

Міністерство освіти і науки України  
Національний авіаційний університет  
Навчально-науковий інститут комп'ютерних інформаційних технологій  
Кафедра комп'ютеризованих систем управління

Лабораторна робота №2  
з дисципліни «Архітектура комп'ютерів»  
на тему «Синтез керуючих автоматів з програмованою логікою»  
Варіант №4

Виконав:  
студент ННІКІТ СП-225  
Клокун В. Д.  
Перевірив:  
Зіньков Ю. Г.

Київ 2018

## 1 Мета роботи

Закріплення теоретичних знань з синтезу керуючих автоматів із програмованою логікою.

## 2 Хід роботи

Виконаємо кодування мікрооперацій. Для цього використаємо метод прямого включення. Спочатку з'ясуємо, які мікрооперації сумісні, а які ні. Для цього побудуємо матрицю сумісності  $S$  за таким принципом:

$$S = \begin{bmatrix} 0 & S_{12} & \dots & S_{1M} \\ S_{21} & 0 & \dots & S_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ S_{M1} & S_{12} & \dots & 0 \end{bmatrix}, \quad S_{ij} = \begin{cases} 1, & \text{якщо } y_i \text{ і } y_j \text{ сумісні,} \\ 0, & \text{якщо } y_i \text{ і } y_j \text{ несумісні.} \end{cases}$$

В результаті отримали булеву симетричну матрицю сумісності  $S$ :

$$S = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{matrix}.$$

Тепер переходимо до методу прямого включення. Його суть полягає в тому, що процес розподілу мікрооперацій  $Y = (y_1, \dots, y_M)$  по полях  $Y_1, Y_2, \dots, Y_k$  мікрокоманди розділяється на  $M$  кроків. На кожному кроці чергової мікрооперації  $y_i$  відшукується поле  $Y_p$ , причому мікрооперація  $y_i$  повинна бути несумісною з жодною з мікрооперацій цього поля. Якщо серед полів  $Y_1, Y_2, \dots, Y_t$  такого поля не існує, то для цієї мікрооперації вводиться нове поле  $Y_{t+1}$ .

Стан процесу включення на кожному кроці характеризується матрицею включення  $R$ , яка будується за таким принципом:

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1M} \\ \vdots & \vdots & \vdots & \vdots \\ r_{k1} & r_{k2} & \dots & r_{kM} \end{bmatrix}, \quad r_{ij} = \begin{cases} 1, & \text{при } y_i \in Y_p, \\ 0, & \text{при } y_i \notin Y_p. \end{cases}$$

Таким чином, умова включення мікрооперації  $y_i$  в поле  $Y_p$  формулюється так: мікрооперація  $y_i$  включається в поле  $Y_p$ , якщо  $i$ -й рядок  $S_i$  матриці  $S$  не перетинається з  $p$ -м рядком  $R_p$  матриці  $R$ , тобто  $S_i \cap R_p = \emptyset$ . Будуємо матрицю включення  $R$ :

$$R = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} & \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix}.$$

Бачимо, що кожна мікрооперація ввійшла до одного з полів один і тільки один раз, тобто розподіл виконано вірно. Отже, отримали такі підмножини:

$$\begin{aligned} Y_1 &= \{y_1, y_3, y_4, y_6, y_8, y_9, y_{10}, y_{11}, y_{13}\}, \\ Y_2 &= \{y_2, y_5, y_7, y_{12}, y_{14}, y_{15}, y_{16}\}, \\ Y_3 &= \{y_{17}\}. \end{aligned}$$

Визначимо довжину операційної частини мікрокоманди. Для цього обчислимо  $n = \lceil \log_2(M_1 + 1) \rceil + \lceil \log_2(M_2 + 1) \rceil + \lceil \log_2(M_3 + 1) \rceil = 4 + 3 + 1 = 8$ . Закодуємо мікрооперації в підмножинах (табл. 1).

$Y_1$		$Y_2$		$Y_3$	
$y_i$	$K(Y_1)$	$y_i$	$K(Y_2)$	$y_i$	$K(Y_3)$
–	0000	–	000	–	0
$y_1$	0001	$y_2$	001	$y_{17}$	1
$y_3$	0010	$y_5$	010		
$y_4$	0011	$y_7$	011		
$y_6$	0100	$y_{12}$	100		
$y_8$	0101	$y_{14}$	101		
$y_9$	0110	$y_{15}$	110		
$y_{10}$	0111	$y_{16}$	111		
$y_{11}$	1000				
$y_{13}$	1001				

Табл. 1: Кодування мікрооперацій за множинами

Визначимо довжину поля  $B$  — адреси переходу на віддалену мікрокоманду для керуючих мікрокоманд. Оскільки загальна кількість мікрокоманд  $P = 23$ , то  $n_b = \log_2 23 = 5$ .

Визначимо довжину сигналів логічних умов та закодуємо їх. Нехай  $N$  — число оригінальних сигналів логічних умов, тоді при  $N = 8$ , довжина сигналу логічних умов  $n_x = \lceil \log_2(N + 1) \rceil = 4$ .

$x_i$	$X$
—	0000
$x_1$	0001
$x_2$	0010
$x_3$	0011
$x_4$	0100
$x_5$	0101
$x_6$	0110
$x_7$	0111
$x_8$	1000

Табл. 2: Кодування сигналів логічних умов

Сформуємо розрядну структуру керуючих та операційних мікрокоманд. Оскільки розрядна сітка має бути однаковою як для керуючих, так і для операційних мікрокоманд, то щоб вирівняти керуючі мікрокоманди з операційними, до перших вводиться один додатковий біт. Також, керуючі і операційні команди мають біт  $S$ . Для керуючих мікрокоманд  $S = 1$ , а для операційних  $S = 0$ . Біт  $U$  в сітці операційних мікрокоманд вводиться для завершення операцій.

Поле	Кількість розрядів	Поле	Кількість розрядів
$S$	1	$S$	1
$Y_1$	4	$X$	4
$Y_2$	3	$B$	5
$Y_3$	1		
$U$	1		

а) б)

Табл. 3: Структура мікрокоманд: а — операційних, б — керуючих

Складаємо та оформлюємо текст мікропрограми (табл. 4) на основі даних, отриманих у попередніх кроках. За допомогою програмного продукту «Мікрокод» перевіряємо отримані дані, а саме матрицю включення, її контрольну суму, тобто той факт, що кожна мікрооперація міститься лише в одному з полів, та результат кодування мікрооперацій за множинами. Аналізуємо результат (рис. 1). Бачимо, що результат програмної обробки вхідних даних повністю збігається з результатами, які були отримані вручну.

№	Адреса мікрокоманди	S	Мікрокоманда
1	00001	1	1000.00.00001
2	00010	0	0001.001.1.0
3	00011	1	0001.00.00101
4	00100	0	0010.100.1.0
5	00101	0	0011.010.0.0
6	00110	1	0010.00.00011
7	00111	1	0011.00.10001
8	01000	0	0100.011.0.0
9	01001	1	0111.00.10110
10	01010	0	0000.001.0.0
11	01011	1	0100.00.01101
12	01100	0	0111.111.0.0
13	01101	1	0101.00.10100
14	01110	0	1001.001.0.0
15	01111	0	1000.101.0.0
16	10000	1	0000.00.00111
17	10001	1	0110.00.01101
18	10010	0	0101.011.0.0
19	10011	1	0000.00.01101
20	10100	0	0010.001.0.0
21	10101	0	0100.011.0.1
22	10110	0	0110.110.0.0
23	10111	1	0000.00.01101

Табл. 4: Мікропрограма

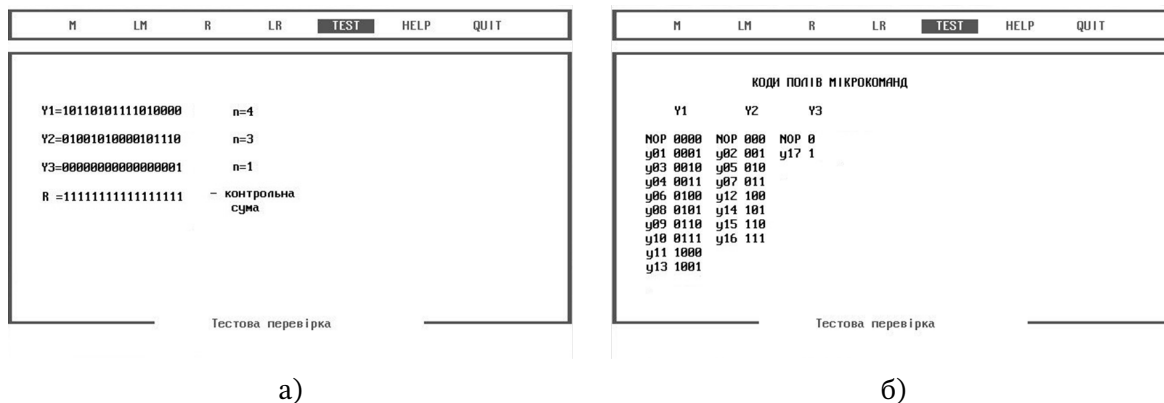


Рис. 1: Результат обробки початкових даних програмою «Мікрокод»: а — матриця включення та контрольна сума, б — результат кодування мікрооперацій за множинами

### **3 Висновок**

Під час виконання даної лабораторної роботи ми закріпили теоретичні знання з синтезу керуючих автоматів з програмованою логікою, навчились синтезувати мікрокоманди, будувати задовану мікропрограму та розробляти структурну схему керуючого автомата з програмованою логікою.