинхронному (з урахуванням затримок) режимах, а також зберігати отримані моделі у вигляді файлів на дисках.

Вигляд моделі логічної схеми, побудованої під час застосування програмного комплексу ПРОГМОЛС 2.0, проілюстровано на рис. 1.1.

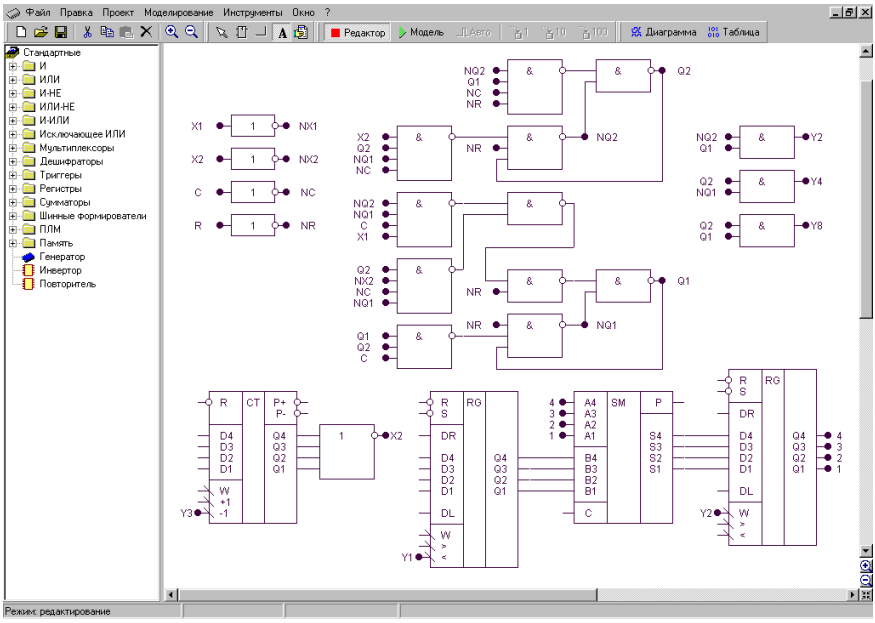


Рис. 1.1. Зображення моделі операційного пристрою

Програмний комплекс включає систему підказок, що полегшує роботу в різних режимах моделювання. Для роботи з програмою використовують систему ієрархічних меню.

Меню містить такі розділи:

— файл;
— виправлення;
— проект;
— моделювання;
— інструменти;
— вікно;
— допомога.

Розглянемо детально розділи меню.

Файл

Створити (сполучення клавіш ***Ctrl* + *N***). Створює новий файл проекту.

Відкрити (*Ctrl* + *O*). Викликає діалогове вікно відкриття проекту.

Зберегти (*Ctrl* + *S*). Зберігає файл проекту на диск під поточним ім'ям.

Зберегти як (*Ctrl* + *A*). Зберігає копію файлу проекту на диск, з вказанням нового ім'я та шляху збереження.

Закрити (*Ctrl* + *F4*). Закриває поточний файл проекту.

Вихід (*Alt* + *F4*). Закриває програму і всі відкриті вікна.

Виправлення

Вирізати (*Ctrl* + *X*). Вирізає фрагмент схеми у внутрішній буфер обміну програми.

Копіювати (*Ctrl* + *C*). Копіює фрагмент схеми у внутрішній буфер обміну програми.

Вставити (*Ctrl* + *V*). Вставляє фрагмент схеми з внутрішнього буфера обміну програми.

Видалити (*Delete*). Видаляє фрагмент схеми.

Проект

Корпус мікросхеми. Показати/сховати корпус мікросхеми поточного проекту.

Редактор корпуса. Викликає діалогове вікно редактора корпуса мікросхеми поточного проекту.

Компіляція (*Alt* + *C*). Компілює поточний проект. Ця команда використовується для відновлення списку змінних у діаграмі і таблиці проекту після зміни схеми без включення режиму моделювання. Під час включення режиму моделювання компіляція здійснюється автоматично.

Додати в бібліотеку. Копіює поточний проект у буфер обміну редактора бібліотек і викликає редактор бібліотек. Для вставки компонента в бібліотеку необхідно викликати команду **Вставити** редактора бібліотек.

Настроювання (*Ctrl* + *F4*). Викликає діалогове вікно настроювання проекту.

Моделювання

Відробити інтервал. Відпрацьовує заданий користувачем інтервал модельного часу.

Відробити до. Відпрацьовує до моменту модельного часу, заданого користувачем.

Генератор (*G*). У синхронному режимі викликає чергову зміну стану генераторів і відпрацьовує схему до закінчення перехідних процесів. В асинхронному режимі відпрацьовує 1 такт модельного часу.

Інструменти

Редактор бібліотек (*Alt* + *L*). Активізує редактор бібліотек.

Настроювання. Викликає діалогове вікно настроювань програми**.**

Діаграма (*Alt* + *D*). Виводить на екран діаграму станів змінних поточного проекту.

Таблиця (*Alt* + *T*). Виводить на екран таблицю станів змінних поточного проекту.

Вікно

Каскадом. Розташовує вікна відкритих проектів каскадом**.**

Розділити по вертикалі. Розташовує вікна відкритих проектів по вертикалі таким чином, щоб вони не перекривалися.

Розділити по горизонталі. Розташовує вікна відкритих проектів по горизонталі таким чином, щоб вони не перекривалися.

Збільшити. Збільшує масштаб у редакторі схеми поточного проекту.

Зменшити. Зменшує масштаб у редакторі схеми поточного проекту.

Допомога

Про програму. Виводить відомості про програму і розробників.

Допомога. Викликає файл довідки.

На окремі панелі винесені елементи керування

Створити (*Ctrl* + *N*). Створює новий файл проекту.

Відкрити (*Ctrl* + *O*). Викликає діалогове вікно відкриття проекту.

Зберегти (*Ctrl* + *S*). Зберігає файл проекту на диск під поточним ім'ям.

Вирізати (*Ctrl* + *X*). Вирізає фрагмент схеми у внутрішній буфер обміну програми.

Копіювати (*Ctrl* + *C*). Копіює фрагмент схеми у внутрішній буфер обміну програми.

Вставити (*Ctrl* + *V*). Вставляє фрагмент схеми з внутрішнього буфера обміну програми.

Видалити (*Delete*). Видаляє фрагмент схеми.

Збільшити. Збільшує масштаб у редакторі схеми поточного проекту.

Зменшити. Зменшує масштаб у редакторі схеми поточного проекту.

Виділити. Дозволяє виділити і перетягнути лівою кнопкою миші фрагмент схеми, а також інвертувати виходи та входи елементу подвійним натисканням.

Елемент. Дозволяє вставити новий елемент у схему.

Лінія. Дозволяє провести зв'язок у схемі.

Змінна. Дозволяє визначити змінну в схемі.

Захоплення. Дозволяє переносити лівою кнопкою миші весь зміст вікна редактора логічних схем.

Редактор. Переводить редактор логічних схем у режим редагування.

Модель. Переводить редактор логічних схем у режим моделювання.

Авто. У натиснутому стані включає режим автоматичного моделювання, а у віджатому – режим покрокового моделювання.

1 такт. Відпрацьовує 1 такт модельного часу.

10 тактів. Відпрацьовує 10 тактів модельного часу.

Діаграма (*Alt* + *D*). Виводить на екран діаграму станів змінних поточного проекту.

Таблиця (*Alt* + *T*). Виводить на екран таблицю станів змінних поточного проекту.

Панель компонентів

Панель компонентів розташована в лівій частині робочої області програми і являє собою ієрархічний список, у якому відображені бібліотеки, підключені до програми, та їх зміст. Кожна бібліотека містить ієрархічну структуру компонентів, подібну до файлової системи. Панель компонентів призначена для вибору компонента і подальшої вставки його в схему.

У загальному випадку для дослідження логічних схем необхідно виконати таку послідовність дій:

1. Створити за допомогою редактора логічну схему на екрані дисплея.
2. Позначити на схемі вхідні та вихідні змінні.
3. Створити заголовок (перелічити вхідні і вихідні змінні) і сформувати послідовність вхідних наборів у таблиці істинності.

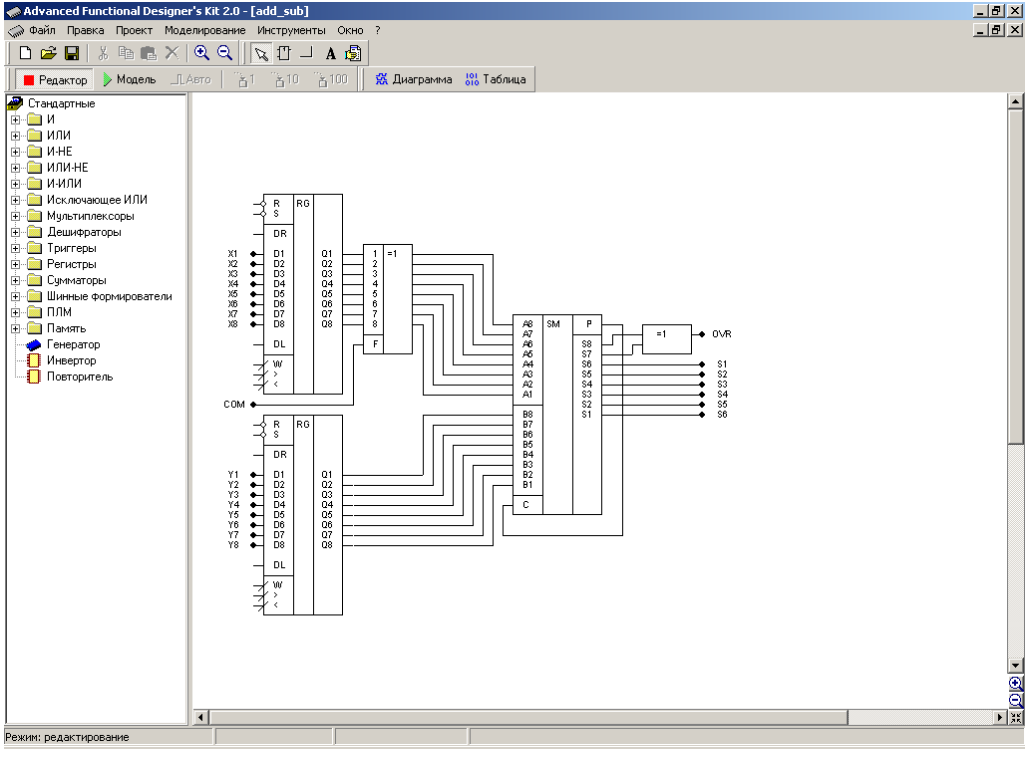
4. Задати необхідні величини затримок сигналів для елементів схеми.

5. Встановити початковий стан схеми.

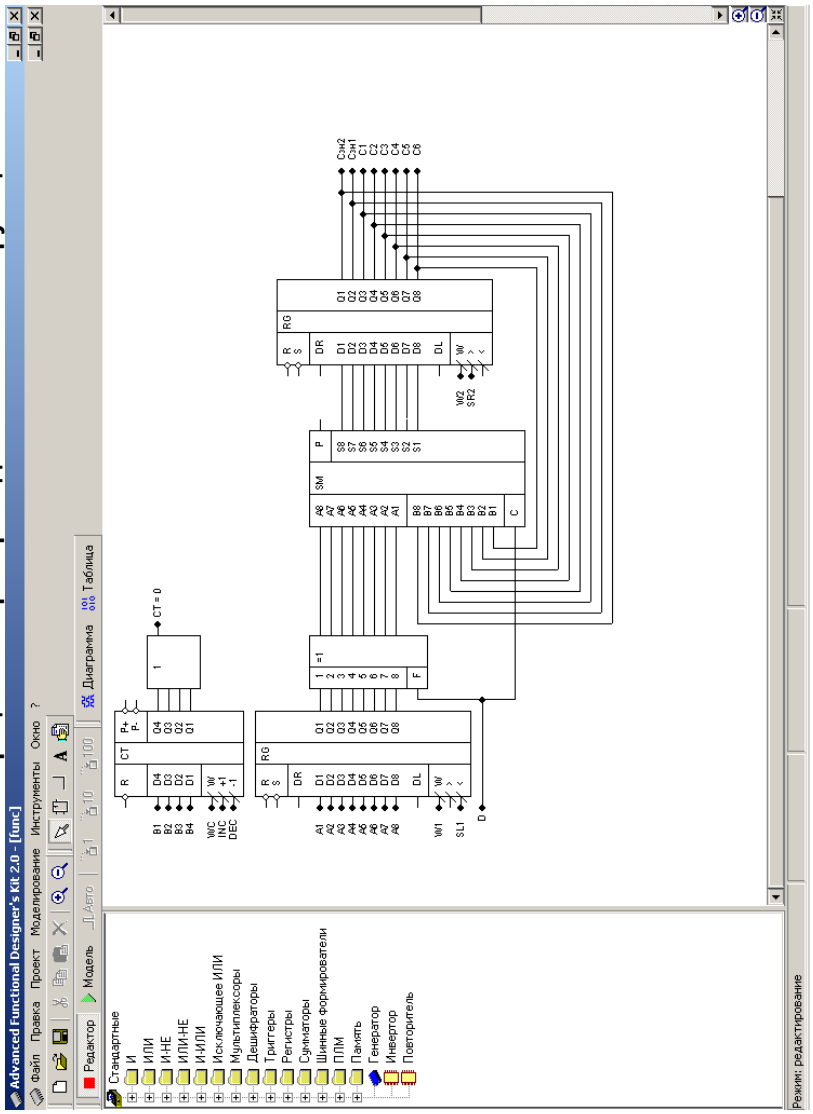
6. Перейти до режиму моделювання схеми.

ПРИКЛАДИ СХЕМ ОПЕРАЦІЙНИХ ПРИСТРОЇВ

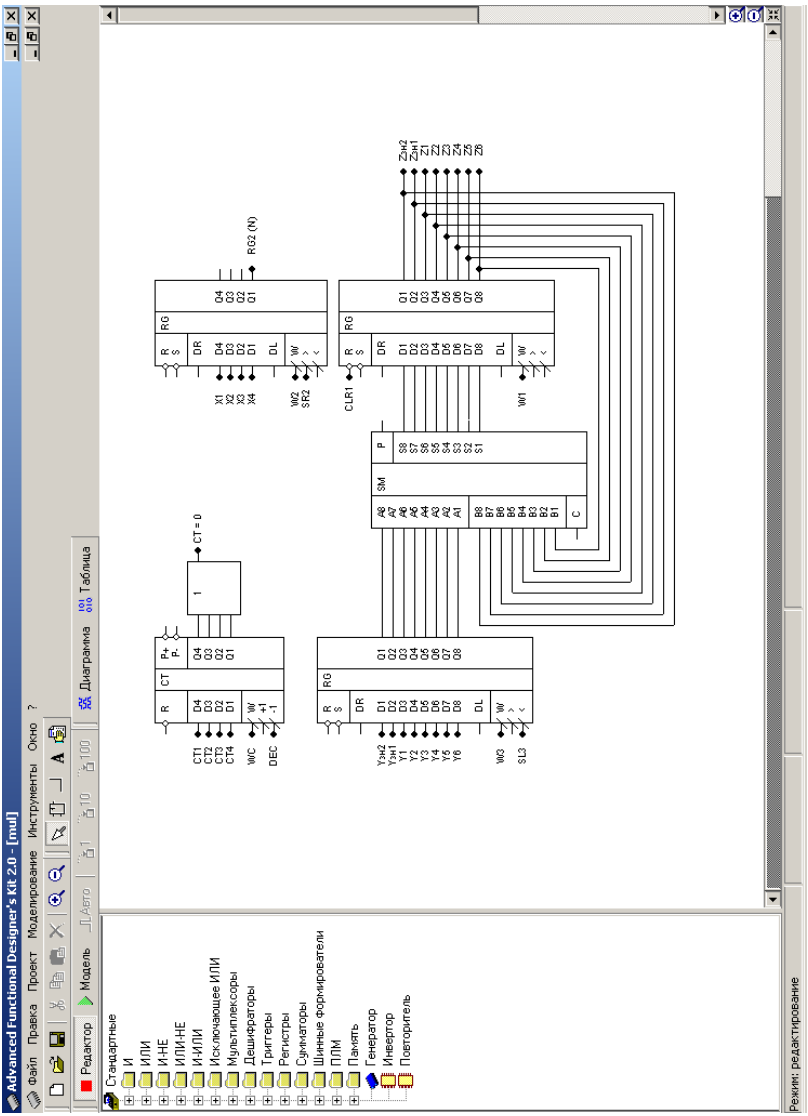
2.1. Схема операційного пристрою для додавання чисел



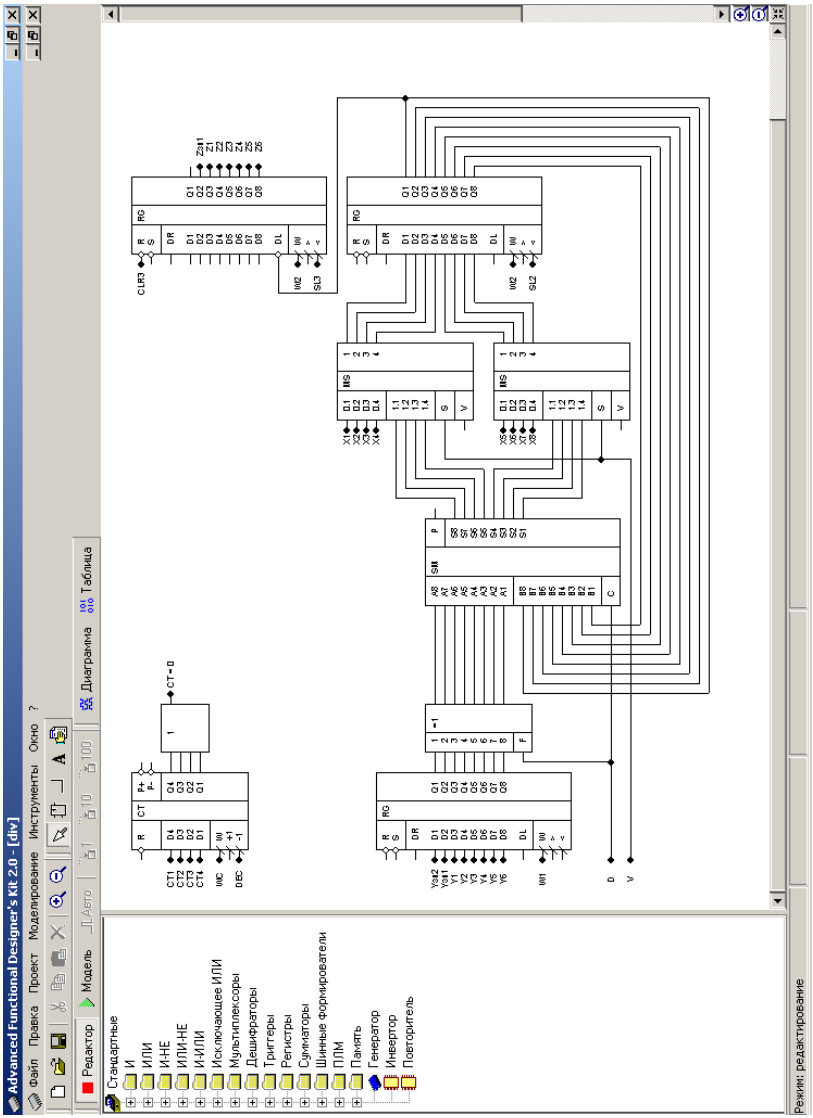
2.2. Схема операційного пристрою для обчислення функції



2.3 Схема операційного пристрою для множення чисел



2.4 Схема операційного пристрою для ділення чисел



Додаток 3

ЗРАЗКИ ОФОРМЛЕННЯ ДОКУМЕНТІВ ДО КУРСОВОЇ РОБОТИ

3.1. Зразок оформлення титульного аркуша курсової роботи

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ
НАЗВА УНІВЕРСИТЕТУ

Назва кафедри

КУРСОВА РОБОТА
з дисципліни
[НАЗВА ДИСЦИПЛІНИ]

Виконав_____

Група _____ Спеціальність _____

Залікова книжка № _____

_____
(допущений до захисту)

_____
(підпис викладача)

_____
(захістив з оцінкою)

_____
(підпис викладача)

2007

3.2. Зразок оформлення опису альбому

№ екз.	Формат	Позначення	Найменування	Кількість аркушів	№ примір.	Примітка
1			Документація загальна			
2						
3			Розроблена заново			
4						
5	А4	НАУ 06 4502 002 Е2	Пристрій управляючий	1		
6			*Схема електрична*			
7			*функціональна*	2		
8						
9	А4	НАУ 06 4502 003 ПЗ	Пристрій управляючий	28		
10			*Пояснювальна записка*			
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						

					НАУ 06 4502 001 ОА			
Зм	Лист	№ докум.	Підпис	Дата				
Виконав		Петров П. П.			Управляючий пристрій *Опис альбому*	Літера	Лист	Листів
Керівник		Іванов І. І.				К П	1	1
Перевірив						ФКС 106 6.091501		
Н. контр.								
Зав. каф.		Васильєв С. С.						

3.3. Основні написи

Основний напис для перших аркушів креслень і схем

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="4" rowspan="4">НАУ 06 4602 002 Е2</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Зм</td><td>Лист</td><td>№ докум.</td><td>Підпис</td><td>Дата</td></tr>
<tr><td colspan="2">Виконав</td><td>Петров П. П.</td><td></td><td></td><td colspan="2" rowspan="3">Пристрій управляючий
Схема електрична функціональна</td><td>Літера</td><td>Лист</td><td>Листів</td></tr>
<tr><td colspan="2">Керівник</td><td>Іванов І. І.</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Консульт.</td><td></td><td></td><td></td><td colspan="3" rowspan="2">ФКС 106
6.091501</td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td><td colspan="2"></td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td><td colspan="2"></td><td colspan="3">Аркуш 1 Аркушів 1</td></tr>
<tr><td colspan="2">Н. контр.</td><td></td><td></td><td></td><td colspan="2" rowspan="2"></td><td colspan="3" rowspan="2"></td></tr>
<tr><td colspan="2">Зав. каф.</td><td>Васильєв С. С.</td><td></td><td></td></tr>
</table>

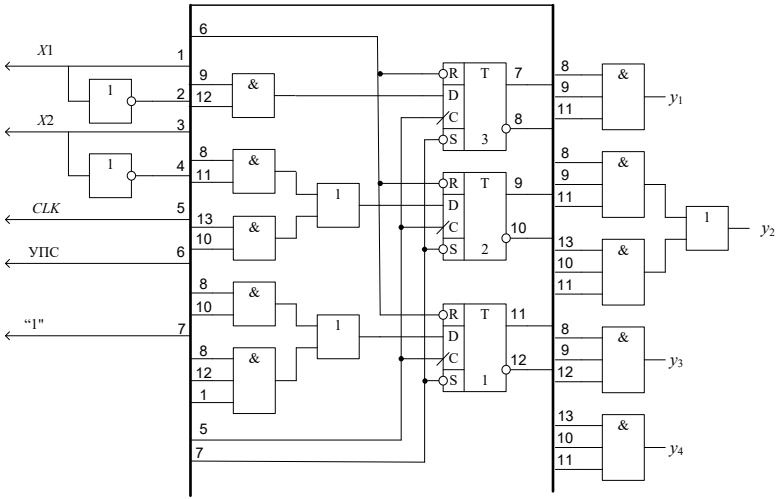
Основний напис для перших аркушів текстових документів

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="4" rowspan="3">НАУ 06 4502 003 ПЗ</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Зм</td><td>Лист</td><td>№ докум.</td><td>Підпис</td><td>Дата</td></tr>
<tr><td colspan="2">Виконав</td><td>Петров П. П.</td><td></td><td></td><td colspan="2" rowspan="4">Пристрій управляючий
Пояснювальна записка</td><td>Літера</td><td>Лист</td><td>Листів</td></tr>
<tr><td colspan="2">Керівник</td><td>Іванов І. І.</td><td></td><td></td><td></td><td>12</td><td>28</td></tr>
<tr><td colspan="2">Перевірив</td><td></td><td></td><td></td><td colspan="3" rowspan="3">ФКС 106
6.091501</td></tr>
<tr><td colspan="2">Н. контр.</td><td></td><td></td><td></td></tr>
<tr><td colspan="2">Зав. каф.</td><td>Васильєв С. С.</td><td></td><td></td></tr>
</table>

Основний напис для наступних аркушів
усіх конструкторських документів

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="2" rowspan="3">НАУ 06 4602 003 ПЗ</td><td>Аркуш</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td rowspan="2">23</td></tr>
<tr><td>Зм</td><td>Лист</td><td>№ докум.</td><td>Підпис</td><td>Дата</td></tr>
</table>

4.1. Функціональна схема управляючого пристрою



					НАУ 06 4502 002 Е2	
Зм	Лит	№ докум.	Підпис	Дата	**Управляючий пристрій** *Схема електрична функціональна*	Літера / Маса / Масштаб
Виконав		Петров П.П.				
Керівник		Іванов І.І.				Аркуш 1 / Аркушів 1
Консульт.						
Н.контр.						ФКС 106 6.091501
Зав.каф.		Васильєв С.С.				