Лабораторна робота № 5

Перехідні процеси в цифрових схемах. Перегони.

Мета роботи: *Розглянути перехідні процеси в цифрових схемах*

1. Формувачі імпульсів

На рис. 5.1 зображено схему формувача імпульсів на логічних елементах І-НЕ. Для формування імпульсу в цій схемі використовується затримка сигналу на логічному елементі.

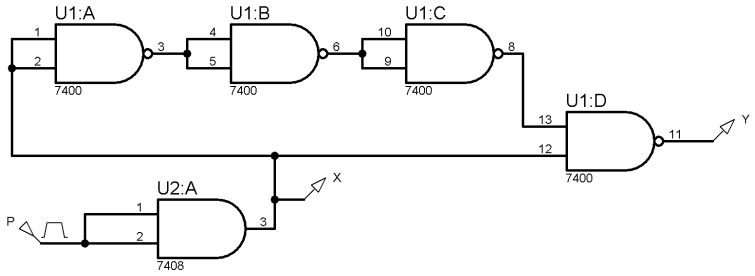

Рис 5.1 Схема формувача імпульсів.

Недоліком цієї схеми є те, що для формування імпульсів досить великої тривалості потрібно використати велику кількість логічних елементів. Цього недоліку немає у схемі на рис. 5.2, де використано затримуюче RC-коло.

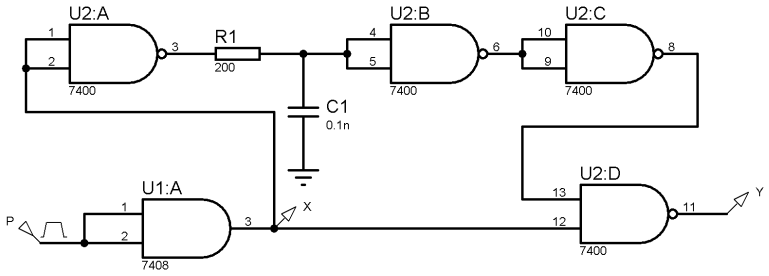

Рис 5.2 Схема формувача імпульсів з затримуючою RC коміркою.

Джерело вхідного імпульсу P є джерелом прямокутних імпульсів. Це є модель віртуального генератора в Proteus VSM. Основні параметри цієї моделі: PER – період повторення імпульсів, FREQ – частота імпульсів, V1 та V2 – низькорівневе та високорівневе значення виходу, PW – тривалість частини імпульса, коли V=V2, TR та TF – відповідно тривалості переднього та заднього фронтів, TD – затримка першого перепаду відносно моменту часу t=0.

Елементи R та C (резистор та ємність) слід обирати в підкатегорії Generic в меню вибору елементів, а елемент GROUND в меню Terminal Modes.

2. Генератори імпульсів

На рис. 5.3 зображено схему генератора серії імпульсів на логічних елементах І-НЕ. Для формування імпульса в цій схемі використовується затримка сигналу на логічному елементі. Серія імпульсів буде генеруватися доти, доки на вході 1 буде напруга логічної одиниці. Цю схему також можна модифікувати із використанням затримуючого RC-кола.

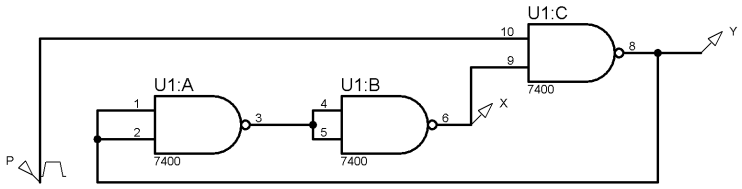

Рис. 5.3 Схема генератора серії імпульсів.

Слід зазначити, що в разі подачі на вхід P схеми на рис. 5.3 сталої логічної одиниці вона починає працювати як автоколивальний мультивібратор. Але моделювання такого результату не дає, якщо не визначити початкові умови (зміщення або імпульс запуску).

3. Перегони в цифрових схемах

Затримки в логічній схемі складаються з затримки логічних елементів та затримки розповсюдження сигналів по колам зв'язку між ними. На рис. 5.4 зображено фрагмент схеми зі зв'язками виконаними таким чином, щоб шлях сигналу на вхід елементів відрізнявся.

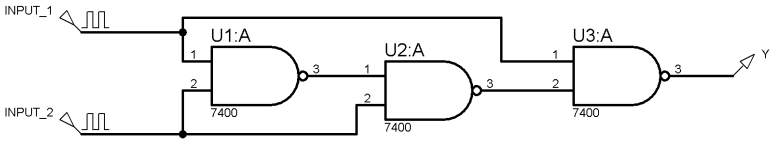

Рис. 5.4 Схема з перегонами.

4. Тригер Вебба

На рис. 5.5.1-5.5.5 зображені синхронні тригери Вебба. Дані тригери мають ще назву тригери з самоблокуванням або схеми

трьох тригерів. Зокрема на рис 5.5.1 зображена схема шестиелементного D-тригера. Дана схема має противоперегонні властивості, вона допускає для своїх елементів будь-яке співвідношення затримок.

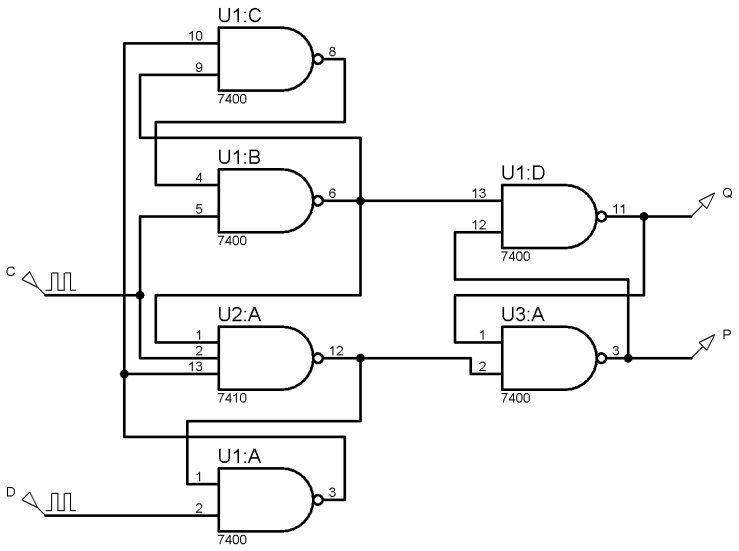

Рис. 5.5.1 Схема шестиелементного D-тригера.

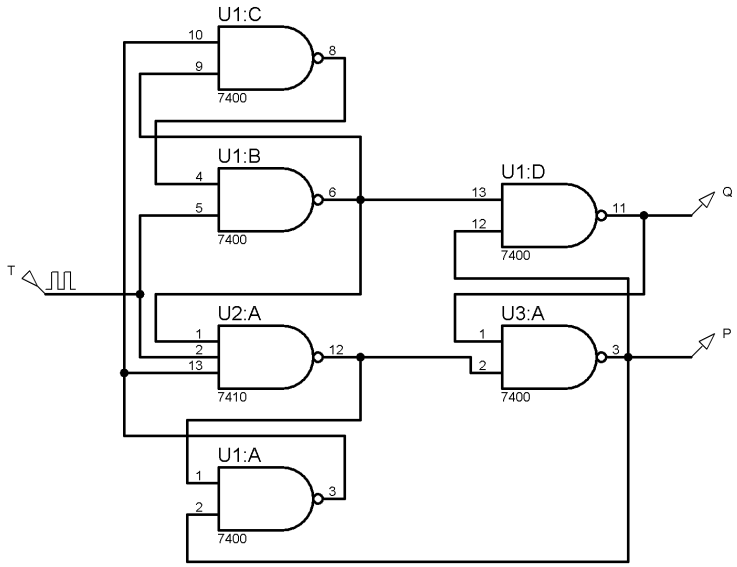

Рис. 5.5.2 Схема шестиелементного Т-тригера.

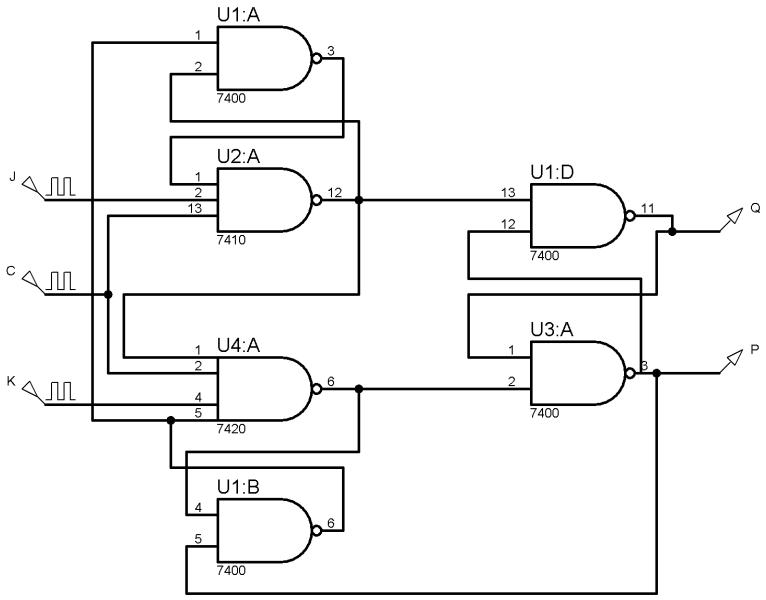

Рис. 5.5.3 Схема шестиелементного JK-тригера.

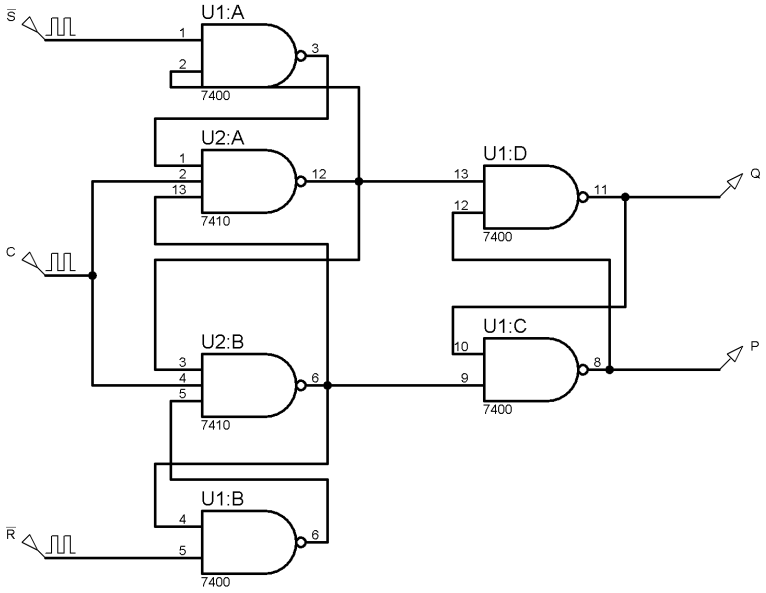

Рис. 5.5.4 Схема шестиелементного RS-тригера з інвертованими входами.

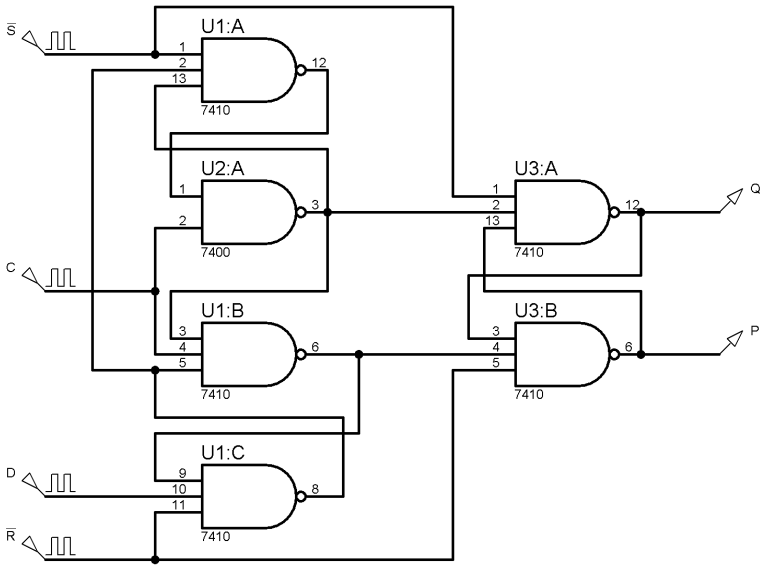

Рис. 5.5.5 Схема шестиелементного D-тригера з додатковими $\overline{S}$ та $\overline{R}$ входами.

Лабораторне завдання

1. Введіть у Proteus схему формувача імпульсів із затримкою на логічних елементах. (рис. 5.1). Вважайте, що амплітуда вхідного імпульсу 4 В, тривалість цього імпульсу 100 нс, його період 200 нс. Тривалості фронту і спаду імпульсу задайте по 1 нс. Затримка першого фронту може бути нульовою.
2. Задайте завдання на моделювання перехідного процесу. Виведіть цифрові епюри для вхідних імпульсів X та вихідних Y та поясніть їх.
3. Повторіть п.2 для випадків одного та п’яти логічних елементів у колі затримки. Поясність зміни у тривалості

вихідного імпульсу. Визначте середню затримку на одному логічному елементі.

4. Введіть у Proteus схему формувача імпульсів із затримуючим RC-колом. (рис. 5.2). Повторіть моделювання за параметрів вхідних імпульсів з п.1. Виведіть та поясніть епюри для X та Y. Визначте тривалість вихідних імпульсів.

5. Змінюючи значення R та C змініть тривалість імпульсу до значення згідно свого варіанту.

Таблиця 5.1

h4	*h5*	*h6*	Тривалість імпульсу
0	0	0	40 нс
0	0	1	45 нс
0	1	0	50 нс
0	1	1	55 нс
1	0	0	60 нс
1	0	1	65 нс
1	1	0	70 нс
1	1	1	75 нс

6. Введіть у Proteus схему генератора серії імпульсів (рис. 5.3). Для параметрів вхідного джерела з п.1 проведіть моделювання. Виведіть та поясніть епюри вхідної напруги V(1) та вихідного сигналу Y.

7. Збільшить у 3 рази тривалість і період вхідних імпульсів і повторіть моделювання.

8. Зберіть схему, що зображена на рис. 5.4. Побудуйте часові залежності роботи даної схеми. Поясніть причини неправильної роботи даної схеми. Запропонуйте модифікацію даної схеми, з метою уникнення проблем, що має дана схема.

9. Згідно свого варіанту зберіть, та промоделюйте роботу тригера Вебба.

Таблиця 5.2

h7	*h8*	*h9*	Схема тригера
0	*0*	*0*	Рис. 5.5.1
0	*0*	*1*	Рис. 5.5.2
0	*1*	*0*	Рис. 5.5.3
0	*1*	*1*	Рис. 5.5.4
1	*0*	*0*	Рис. 5.5.5
1	*0*	*1*	Рис. 5.5.3
1	*1*	*0*	Рис. 5.5.4
1	*1*	*1*	Рис. 5.5.5

Побудуйте часові залежності сигналів на виходах всіх логічних елементів Вашого тригера. Поясніть дані залежності.

10. Побудуйте та промоделюйте роботу 6-розрядних лічильників з послідовним та паралельним переносом. Оцініть величину затримок вихідних сигналу. Зробіть висновок, щодо доцільності використання лічильника з паралельним переносом.

11. Використавши два джерела PULSE та логічний елемент АБО-НІ створіть схему для демонстрації перегонів по входу. Поясніть часові залежності. Визначте граничні значення вхідної напруги, в межах якої вхідний сингал вважається одиницею (нулем).

Контрольні запитання:

1. Поясність принцип роботи тригера Вебба.
2. Які Ви знаєте способи боротьби з перегонами в схемах?
3. Поясніть принцип роботи генератора імпульсів, зображеного на рис. 3.
4. Які переваги лічильників з паралельним перенесенням?
5. Що таке перегони по входу.

Лабораторна робота № 6

Синтез мікропрограмного (керуючого) автомата у вигляді автомата Мілі

Мета роботи: *Провести структурний синтез керуючого автомата Мілі.*

Керуючий автомат призначається для управління операційним автоматом і виконання на ньому певного мікроалгоритму. Взагалі цифровий автомат – це пристрій, який перетворює вхідну інформацію в вихідну по заданому правилу. Виділяють абстрактну та структурну теорії автоматів. Абстрактна теорія є продовженням теорії алгоритмів, вона вивчає поведінку автомата у зовнішньому середовищі, а структурна теорія є продовженням абстрактної, і дозволяє визначити структуру вхідних, вихідних сигналів та структуру самого автомата.

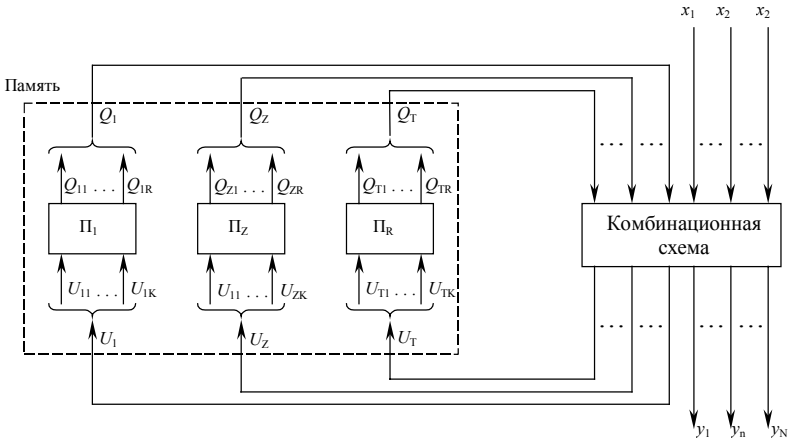

Рис. 6.1. Принципова схема керуючого автомата

Склад пристрою

Автомат складається з пам’яті (певного набору тригерів) та комбінаційної схеми (рис. 6.1), що складається з елементарних логічних елементів і виробляє вихідні керуючі сигнали та функції збудження для тригерів.

Етапи проектування

1. Побудова змістовної схеми алгоритму.
2. Побудова блок-схеми закодованого мікроалгоритму.
3. Побудова граф-схеми переходів автомата Мілі.
4. Побудова таблиць переходів автомата Мілі.
5. Кодування станів автомата.
6. Побудова структурної таблиці переходів-виходів автомата Мілі.
7. Визначення системи рівнянь переходів.
8. Визначення системи рівнянь виходів.
9. Побудова функціональної схеми автомата.

Приклад побудови мікропрограмного автомата у вигляді автомата Мілі

Задача:

1. Розробити функціональну схему керуючого автомата, що обчислює суму позитивних елементів непарних стовпців масиву A[n,m].
2. Мікропроцесорний автомат необхідно реалізувати у вигляді автомата Мілі.
3. Оптимальну функціональну схему керуючих частин автомата синтезувати на елементах І, АБО, НЕ.
4. В якості пам’яті використовуйте JK-тригери, доповнюючи її необхідними по алгоритму функціональними схемами.

Змістовна схема алгоритму

До складу змістовної схеми алгоритму рис. 6.4 входять операційні та умовні вершини. Наш алгоритм виконує обчислення суми позитивних елементів непарних стовпців масиву $A[n,m]$, використовуючи при цьому чотири (3) умовні вершини і десять (10) операційних вершин. Позначення операційних вершин показано на рис. 6.2. Позначення умовних вершин на рис. 6.3. Перевірка елементів масивів виконується від стовпчика до стовпчика.

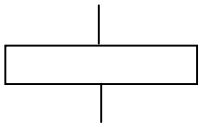

Рис.6.2. Операційна вершина

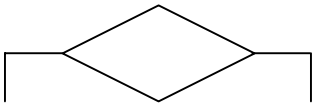

Рис. 6.3. Умовна вершина

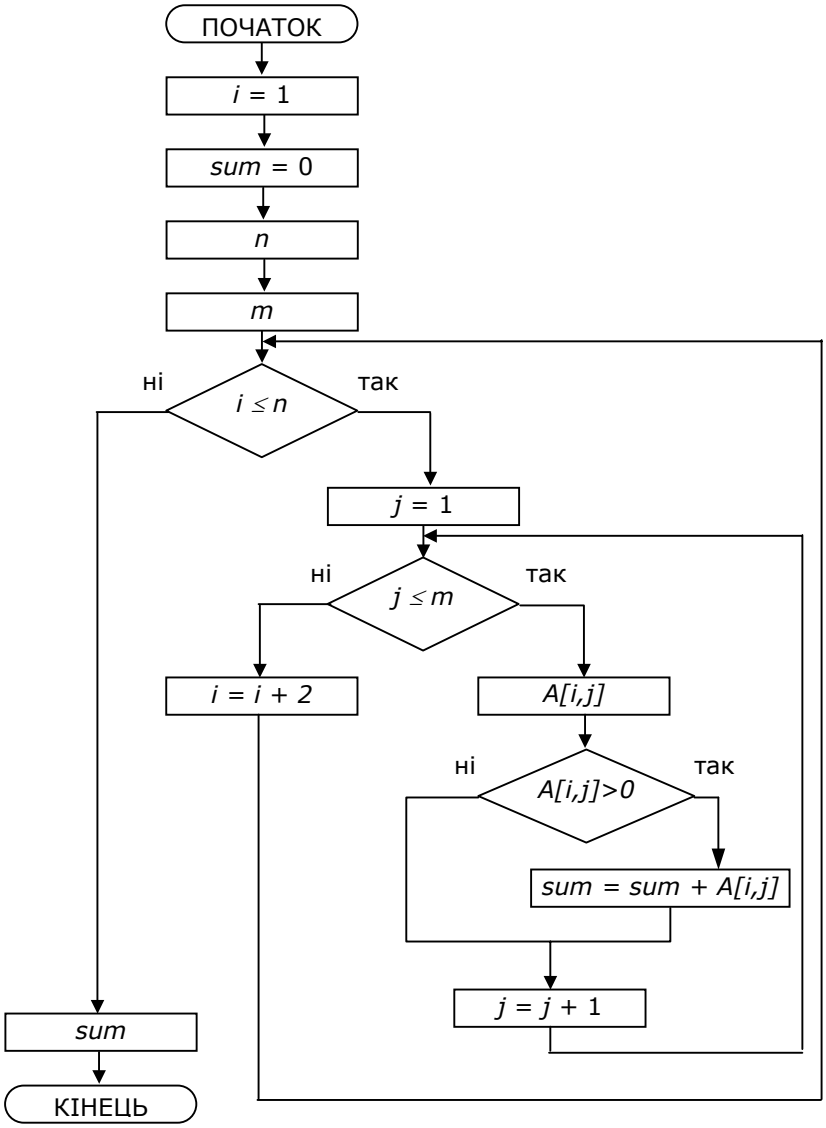

Рис. 6.4. Змістовна схема алгоритму

Змістовна таблиця кодування операційних та умовних вершин

Кожна вершина, чи то операційна чи умовна, кодується. При чому, якщо, мікрооперації повторюються і умовні вершини повторюються, вони кодуються однаково. У даному прикладі мікрооперації повторюються двічі, тому, однакові вершини ми можемо кодувати одним кодом. Таблиця кодування вершин зображена у таблиці 6.1.

Таблиця 6.1. Таблиця кодування вершин

Код	Зміст	Примітка
mY_1	$i = 1$	ініціалізація лічильника кількості стовпчиків
mY_2	$sum = 0$	ініціалізація результуючого значення
mY_3	n	завантаження до відповідного регістру значень розмірності матриці
mY_4	m	
mY_5	$j = 1$	ініціалізація лічильника кількості елементів в поточному стовпчику
mY_6	$A[i,j]$	завантаження до відповідного регістру значення елемента матриці
mY_7	$sum = sum + A[i,j]$	додавання до результуючого значення величини елемента, який задовольняє всім умовам фільтрації
mY_8	$j = j + 1$	перехід до дослідження наступного елемента стовпчика матриці

mY_9	$i = i + 2$	перехід до дослідження наступного непарного стовпчика матриці (парний стовпчик пропускається)
mY_{10}	*sum*	виведення результату
X_1	$i \le n$	умовна вершина: так – дослідження чергового стовпчика, ні – всі стовпчики досліджені
X_2	$j \le m$	умовна вершина: так – дослідження чергового елемента, ні – всі елементи чергового стовпчика досліджені
X_3	$A[i,j] > 0$	умовна вершина: так – елемент матриці є додатним, ні – умова фільтрації не виконується

mYi – мікрооперації, що виконує операційний автомат, X_j – сигнали, що видає операційний автомат керуючого автомату.

Закодована мікрооперaційна схема алгоритму

Закодована мікрооперaційна схема алгоритму будується на основі змістовної схеми алгоритму (рис. 6.4) і таблиці кодування операційних та умовних вершин (табл. 6.1), шляхом заміни відповідних блоків. Схема алгоритму зображена на рис. 6.5.

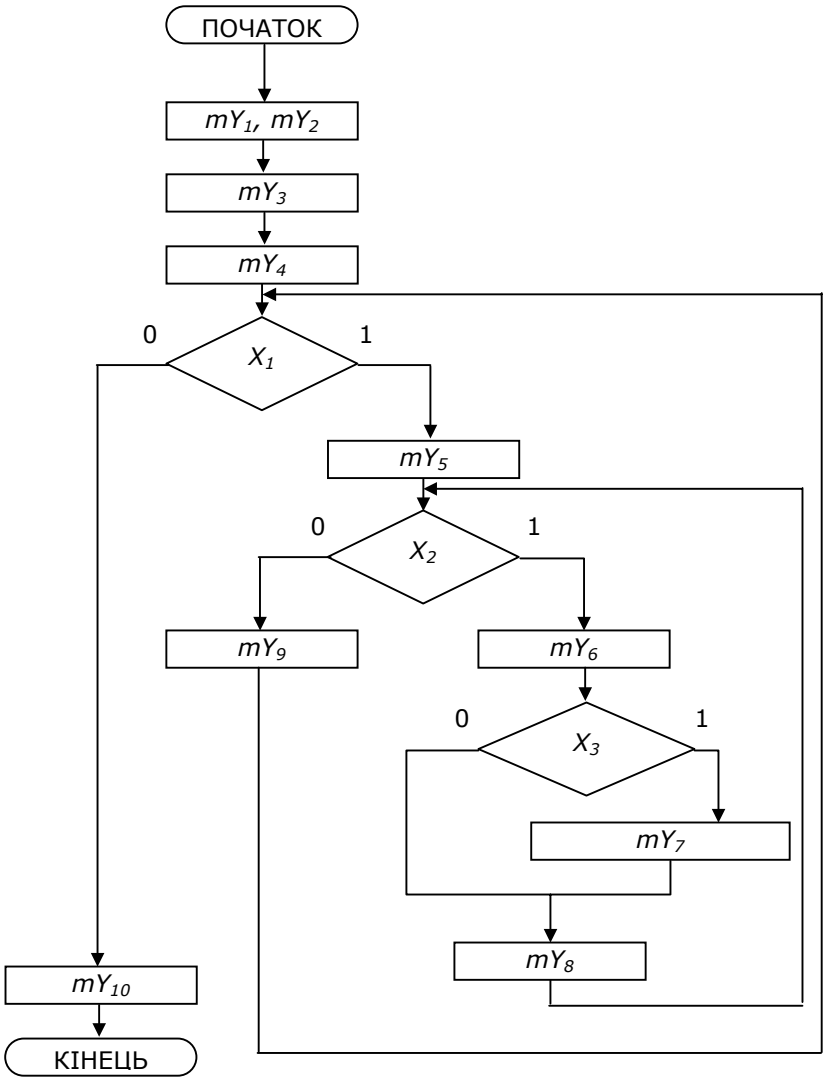

Рис. 6.5. Закодована схема ГСА

Об’єднання мікрооперацій mY_1 та mY_2 в одну операційну вершину (в одну мікрокоманду) можливо лише тоді, коли обидві мікрооперації є незалежними одна від іншої (результати виконання одної мікрооперації не впливають на результати іншої) та можуть бути виконані за один такт одночасно.

Синтез автомата Мілі

На етапі отримання зазначеної граф-схеми автомата (ГСА) входи вершин, наступних за операторними, відзначають символами a_0, a_1,… за такими правилами:

1) символом a_0 відзначають вхід вершини, наступної за початкової, а також вхід кінцевої вершини.
2) входи усіх вершин, наступних за операторними, повинні бути відзначені.
3) входи різних вершин, за винятком кінцевої, відзначаються різними символами.
4) якщо вхід вершини відзначається, то тільки одним словом.

Використавши дані правила, побудуємо відмічений ГСА для автомата Мілі. В відміченому ГСА замінимо всі мікрооперації mY_j на відповідні керуючі сигнали Y_j. Складаємо закодовану мікрокомандну схему алгоритму (рис. 6.6).

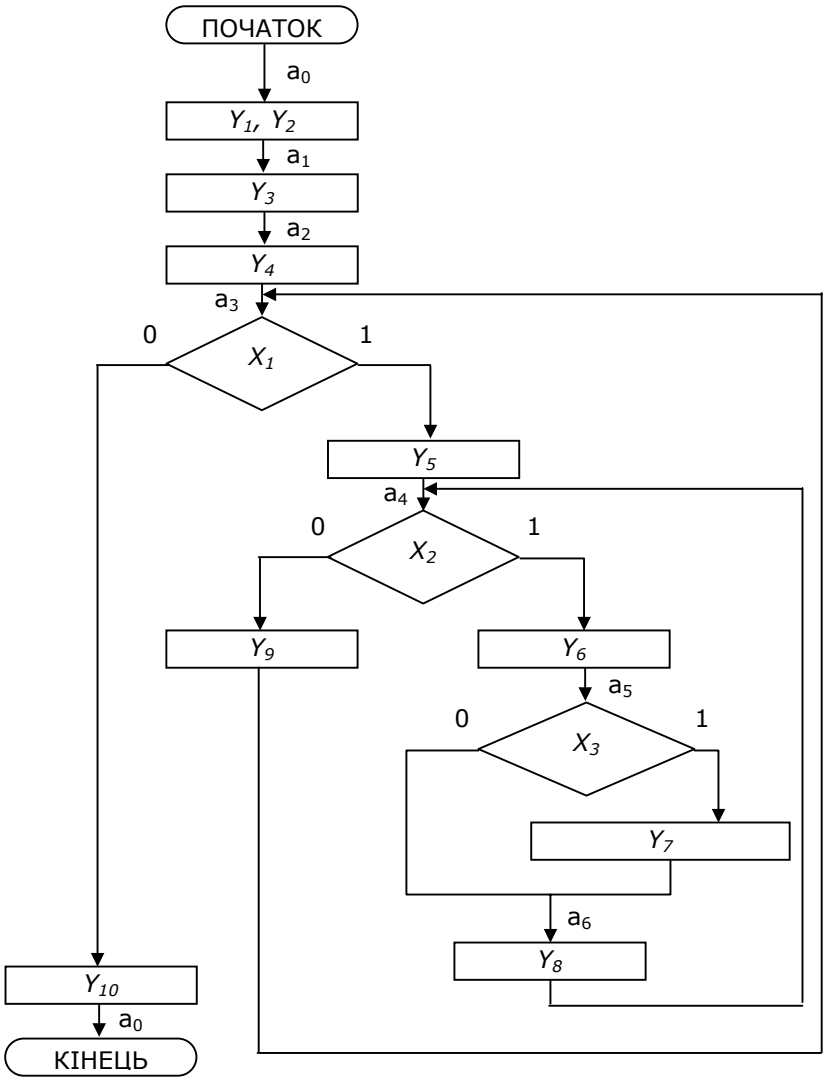

Рис. 6.6. Відмічена ГСА

Граф-схема переходів

Будуємо граф-схему переходів автомата Мілі (рис. 6.7). Граф-схема будується на основі рис. 6.6. Кружечками позначаються можливі стани автомата. Стрілки указують на перехід зі стану і до стану j. Над стрілкою указується, під яким вхідним сигналом станеться перехід, і, що при цьому буде на виході автомата.

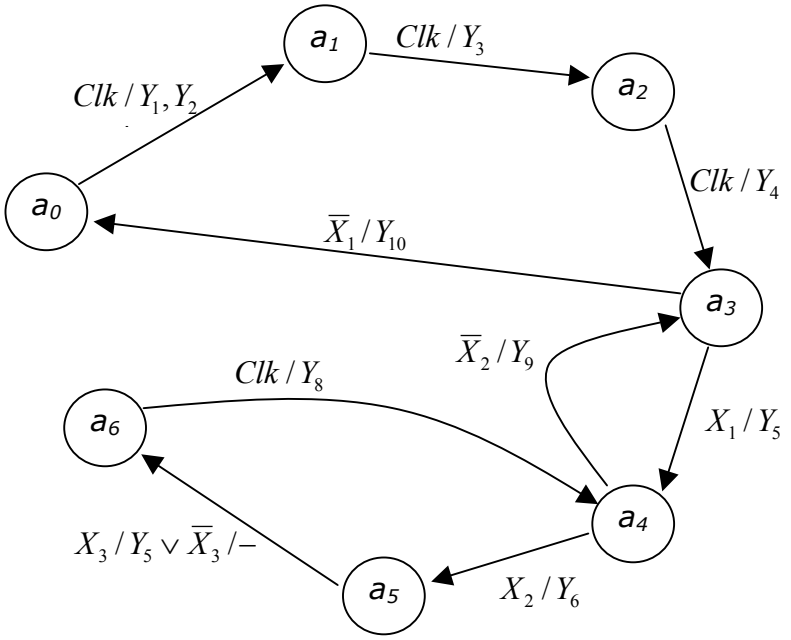

Рис. 6.7. Граф-схема переходів керуючого автомата

На підставі відміченої ГСА чи граф-схеми переходів керуючого автомата можна побудувати таблиці переходів-виходів. Для мікропрограмних автоматів таблиця переходів-виходів будується у вигляді списку і розрізняються пряма і

зворотна таблиці. Для даного автомата пряма таблиця представлена в таблиці 6.2., зворотна – в таблиці 6.3.

У наведених таблицях a_m – початковий стан, a_s – стан переходу, X – умова (вхідний сигнал), зо забезпечує перехід станів, Y – вихідний сигнал, що виробляється автоматом при переході між станами.

Таблиця 6.2. Прямая таблиця переходів-виходів автомата Мілі

Початковий стан	Стан переходу	X (умова переходу)	Y (вихідний сигнал, що виробляється при переході)
a_0	a_1	1	Y_1, Y_2
a_1	a_2	1	Y_3
a_2	a_3	1	Y_4
a_3	a_4	X_1	Y_5
	a_0	$\overline{X}_1$	Y_{10}
a_4	a_5	X_2	Y_6
	a_3	$\overline{X}_2$	Y_9
a_5	a_6	X_3	Y_7
	a_6	$\overline{X}_3$	–
a_6	a_4	1	Y_8

Таблиця 6.3. Зворотна таблиця переходів-виходів автомата Мілі

Початковий стан	Стан переходу	X (умова переходу)	Y (вихідний сигнал, що виробляється при переході)
a_3	a_0	$\overline{X}_1$	Y_{10}
a_0	a_1	1	Y_1, Y_2
a_1	a_2	1	Y_3
a_2	a_3	1	Y_4
a_4		$\overline{X}_2$	Y_9
a_3	a_4	X_1	Y_5
a_6		1	Y_8
a_4	a_5	X_2	Y_6
a_5	a_6	X_3	Y_7
a_5		$\overline{X}_3$	–

У вихідному автоматі кількість станів M = 7, отже, число елементів пам'яті

$$m =]\log_2 M[=]\log_2 7[= 3$$

Для синтезу використовуємо JK-тригери.

Кодуємо внутрішні стани автомата. використовуючи для цього карту Карно (табл. 6.4) методом сусіднього кодування. Рекомендується самостійно закодувати стан за допомогою евристичного алгоритму.

Таблиця 6.4. Кодування станів автомата

$Q_1\backslash Q_2Q_3$	00	01	11	10
0	a_0	a_1	a_6	a_5
1	a_3	a_2		a_4

$a_0 – 000$
$a_1 – 001$
$a_2 – 101$
$a_3 – 100$
$a_4 – 110$
$a_5 – 010$
$a_6 – 011$

Будуємо пряму структурну таблицю переходів-виходів автомата Мілі (табл. 6.5). Уданій таблиці у стовпцях $K(a_m)$ і $K(a_s)$ вказується код вихідного стану та стану переходу відповідно. У стовпці функцій збудження ФЗ вказуються ті значення функцій збудження, які на даному переході обов'язкові. Решта (тобто рівні 0 або приймаючі невизначені значення) не вказуються. Це еквівалентно тому, що всім невизначеним значенням функцій збудження приписується значення 0, що в загальному випадку не дає мінімальної функції, проте в реальних автоматах мінімізація зазвичай не робиться по причині її неефективності.

Для прикладу: при переході з стану a_0 (000) в стан a_1 (001) стан третього тригера має змінитись з 0 на 1. Це можливо подавши на J_3 високий сигнал.

Таблиця 6.5. Структурна таблиця переходів-виходів автомата Мілі.

Початковий стан (a_m)	$K(a_m)$	Стан переходу (a_s)	$K(a_s)$	X (умова переходу)	Y (вихідний сигнал, що виробляється при переході)	ФЗ
a_0	000	a_1	001	1	Y_1, Y_2	J_3
a_1	001	a_2	101	1	Y_3	J_1

a_2	101	a_3	100	1	Y_4	K_3
a_3	100	a_4	110	X_1	Y_5	J_2
		a_0	000	$\overline{X}_1$	Y_{10}	K_1
a_4	110	a_5	010	X_2	Y_6	K_1
		a_3	100	$\overline{X}_2$	Y_9	K_2
a_5	010	a_6	011	X_3	Y_7	J_3
		a_6	011	$\overline{X}_3$	–	J_3
a_6	011	a_4	110	1	Y_8	J_1K_3

Система рівнянь переходів

Для отримання функцій збудження (рівн. 6.1) поступаємо таким чином. Вираз для кожної функції виходить у вигляді логічної суми додатків виду a_iX, де a_i – початковий стан, X – умова переходу. Для спрощення отриманих виразів виконуємо всі можливі операції склеювання і поглинання. Щоб отримати на вхід K_1 сигал, що відповідає логічній одиниці, потрібно щоб автомат знаходився в стані a_3 і прийшов вхідний сигнал $\overline{X}_1$, або щоб автомат знаходився в стані a_4 і прийшов вхідний сигнал X_2 і.т.д.

$$\begin{cases} J_1 = a_1 \vee a_6 \\ J_2 = a_3 X_1 \\ J_3 = a_0 \vee a_5 \\ K_1 = a_3 \overline{X}_1 \vee a_4 X_2 \\ K_2 = a_4 \overline{X}_2 \\ K_3 = a_2 \vee a_6 \end{cases} \qquad (6.1)$$

Система рівнянь виходів

Складаємо систему рівнянь виходів (6.2). Ця система складається на основі таблиці 6.5., де X – вхідні, Y – вихідні сигнали. Для першого рівняння системи справедливими є твердження, що ми отримуємо вихідний стан Y_1 при переході зі стану a_0 під впливом синхроімпульсу і.т.д.

$$\begin{cases} Y_1 = a_0 \\ Y_2 = a_0 \\ Y_3 = a_1 \\ Y_4 = a_2 \\ Y_5 = a_3 X_1 \\ Y_6 = a_4 X_2 \\ Y_7 = a_5 X_3 \\ Y_8 = a_6 \\ Y_9 = a_4 \overline{X}_2 \\ Y_{10} = a_3 \overline{X}_1 \end{cases} \qquad (6.2)$$

Побудова функціональної схеми автомата

Для побудови функціональної схеми автомата за отриманими виразами необхідно або замінити a_i його значенням через $Q_1Q_2Q_3$ або отримати сигнал, відповідний a_i. Зазвичай використовують другий спосіб і для отримання сигналу a_i застосовують так званий дешифратор станів, на вхід якого надходять сигнали з виходів елементів пам'яті $Q_1Q_2Q_3$. Крім того, при побудові схеми намагаються виділити загальні частини, що зустрічаються у функціях збудження або вихідних сигналах.

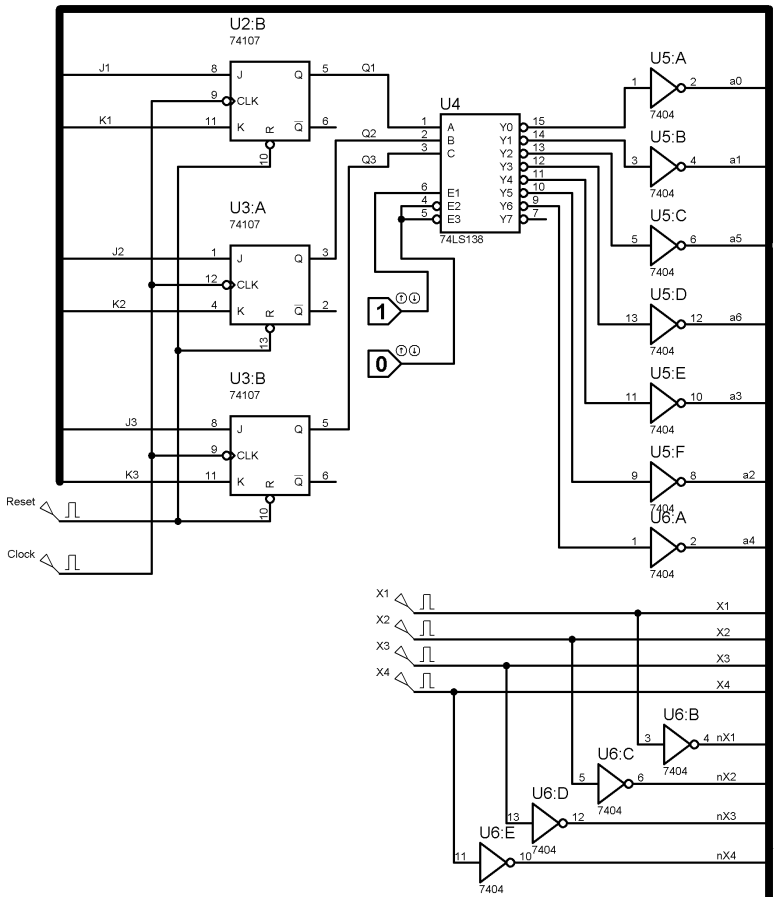

Рис. 6.6. Частина функціональної схема автомата Мілі, що складається з комірок пам’яті (JK тригери) та дешифратор (74LS138).

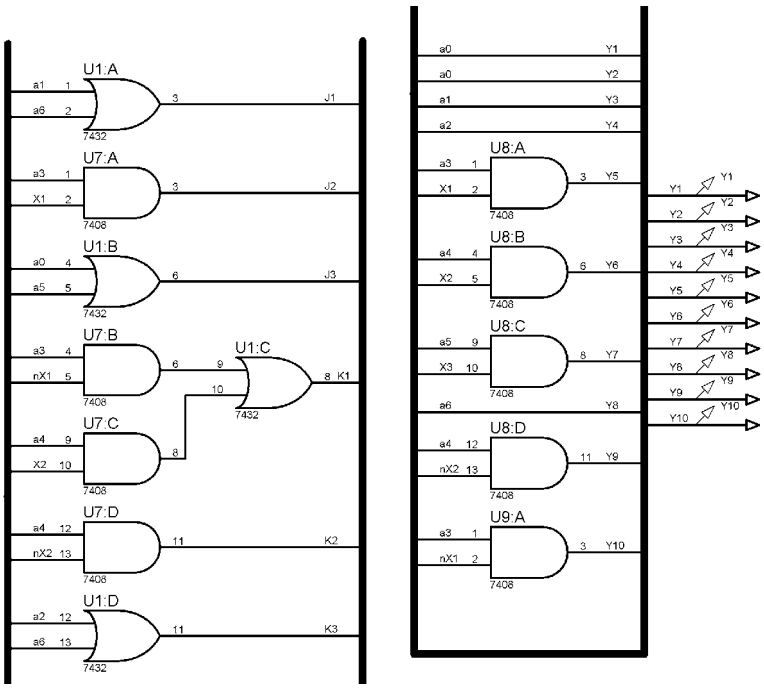

Рис. 6.7. Частини функціональної схема автомата Мілі, що складається з комбінаційних схем визначення функцій переходів (схема ліворуч) та функцій виходів (схема праворуч).

На рисунках 6.6. та 6.7. зображена схема автомата Мілі. Шина, що використовується в данних рисунках спільна.

Лабораторне завдання

1. Згідно вашого варіанту, розробіть функціональну схему керуючого автомата, що:

Таблиця 6.6

h4	*h5*	*h6*	*h7*	*h8*	**Завдання**
0	0	0	0	0	обчислює суму трьох максимальних елементів масиву A(n)
0	0	0	0	1	знаходить максимальні парні елементи масивів A(n), B(v) та міняє їх місцями
0	0	0	1	0	знаходить максимальний парний елемент масиву A(n,n) і замінює його мінімальним
0	0	0	1	1	знаходить максимальний парний елемент у кожнім рядку масиву A(n,n)
0	0	1	0	0	обчислює кількість парних елементів масивів A(n),B(m)
0	0	1	0	1	обчислює суму позитивних елементів масивів A(n),B(m)
0	0	1	1	0	обчислює суму позитивних діагональних елементів масиву A(n,n) та змінює їх за законом A(i,i)= A(n-i+1,n-i+1)*s (де s сума позитивних діагональних елементів масиву)
0	0	1	1	1	обчислює суму позитивних елементів непарних стовпців масиву A(n,m)
0	1	0	0	0	обчислює суму і кількість позитивних елементів у кожнім рядку масиву A(n,m)
0	1	0	0	1	обчислює суму парних позитивних елементів у кожнім непарному рядку масиву A(n,m)
0	1	0	1	0	обчислює суму парних позитивних елементів у масивах A(n,m), B(p)
0	1	0	1	1	виконує перестановку елементів першого й останнього стовпців A(n,n)
0	1	1	0	0	виконує перестановку максимальних елементів масивів A(n,n) і D(n,n)
0	1	1	0	1	виконує перетворення масиву A(n,n) у такий спосіб: до парних елементів масиву додасть 5, до непарних 3

0	1	1	1	0	обчислює суму трьох максимальних елементів масиву A(n,n,n)
0	1	1	1	1	виконує перетворення масивів A(n), B(v) у такий спосіб: четверті по номері негативні елементи масивів виконати операцію додавання між собою та збереження результату, а потім перші елементи масивів міняє місцями
1	0	0	0	0	виконує перетворення масиву A(n) у такий спосіб: поміняти місцями максимальний і мінімальний елементи цього масиву
1	0	0	0	1	виконує перетворення масивів A(n), B(n) у такий спосіб: поміняти місцями мінімальні елементи цих масивів
1	0	0	1	0	виконує перетворення масивів A(n), B(n) у такий спосіб: другі по номері парні елементи масивів виконати операцію додавання між собою, а потім замінити ним треті парні елементи
1	0	0	1	1	обчислює суму двох максимальних елементів масиву A(n,n)
1	0	1	0	0	виконує перетворення масиву A(n,n) у такий спосіб: виконати операцію додавання максимального елементи у кожному рядку і замінити значення першого елемента масиву на значення знайденої суми
1	0	1	0	1	обчислює суму трьох максимальних елементів масиву A(n,n)
1	0	1	1	0	виконує перетворення масиву A(n) у такий спосіб: до позитивних елементів масиву додасть значення максимального елемента цього масиву
1	0	1	1	1	виконує перетворення масиву A(n,n,n)

					у такий спосіб: до парних елементів масиву додасть 5, до непарних 3
1	1	0	0	0	знаходить максимальний та мінімальний елемент масиву A(n,n,n) та міняє їх місцями
1	1	0	0	1	обчислює кількість парних елементів масивів A(n,n),B(m)
1	1	0	1	0	знаходить максимальний парний елемент у кожнім рядку масиву A(n,n) та збільшує його на 2
1	1	0	1	1	виконує перестановку мінімальних елементів масивів A(n,n) і D(n,n)
1	1	1	0	0	виконує перетворення масивів A(n), B(v) у такий спосіб: до парних елементів масиву додасть 4, до непарних 2
1	1	1	0	1	виконує перетворення масиву A(n) у такий спосіб: до позитивних елементів масиву додасть суму парних елементів цього масиву
1	1	1	1	0	знаходить максимальні парні елементи масивів A(n,n), B(v) та міняє їх місцями
1	1	1	1	1	обчислює суму чотирьох максимальних елементів масиву A(n)

Мікропроцесорний автомат необхідно реалізувати у вигляді автомата Мілі. Функціональну схему керуючих частин автомата синтезувати на елементах:

Таблиця 6.7.

h9	*h 10*	Логічні елементи
0	0	І, АБО, НЕ
0	1	І-НЕ
1	0	АБО-НЕ
1	1	І, АБО- НЕ

В якості пам’яті використайте:

Таблиця 6.8.

h1	*h2*	*Тип тригера,*
0	*0*	*RS*
0	*1*	*T*
1	*0*	*JK*
1	*1*	*D*

2. Промоделюйте отриману функціональну схему керуючого автомата у Proteus. Переконайтесь у правильності її роботи.

3. Виведіть відповідні графіки для вхідних та вихідних сигналів, що будуть в повній мірі описувати роботу розробленого Вами автомату.

Контрольні питання:

1. Дайте визначення цифрового автомату.
2. З яких етапів складається структурний синтез цифрового автомата Мілі?
3. В чому полягає евристичний алгоритм визначення внутрішніх станів автомата?
4. Чим відрізняється керуючий автомат від операційного автомата?
5. JK-тригер є автоматом Мілі?

Лабораторна робота № 7

Синтез мікропрограмного (керуючого) автомата у вигляді автомата Мура

Мета роботи: *Провести структурний синтез керуючого автомата Мура.*

Керуючий автомат призначається для управління операційним автоматом та виконання на ньому певного мікроалгоритму.

Етапи проектування автомата Мура близькі до етапів проектування автоматів Мілі.

Етапи проектування

1. Побудова змістовної схеми алгоритму.
2. Побудова блок-схеми закодованого мікроалгоритму.
3. Побудова граф-схеми переходів автомата Мура.
4. Побудова таблиць переходів-виходів автомата Мура. Кодування станів автомата.
5. Побудова структурної таблиці переходів-виходів автомата Мура.
6. Визначення системи рівнянь переходів.
7. Визначення системи рівнянь виходів.
8. Побудова функціональної схеми автомата.

Приклад побудови мікропрограмного автомата у вигляді автомата Мура

Задача

1. Побудувати керуючий автомат, який обчислює кількість парних елементів масивів A(n), B(m).
2. Мікропроцесорний автомат необхідно реалізувати у вигляді автомата Мура.
3. Оптимальну функціональну схему керуючих частин автомата синтезувати на елементах системи І, АБО, НЕ, АБО-НЕ.
4. В якості пам’яті використайте D-тригери, доповнюючи її необхідними по алгоритму функціональними схемами.

Змістовна схема алгоритму

До складу змістовної схеми алгоритму (рис. 7.3) входять операційні та умовні вершини. Наш алгоритм виконує знаходження кількості парних елементів в двох масивах, використовуючи при цьому чотири (4) умовні вершини та дванадцять (12) операційних. Позначення операційних вершин показано на рис. 7.1., умовних – на рис. 7.2.

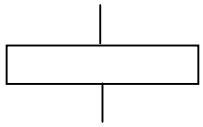

Рис.7.1. Операційна вершина

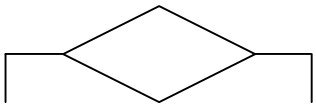

Рис. 7.2. Умовна вершина

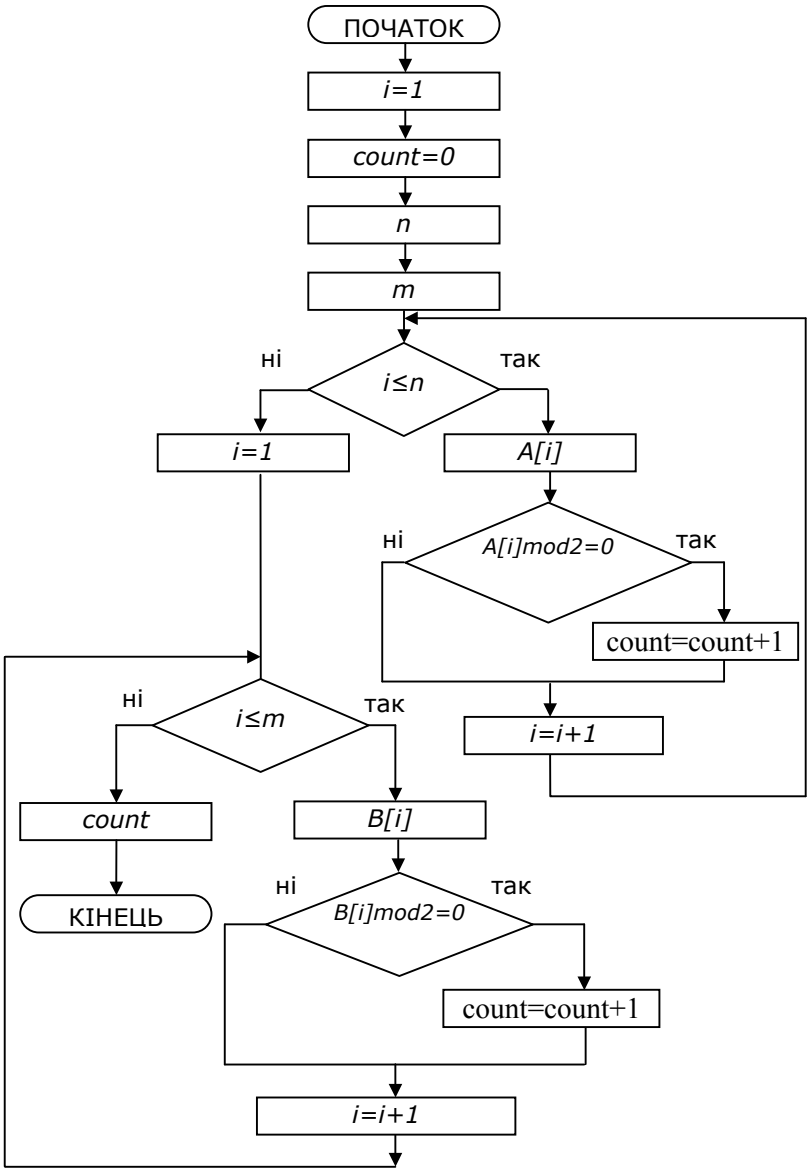

Рис. 7.3. Змістовна схема алгоритму

Змістовна таблиця кодування операційних та умовних вершин

Кожна вершина, чи то операційна чи умовна, кодується. При чому, якщо, мікрооперації повторюються і умовні вершини повторюються, вони кодуються однаково. У даному прикладі мікрооперації повторюються двічі, тому, однакові вершини ми можемо кодувати одним кодом. Таблиця кодування вершин зображена у табл. 7.1.

Таблиця 7.1.
Таблиця кодування вершин

Код	**Зміст**	**Примітка**
mY_1	$i = 1$	ініціалізація лічильника кількості елементів у масиві
mY_2	*count = 0*	ініціалізація результуючого значення
mY_3	*n*	завантаження до відповідного регістру значень розмірності масивів
mY_4	*m*	
mY_5	*A[i]*	завантаження до відповідного регістру значення елемента масиву А
mY_6	*count = count + 1*	врахування елемента, який задовольняє всім умовам фільтрації
mY_7	*i = i + 1*	перехід до дослідження наступного елемента масиву
mY_8	*B[i]*	завантаження до відповідного регістру значення елемента масиву В
mY_9	*count*	виведення результату

X_1	$i \le n$	умовна вершина: **так** – дослідження наступного елемента масиву A, **ні** – всі елементи досліджені
X_2	*A[i] mod 2 = 0*	умовна вершина: **так** – елемент масиву A є парним, **ні** – умова фільтрації не виконується
X_3	$i \le m$	умовна вершина: **так** – дослідження наступного елемента масиву B, **ні** – всі елементи досліджені
X_4	*B[i] mod 2 = 0*	умовна вершина: **так** – елемент масиву B є парним, **ні** – умова фільтрації не виконується

mYi – мікрооперації, що виконує операційний автомат, X_j – сигнали, що видає операційний автомат керуючого автомату.

Закодована мікрооперaційна схема алгоритму

Закодована мікрооперaційна схема алгоритму будується на основі змістовної схеми алгоритму (рис. 7.3) і таблиці кодування операційних та умовних вершин (табл. 7.1), шляхом заміни відповідних блоків. Схема алгоритму зображена на рис. 7.4.

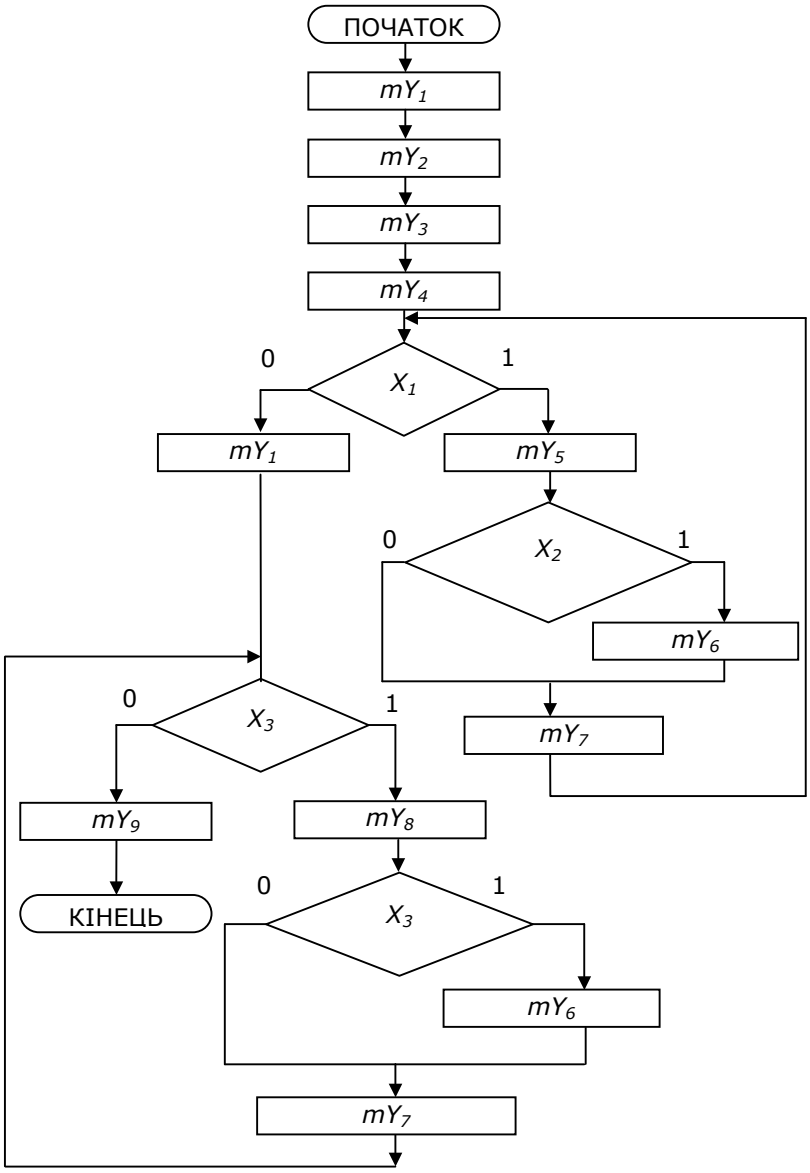

Рис. 7.4. Закодована мікрокомандна схема алгоритму

Об’єднання мікрооперацій mY_1,mY_2 в одному операційну вершину (в одну мікрокоманду) можливо лише тоді, коли обидві мікрооперації є незалежними одна від іншої (результати виконання одної мікрооперації не впливають на результати іншої) та можуть бути виконані за один такт одночасно.

Синтез автомата Мура

Для автомата Мура на етапі одержання зазначеної граф-схема автомата (ГСА) розмітка проводиться згідно з такими правилами:

1) символом S_0 відзначається початкова та кінцева вершини;
2) різні операторні вершини відзначаються різними символами;
3) усі операторні вершини повинні бути відзначені;

Використавши дані правила, побудуємо відмічений ГСА для автомата Мура, у якому замінимо всі мікрооперації mY_j на відповідні керуючі сигнали Y_j.

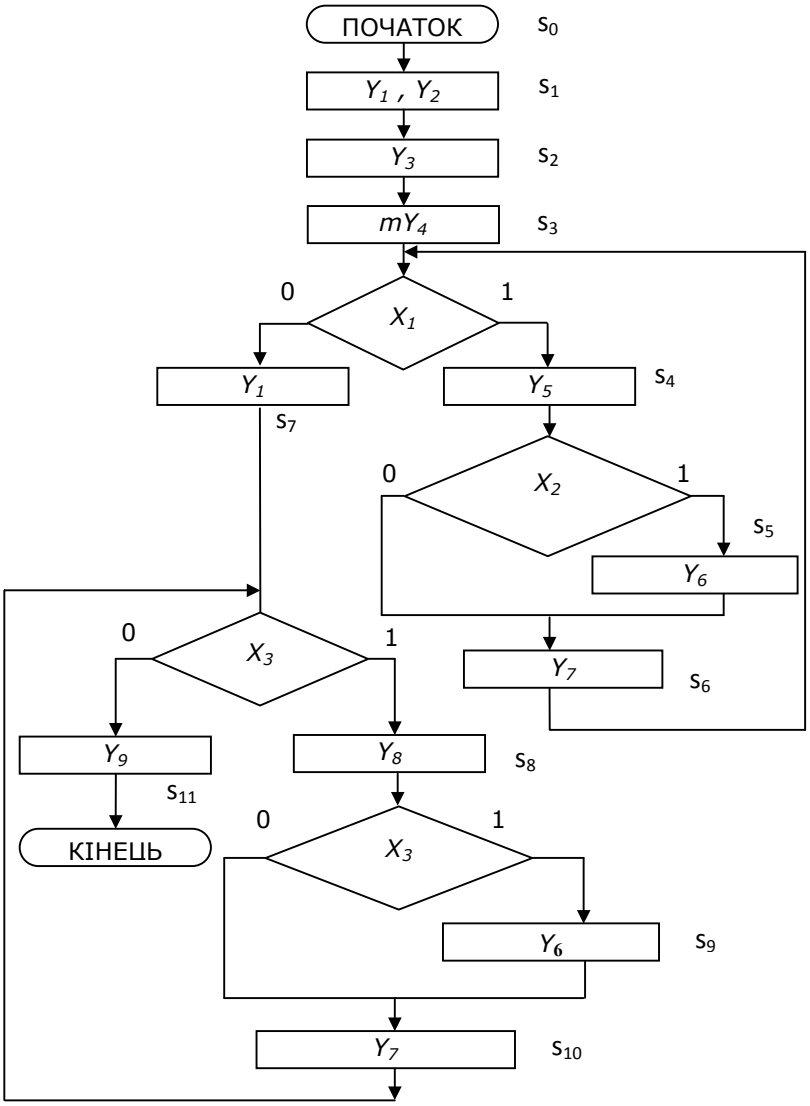

Рис. 7.5. Відмічена ГСА

Граф-схема переходів

Будуємо граф-схему переходів автомата Мура (рис. 7.6). Граф-схема будується на основі рис. 7.5. Кружечками позначаються можливі стани автомата. Стрілки вказують на перехід зі стану і до стану j. Над стрілкою указується, під яким вхідним сигналом станеться перехід. Коло станів відмічені виходи автомата.

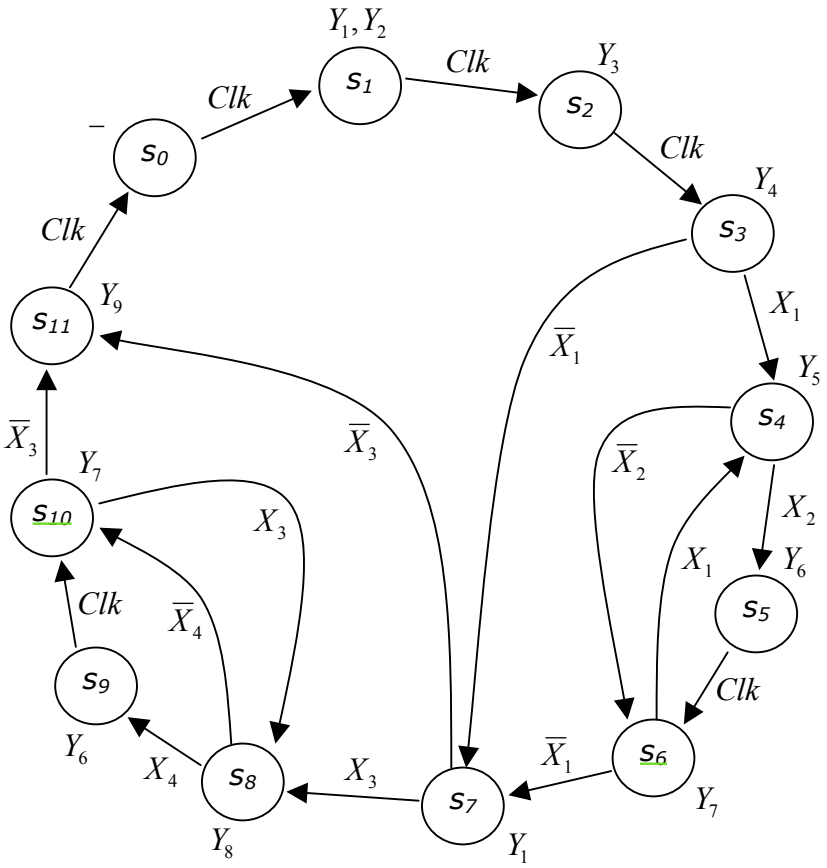

Рис. 7.6. Граф-схема переходів керуючого автомата

На підставі відміченої ГСА чи граф-схеми переходів керуючого автомата можна побудувати таблиці переходів-виходів. Для мікропрограмних автоматів таблиця переходів-виходів будується у вигляді списку і розрізняються пряма і зворотна таблиці. Для даного автомата пряма таблиця представлена в табл. 7.2, зворотна – в табл. 7.3.

Таблиця 7.2. Прямая таблиця переходів-виходів автомата Мура

Початковий стан s_m	Y (вихідний сигнал, що виробляється при переході)	Стан переходу s_k	X (умова переходу)
s_0	–	s_1	1
s_1	Y_1, Y_2	s_2	1
s_2	Y_3	s_3	1
s_3	Y_4	s_4	X_1
		s_7	$\overline{X}_1$
s_4	Y_5	s_5	X_2
		s_6	$\overline{X}_2$
s_5	Y_6	s_6	1
s_6	Y_7	s_7	$\overline{X}_1$
		s_4	X_1
s_7	Y_1	s_8	X_3
		s_{11}	$\overline{X}_3$
s_8	Y_8	s_9	X_4
		s_{10}	$\overline{X}_4$
s_9	Y_6	s_{10}	1

s_{10}	Y_7	s_8	X_3
		s_{11}	$\overline{X}_3$
s_{11}	Y_9	s_0	1

Таблиця 7.3. Зворотна таблиця переходів-виходів автомата Мура

Початковий стан s_m	Стан переходу s_k	Y (вихідний сигнал, що виробляється при переході)	X (умова переходу)
s_{11}	s_0	–	1
s_0	s_1	Y_1, Y_2	1
s_1	s_2	Y_3	1
s_2	s_3	Y_4	1
s_3	s_4	Y_5	X_1
s_6			X_1
s_4	s_5	Y_6	X_2
s_5	s_6	Y_7	1
s_4			$\overline{X}_2$
s_6	s_7	Y_1	$\overline{X}_1$
s_3			$\overline{X}_1$
s_7	s_8	Y_8	X_3
s_{10}			X_3
s_8	s_9	Y_6	X_4
s_9	s_{10}	Y_7	1
s_8			$\overline{X}_4$

s_{10}	s_{11}	Y_9	$\overline{X}_3$
s_7			$\overline{X}_3$

У вихідному автоматі кількість станів M = 12, отже, число елементів пам'яті

$$m =]\log_2 M[=]\log_2 12[= 4$$

Аналіз канонічного методу структурного синтезу автомата показує, що різні варіанти кодування станів автомата призводять до різних виразів функцій збудження пам'яті та функцій виходів, в результаті чого складність комбінаційної схеми істотно залежить від обраного кодування. Тому застосуємо для кодування станів автомата спеціальний алгоритм для D-тригерів. Алгоритм складається з декількох кроків:

1. Кожному стану автомата s_m ставиться у відповідність ціле число N_m, що дорівнює кількості переходів в стан s_m (N_m дорівнює числу появ s_m в полі таблиці переходів або кількості дуг, що входять в s_m за графічного способу задання автомата).
2. На наступному етапі числа $N_1, N_2, N_3, \ldots, N_m$ впорядковуються за спаданням.
3. Стан s_k з найбільшим N_k кодується кодом: 0000, оскільки автомат містить 4 елементи пам'яті.
4. Наступні 4 стани з отриманого списку кодуються кодами, що містять лише одну одиницю: 0001, 0010, 0100, 1000.
5. Для інших станів використовуються коди з двома одиницями, потім з трьома і так далі, поки не будуть закодовані всі стани.

Використаємо даний алгоритм для кодування станів нашого автомата.

Стан автомата s_m	Кількість переходів N_m	**Код $K(s_m)$**
s_4	2	0000
s_6	2	0001
s_7	2	0010
s_8	2	0100
s_{10}	2	1000
s_{11}	2	1100
s_0	1	0110
s_1	1	0011
s_2	1	1010
s_3	1	0101
s_5	1	1001
s_9	1	1110

Таким чином, маємо кодування, за якого чим більше існує переходів в деякий стан, тим менше одиниць в його коді. Оскільки для D-тригерів функції збудження однозначно визначаються кодом стану переходу, то очевидно, що вирази для функцій збудження будуть простішими.

Будуємо структурну таблицю переходів-виходів автомата Мура (табл. 7.4). У даній таблиці у стовпцях $K(s_m)$ і $K(s_k)$ вказується код вихідного стану та стану переходу відповідно. У стовпці функцій збудження ФЗ вказується ті значення функцій збудження, які на даному переході обов'язково рівні одиниці. Решта (тобто рівні 0) не вказуються. Це еквівалентно тому, що всім невизначеним значенням функцій збудження приписується значення 0.

Таблиця 7.4. Структурна таблиця переходів-виходів автомата Мура.

s_m	$K(s_m)$	s_k	$K(s_k)$	Y вихідний сигнал	X умова переходу	ФЗ
s_{11}	1100	s_0	0110	–	1	D_2D_3
s_0	0110	s_1	0011	Y_1,Y_2	1	D_3D_4
s_1	0011	s_2	1010	Y_3	1	D_1D_3
s_2	1010	s_3	0101	Y_4	1	D_2D_4
s_3	0101	s_4	0000	Y_5	X_1	–
s_6	0001				X_1	–
s_4	0000	s_5	1001	Y_6	X_2	D_1D_4
s_5	1001	s_6	0001	Y_7	1	D_4
s_4	0000				$\overline{X}_2$	D_4
s_6	0001	s_7	0010	Y_1	$\overline{X}_1$	D_3
s_3	0101				$\overline{X}_1$	D_3
s_7	0010	s_8	0100	Y_8	X_3	D_2
s_{10}	1000				X_3	D_2
s_8	0100	s_9	1110	Y_6	X_4	$D_1D_2D_3$
s_9	1110	s_{10}	1000	Y_7	1	D_1
s_8	0100				$\overline{X}_4$	D_1
s_{10}	1000	s_{11}	1100	Y_9	$\overline{X}_3$	D_1D_2
s_7	0010				$\overline{X}_3$	D_1D_2

Система рівнянь переходів

На наступному етапі необхідно отримати вирази для функцій збудження. Вони мають вигляд логічної суми доданків виду $s_i X$, де s_i – початковий стан, X – умова переходу. Маємо:

$$\begin{cases} D_1 = s_1 \vee s_4 X_2 \vee s_8 X_4 \vee s_9 \vee s_8 \overline{X}_4 \vee s_{10} \overline{X}_3 \vee s_7 \overline{X}_3 \\ D_2 = s_{11} \vee s_2 \vee s_7 X_3 \vee s_{10} X_3 \vee s_8 X_4 \vee s_{10} \overline{X}_3 \vee s_7 \overline{X}_3 \\ D_3 = s_{11} \vee s_0 \vee s_1 \vee s_6 \overline{X}_1 \vee s_3 \overline{X}_1 \vee s_8 X_4 \\ D_4 = s_0 \vee s_2 \vee s_4 X_2 \vee s_5 \vee s_4 \overline{X}_2 \end{cases}. \quad (7.1)$$

Для спрощення отриманих виразів виконуємо всі можливі операції склеювання та поглинання:

$$\begin{cases} D_1 = s_1 \vee s_4 X_2 \vee s_8 \vee s_9 \vee s_{10} \overline{X}_3 \vee s_7 \overline{X}_3 \\ D_2 = s_{11} \vee s_2 \vee s_7 \vee s_{10} \vee s_8 X_4 \\ D_3 = s_{11} \vee s_0 \vee s_1 \vee s_6 \overline{X}_1 \vee s_3 \overline{X}_1 \vee s_8 X_4 \\ D_4 = s_0 \vee s_2 \vee s_4 \vee s_5 \end{cases}. \quad (7.2)$$

Система рівнянь виходів

Складаємо систему рівнянь виходів (рів. 7.3). Ця система складається на основі таблиці 7.4.

$$\begin{cases} Y_1 = s_1 \vee s_7 \\ Y_2 = s_1 \\ Y_3 = s_2 \\ Y_4 = s_3 \\ Y_5 = s_4 \\ Y_6 = s_5 \vee s_9 \\ Y_7 = s_6 \vee s_{10} \\ Y_8 = s_8 \\ Y_9 = s_{11} \end{cases} \quad (7.3)$$

Побудова функціональної схеми автомата

Для побудови функціональної схеми автомата за отриманими виразами необхідно або замінити s_i його значенням через $Q_1Q_2Q_3Q_4$ або отримати сигнал, відповідний s_i. Зазвичай використовують другий спосіб і для отримання сигналу s_i застосовують так званий дешифратор станів, на вхід якого надходять сигнали з виходів елементів пам'яті $Q_1Q_2Q_3Q_4$. Для побудови керуючого автомата використаємо дешифратор 74145. В якості елементів пам'яті використані інтегральні мікросхеми 74HC74 – D-тригери з входами Set та Reset, для початкового встановлення станів тригерів. (7.2)

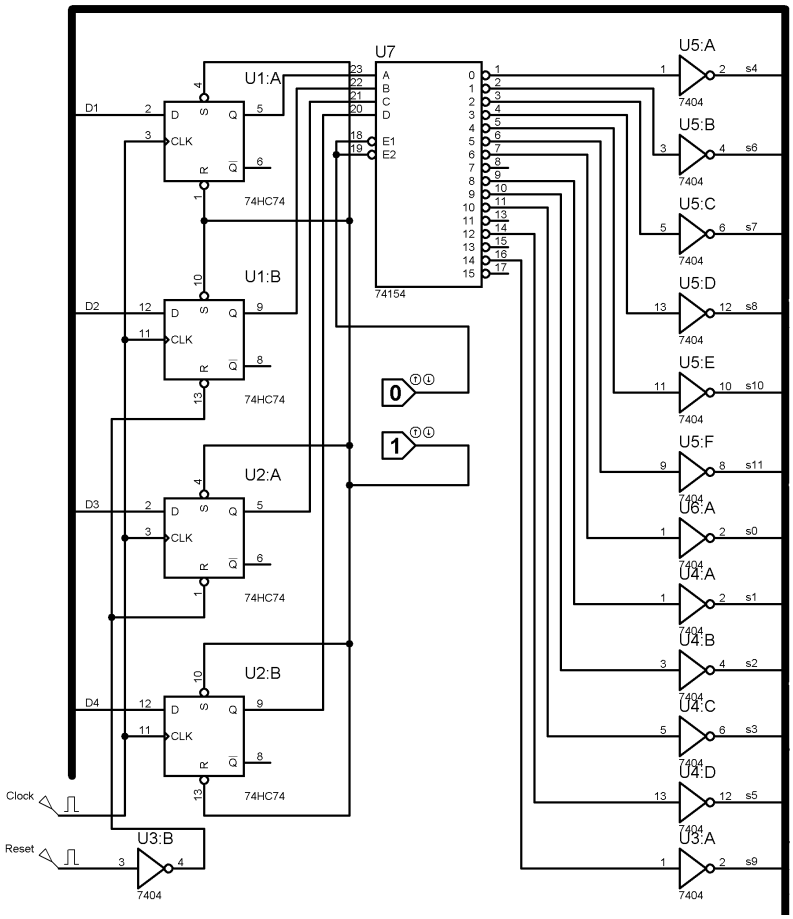

Рис. 7.7 Частина функціональної схема автомата Мура, що складається з комірок пам’яті (D тригери) та дешифратор (74145).

При подачі нуля на **Reset** автомат переходить в стан s_0.

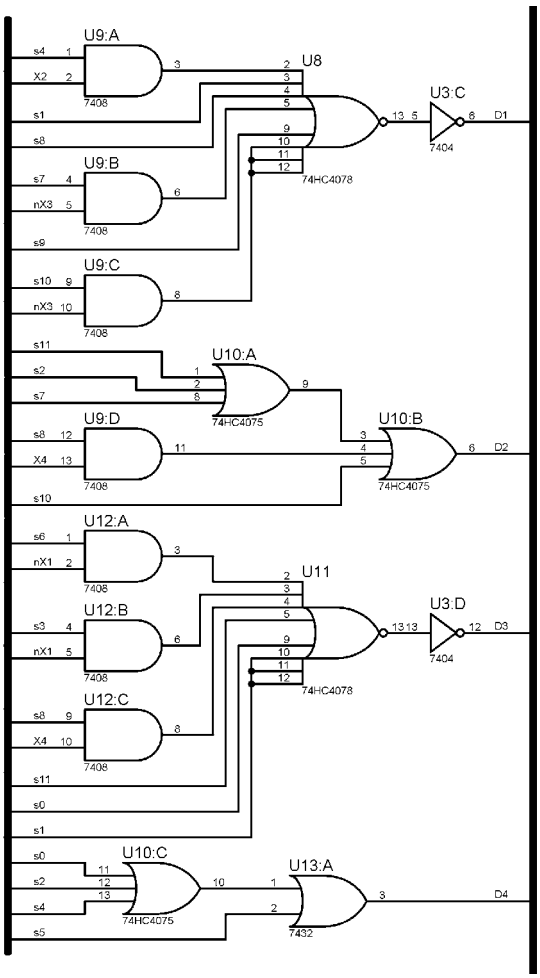

Рис. 7.8. Частини функціональної схема автомата Мура, що складається з комбінаційної схеми визначення функцій переходів.

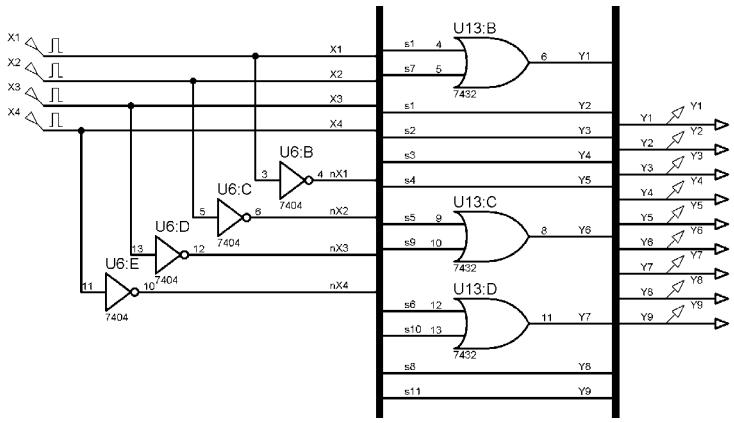

Рис. 7.9. Частини функціональної схема автомата Мура, що складається з комбінаційної схеми визначення функцій виходів.

На рисунках 7.7., 7.8. та 7.9. зображена схема автомата Мура. Шина, що використовується в данних рисунках спільна.

Лабораторне завдання

1. Згідно вашого варіанту, розробіть функціональну схему керуючого автомата, що:

Таблиця 7.5

h6	***h7***	***h8***	***h9***	***h10***	***Завдання***
0	0	0	0	0	обчислює суму трьох максимальних елементів масиву A(n)
0	0	0	0	1	знаходить максимальні парні елементи масивів A(n), B(v) та міняє їх місцями
0	0	0	1	0	знаходить максимальний парний елемент масиву A(n,n) ш замінює його мінімальним

0	0	0	1	1	знаходить максимальний парний елемент у кожнім рядку масиву A(n,n)
0	0	1	0	0	обчислює кількість парних елементів масивів A(n),B(m)
0	0	1	0	1	обчислює суму позитивних елементів масивів A(n),B(m)
0	0	1	1	0	обчислює суму позитивних діагональних елементів масиву A(n,n) та змінює їх за законом A(i,i)= A(n-i+1,n-i+1)*s (де s сума позитивних діагональних елементів масиву)
0	0	1	1	1	обчислює суму позитивних елементів непарних стовпців масиву A(n,m)
0	1	0	0	0	обчислює суму і кількість позитивних елементів у кожнім рядку масиву A(n,m)
0	1	0	0	1	обчислює суму парних позитивних елементів у кожнім непарному рядку масиву A(n,m)
0	1	0	1	0	обчислює суму парних позитивних елементів у масивах A(n,m), B(p)
0	1	0	1	1	виконує перестановку елементів першого й останнього стовпців A(n,n)
0	1	1	0	0	виконує перестановку максимальних елементів масивів A(n,n) і D(n,n)
0	1	1	0	1	виконує перетворення масиву A(n,n) у такий спосіб: до парних елементів масиву додасть 5, до непарних 3

0	1	1	1	0	обчислює суму трьох максимальних елементів масиву A(n,n,n)
0	1	1	1	1	виконує перетворення масивів A(n), B(v) у такий спосіб: четверті по номері негативні елементи масивів виконати операцію додавання між собою та збереження результату, а потім перші елементи масивів міняє місцями
1	0	0	0	0	виконує перетворення масиву A(n) у такий спосіб: поміняти місцями максимальний і мінімальний елементи цього масиву
1	0	0	0	1	виконує перетворення масивів A(n), B(n) у такий спосіб: поміняти місцями мінімальні елементи цих масивів
1	0	0	1	0	виконує перетворення масивів A(n), B(n) у такий спосіб: другі по номері парні елементи масивів виконати операцію додавання між собою, а потім замінити ним треті парні елементи
1	0	0	1	1	обчислює суму двох максимальних елементів масиву A(n,n)
1	0	1	0	0	виконує перетворення масиву A(n,n) у такий спосіб: виконати операцію додавання максимального елементи у

					кожному рядку і замінити значення першого елемента масиву на значення знайденої суми
1	0	1	0	1	обчислює суму трьох максимальних елементів масиву A(n,n)
1	0	1	1	0	виконує перетворення масиву A(n) у такий спосіб: до позитивних елементів масиву додасть значення максимального елемента цього масиву
1	0	1	1	1	виконує перетворення масиву A(n,n,n) у такий спосіб: до парних елементів масиву додасть 5, до непарних 3
1	1	0	0	0	знаходить максимальний та мінімальний елемент масиву A(n,n,n) та міняє їх місцями
1	1	0	0	1	обчислює кількість парних елементів масивів A(n,n),B(m)
1	1	0	1	0	знаходить максимальний парний елемент у кожнім рядку масиву A(n,n) та збільшує його на 2
1	1	0	1	1	виконує перестановку мінімальних елементів масивів A(n,n) і D(n,n)
1	1	1	0	0	виконує перетворення масивів A(n), B(v) у такий спосіб: до парних елементів масиву додасть 4, до непарних 2
1	1	1	0	1	виконує перетворення масиву A(n) у такий спосіб: до позитивних елементів масиву

					додасть суму парних елементів цього масиву
1	1	1	1	0	знаходить максимальні парні елементи масивів A(n,n), B(v) та міняє їх місцями
1	1	1	1	1	обчислює суму чотирьох максимальних елементів масиву A(n)

Мікропроцесорний автомат необхідно реалізувати у вигляді автомата Мура. Функціональну схему керуючих частин автомата синтезувати на елементах:

Таблиця 7.6.

h1	*h10*	Логічні елементи
0	0	І, АБО, НЕ
0	1	І-НЕ
1	0	АБО-НЕ
1	1	І, АБО- НЕ

Як елемент пам’яті, використайте:

Таблиця 7.7.

h5	*h4*	Тип тригера,
0	0	RS
0	1	T
1	0	JK
1	1	D

2. Промоделюйте отриману функціональну схему керуючого автомата у Proteus. Переконайтесь у правильності її роботи.

3. Виведіть відповідні графіки для вхідних та вихідних сигналів, що будуть в повній мірі описувати роботу розробленого Вами автомату.

Контрольні питання:

1. Дайте визначення цифрового автомату.
2. З яких етапів складається структурний синтез цифрового автомата Мура?
3. Чим відрізняється автомат Мілі від автомата Мура?
4. RS-тригер є автоматом Мура?
5. Навіщо використано дешифратор в схемі на рис. 7.7?

Лабораторна робота № 8

Синтез керуючого автомата Мура на базі регістра зсуву

***Мета роботи**: Провести структурний синтез керуючого автомата Мура на базі регістра зсуву.*

Керуючий автомат призначається для управління операційним автоматом та виконання на ньому певного мікроалгоритму.

Крім розглянутого раніше (В лабораторних роботах 6 та 7) канонічного методу, існують і інші методи синтезу керуючих автоматів, серед яких найбільш широко використовується синтез на базі регістра зсуву. Цей метод дозволяє при побудові схеми відмовитися від дешифратора, тому що стани кодуються унітарним кодом. В такому автоматі кількість елементів пам'яті вибирається рівною кількості внутрішніх станів. У кожен момент часу тільки один тригер знаходиться в стані 1, інші в стані 0. Зазвичай при синтезі на базі регістра зсуву використовуються D- тригери. Дуже ефективний цей метод для так званих лінійних мікропрограм, тобто мікропрограм без розгалужень (відсутні логічні умови).

Етапи проектування автомата Мура на базі регістра зсуву ті самі, що були в роботі 7.

Етапи проектування

1. Побудова змістовної схеми алгоритму.
2. Побудова блок-схеми закодованого мікроалгоритму.
3. Побудова граф–схеми переходів автомата Мура
4. Побудова таблиць переходів-виходів автомата Мура
5. Кодування станів автомата
6. Побудова структурної таблиці переходів-виходів автомата Мура.
7. Визначення системи рівнянь переходів
8. Визначення системи рівнянь виходів
9. Побудова функціональної схеми автомата

Приклад побудови мікропрограмного автомата у вигляді автомата Мура на базі регістра зсуву.

Візьмемо, для прикладу, задачу, що була розглянута в лабораторній роботі 7.

Задача

1. Побудувати керуючий автомат, який обчислює кількість парних елементів масивів A(n), B(m).
2. Мікропроцесорний автомат необхідно реалізувати у вигляді автомата Мура на базі регістра зсуву.
3. Оптимальну функціональну схему керуючих частин автомата синтезувати на елементах системи І, АБО, НЕ.
4. В якості пам’яті використайте D-тригери, доповнюючи її необхідними по алгоритму функціональними схемами.

Етапи проектування 1-4 повністю повторюють етапи проектування 1-4 з лабораторної роботи № 7.

Відмічена ГСА мікропрограми має вигляд рис. 7.5. лабораторної роботи № 7. Маємо сім станів . Отже число

тригерів m = 12. Виконаємо синтез з використанням D- тригерів. Закодуємо стану унітарним кодом : s_0 = 100000000000 , s_1 = 010000000000 , ... , s_{11} = 000000000001.

Будуємо структурну таблицю переходів-виходів автомата Мура (табл. 8.1). У даній таблиці у стовпцях К(s_m) і К(s_k) вказується код вихідного стану та стану переходу відповідно. У стовпці функцій збудження ФЗ вказується ті значення функцій збудження, які на даному переході обов'язково рівні 1. Решта (тобто рівні 0) не вказуються.

Таблиця. 8.1. Структурна таблиця переходів-виходів автомата Мура.

s_m	$K(s_m)$	s_k	$K(s_k)$	Y	X	ФЗ
s_{11}	000000000001	s_0	100000000000	–	1	D_1
s_0	100000000000	s_1	010000000000	Y_1,Y_2	1	D_2
s_1	010000000000	s_2	001000000000	Y_3	1	D_3
s_2	001000000000	s_3	000100000000	Y_4	1	D_4
s_3	000100000000	s_4	000010000000	Y_5	X_1	D_5
s_6	000000100000				$\overline{X}_1$	
s_4	000010000000	s_5	000001000000	Y_6	X_2	D_6
s_5	000001000000	s_6	000000100000	Y_7	1	D_7
s_4	000010000000				$\overline{X}_2$	
s_6	000000100000	s_7	000000010000	Y_1	$\overline{X}_1$	D_8
s_3	000100000000				$\overline{X}_1$	
s_7	000000010000	s_8	000000001000	Y_8	X_3	D_9
s_{10}	000000000010				X_3	

s_8	000000001000	s_9	000000000100	Y_6	X_4	D_{10}
s_9	000000000100	s_{10}	000000000010	Y_7	1	D_{11}
s_8	000000001000				$\overline{X}_4$	
s_{10}	000000000010	s_{11}	000000000001	Y_9	$\overline{X}_3$	D_{12}
s_7	000000010000				$\overline{X}_3$	

Система рівнянь переходів

Для отримання функцій збудження (рівняння 8.1) поступаємо таким чином. Вираз для кожної функції виходить у вигляді логічної суми додатків виду s_iX, де s_i - початковий стан, X-умова переходу. Маємо:

$$\begin{cases} D_1 = s_{11} \\ D_2 = s_0 \\ D_3 = s_1 \\ D_4 = s_2 \\ D_5 = s_3X_1 \vee s_6X_1 \\ D_6 = s_4X_2 \\ D_7 = s_5 \vee s_4\overline{X}_2 \\ D_8 = s_6\overline{X}_1 \vee s_3\overline{X}_1 \\ D_9 = s_7X_3 \vee s_{10}X_3 \\ D_{10} = s_8X_4 \\ D_{11} = s_9 \vee s_8\overline{X}_4 \\ D_{12} = s_{10}\overline{X}_3 \vee s_7\overline{X}_3 \end{cases} \qquad (8.1)$$

Для спрощення отриманих виразів виконуємо всі можливі операції склеювання і поглинання.

$$
\begin{cases}
D_1 = s_{11} \\
D_2 = s_0 \\
D_3 = s_1 \\
D_4 = s_2 \\
D_5 = X_1(s_3 \vee s_6) \\
D_6 = s_4 X_2 \\
D_7 = s_5 \vee s_4 \overline{X}_2 \\
D_8 = \overline{X}_1(s_6 \vee s_3) \\
D_9 = X_3(s_7 \vee s_{10}) \\
D_{10} = s_8 X_4 \\
D_{11} = s_9 \vee s_8 \overline{X}_4 \\
D_{12} = \overline{X}_3(s_{10} \vee s_7)
\end{cases}
\qquad (8.2)
$$

Система рівнянь виходів

Складаємо систему рівнянь виходів (8.3). Ця система складається на основі таблиці 8.1., де X – вхідні, Y – вихідні сигнали. Для першого рівняння системи справедливим є твердження, що ми отримуємо вихідний стан $\boldsymbol{Y_1}$ при переході в стан $\boldsymbol{s_1}$, або $\boldsymbol{s_7}$ і т.д.

$$\begin{cases} Y_1 = s_1 \vee s_7 \\ Y_2 = s_1 \\ Y_3 = s_2 \\ Y_4 = s_3 \\ Y_5 = s_4 \\ Y_6 = s_5 \vee s_9 \\ Y_7 = s_6 \vee s_{10} \\ Y_8 = s_8 \\ Y_9 = s_{11} \end{cases} \quad (8.3)$$

Побудова функціональної схеми автомата

Так як стани автомата закодовані унітарним кодом, то існує можливість ототожнити кожний стан з виходом відповідного тригера, тобто прийняти $s_i = Q_i$. Для прийнятого способу кодування перехід з одного стану в інший супроводжується зрушенням коду, записаного в дванадцяти-розрядному регістрі. Цим і пояснюється назва методу. Функціональна схема автомата Мура, побудована за отриманими рівняннями, наведена на рисунках 8.1., 8.2. та 8.3.. В результаті набуття певних навичках синтез автомата Мура на базі регістра зсуву виконується безпосередньо по зазначеній ГСА без побудови структурної таблиці переходів - виходів.

При подачі одиниці на ***Reset*** автомат переходить в стан s0.

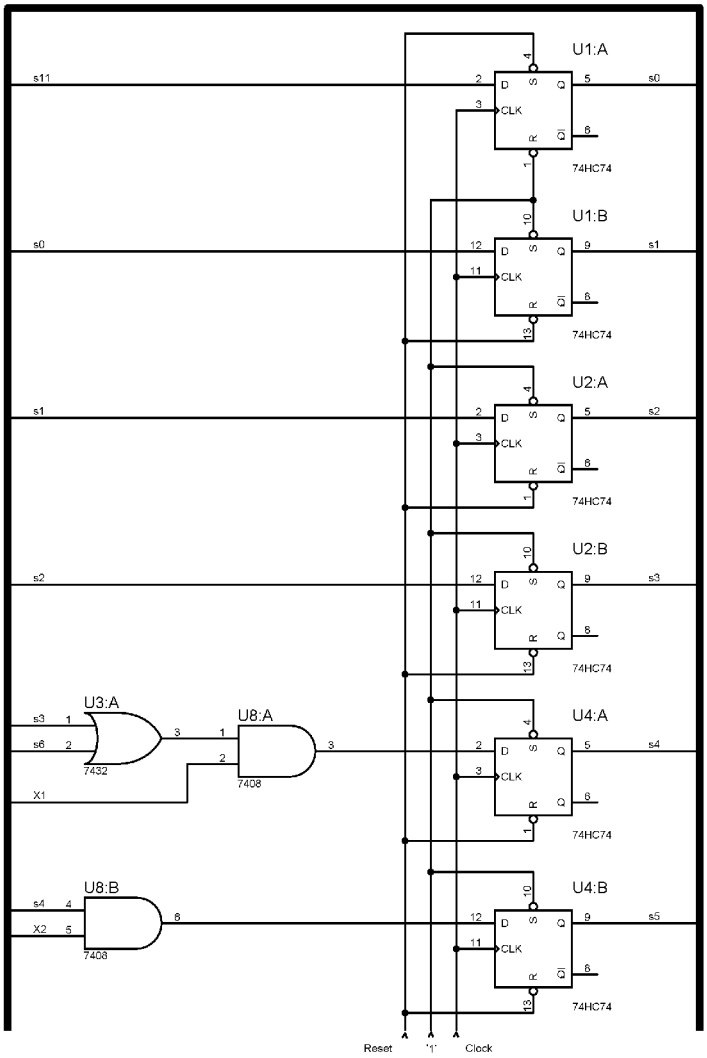

Рис. 8.1 Частина функціональної схема автомата Мура на регістрах зсуву, що складається з комірок пам’яті (D тригери) та з комбінаційної схеми визначення функцій переходів.

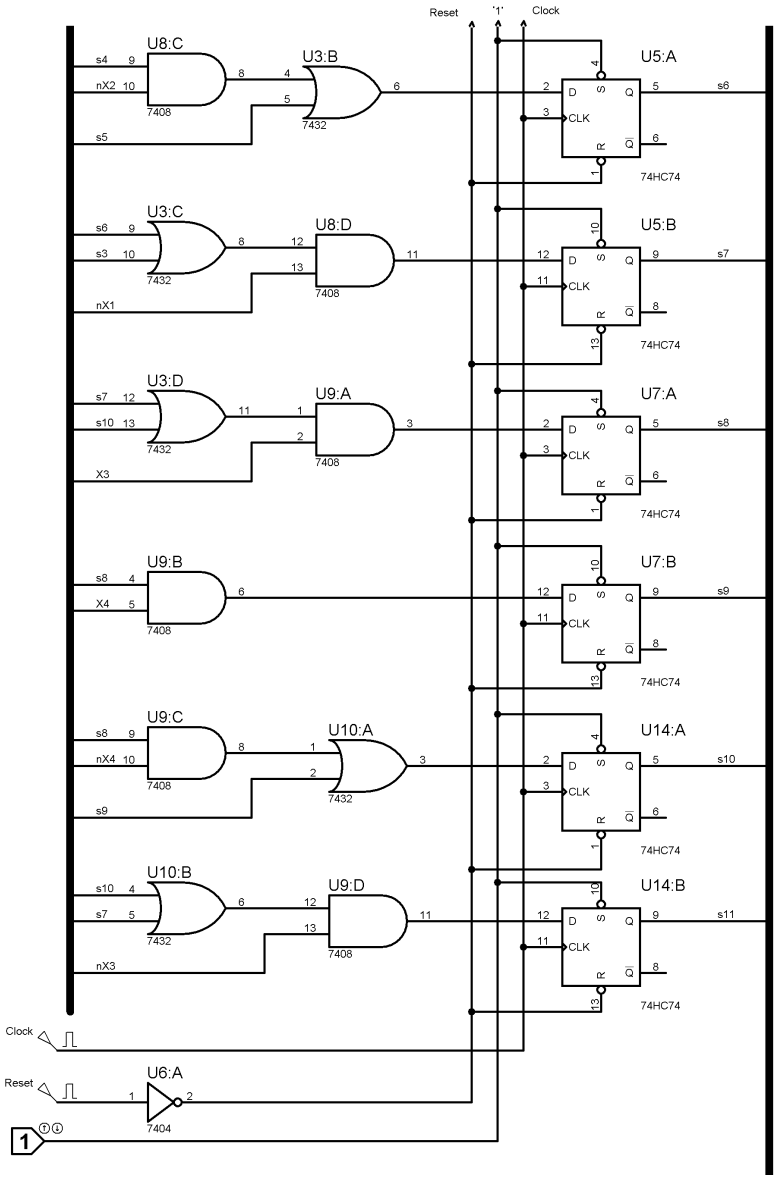

Рис. 8.2. Частина функціональної схема автомата Мура на регістрах зсуву. (продовження рис. 8.1.)

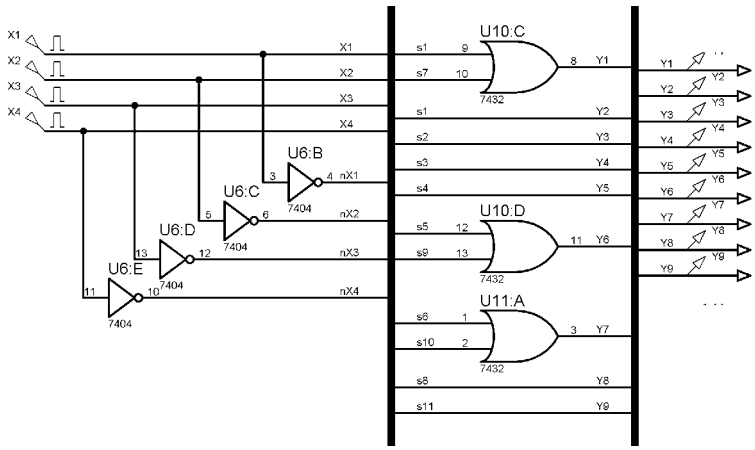

Рис. 8.3. Частини функціональної схема автомата Мура, що складається з комбінаційної схеми визначення функцій виходів.

На рисунках 8.1., 8.2. та 8.3. зображена схема автомата Мура на регістрах зсуву. Шина, що використовується в данних рисунках спільна.

Лабораторне завдання

1. Згідно Вашого варіанту, розробіть функціональну схему керуючого автомата
2. В якості завдання візьміть завдання дане в лабораторній роботи №7 (табл. 7.5).
3. Мікропроцесорний автомат необхідно реалізувати у вигляді автомата Мура на базі регістра зсуву. Як елемент пам'яті використовуйте D-тригери. Функціональну схему керуючих частин синтезувати на елементах:

Таблиця 8.2.

h3	*h4*	Логічні елементи
0	0	І, АБО, НЕ
0	1	І-НЕ
1	0	АБО-НЕ
1	1	І, АБО- НЕ

4. Промоделюйте отриману схему у Proteus та виведіть відповідні графіки для вхідних та вихідних сигналів, що будуть в повній мірі описувати роботу розробленого вами автомату.

Контрольні питання:

1. Коли доцільно використовувати в якості керуючого автомата автомат Мура на базі регістра зсуву?
2. Чи можна вважати регістр зсуву автоматом Мура на базі регістра зсуву?
3. Чи можна в якості пам'яті автомата Мура на базі регістра зсуву використати Т-тригери?

Лабораторна робота № 9

Принципи захисту інформації від втрат. Коди Хемінга

***Мета роботи**: Ознайомлення з основами захисту інформації від втрат. Провести кодування, декодування та корекцію помилок за допомогою кодів Хемніга.*

Вступ.

Код Хемінга являє собою блочний код, який дозволяє виявити та виправити помилково переданий біт в межах переданого блоку.

Цей алгоритм дозволяє закодувати певне інформаційне повідомлення певним чином і після передачі (наприклад по мережі) визначити чи з'явилася якась помилка в цьому

повідомленні (наприклад через перешкоди) і, при можливості, відновити це повідомлення.

В цій роботі буде розглянуто найпростіший алгоритм Хемінга, який може виправити лише поодиноку помилку в блоці. Також варто зазначити, що існують більш вдосконалені модифікації даного алгоритму, які дозволяють виявляти (і якщо можливо виправляти) більшу кількість помилок.

Відразу варто зазначити, що лабораторна робота з Кодами Хемінга складається з двох частин. Перша частина полягає в кодуванні вихідного повідомлення, вставляючи в нього в визначених місцях контрольні біти (обчислені певним чином). Друга частина роботи полягає в отриманні вхідного повідомлення із блоку закодованого кодом Хемінга (з виправленням можливої поодинокої помилки в блоці). Перевірка на помилку відбувається шляхом перевірки контрольних бітів. Якщо всі обчислені контрольні біти збігаються з визначеними, то повідомлення отримано без помилок. В іншому випадку, виводиться повідомлення про помилку і при можливості помилка виправляється.

Розглянемо принцип роботи алгоритму на прикладі:

Нехай у нас є 16-ти бітне слово:

0100 0100 0011 1101

Задача передати його, від відправника до отримувача. Під час передачі пакету передбачена втрата до 1 біту інформації.

Для забезпечення цільності пакету додаємо контрольні біти на позиції, які відповідають степеням двійки. Їх кількість визначається за принципом: $m = n + k < 2^k - 1$, де n – кількість інформаційних бітів, k – контрольних, m – загальна кількість бітів в пакеті.

Отже, для 16-ти бітного числа буде додано 5 контрольних біт на позиціях 1, 2, 4, 8, 16:

Наш паке набуває вигляду:

**0* 100* 0100 001* 1110 1, де * - місця контрольних бітів.

Після перетворення, та внесення контрольних бітів наш пакт стане 21-бітним.

Обчислення контрольних бітів

Визначення значень контрольних бітів відбувається за наступними правилами:

1. Нумеруємо біти нашого пакету від 1 до 21 в числами в двійковій системі числення.
2. Розбиваємо наші біти на групи, за законом: одиниці в двійковому номері біта вказують, до яких груп належить даний біт. (на приклад п’ятий біт має номер 00101, молодша одиниця вказує, що він належить до молодшої (першої) групи, а 3 одиниця показує, що даний біт належить також і до третьої групи). Див. таблицю 9.1.
3. Якщо, в номері біта одна одиниця, то цей біт є контрольним для групи в якій даний біт також є одиницею (на приклад перший біт має номер 00001, молодша одиниця вказує, що він належить до молодшої (першої) групи, а в силу того що одиниця одна то він є контрольним бітом першої групи). Див. таблицю 9.1.
4. Визначаємо значення контрольних бітів. Значення визначається такип чином, щоб кількість одиниць в кожній групі була парною.

В таблиці 9.1. символом “X” позначені біти, що належать відповідним контрольних групам. Контрольні біти груп відмічені написом (кб) в першому стовпці таблиці.

Таблиця. 9.1. Визначення контрольних груп.

Номер біта в пакеті	Номер в двійковій системі	Біти 1 групи (1)	Біти 2 групи (2)	Біти 3 групи (4)	Біти 4 групи (8)	Біти 5 групи (16)
1 - (кб)	00001	X				
2 - (кб)	00010		X			
3	00011	X	X			
4 - (кб)	00100			X		
5	00101	X		X		
6	00110		X	X		
7	00111	X	X	X		
8 - (кб)	01000				X	
9	01001	X			X	
10	01010		X		X	
11	01011	X	X		X	
12	01100			X	X	
13	01101	X		X	X	
14	01110		X	X	X	
15	01111	X	X	X	X	
16 - (кб)	10000					X
17	10001	X				X
18	10010		X			X
19	10011	X	X			X
20	10100			X		X
21	10101	X		X		X

Далі підраховується кількість одиниць, серед відповідних інформаційних бітів. Якщо їх кількість парна, то контрольний біт встановлюється в 0, інакше – 1.

Таким чином отримаємо наступне число:

100**1** 100**0** 0100 001**0** 1110 1

Декодування та виправлення помилок

Порядок контрольних біт, та місце інформаційних бітів в пакеті дозволяють легко визначити та виправити поодиноку помилку. Наприклад, ми отримали закодоване повідомлення, яке прийшло з помилкою. Припустимо, що 11-й біт був переданий неправильно:

100**1** 100**0** 01**1**0 001**0** 1110 1

Для визначення помилки потрібно підрахувати значення контрольних біт, для отриманого повідомлення:

010**1** 100**1** 01**1**0 001**0** 1110 1

Порівнявши значення отриманих контрольних біт з обрахованими, видно, що біти 1, 2 та 8 не співпадають. Підрахувавши суму номерів некоректних біт:

$$1 + 2 + 8 = 11$$

отримуємо позицію помилкового біту та інвертуємо його

100**1** 100**0** 01**0**0 001**0** 1110 1

Під час виконання роботи слід розробити 2 схеми. Перша отримує вхідний пакет бітів, доповнює їх контрольними бітами та утворює пакет для передачі. На рис. 9.1. дана схема позначена як “Кодер”. Друга схема має отримати пакет даних, виправити помилку в пакеті (якщо вона існує) та видає інформаційні біти. На рис. 9.1. друга схема позначена як “Декодер”.

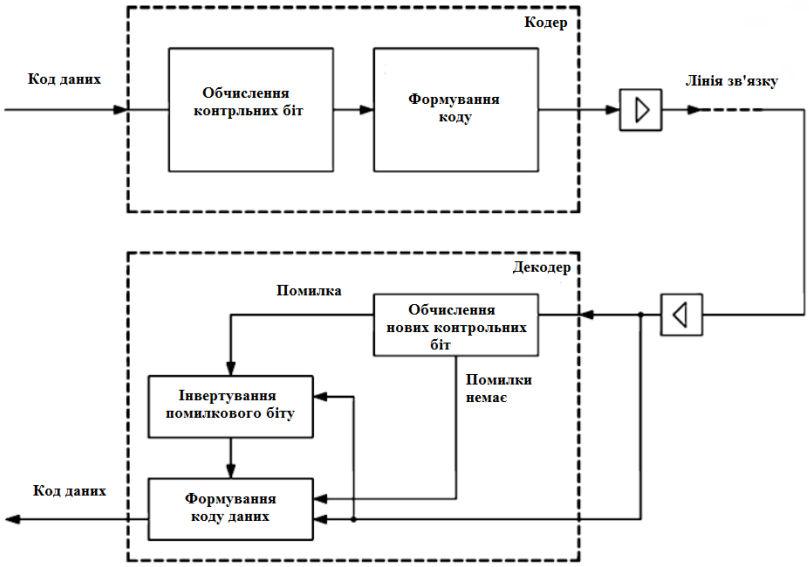

Рис. 9.1 Принципова схема роботи

Для виявлення подвійних помилок можна модифікувати код Хемінга. Дана модифікація дасть можливість як і раніше визначати та виправляти поодиноку помилку, та лише виявляти подвійну помилку.

В модифікованому коді Хемінга в пакет додається ще один додатковий біт (m+1). Даний біт визначає, чи була парна кількість одиниць в попередніх m бітах пакету. Декодування нового пакету набуває наступного вигляду:

1. Контрольні біти груп співпадають, останній контрольний біт не змінився – помилок в пакеті немає.
2. Контрольні біти груп не співпадають, останній контрольний біт змінився – поодинока помилка, що може бути виправлена. Біти контрольних груп, з помилками визначають адресу помилки.
3. Контрольні біти груп не співпадають, останній контрольний біт не змінився – синдром подвійної помилки (можливий варіант парної кількості помилок).

Дані помилки виправити неможливо, пакет використовувати не можна.

4. Контрольні біти груп співпадають, останній контрольний біт змінився – груповий синдром непарної кратності (можливий помилка лише в останньому контрольному біті). Пакет використовувати не можна.

Лабораторне завдання

1. Згідно Вашого варіанту, розробіть функціональні схеми “Кодера” та “Декодера”.

Довжина слова:

Таблиця. 9.2.

H7	H8	H9	H10	Кількість інформаційних бітів
0	0	0	0	14
0	0	0	1	13
0	0	1	0	17
0	0	1	1	11
0	1	0	0	19
0	1	0	1	16
0	1	1	0	12
0	1	1	1	22
1	0	0	0	18
1	0	0	1	23
1	0	1	0	15
1	0	1	1	25
1	1	0	0	10
1	1	0	1	27
1	1	1	0	20
1	1	1	1	29

Функціональну схему синтезувати на елементах:

Таблиця. 9.3.

H5	H4	Логічні елементи
0	0	І, АБО, НЕ, Виключне АБО
0	1	І-НЕ, Виключне АБО
1	0	АБО-НЕ, Виключне АБО
1	1	І, АБО-НЕ, Виключне АБО

2. Введіть у Proteus функціональні схеми.
3. Проведіть моделювання отриманих Вами схем. Занесіть помилку у вхідний пакет “Декодера”. Переконайтесь в коректності роботи Вашої схеми.
4. Занесіть додатковий біт парності для пакету інформації, що передається (для виявлення подвійної помилки).
5. Модернізуйте Ваші схеми для створення та обробки даних пакетів.
6. Проведіть моделювання отриманих Вами схем. Занесіть поодинокі та подвійні помилки в вхідний пакет “Декодера”. Проаналізуйте отримані Вами результати роботи схем.

Контрольні питання:

1. Які Ви знаєте коди, що дозволяють виявити та виправити помилки?
2. В чому полягають принципи алгоритмів виправлення помилок.
3. Яку мінімальну кількість контролюючих бітів треба внести в вхідне слово, щоб виявити та виправити поодиноку помилку.