Розділ 8

Пристрій керування

8.1. Функції та методи побудови пристрою керування

Пристрій керування виробляє послідовність сигналів, необхідних для виконання команди, та послідовності команд, тобто програми. Команда в комп'ютері виконується за один або за декілька тактів, в кожному із яких виконується одна або декілька мікроопе-рацій. Кожна мікрооперація представляє собою деяку елементарну дію передачі або перетворення інформації, яка ініціюється поступленням керуючого сигналу (мікронаказу) на вхід керування відповідного пристрою. Прикладом може бути керуючий сигнал, який встановлює або очищує прапорець стану, керуючий сигнал запису до регістра, керуючий код на вході мультиплексора і т. д. Для реалізації команди необхідно на відповідні керуючі входи подати розподілену в часі послідовність керуючих сигналів.

Пристрій керування є одним з вузлів процесора. Як приклад на рис. 8.1 показана взаємодія в процесорі між пристроєм керування та арифметико-логічним пристроєм і регістровою пам'яттю.

Рис. 8.1. Взаємодія пристрою керування з іншими вузлами процесора

Процес функціонування процесора в часі складається з послідовності тактових інтервалів, в яких арифметико-логічний пристрій виконує операції над операндами та видає результати обробки. Виконання даних операцій арифметико-логічний пристрій здійснює на основі відповідних сигналів керування (мікронаказів) з пристрою керування. Послідовність елементарних мікронаказів пристрій керування формує на основі коду операції та службових сигналів стану з регістрової пам'яті процесора.

Відомі два основні методи побудови логіки формування керуючих сигналів. Перший з них виражається в тому, що для кожної команди процесора існує набір логічних схем, які в потрібних тактах збуджують відповідні сигнали керування. Такий принцип керування одержав назву "жорсткої" або "запаяної" логіки.

Другий метод, який дістав назву принципу мікропрограмного керування, передбачає формування керуючих сигналів за вмістом регістра мікрокоманд, в який мікрокоманди записуються із пам'яті мікрокоманд. Шляхом послідовного зчитування мікрокоманд із пам'яті в цей регістр організується потрібна послідовність керуючих сигналів.

Крім пристрою керування процесора в комп'ютері можуть використовуватись пристрої керування вузлами комп'ютера, наприклад, пристрої керування операційними пристроями АЛП, пристрій керування процесора введення-виведення і т. д. Принципи побудови вказаних пристроїв є ідентичними.

8.2. Пристрій керування з жорсткою логікою

8.2.1. Структура пристрою керування з жорсткою логікою

Типова структурна схема пристрою керування з жорсткою логікою (в англійській термінології hardwired control) представлена на рис. 8.2.

Рис. 8.2. Типова структурна схема пристрою керування з жорсткою логікою

До складу пристрою керування входить блок синхроімпульсів, який генерує тактові імпульси ТІ, потрібні для синхронізації роботи пристрою керування, лічильник тактів, в якому зберігається номер виконуваного в даний час такту, дешифратор коду операції та дешифратор тактів, які перетворюють двійковий код в однорядний, логічні схеми формування сигналів керування. Дешифратор коду операції по коду операції із регістра команд РгК формує сигнал активізації мікрооперації МО на відповідній шині. З кожним тактом до лічильника тактів додається +1 сигналом із блоку синхроімпульсів. Дешифратор тактів формує сигнали, відповідні поточному такту.

Логічні схеми формування сигналів керування відповідно до сигналів із дешифратора коду операції, дешифратора тактів та кодів умов і кодів станів із регістра станів формують сигнали керування для виконання необхідних в даному такті мікрооперацій.

Окрім наведених вище компонентів пристрою керування, до його складу входить контролер послідовності сигналів керування, який отримує тактові імпульси з блоку синхроімпульсів, а також код режиму роботи комп'ютера. Він має два окремих режими роботи: звичайний режим та режим запуску комп'ютера. Контролер послідовності сигналів керування є ядром пристрою керування. Принципи його роботи будуть наведені далі при розгляді пристрою мікропрограмного керування.

8.2.2. Методи проектування пристрою керування з жорсткою логікою

Методи проектування пристрою керування з жорсткою логікою, які застосовуються на практиці, часто є спеціально створеними для побудови конкретного пристрою і евристичними за природою, тому не можуть легко бути формалізованими. Для ілюстрації найбільш широко застосовуваних підходів, розглянемо три методи:

- Перший метод – це стандартний алгоритмічний підхід до проектування послідовнісних схем. Його називають методом таблиць станів, оскільки передбачає побудову таблиць станів пристрою керування.
- Другий метод є евристичним і ґрунтується на використанні тактованих елементів часової затримки для побудови часової діаграми керуючих сигналів.
- Спорідненим з другим є третій метод, який передбачає використання лічильників для побудови часової діаграми керуючих сигналів.

Перший метод є найбільш формалізованим і дозволяє застосувати методи мінімізації кількості вентилів та елементів пам'яті. Два інші методи є менш формалізовані і передбачають синтез пристрою керування з часової діаграми сигналів керування.

8.2.3. Пристрій керування на основі таблиць станів

8.2.3.1. Абстрактні автомати

Метод таблиць станів передбачає розгляд пристрою керування як цифрового автомату, тобто логічного пристрою, який забезпечує формування сигналів керування за відповідним алгоритмом з врахуванням своїх внутрішніх станів.

Цифровий автомат можна подати у вигляді його математичної (абстрактної) і структурної моделей, які відповідно називаються абстрактним та структурним автоматами. Абстрактну модель використовують на першому етапі проектування, коли описують функціонування автомату, тобто правила переробки вхідної інформації у вихідну. На цьому етапі автомат подається у вигляді "чорної скриньки". Розгляд абстрактної моделі цифрового автомату дозволяє проводити його попередню оптимізацію ще до етапу структурного синтезу. Структурну модель застосовують для побудови схеми цифрового автомату.

Абстрактним автоматом (математичною моделлю цифрового автомату) називають сукупність з п'яти об'єктів $A = \{X, S, Y, \delta, \lambda\}$, де:

$X = \{x_i\}$, $i \in \overline{1, m}$ – множина вхідних сигналів (вхідний алфавіт);

$S = \{s_j\}$, $j \in \overline{1, n}$ – множина станів (внутрішніх) автомату (алфавіт станів автомату);

$Y = \{y_k\}$, $k \in \overline{1, l}$ – множина вихідних сигналів (вихідний алфавіт);

$\delta :: XxS \to S$ – функція переходів автомату. Функція переходів показує, що автомат, який перебуває у стані s_j при поданні вхідного стану x_i переходить в деякий стан s_p, тобто $s_p = \delta(s_j, x_i)$;

$\lambda :: XxS \to Y$ – функція виходів автомату. Функція виходів λ показує, що автомат, який перебуває у стані s_j, при появі вхідного сигналу x_i видає вихідний сигнал $y_k = \lambda(s_j, x_i)$.

Абстрактний автомат має один вхідний канал і один вихідний канал (рис. 8.3).

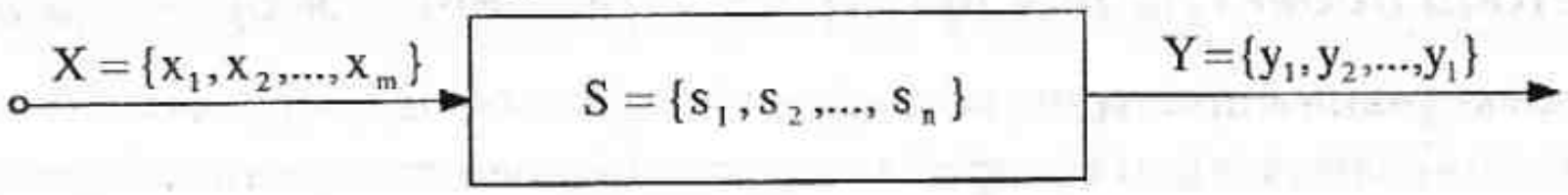

Рис. 8.3. Абстрактний автомат

В подальшому будемо використовувати так званий скінчений абстрактний автомат, в якого множина внутрішніх станів і множина вхідних сигналів (а, отже, й множина вихідних сигналів) є скінченими множинами, повністю визначений (детермінований) абстрактний автомат, в якого функція переходів δ і функція виходів λ визначені для всіх пар (x_i, s_j), та ініціальний абстрактний автомат, в якого один із станів $s_0 \in S$ виділено як початковий стан, з якого автомат завжди починає роботу.

Отже, на абстрактному рівні функціонування цифровий автомат розглядається як перетворювач вхідних слів у вихідні слова, які складаються з букв вхідного і вихідного алфавіту. Внутрішні стани автомату – це інформація про минуле (передісторію) розвитку процесу керування в часі. Вона дозволяє використати час як явну вхідну змінну. Потрібно відзначити, що абстрактний автомат функціонує в дискретному часі, а переходи з одного стану в інший проводяться миттєво.

Залежно від способу генерування значень вихідних сигналів розрізняють три типи автоматів: Мілі, Мура, С-автомат.

Автомат Мілі описується наступною системою рівнянь:

$$y(t) = \lambda\left(x(t)\ s(t)\right)$$

$$s(t+1) = \delta\left(x(t)\ s(t)\right).$$

Автомат Мілі можна представити у вигляді структурної схеми (рис. 8.4), вузли якої представляють відповідно функцію виходів λ, функцію переходів δ і пам'ять станів S, та з'єднані між собою відповідними зв'язками. Значення на його виході в момент часу t визначається значенням в даний момент на його вході та його станом, а також функцією виходів. Стан автомату Мілі в момент часу t+1 визначається значенням в даний момент на його вході та його станом, а також функцією переходів.

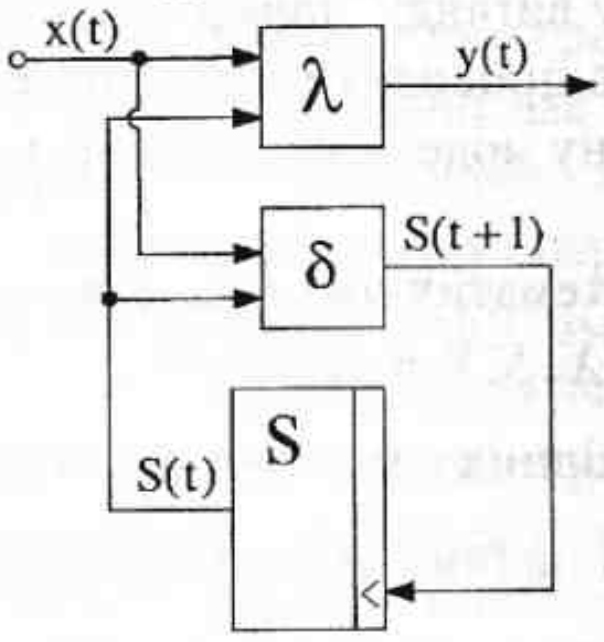

Рис. 8.4. Автомат Мілі

Автомат Мура описується наступною системою рівнянь:

$y(t) = \lambda(s(t))$

$s(t+1) = \delta(x(t)\ s(t))$.

Автомат Мура також можна представити у вигляді структурної схеми (рис. 8.5), вузли якої представляють відповідно функцію виходів λ, функцію переходів δ і пам'ять станів S, та з'єднані між собою відповідними зв'язками.

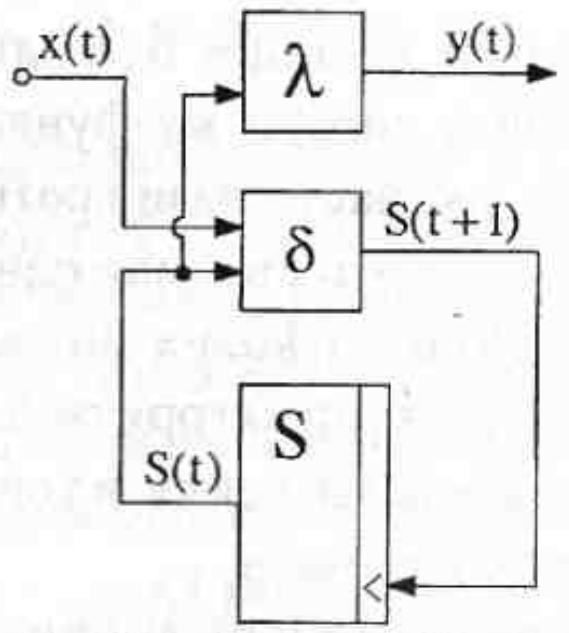

Рис. 8.5. Автомат Мура

Значення на його виході в момент часу t визначається його станом в даний момент, а також функцією виходів. Стан автомату Мура в момент часу t+1 визначається значенням в даний момент на його вході та його станом, а також функцією переходів.

С-автомат описується наступною системою рівнянь:

$y_1(t) = \lambda_1(x(t)\ s(t))$

$y_2(t) = \lambda_2(s(t))$

$s(t+1) = \delta(x(t)\ s(t))$.

С-автомат можна представити у вигляді структурної схеми (рис. 8.6), вузли якої представляють відповідно функції виходів $\lambda 1$ та $\lambda 2$, функцію переходів δ і пам'ять станів S, та з'єднані між собою відповідними зв'язками. Цей автомат має два виходи. Значення на першому його виході в момент часу t визначається значенням в даний момент на його вході та його станом, а також першою функцією виходів. Значення на його другому виході в момент часу t визначається його станом в даний момент, а також другою функцією виходів. Стан автомату в момент часу t+1 визначається значенням в даний момент на його вході та його станом, а також функцією переходів.

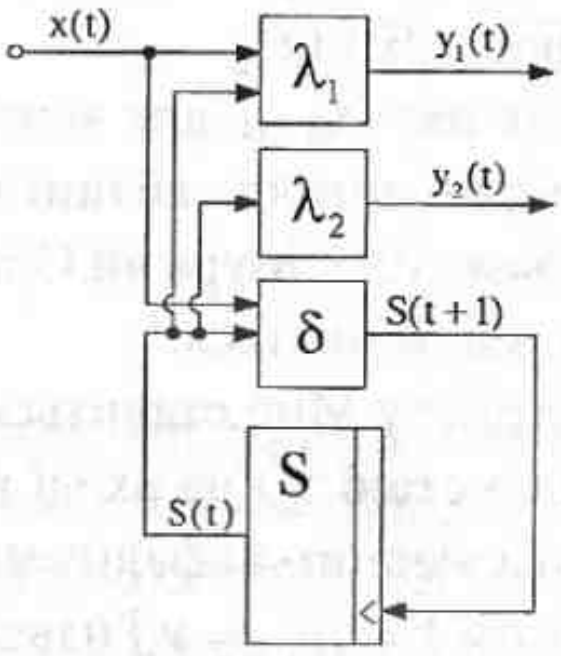

Рис. 8.6. С-автомат

Фактично С-автомат є комбінацією автоматів Мілі та Мура.

8.2.3.2. Мови опису функціонування автоматів

Для того, щоб задати абстрактний автомат, потрібно задати всі п'ять об'єктів $\{X, S, Y, \delta, \lambda\}$.

Множини X, S, Y задаються як звичайні множини в математиці, наприклад, простим перелічуванням всіх її елементів, тому їх задання на практиці не викликає ніяких труднощів.

Найбільш трудомістким є задання функцій δ, λ, які власне і визначають алгоритм функціонування автомату. Для опису алгоритму функціонування автомату, тобто для задання δ, λ, існують різні засоби, які часто називають мовами. Існують стандартні та початкові мови. Стандартні мови задають автомат одним із трьох способів: матрично (таблично), графічно, аналітично. До початкових мов відносять первісні таблиці включень, логічні схеми алгоритмів і граф-схеми алгоритмів. Стандартні мови частіше застосовуються для задання автоматів загального виду, в той же час початкові мови знайшли широке застосування для часткових автоматів.

Мова матриць (таблиць) передбачає наявність двох таблиць: таблиці переходів і таблиці виходів, або однієї таблиці з'єднань.

Таблиця переходів задає відображення $X \times S \to S$, тобто задає функцію переходів δ.

Приклад. Нехай на автомат поступають вхідні сигнали, які мають три букви, та нехай він має чотири стани: $X = \{x_1, x_2, x_3\}$, $S = \{s_1, s_2, s_3, s_4\}$. В табл. 8.1 описано повністю визначений автомат для даного прикладу.

Таблиця 8.1

S(t) \ X(t)	X1	X2	X3
S1	S1	S2	S1
S2	S3	S4	S2
S3	S3	S1	S4
S4	S3	S1	S2

З першого рядка табл. 8.1 видно, що перебуваючи в стані S1 при поступленні вхідного сигналу X1 автомат не змінює свого стану, так само як і при поступленні вхідного сигналу X3, а при поступленні вхідного сигналу X2 він перейде в стан S2. Подібним чином можна провести аналіз інших рядків таблиці.

Якщо автомат частковий, то для пар (x_i, s_j), для яких стан не визначений, в клітинці таблиці ставиться прочерк. Як видно, вигляд таблиці переходів не залежить від того, який тип автомату використовується: Мілі, Мура чи С-автомат.

Таблиці виходів цих автоматів відрізняються.

У клітинці таблиці виходів автомату Мілі ставиться вихідний сигнал y_k, який формує автомат Мілі, що знаходиться в стані S_j і на вході якого діє сигнал x_i. Приклад повністю визначеного автомату Мілі з вхідним алфавітом $X = \{x_1, x_2, x_3\}$, алфавітом станів; $S = \{s_1, s_2, s_3, s_4\}$ та вихідним алфавітом $Y = \{y_1, y_2, y_3\}$ наведено в табл. 8.2.

Таблиця 8.2

S(t) \ X(t)	X1	X2	X3
S1	Y1	Y1	Y2
S2	Y3	Y3	Y3
S3	Y2	Y1	Y3
S4	Y2	Y1	Y3

З першого рядка табл. 8.2 видно, що перебуваючи в стані S1 при поступленні вхідного сигналу X1 на виході автомату буде сформовано сигнал Y1, так само, як і при поступленні вхідного сигналу X2, а при поступленні вхідного сигналу X3 на виході автомату буде сформовано сигнал Y2. Подібним чином можна провести аналіз інших рядків таблиці.

В таблиці виходів повністю визначеного автомату Мура кожному стану автомату призначається відповідний вихідний сигнал y_k. Приклад повністю визначеного автомату Мура з алфавітом станів $S = \{s_1, s_2, s_3, s_4\}$ та вихідним алфавітом $Y = \{y_1, y_2, y_3\}$ наведено в табл. 8.3.

Таблиця 8.3

S(t)	Y(t)
S1	Y1
S2	Y1
S3	Y2
S4	Y3

З табл. 8.3 видно, що стану S1 та S2 автомату відповідає вихідний сигнал Y1, стану S3 відповідає вихідний сигнал Y2, а стану S4 відповідає вихідний сигнал Y3.

С-автомат буде задаватися двома таблицями виходів, перша з яких відповідає таблиці виходів автомату Мілі, а друга – таблиці автомату Мура.

На практиці таблиці переходів і таблиці виходів часто суміщаються в одну суміщену таблицю. Табл. 8.4 є суміщеною таблицею автомату Мілі для вищенаведеного прикладу.

Таблиця 8.4

S(t) \ X(t)	X1	X2	X3
S1	S1/Y1	S2/Y1	S1/Y2
S2	S3/Y3	S4/Y3	S2/Y3
S3	S3/Y2	S1/Y1	S4/Y3
S4	S3/Y2	S1/Y1	S2/Y3

З першого рядка табл. 8.4 видно, що перебуваючи в стані S1 при поступленні вхідного сигналу X1 автомат залишиться в тому ж стані, а на виході автомату буде сформовано сигнал Y1, при поступленні вхідного сигналу X2 автомат перейде в стан S2, а на виході автомату буде сформовано сигнал Y1, при поступленні вхідного сигналу X3 автомат залишиться в тому ж стані, а на виході автомату буде сформовано сигнал Y2. Подібним чином можна провести аналіз інших рядків таблиці.

Як вже зазначилось вище, можна задати керуючий автомат за допомогою єдиної таблиці з'єднань. Таблиця з'єднань абстрактного автомату є квадратною і містить стільки стовпців та рядків, скільки різних станів має даний автомат. В клітинці ставиться вхідний сигнал, під дією якого відбувається перехід автомату зі стану в стан. Якщо матрицею з'єднань задається автомат Мілі, то разом з вхідним сигналом вказується вихідний сигнал, який автомат Мілі видає, виконуючи перехід (табл. 8.5).

Таблиця 8.5

S(t) \ S(t)	S1	S2	S3	S4
S1	X1/Y1 X3/Y2	X2/Y1		
S2		X3/Y3	X1/Y3	X2/Y3
S3	X2/Y1		X1/Y2	X3/Y3
S4	X2/Y1	X3/Y2	X1/Y2	

З першого рядка табл. 8.5 видно, що автомат залишається в тому ж стані S1 при поступленні вхідних сигналів X1 та X3, і при цьому на його виході будуть відповідно сигнали Y1, та Y2, та переходить в стан S2 при поступленні вхідного сигналу X2, і при цьому на його виході буде сигнал Y1. Подібним чином можна провести аналіз інших рядків таблиці.

Для автомату Мура в матриці з'єднань вихідні сигнали ставляться біля станів автомату, які ідентифікують рядки матриці.

Мова графіки передбачає застосування для задання абстрактного автомату орієнтованого графа. Стан автомату зображається вершинами графа, а переходи між станами – дугами між відповідними вершинами. При цьому конкретній дузі графа приписується буква x_i вхідного алфавіту автомату, яка вказує на перехід при поступленні цього сигналу.

Якщо граф зображає автомат Мілі (рис. 8.7), то вихідні сигнали автомату ставляться на дугах графа (згідно з таблицею виходів) разом з буквою вхідного сигналу. Тут в якості прикладу взято автомат Мілі, описаний в табл. 8.4.

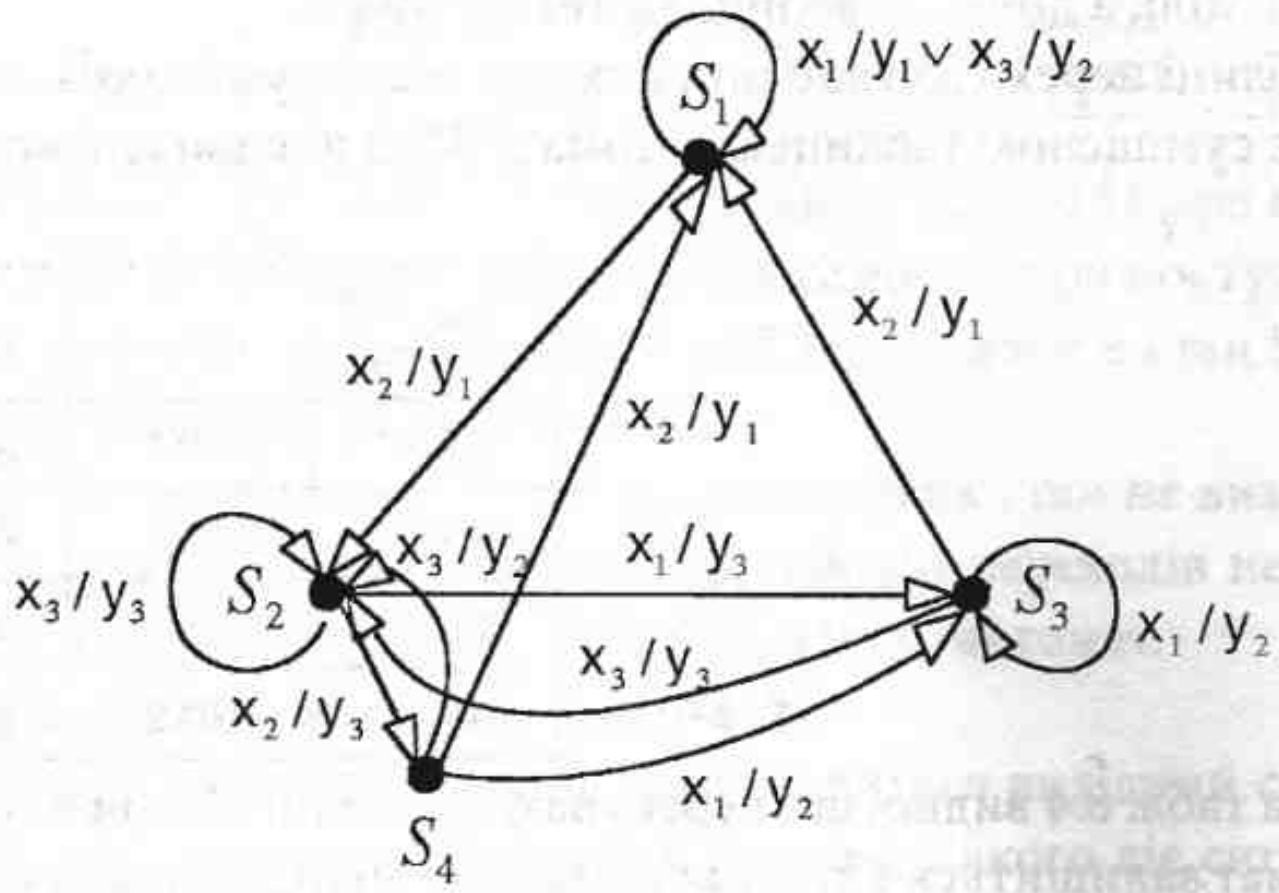

Рис .8.7. Граф автомату Мілі

Якщо графом зображається автомат Мура (рис. 8.8), то вихідні сигнали автомату ставляться біля вершини графа відповідно до таблиці виходів автомату (табл. 8.3).

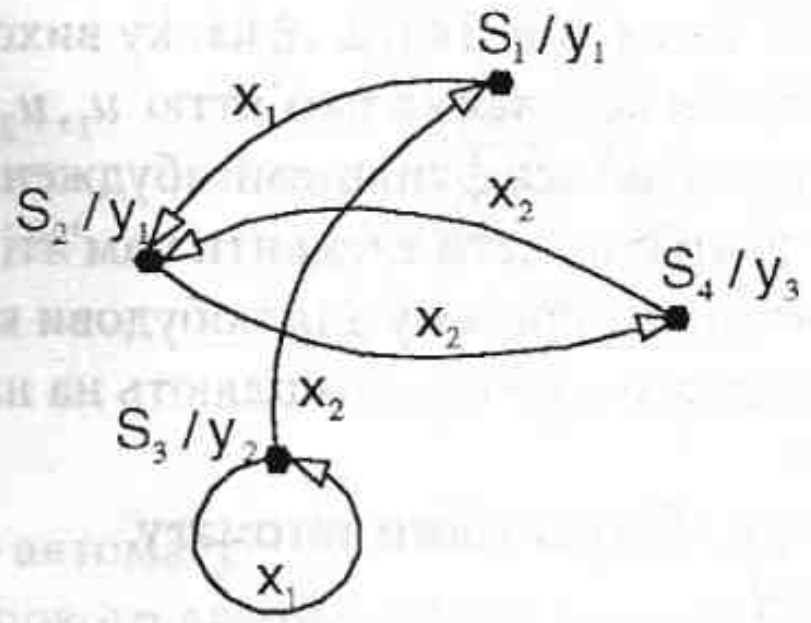

Рис. 8.8. Граф автомату Мура

Мова аналітичних виразів передбачає задання автомату шляхом запису для кожного стану автомату відображення F_{sj}, яке містить набори з трьох об'єктів s_n, x_i, y_k, причому тільки таких, які вказують на наявність переходу автомату зі стану s_j в стан s_n при дії вхідного сигналу x_i і видачі при цьому вихідного сигналу y_k.

Приклад:

$$F_{s1} = \{S_1(x_1 / y_1), S_2(x_2 / y_1), S_1(x_3 / y_2)\}$$

$$F_{s2} = \{S_3(x_1 / y_3), S_4(x_2 / y_3), S_2(x_3 / y_3)\}.$$

8.2.3.3. Структурний синтез цифрових автоматів

Процес одержання структурної схеми, яка відображає склад логічних елементів та їхні зв'язки, називають структурним синтезом. В загальному випадку задача структурного синтезу зводиться до композиції деяких простих автоматів, тобто до пошуку способу з'єднань цих автоматів між собою. Як правило, ефективно розв'язується задача структурного синтезу тільки для певного набору простих автоматів певного виду – елементарних автоматів, які складаються з елементів пам'яті, що мають більше одного стійкого стану (елементарних автоматів з пам'яттю) та комбінаційних схем (елементарних автоматів без пам'яті).

Метод синтезу, в основу якого покладені елементарні автомати, отримав назву канонічного методу структурного синтезу автоматів. Загальна структура елементарного автомату, що складається з пам'яті та комбінаційної схеми, представлена на рис. 8.9.

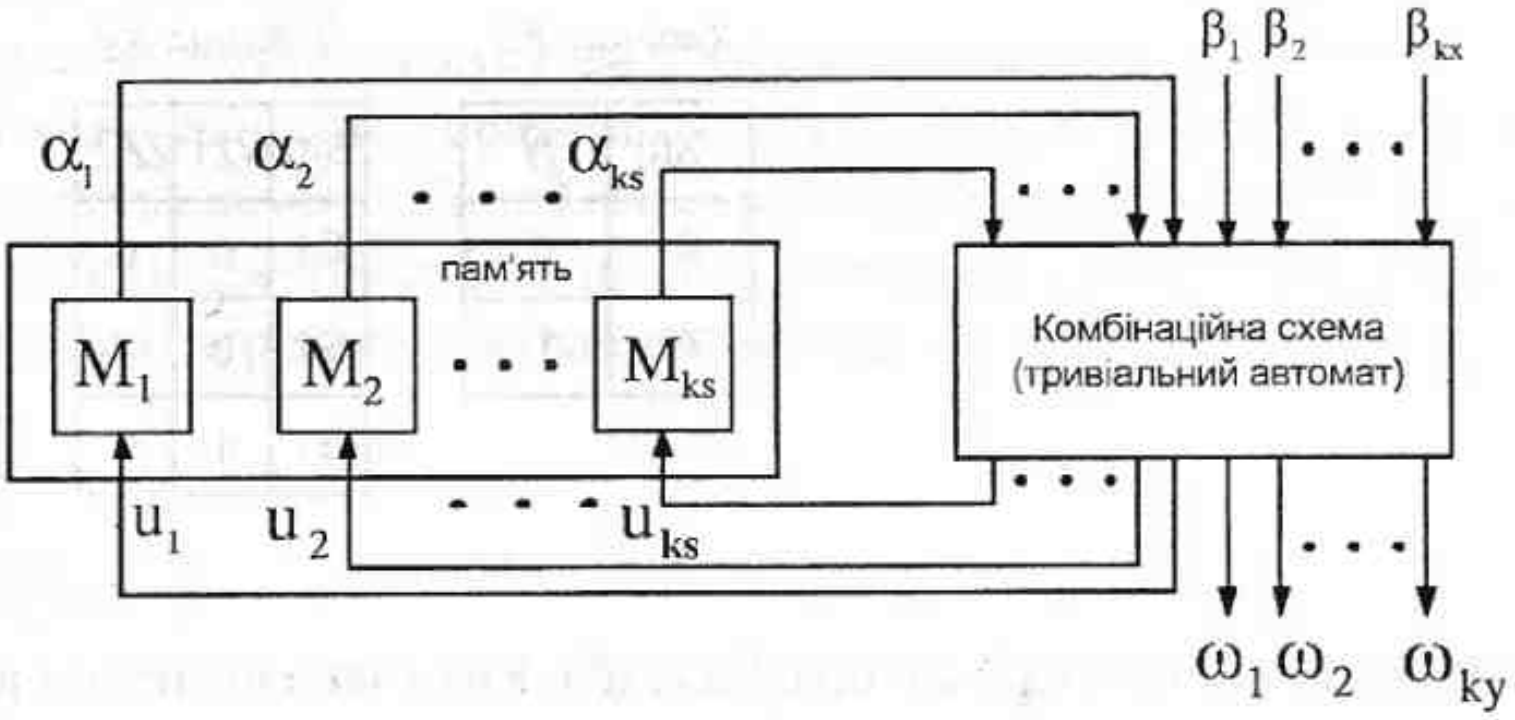

Рис. 8.9. Загальна структура елементарного автомату

Елементарний автомат має kx входів β 1 β 2... β kx, ky виходів $\omega_1\omega_2 ... \omega_{ky}$ та ks виходів пам'яті станів $\alpha_\alpha\alpha_2 ... \alpha_{ks}$. Сигнали керування пам'яттю $u_1, u_2, ... u_{ks}$ описуються за допомогою булевих функцій, які називаються функціями збудження. Таким чином, для побудови структурного автомату потрібно мати елементи пам'яті і набір логічних елементів, які утворюють функціонально повну систему для побудови комбінаційної схеми.

Канонічний метод структурного синтезу розділяють на наступні етапи:

- кодування,
- вибір типу та структури абстрактного автомату,
- вибір елементів пам'яті,
- побудова рівнянь булевих функцій збудження і виходів автомату,
- побудова структурної схеми автомату.

Розглянемо кожен з етапів.

Кодування. Нагадаємо, що абстрактний автомат задається в вигляді $A = \{X, S, Y, \delta, \lambda\}$. При переході на структурний рівень множини сигналів X та Y, а також сигнали S потрібно зобразити у вигляді двійкового вектору.

Нехай $S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$, тоді $k_s =] log_2 M_s [=] log_2 6 [= 3$. Тобто для нумерації кожного стану потрібно 3 розряди, тоді $S = \{000,001,010,011,100,101\}$. Фізично в структурному автоматі буде три стани, кожен з яких може прийняти тільки два значення 0 або 1. Сукупність значень цих трьох станів буде відповідати одному із станів абстрактного автомату.

Приклад: автомат описується суміщеною таблицею переходів та виходів (табл. 8.6).

Таблиця 8.6

S(t) \ X(t)	X1	X2
S1	S2/Y1	S1/Y3
S2	S3/Y2	S1/Y4
S3	S3/Y1	S2/Y2

Тобто він має три стани $M_s = n = 3$, два вхідних сигнали $M_x = m = 2$ та чотири вихідних сигнали $M_y = l = 4$. Вони відповідно можуть бути закодовані наступною кількістю розрядів: $K_s = 2$, $K_x = 1$ та $K_y = 2$.

Результати кодування вхідних сигналів наведено в табл. 8.7, станів – в табл. 8.8, та вихідних сигналів – в табл. 8.9.

Таблиця 8.7

X(t)	β
X1	0
X2	1

Таблиця 8.8

S(t)	$\alpha 1$	$\alpha 2$
S1	0	0
S1	0	1
S1	1	0

Таблиця 8.9

Y(t)	$\omega 1$	$\omega 2$
Y1	0	0
Y2	0	1
Y3	1	0
Y4	1	1

Тоді суміщена таблиця переходів та виходів (табл. 8.6) з закодованими входами, станами та виходами, буде мати вигляд табл. 8.10.

Таблиця 8.10

$\alpha_1\alpha_2$ \ β	0 $u_1u_2 / \omega_1\omega_2$	1 $u_1u_2 / \omega_1\omega_2$
00	01/00	00/10
01	10/01	00/11
10	10/00	01/01

Побудова абстрактного автомату.

Структурна схема цифрового автомату Мілі для розглядуваного прикладу має вигляд, показаний на рис. 8.10.

Рис. 8.10. Структурна схема цифрового автомату для розглядуваного прикладу

Вибір елементів пам'яті.

В якості елементів пам'яті структурного автомату можуть бути використані всі відомі типи тригерів, зокрема D-тригери, RS-тригери, T-тригери, JK-тригери.

Якщо в якості елементів пам'яті вибираються тригери, які мають вхід синхронізації, то структурний автомат буде синхронним, а якщо вибираються асинхронні тригери, то автомат буде асинхронним.

Побудова рівнянь булевих функцій збудження і виходів автомату.

Провівши кодування та вибравши систему логічних елементів, можна однозначно визначити структуру комбінаційних схем автомату. Рівняння булевих функцій будуються на основі таблиці істинності функції збудження, яка в свою чергу будується на основі структурної таблиці переходів і таблиці переходів елемента пам'яті. Для наведеної вище таблиці маємо:

$$u_1 = \alpha_1\alpha_2\overline{\beta}$$

$$u_2 = \overline{\alpha_1\alpha_2}\,\overline{\beta} \vee \alpha_1\overline{\alpha}_2\beta$$

$$w_1 = \beta\overline{\alpha_1}$$

$$w_2 = \overline{\alpha_1}\alpha_2 \vee \alpha_1\overline{\alpha}_2\beta \quad .$$

Побудова структурної схеми автомату.

Маючи наведені логічні рівняння, синтезується схема автомату, наведена на рис. 8.11.

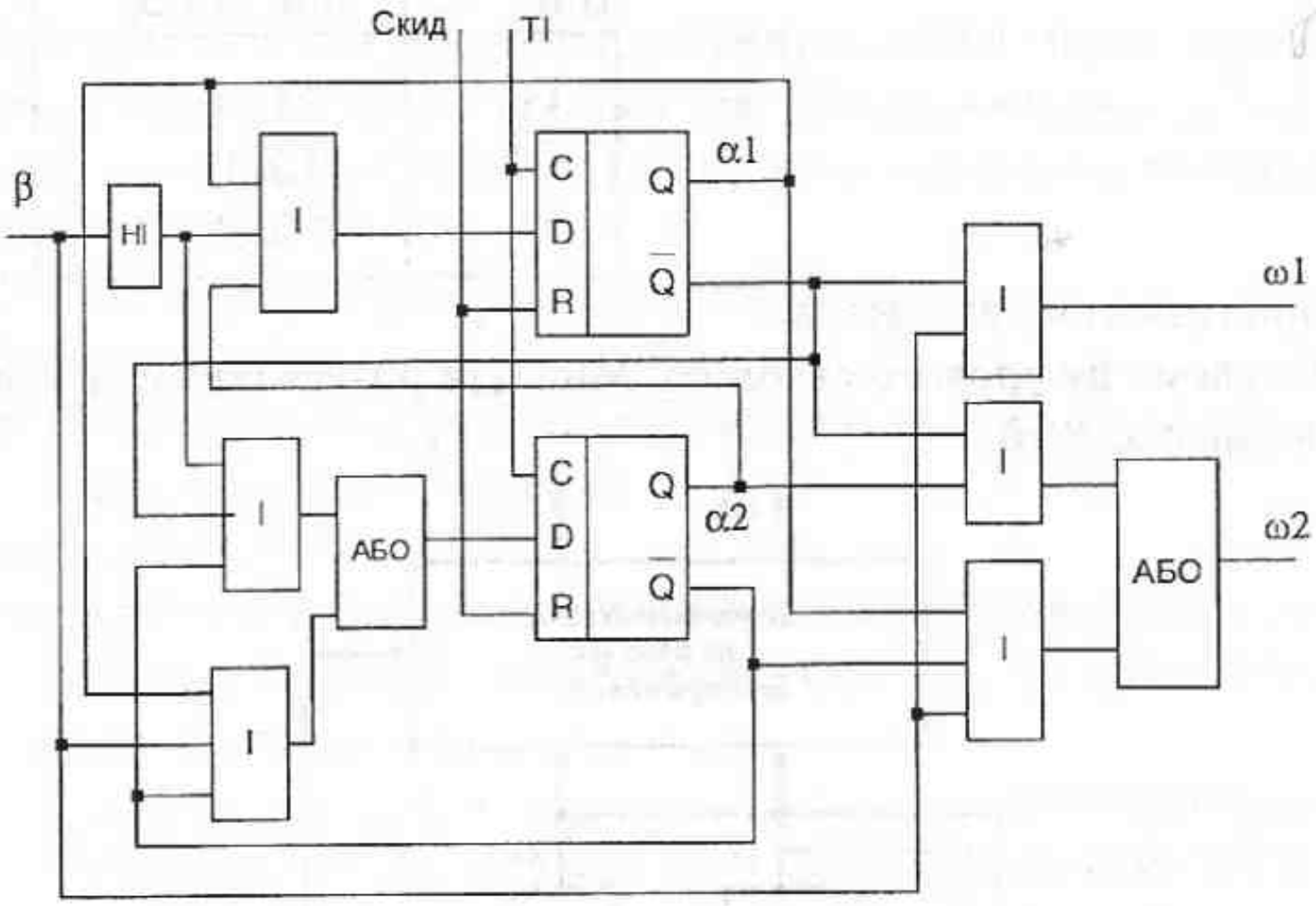

Рис. 8.11. Схема цифрового автомату для розглянутого прикладу

8.2.4. Пристрій керування на основі синхронних елементів часової затримки

Пристрій керування на основі синхронних елементів часової затримки будується з блок-схеми, яка вказує порядок формування послідовності сигналів керування. Структура таким чином побудованого пристрою керування по суті повторює структуру блок-схеми, тобто схема пристрою керування відображає послідовність формування сигналів керування. В основу методу побудови пристрою керування вказаного типу покладено те, що формування набору керуючих сигналів в послідовні моменти часу можна здійснити шляхом здійснення їх часової затримки. Тобто, маючи сигнал C1 в момент часу t1, шляхом його затримки на один такт, можна сформувати сигнал C2 в момент часу t2 і т. д. В доволі спрощеному вигляді правила побудови пристрою керування на основі блок-схеми є наступними:

1. Кожні дві послідовно з'єднані мікрооперації замінюються одним елементом затримки, як це показано на рис. 8.12a. В якості елемента затримки може бути використаний тригер типу D, який керується спільним тактовим сигналом. Мікронакази подаються на вхід керування відповідного вузла комп'ютера прямо з входу та виходу елемента затримки.

2. Лінії зв'язків блок-схеми, які об'єднуються в одну лінію, перетворюються в k-входову логічну схему АБО, де k – кількість ліній, як це показано на рис. 8.12b.

3. Умовна вершина, яка відображає умовний перехід в потоці керуючих сигналів, реалізується двома логічними елементами І, як це показано на рис. 8.12c. Фактично ця вершина є однорозрядним демультиплексором, який керується сигналом x.

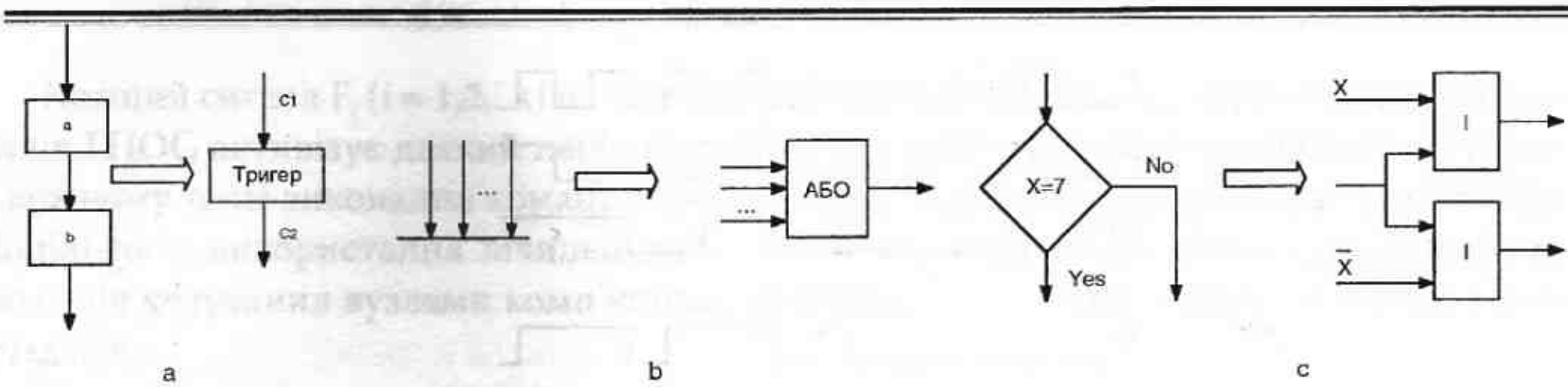

Рис. 8.12. Заміна двох послідовно з'єднаних мікрооперацій на один елемент затримки a), к ліній на k-входовий елемент АБО b), та умовну вершину на два елементи І c)

На рис. 8.13а показано фрагмент типової блок-схеми, що задає сигнали керування, які потрібно сформувати в послідовних тактах, а на рис .8.13b показано відповідний їй фрагмент схеми пристрою керування, отриманий за вище описаними правилами.

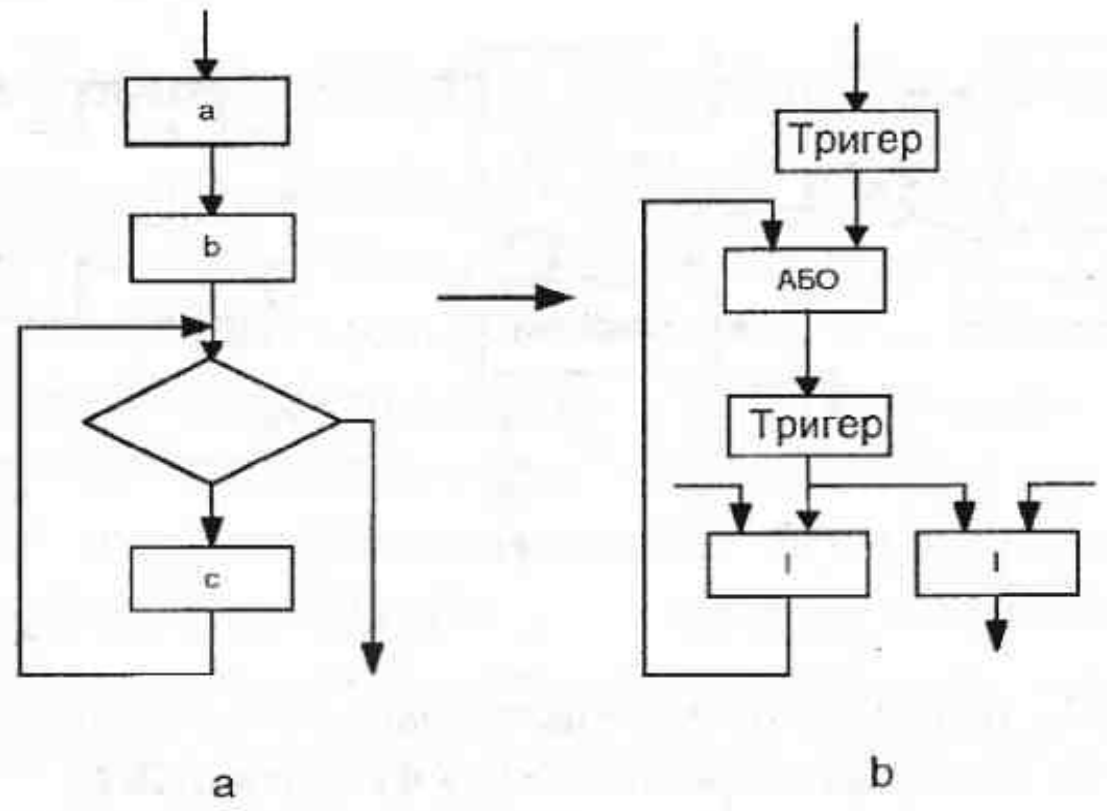

Рис. 8.13. Фрагмент блок-схеми, яка задає сигнали керування a), та відповідний їй фрагмент схеми пристрою керування b)

Не дивлячись на простоту описаного методу проектування пристрою керування на основі синхронних елементів затримки, цей метод має той недолік, що число потрібних схем затримки приблизно рівне числу станів n, тоді як в раніше розглянутому методі таблиць станів кількість елементів пам'яті, які виступають в даному випадку в ролі елементів затримки, рівна $\log_2 n$. Крім того, тут існує проблема синхронізації багатьох розподілених елементів затримки.

8.2.5. Пристрій керування на основі лічильників

В основу методу побудови пристрою керування на основі лічильників покладено часову діаграму роботи комп'ютера, яка відображає зміну в часі кожного сигналу керування. В якості прикладу на рис. 8.14 наведено фрагмент часової діаграми роботи комп'ютера, де ТІ – тактові імпульси, які поступають з блоку синхроімпульсів (рис. 8.2), С1-С5 – частина сигналів керування, які мають бути вироблені пристроєм керування.

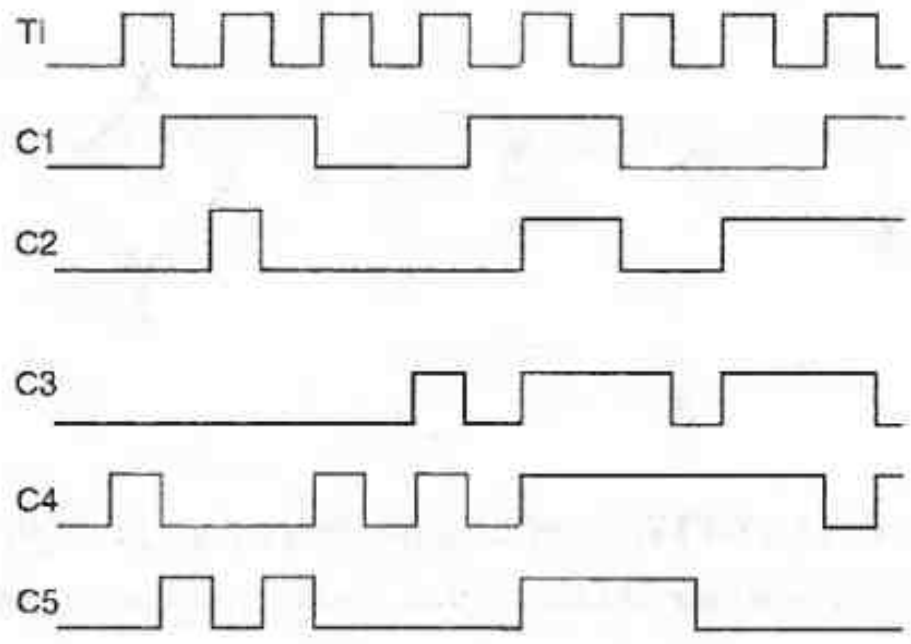

Рис. 8.14. Фрагмент часової діаграми роботи комп'ютера

Основним елементом пристрою керування на основі лічильників є лічильник за модулем k, виходи якого з'єднані з дешифратором (рис. 8.15a).

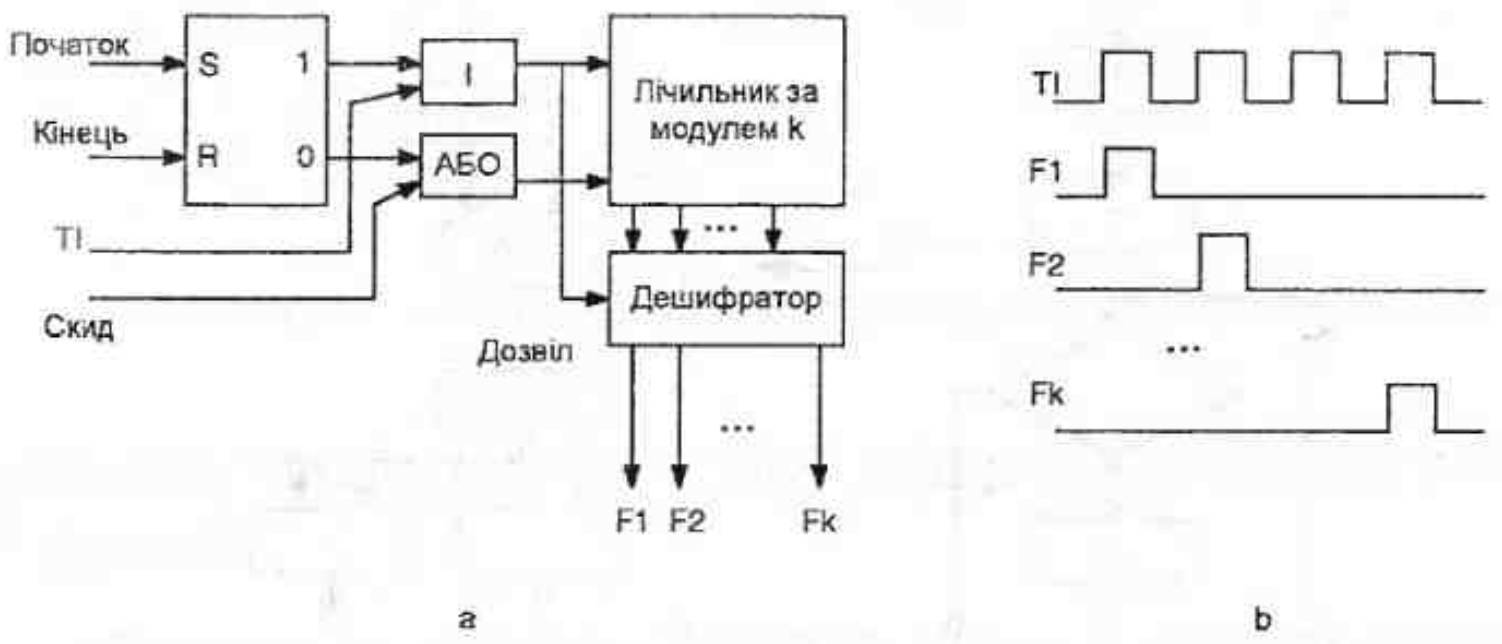

Рис. 8.15. Генератор послідовності одноімпульсних сигналів a) та часова діаграма сигналів на його виході b)

Коли на вході лічильника за модулем k є тактові імпульси, він проводить їх підрахунок від нульового до k-го імпульсу, після чого цикл повторюється. В результаті на виході дешифратора буде формуватися послідовність одноімпульсних сигналів F1, F2,..., Fk, часова діаграма яких наведена на рис. 8.15b. Кожний з цих сигналів має одиничне значення лише протягом одного тактового періоду. Тим самим, час одного циклу роботи лічильника поділено на k рівних частин. Два додаткових вхідних сигнали початку та кінця роботи та тригер RS типу забезпечують формування сигналів дозволу роботи лічильника та його скиду. Назвемо схему, представлену на рис. 8.15a, генератором послідовності одноімпульсних сигналів (ГПОС). Тоді базова частина схеми пристрою керування на основі лічильника буде мати вигляд, показаний на рис. 8.16.

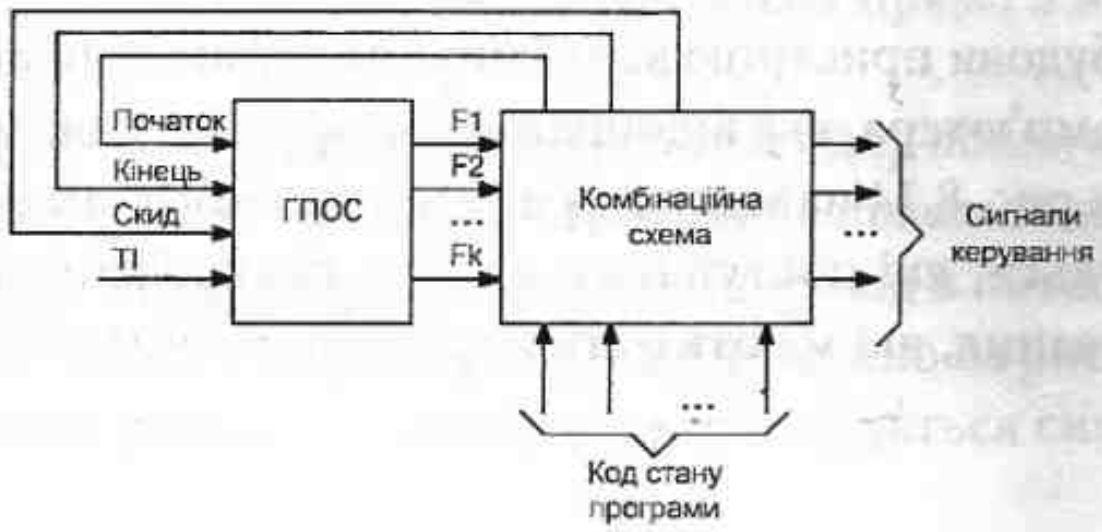

Рис. 8.16. Схема пристрою керування на основі лічильника

Кожний сигнал F_i (i = 1,2,..k) на виході генератора послідовності одноімпульсних сигналів ГПОС активізує деякий набір сигналів керування на виході комбінаційної схеми в кожному такті виконання команди комп'ютером, з врахуванням коду стану програми. Доцільність використання лічильника за модулем пояснюється циклічною природою сигналів керування вузлами комп'ютера, що неодноразово було показано в попередніх розділах.

Потрібно відзначити, що лічильник за модулем k може бути використаний і в схемі пристрою керування на основі синхронних елементів часової затримки взамін k послідовно з'єднаних тригерів, так само як k послідовно з'єднаних тригерів можуть замінити лічильник за модулем k та дешифратор у вище наведеній схемі (рис. 8.15).

8.3. Пристрій мікропрограмного керування

8.3.1. Організація роботи пристрою мікропрограмного керування

Пристрій мікропрограмного керування виробляє послідовність сигналів, необхідних для виконання програми в комп'ютері. Програма складається з деякої послідовності команд. Команда в комп'ютері виконується за один або за декілька тактів, в кожному із яких виконується одна або декілька мікрооперацій. Кожна мікрооперація представляє собою деяку елементарну дію передачі або перетворення інформації, яка ініціюється поступленням керуючого сигналу (мікронаказу) на вхід керування відповідного пристрою. Послідовність елементарних мікронаказів, які пристрій керування формує в одному такті, називають мікрокомандою. Послідовність мікрокоманд, що необхідно виконати для виконання однієї команди, називаються мікропрограмою. Звичайно, мікропрограма може складатися і лише з однієї мікрокоманди.

Основними принципами, які покладені в основу побудови пристрою мікропрограмного керування, є наступні:

1. Всі мікронакази, які повинні бути виконані в одному такті роботи комп'ютера, збираються в одне керуюче слово, яке називають мікрокомандою.

2. Кожній команді з системи команд комп'ютера ставиться у відповідність послідовність мікрокоманд, необхідних для її виконання, тобто мікропрограма виконання команди в комп'ютері.

3. Всі мікрокоманди зберігаються в пам'яті. Це може бути основна пам'ять комп'ютера, але в більшості комп'ютерів для зберігання мікрокоманд використовується окрема пам'ять, яку називають пам'яттю мікрокоманд.

4. Для реалізації деякої команди необхідно зчитати з пам'яті мікрокоманд відповідну послідовність мікрокоманд (мікропрограму) та подати розподілену в часі послідовність керуючих сигналів на відповідні керуючі входи вузлів комп'ютера.

На рис. 8.17 показано основний елемент пристрою мікропрограмного керування – пам'ять мікрокоманд, та вузли на її входах і виходах, а саме мікропрограмний лічильник (МКПЛ) для зберігання адреси мікрокоманди та регістр мікрокоманди (РгМК).

Рис. 8.17. Пам'ять мікрокоманд з регістром адреси мікрокоманди на вході та регістром мікрокоманди на виході

На рис. 8.17 показано і формат самої мікрокоманди, до складу якої входять наступні поля: код мікрооперації, за яким формуються мікронакази, що виконуються в одному такті роботи комп'ютера, код умов, котрий вказує, при яких умовах може бути змінено послідовність читання мікрокоманд з пам'яті, а також адреса наступної мікрокоманди.

Структура пристрою мікропрограмного керування представлена на рис. 8.18.

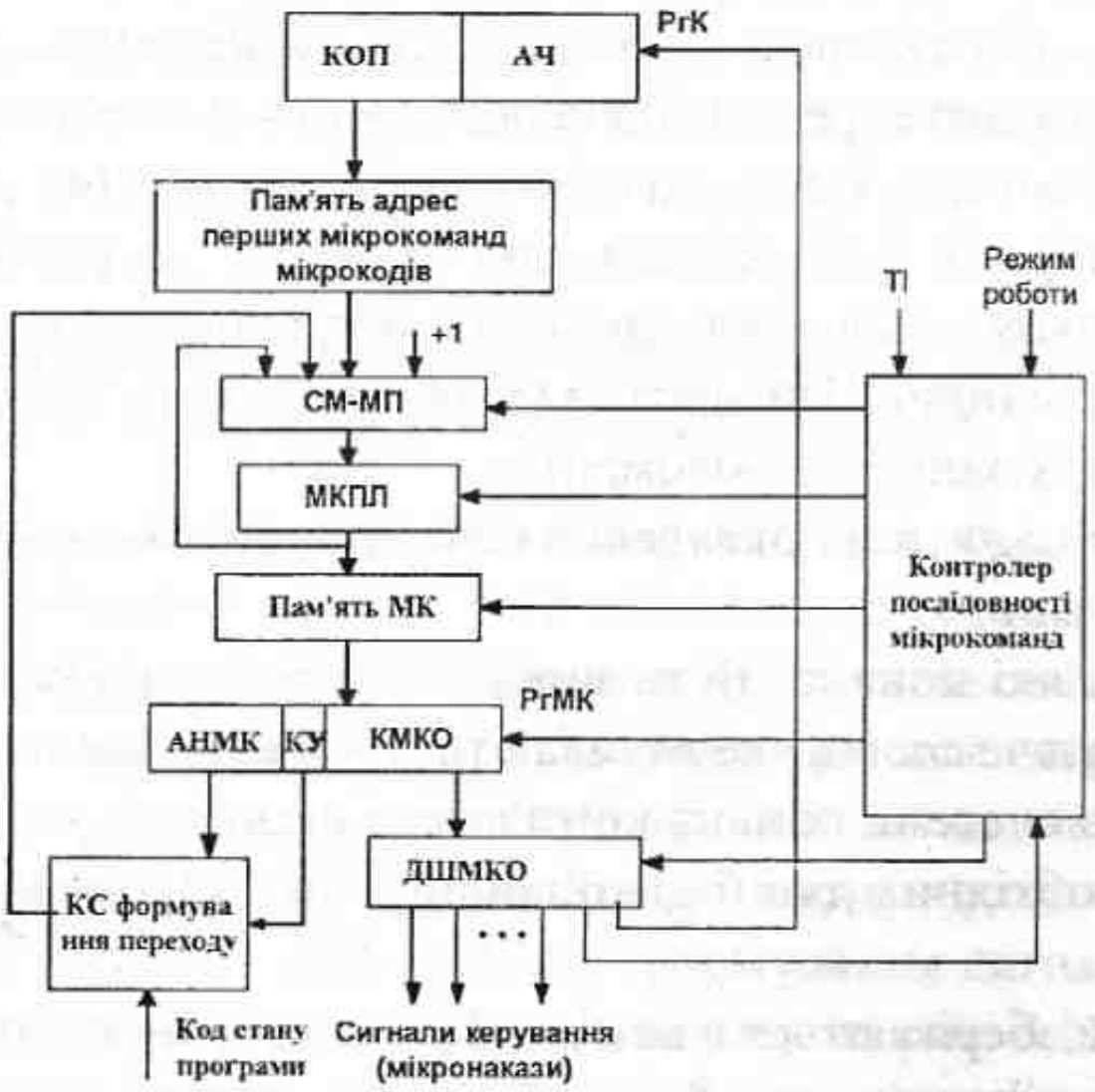

Рис. 8.18. Структура пристрою мікропрограмного керування

В регістрі команди РгК зберігається команда, яка підлягає виконанню в комп'ютері. За її кодом операції КОП з пам'яті адрес перших мікрокоманд мікрокодів зчитується адреса пам'яті мікрокоманд, в якій знаходиться перша мікрокоманда із послідовності мікрокоманд (мікропрограми, або як її іще називають, мікрокоду) її виконання. Пам'ять мікрокоманд в більшості випадків реалізується на основі постійного запам'ятовуючого пристрою ПЗП, хоча може бути реалізована і на основі оперативного запам'ятовуючого пристрою ОЗП, особливо на етапах відлагодження комп'ютера. В цій пам'яті зберігаються мікрокоманди для всіх команд комп'ютера, а також для початку роботи комп'ютера

та для обробки переривань. В схемі формування адреси СФА, до складу якої входять пам'ять адрес перших мікрокоманд мікрокодів, суматор-мультиплексор СМ-МП, мікропрограмний лічильник МКПЛ та комбінаційна схема КС формування переходу, визначається адреса комірки пам'яті мікрокоманд, в якій знаходиться наступна мікрокоманда. Ця адреса формується з врахуванням полів адреси наступної мікрокоманди АНМК та коду умови з регістра мікрокоманди, а також сигналів стану, які поступають з регістра слова стану програми. Мікрокоманда зчитується із пам'яті в регістр мікрокоманди РгМК, в якому вона зберігається протягом одного такту роботи комп'ютера. На основі коду мікрооперації КМКО з регістра мікрооперації на виході дешифратора ДШМКО формуються сигнали керування.

Робота компонента пристрою керування синхронізується з контролера послідовності мікрокоманд, який отримує ззовні тактові імпульси з генератора тактових імпульсів, а також код режиму роботи комп'ютера. Контролер послідовності мікрокоманд є ядром пристрою керування. Він має два окремих режими роботи: звичайний режим та режим запуску комп'ютера.

В звичайному режимі контролер послідовності мікрокоманд генерує сигнали керування роботою пристрою керування. Тактові імпульси на його вході забезпечують його часову синхронізацію, що дозволяє генерувати наступні послідовності сигналів:

1. Керування процесом формування адреси мікрокоманди. Ця адреса одержується шляхом:

- запису в мікропрограмний лічильник МКПЛ адреси першої мікрокоманди мікропрограми виконання відповідної команди з пам'яті адрес перших мікрокоманд мікрокодів;
- запису в мікропрограмний лічильник МКПЛ адреси наступної мікрокоманди мікропрограми виконання відповідної команди після виконання в суматорі-мультиплексорі СМ-МП операції приросту вмісту МКПЛ на 1;
- запису в мікропрограмний лічильник МКПЛ адреси наступної мікрокоманди АНМК з адресного поля РгМК, з врахуванням коду умови КУ з поля умовного переходу РгМК, та коду стану програми з регістра слова стану програми, що здійснюється в комбінаційній схемі КС формування переходу.

Керування зчитуванням мікрокоманд з пам'яті мікрокоманд за адресами з мікропрограмного лічильника МКПЛ та їх записом до регістра мікрокоманд РгК.

Стимулювання дешифратора мікрооперацій ДШМКО, який здійснює дешифрування коду мікрооперації КМКО з відповідного поля РгМК, та видає сигнали керування (мікронакази).

Кожна команда процесора ініціює виконання відповідної мікропрограми. Коли пристрій керування закінчує виконання мікропрограми однієї команди комп'ютера, про що його контролер послідовності мікрокоманд інформується сигналом з виходу дешифратора мікрооперацій ДШМКО, він вибирає з основної пам'яті наступну команду, та записує її в регістр команди РгК сигналом з виходу дешифратора мікрооперацій ДШМКО, після чого приступає до її виконання шляхом зчитування з пам'яті мікрокоманд наступної мікропрограми.

В ході виконання різних команд можуть використовуватися загальні ділянки мікропрограм, які називаються мікропідпрограмами.

В режимі запуску комп'ютера пристрій керування встановлює вміст різних регістрів комп'ютера в початковий стан шляхом їх скиду, або запису до них деяких конкретних значень. Після цього він записує апаратно генеровану адресу до програмного лічильника ПЛ (не мікропрограмного лічильника МКПЛ), та починає виконання програми. Для деяких комп'ютерів апаратно генерована адреса – це вектор скиду, який є адресою першої команди комп'ютера, яка виконується після старту. Для інших комп'ютерів апаратно генерована адреса – це адреса вектора скиду, який є адресою першої команди комп'ютера, яка виконується після старту. В цьому випадку пристрій керування спочатку повинен вибрати з основної пам'яті вектор скиду, та записати його до програмного лічильника ПЛ.

8.3.2. Організація мікропрограм в пам'яті мікрокоманд

Є багато шляхів можливої організації мікропрограм в пам'яті мікрокоманд. Один з них, ілюстрований на рис. 8.23, передбачає розміщення мікрокоманд в пам'яті послідовно. Кожна машинна команда має свою власну послідовність мікрокоманд в пам'яті мікрокоманд. Додатково пам'ять мікрокоманд містить мікрокоманди для проведення вибірки команд з основної пам'яті, запуску переривання, та деяких інших дій керування.

Розглянемо цей тип організації мікропрограм в пам'яті мікрокоманд. Після того, як пристрій керування вибирає машинну команду з основної пам'яті і розміщує її в регістрі команди РгК, він повинен генерувати так звану адресу точки входу для коду операції команди КОП, яка є адресою першої мікрокоманди мікропрограми. Наприклад, якщо пристрій керування вибирає мікрокоманду для коду операції КОП1, він повинен генерувати адресу Al, як показано на рис. 8.19. Формування адреси точки входу є задачею пам'яті адрес перших мікрокоманд мікрокодів блоку обчислення адреси (рис. 8.18). Після того, як пристрій керування формує адресу точки входу, контролер послідовності мікрокоманд збільшує вміст мікропрограмного лічильника МКПЛ, щоб одержати адресу кожної наступної мікрокоманди.

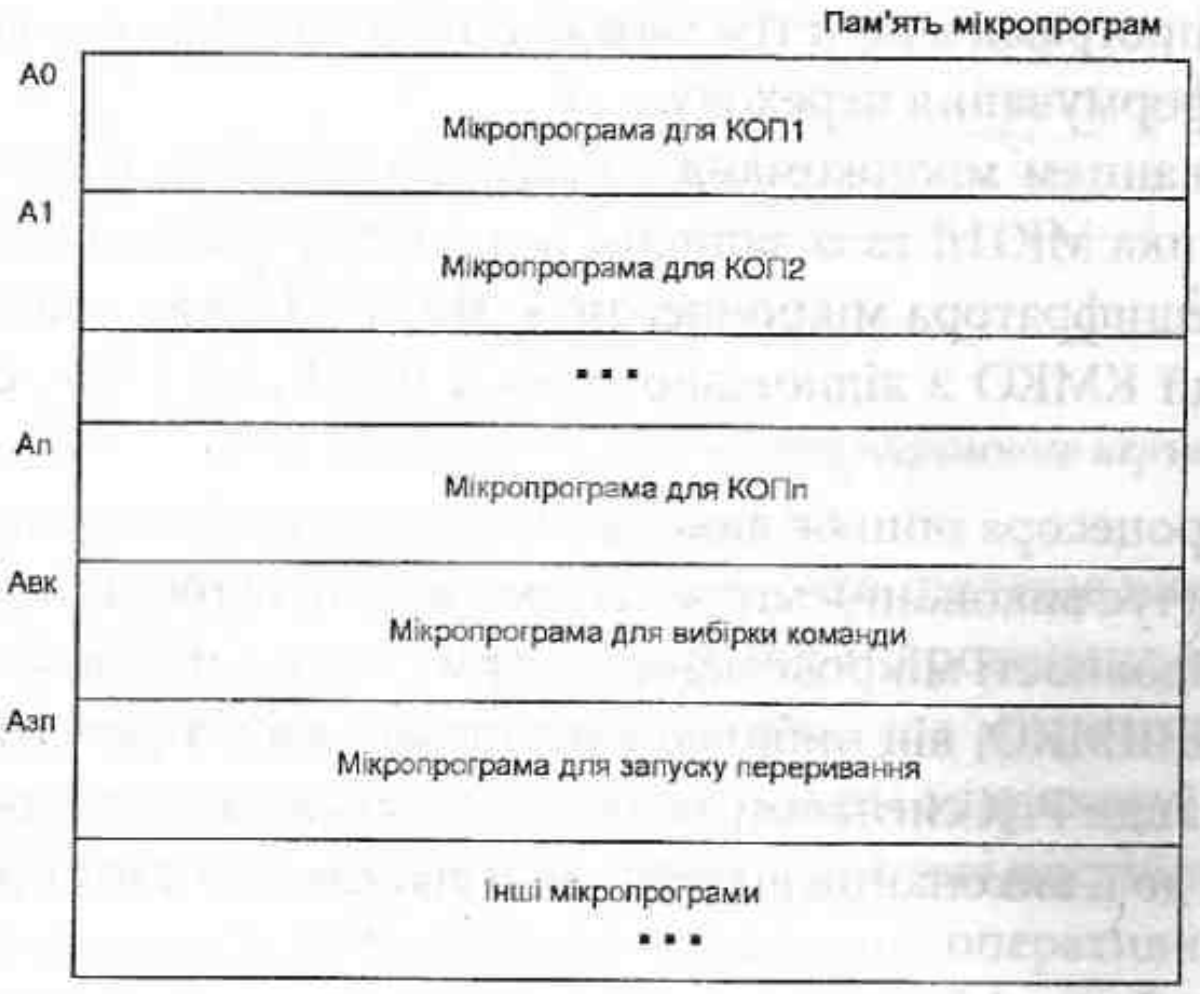

Рис. 8.19. Розміщення мікрокоманд послідовно в пам'яті

Після виконання останньої мікрокоманди в мікропрограмі (тобто, після завершення виконання однієї машинної команди), контролер послідовності мікрокоманд повинен ще раз виконувати мікропрограму вибірки наступної команди з основної пам'яті. На рис. 8.19 це є перехід до комірки $A_{ВК}$ пам'яті мікрокоманд. Постає питання – як може контролер послідовності мікрокоманд керувати розгалуженнями в межах мікропрограм пам'яті мікрокоманд? На рис. 8.18 показано один з варіантів вирішення цього завдання. До цих пір ми розглядали мікропрограми без розгалужень. Однак існують мікрокоманди переходу, які містять адресу переходу та мікронакази для схеми формування адреси СФА додатково до мікронаказів для інших вузлів комп'ютера. Контролер послідовності мікрокоманд разом з схемою формування адреси СФА, використовує адресу наступної мікрокоманди (АНМК) та код умови переходу (КУ) для визначення адреси наступної мікрокоманди.

Для керування переходом в коді мікрооперації КМКО кожної мікрокоманди наявна інформація про належність цієї мікрокоманди до мікрокоманд переходу. Тому в кожному такті на виході дешифратора мікрокоманд ДШМКО генерується мікронаказ, який інформує контролер послідовності мікрокоманд про належність, або неналежність, даної мікрокоманди до мікрокоманд переходу. Якщо це мікрокоманда переходу, то контролер послідовності мікрокоманд повідомляє схему формування адрес про необхідність формування адреси переходу, а не приріст на одиницю вмісту мікропрограмного лічильника МКПЛ. При цьому, якщо це безумовний перехід, то перехід відбувається за адресою з адресного поля мікрокоманди. Якщо ж це умовний перехід, то адреса наступної мікрокоманди формується з врахуванням коду умов переходу та коду стану програми, який поступає з регістра стану програми регістрової пам'яті процесора.

8.3.3. Горизонтальне та вертикальне мікропрограмування

Завдання вибору формату мікрокоманди є досить складним. З одного боку, мікрокоманда повинна вміщувати коди керування вузлами комп'ютера, забезпечувати організацію послідовності мікрокоманд в мікропрограму та можливість зміни порядку мікрокоманд в мікропрограмі. З іншого боку, ці функції повинні виконуватись з мінімальною кількістю бітів в мікрокоманді, з мінімальною кількістю слів в пам'яті мікрокоманд, та з мінімальним часом на виконання мікропрограми. Розглянемо деякі питання вибору ефективного формату мікрокоманди.

За способом формування керуючих сигналів розрізняють горизонтальне і вертикальне мікропрограмування.

При використанні горизонтального мікропрограмування кожний розряд поля коду мікрооперації мікрокоманди формує один керуючий сигнал (мікронаказ) для відповідного входу керування функціонального вузла комп'ютера (рис. 8.20). Кількість розрядів мікрокоманди визначається з виразу $M = n+k+m$, де n – розрядність адреси мікрокоманди, яка рівна $n = \log_2 N$, де N – кількість мікрокоманд в пам'яті, k – кількість мікронаказів, необхідних для керування вузлами пристрою керування, m – кількість мікронаказів, необхідних для керування вузлами комп'ютера. В цьому випадку відпадає потреба в дешифраторі мікрокоманд.

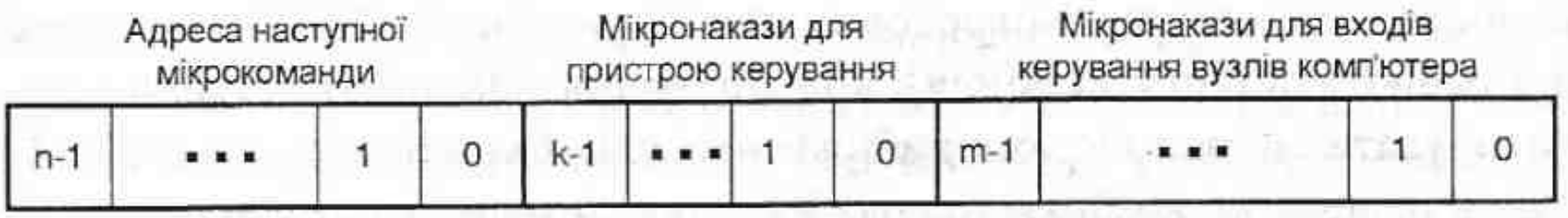

Рис. 8.20. Формат мікрокоманди при використанні горизонтального мікропрограмування

Оскільки у комп'ютерах кількість мікронаказів може досягати декількох сотень, мікрокоманда в цьому випадку стає дуже широкою.

Кількість керуючих бітів мікрокоманди зменшують використовуючи наступні способи:

- Групуванням бітів. В групи об'єднуються такі мікронакази, які завжди виконуються одночасно. При цьому для групи виділяється лише один біт.
- Групування форматів. В групи об'єднуються такі мікронакази, з яких в даному такті виконується тільки один. Ці мікронакази кодуються в полі, де кількість бітів $k = \log_2 L$, де L – кількість мікронаказів.
- Групування мікронаказів. Групі мікронаказів виділяється один біт мікрокоманди та використовується багататактова синхронізація.

При використанні вертикального мікропрограмування мікрокоманда складається з полів коду мікрооперації, коду умов переходу і адреси наступної мікрокоманди, як це було розглянуто раніше (рис. 8.17). Тобто формат мікрокоманди подібний до формату команди комп'ютера. Цей метод дозволяє більш ефективно використовувати поля мікрокоманди, тобто команда є коротшою, а об'єм пам'яті меншим, порівняно з горизонтальним мікропрограмуванням. Разом з тим, горизонтальне мікропрограмування є швидшим, оскільки не вимагає використання дешифраторів.

8.4. Порівняння пристроїв керування з жорсткою логікою та пристроїв мікропрограмного керування

Вище були розглянуті два основних методи побудови логіки формування сигналів керування. Перший з них, який одержав назву "жорсткої" або "запаяної" логіки, виражається в тому, що для кожної команди процесора існує набір логічних схем, які в потрібних тактах збуджують відповідні сигнали керування. Другий метод, який називають принципом мікропрограмного керування, передбачає формування сигналів керування за вмістом регістра мікрокоманд, в який мікрокоманди записуються із пам'яті мікрокоманд. Шляхом послідовного зчитування мікрокоманд із пам'яті в цей регістр організується потрібна послідовність сигналів керування. Завдяки тому, що мікрокоманди записуються до пам'яті, вміст якої при потребі можна частково, або повністю замінити, пристрої мікропрограмного керування мають наступні основні переваги в порівнянні з пристроями керування з жорсткою логікою:

- В них можна використовувати мікропрограми, які вже були відлагоджені та апробовані на інших комп'ютерах.
- Шляхом заміни мікропрограми в пам'яті мікрокоманд комп'ютер можна модифікувати з метою покращання технічних характеристик чи розширення функцій, і, тим самим, продовжити термін його використання.

- Можуть бути використані наробки мікропрограм в наступних поколіннях комп'ютерів однієї сім'ї.
- Мікропрограмування є простішим, ніж керування з жорсткою логікою, що спрощує розробку пристрою керування.
- Простішим є обслуговування мікропрограмованих комп'ютерів та їх відлагодження завдяки простішій заміні мікрокоманд та мікропрограм.

В швидкодії мікропрограмне керування програє керуванню з "жорсткою" логікою. Тому, завдяки створенню мов опису апаратних засобів комп'ютера та потужних програмних засобів високорівневого проектування, пристрої керування з жорсткою логікою знайшли ширше застосування в сучасних комп'ютерах.

8.5. Короткий зміст розділу

Пристрій керування є одним з вузлів процесора. Відомі два основних методи побудови пристроїв керування: пристрої керування з жорсткою логікою та пристрої мікропрограмного керування. В комп'ютері, крім пристрою керування центрального процесора, можуть використовуватись пристрої керування вузлами комп'ютера, наприклад, пристрої керування операційними пристроями АЛП, пристрій керування процесора введення-виведення і т. д. Принципи побудови вказаних пристроїв є ідентичними.

Розглянуто структуру та організацію роботи пристрою керування з жорсткою логікою, а також методи проектування пристроїв керування з жорсткою логікою: на основі таблиць станів, на основі тактованих елементів часової затримки, та на основі лічильників.

Метод таблиць станів передбачає розгляд пристрою керування як цифрового автомату. Цифровий автомат подано у вигляді його математичної (абстрактної) і структурної моделей, які відповідно називаються абстрактним та структурним автоматами. Абстрактну модель використано на першому етапі проектування, коли описується функціонування автомату, тобто правила переробки вхідної інформації у вихідну. На цьому етапі автомат подано у вигляді автоматів Мілі, Мура та С-автомату. Слід зауважити, що розгляд абстрактної моделі цифрового автомату дозволяє проводити його попередню оптимізацію ще до етапу структурного синтезу. Показано приклад застосування структурної моделі для побудови схеми цифрового автомату.

Описано основні засади проектування пристроїв керування з жорсткою логікою на основі синхронних елементів часової затримки та на основі лічильників. Відзначено, що вихідними даними для такого проектування є часові зміни сигналів керування, отримані з часової діаграми роботи комп'ютера.

Описана робота пристрою мікропрограмного керування. Наведені основні поняття мікропрограмування: мікронаказ, мікрокоманда, мікропрограма, горизонтальне та вертикальне мікропрограмування. Сформовані принципи, покладені в основу побудови пристрою мікропрограмного керування, серед яких в першу чергу необхідно відзначити те, що кожній команді з системи команд комп'ютера ставиться у відповідність мікропрограма її виконання в комп'ютері, всі мікрокоманди зберігаються в пам'яті мікрокоманд,

а для реалізації деякої команди необхідно зчитати з пам'яті мікрокоманд відповідну мікропрограму та подати розподілену в часі послідовність керуючих сигналів на відповідні керуючі входи вузлів комп'ютера. Розглянуті питання розміщення мікрокоманд в пам'яті, формат мікрокоманди та способи його оптимізації. Порівняння пристроїв керування з жорсткою логікою та пристроїв мікропрограмного керування показало, що перші є швидшими, а другі, завдяки тому, що мікрокоманди записуються до пам'яті, вміст якої при потребі можна частково, або повністю, замінити, є простішими при проектуванні та обслуговуванні.

8.6. Література для подальшого читання

До перших публікацій з питань проектування пристрою керування комп'ютера на основі цифрових автоматів Мілі, Мура, С-автомату, а також принципи побудови пристроїв мікропрограмного керування належать праці [5–9]. Серед перших книг з питань синтезу цифрових автоматів необхідно виділити книгу [3]. Цьому ж питанню присвячена і книга [17]. Побудова пристроїв керування на основі синхронних елементів часової затримки та лічильників описана в роботі [10]. З питань оптимізації мікропрограм доцільно почитати роботи [11–13], а в роботах [14–16] описані основні принципи мікропрограмування.

8.7. Література до розділу 8

1. Лазарев В. Г., Пийль Е. И. Синтез управляющих автоматов. – М.: Энергия, 1978. – 408 с.

2. Баранов С. И. Синтез микропрограммных автоматов. – Л.: Энергия, 1974. – 215 с.

3. Глушков В. М. Синтез цифровых автоматов. – М.: Физматгиз, 1962. – 476 с.

4. Huffman D. A. The synthesis of sequential switching circuits. 1954, vol. 257, № 3, p. 161–190; № 4. p. 275–303.

5. Kochen M. Extension of moore-shannon model for relay circuits. – "IBM Journ. Res. and Devel.", 1959, vol. 3, № 2, p. 169–186.

6. Mealy G. H. A method for synthesizing sequential circuits. – "BSTJ", 1955, vol. 34, № 5, p. 1045–1079.

7. Wilkes M. V., Stringer J. B. Microprogramming and the design of the control circuits in an electronic digital computer. – "Proc. Cambridge Philos. Soc". 1953, vol. 49, № 4, p. 230.

8. Wilkes M. V. Microprogramming. – "Proc. East. Joint Corn-put. Conf.", 1959, vol. NT-114, № 7. p. 18–20.

9. Wilkes M. V., Renwick W., Wheeler D. J. The design of the control unit of an electronic digital computer. – "Proc. of the Inst of El. Eng.", pt. B, 1958, vol. 105, № 20, 121 p.

10. Hayes J.P. Computer Architecture and Organization. McGRAW-Hill, 1988.

11. Agerwala, T.: "Microprogram Optimization: A Survey", IEEE Trans. Comput., vol. C-25, pp. 962–973, October 1976.

12. Das, S. R., D. K. Banerji, and A. Chattopadhyay: "On Control Memory Minimization in Microprogrammed Computers", IEEE Trans. Comput., vol. C-23, pp. 845–848, September 1973.

13. Davidson, S., et al.: "Some Experiments in Local Microcode Compaction for Horizontal Machines", IEEE Trans. Comput., vol. C-30, pp. 4«M77, July 1981.

14. Agrawala, A. K., and T. G. Rauscher: Foundations of Microprogramming: Architecture, Software, and Applications, Academic, New York, 1976.

15. Andrews, M.: Principles of Firmware Engineering in Microprogram Control, Computer Science Press, Potomac, Md., 1980.

16. Husson, S. S.: Microprogramming: Principles and Practices, Prentice-Hall, Englewood Cliffs, N.J., 1970.

17. Kohavi, Z., Switching and Finite Automata Theory, 2d ed., McGraw-Hill, New York, 1978.

8.8. Питання до розділу 8

1. Призначення пристрою керування
2. Що таке мікрооперація?
3. Що таке мікронаказ?
4. Що таке мікрокоманда?
5. Що таке мікропрограма?
6. Назвіть місце поступлення керуючих сигналів
7. Назвіть два основних методи побудови логіки формування керуючих сигналів
8. В чому заключається принцип керування "жорсткої" або "запаяної" логіки?
9. В чому заключається принцип мікропрограмного керування?
10. Наведіть типову структурну схему пристрою керування з жорсткою логікою та поясніть її роботу
11. Для чого призначений блок синхроімпульсів?
12. Для чого призначений лічильник тактів?
13. Для чого призначені дешифратор коду операції та дешифратор тактів?
14. Назвіть методи проектування пристрою керування з жорсткою логікою
15. В чому заключається суть методу методом таблиць станів?
16. Що таке абстрактна та структурна моделі цифрового автомату?
17. Наведіть формальний опис абстрактного автомату
18. Опишіть автомат Мілі
19. Опишіть автомат Мура
20. Опишіть С-автомат
21. Опишіть таблицю переходів, таблицю виходів та таблицю з'єднань автомату Мілі
22. Опишіть таблицю переходів, таблицю виходів та таблицю з'єднань автомату Мура
23. Опишіть таблицю переходів, таблицю виходів та таблицю з'єднань С-автомату.
24. Як будується граф автомату Мілі?
25. Як будується граф автомату Мура?
26. Як будується граф С-автомату?
27. Поясніть етапи канонічного методу структурного синтезу цифрового автомату
28. Як використовують тактовані елементи часової затримки при побудові пристрою керування?
29. Як використовують лічильники при побудові пристрою керування?
30. Як будується часова діаграма роботи комп'ютера?
31. Які основні принципи покладені в основу побудови пристрою мікропрограмного керування?
32. Приведіть формат мікрокоманди

33. Приведіть структуру пристрою мікропрограмного керування та поясніть організацію його роботи.

34. Які функції контролера послідовності мікрокоманд?

35. Як формується адреса мікрокоманди?

36. Як організовані мікропрограми в пам'яті мікрокоманд?

37. Що таке горизонтальне мікропрограмування?

38. Що таке вертикальне мікропрограмування?

39. Які є способи зменшення кількості керуючих бітів мікрокоманди при використанні горизонтального мікропрограмування?

40. Порівняйте пристрої керування з жорсткою логікою та пристрої мікропрограмного керування.