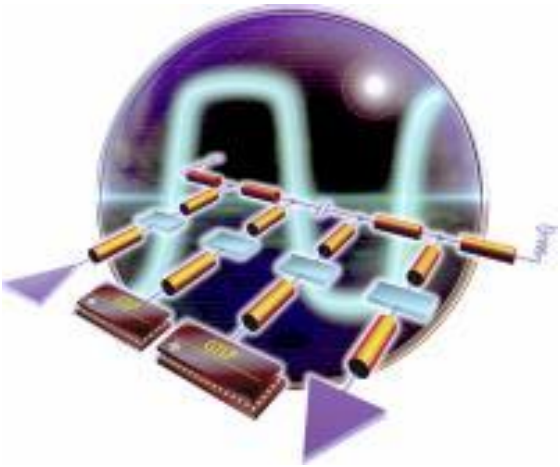
В. А. Лахно, Г. А. Могильний

„ЛАБОРАТОРНИЙ ПРАКТИКУМ З ПРИКЛАДНОЇ ТЕОРІЇ ЦИФРОВИХ АВТОМАТІВ”



Міністерство освіти і науки, молоді та спорту України
ДЗ «Луганський національний університет
імені Тараса Шевченка»

В. А. Лахно, Г. А. Могильний

„ЛАБОРАТОРНИЙ ПРАКТИКУМ З ПРИКЛАДНОЇ ТЕОРІЇ ЦИФРОВИХ АВТОМАТІВ”

для студентів напряму 0915 „Комп’ютерна інженерія”
спеціальності 7.091501
„Комп’ютерні системи та мережі”

Луганськ
ДЗ „ЛНУ імені Тараса Шевченка”
2010

УДК 681.3.07
ББК 32.973.26-018
Л29

Рецензенти:

Меняйленко О. С. – доктор технічних наук, професор Луганського національного університету імені Тараса Шевченка.

Леві Л. І. – доктор технічних наук Луганського національного аграрного університету.

Сквірський В. Д. – кандидат технічних наук, доцент Луганського національного університету імені Тараса Шевченка.

Лахно В. А.

Л29 „Лабораторний практикум з Прикладної теорії цифрових автоматів" : для студ. спец. 7.091501 „Комп'ютерні системи та мережі" / В. А. Лахно, Г. А. Могильний ; Держ. закл. „Луган. нац. ун-т імені Тараса Шевченка". – Луганськ : Вид-во ЛНУ імені Тараса Шевченка, 2010. – 138 с.

Навчальний посібник – лабораторний практикум з курсу „Прикладна теорія цифрових автоматів" містить базові теоретичні матеріали, а також лабораторні роботи, які виконуються у різних середовищах автоматизованого проектування цифрових автоматів, варіанти завдань та зразки їх виконання. У навчальному посібнику наведено сучасні методи аналізу й синтезу логічних електронних схем та цифрових автоматів із застосуванням найбільш поширених CASE-засобів – EWB, Multisim, MC8.

Призначено для студентів всіх форм навчання спеціальності 7.091501 „Комп'ютерні системи та мережі".

УДК 681.3.07
ББК 32.973.26-018

Рекомендовано до друку навчально-методичною радою
Луганського національного університету імені Тараса Шевченка
(протокол № 4 від 1 грудня 2010 року)

ЗМІСТ

ВСТУП

Зважаючи на важливість і обширність матеріалу, пов'язаного з питаннями методів аналізу й синтезу логічних електронних схем та цифрових автоматів (ЦА) із застосуванням CASE-засобів, його вивчення в сучасній методиці викладання дисципліни „Прикладна теорія цифрових автоматів" займає провідну позицію.

Матеріал, пов'язаний з вивченням CASE-засобів, які використовуються для проектування цифрових автоматів, складає значну частину курсу „Прикладна теорія цифрових автоматів". Дисципліна „Прикладна теорія цифрових автоматів" є однією з базових у системі знань та вмінь, що формують бакалавра та спеціаліста зі спеціальності „Комп'ютерні системи та мережі".

У цьому навчальному посібнику, відповідно до навчального плану, містяться п'ятнадцять лабораторних робот, які в певній мірі дозволяють студентам оволодіти методами аналізу й синтезу цифрових автоматів за допомогою найбільш поширених CASE-засобів.

Мета лабораторного практикуму – сприяти вирішенню завдань підготовки спеціалістів у галузі комп'ютерних технологій, допомогти студентам оволодіти практичними навичками роботи з проектування логічних схем та автоматів із елементами пам'яті, зокрема автоматами Мілі та Мура.

Спробою досягти цієї мети і визначається структура цих методичних вказівок. Частина лабораторних робіт виконується відповідно до індивідуальних варіантів, а частина робіт – студентом відповідно до завдання, яке міститься у розділі – хід роботи.

Після виконання лабораторної роботи кожен студент повинен оформити звіт на аркуші формату А4, який містить такі пункти:

– мета роботи;

– варіант завдання;

– логічна схема цифрового автомата, побудована у відповідному середовищі – EWB, MC8, Multisim 9 або Multisim 10;

– таблиця із результатами експериментів;

– висновок щодо роботи.

Під час захисту роботи студент відповідає на питання, які наведено після кожної лабораторної роботи (питання для самоперевірки).

Критерії оцінювання робіт. Оцінюється відсоток виконаних лабораторних робот:

A –	“5”	— 4,5 – 5,0	— 91% – 100%
B –	“4”	— 4,0 – 4,4	— 86% – 90%
C –	“4”	— 3,5 – 3,9	— 81% – 85%
D –	“3”	— 3,0 – 3,4	— 76% – 80%
E –	“3”	— 2,5 – 2,9	— 71% – 75%
FX –	“2”	— 2,0 – 2,4	— 66% – 70%
F –	“2”	— 1,5 – 1,9	— 60% – 65%

Виклад навчального матеріалу супроводжено численними прикладами реального застосування прикладної теорії цифрових автоматів для розв’язання конкретних задач, пов’язаних з методами аналізу й синтезу логічних електронних схем, принципів розробки програмних алгоритмів у цифрових автоматах.

Модуль 1 Інформаційні основи цифрових автоматів. Представлення інформації в цифровому автоматі

Лабораторна робота № 1. Арифметичні та логічні основи цифрових автоматів

Мета роботи: вивчення систем числення, принципів організації арифметичних дій з двійковими і двійково-десятковими числами.

Загальні теоретичні відомості

Під системою числення прийнято розуміти спосіб запису чисел цифровими знаками (цифрами). Числа можна зображувати або комбінацією символів якого-небудь алфавіту, або стійкими станами рівноваги одного чи кількох пристроїв. Основою системи числення звичайно є спосіб зображення цілих чисел. Розрізняють позиційні й непозиційні системи числення.

Система числення називається позиційною, якщо значення кожної цифри, що входить у запис числа, залежить від її положення (позиції) у ряді цифр, що зображують це число.

Непозиційною називається така система числення, у якій значення цифри не залежить від її положення в ряді цифр, що зображують число.

У цифрових обчислювальних машинах використовуються тільки позиційні системи числення, тому що непозиційні системи незручні для обчислень.

У цей час загальноприйнята десяткова система числення. Названа вона так тому, що для запису чисел у ній використовуються десять різних цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 і 9, що зображують цілі числа від нуля до дев'яти.

Крім десяткової, існують й інші позиційні системи числення. Беручи за основу системи різні числа: два, три, п'ять, вісім й інші, можна одержати відповідно двійкову, трійкову, восьмеричну й інші системи. При цьому в кожній з них, як і в десятковій системі, число

буде являти собою суму ступенів основи системи з відповідними коефіцієнтами:

$$A_q = a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + ... + a_k q^k + a_0 q^0 + a_{-1}q^{-1} + ... + a_{-m}q^{-m}, \quad (1.1)$$

де A_q – число в *q-й* системі числення; q – основа системи; n – номер розряду.

У восьмеричній системі для запису чисел використовуються вісім цифр (0, 1, 2, 3, 4, 5, 6, 7), що позначають цілі числа від нуля до семи. Число вісім (основа системи) записується двома цифрами у вигляді 10.

*Приклад. З*апишемо у восьмеричній системі число $A=266_{10}$. Відповідно до виразу (1) ці числа необхідно розкласти за ступенями вісімок, тому що основа системи - вісім.

$$\frac{266}{8} = 33 + залишок 2 \quad (МЗБ)$$

$$\frac{33}{8} = 4 + залишок \quad 1 \quad (МЗБ)$$

$$\frac{4}{8} = 0 + залишок \quad (СЗБ)$$

$$266_{10} = \quad 4 \quad 1 \quad 2_8$$

*Приклад. З*апишемо восьмеричне число $A=372_8$ *у д*есятковій системі числення.

$$372_8 = 3\cdot 8^2 + 7\cdot 8^1 + 2\cdot 8^0 = 3\cdot 64 + 7\cdot 8 + 3\cdot 1 = 250_{10}.$$

Зараз у техніці найбільш поширена двійкова система числення у зв'язку з порівняно легким створенням пристрою з двома стійкими станами рівноваги. Двійкова система числення має тільки дві цифри: 0 і 1.

Відповідно до виразу (1) двійкове число *N2* являє собою суму:

$$A2 = xп \bullet 2n + xп - 1 \bullet 2\,n - 1 + ... + x1 \bullet 21 + x0 \bullet 20 + \ldots$$

Тут коефіцієнти x_i можуть приймати тільки два значення: 0 і 1.

*Приклад. З*апишемо число 69_{10} у двійковій системі:

$$6910 = 1 \bullet 26 + 0 \bullet 25 + 0 \bullet 24 + 0 \bullet 23 + 1 \bullet 22 + 0 \bullet 21 + 1 \bullet 2°.$$
$$1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1$$

Таким чином, отримаємо $69_{10}=1000101_2$.

Порівняно з іншими позиційними системами числення двійкова система примітна тим, що в ній арифметичні дії найбільш прості.

У двійковій системі арифметичні операції особливо прості. У двійковій системі не існує „таблиці складання", яку потрібно б було запам'ятовувати, оскільки перенесення в старший розряд розпочинається з 1+1=10. При складанні великих чисел необхідно лише складати за стовпцями або розрядами, як у десятковій системі, пам'ятаючи лише про те, що як тільки сума у стовпці досягне числа 2, двійка переноситься у наступний стовпець (вліво) у вигляді одиниці старшого розряду. Віднімання проводиться так само, як у десятковій системі, не замислюючись про те, що тепер у разі потреби потрібно займати зі стовпця зліва 2, а не 10.

Проте за цю легкість доводиться платити великим числом знаків при множенні навіть невеликих чисел.

У таблиці 1.1 наведено правила складання, віднімання і множення двійкових чисел.

Таблиця 1.1

Правила складання, віднімання і множення двійкових чисел

Складання	**Віднімання**	**Множення**
0 + 0 = 0	0 - 0 = 0	0 x 0 = 0
1 + 0 = 1	1 - 0 = 1	1 x 0 = 0
0 + 1 = 1	1 - 1 = 0	0 x 1 = 0
1 + 1 = 10	10 - 1 = 1	1 x 1 = 1

У загальному випадку процедури складання і віднімання двох чисел у будь-якій позиційній системі числення розпочинається з молодших розрядів.

Код суми кожного *і-того* розряду c_i виходить у результаті складання a_i+b_i+1, де одиниця відповідає перенесенню з молодшого *(і-1)* – розряду в *і-й*, якщо в молодшому розряді код суми вийшов більший або рівний основі системи числення.

Код різниці кожного *і-го* розряду виходить у результаті віднімання a_i-b_i-1, де одиниця відповідає „займу", якщо він був, у молодші розряди величини, основі підставі системи числення.

Отже, правила й методи складання і відіймання у будь-якій позиційній системі числення залишаються такими ж, як у десятковій системі.

Віднімання також проводиться порозрядно, починаючи з молодшого розряду. При відніманні в цьому розряді з нуля одиниці необхідно зайняти одиницю із сусіднього старшого розряду, яка дорівнює двом одиницям цього розряду.

Підсумовування двійкових чисел у комп'ютерах здійснюється за допомогою двійкових суматорів. Віднімання можна організувати за допомогою процедури складання, тобто за допомогою двійкових суматорів, якщо від'ємник представити в додатковому або зворотному коді.

Приклади

Приклад 1. Знайдемо суму двох чисел 1010 та 1011. Запишемо ці числа в стовпчик і складемо за загальним правилом:

0.0010 (2_{10})	+1010 (10_{10})	0.100111 (39_{10})
+0.0100 (4_{10})	1011 (11_{10})	+0.001101 (13_{10})
0.0110 (6_{10})	10101 (21_{10})	0.110100 (52_{10})
а)	б)	в)

Приклад 2. Визначимо різницю таких двох чисел: 10101 та 1010.

_ 10101 (21_{10})
1010 (10_{10})
1011 (11_{10}).

При знаходженні різниці необхідно пам'ятати, що зайнята в найближчому старшому розряді одиниця дає дві одиниці молодшого розряду. Якщо в сусідніх старших розрядах містяться нулі, то доводиться займати одиницю через кілька розрядів. При цьому одиниця, зайнята в найближчому значущому старшому розряді, дає дві одиниці молодшого розряду й одиниці в усіх нульових розрядах, що містяться між молодшим і тим старшим розрядом, у якого бралася позика.

Приклад 3.

0111 + 1 $\leftarrow$ після позики одиниці
Зменшуване 1000 (8_{10})
Від'ємник 11 (3_{10})
Різниця 101 (5_{10}).

Приклад 4.

0.0101 (5_{10})
x 0.0011 (3_{10})
0101
\+ 0101
0.1111 = 15_{10}

Ділення двійкових чисел. При діленні двійкових чисел використовуються двійкові таблиці множення і віднімання.

Приклад 5.

$X=430_{10}=110101110_2$; $y=10_{10}=1010$; $x/y=43_{10}$

```
_ 110101110 | 1010
  1010      | 101011
  _ 1101
    1010
    _1111
     1010
     _1010
      1010
      0000
```

а)

$x=204_{10}=11001100_2$, $y=12_{10}=1100_2$, $x/y=17_{10}$

```
          11001100 | 1100
          1100     | 10001
залишок   00001
          - 0
            11
          - 0
            110
          - 0
            1100
          - 1100
            0000
```

б)

Як видно з наведених вище прикладів, правила виконання арифметичних дій у десятковій і двійковій системах однакові.

Однак у двійковій системі арифметичні дії набагато простіші. Особливо простим є множення й ділення.

Дробові числа в позиційних системах числення зображуються таким чином: як і в цілих числах, вага двох сусідніх розрядів відрізняється в *q*-разів; вага зменшується зліва направо. Різниця полягає тільки в тому, що вага старшого розряду дорівнює *q-1* , молодшого (при *n*-розрядах) *q-n*.

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Отримати варіант завдання (табл. 1.2. Завдання під номером у таблиці).
4. Виконати операцію *x+y.*
5. Виконати операцію *x-y.*
6. Виконати операцію *x*y*.
7. Виконати операцію *x/y.*
8. Скласти звіт з лабораторної роботи.
9. Зробити висновки за результатами роботи.
10. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. Дайте визначення поняттю „система числення”.
2. Що являє собою двійкова система числення?
3. Що являє собою позиційна система числення?
4. Що являє собою непозиційна система числення?
5. Наведіть правила виконання арифметичних операцій у двійковій системі числення.
6. Якого порядку слід дотримуватися при розташуванні цифр та літер шістнадцятиричної системи числення?
7. Чому шістнадцятирична система числення вважається більш економічною та використовується для розташування даних у пам’яті комп’ютера?
8. Які недоліки вносить позиційність у запис числа порівняно з використанням його непозиційного подання?
9. Що таке нормалізована форма представлення числа?
10. Які критерії вибору системи числення для цифрових автоматів?

11. Як визначають похибки представлення чисел у цифрових автоматах?
12. Що таке поле даних?
13. Що таке упакований і розпакований формат?
14. Що таке прямий, додатковий і обернений коди?
15. Дайте визначення форми представлення числа в цифровому автоматі.
16. Що становить собою машинне (автоматне) зображення числа?
17. Що таке абсолютна похибка представлення числа в цифровому автоматі?

Таблиця 1.2

Початкові дані

Число / Варіант	**1**	**2**	**3**	**4**	**5**
x	010101	011011	011010	010010	010000
y	000011	000011	000010	000011	000100
Число / Варіант	**6**	**7**	**8**	**9**	**10**
X	001111	001010	001000	001001	001010
Y	000101	000101	000100	000011	000010
Число / Варіант	**11**	**12**	**13**	**14**	**15**
X	010010	010000	001111	001110	001100
Y	000110	000010	000011	000111	000110
Число / Варіант	**16**	**17**	**18**	**19**	**20**
X	000110	100000	011110	000110	010100
Y	000011	000100	000101	000010	000010
Число / Варіант	**21**	**22**	**23**	**24**	**25**
X	001110	001110	001100	001001	001110
Y	000101	000101	000100	0000001	000011

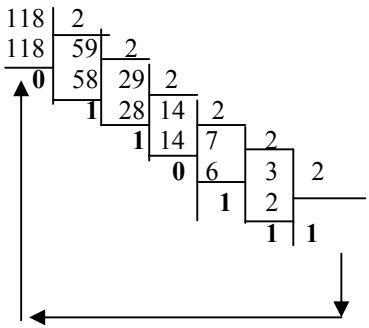
Лабораторна робота № 2. Переведення чисел з однієї позиційної системи числення в іншу

Мета роботи: закріплення знань з переведення чисел з однієї системи в іншу та форм представлення чисел у ЦА.

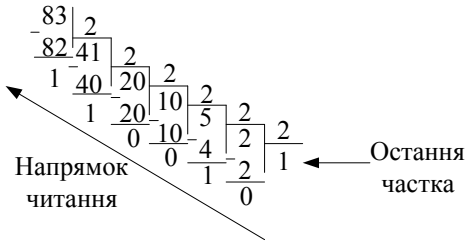
Загальні теоретичні відомості

Загальне правило переводу цілих чисел. Для переводу цілого числа з однієї позиційної системи в іншу його треба послідовно поділити на основу *q* тієї системи, у яку воно переводиться, доти, доки не вийде частка, менше *q*. Число в новій системі запишеться у вигляді залишків ділення, починаючи з останнього. Остання частка дає більшу цифру числа. Пояснимо це правило на прикладі.

Приклад 1. Перевести число 118 з десяткової системи числення у двійкову.



Приклад 2. Десяткове число 83 перевести у двійкову систему числення:



Приклад 3. Число 372_8 із восьмеричної системи числення у десяткову.

$3728 = 3 \times (82) + 7 \times (81) + 2 \times (80) =$
$= 3 \times 64 + 7 \times 8 + 2x\ 64 = 25010$

Приклад 4. Переведемо число 135_8 у десяткову систему числення.

$135_8 \mid 12_8$
12_8 | 11_8
15_8
12_8
3_8

Одержали неподільне – 11_8 і залишок 3_8, які представляють значення цифр десяткового числа. Ураховуючи, що $3_8 = 3_{10}$, а $11_8 = 9_{10}$, одержимо $135_8 = 93_{10}$.

Приклад 5. Переведемо число $24{,}6_8$ у десяткову систему числення.

$24{,}68 = 2 \times (81) + 4 \times (80) + 6 \times (8\text{-}1)$
$= 2 \times 8 + 4 \times 1 + 6 \times 0{,}8$
$= 16 + 4 + 0{,}75 = 20{,}7510$.

Загальне правило переведення правильних дробів. Для переведення правильного дробу з однієї позиційної системи в іншу

його потрібно послідовно множити на основу тієї системи, у яку він переводиться. Перемножуються тільки дробові частини; дріб у новій системі запишеться у вигляді цілих частин добутків, що виходять, починаючи з першого.

Приклад 6. Десятковий дріб 0,3125 перевести у двійкову систему числення:

Напрямок читання ↓

$$\begin{array}{l} {}_{x}0{,}3125 \\ \underline{\qquad 2} \\ {}_{x}0{,}6250 \\ \underline{\qquad 2} \\ {}_{x}1{,}2500 \\ \underline{\qquad 2} \\ {}_{x}0{,}5000 \\ \underline{\qquad 2} \\ 1{,}0000 \end{array}$$

Отже, $0{,}3125_{10} = 0{,}0101_{2}$.

Двійкове число може бути розкладене за ступенями 2 у будь-якій системі числення.

Приклад 7. Переведення ціле число з двійкової системи числення в десяткову:

$$100111 = 1\cdot 2^{5}+0\cdot 2^{4}+0\cdot 2^{3}+1\cdot 2^{2}+1\cdot 2^{1}+1\cdot 2^{0}=32+4+2+1=39.$$

Переведення дробу з двійкової системи числення в іншу проводиться за допомогою послідовного множення дробу на основу системи.

Приклад 8. Перевести дробове число з двійкової системи числення в десяткову:

$$1011{,}01101 =$$

$$1\cdot 2^{3}+0\cdot 2^{2}+1\cdot 2^{1}+1\cdot 2^{0}+0\cdot 2^{-1}+1\cdot 2^{-2}+1\cdot 2^{-3}+0\cdot 2^{-4}+1\cdot 2^{-5}=8+2+1+\frac{1}{4}$$

$$+\frac{1}{8}+\frac{1}{32}=11\frac{13}{32}.$$

Переведення двійкового числа в десятковий його еквівалент можна виконати за допомогою такого дуже простого способу. Перетворення здійснюється шляхом підсумовування значень ступенів числа 2, що відповідають тим розрядам двійкового числа, що переводиться, у яких містяться одиниці. Наприклад, переведемо 101110102 у десяткове число.

Розряди числа, що переводиться	7	6	5	4	3	2	1	0
Число	1	0	1	1	1	0	1	0
Ступінь числа 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Результат	128	0	32	16	8	0	2	0

Шістнадцятирична система числення – це система числення з основою 16, в якій для зображення чисел слугують такі символи: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Буквені позначення введено тому, що незручно зображувати одну цифру двома символами. Для полегшення переведення чисел з однієї позиційної системи числення в іншу можна скористатися таблицею 2.1.

Таблиця 2.1

Переведення десяткових чисел у шістнадцятирічні числа

Множник	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	1	2	3	4	5	6	7	8	9
10	A	14	1E	28	32	3C	46	50	5A
10^2	64	C8	12C	190	1F4	258	2BC	320	384
10^3	3E8	7D0	BB8	FA0	1388	1770	1B58	1F40	2328

Множник	1	2	3	4	5	6	7	8	9
10^4	2710	4E20	7530	9C40	C350	EA60	11170	13880	15F90
10^5	186A0	30D40	493E0	61A80	7A120	927C0	AAE60	C3500	DBBA0
Приклад: 1234=1000+200+30+4=(3E8)$_{16+}$(C8)$_{16+}$(1E)$_{16}$+(4)$_{16}$=(4D2)$_{16}$									

Приклад 9. Шістнадцятиричне число IFA,C24$_{16}$ переводиться у двійкове число:

I	F	A,	C	2	4
↓	↓	↓	↓	↓	↓
0001	1111	1010,	1100	0010	0100
7	6	3,	2	2	4
↓	↓	↓	↓	↓	↓
111	110	011,	010	010	100

Остаточно одержимо IFA,C24$_{16}$=111111010,1100001$_2$.

Для переведення чисел з двійкової системи числення у шістнадцятиричну (восьмеричну) необхідно двійкові цифри числа, що переводиться, згрупувати по чотири (три) в обидві сторони від коми. За необхідності неповні групи цифр доповнити 0. Кожну з одержаних груп двійкових цифр замінити відповідною їй цифрою у новій системі числення, а одержані шістнадцятиричні (восьмеричні) цифри розташувати на місцях замінюваних двійкових кодів.

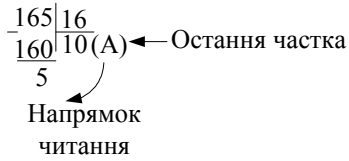
Наприклад, переведемо число 111111010,1100001$_2$ у шістнадцятиричне.

0001	1111	1010	1100	0010	0100
I	F	A	C	2	4

Приклад 10. Перевести з шістнадцятиричної системи числення число BA у десяткову: $BA=B\cdot16^1+A\cdot16^0=11\cdot16+10\cdot1=186$.

Переведення з десяткової системи в шістнадцятиричну проводиться за загальним правилом переведення цілих чисел.

Приклад 11. Перевести число 165 з десяткової системи числення в шістнадцятиричну:



Для того, щоб перевести з двійкової системи числення в шістнадцятиричну, достатньо подати це число у вигляді умовно поділеного на чотирьохрозрядні двійкові коди (тетради).

Приклад 12. Перевести число 10110110001101 з двійкової системи числення в шістнадцятиричну:

10110110001101(2)= 0010 1101 1000 1101 = 2D8D(16).

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Обрати за методичними вказівками варіант завдання.
4. Зробити переведення десяткового числа (*x)* у шістнадцятиричну та двійкову системи числення. Виконати перевірку переводу із допомогою інженерного калькулятора.
5. Перевести шістнадцятиричне число (*y)* у десяткову систему числення.
6. Перевести правильний десятковий дріб (число *z*) у двійкову систему числення.
7. Скласти звіт з лабораторної роботи.
8. Зробити висновки щодо роботи.
9. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. Привести загальне правило переведення цілих чисел.
2. Привести загальне правило переведення правильних дробів.
3. Як перевести число з двійкової у будь-яку систему числення?
4. Що являє собою шістнадцятирична система числення?
5. Як проводиться переведення з шістнадцятиричної у будь-яку систему числення?
6. З якою метою у запису чисел використовуються тетради?
7. Яке значення для сприйняття числа має напрямок читання його розрядів при виконанні переведення з іншої системи числення?

Таблиця 2.2

Початкові дані

Число / Варіант	**1**	**2**	**3**	**4**	**5**	**6**
x	68104	31508	76312	11127	29315	86114
y	F4A1	AF7B	D3AA	EEA7	BC61	73AF
z	0,3287415	0,0231578	0,4403212	0,5401117	0,2213454	0,00245963
Число / Варіант	**7**	**8**	**9**	**10**	**11**	**12**
x	44801	76503	20422	14711	12312	96433
y	A21D	BC1D	A1BD	DA01	F34A	DC23
z	0,5585937	0,4325212	0,3387423	0,1218973	0,2631874	0,1591979
Число / Варіант	**13**	**14**	**15**	**16**	**17**	**18**
x	82040	43561	75423	67141	45806	54302
y	4D18	AFC7	A38C	A38C	BF7C	EF6A
z	0,2610195	0,2101489	0,2912201	0,0801234	0,1504484	0,3405401
Число / Варіант	**19**	**20**	**21**	**22**	**23**	**24**
x	15919	84432	24611	18441	32412	54716
y	AE76	1EBA	E3F7	A02B	B1AE	AEF1
z	0,2203445	0,7442612	0,3214572	0,8431454	0,3071629	0,9261413

Приклади тестових завдань

1. Перевести число 76 з десяткової в двійкову систему числення:

а) 0110011; б) 1001100; в) 0011001.

2. Перевести десятковий дріб 0,125 у двійкову систему числення:

а) 0,00111; б) 0,11100; в) 0,11000.

3. Перевести число 100110001 з двійкової системи числення в десяткову: а) 817; б) 318; в) 1015.

4. Перевести дробове число 101,0110 з двійкової системи числення в десяткову:

а) 6 1/8; б) 101, 5; в) 5 3/8.

5. Перевести число АС з шістнадцятиричної системи числення в десяткову:

а) 172; б) 22; в) 1012.

Модуль 2 Арифметичні дії в ЦА. Контроль роботи ЦА.

Лабораторна робота № 3. Мінімізація логічної функції, що описує роботу цифрового автомата

Мета роботи: закріплення знань з мінімізації логічних функцій.

Загальні теоретичні відомості

Прагнення до алгоритмізації пошуку елементарних добутків зумовило розробку табличних методів мінімізації логічних функцій. Одним з них є метод, що ґрунтується на використанні карт Карно.

Карти Карно (їх різновидом є карти Вейча, побудовані як розгортки куба на площині), є *графічним представленням таблиць істинності*. Тому їх будують або за таблицею істинності аналізованої функції, або ж за її *ДНДФ*.

Нагадаємо, що кожен рядок таблиці істинності, для якої функція дорівнює одиниці, відповідає *мінтерму* функції, представленої у ДНДФ. Рядок, для якого функція дорівнює нулю, є *макстермом* функції, представленої в ДНКФ.

Карта Карно становить собою прямокутник, розбитий на квадрати, кількість яких дорівнює загальній кількості наборів для цієї функції n змінних, тобто воно дорівнює 2^n. Так, для функції чотирьох змінних квадратів буде 16, для п’яти змінних – 32 і т. ін. Кожен квадрат відповідає певному набору або терму, причому набори розташовано так, щоб *сусідні набори або терми й за горизонталлю, і за вертикаллю відрізнялися б лише значенням однієї змінної: в одному квадраті вона з інверсією, а в іншому, сусідньому – без*. Функцію у *ДНДФ* наносять на карту, позначаючи, наприклад, знаком 1 квадрати, що відповідають тим наборам, на яких функція дорівнює одиниці, тобто у *ДНДФ* функції є відповідний *мінтерм*. Інші квадрати позначають знаком 0. Іноді в

куті квадрата ставлять номер набору. Такий спосіб використовують, якщо функцію задано числовим способом, але він незручний для процедури мінімізації.

Перерахуємо послідовність дій, виконуваних для мінімізації функції за методом Карно:

1) вихідна функція, яка підлягає мінімізації, має бути представлена у *НДФ*. Потім її треба представити у *ДНДФ*. Або ж складають таблицю істинності мінімізованої функції. Як уже було відзначено, між рядками таблиці істинності й клітинами (гніздами) на карті Карно існує взаємооднозначна відповідність. Коли карта Карно складається за *ДНДФ* мінімізованою функцією, то очевидно, що кожна змінна без заперечення замінюється її значенням 1, а із запереченням – 0;

2) потім будують карту Карно за принципом, описаним вище. Подамо систему координат, у якій, наприклад, для функції двох перемінних на горизонтальній осі відкладаються значення одного аргументу, а на вертикалі – іншого. На перетині відповідних координат отримуємо гніздо, куди записуємо значення функції (0 або 1), що відповідає цьому набору. Якщо функцію подано у *ДНДФ*, то в гнізді, що відповідає існуючому *мінтерму,* записуємо 1, а в гнізді неіснуючого *мінтерму* – 0;

3) після цього суміжні гнізда, у яких записано одиниці, об'єднують у групи в такий спосіб: об'єднують обов'язково парну кількість сусідніх гнізд з одиницями, і за вертикаллю, і за горизонталлю. Причому, кожне гніздо з 1 може потрапити одночасно в дві групи, отримані внаслідок об'єднання одиниць і за вертикаллю, і за горизонталлю;

4) кожній групі ставлять у відповідність новий *мінтерм* для зображення вихідної функції у формі мінімальної *НДФ*;

5) зображення кожного нового *мінтерму* формують за таким алгоритмом:

а) змінну, яка в кожному гнізді утвореної групи має значення тільки 0, зображують її інверсією;

б) змінну, яка в кожному гнізді утвореної групи має значення тільки 1, зображують без інверсії;

в) змінну, яка в межах утвореної групи змінює своє значення, не зображують, тобто відкидають.

Таким чином, карту Карно можна розглядати як графічне представлення сукупності всіх (існуючих і неіснуючих) *мінтермів* функції в *ДНДФ* певної кількості логічних змінних.

Розглянемо табл. 3.1, у якій уміщено таблицю істинності й карту Карно функції $y = f(A,B,C) = \overline{A} \vee \overline{B} \cdot C.$ Гнізда, у яких функція набуває значення, що дорівнюють одиниці, заповнено одиницями. В інші гнізда записано нулі. Процес мінімізації полягає у формуванні прямокутників, що містять по 2^k гнізда, де k – ціле число. У прямокутники об'єднано сусідні гнізда, що відповідають сусіднім елементарним добуткам. Наприклад, об'єднано гнізда з координатами 001 і 101. При об'єднанні цих гнізд утворився прямокутник, у якому змінна A змінює своє значення. Отже, вона зникає при склеюванні відповідних елементарних добутків $\overline{A} \cdot \overline{B} \cdot C$ і $A \cdot \overline{B} \cdot C.$ Гнізда, розташовані в першому рядку карти Карно, містять одиниці та є сусідніми. Тому всі вони об'єднані в прямокутник, що містить 2^2=4 гнізда.

Сукупність прямокутників, які покривають усі одиниці, називають покриттям. Слід відзначити, що одне й те саме гніздо (наприклад, з координатами 001) може покриватися два й кілька разів.

Таблиця 3.1

Таблиця істинності та структура карти Карно
для логічної функції трьох змінних
$y = f(A,B,C) = \overline{A} \vee \overline{B} \cdot C.$

Таблиця істинності логічної функції трьох перемінних			
Перемінні			**Функція**
A	*B*	*C*	*f(A,B,C)*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Карта Карно логічної функції трьох перемінних
$y = f(A,B,C) = \overline{A} \vee \overline{B} \cdot C.$

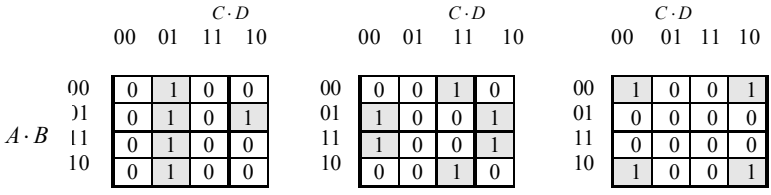
	BC			
	00	**01**	**11**	**10**
0	**1**	**1**	**1**	**1**
A				
1	**0**	**1**	**0**	**0**

Змінні B і C у межах прямокутника змінюють своє значення, отже, вони зникнуть з результуючого елементарного добутку. Перемінна A залишається незмінною й дорівнює нулю. Таким чином, елементарний добуток, отриманий у результаті об'єднання гнізд першого рядка, містить лише один елемент $\overline{A}$. Це, зокрема, випливає з того, що чотирьом гніздам першого рядка відповідає сума чотирьох елементарних добутків:

$$\overline{A}\cdot\overline{B}\cdot\overline{C}\vee\overline{A}\cdot\overline{B}\cdot C\vee\overline{A}\cdot B\cdot C\vee\overline{A}\cdot B\cdot\overline{C}=\overline{A}\cdot\overline{B}\cdot(\overline{C}\vee C)\vee\overline{A}\cdot B\cdot(C\vee\overline{C})=$$
$$=\overline{A}\cdot\overline{B}\vee\overline{A}\cdot B=\overline{A}\cdot(B\vee\overline{B})=\overline{A}.$$

Таким чином, формула, яку отримуємо в результаті мінімізації логічної функції за допомогою карт Карно, містить суму стількох елементарних добутків, скільки прямокутників є в покритті. Чім більше гнізд у прямокутнику, тим менше змінних міститься у відповідному йому елементарному добутку. Наприклад, для карти Карно, зображеної на рис. 3.1 а, прямокутнику, який містить чотири гнізда, відповідає елементарний добуток $\overline{C}\cdot D$ двох змінних, а квадрата, що складається лише з одного гнізда, – елементарний добуток $\overline{A}\cdot B\cdot C\cdot\overline{D}$, що містить усі чотири змінні. Функція, що відповідає покриттю, показаному на рис. 3.1 а, має вигляд

$$y=f(A,B,C,D)=\overline{C}\cdot D\vee\overline{A}\cdot B\cdot C\cdot\overline{D}.$$



а б в

Рис. 3.1. Карти Карно функції чотирьох змінних

Незважаючи на те, що карти Карно зображують на площині, сусідство квадратів установлюють на поверхні тора. Верхня й

нижня межі карти Карно ніби склеюються, утворюючи поверхню циліндра. При склеюванні бокових меж утворюється поверхня у вигляді тора. Таким чином, гнізда з координатами 1011 і 0011, зображені на рис. 3.1 б, є сусідніми, об'єднуються в прямокутник. Указаним гніздам відповідає сума елементарних добутків $A \cdot \overline{B} \cdot C \cdot D \vee \overline{A} \cdot \overline{B} \cdot C \cdot D = (A \vee \overline{A}) \cdot \overline{B} \cdot C \cdot D = \overline{B} \cdot C \cdot D.$

Аналогічно об'єднуються й інші чотири одиничні гнізда. У результаті їх об'єднання отримуємо елементарний добуток $B \cdot \overline{D}$. Остаточно функція, що відповідає покриттю, зображеному на рис. 3.1 б, має вигляд $f(A,B,C,D) = \overline{B} \cdot \overline{D} \vee \overline{B} \cdot C \cdot D.$

Розглянемо ще один приклад. Карта Карно, наведена на рис. 3.1 в, містить одиничні гнізда, розташовані в кутах. Усі чотири гнізда є сусідніми й після об'єднання дадуть елементарний добуток $\overline{B} \cdot \overline{D}$.

При формуванні на базі карти Карно нових мінтермів для представлення функції в мінімальній НДФ або ж у мінімальній НКФ значення відповідних змінних, що дорівнюють 1, замінюються зображенням змінних без заперечення, а при значеннях, що дорівнюють 0, – із запереченням.

Розглянемо процес мінімізації на прикладі функції, заданої таким логічним рівнянням:

$$f(A,B,C,D) = B \cdot C \cdot D + \overline{A} \cdot B \cdot D + \overline{B} \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{C} \cdot D + \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot \overline{C}.$$

Представимо цю функцію в *ДДНФ*:

$$\begin{aligned} f = {} & B \cdot C \cdot D \cdot (A + \overline{A}) + \overline{A} \cdot B \cdot D \cdot (C + \overline{C}) + B \cdot C \cdot D \cdot (A + \overline{A}) + \\ & + A \cdot \overline{B} \cdot \overline{C} \cdot (D + \overline{D}) + A \cdot \overline{C} \cdot D \cdot (B + \overline{B}) + \\ & + \overline{B} \cdot \overline{C} \cdot D \cdot (A + \overline{A}) + \overline{A} \cdot B \cdot C \cdot (D + \overline{D}) + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot (D + \overline{D}) = \\ & A \cdot B \cdot C \cdot D + \overline{A} \cdot B \cdot C \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot D + \\ & + A \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot D + \\ & \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D. \end{aligned}$$

Нижче зображено карту Карно, що відповідає аналізованій функції (див.: рис. 3.2).

Мінтерми функції утворюють у карті чотири групи: $\overline{A}\cdot\overline{B}\cdot\overline{C}$, $A\cdot\overline{B}\cdot\overline{C}$, D, $(\overline{A}\cdot B\cdot C)$.

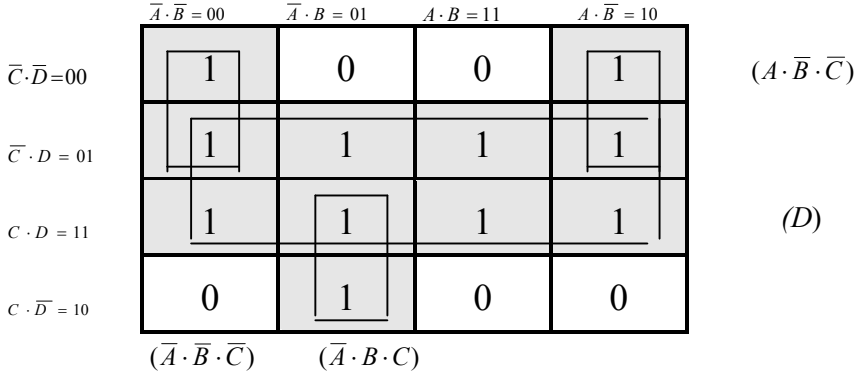


Рис. 3.2. Карта Карно

Таким чином, отримаємо

$$f=\overline{A}\cdot\overline{B}\cdot\overline{C}+A\cdot\overline{B}\cdot\overline{C}+\overline{A}\cdot B\cdot C+D=\overline{B}\cdot\overline{C}\cdot(A+\overline{A})+\overline{A}\cdot B\cdot C+D=$$
$$=\overline{B}\cdot\overline{C}+\overline{A}\cdot B\cdot C+D.$$

Разом з тим, треба мати на увазі, що в загальному випадку функція може мати кілька мінімальних форм.

Хоча зазвичай карти Карно для функцій трьох і чотирьох змінних зображаються на площині, але, як було відзначено, з огляду на формування прямокутних груп карту треба вважати тривимірною.

Для карт Карно функції трьох змінних карту слід розглядати як циліндр зі склеєними правим і лівим краями. Оскільки прямокутні групи формуються на такому циліндрі, на плоскому рисунку та чи інша група може виявитися розірваною.

Розглянемо раніше наведений приклад, але тепер представимо карту у вигляді циліндра. Тоді в нас вийде три групи мінтермів замість чотирьох (див.: рис. 3.3).

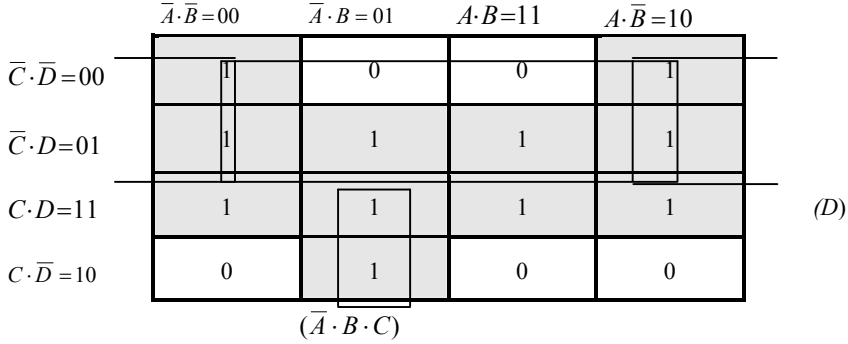


Рис. 3.3. Карта Карно

Мінтерми функції утворюють на карті три групи: $\overline{B}\cdot\overline{C}$ – мінтерм, що відповідає групі з чотирьох верхніх одиниць, другий мінтерм – D; третій – $\overline{A}\cdot B\cdot C$. Отже, отримаємо $f=\overline{B}\cdot\overline{C}+\overline{A}\cdot B\cdot C+D$.

При отриманні мінімальної форми для *ДНКФ* функцію задано термами, які набувають нульового значення на відповідних наборах, тобто макстермами. Тому в клітках мінімізуючих карт пишуть нулі, які групують за описаною раніше методикою.

За наявності сформованої мінімальної *НДФ* функції мінімальну *НКФ* можна також знайти так. Беруть заперечення від знайденої мінімальної *НДФ* і здійснюють перетворення за формулою де Моргана. Таким чином, виходить мінімальна *НКФ* функції. Наприклад, якщо *МНДФ* функції дорівнює $f(A,B,C)=\overline{\overline{A}\cdot B+A\cdot\overline{C}}$, то, провівши заперечення й застосувавши теорему де Моргана, отримаємо *МКФ* – $f(A,B,C)=\overline{\overline{A}\cdot B+A\cdot\overline{C}}=(A+\overline{B})(\overline{A}+C).$

Завдання: за допомогою карт Карно мінімізувати логічний вираз (див.: табл. 3.1). Перевірку здійснити за допомогою іншого метода мінімізації, наприклад Квайна (Квайна-Мак-Класкі) або використовуючи теореми та правила алгебри логіки.

Таблиця 3.1

Варіанти завдання лабораторної роботи

Варіант	Логічний вираз
1	$y = A \cdot (\overline{B} \cdot \overline{D} + D) + A \cdot \overline{B} \cdot C + \overline{D} + \overline{A \cdot \overline{B} \cdot C}.$
2	$y = \overline{\overline{A} \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot D} + \overline{A \cdot \overline{B} \cdot C} + \overline{D} \cdot \overline{B} \cdot C.$
3	$y = \overline{A} \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot D + \overline{D} + \overline{A \cdot \overline{B} \cdot \overline{C}}.$
4	$y = A \cdot \overline{B} \cdot C + A \cdot (\overline{\overline{B} \cdot \overline{D} + C}) + \overline{D} + A \cdot \overline{B} \cdot C.$
5	$y = A \cdot \overline{B} + A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A \cdot \overline{B} \cdot C.$
6	$y = A \cdot \overline{B} + A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D \cdot \overline{A} \cdot \overline{B} \cdot \overline{C}}.$
7	$y = \overline{A} \cdot \overline{B} + A \cdot (\overline{B} \cdot \overline{D} + C \cdot \overline{B}) + \overline{D} \cdot A \cdot \overline{B} \cdot C + \overline{A} \cdot B.$
8	$y = \overline{A} \cdot \overline{B} \cdot D + A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C.$
9	$y = A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{D} + C.$
10	$y = \overline{C} (\overline{A} \cdot \overline{B} \cdot \overline{D} + D) + \overline{A \cdot \overline{B} \cdot C + \overline{D}}.$
11	$y = A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot (A \cdot \overline{B} \cdot \overline{C} + \overline{\overline{A} \cdot B \cdot \overline{C}}).$
12	$y = A \cdot (\overline{B} \cdot \overline{D} + C) + D \cdot (A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}).$
13	$y = B \cdot (A \cdot \overline{D} + C) + \overline{D} \cdot (\overline{B} \cdot \overline{C} \cdot A + \overline{A} \cdot B \cdot C) + D.$
14	$y = A \cdot \overline{B} \cdot \overline{D} \cdot C + A \cdot (\overline{B} \cdot \overline{D} + C \cdot D) + D \cdot \overline{A} \cdot \overline{B} \cdot \overline{C}.$
15	$y = C \cdot (\overline{A} \cdot \overline{D} + B) + \overline{D} \cdot (A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}).$
16	$y = A \cdot \overline{B} \cdot \overline{D} + C \cdot \overline{D} + \overline{A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}}.$
17	$y = (\overline{C + D}) + \overline{A \cdot \overline{C} \cdot \overline{D}} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C \cdot D + A \cdot C \cdot D.$
18	$y = A \cdot B \cdot (\overline{\overline{C} \cdot D}) + A \cdot \overline{B} \cdot D + \overline{B} \cdot \overline{C} \cdot \overline{D}.$
19	$y = A \cdot (\overline{\overline{B} \cdot \overline{D} + C}) + \overline{D} \cdot A \cdot \overline{B} \cdot \overline{C} + \overline{\overline{A} \cdot B \cdot \overline{C}} + A \cdot \overline{D} + C.$
21	$y = \overline{A} \cdot \overline{B} + \overline{A \cdot (\overline{B} \cdot \overline{D} + C \cdot \overline{B})} + \overline{\overline{D} \cdot A \cdot \overline{B} \cdot C + \overline{A} \cdot B}.$
22	$y = A \cdot \overline{B} + \overline{A \cdot (\overline{B} \cdot \overline{D} + C)} + \overline{D \cdot \overline{A} \cdot \overline{B} \cdot \overline{C}}.$
23	$y = \overline{A \cdot (\overline{B} \cdot \overline{D} + D) + A \cdot \overline{B} \cdot C} + \overline{D} + \overline{A \cdot \overline{B} \cdot C}.$
24	$y = A \cdot B \cdot (\overline{\overline{C} \cdot D}) + A \cdot \overline{B} \cdot D + \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{C \cdot \overline{B} \cdot D}.$

Хід виконання роботи

1. Вивчити теоретичні відомості.
2. Отримати допуск до роботи.
3. Обрати за методичними вказівками варіант завдання та мінімізувати логічний вираз.
4. Переконатися в правильності отриманого виразу за допомогою програми EWB.

Контрольні питання

1. Сформулювати визначення логічної функції, конституенти, імпліканти функції.
2. Що таке досконала, скорочена, тупикова й мінімальна ДНФ?
3. Дати визначення функціонально повної системи перемикальних функцій.
4. У чому сутність проблеми мінімізації перемикальних функцій?
5. Схарактеризувати основні етапи мінімізації перемикальних функцій різними методами.
6. Як побудувати операторні форми представлення функцій?
7. Дайте порівняльну оцінку методів мінімізації функцій.

Модуль 3 Основи алгебри логіки. Мінімізація логічних функцій.

Лабораторна робота № 4. Експериментальні дослідження комбінаційних схем у програмах Electronics Workbench та Multisim

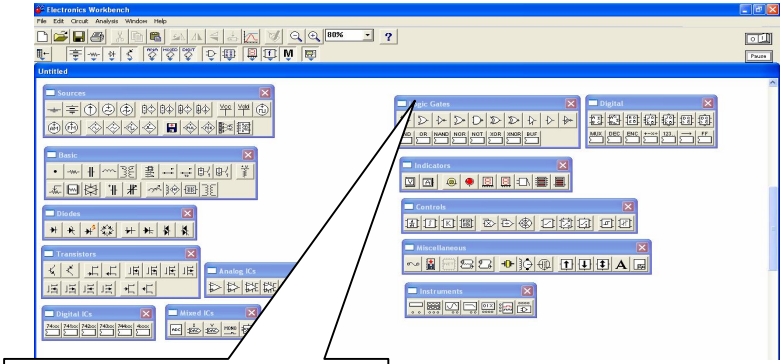
Мета роботи: навчитися моделювати роботу комбінаційних схем у програмах Electronics Workbench та Multisim.

Загальні теоретичні відомості

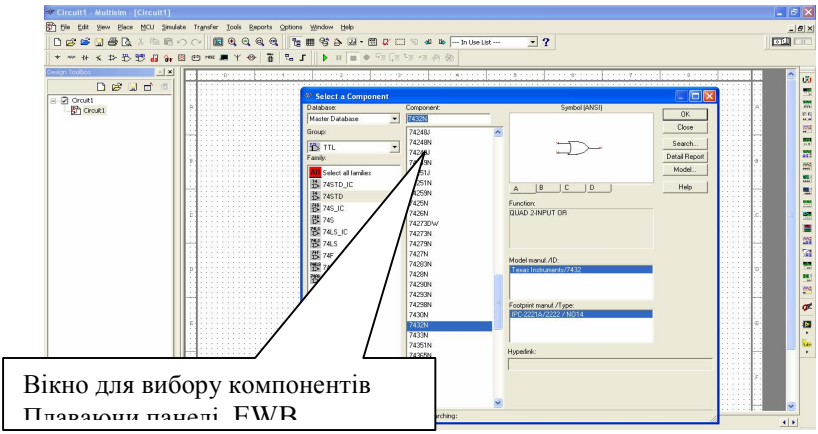
Системи схемотехнічного моделювання *Electronics Workbench* та *Multisim* призначено для моделювання й аналізу електричних схем (див.: рис. 4. 1).

Системи *Electronics Workbench* (рис. 4.1 а) та *Mulsim* (рис. 4.1 б) дозволяють моделювати аналогові, цифрові й цифро-аналогові схеми великого ступеня складності. Бібліотеки, що є в програмах, містять великий набір поширених електронних компонентів. Є можливість підключення і створення нових бібліотек-компонентів. Параметри компонентів можна змінювати в широкому діапазоні значень. Прості компоненти описуються набором параметрів, значення яких можна змінювати безпосередньо з клавіатури, активні елементи – у вигляді моделі, що описує конкретний елемент або його ідеальне уявлення. Модель обирається зі списку бібліотек-компонентів, параметри моделі також можуть бути змінені користувачем. Широкий набір приладів дозволяє проводити вимірювання різних величин, задавати вхідні дії, будувати графіки. Усі прилади зображаються у вигляді, максимально наближеному до реального. Результати моделювання можна вивести на принтер або імпортувати в текстового або графічного редактора для їх подальшої обробки. Програми *Electronics Workbench (Mulsim)* сумісні з програмою *P-SPICE,* тобто надає можливість експорту й імпорту схем і результатів

вимірювань у різні її версії. Працюючи з *Electronics Workbench (Mulsim)*, експериментатор застрахований від випадкової поразки струмом, а прилади не вийдуть з ладу через неправильно зібрану схему. Завдяки цій програмі в розпорядженні користувача є такий широкий набір приладів, який навряд чи буде доступний у реальному житті. Таким чином, у Вас завжди є унікальна можливість для планування і проведення широкого спектра досліджень електронних схем за мінімальних витрат часу [3, с.7].



а)



б)

Рис. 4.1. Загальний вигляд Electronics Workbench та Mulsim 10

Усі операції проводяться за допомогою миші та клавіатури. Управління тільки з клавіатури є неможливим. Шляхом настроювання приладів можна: змінювати шкали приладів залежно від діапазону вимірювань, задавати режим роботи приладу, задавати вид вхідних дій на схему (постійні й гармонійні струми й напруги, трикутні та прямокутні імпульси). Графічні можливості програми дозволяють: одночасно спостерігати кілька кривих на графіку, відображати криві на графіках різними кольорами, вимірювати координати точок на графіку, імпортувати дані до графічного редактора, що дозволяє провести необхідні перетворення малюнка і виведення його на принтер. *Electronics Workbench* (*EWB*) дозволяє використовувати результати, одержані в програмах *P-SPICE, PCB*, а також передавати результати з *Electronics Workbench* в ці програми. Можна вставити схему або її фрагмент у текстовому редакторі й надрукувати в ньому пояснення або зауваження щодо роботи схеми.

При побудові й редагуванні схем виконуються такі операції [3, с.7]:

- вибір компонента з бібліотеки компонентів;
- виділення об’єкта;
- переміщення об’єкта;
- копіювання об’єктів;
- видалення об’єктів;
- з’єднання компонентів схеми провідниками;
- установка значень компонентів;
- підключення приладів.

Якщо схема не вміщається на екрані монітора, будь-яку її ділянку можна проглянути за допомогою лінійок прокрутки, розташованих справа й під робочим полем. Після побудови схеми й підключення приладів аналіз її роботи розпочинається після натиснення вимикача в правому верхньому кутку вікна програми.

Панель компонентів складається з піктограм полів компонентів, поле компонентів – з умовних зображень компонентів.

До бібліотек елементів програми *EWB* належать аналогові, цифрові й цифро-аналогові компоненти. Усі компоненти можна умовно розподілити на такі групи:

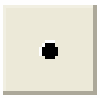
- базові компоненти;

• джерела;
• лінійні компоненти;
• ключі;
• нелінійні компоненти;
• індикатори;
• логічні компоненти;
• вузли комбінаційного типу;
• вузли послідовного типу;
• гібридні компоненти.

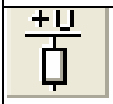
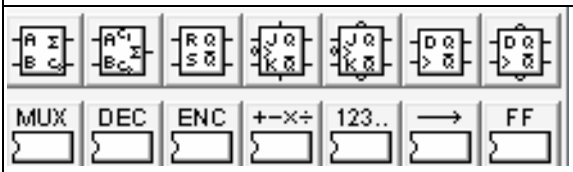
У таблиці 4.2 наведеніо основні компоненті *EWB,* які ми будемо використовувати в подальшій роботі при моделюванні цифрових пристроїв та їхній опис (компоненти пакета *Mulsim* мають аналогічний вигляд, тому в таблиці 4.2 не наведено).

Таблиця 4.2

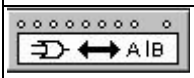
Основні компоненті *EWB* для моделювання цифрових пристроїв

Пікто-грама	Призначення компонента
	*Сполучний вузол*. Застосовується для з'єднання провідників і створення контрольних точок. До кожного вузла може під'єднуватися не більше чотирьох провідників.
	*Заземлення*. Не всі схеми потребують заземлення для моделювання, проте, будь-яка схема, що містить: • операційний підсилювач; • трансформатор; • кероване джерело; • осцилограф повинна бути обов'язково заземлена, інакше прилади не проводитимуть вимірювання або їх свідчення виявляться неправильними.
	*Генератор тактових імпульсів*. Виробляє послідовність прямокутних імпульсів. Відлік амплітуди імпульсів генератора проводиться від «-».

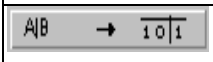
Продовження таблиці 4.2

	*Джерело напруги +5 В*. Використовуючи це джерело напруги, можна встановлювати фіксований потенціал вузла 5 В або рівень логічної одиниці.
	*Джерело сигналу „логічна одиниця”*. За допомогою цього джерела встановлюють рівень логічної одиниці у вузлі схеми.
	***Ключ, керований клавішею.*** Ключі можуть бути замкнуті або розімкнені за допомогою керівних клавіш на клавіатурі. Ім’я укерівної клавіші можна ввести з клавіатури в діалоговому вікні, що з’являється після подвійного клацання мишею на зображенні ключа. Клавіші-ключі, що використовуються – букви ввід *A* до *Z*; цифри від 0 до 9; клавіша Enter на клавіатурі; клавіша „пробіл”.
	*Дешифрувальний семисегментний індикатор*. Слугує для відображення на своєму дисплеї шістнадцятиричних чисел від 0 до і, що задаються станом на вході індикатора.
	*Пробник логічного рівня*. Визначає логічний рівень (0 або 1) у конкретній точці схеми. Якщо досліджуваний вузол має рівень логічної одиниці, індикатор спалахує червоним кольором. За допомогою команди *Value* у меню *Circuit* можна змінити колір свічення пробника.
	Логічні елементи та відповідні інтегральні схеми
	Вузли комбінаційного типу та послідовного типу (тригери)

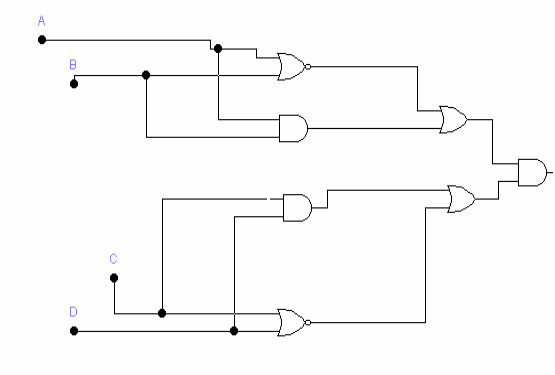
Продовження таблиці 4.2

	На панелі розташовані прилади для формування і нагляду аналогових величин: мультиметр, функціональний генератор, осцилограф, Боде-плотер, генера-тор слів, логіч-ний аналізатор, логічний перетворювач
	*Логічний перетворювач* – прилад, який не має аналогів у реальному світі. Він призначений для виконання різних функціональних перетворень у схемі. З його допомогою можна здійснювати такі операції: а) отримання таблиці істинності досліджуваної схеми; б) перетворення таблиці істинності в логічний вираз; в) перетворення логічного виразу в таблицю істинності; г) створення логічних схем за заданим логічним виразом; д) синтез логічних схем на елементах *І-НЕ по заданному логічному виразу. На екран виводиться зменшене зображення логічного перетворювача*.
Основні кнопки логічного перетворювача	
	*Отримання таблиці істинності схеми.* Необхідно підключити входи **A, B, C, D, E, F, G, H** логічного перетворювача до входів схеми (не більше восьми) та вихід логічного перетворювача з'єднати із виходом схеми. Після натискання кнопки в лівій частині екрана логічного перетворювача з'явиться таблиця істинності.

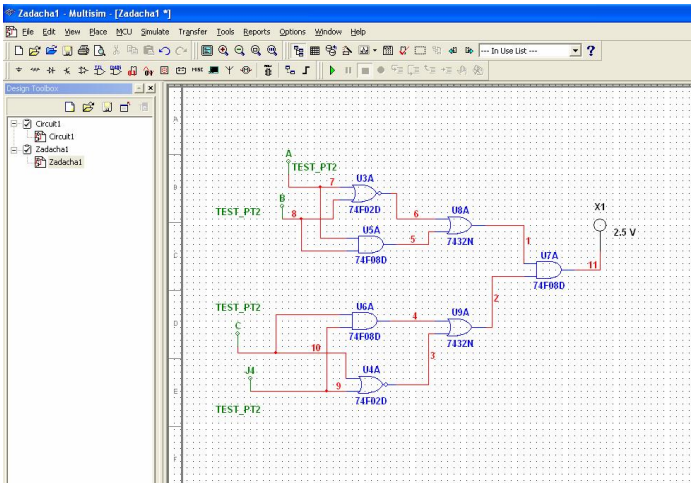
Продовження таблиці 4.2

	*Уведення і перетворення таблиці істинності*. Для того, щоб створити таблицю істинності, необхідно в лівій верхній частині приладу обрати число змінних від **A** до **H** (натискувати на відповідну букву лівою кнопкою миші). Уся ліва половина екрана заповниться комбінаціями нулів і одиниць, які визначають початкові вхідні умови. Трохи правіше розташований стовпець вихідних значень **OUT** (реакція на вхід), заповнений первинно нулями. Змінюючи в правій колонці нулі на одиниці або **X** (байдужий стан), можна описати стан виходу для будь-якої початкової умови. Після клацання на кнопці в нижньому рядку зображення приладу з'явиться логічний вираз.
	*Спрощення виразу Булевої алгебри*. Якщо таблиця істинності містить велике число змінних, то логічний вираз функції виходить громіздким. Для його перетворення в компактну форму слід натискувати цю кнопку.
	*Уведення і перетворення логічного виразу*. Для отримання таблиці істинності функції, заданої логічним виразом, необхідно: а) у рядок перетворювача ввести за допомогою клавіатури логічний вираз; б) натискувати кнопку. При введенні виразів інверсія позначається апострофом, логічне складання «+». Логічне множення не позначається.
	*Синтез схеми по логічному виразу*. За допомогою логічного перетворювача можна одержати схему, що реалізовує функцію, задану логічним виразом. Для цього в нижньому рядку перетворювача слід увести логічний вираз і натискувати кнопку. Після натиснення цієї кнопки на робочому полі EWB з'явиться еквівалентна логічному виразу схема. Усі елементи в схемі будуть виділені червоним кольором.

Експеримент 1. Знайдіть аналітичний вираз функції, яка реалізується схемою, наведеною на рис. 4.2. Зберіть схему в *Electronics Workbench* (рис. 4.2 а) або *Mulsim* (рис. 4.2 б), підключіть входи *D, C, B, A* до джерела логічних сигналів, а вихід – до логічного пробника. Включіть схему й перевірте правильність аналітичного виразу.



а)



б)

Рис. 4.2. Схема до експерименту №1

Експеримент 2. Викличте генератор слів і логічний аналізатор. Запрограмуйте генератор на формування послідовності чотирьох розрядних слів, що відповідають числам натурального ряду від 0 до 15. Підключіть його виходи до відповідних входів схеми, наведеної на рис. 4.2. (*A* – молодший розряд числа, *D* – старший). Досліджуйте роботу схеми в режимах *„STEP”* та *„CYCLE”*.

Намалюйте тимчасові діаграми сигналів на виходах усіх логічних елементів схеми для всіх можливих комбінацій вхідних сигналів. Перевірте правильність виконання завдання за допомогою логічного аналізатора.

Експеримент 3. Проведіть аналіз роботи схеми, зображеної на рис. 4.3, для чого складіть таблиці реалізованих функцій, якщо сигнал у точці 1 сприймається елементом **„АБО”**:

а) як логічна 1;

б) як логічний 0.

Оберіть необхідні інструменти для проведення експериментальної перевірки схеми і визначте, як сприймається сигнал на непідключеному вході при роботі базових елементів.

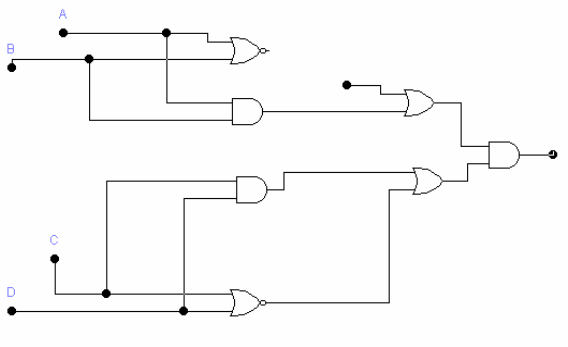


Рис. 4.3. Схема до експерименту № 3

Експеримент 4. Проведіть аналіз роботи логічного пристрою, зібраного на мікросхемах *7404* і *7410*, наведенного на рис. 4.4. Визначте, яку математичну операцію виконує цей пристрій, якщо комбінації логічних рівнів на вході розглядати як

числа. Зберіть схему, підключіть необхідні прилади та проведіть експериментальне дослідження роботи схеми.

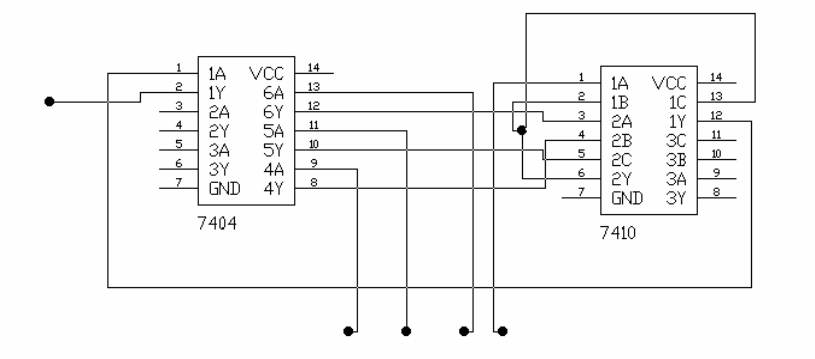
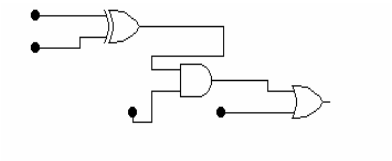
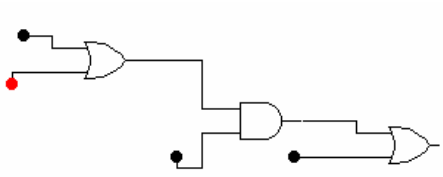


Рис. 4.4. Схема до експерименту № 4

Експеримент 4. При монтажі схеми, наведеної на рис. 4.5 а, було допущено помилку: замість елемента **XOR** був використаний елемент **OR** (рис. 4.5 б). Знайдіть комбінації вхідних сигналів, які дозволяють виявити помилку монтажника.



а)



б)

Рис. 4.5. Схема до експерименту № 5

Експеримент 5. Спростіть схему, наведену на рис. 4.6. Перевірте еквівалентність сигналів на виході попередньої схеми та отриманої при спрощенні.

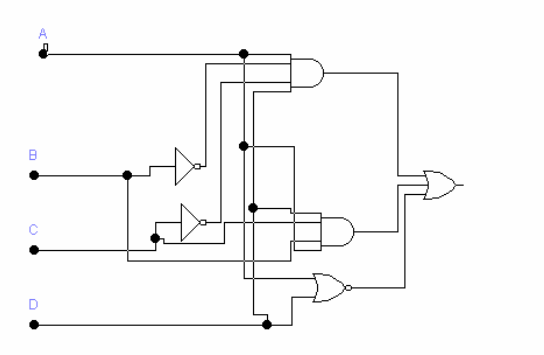


Рис. 4.6. Схема до експерименту № 6

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Скласти в *Electronics Workbench* або *Mulsim* відповідні схеми.
4. Провести відповідні експерименти. Заповнити протокол та скласти звіт про виконувану роботу.
5. Відповідно до отриманих виразів у лабораторній роботі № 3 побудувати відповідні схеми у *EWB* або *Mulsim.*
6. Зробити висновки щодо роботи.
7. Відповісти на питання для самоперевірки.

Питання для самоперевірки

1. Чим відрізняється робота у *EWB* та *Mulsim*?
2. За допомогою яких вікон або панелей побудувати комбінаційну схему у *EWB* та *Mulsim*?
3. Як перевірити працездатність схеми в *EWB* або *Mulsim*?
4. За допомогою яких інструментів можна спростити схему в *EWB* або *Mulsim*?

Лабораторна робота № 5.
Проектування комбінаційної схеми перетворювача кодів

Мета роботи: спроектувати перетворювач двійково-десяткового коду в спеціальний семи розрядний код, який керує роботою семиелементного цифрового індикатора.

Загальні теоретичні відомості

Принципи побудови комбінаційної схеми розглянемо на прикладі перетворювача двійково-десяткового коду в спеціальний семи розрядний код, який керує роботою семиелементного цифрового індикатора, схема включення якого наведена на рис. 1.1. Підсвічування кожного з його елементів, що утворюють цифри від 0 до 9, здійснюється подачею одиничного сигналу на вхід, що відповідає цьому елементу. Сигнали надходять на цифровий індикатор з виходів перетворювача кодів. На входи перетворювача подаються логічні сигнали чотирьохрозрядних двійково-десяткових кодів. Завдання полягає в синтезі такої логічної схеми перетворювача, яка сформує на індикаторі зображення цифр, що відповідають вхідним двійково-десятковим кодам (див.: рис. 5.1).

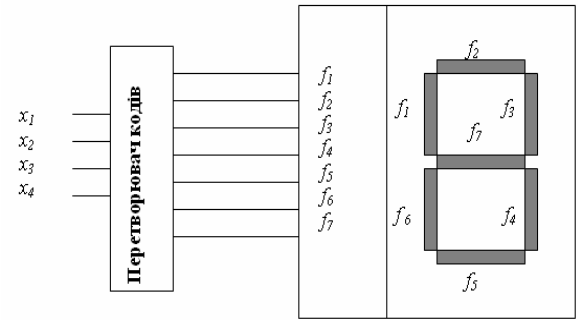


Рис. 5.1. Схема включення перетворювача кодів

Вихідні логічні змінні $f_1...,f_7$ є функціями вхідних змінних x_1 ..., x_4. Таблиця істинності цих логічних функцій представлена в

таблиці 5.1. Одиничні значення вихідних змінних відповідають елементам, що підсвічуються при зображенні цифри. Наприклад, десяткова цифра 0 задається двійково-десятковим кодом 0000. При цьому підсвічуються всі елементи, за винятком f_7. Цифра 7 задається кодом 0111, а підсвічуються елементи f_2, f_3 і f_4. Логічні функції в таблиці визначено не повністю: останні шість кодових комбінацій не відповідають ніяким десятковим цифрам і є забороненими.

Приклад виконання. Зобразимо карту Карно для логічної функції f_1, (див.: рис. 5.2) а. Осередки, у яких функція не визначена, відзначимо знаком «*» і до визначимо їх одиницями. Покриття містить чотири прямокутники. Отже, мінімізована ДНФ цієї функції повинна містити чотири елементарні кон'юнкції.

Аналогічно, що визначивши одиницями всі невизначені значення функцій f_2 і f_3, отримаємо для них якнайкращі покриття і мінімальні диз'юнктивні форми (див.: рис. 5.2 б, в та табл.5.2).

Таблиця 5.1

Таблиця істинності логічних функцій перетворювача кодів

Цифра	x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	0	1	1	0	0	0
2	0	0	1	0	0	1	1	0	1	1	1
3	0	0	1	1	0	1	1	1	1	0	1
4	0	1	0	0	1	0	1	1	0	0	1
5	0	1	0	1	1	1	0	1	1	0	1
6	0	1	1	0	1	1	0	1	1	1	1
7	0	1	1	1	0	1	1	1	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	1	0	1
-	1	0	1	0	*	*	*	*	*	*	*
-	1	0	1	1	*	*	*	*	*	*	*
-	1	1	0	0	*	*	*	*	*	*	*
-	1	1	0	1	*	*	*	*	*	*	*
-	1	1	1	0	*	*	*	*	*	*	*
-	1	1	1	1	*	*	*	*	*	*	*

Завдання

Кожну з розглянутих вище функцій реалізувати в тому або іншому елементному базисі. Побудувати комбінаційні схеми реалізації функції f_1 у базисі елементів **„І”, „АБО”, „НІ”**, а також - реалізації цієї ж функції в базисі елементів „**І - НІ”** (Відповідна логічна формула встановлюється застосуванням закону інверсії:

$$f_1 = \overline{\overline{x_1 \vee x_2 \cdot \overline{x_4} \vee x_2 \cdot \overline{x_3} \vee \overline{x_3} \cdot \overline{x_4}}} = \overline{\overline{x}_1 \cdot \overline{x_2 \overline{x}_4} \, \overline{x_2 \overline{x}_3} \, \overline{\overline{x}_3 \overline{x}_4}}\ ).$$

При мінімізації функції f_4 гніздо з координатами 1010 довизначити нулем (формула мінімізованої функції при цьому істотно спроститься).

При мінімізації функції f_6 чотири гнізда карти Карно довизначити нулями.

Формули для решти функцій (f_i) отримують аналогічно. Ці формули, залежні від змінних x_1 ..., x_4, утворюють систему логічних функцій цих змінних.

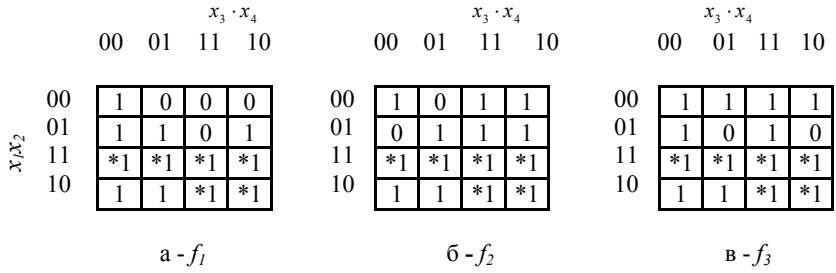


Рис. 5.2. Карти Карно функцій f_1, f_2, f_3

Таблиця 5.2

Таблиця логічних функцій f_1, f_2, f_3 перетворювача кодів

f_1	$f_1 = x_1 \vee x_2 \cdot \overline{x}_4 \vee x_2 \cdot \overline{x}_3 \vee \overline{x}_3 \cdot \overline{x}_4$
f_2	$f_2 = x_1 \vee x_3 \vee x_2 \cdot x_4 \vee \overline{x}_2 \cdot \overline{x}_4$
f_3	$f_3 = x_1 \vee x_2 \vee x_3 \cdot x_4 \vee \overline{x}_3 \cdot \overline{x}_4$

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Розробити відповідну схему для кожного сегмента перетворювача кодів. Перевірити схему в EWB.
4. Скласти звіт з лабораторної роботи.
5. Зробити висновки щодо роботи.
6. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. Схарактеризуйте методики мінімізації логічних виразів.
2. Дайте визначення нормальної диз'юнктивної форми.
3. Сформулюйте правило запису досконалої нормальної диз'юнктивної форми.
4. Назвіть основні відмінні особливості логічних змінних.
5. Які способи представлення логічних функцій?
6. Якщо в таблиці істинності логічної функції **„І"** двох змінних *(A, B)* значення всіх елементів не інверсні, то яка вийде логічна функція двох змінних?
7. Запишіть довільну кон'юнкцію для логічної функції чотирьох змінних.
8. Назвіть переваги мінімізації за допомогою карт Карно.
9. Скільки гнізд містить карта Карно для логічної функції п'яти змінних?
10. Яким критерієм оцінюється якість мінімізації логічної функції?
11. Яка відмінність карт Карно та діаграм Вейча?
12. Сформулюйте теорему Квайна.
13. Які закони алгебри логіки використовують для мінімізації логічних функцій?
14. У чому полягає метод мінімізації Квайна-Мак-Класкі?

Лабораторна робота № 6. Експериментальні дослідження тригерів у програмах Electronics Workbench та Multisim

Мета роботи: навчитися моделювати роботу цифрових пристроїв з елементами пам'яті в програмах Electronics Workbench та Multisim.

Загальні теоретичні відомості

Тригер має два стійкі стани: *Q=1* і *Q=0*, тому його іноді називають бістабільною схемою. У якому з цих станів опиниться тригер, залежить від сигналів на входах тригера і від його попереднього стану, тобто він має пам'ять. Можна сказати, що тригер є елементарним елементом пам'яті. Тип тригера визначається алгоритмом його роботи. Залежно від алгоритму роботи тригер може мати настановнчі, інформаційні й керівні входи. Настановні входи встановлюють стан тригера незалежно від стану інших входів. Входи управління дозволяють запис даних, що подаються на інформаційні входи. Найпоширенішими є тригери **RS**, **JK**, **D** і **T**- типів [5,с.6].

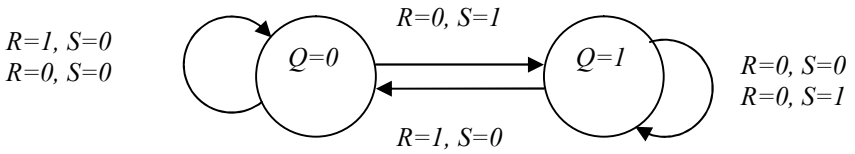
1. Тригер типу RS

RS-тригер – найпростіший автомат з пам'яттю, який може перебувати в двох станах. Тригер має два входи: установки S (**SET** – установка) і скидання R (**RESET** – скидання), на які подаються вхідні сигнали від зовнішніх джерел. При подачі на вхід установки активного логічного рівня тригер установлюється в 1 ($Q = 1, \quad \overline{Q} = 0$), при подачі активного рівня на вхід скидання тригер установлюється в 0 ($Q = 0, \quad \overline{Q} = 1$). Якщо подати на обидва входи установки (збудження) пасивний рівень, то тригер зберігатиме попередній стан виходів: $Q = 0 \quad (\overline{Q} = 1) \quad$ *або* $\quad Q = 1 \quad (\overline{Q} = 0)$.

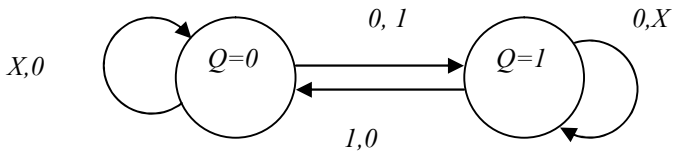
Опис роботи **RS**-тригера можна доповнити графом (рис. 6.1) (графічний спосіб).

Графік на рис. 6.1 а показує, що схема, яка знаходилася в стані *Q=0*, зберігає цей стан як при дії вхідного набору *R=0*, *S=0*, так і при дії *R=1*, *S=0*. Якщо ж на вхід схеми, що перебуває в стані *Q=0*, подіяти набором *R=0*, *S=1*, то вона переходить в стан *Q=1* і зберігає його при вхідних наборах *R=0*, *S=1*, або *R=0*, *S=0*.

На рис. 6.1 б той же граф тригера намальований більш компактно. Вхідні сигнали, які можуть приймати будь-які значення (як *0*, так і *1*), позначені як *X*, а позиція позначення відповідає послідовності *R*, *S*.



а)



б)

Рис. 6.1. Граф роботи RS-тригера

2. JK- тригер

Тригер JK-типу має складнішу, порівняно з RS-тригером, структуру й більш широкі функціональні можливості. Крім інформаційних входів *J* і *K* і прямого та інверсного виходів, JK-тригер має вхід управління *C* (цей вхід також називають тактуючим або рахунковим), а також асинхронні *R* і *S* – входи. Входи *R* і *S* мають пріоритет над іншими. Активний рівень сигналу

на вході *S* установлює тригер у стан 0=1, а активний рівень сигналу на вході R – у стан *Q=0*, незалежно від сигналів на решті входів.

3. D-тригер.

D-тригер має один інформаційний вхід *D* (data - дані). Інформація з входу *D* заноситься в тригер по позитивному перепаду імпульсу на рахунковому вході *C* і зберігається до наступного позитивного перепаду на рахунковому вході тригера. Крім рахункового *C* й інформаційного *D* входів, тригер забезпечений асинхронними *R* і *S* входами. На рис. 6.2 зображено граф роботи D тригера [7].

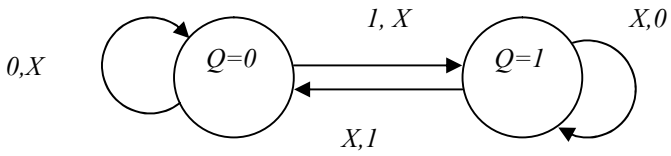


Рис. 6.2. Граф роботи D-тригера

Експеримент 1. Зберіть схему, зображену на рис. 6.3. Включіть схему. Послідовно подайте на схему такі сигнали: *S=1, R=0; S=0, R=0*. Поясніть отримані результати.

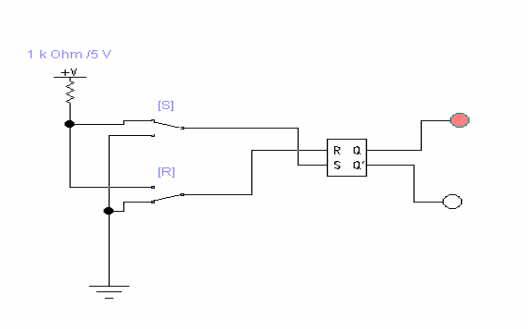


Рис. 6.3. Дослідження RS-тригера

Експеримент 2. Дослідження **JK**-тригера, побудованого на базі логічних елементів **„І” RS**-тригерів.

Зберіть схему, зображену на рис. 6.4. Включіть схему. Змінюючи рівень сигналу на вході *C*, складіть тимчасові діаграми сигналів на виходах *Q1* і *Q2* обох **RS**-тригерів і замалюйте їх у розділі „Результати експериментів”. Вкажіть режим роботи тригера. Визначте моменти зміни сигналів *Q1* і *Q2* стосовно до моментів зміни сигналу *C*. Відобразіть відмінність у перемикання **RS**-тригерів на діаграмах.

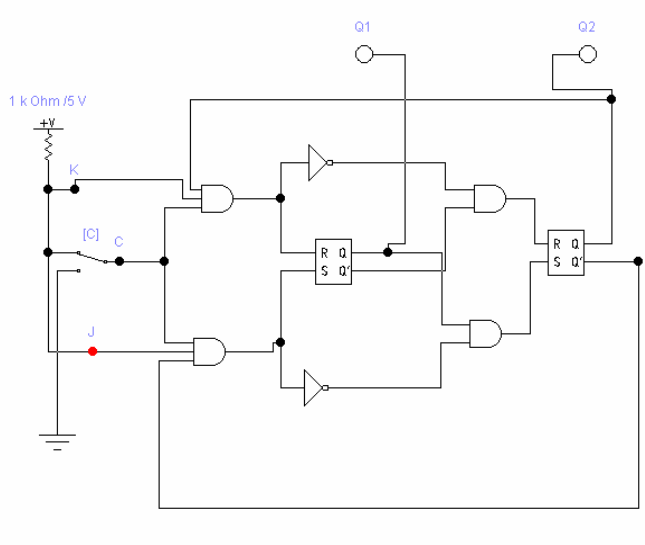


Рис. 6.4. Дослідження *JK*-тригера, побудованого на базі логічних елементів „І” *RS*-тригерів

Експеримент 3. Дослідження роботи **D**-тригера в рахунковому режимі.

Зберіть схему, зображену на рис. 6.5. Подаючи на рахунковий вхід *C* тактові імпульси за допомогою ключа *[C]* і визначаючи стан виходів тригера за допомогою пробників складіть тимчасові діаграми роботи тригера в рахунковому режимі й занесіть їх у розділ „Результати експериментів”.

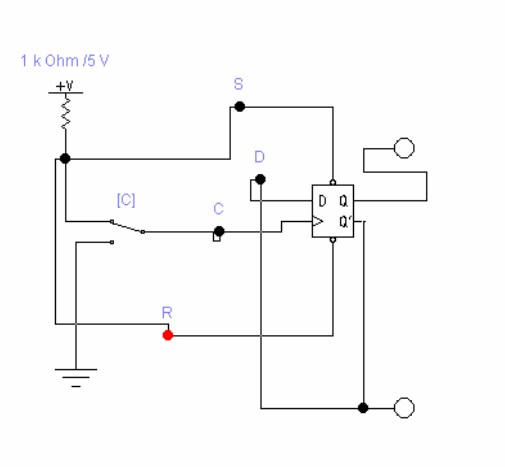


Рис. 6.5. Дослідження роботи D-тригера в рахунковому режимі

Експеримент 6. Дослідження **JK**-тригера в рахунковому режимі (**T**-тригер). Зберіть схему, зображену на рис. 6.6. Включіть схему. Змінюючи стан входу *C* відповідним ключем, замалюйте в розділі „Результати експериментів” діаграми роботи тригера в рахунковому режимі.

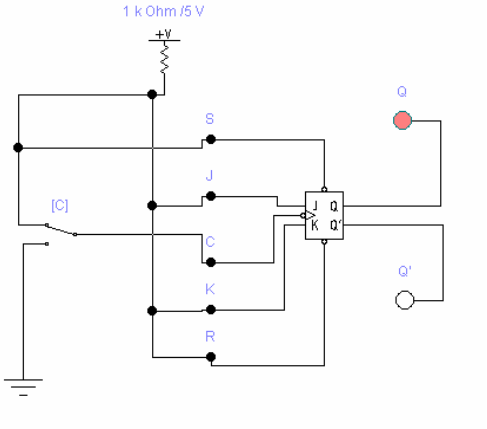


Рис. 6.6. Дослідження роботи JK-тригера в рахунковому режимі

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Скласти в *Electronics Workbench* або *Mulsim* відповідні схеми.
4. Провести відповідні експерименти. Заповнити протокол та скласти звіт з лабораторної роботи.
5. Підключити до схем осцилограф та замалювати у звіті відповідні тимчасові діаграми роботи тригерів.
6. Зробити висновки щодо виконаної роботи.
7. Відповісти на питання для самоперевірки.

Питання для самоперевірки

1. Чим відрізняється робота **RS**-тригера з прямими входами від роботи **RS**-тригера з інверсними входами?
2. Чому комбінація сигналів *11* на входах **RS**-тригера називається „забороненою”?
3. У чому відмінність таблиці переходів тригера від таблиці функцій збудження?
4. Як властивість запам’ятовування відображається в характеристичних рівняннях тригерів?

5. У чому принципова відмінність роботи синхронних тригерів від асинхронних?
6. Яка пріоритетність інформаційних і настановчих входів у синхронних тригерах?
7. Чому **JK**-тригер при *J=K=1* не перетворюється на автогенератор?
8. Чому **T**-тригер одержав назву рахункового? Яке число імпульсів він може злічити?
9. Як працює **D**-тригер, якщо *D=Q*?
10. Намалюйте граф-схему роботи **JK**-тригера.
11. Намалюйте тимчасові діаграми роботи відповідно **D**, **RS** та **JK** тригерів.

Лабораторна робота № 7. Методика проектування автомата з пам'яттю

Мета роботи: *опанувати методику проектування автоматів із елементами.*

Загальні теоретичні відомості

Проектування автоматів із пам'яттю (АП) містить такі етапи:

- початкове завдання функціонування;
- формалізоване завдання функціонування;
- мінімізація станів;
- кодування станів;
- складання таблиці переходів;
- визначення функцій збудження елементів пам'яті (тригерів);
- мінімізація функцій збудження тригерів;
- перехід до базису обраної для реалізації схемотехнології;
- складання логічної схеми;
- збірка й перевірка автомата.

Початкове завдання функціонування може мати різну форму, зокрема й словесну. Від неї переходять до формалізованого завдання – таблиць, формул, діаграм стану й т.ін. Далі виконується мінімізація і кодування станів автомата, унаслідок чого виходить таблиця переходів, на підставі якої можна знайти функції збудження тригерів.

Мінімізація і кодування станів у загальному випадку є завданням, вирішення якого може зажадати значних зусиль, проте при проектуванні вузлів ЕОМ і цифрової автоматики воно найчастіше за все є простим, і його рішення підказується самим формулюванням завдання на проектування. Традиційно *широко застосовується кодування станів автомата двійковими кодами,* при якому тригери використовуються у схемі економно.

Функції збудження тригерів, що забезпечують переходи АП з одного стану в інший, реалізуються його комбінаційною частиною.

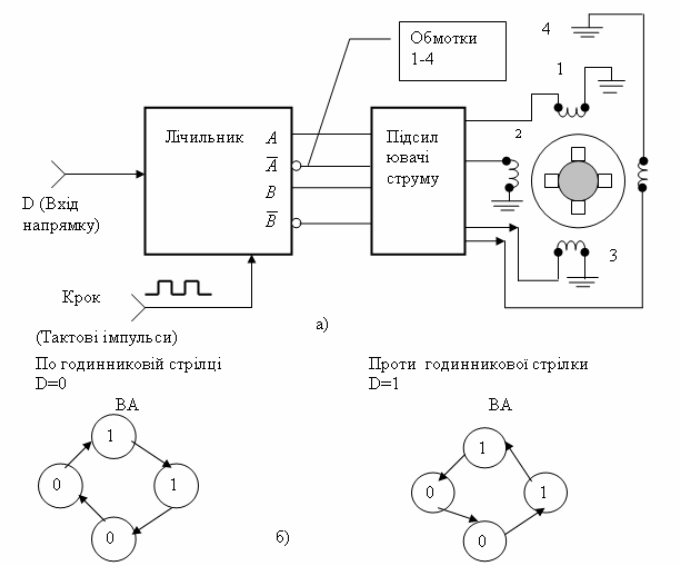
Вони, як сказано в переліку етапів проектування, мінімізуються і переводяться в базис обраних засобів реалізації автомата. Це положення слід розуміти в широкому значенні, оскільки залежно від засобів реалізації КС вимоги до форм представлення функцій збудження можуть істотно розрізнятися. Точніше, можна говорити про приведення функцій до вигляду, зручного для відтворення цими засобами.

Приклад 1. Крековий двигун постійно обертається по кроках (15° за крок). Котушка електромагніту або обмотка всередині двигуна повинна підключатися і відключатися в особливій послідовності, яка й організовуватиме покрокову роботу. Звичайно для управління струмом в обмотках двигуна застосовують цифрові сигнали. Крокові двигуни часто використовуються у випадках, коли потрібна точна установка частин приладу в певну точку, наприклад при установці головки зчитування (записи на магнітних дисках, керуванні друкуючою головкою принтера, автоматах і т.ін.).

На рис. 7.1 а), представлена схема типового крокового двигуна з чотирма обмотками. Для правильного обертання обмотки 1 і 2 повинні бути завжди в протилежних станах. Це значить, що коли на обмотку 1 подано живлення, то обмотка 2 відключена, і навпаки. Так само обмотки 3 і 4 також повинні бути завжди в протилежних станах. Для управління струмом чотирьох обмоток використовуються виходи двохрозрядного синхронного лічильника. Виходами A і $\overline{A}$ управляються обмотки 1 і 2, а виходами B і $\overline{B}$ – обмотки 3 і 4. Оскільки з виходів тригерів не видаються значення струму, що необхідно для безпосереднього керування обмотками, то між виходами тригерів і обмотками потрібно встановити підсилювачі струму.

Таким чином, цей крековий двигун може обертатися як по годинниковій (CW), так і проти годинникової стрілки (CCW), для чого є вхід напряму D, який керує напрямом обертання. На рис. 7.1 б) діаграма станів показує два випадки. Для обертання за годинниковою стрілкою потрібен стан D = 0, а стани лічильника BA повинні відповідати рахунковій послідовності: 11, 10, 00, 01, 11, 10 ... і т.ін., неначебто лічильник тактувався сигналом з входу *Step*. Для обертання

проти годинникової стрілки необхідний стан входу D = 1, і лічильник повинен працювати в такій рахунковій послідовності: 11, 01, 00, 10, 11, 01 і т.ін.



а) Синхронний лічильник, який за допомогою рахункової послідовності керує кроковим двигуном; б) діаграми переходів станів для обох станів входу напряму D

Рис. 7.1. Схема крокового двигуна

Розглянемо більш детально методику проектування автоматів, що містять тригери.

У тактованих автоматах елементами пам'яті слугують синхронні тригери, причому будь-який автомат можна побудувати на будь-якому типі тригера (**D, JK, RS, T** та ін.).

При двійковому кодуванні станів автомата число тригерів у його схемі дорівнює $n = \lceil \log_2 N \rceil$, де N – число станів автомата, $\lceil \rceil$ – знак округлення до найближчого справа цілого числа.

При кодуванні кодом „1 з N" число тригерів дорівнює числу станів автомата: n=N, оскільки кожному стану відповідає один тригер в одиничному стані при нульовому стані інших.

Вважатимемо, що закон функціонування автомата визначений, і кодування його станів проведено. Значить, відома послідовність станів тригерів, що приймаються ними в кожному такті під управлінням вхідних сигналів x_1, x_2,…,$x_к$ і поточного стану Q_1, Q_2,..., Q_n. Предмет синтезу – отримання функцій збудження φ_i і ψ_i для кожного входу всіх тригерів, що забезпечують необхідні переходи автомата.

Функції виходу для автоматів Мура залежать тільки від стану автомата, тому знаходження виходів Z_1, Z_2,..., Z_j здійснюється комбінаційною схемою, на яку подаються тільки виходи тригерів Q_1, Q_2,…Q_n. Усі тригери тактуються загальним синхросигналом С. Після завершення вироблення функцій збудження комбінаційною схемою надходить черговий тактовий сигнал С, що переводить тригери в новий стан.

Вид функцій збудження залежить від логічного типу тригерів. Тому одним із засобів синтезу слугують „словники" для тригерів.

При пошуку функцій збудження тригерів спочатку складається таблиця, що містить такі дані:

Таблиця 7.1

Таблиця станів тригера

Входи в момент часу t				**Стани тригерів**								**Необхідні сигнали на всіх входах кожного тригера**				
				Q (старе)				***$Q_н$ (нове)***								
x_1	x_2	.	x_k	Q_1	Q_2	.	Q_n	Q_1	Q_2	.	Q_n	φ_1	ψ_1	.	φ_n	ψ_n

Стовпці φ_i, ψ_i,…, φ_n, ψ_n визначають функції збудження тригерів.

Багатоваріантність реалізацій автомата пов'язана з вибором типу тригерів і комбінаційної частини.

Щодо найпоширеніших типів тригерів **JK** і **D** можна відзначити таке. Тригер типу **JK** характеризується більш розвиненими логічними функціями, тому для нього функції збудження в середньому більш прості, але число їх удвічі більше,

ніж для тригера **D.** Що ж дасть більш просте рішення, наперед невідомо.

Комбінаційна частина автомата може бути побудована на логічних елементах, мультиплексорах, ІС програмованій пам'яті, програмованих логічних матрицях і т.ін. Стани автомата можна кодувати двійковими кодами, кодами „1 з N" та ін.

Автомат можна побудувати, пристосувавши до необхідного функціонування типову ІС середнього рівня інтеграції (лічильник, що зсовує *j* регістр), додавши до них спеціально спроектовану логічну частину.

Необхідно спроектувати автомат з двома режимами роботи, керований вхідним сигналом М. При М=0 автомат працює як двійковий лічильник з модулем рахунку 8, при М = 1 як лічильник у коді Грея.

Код Грея використовується в системах контролю ЦА, перетворювачах механічних переміщень у цифровий код і т.ін. При переході від попередньої кодової комбінації до наступної в коді Грея змінюється тільки один розряд. Перші вісім комбінацій коду Грея наведено в таблиці 7.2.

Таблиця 7.2

Десяткова цифра		Код Грея	
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Кодування станів автомата, що є автоматом Мура, визначається власне постановкою завдання. Діаграма станів автомата показана на рис 7.2. Зміна керівного сигналу *M* відразу ж веде до зміни режиму, тобто наступний стан належатиме вже іншому коду.

Автомат побудований на тригерах **JK** і логічних елементах **І-НЕ.** Таблиця 7.3 переходів автомата, відповідає діаграмі його станів.

Синтез функцій збудження для автоматів з тригерами **JK** має цікаву особливість. Вони можуть бути одержані не тільки вказаним на лекціях шляхом, а й без пошуку в таблицях функцій *J* і *K*, стовпці яких можна в цьому випадку виключити.

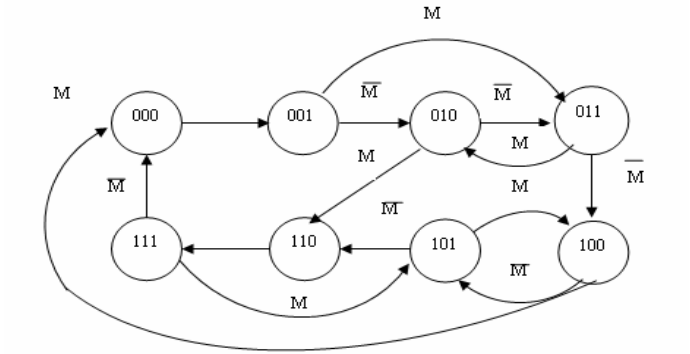


Рис. 7.2. Діаграма станів автомата

Таблиця 7.3

Таблиця переходів автомата

Вхідний керуючий сигнал	**Початковий стан**			**Новий стан**			**Вхідний керуючий сигнал**	**Початковий стан**			**Початковий стан**		
M	Q_2	Q_1	Q_0	$Q_{2н}$	$Q_{1н}$	$Q_{0н}$	*M*	Q_2	Q_1	Q_0	$Q_{2н}$	$Q_{1н}$	$Q_{0н}$
0	0	0	0	0	0	1	1	0	0	0	0	0	1
0	0	0	1	0	1	0	1	0	0	1	0	1	1
0	0	1	0	0	1	1	1	0	1	0	1	1	0
0	0	1	1	1	0	0	1	0	1	1	0	1	0
0	1	0	0	1	0	1	1	1	0	0	0	0	0
0	1	0	1	1	1	0	1	1	0	1	1	0	0
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	0	0	0	1	1	1	1	1	0	1

З даних про функціонування автомата можна одержати функцію переходів кожного тригера

$Q_{ні}=F(x_1, x_2, \ldots, x_k, Q_1, Q_2, \ldots, Q_n)$.

Цю функцію можна розкласти таким чином

$$Q_{ні} = f_i \overline{Q_i} \vee g_i Q_i , \qquad (7.1)$$

де функції f і g уже не містять відповідно змінних Qi та $\overline{Q_i}$.

Характеристичне рівняння тригера типа **JK** має вигляд:

$$Q_{ні} = J_i \overline{Q_i} \vee \overline{K_i} Q_i , \qquad (7.2)$$

Порівнявши вирази (3.1) і (3.2), одержимо $f_i = J_i$, $\overline{g_i} = K_i$.

Отже, поклавши у функції переходів тригера $Q_i=0$, відразу одержуємо функцію збудження для входу J_i:

$$Q_{ні}\Big|_{Q_i=0} = f_i = J_i ,$$

а прийнявши умову $Q = 1$, можна одержати функцію збудження для входу K_i:

$$Q_{ні}\Big|_{Q_i=1} = g_i = \overline{K_i} \text{ , т. ч. } \overline{K_i} = g_i$$

При цьому члени формул для $Q_{н.і}$, що не містять $\overline{Q_i}$, *та* Q_i, перетворяться шляхом множення на розширену форму $Q_i \vee \overline{Q_i} = 1$, у якій усі доданки містять змінні $\overline{Q_i}$ або Q_i.

У результаті отримаємо такі вирази:

$$J_2 = \overline{M}Q_1Q_0 \vee M\overline{Q_1}\overline{Q_0} = \overline{\overline{\overline{M}Q_1Q_0} \bullet \overline{M\overline{Q_1}\overline{Q_0}}};$$

$$K_2 = \overline{M}Q_1Q_0 \vee M\overline{Q_1}\overline{Q_0} = \overline{\overline{\overline{M}Q_1Q_0} \bullet \overline{M\overline{Q_1}\overline{Q_0}}};$$

$$J_1 = \overline{M}Q_0 \vee \overline{Q_2}Q_0 = \overline{\overline{\overline{M}Q_0} \bullet \overline{\overline{Q_2}Q_0}};$$

$$K_1 = \overline{M}Q_0 \vee Q_2Q_0 = \overline{\overline{\overline{M}Q_0} \bullet \overline{Q_2Q_0}};$$

$$J_0 = \overline{M} \vee \overline{Q_2}\overline{Q_1} \vee Q_2Q_1 = \overline{M \bullet \overline{\overline{Q_2}\overline{Q_1}} \bullet \overline{Q_2Q_1}};$$

$$K_0 = \overline{M} \vee \overline{Q_2}Q_1 \vee Q_2\overline{Q_1} = \overline{M \bullet \overline{\overline{Q_2}Q_1} \bullet \overline{Q_2\overline{Q_1}}}.$$

Таким чином, у результаті отримаємо схему пристрою, яка показана на рис. 7.3.

Далі потрібно реалізувати схему в Electronics Workbench або Multisim.

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Розробити відповідну схему. Перевірити схему в EWB.
4. Скласти звіт з лабораторної роботи.
5. Зробити висновки щодо виконаної роботи.
6. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. Назвіть етапи проектування АП.
2. Як визначити кількість тригерів, необхідних у АП?
3. Як здійснюється початкове завдання функціонування АП?
4. Як здійснюється кодування станів АП?

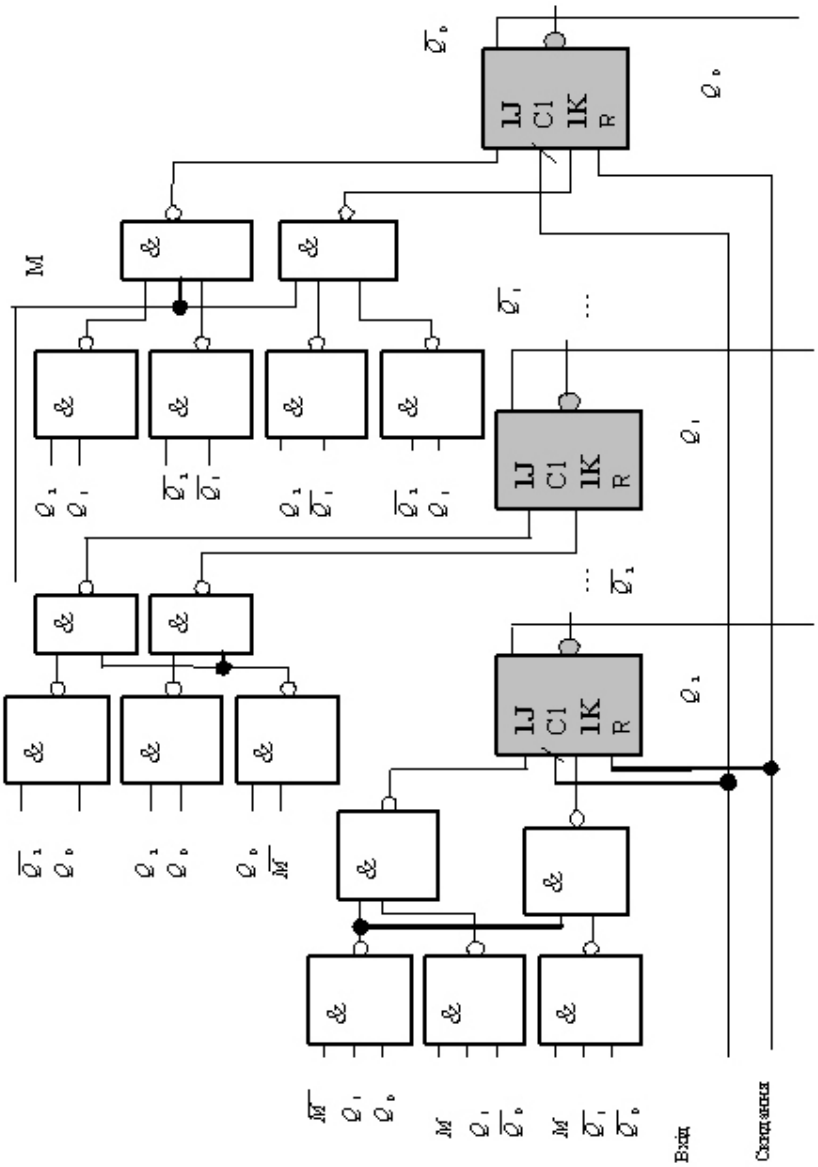


Рис. 7.3. Отримана схема ЦА

Модуль 4 Основи алгебри логіки. Мінімізація логічних функцій.

Лабораторна робота № 8. Експериментальні дослідження лічильників та регістрів у програмах Electronics Workbench та Multisim

Мета роботи: навчитися моделювати роботу лічильників у програмах Electronics Workbench та Multisim.

Загальні теоретичні відомості

Лічильник – пристрій для підрахунку числа вхідних імпульсів. Число, що представляється станом його виходів по фронту кожного вхідного імпульсу, змінюється на одиницю. Лічильник можна реалізувати на кількох тригерах. У лічильниках, що підсумовують, кожний вхідний імпульс збільшує число на його виході на одиницю, у піднімальних лічильниках кожний вхідний імпульс зменшує це число на одиницю. Найпростіші лічильники – двійкові. При цьому тригери з'єднуються послідовно. Вихід кожного тригера безпосередньо діє на тактовий вхід наступного. Число, утворюване станом інверсних виходів тригерів лічильника, пов'язане з числом освіченим станом прямих виходів тригерів таким співвідношенням [8]:

$$N_{пр} = 2^n - N_{інв} - 1,$$

де n – розрядність виходу лічильника.

Лічильники характеризуються числом станів протягом одного періоду (циклу). Число станів називають коефіцієнтом перерахунку $K_{пер}$, який дорівнює відношенню числа імпульсів N_c на вході до числа імпульсів $N_{Q_{ст}}$ на виході старшого розряду за період:

$$K_{пер} = \frac{N_c}{N_{Q_{ст}}}.$$

Для побудови підсумовуючого лічильника із $K_{пер} = 5$ треба, щоб після формування останнього числа з послідовності *{0, 1, 2, 3, 4}* лічильник переходив не до числа 5, а до числа 0. У двійковому коді це означає, що від числа 100 потрібно перейти до числа 000, а не 101. Зміна природного порядку рахунку можлива при введенні додаткових зв'язків між тригерами лічильника. Можна скористатися таким способом: як тільки лічильник потрапляє в неробочий стан (у цьому випадку 101), цей факт повинен бути впізнаний і спричинити подальше вироблення сигналу, який перевів би лічильник у стан 000. Розглянемо цей спосіб більш детально. Факт попадання лічильника в неробочий стан описується логічним рівнянням:

$$F = (101) \vee (110) \vee (111) = Q_3 \cdot \overline{Q_2} \cdot Q_1 \vee Q_3 \cdot Q_2 \cdot \overline{Q_1} \vee Q_3 \cdot Q_2 \cdot Q_1 =$$
$$= Q_3 \cdot Q_1 \vee Q_3 \cdot Q_2.$$

Стани 110 і 111 також є неробочими й тому враховані при складанні рівняння. Якщо на виході еквівалентної логічної схеми *F=0*, це означає, що лічильник знаходиться в одному з робочих станів: $0 \vee 1 \vee 2 \vee 3 \vee 4$. Як тільки він потрапляє в один з неробочих станів $5 \vee 6 \vee 7$, формується сигнал *F=1*. Поява сигналу *F=1* повинна переводити лічильник в початковий стан 000. Отже, цей сигнал потрібно використовувати для дії на входи тригерів лічильника, які здійснювали б скидання лічильника в стан *Q1=Q2=Q3=0*. При реалізації лічильника на тригерах з входами установки логічним нулем для скидання тригерів вимагається подати на входи скидання сигнал *R=0*. Для виявлення факту потрапляння в неробочий стан використовуємо схему, що реалізовує функцію *F* і виконану на елементах **„І-НЕ"**. Для цього спростимо вираз для функції *F*:

$$F = \overline{Q_3 \cdot Q_1 \vee Q_3 \cdot Q_2} = \overline{Q_3 \cdot (\overline{Q}_1 \vee \overline{Q}_2)}.$$

Відповідну схемну реалізацію наведено на рис. 8.1.

Лічильник працюватиме таким чином: при рахунку від 0 до 4 усе відбувається як у звичайному підсумовуючому лічильнику, із $K_{пер}=8$. Установлюючи сигнали дорівнюють одиниці та порядку рахунку {0, 1, 2, 3, 4} не перешкоджають. Рахунок здійснюється по позитивному фронту імпульсу на рахунковому вході *C*. У той момент, коли лічильник знаходиться в стані 4 (100), наступний тактовий імпульс спочатку переводить лічильник у стан 5 (101), що негайно (задовго до приходу наступного тактового імпульсу) приводить до формування сигналу скидання, який надходить на встановлюючи входи скидання тригерів. У результаті лічильник скидається в 0 і чекає приходу наступного тактового імпульсу на рахунковий вхід.

Один цикл рахунку закінчився, лічильник готовий до початку наступного циклу. Застосовуючи такі схеми зі зворотним зв'язком для скидання лічильника, потрібно мати на увазі, що операція скидання займає кінцевий час, тому безпосередньо перед скиданням лічильника в 0 на виході першого тригера з'являються короткочасні імпульси. Це не має значення при підключенні лічильника напряму до індикатора, але при використовуванні цього виходу лічильника як джерела тактових імпульсів можуть виникнути певні проблеми. Схему, на якій це явище усунено, наведено на рис. 8.2. Важливою відмінністю є те, що схема знаходить не факт попадання в неробочий стан 101, а факт попадання в стан 100 і в наступному такті виробляє сигнал скидання.

Експеримент 1. Дослідження підсумовуючого лічильника.

Зберіть схему, зображену на рис. 8.3. Включіть схему. Подаючи на вхід схеми тактові імпульси за допомогою ключа *C* і спостерігаючи стан виходів лічильника за допомогою логічних пробників, складіть тимчасові діаграми роботи лічильника, що підсумовує. Визначте коефіцієнт перерахунку лічильника.

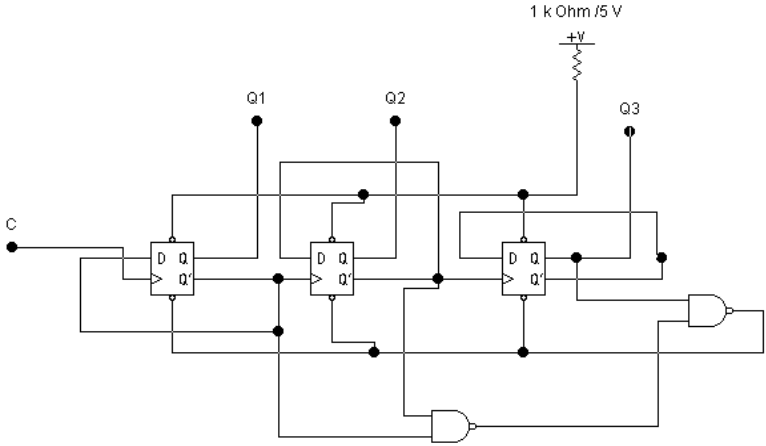


Рис. 8.1. Схема підсумовуючого лічильника

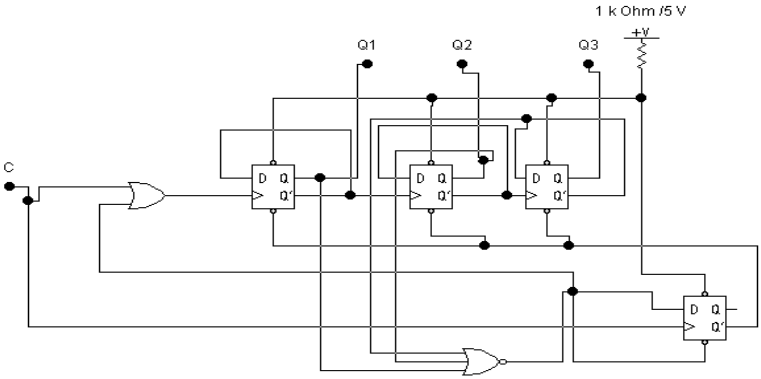


Рис. 8.2. Модернізована схема підсумовуючого лічильника

Результати занесіть в розділ "Результати експериментів". Зверніть увагу на числа, формовані станами інверсних виходів лічильника.

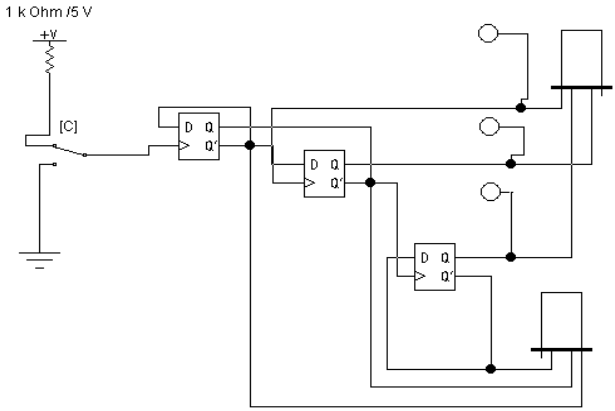


Рис. 8.3. Схема підсумовуючого лічильника

Експеримент 2. Дослідження відіймаючого лічильника.

Зберіть схему, зображену на рис. 8.4. Включіть схему. Замалюйте тимчасові діаграми роботи відіймаючого лічильника в розділ „Результати експериментів". У схемі на рис. 8.4 входи логічного аналізатора підключіть до інверсних входів тригерів. Включіть схему. Замалюйте одержані тимчасові діаграми в розділ „Результати експериментів" і порівняйте їх з діаграмами, одержаними в попередньому експерименті.

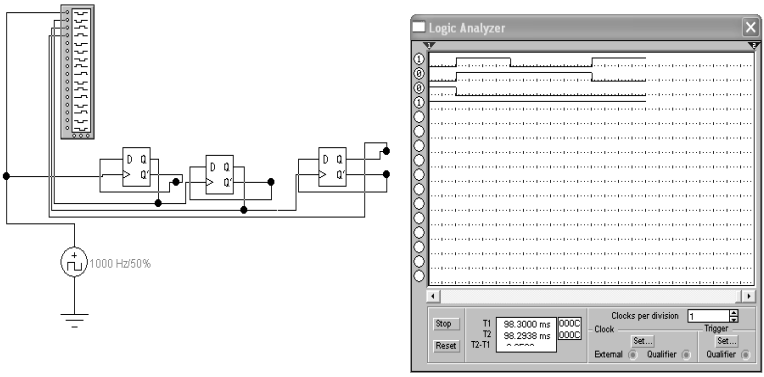


Рис. 8.4. Схема відіймаючого лічильника

Експеримент 3. Дослідження лічильника зі зміненим коефіцієнтом перерахунку.

Зберіть схему, зображену на рис. 8.5. Включіть схему. Подаючи на вхід схеми тактові імпульси за допомогою ключа *C* і спостерігаючи стан виходів лічильника за допомогою логічних пробників, складіть тимчасові діаграми роботи лічильника і визначте коефіцієнт перерахунку. Результати занесіть у розділ „Результати експериментів”.

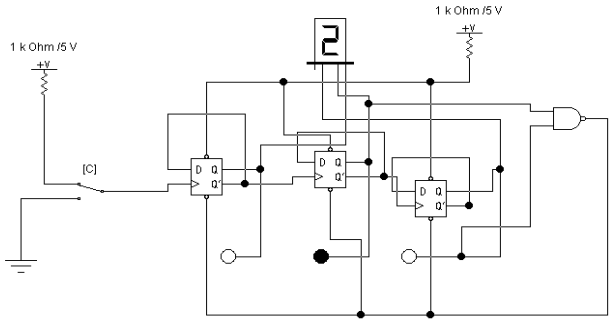


Рис. 8.5. Схема лічильника зі зміненим коефіцієнтом перерахунку

Експеримент 4. Дослідження лічильника, побудованого на **JK** – тригерах.

Зберіть схему, зображену на рис. 8.6. Включіть схему. Лічильник виконаний на трьох **JK**-тригерах у рахунковому режимі (на *J*- та *K*-входи подані сигнали 1). Для забезпечення коефіцієнта рахунку $K_{пер}= 6$ використаний зворотний зв’язок на елементі **„І”** (*AND_1*), при цьому сигнал 1 з його виходу через елемент **„АБО”** (*OR_1*) надходить на *R*-входи тригерів, переводячи їх у нульовий стан. До другого входу елемента **„АБО”** (*OR_1*) підключений вхід *R* для подачі зовнішнього сигналу скидання.

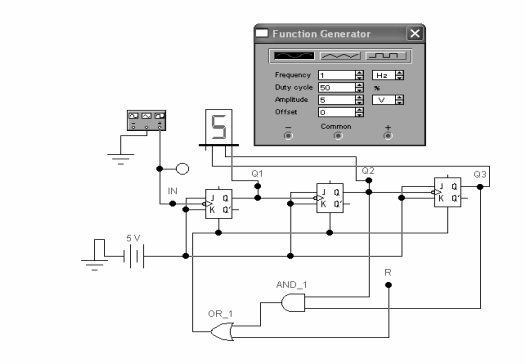


Рис. 8.6. Дослідження лічильника на JK-тригерах

Спроектуйте лічильник із коефіцієнтом рахунку $K_{пер} = 9$.

Експеримент 5. Дослідження регістра Джонсона, реалізованого на **JK**-тригерах.

Зберіть схему, зображену на рис. 8.7. Включіть схему. Встановіть ключ *S* у верхнє положення (на вхід *S* другого тригера подається сигнал логічної одиниці). Побудуйте тимчасові діаграми роботи схеми й занесіть їх у розділ „Результати експериментів". Встановіть схему в стан 000. Подайте за допомогою ключа *S* короткочасний імпульс на вхід *S* другого тригера. При цьому схема повинна встановитися в стан 010. Подаючи на вхід *C* схеми тактові імпульси за допомогою відповідного ключа та спостерігаючи стан виходів схеми за допомогою логічних пробників, складіть тимчасові діаграми роботи пристрою. Визначте коефіцієнт перерахунку схеми. Результати занесіть у розділ „Результати експериментів".

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Скласти в *Electronics Workbench* або *Mulsim* відповідні схеми до експериментів.
4. Провести відповідні експерименти. Заповнити протокол та скласти звіт з роботи.
5. Зробити висновки щодо виконаної роботи.
6. Відповісти на питання для самоперевірки.

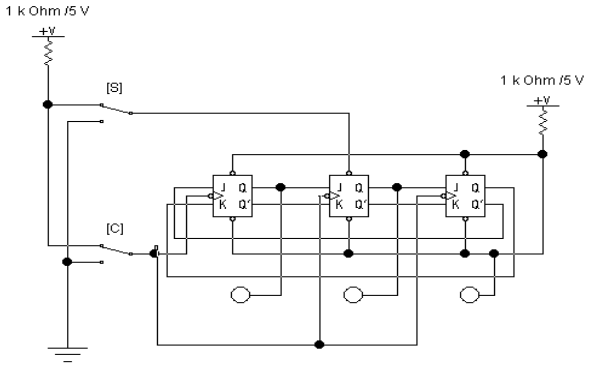


Рис. 8.7. Дослідження регістра Джонсона, реалізованого на JK-тригерах

Питання для самоперевірки

1. Чому при підключенні рахункових входів тригерів до інверсних виходів попередніх каскадів лічильник на *D*-тригерах працює, якщо підсумовує, а при підключенні до прямих - як віднімаючий?

2. У якому режимі працюватиме лічильник на *JK*-тригерах при підключенні рахункових входів тригерів до прямих виходів попередніх каскадів? Як зміниться режим роботи лічильника при підключенні рахункових входів тригерів до інверсних виходів?

3. Який коефіцієнт перерахунку має регістр Джонсона?

4. Яким чином можна змінити коефіцієнт перерахунку лічильника?

5. У двійковому лічильнику коефіцієнт перерахунку дорівнює 8, кількість тригерів – 3. Під час подачі тактових імпульсів на рахунковий вхід лічильник змінює свій стан у такій послідовності: *000-001-010-011-100-101-110-111-000*. Скільки тригерів у лічильнику змінюють свій стан одночасно на кожному з переходів? Чи дійсно тригери змінюють свій стан одночасно? Як відбувається перехід лічильника із стану *111* в стан *000*? Який з тригерів першим змінить свій стан? Що стане причиною перемикання другого тригера? Як розвивається процес зміни стану тригерів під час переходу лічильника із стану *011* у стан *100*?

Лабораторна робота № 9. Проектування цифрового автомата зі змінним коефіцієнтом рахунку лічильника

Мета роботи: *опанувати методику проектування лічильників із змінним коефіцієнтом рахунку.*

Загальні теоретичні відомості

В окремих пристроях вимагається змінювати коефіцієнт рахунку лічильників. Один з можливих способів його зміни полягає в зміні логічної структури схеми залежно від сигналів управління коефіцієнтом рахунку.

Порядок роботи лічильників задається таблицею, що відображає стан $Q_n^+,...,Q_o^+$, у які переходять тригери залежно від попередніх станів $Q_{n,...,}Q_0$ та сигналів $X_{1,...,}X_m$.

Приклад. Необхідно спроектувати паралельний лічильник, який веде рахунок по модулю 5 або по модулю 7. Коефіцієнт рахунку визначається сигналом *X:* при *X=0* рахунок здійснюється по модулю 5, при *X=1* – по модулю 7.

Мінімальна кількість тригерів – три. (Лічильник, який містить три тригери, може перебувати в одному з восьми станів (включно стан 000). Початкові дані задано в таблиці 9.1. У кожному її рядку зазначено значення вхідної змінної *X*, стани тригерів Q_2, Q_1, Q_0, до подачі наступного синхронізуючого імпульсу та стани Q_2^+, Q_1^+, Q_0^+ , у котрі повинні перейти тригери після закінчення цього імпульсу.

Для Т-тригерів $T_0 = f_0 (X, Q_2, Q_1, Q_0)$; $T_1 = f_1 (X, Q_2, Q_1, Q_0)$; $T_2 = f_2 (X, Q_2, Q_1, Q_0)$.

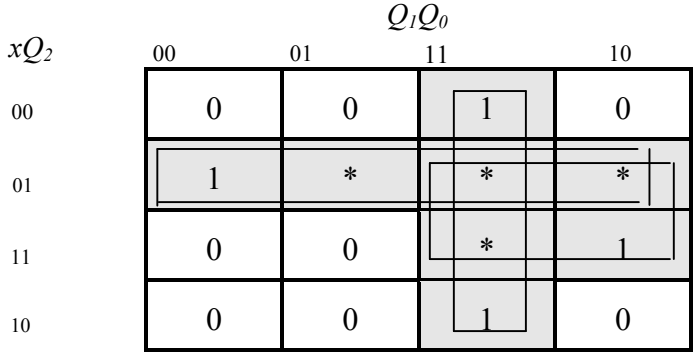
Побудуємо схему лічильника, використовуючи Т-тригери.

Таблиця 9.1

Початкові дані

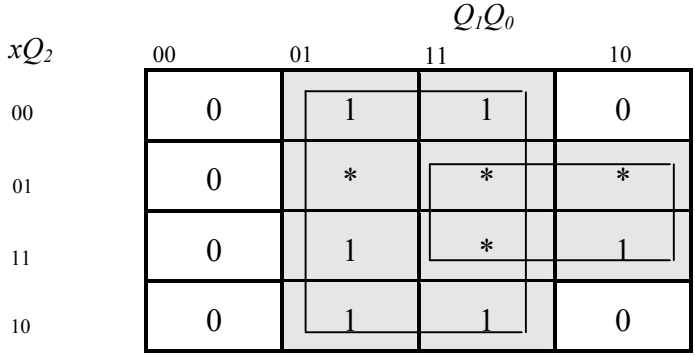
№ п/п	X	**Стани**						**Функція збудження тригерів**		
		Попередні			**Наступні**			T		
		Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	T_2	T_1	T_0
0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	1	0	1	0	0	1	1
2	0	0	1	0	0	1	1	0	0	1
3	0	0	1	1	1	0	0	1	1	1
4	0	1	0	0	0	0	0	1	0	0
5	0	1	0	1	*	*	*	*	*	*
6	0	1	1	0	*	*	*	*	*	*
7	0	1	1	1	*	*	*	*	*	*
8	1	0	0	0	0	0	1	0	0	1
9	1	0	0	1	0	1	0	0	1	1
10	1	0	1	0	0	1	1	0	0	1
11	1	0	1	1	1	0	0	1	1	1
12	1	1	0	0	1	0	1	0	0	1
13	1	1	0	1	1	1	0	0	1	1
14	1	1	1	0	0	0	0	1	1	0
15	1	1	1	1	*	*	*	*	*	*

На рис. 9.1.– 9.3 наведено Карти Карно



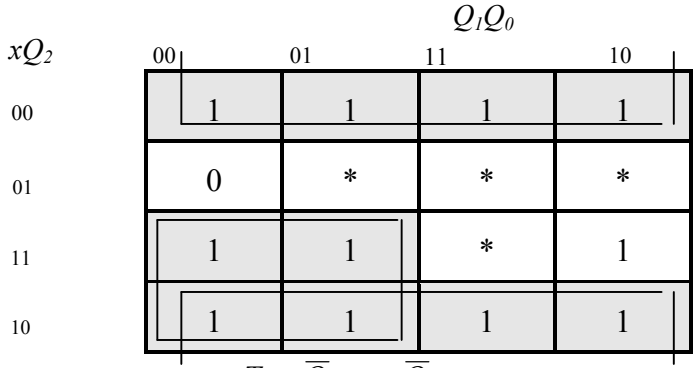
$$T_2 = Q_1 \cdot Q_2 \vee \bar{x} \cdot Q_2 \vee Q_1 \cdot Q_0$$

Рис. 9.1. Карта Карно для входу T_2



$$T_1 = Q_0 \vee Q_2 \cdot Q_1$$

Рис. 9.2. Карта Карно для входу T_1



$$T_0 = \overline{Q_2} \vee x \cdot \overline{Q_1}$$

Рис. 9.3. Карта Карно для входу T_0

За результатами мінімізації отримаємо таку схему лічильника (див.: рис. 9.4). Далі потрібно скласти схему в Electronics Workbench або Multisim та перевірити її роботу.

Приклади схем лічильників із змінним коефіцієнтом рахунку наведено на див. рис. 9.5 а (3 або 8) та б (4 або 5).

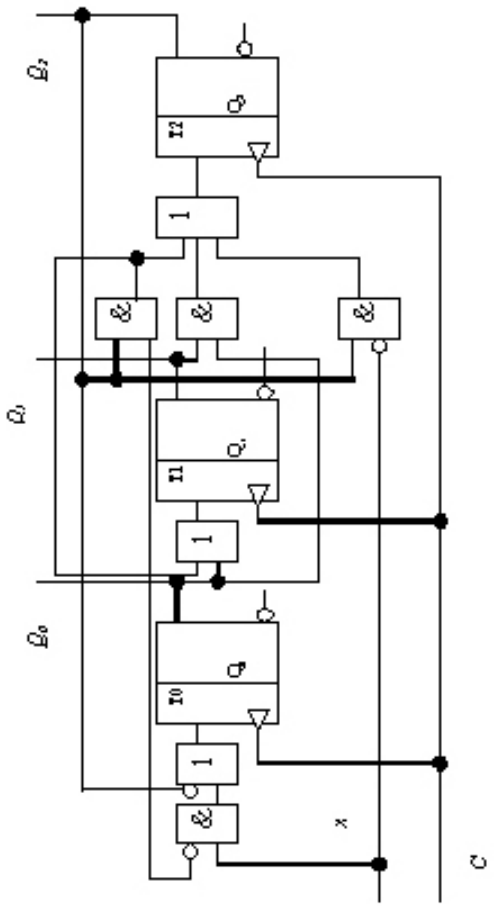
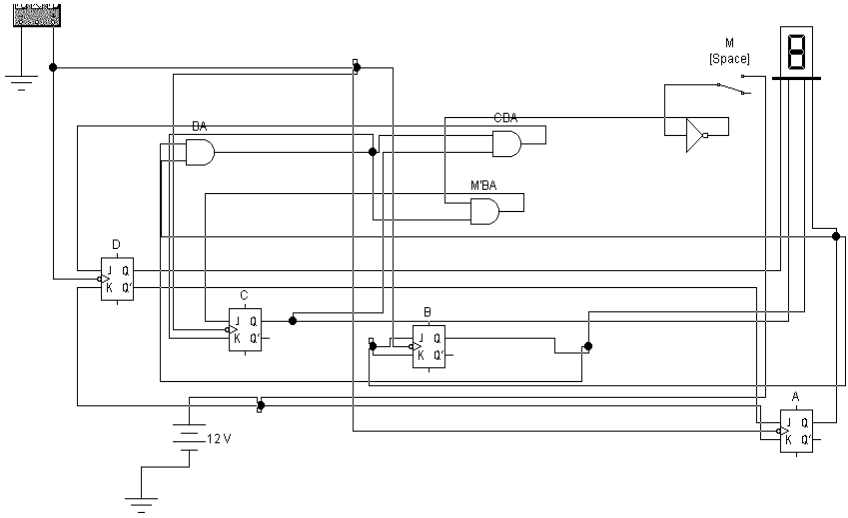


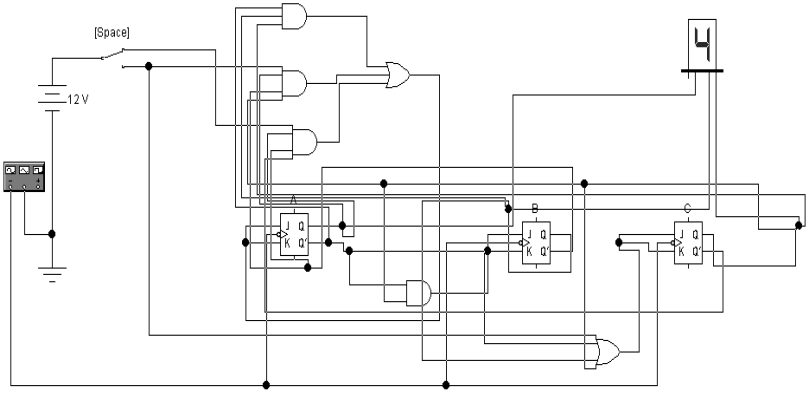
Рис. 9.4. Схема лічильника з коефіцієнтами рахунку по модулях 5 та 7 на базі Т -тригерів

Лічильник на рис. 9.5 а працює від 0 до 8. У такому режимі він буде працювати до команди „СТОП” або до натискання перемикача (клавіша „Space”), тоді лічильник працює від 0 до 3.

Лічильник на рис. 9.5 б працює від 0 до 4. У такому режимі він буде працювати до команди „СТОП” або до натискання перемикача (клавіша „Space”), тоді лічильник працює від 0 до 5.



а) Коефіцієнт рахунку 3 або 8



б) Коефіцієнт рахунку 4 або 5

Рис. 9.5. Приклади реалізації лічильника зі змінним коефіцієнтом рахунку

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.

3. Обрати варіант відповідно до таблиці 9.2 та спроектувати схему лічильника зі змінним коефіцієнтом рахунку.
4. Скласти в *Electronics Workbench* або *Mulsim* відповідні схеми та перевірити їх працездатність.
5. Скласти звіт щодо виконаної роботи.

Таблиця 9.2

Початкові дані

№ варианту	Керуючий сигнал X	*Значення коефіцієнтів рахунку*
1	X =0, X =1	K=5, K=4
2	X =0, X =1	K=5, K=8
3	X =0, X =1	K=8, K=3
4	X =0, X =1	K=7, K=4
5	X =0, X =1	K=6, K=5
6	X =0, X =1	K=3, K=8
7	X =0, X =1	K=7, K=6
8	X =1, X =0	K=8, K=7
9	X =0, X =1	K=6, K=4
10	X =0, X =1	K=5, K=4
11	X =1, X =0	K=8, K=5
12	X =0, X =1	K=7, K=4
13	X =1, X =0	K=6, K=7
14	X =0, X =1	K=2, K=8
15	X =1, X =0	K=7, K=6
16	X =0, X =1	K=2, K=7
17	X =0, X =1	Інші за вибором викладача

Питання для самоперевірки

1. Чим відрізняється методика проектування лічильника зі змінним коефіцієнтом рахунку?

2. Наведіть приклади цифрових пристроїв зі змінним коефіцієнтом рахунку.

Лабораторна робота № 10. Експериментальні дослідження дешифраторів та мультиплексорів у програмах Electronics Workbench та Multisim

Мета роботи: навчитися моделювати роботу дешифраторів та мультиплексорів у програмах Electronics Workbench та Multisim.

Загальні теоретичні відомості

Дешифратор – логічна комбінаційна схема, яка має n інформаційних входів і 2^n виходів. Кожній комбінації логічних рівнів на входах відповідатиме активний рівень на одному з 2^n виходів. Звичайно n дорівнює 2,3 або 4.

На рис. 10.1 зображений дешифратор з $n=3$, активним рівнем є рівень логічного нуля. На входи *C, B, A* можна подати такі комбінації логічних рівнів: 000, 001, 010...111, усього 8 комбінацій. Схема має 8 виходів, на одному з яких формується низький потенціал, на інших – високий. Номер цього єдиного виходу, на якому формується активний (нульовий) рівень, відповідає числу N, визначуваному станом входів *C, B, A* таким чином: $N=C\cdot 2^2 + B\cdot 2^1 + A\cdot 2^0$. Наприклад, якщо на входи подана комбінація логічних рівнів 011, то з восьми виходів мікросхеми (*Y0, Y1...Y7*) на виході з номером $N=3$ установиться нульовий рівень сигналу ($Y3=0$), а вся решта виходів матиме рівень логічної одиниці. Цей принцип формування вихідного сигналу можна описати таким чином [9,10]:

$$Y_i = \begin{cases} 0, & якщо \quad i = k; \\ 1, & якщо \quad i \neq k; \end{cases}$$
$$k = 2^2 \cdot C + 2^1 \cdot B + 2^0 \cdot A,$$

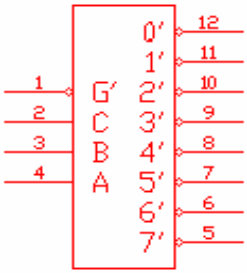


Рис. 10.1. Схема дешифратора з $n=3$

Можна записати вирази для кожного виходу дешифратора:

$$Y0 = \overline{\overline{C} \cdot \overline{B} \cdot \overline{A}}, \qquad Y4 = \overline{C \cdot \overline{B} \cdot \overline{A}},$$
$$Y1 = \overline{\overline{C} \cdot \overline{B} \cdot A}, \qquad Y5 = \overline{C \cdot \overline{B} \cdot A},$$
$$Y2 = \overline{\overline{C} \cdot B \cdot \overline{A}}, \qquad Y6 = \overline{C \cdot B \cdot \overline{A}},$$
$$Y3 = \overline{\overline{C} \cdot B \cdot A}, \qquad Y7 = \overline{C \cdot B \cdot A}.$$

Крім інформаційних входів *A,B,C* дешифратори мають додаткові входи управління *G*. Сигнали на цих входах, наприклад, дозволяють функціонування дешифратора або переводять його в пасивний стан, при якому, незалежно від сигналів на інформаційних входах, на всіх виходах установиться рівень логічної одиниці. Можна сказати, що існує деяка функція дозволу, значення якої визначається станами керуючих входів. Дозвільний вхід дешифратора може бути прямим або інверсним. У дешифраторів з прямим дозвільним входом активним рівнем є рівень логічної одиниці, у дешифраторів з інверсним входом – рівень логічного нуля. На рис. 10.1 представлений дешифратор з одним інверсним входом керування.

У дешифратора з кількома входами управління функція дозволу, як правило, є логічним множенням усіх дозвільних сигналів управління. Наприклад, для дешифратора 74138 з одним прямим входом управління *G1* і двома інверсними *G2A* і *G2B* (рис. 10.2) функції виходу Y_i і дозволу *G* мають вигляд:

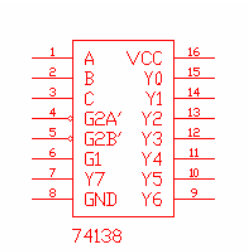


Рис. 10.2. Схема дешифратора 74138

$$Y_i = \begin{cases} \overline{1 \cdot \overline{G}}, & якщо \quad i = k; \\ 1, & якщо \quad i \neq k; \\ k = 2^2 \cdot C + 2^1 \cdot B + 2^0 \cdot A, \end{cases}$$

$$G = G1 \cdot \overline{G2A} \cdot \overline{G2B}.$$

Входи керування використовуються для збільшення розрядності дешифраторів або при паралельній роботі кількох схем на загальній вихідній лінії.

Використовування дешифратора як демультиплексора. Дешифратор може бути використаний і як демультиплексор – логічний комутатор, що підключає вхідний сигнал до одного з виходів. У цьому випадку функцію інформаційного входу виконує один з входів дозволу, а стан входів *C, B* і *A* задає номер виходу, на який передається сигнал з входу дозволу. Порядок проведення експериментів такий:

Експеримент 1. Дослідження принципу роботи дешифратора в основному режимі. Побудуйте схему, зображену на рис. 10.3. Включіть схему. Подайте на вхід *G* рівень логічної одиниці. Для цього клавішею *G* ключ *G* установити у верхнє положення. Визначте й запишіть рівні сигналів на виходах *Y0...Y7* у таблицю істинності при *G=1* (табл. 10.1 у розділі „Результати експериментів"). Подайте на вхід *G* рівень логічного нуля (ключ *G* установите в нижнє положення). Переконайтеся, що дешифратор перейшов у робочий режим і на одному з виходів установився рівень логічного нуля. Подаючи всі можливі комбінації рівнів логічних сигналів на входи *A, B, C* за допомогою однойменних ключів і визначаючи за допомогою логічних пробників рівні логічних сигналів на виході схеми, заповніть таблицю істинності дешифратора при *G=0* (табл. 10.1. у розділі „Результати експериментів").

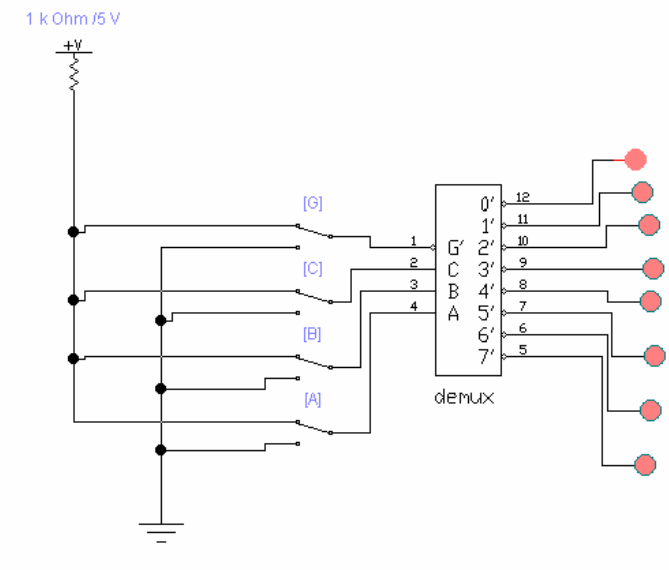


Рис. 10.3. Схема дослідження дешифратора

Таблиця 10.1

Експеримент 1. Результати експериментів.
Дослідження роботи дешифратора в основному режимі

C	*B*	*A*	*G*	*Y0*	*Y1*	*Y2*	*Y3*	*Y4*	*Y5*	*Y6*	*Y7*
0	0	0	1								
0	1	1	1								
0	0	1	0								
0	1	0	0								
0	1	1	0								
1	0	0	0								
1	0	1	0								
1	1	0	0								
1	1	1	0								

Експеримент 2. Дослідження роботи дешифратора як демультиплексора. Побудуйте схему, зображену на рис. 10.4.

Включіть схему. У покроковому режимі роботи генератора слів подайте на входи *C, B, A* демультиплексора слова, еквівалентні числам від 0 до 7. Спостерігаючи за допомогою логічних пробників рівні сигналів на виходах, заповніть таблицю функціонування (табл. 10.2 у розділі „Результати експериментів”). Переконайтеся, що сигнал, що змінюється, на вході *G* по черзі з’являється на виходах дешифратора.

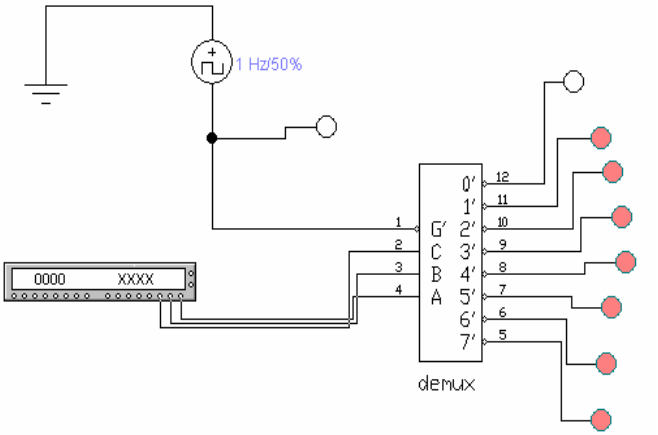


Рис. 10.4. Схема дослідження дешифратора як демультиплексора

Таблиця 10.2

Експеримент 2. Результати експериментів.
Дослідження роботи дешифратора як демультиплексора

C	*B*	*A*	*Y0*	*Y1*	*Y2*	*Y3*	*Y4*	*Y5*	*Y6*	*Y7*
0	0	0								
0	0	1								
0	1	0								
0	0	1								
1	0	1								
1	0	1								
1	0	1								
1	0	1								

Мультиплексори

Мультиплексор – комбінаційна логічна схема, яка керується перемикачем, який підключає до виходу один з інформаційних входів даних. Номер входу, що підключається, дорівнює числу (адресі), визначуваному комбінацією логічних рівнів на входах управління. Окрім інформаційних і керуючих входів, схеми мультиплексорів містять вхід дозволу, при подачі на який активного рівня мультиплексор переходить в активний стан. При подачі на вхід дозволу пасивного рівня мультиплексор перейде в пасивний стан, для якого сигнал на виході зберігає постійне значення незалежно від значень інформаційних і керуючих сигналів. Число інформаційних входів у мультиплексорів звичайне 2, 4, 8 або 16. На рис. 10.5 представлений мультиплексор з інверсним входом дозволу *G*, прямим *У* і інверсним *W*-виходами $W = \overline{Y}$.

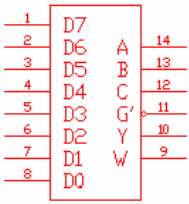


Рис. 10.5. Схема мультиплексора

Функціонування мультиплексора, представленого на рис. 10.5, описується характеристичним рівнянням, що зв'язує сигнал на виході (*У*) з дозволяючим (*G*), вхідними інформаційними (*DO...D7*) і керуючими (*A, B, C*) сигналами:

$$Y = \left(\begin{array}{l} \overline{C} \cdot \overline{B} \cdot \overline{A} \cdot D0 \vee \overline{C} \cdot \overline{B} \cdot A \cdot D1 \vee \overline{C} \cdot B \cdot \overline{A} \cdot D2 \vee \overline{C} \cdot B \cdot A \cdot D3 \vee \\ C \cdot \overline{B} \cdot \overline{A} \cdot D4 \vee C \cdot \overline{B} \cdot A \cdot D5 \vee C \cdot B \cdot \overline{A} \cdot D6 \vee C \cdot B \cdot A \cdot D7 \end{array} \right) \cdot \overline{G}.$$

Логічна функція *n* змінних визначена для 2^n комбінація значень змінних. Кожній комбінації значень аргументів відповідає єдиний інформаційний вхід мультиплексора, на який подається значення функції. Наприклад, вимагається реалізувати функцію

$$F1 = \overline{C} \cdot \overline{B} \cdot \overline{A} \vee C \cdot B \cdot A \vee C \cdot B \cdot \overline{A} \vee \overline{C} \cdot B \cdot A .$$

Ця функція визначена тільки для 8 комбінацій значень змінних, тому для її реалізації пропонується використовувати мультиплексор з трьома керуючими входами. Складемо таблицю істинності функції:

Таблиця 10.3

Таблиця істинності.

N	***C***	***B***	***A***	***F1***
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

З таблиці видно, що для реалізації функції на мультиплексорі необхідно подати на інформаційний вхід мультиплексора з номером *N* сигнал, значення якого дорівнює відповідному значенню функції *F1*, тобто на входи з номерами *1, 2, 4, 5* слід подати рівень логічного нуля, а на інші – рівень логічної одиниці. Таким чином, при подачі комбінації логічних рівнів на керуючі входи мультиплексора, до його виходу підключиться вхід, значення сигналу на якому дорівнює відповідному значенню функції. Схемна реалізація наведена на рис. 10.6. При реалізації логічних функцій на інформаційні входи можна подавати не тільки константи, але і вхідні сигнали.

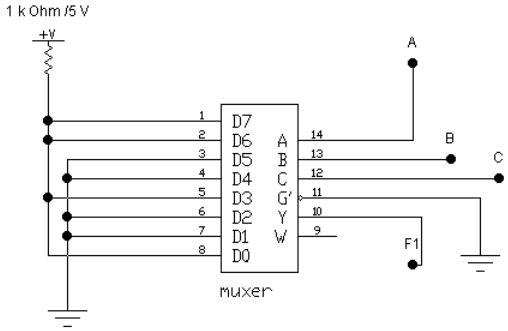


Рис. 10.6. Схема реалізації функцій мультиплексора

Експеримент 3. Дослідження мультиплексора.

Побудуйте схему, зображену на. рис. 10.7. Включіть схему. За допомогою ключа *G* установіть на вході *G* мультиплексора рівень логічного нуля. По черзі подаючи всі можливі комбінації логічних рівнів за допомогою ключів *A, B, C* на відповідні входи мультиплексора, для кожної комбінації за допомогою логічних пробників визначте, перемикання якого з ключів у лівій частині схеми змінює стан виходів мультиплексора. Позначення відповідного входу мультиплексора запишіть в таблицю 10.4 у розділі „Результати експериментів", зазначивши при цьому, як передається вхідний сигнал на виходи мультиплексора (напряму або з інверсією). Наприклад, якщо перемикання ключа 4 змінює стан виходів мультиплексора, у таблиці в рядку з відповідною комбінацією рівнів сигналів на входах *A, B, C* слід записати для виходу *Y - D4*, для виходу *W - D4*. Установіть за допомогою ключа *G* рівень логічної одиниці на вході *G* мікросхеми. У розділ „Результати експериментів" запишіть позначення висновків, які при перемиканні відповідних ключів у лівій частині схеми не впливають на стан виходів мікросхеми.

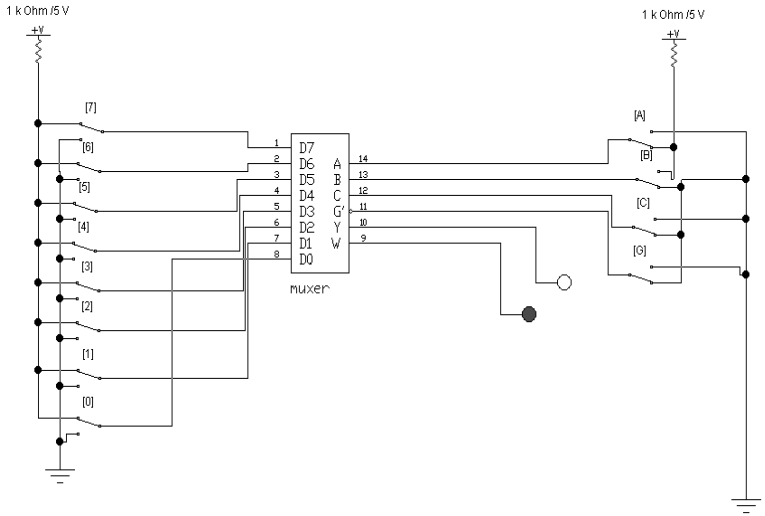


Рис. 10.7. Схема для дослідження мультиплексора

Таблиця 10.4

Експеримент 3. Результати експериментів.
Дослідження роботи мультиплексора

A	***B***	***C***	***Y***	***W***
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Скласти в *Electronics Workbench* або *Mulsim* відповідні схеми та провести експерименти. Заповнити протокол та скласти звіт щодо виконаної роботи.

4. Зробити висновки з виконання лабораторної роботи.
5. Відповісти на питання для самоперевірки.

Питання та завдання для самоперевірки

1. Функцію якого електричного пристрою виконує мультиплексор для логічних сигналів?

2. Яким аналітичним рівнянням описується робота мультиплексора з керуючими входом?

3. Як реалізувати схему мультиплексора з керуючим входом на елементах **„І-НЕ”**?

4. Розробіть, зберіть і випробуйте схеми на основі базового дешифратора й елементів „І-НЕ” або тих, що які реалізують задану функцію F. На вході дозволу встановити активний рівень. Варіанти задач наведено нижче.

1) $F = \overline{C} \cdot B \cdot \overline{A} \vee C \cdot B \cdot A \vee \overline{B} \cdot \overline{A}$;
2) $F = \overline{B} \cdot \overline{A} \vee C \cdot \overline{B} \vee \overline{C} \cdot B \cdot A$;
3) $F = \overline{C} \cdot \overline{A} \vee C \cdot \overline{B}$;
4) $F = \overline{C} \cdot \overline{A} \vee B$;
5) $F = \overline{C} \cdot \overline{B} \cdot A \vee C \cdot \overline{B} \cdot \overline{A} \vee \overline{C} \cdot B \cdot \overline{A}$;
6) $F = C \cdot B \cdot A \vee C \cdot B \cdot A \vee C \cdot B \cdot A$;
7) $F = \overline{C} \cdot \overline{B} \cdot \overline{A} \vee B \cdot A \vee C \cdot B$;
8) $F = \overline{C} \cdot B \cdot \overline{A} \vee \overline{B} \cdot A \vee C \cdot B$;
9) $F = \overline{C} \cdot \overline{B} \cdot \overline{A} \vee B \cdot A \vee \overline{C} \cdot \overline{B} \vee C \cdot A$;
10) $F = C \cdot B \cdot \overline{A} \vee B \cdot A \vee C \cdot B \vee \overline{C} \cdot B$;
11) $F = \overline{C} \cdot \overline{B} \cdot \overline{A} \vee C \cdot B \vee C \cdot \overline{A}$;
12) $F = C \vee \overline{B} \cdot A \vee \overline{C} \cdot \overline{A}$.

Лабораторна робота № 11. Основи проектування цифрових пристроїв у пакеті MicroCAP-8. Проектування цифрового автомата в MicroCAP-8

Мета роботи: отримати навички роботи в пакеті схемотехнічного аналізу MicroCAP-8. Спроектувати цифровий автомат у MicroCAP-8.

Загальні теоретичні відомості

MicroCAP-8 – це універсальний пакет програм схемотехнічного аналізу, призначений для вирішення широкого кола завдань. Важливою особливістю цього пакета є наявність зручного та дружнього графічного інтерфейсу, що робить його особливо привабливим для непрофесійної студентської аудиторії. Незважаючи на достатньо скромні вимоги до програмно-апаратних засобів ПК (процесор не нижче Pentium II, ОС Windows 95/98/ME або Windows NT 4/2000/XP, пам'ять не менша 64 Мб, монітор не гірший SVGA), його можливості достатньо великі. З його допомогою можна аналізувати не тільки аналогові, але й *цифрові пристрої*. Можливим є також і змішане моделювання аналого-цифрових електронних пристроїв.

Система MicroCap-8 має у своєму складі один здійснений модуль – MC8.exe. –  .

MicroCAP-8 є програмою з багатовіконним графічним інтерфейсом, що дозволяє будувати й редагувати схеми, моделі й зображення компонентів, а також представляти результати розрахунків у зручному графічному вигляді. Для роботи з цією програмою необхідно використовувати маніпулятор „миша". За допомогою миші можна змінювати розмір вікон, а також обирати команди меню. Одночасно можна редагувати кілька схемних файлів, розміщених у різних вікнах.

В інтегрованому пакеті MC8 використовується стандартний багатовіконний інтерфейс із спадаючими і розгорнутими меню (див. рис. 11.1).

Верхній рядок вікна – **рядок системного меню програми MC8**. На ньому зліва знаходиться *кнопка системного меню, що* дублює за зображенням ярлик програми. На цьому ж рядку знаходиться заголовок. Якщо відкрито вікно схем, то вказується ім'я файлу схеми й каталогу, у якому він розташований. Якщо ж відкрито вікно аналізу характеристик **Analysis** – указується вид аналізу.

Наступний рядок – **рядок меню команд** програми. Зліва на ньому розташована кнопка меню схеми, що керує розміром вікна схеми, а справа три стандартні в WINDOWS кнопки, що виконують аналогічне управління. Між цими кнопками розташовується **спадаюче меню команд програми MC8.**

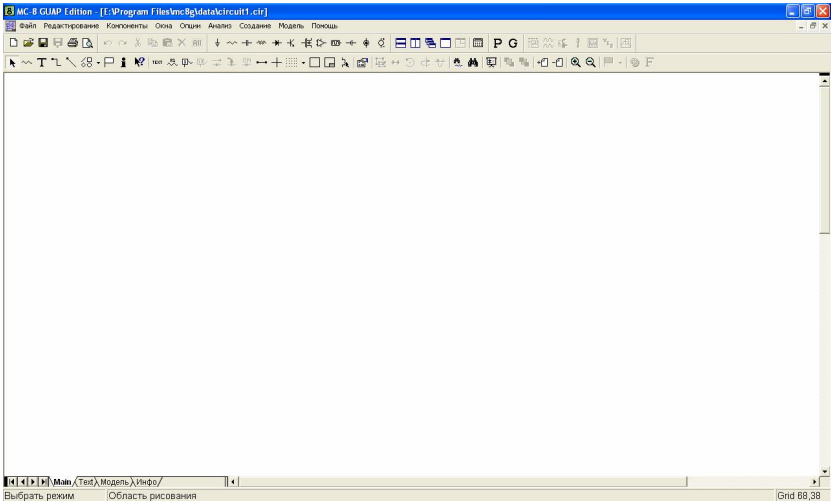


Рис. 11.1. Загальний вигляд програми MC8

Рядок інструментів. На цьому рядку розміщено піктограми найуживаніших команд, усі вони будуть описані нижче при описі відповідних пунктів меню. Піктограми команд негайної дії залишаються натиснутими нетривалий час і потім відновлюють своє первинне положення. Піктограми команд, що перемикають режими, залишаються в положенні „включено" до виконання подальшої команди.

Лінійки прокрутки. Дві лінійки прокрутки дозволяють панорамувати вікно схем або тексту по горизонталі або вертикалі.

Закладки перемикання вікна схем, вікна тексту. Натиснення на закладку *Text* виводить у робоче вікно зміст текстового вікна, у якому можна розміщувати опис математичних моделей компонентів поточної схеми, директиви й іншу текстову інформацію або зміст вікна схем. Перемикання між поточним вікном схеми й текстовим вікном проводиться також натисненням гарячих клавіш *Ctrl- G.*

Натискання на закладки Page1, Page2... відкриває відповідну сторінку схеми. Нова сторінка схеми створюється по команді панелі *Add+Page,* що відкривається натисканням правої кнопки миші, коли курсор знаходиться на рядку назви сторінки.

Натиснення на закладку *Models* відкриває текстову сторінку схеми, у якій знаходяться модельні директиви, що вміщуються туди по команді меню ***EDIT->Refresh Models***.

При натисненні правої кнопки миші у вікні схем, курсор набуває форми руки, і його переміщення при натиснутій кнопці дозволяє переміщувати (панорамувати) схему.

При роботі з MC8 використовується поняття *вибору об'єкта* (компонента схеми, його позиційного позначення, значення параметра, електричного ланцюга, блоку схеми або рядка тексту). Вибір окремого об'єкта виконується клацанням миші, вибір блоку — висновком його в прямокутну рамку (для цього потрібно клацнути кнопкою миші, умістивши курсор в один з кутів прямокутної області й, не відпускаючи її, розтягнути рамку до необхідних розмірів, після чого відпустити кнопку). Обраний об'єкт змінює колір; його можна перетягувати за допомогою миші і редагувати.

Для прискорення роботи з програмою використовується не тільки миша, але й клавіатура. Якщо команда меню має підкреслений символ, то ця команда викликається одночасним натисненням клавіш *Alt+підклеслений символ.* Наприклад, меню **Edit** відкривається натисненням клавіш *Alt+E***.**

Команди спадаючого підміню, наприклад **Select All,** викликаються натисненням підкресленого символу, у цьому прикладі символу *A*.

Багато команд, крім піктограм, викликаються натисканням *гарячих клавіш* і комбінацій клавіш. Наприклад, команда видалення

з копіюванням у буфер обміну ***Edit->Cut*** викликається натисканням піктограми  або комбінації клавіш **Ctrl+X.**

Зупинимося докладніше на **спадаючому меню команд програми MC8.**

New ( , **CTRL + N**) - створити новий схемний файл (або у форматі MC8 – у вигляді схеми, або у вигляді текстового файлу формату SPICE), або бібліотечний файл, або файл *.MDL моделі для Model.

Open ( , **CTRL + O**) - відкрити для редагування або аналізу схемний файл. Команда викликає діалогове вікно відкриття файлу, за допомогою якого можна відкрити схемний (.CIR, .MAC, .CKT) або бібліотечний файл (.LIB, .LBR, .STM, .MDL, .RES, .CAP, .IND).

Save ( , **CTRL + S**) - збереження схеми з активного з ім'ям і шляхом, зазначеним у рядку заголовка.

Save as - збереження схеми з активного вікна в новому файлі, ім'я якого зазначається в подальшому вікні, що відкривається.

Protect - зберігає файл, що знаходиться в схемному вікні, з паролем у зашифрованому форматі. Після цього зазначений файл можна проглянути або завантажити у вікно схемного редактора тільки ввівши пароль. Захищена схема може бути використана при моделюванні без пароля, якщо вона використовується іншою схемою як макроозначення. Команда використовується для захисту макромоделей від некомпетентного редагування. Для видалення захисту необхідно зберегти файл командою **Save As**.

Paths - вказівка шляхів розташування робочих схем (DATA), бібліотечних файлів і файлів (Model Library and Include Files), малюнків, що включаються (picture). Якщо необхідно вказати кілька шляхів, то вони розподіляються у відповідному рядку символом «;». MC8 у першу чергу шукає інформацію про моделі компонентів шляхами, зазначеними в рядку даних (DATA) для схем. Потім програма проглядає шляхи, зазначені в другому рядку (Model Library and Include Files). Також можна змінити шляхи пошуку

моделей, зазначивши в самому схемному файлі текстову директиву .PATH.

Cleanup - дозволяє навести лад у робочому каталозі шляхом видалення непотрібних додаткових файлів, що виникають при запуску аналізу. Частіше за все необхідно видаляти вихідні текстові файли з таблицями значень (*.TNO *.ANO *.DNO) і файли даних для графічного постпроцесора PROBE (*.TSA *.ASA *.DSA).

Migrate - дозволяє передати файли компонентів з більш ранньої версії програми MicroCAP. Після того, як буде вказаний розташувальний файл MCAP.DAT старішої версії програми, MC8 прочитає його і сформує список файлів, які можуть бути імпортовані (*.cmp, *.shp, *.pkg).

Translate - перетворення форматів схемних файлів (текстового SPICE в графічний MicroCAP і навпаки, схемного MC8 у схемний більш ранніх версій MicroCAP і ін.).

Binary Library to SPICE Text File: Перетворить бінарний бібліотечний файл формату 1BR у текстовий файл формату .LIB, що містить текстовий опис моделі.

SPICE Text File to Binary Library: Перетворить тестовий бібліотечний SPICE-файл .LIB у бінарний файл програми MC8 формату 1BR.

Schematic to SPICE Text File: Перетворить схемний файл MicroCAP в активному вікні в текстовий схемний SPICE-файл.

Schematic to Printed Circuit Board: Створює файл таблиці з'єднань для подальшого використовування в програмах розводки друкарської плати Protel, Accel, OrCad, або PCADS.

- ***Schematic to Old Version:*** Перетворить схему в активному вікні у формати схем попередніх версій MicroCAP: MC5, MC6, MC7.
- ***Bill Materials:*** Команда створює повний список інформації про компоненти схеми (ім'я, тип, величина, кількість та інші атрибути.
- ***Model to SPICE File:*** Перетворить файл даних Model у текстові модельні повідомлення формату SPICE у вигляді текстового SPICE-файлу.

- ***IBIS to SPICE File:*** Викликає діалогове вікно IBIS, яке створює еквівалентну SPICE-модель для IBIS-файлу (тимчасові діаграми інтерфейсу в спеціальному форматі). Одержаний SPICE-файл можна надалі використовувати при аналізі перехідних процесів.
- ***Touchstone Files:*** Конвертує вихідні параметри S, У, Z, G, H чотириполюсника в будь-який зазначений формат (S, У, Z, G, H).

Load MC File - завантаження файлів результатів розрахунку за методом Монте-Карло (*.DNO *.TNO) з подальшим пошуком у ньому варіантів, за яких схема не працює належним чином (установка Report When в MonteCarlo Options). Потім на основі аналізу помилкових варіантів створюються схеми з відповідними параметрами й завантажуються до схемного редактора.

Revert ( , **CTRL+ALT+R**) - відновлення вмісту файлу поточного вікна з диска.

Close (CTRL + F4) - завершення роботи за схемою, що знаходиться в активному вікні.

Print Preview () - попередній перегляд зображення перед друком.

Print ( , **CTRL + P**) - висновок на друк зображення в активному вікні відповідно до параметрів, заданих у вікні Print Setup.

Print Setup - вибір принтера й параметрів паперу.

Exit (**ALT + F4**) - завершення роботи з програмою MC8.

EDIT:

Undo ( , **CTRL + Z**) - відміна останньої команди редагування (відкіт назад), залежить від об'єму доступної пам'яті, звичайно не менше 20 дій.

Redo ( ,**CTRL +Y**) - повторення останньої відміненої команди (відкіт уперед).

Cut ( , **CTRL + X**) - видалення обраного () об'єкта й розміщення його в буфері обміну.

Copy ( , **CTRL** + C) - копіювання обраного об'єкта в буфер обміну.

Paste ( , **CTRL + V**) - копіювання вмісту буфера обміну в поточне вікно в місце, на яке вказує курсор.

Clear ( , **DEL**) - видалення обраного об'єкта без копіювання в буфер. **Clear Cut Wire** (**CTRL+DEL**) - обрізання ліній з'єднань (дротів), точно по межі виділеної прямокутної області.

Select all ( , **CTRL + A**) - виділення всіх об'єктів у поточному вікні або всього тексту в текстовому вікні.

Copy to Clipboard - копіювання поточного вікна (або виділеного блоку в ньому) у вигляді різних графічних файлів (відкриваються підміню) у буфер обміну.

Add Page () - додавання до схеми нової сторінки (великі схеми можуть розміщуватися на кількох сторінках).

Delete Page () - видалення однієї (кількох) сторінок схеми.

Refresh Models () - розміщення у вікні Model текстових описів моделей компонентів, які ще не були вміщені в жодне з текстових вікон. Ця команда копіює у зазначене вікно інформацію про моделі з бібліотек. Необхідна в тому випадку, якщо потрібно передати схемний файл іншому користувачу, що не має аналогічних бібліотек або для відновлення бібліотечних параметрів моделей у тому випадку, якщо інформація в текстових вікнах редагувалася.

Box - редагування об'єктів, укладених у прямокутну рамку (установлюється мишею в режимі ). Що відкривається підменю або кнопки піктограм на панелі інструментів  дозволяє копіювати блок вказаного число раз, створювати дзеркально відображений фрагмент, обертати проти годинникової стрілки на 90°, дзеркальне відображати щодо вертикальної і горизонтальної вісі, що розташована посередині блоку.

Окремо слід зазначити команду в меню **Make Macro**, що відкривається (**CTRL + M**). Ця команда створює макроозначення зі схеми, що міститься усередині прямокутного блоку (дає імена зовнішнім висновкам, зберігає під обраним ім'ям з розширенням .MAC у каталозі бібліотек, створює компонент у бібліотеці компонентів MACRO.CMP).

Change - зміна низки параметрів відображення схеми (стає зрозумілим з тих, що відкриваються підменю).

- • **Properties (F10):** Відкриває діалогове вікно *Properties* для вікна схеми, у якому можна змінити колір елементів схеми.
- **Graphic Object Properties:** Відкриває діалогове вікно *Graphic Object,* у якому можна змінити властивості відображення графічних об'єктів, таких як лінія, дуга, прямокутник, еліпс та ін. Можна змінити тип лінії, спосіб заливки, кольору. Зміни зачіпатимуть лише нові графічні об'єкти, що додаються редактором. Для зміни властивостей раніше введеного окремого графічного об'єкта необхідно ввійти до його властивостей за допомогою подвійного щиголя миші.
- **Attributes (CTRL+SHIFT+A):** Дозволяє встановлювати/ скидати відображення 5 основних атрибутів усіх компонентів схеми.
- **Apply Display Properties:** Копіює параметри відображення обраного компонента на всі інші компоненти цього класу для схеми в активному вікні.
- **Color:** Дозволяє змінити колір обраного компонента схеми (електронного компонента, тексту, сполучного провідника).
- **Font:** Дозволяє змінити параметри шрифту відображення атрибутів обраного компонента схеми.
- **Rename Components:** Змінює номери позиційних позначень компонентів згідно з загальноприйнятими угодами, так щоб номери вузлів і компонентів збільшувалися в обраному напрямі (або зліва направо, або зверху вниз).
- **Rename Defines:** Команда перейменовує символи, визначені директивою Define у тих випадках, коли є

конфлікт між символьними іменами в різних директивах **.Define**.

- **Reset Node Positions:** Номери вузлів, потенціали вузлів, струми в гілках та ін. інформація можуть пересуватися мишею по полю схеми з метою легкості для читання. Виконання цієї команди приводить до відновлення прийнятого за умовчанням положення на схемі зазначеної інформації.

Bring to Front () - переміщує нижній об'єкт, що перекривається, угору.

Send to Back () - переміщує верхній об'єкт, що перекривається, униз.

Go To Flag. Виконання команди приводить до появи діалогового вікна **Go To Flag.** У ньому можна обрати потрібну мітку схеми (прапор). При виборі відповідного прапора змінюється область схеми, що показується у вікні схемного редактора. Вибір мітки в діалоговому вікні приводить до того, що на екрані відображатиметься частина схеми таким чином, що обраний прапорець знаходитиметься посередині вікна схемного редактора. Прапорець ставиться в потрібне місце схеми в режимі встановлення прапорця ().

Find ( , **CTRL+F**) - пошук у поточному вікні схем або тексту різноманітних об'єктів.

Repeat Last Find: ( , F3) - повторює команду **Find** з поточними параметрами.

Replace: Проводить заміну тексту в текстовому вікні схеми або в текстовому описі схеми на мову SPICE. Функція заміни текстових атрибутів компонента схеми виконується установками діалогового вікна Attribute, що викликається за допомогою відповідної команди меню Change.

Find in Files: Команда викликає діалогове вікно *Find in Files,* яке використовується для пошуку на диску файлів з певним змістом. Можна задати, наприклад, фрагмент тексту (TEXT), ім'я (SHAPE), визначення моделі компонента (DEFINITION), ім'я компонента (COMPONENT), величину (ATTRIBUTE). Команда має значення в тих випадках, коли ім'я схемного файлу безнадійно забуто, зате

користувач добре пам’ятає, який компонент використовувався в ній.

COMPONENT:

Меню COMPONENT містить ієрархічний каталог бібліотек аналогових і цифрових компонентів. Він має систему розвертаються ієрархічних меню, що відкриваються при наведенні на них курсору мишею. В основному назви компонентів зрозумілі з найменування англійською мовою, проте нижче наведений список основних компонентів російською для кращої орієнтації в програмному пакеті (див.: рис. 11.2).

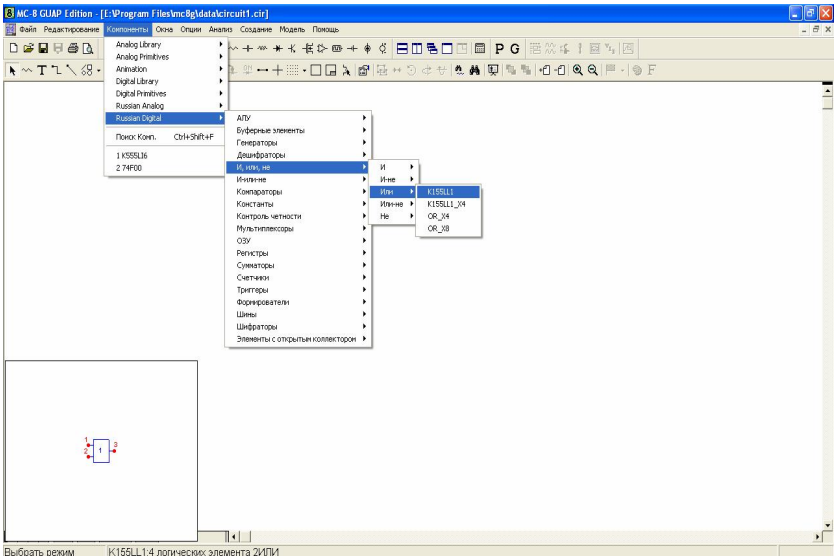


Рис. 11.2. Вибір компонента для синтезу логічної схеми автомата

Експеримент 1. Дослідження логічних станів електронної схеми у МС-8. Зберіть схему в МС-8 *(*див.: рис. 11.3). Включіть схему та дослідіть перехідні процеси в окремому вікні.

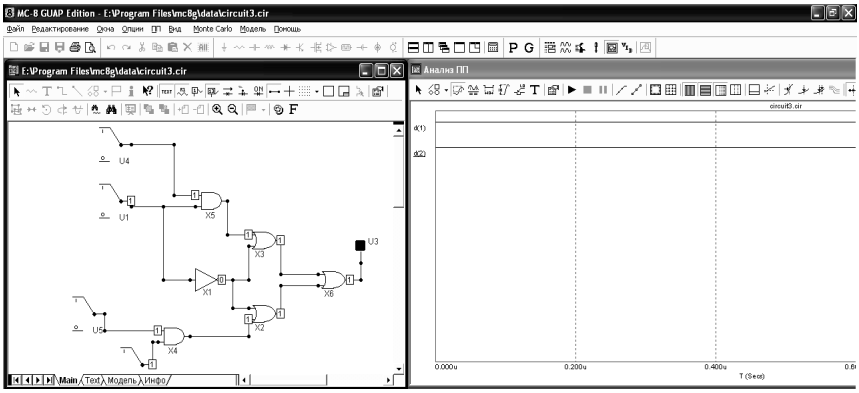


Рис. 11.3. Дослідження логічних станів електронної схеми

Експеримент 2. Дослідження роботи шифратора. Побудуйте схему, зображену на рис. 11.4. Включіть схему. Змінюючи за допомогою перемикачів рівні сигналів на виходах, заповніть таблицю функціонування шифратора.

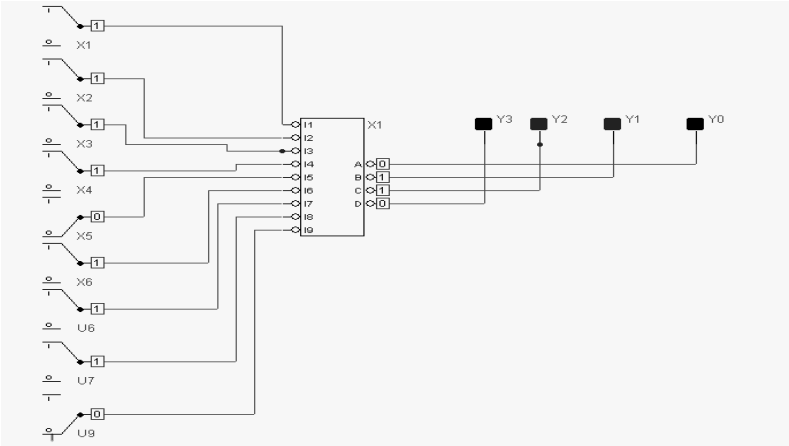


Рис. 11.4. Шифратор на мікросхемі 74147

Експеримент 3. Дослідження роботи дешифратора. Побудуйте схему, зображену на рис. 11.5. Включіть схему. У покроковому режимі роботи подайте на входи *X1, X2, X3, X4, X5*

слова, еквівалентні числам від 0 до 9. Заповніть таблицю функціонування дешифратора.

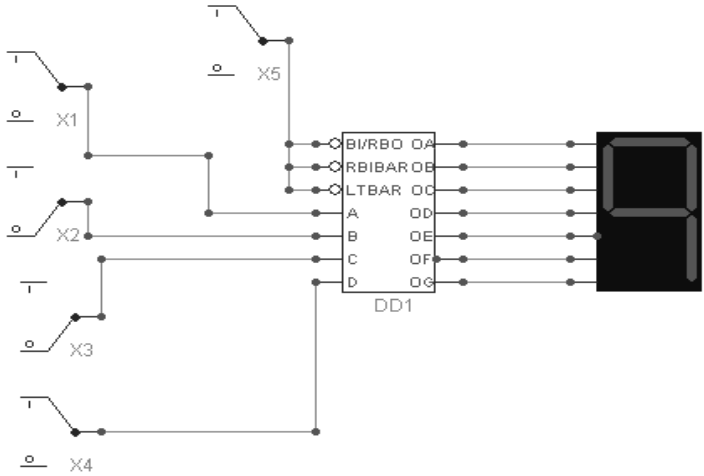


Рис. 11.5. Дешифратор двійкового коду в код семи сегментного індикатора на мікросхемі 7448

Приклад синтезу й моделювання цифрового автомата

Розглянемо синтез і моделювання цифрового автомата на дискретних компонентах. Припустимо є два логічні сигнали – *Y* і *Z*. Цифровий автомат повинен виділяти перший повний імпульс з послідовності цифрових сигналів *G* після надходження керівного сигналу *Y* і другу повну паузу – після приходу управляючого сигналу *Z*. Теорія цифрових синхронних і асинхронних автоматів і методи їх синтезу на основі RS-тригерів і мультиплексорів детально висловлено в [11]. Синтезуємо заданий автомат як асинхронний автомат Мура на основі асинхронних RS-тригерів. Автомати Мура описуються функціями переходів і виходів

$$a_{t+1} = f(a_1, x_1),\ z_1 = \varphi(a_1),$$

де a_1 и z_1 - стан автомата та його вихідний сигнал у момент часу t відповідно.

Кожний новий стан автомата однозначно визначається попереднім станом і вхідним сигналом, а стан автомата в цей момент часу однозначно визначає його вихідний сигнал. Таким чином, вихідний сигнал автомата Мура визначається тільки станом автомата й напряму не залежить від вхідних сигналів в у разі їх зміни без зміни стану автомата, вихідний сигнал не змінюється. Тому стани автомата (вихідні сигнали) можна зазначити у вершинах графа переходів.

Складемо граф переходів пристрою, що розробляється як автомат Мура (див. визначення вище). Вершини графа (стійкі стани автомата) визначимо з використовування коду Грея. Початковий стан автомата — 000.

Умовою можливості виділення повного імпульсу є наявність керуючого сигналу *У* і паузи в послідовності, а умовою виділення повної паузи – наявність сигналу *Z* й імпульсу в послідовності.

Зазначені комбінації сигналів визначають два шляхи переходу автомата з початкового стану 000 у стан 001 або 100. Ці шляхи показано на рис. 11.6 відповідними стрілками. Після закінчення дії керуючих сигналів необхідно передбачити повернення автомата в початковий стан. Керуючою комбінацією для цього переходу слугує ($У \bullet Z$). Охопимо замкнутою лінією всі стани на графі переходів, у яких значення однієї і тієї ж змінної (стан одного елемента пам'яті) дорівнює одиниці.

Ці замкнуті криві показано суцільною (Q_i=1), штриховою (Q_2=1) і пунктирною лініями (Q_3=1) відповідно. Вхід автомата в подібну замкнуту область станів і вихід з неї вимагають певних сигналів перемикання тригерів. Стрілками, що входять в область, позначають сигнали установки (які привласнюють змінній одиничне значення), а стрілками, що виходять з області, – сигнали скидання (ті, що привласнюють їй нульове значення). Для кожного тригера складається два вирази у формі суми кон'юнкцій (ДНФ): одне для сигналу установки (Prebar - S), а інше – для сигналу скидання (ClrBar - R). Кожна кон'юнкція повинна містити вхідні змінні (сигнали, якими здійснюється перехід) і вторинні змінні, пов'язані з цим переходом, але які не змінюють своє значення.

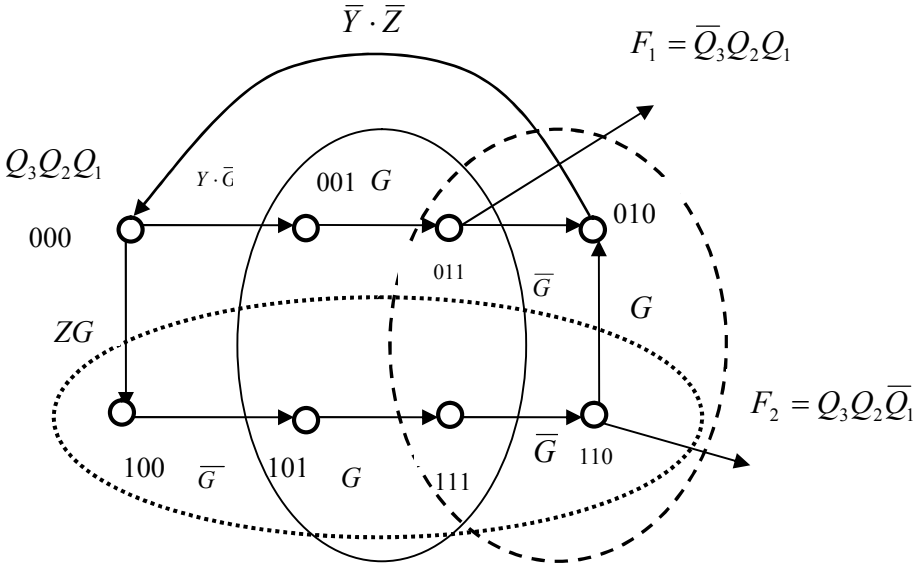


Рис. 11.6. Граф переходів асинхронного автомата Мура

Як вторинні змінні постають двійкові розряди коду стану в коді Грея. Включення у вирази вторинних змінних гарантує виконання переходів у правильній послідовності.

$$S_1=\overline{Q}_3\overline{Q}_2Y\overline{G}\vee Q_3\overline{Q}_2\overline{G},$$
$$R_1=\overline{Q}_3Q_2\overline{G}\vee Q_3Q_2\overline{G}=Q_2\overline{G};$$
$$S_2=Q_1G,$$
$$R_2=\overline{Q}_3\overline{Q}_1\overline{YZ};$$
$$S_3=\overline{Q}_2\overline{Q}_1ZG,$$
$$R_3=Q_2\overline{Q}_1G;$$
$$F_1=\overline{Q}_3Q_2Q_1,$$

$$F_2 = Q_3 Q_2 \overline{Q}_1,$$
$$F_3 = \overline{Q}_3 Q_2 Q_1 \vee Q_3 Q_2 \overline{Q}_1.$$

Після цього будується схема на основі асинхронних **RS**-тригерів, що реалізовує одержані логічні вирази. Для моделювання за допомогою програми MicroCap-8 як асинхронні **RS**-тригери використовуються **JK**- або **D**-тригери з асинхронними входами установки (PREBAR) і скидання (CLRBAR). При цьому входи синхронізації й інформаційні входи не задіються. Схема для моделювання наведена на рис. 11.7. Цифрові мікросхеми обираються з розділу **Component>Digital Library>...** У цьому прикладі для реалізації автомата використана серія ІМС - ТТЛ - 74NNN. Слід зазначити, що при малюванні цифрових пристроїв і оптимізації зображення схеми зручно використовувати інструменти „гумова лінія" і „ортогональний провідник".

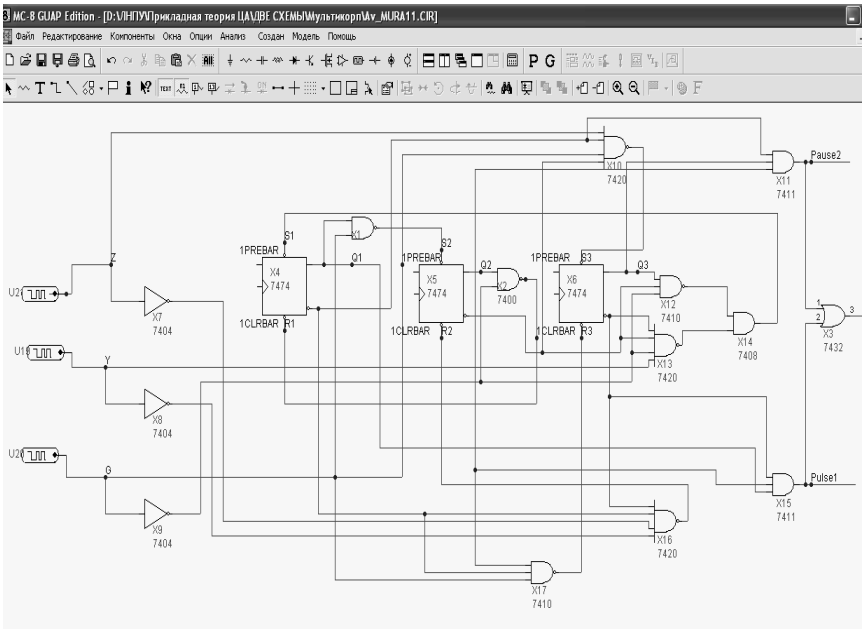


Рис. 11.7. Схема цифрового автомата у вікнах схемного редактора MicroCap 8

Для налагодження параметрів відповідного елемента, який був попередньо розмішений на екрані, потрібно виділити елемент на екрані (див.: рис. 11.8) та двійним щиглем лівою клавішею мишки відкрити відповідне вікно.

Для виконання моделювання необхідно потрібним чином запрограмувати тестові послідовності для однобітових генераторів цифрових сигналів U1(У), U2(Z), U3(G). однобітові цифрові генератори обираються командою Component->Digital Primitives->Stimulus Generators->Stim1. При розміщенні генератора на полі схеми (або при його виборі подвійним щиглем миші) відкривається вікно завдання його параметрів, у нижній частині якого можна спеціальною мовою задати послідовність логічних станів (рис. 11.9). До цієї процедури можна порекомендувати розкреслити олівцем на аркуші клітчастого паперу тестові цифрові сигнали, клітка відповідатиме тривалості одного такту (у нашому випадку – 1 мкс). Після введення запрограмованої послідовності у вікні завдання параметрів генератора й натиснення ОК текст послідовності з'являється в текстовому вікні Text (див.: рис. 11.10). Тут вікно схемного редактора розподілено на два командою *Windows-> Split Horizontal*. У нижньому вікні показується текстова область схемного редактора.

Відзначимо, що для програмування цифрових сигналів у цьому прикладі використана вказівка відносних часів (щодо моменту останньої події – зміни стану). Для цього використовується префікс у вигляді знака *«+»*. Другий знак *«+»* ставиться для позначення перенесення *SPICE*-тексту на наступний рядок. Для зручності моделювання в схемі застосовано текстові позначення цифрових вузлів, виконаних у режимі  : *Y, Z, G, Pause2, Pulse1* і т. ін.

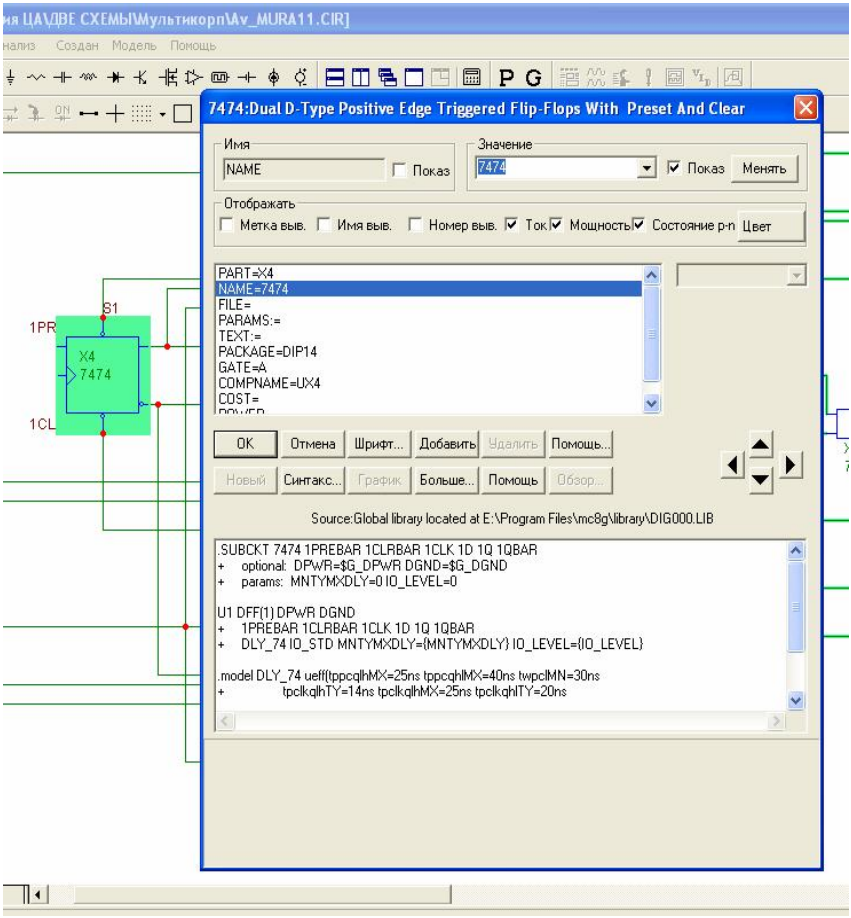


Рис. 11.8. Вікно властивостей компонента в MicroCap 8

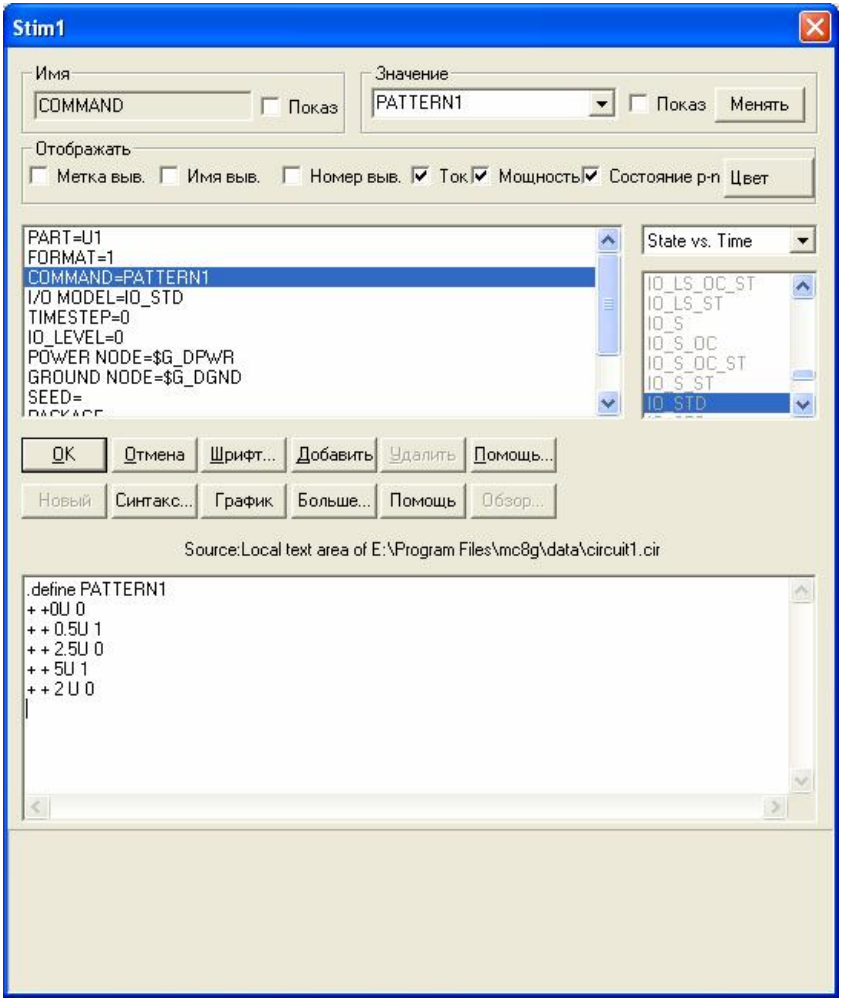


Рис. 11.9. Вікно завдання параметрів генератора цифрового сигналу

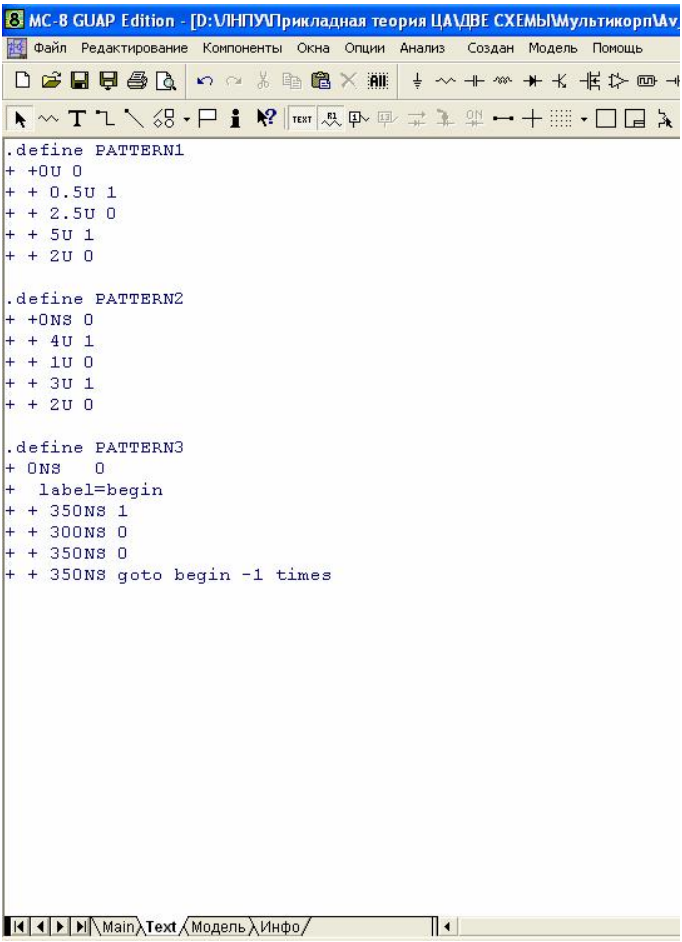


Рис. 11.10. Закладка Text у вікні MicroCap 8

Для виконання аналізу перехідних процесів необхідно обрати команду меню ***Analysis ->Transient*** і заповнити вікно *Analysis Limits*, що відкрилося (див.: рис. 11.12). На відміну від позначень аналогових напруг цифрові стани позначаються 0 (<вузол>), скорочено від Digital (цифровий). Можна виводити двійковий (*Bin(Q3,Q2,Q1)*), десятковий (*DEC(Q3,Q2,Q1)*), восьмеричний (ОСТ), шістнадцятиричний (HEX) код кількох сигнальних ліній. У

цьому прикладі це зручно зробити для висновку двійкового коду стану автомата *Bin(Q3,Q2,Q1)*. Після заповнення необхідних полів залишилося натискувати кнопку RUN і переконатися у правильній роботі синтезованого цифрового автомата (рис. 11.13). Тепер, щоб додати виконаній роботі завершенності, можна на графіки нанести текстові пояснення за допомогою команди T.

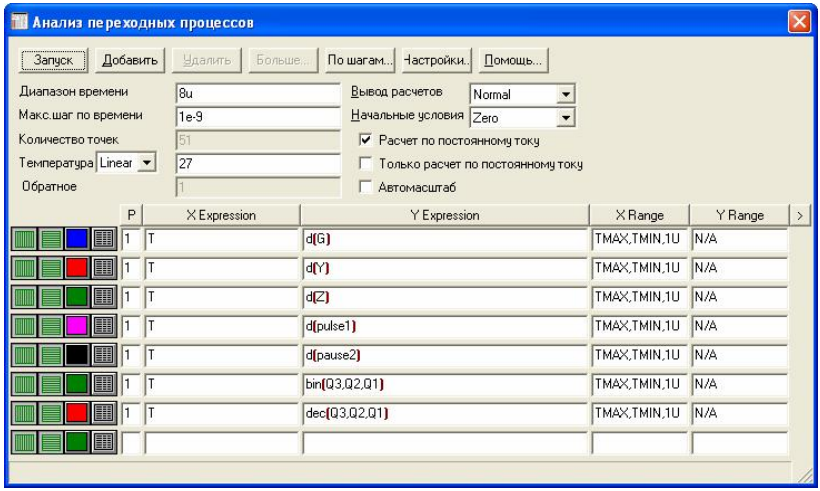


Рис. 11.12. Вікно налагодження аналізу перехідних процесів

Відзначимо, що відладку неправильно працюючої цифрової схеми й пошук несправностей у ній зручно проводити в режимі *Probe Transient*. Після запуску цього режиму щиголем мишею на відповідному вузлі (що цікавить), приводить до появи (зникнення) графіка сигналу в цьому вузлі в лівій частині екрана. Так можна достатньо швидко та просто діагностувати несправність.

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Розробити відповідну схему. Перевірити схему в MC8.
4. Скласти звіт з лабораторної роботи.
5. Зробити висновки щодо виконаної роботи.
6. Відповісти на питання для самоперевірки.

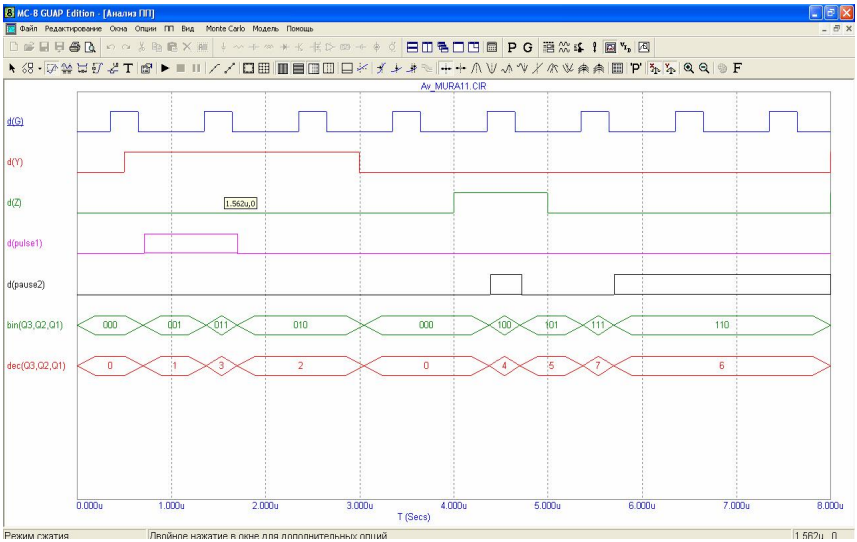


Рис. 11.13. Тимчасові діаграми роботи автомата МУРА

Питання та завдання для самоперевірки

1. Як отримати тимчасові діаграми роботи цифрового автомата в MC8?

2. Розробіть, зберіть і випробуйте схеми у MC8 на основі базового дешифратора й елементів „I-НЕ” або, тих що реалізують задану функцію F.

1) $F = \overline{C} \cdot B \cdot \overline{A} \vee C \cdot B \cdot A \vee \overline{B} \cdot \overline{A}$; 2) $F = \overline{B} \cdot \overline{A} \vee C \cdot \overline{B} \vee \overline{C} \cdot B \cdot A$;

3) $F = \overline{C} \cdot \overline{A} \vee C \cdot \overline{B}$; 4) $F = \overline{C} \cdot \overline{A} \vee B$; 5) $F = \overline{C} \cdot \overline{B} \cdot A \vee \overline{C} \cdot B \cdot \overline{A}$;

6) $F = C \cdot B \cdot A \vee C \cdot B \cdot A \vee C \cdot B \cdot A$; 7) $F = \overline{C} \cdot \overline{B} \cdot \overline{A} \vee B \cdot A \vee C \cdot B$.

Лабораторна робота № 12.
Моделювання суматорів у Electronics Workbench та MicroCAP- 8

Мета роботи: розробити схеми суматорів за допомогою EWB і MC8 та отримати тимчасові діаграми його роботи.

Загальні теоретичні відомості

Напівсуматори застосовують лише для складання однорозрядних двійкових кодів або наймолодших значущих розрядів багаторозрядних слів. При складанні старших розрядів багаторозрядних двійкових слів необхідно в схемі врахувати перенесення з попереднього (молодшого) розряду.

Такий пристрій є повним суматором: він повинен мати три входи (*A, B,* C_{in} – вхід перенесення) і два виходи: *S* – суми, і C_{out} – перенесення.

Складемо логічну функцію повного суматора.

$$S = \overline{A} \cdot \overline{B} \cdot C_{in} + \overline{A} \cdot B \cdot \overline{C}_{in} + A \cdot \overline{B} \cdot \overline{C}_{in} + A \cdot B \cdot C_{in}.$$

$$C_{out} = \overline{A} \cdot B \cdot C_{in} + A \cdot B \cdot \overline{C}_{in} + A \cdot B \cdot \overline{C}_{in} + A \cdot B \cdot C_{in}.$$

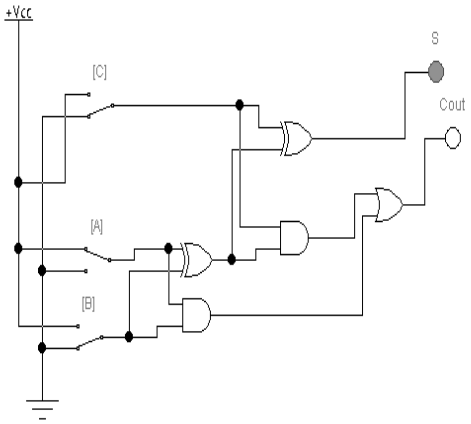
У результаті мінімізації отримаємо такі вирази:

$$S = (A \otimes B) \otimes C_{in} \quad \text{та} \quad C_{out} = A \cdot B + C_{in}(A \otimes B).$$

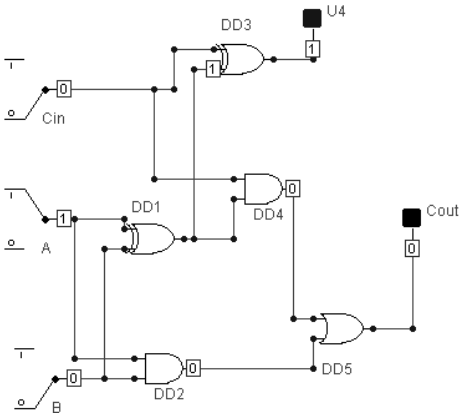
Аналізуючи останні вирази, бачимо, що над *A* і *B* треба провести операцію **XOR** і одержаний результат використовувати двічі: ще раз **XOR** з C_{in} – це дасть *S*, потім операція **AND** з C_{in}, а з її результатом операцію **OR** з *A·B* (яке виходить як перенесення з першого напівсуматора), і це дасть перенесення C_{out}. Таким чином, для побудови повного суматора потрібно 5 логічних елементів (ЛЕ) 2 типу **XOR**, 2 типи **AND** і 1 тип **OR**. Виведемо їх на робоче поле й

з’єднаємо відповідно до наведених ЛФ (див.: рис. 12.1, а,б). Задаючи різні значення вхідним сигналам, можна спостерігати за роботою суматора. У програмі МС при моделюванні показуються логічні стани всіх цифрових вузлів на схемі, що дозволяє глибше познайомитися з роботою пристрою.

Таким чином, використовуючи два напівсуматори й ЛЕ **OR**, можна побудувати повний суматор.



а)



б)

Рис. 12.1. Логічні схеми повного суматора в EWB та MC8

Суматори на інтегральних мікросхемах

Деякі типи суматорів випускаються в інтегральному виконанні та є однокристальними пристроями. Як приклад розглянемо роботу КМОП мікросхеми 4008 (чотирьох розрядного повного суматора). Вона складається з чотирьох однорозрядних суматорів і схеми прискореного паралельного перенесення.

У програмі EWB зберемо на основі цієї мікросхеми суматор (див.: рис. 12.2). Занесення в цей суматор двох чотирьохрозрядних складових $A_3A_2A_1A_0$ і $B_3B_2B_1B_0$ проводиться ключами [1]...[8]. Для зручності відліку двійкових чисел, що вводяться і виводяться, використані семисегментні індикатори з декодерами (*A, B* і *SUM*).

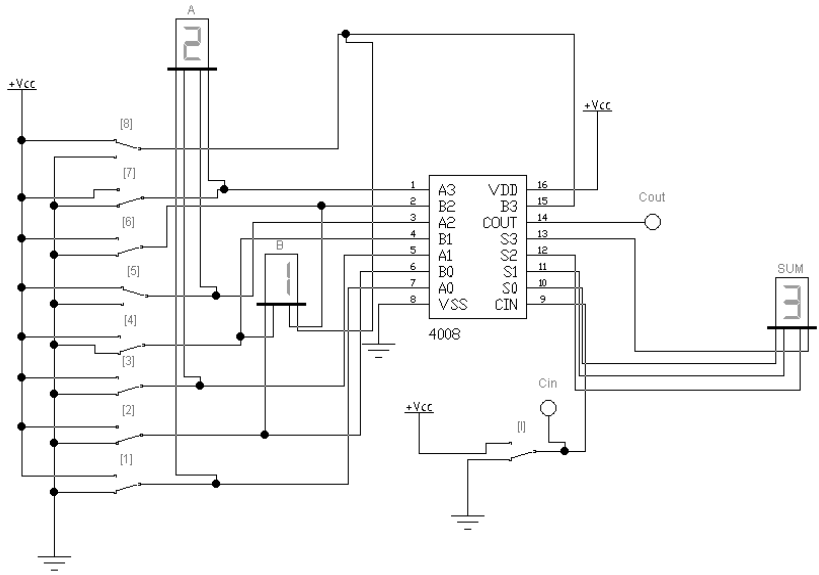


Рис. 11.2. Схема повного суматора на мікросхемі 4008

На вході перенесення з попереднього розряду C_{in} доданий ключ [I]. Цей вхід і вихід перенесення C_{out} контролюються логічними пробниками.

Порада. Перш ніж проводити підсумовування чисел, перевірте роботу схеми, подаючи спочатку тільки на входи

$A_3A_2A_1A_0$ послідовно всі двійкові числа від 0 до 15, потім, аналогічно, на входи $B_3B_2B_1B_0$.

Суматори на мікросхемах можна сполучати один з одним, підключаючи вихід C_{out} попередньої мікросхеми до входу C_{in} – наступної. Так, з двох мікросхем 4008 можна скласти восьмирозрядний двійковий суматор.

У програмі MC також розглянемо чотирьохрозрядний повний двійковий суматор, але на основі мікросхем ТТЛ 74283 (див.: рис. 11.3 та рис. 11.4).

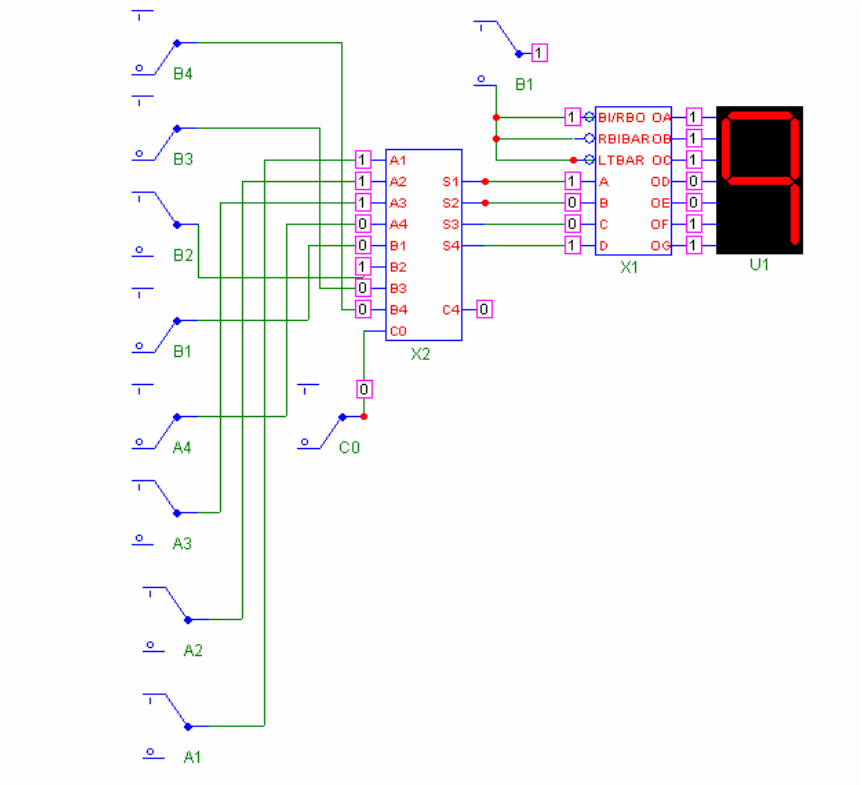


Рис. 11.3. Схема повного суматора на мікросхемі 74283

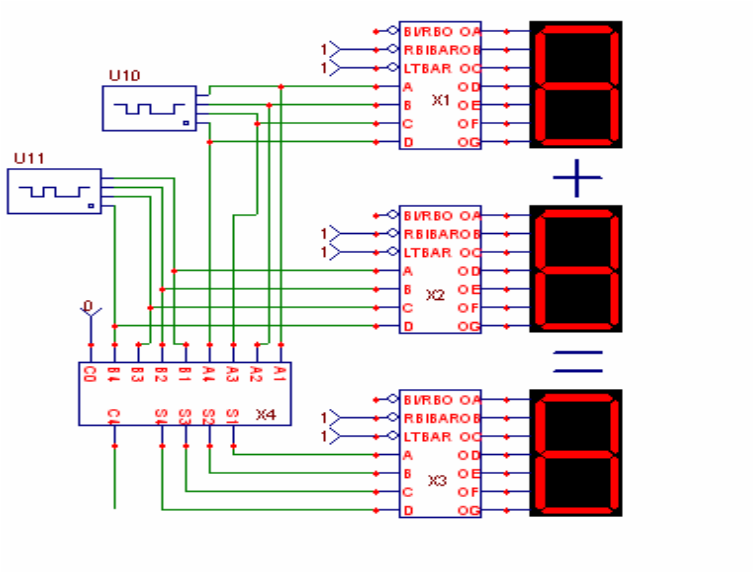


Рис. 11.4. Суматор на мікросхемі 7448

Для налагодження параметрів відповідного елемента, який був попередньо розмішений на екрані, потрібно виділити елемент на екрані (див.: рис. 11.4) та подвійним щиглем лівою клавішею мишки відкрити відповідне вікно (див.: рис. 11.5) та задати відповідні режими роботи генераторів.

Для виконання моделювання необхідно потрібним чином запрограмувати тестові послідовності для одно бітових генераторів цифрових сигналів *U10, U11*. Однобітові цифрові генератори обираються командою *Component->Digital Primitives->Stimulus Generators->Stim1*. При розміщенні генератора на полі схеми (або при його виборі подвійним щиглем миші) відкривається вікно завдання його параметрів, у нижній частині якого можна спеціальною мовою задати послідовність логічних станів. До цієї процедури можна порекомендувати розкреслити олівцем на клітчастому аркуші паперу тестові цифрові сигнали, клітка відповідатиме тривалості одного такту (у нашому прикладі – 1 мкс). Після введення запрограмованої послідовності у вікні завдання параметрів генератора й натиснення ОК текст

послідовності з'являється в текстовому вікні Text. Тут вікно схемного редактора розподілено на дві команди *Windows-> Split Horizontal*. У нижньому вікні показується текстова область схемного редактора.

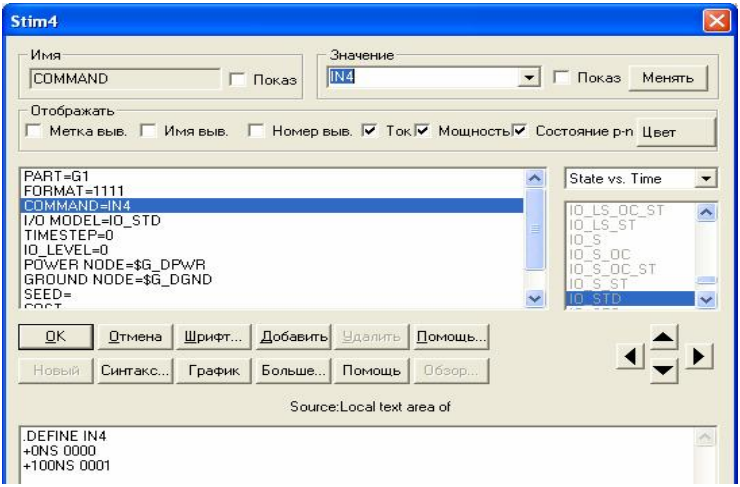


Рис. 11.5. Програмування тестових послідовностей для однобітових генераторів цифрових сигналів

Таблиця 11.1

Тестові послідовності для генераторів *U10, U11*

Для U10	**Для U11**
.define ina	.define inb
+0ns 0	+0ns 7
+100ns 3	+100ns 6
+200ns 1	+200ns 1
+300ns 0	+300ns 4
+400ns 7	+400ns 1
+500ns 3	+500ns 2
+600ns 5	+600ns 1
+700ns 2	+700ns 0
+800ns 6	+800ns 2
+900ns 3	+900ns 0

Для виконання аналізу перехідних процесів необхідно овибрати команду меню ***Analysis>Transient*** і заповнити вікно *Analysis Limits*, що відкрилося (див.:рис. 11.6).

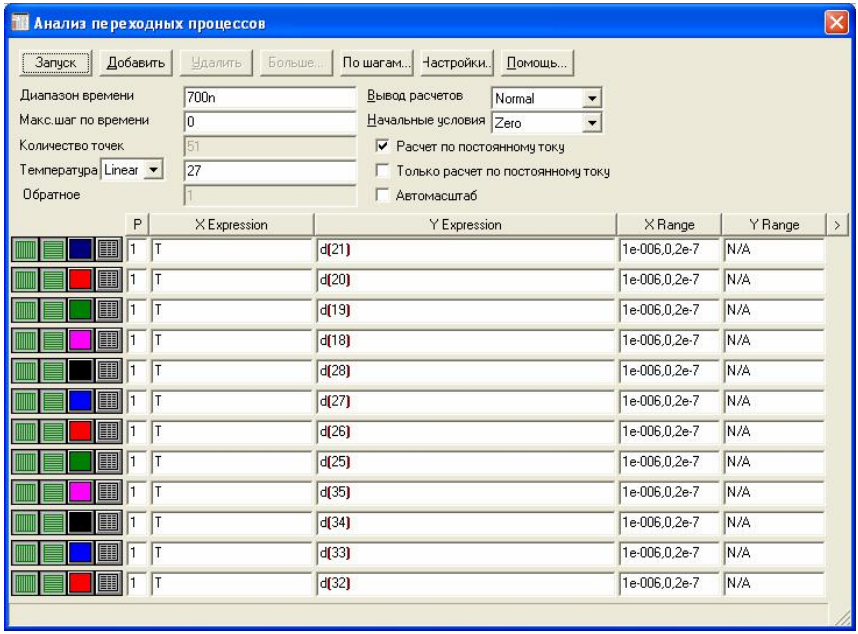


Рис. 11.6. Вікно – „Аналіз перехідних процесів”

Після заповнення необхідних полів залишилося натискувати кнопку RUN і переконатися в правильній роботі синтезованого суматора (див.:рис. 11.7).

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Скласти в *MC8* та *EWB* відповідні схеми повних восьмирозрядних суматорів. Провести відповідні експерименти.
3. Заповнити протоколи та скласти звіт з виконання лабораторної роботи.

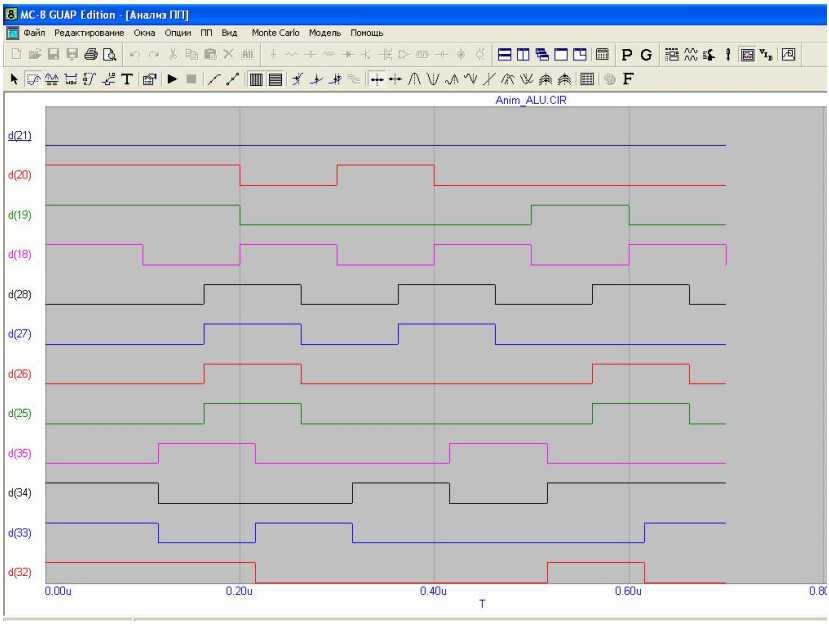


Рис. 11.7. Вікно зі тимчасовими діаграмами роботи повного суматора

Запитання для самоперевірки

1. Яким чином отримати схему повного шістнадцятирозрядного суматора в *MC8* та *EWB*?
2. Як перевірити працездатність схеми повного сумматора в *MC8*?
3. За допомогою яких логічних елементів можна спростити схему в повних суматорах *MC8* та *EWB*?

Лабораторна робота № 13. Моделювання АЛП у Electronics Workbench

Мета роботи: ознайомитися із роботою найпростіших АЛП за допомогою EWB та отримати тимчасові діаграми його роботи.

Загальні теоретичні відомості

Класична ЕОМ складається з трьох основних пристроїв: арифметико-логічного пристрою, пристрою управління і запам'ятовуючого пристрою. Розглянемо особливості організації цих пристроїв. Перш за все, розглянемо структуру арифметико-логічного пристрою.

Арифметично-логічний пристрій – АЛП, англійською мовою, відповідно, Arithmetic-Logic Unit – ALU, є складною цифровою мікросхемою, яка може виконувати всі комбінаційні функції, а також виконувати низку інших операцій. Не випадково тому, велика частина інформації в комп'ютерах обробляється саме в АЛП, яке входить складовою частиною в спеціальну інтегральну схему, іменовану центральним процесором (Central Processor Unit – CPU). У первинному комп'ютері IBM PC мікросхема CPU була виготовлена Intel Corporation і названа 8080. АЛП виконує одну з головних функцій мікропроцесора – опрацювання даних. Звичайно, АЛП має два вхідні порти й один вихідний. Вхідні порти забезпечуються буферними регістрами для зберігання одного слова даних. Останні в МВ не розглядаються, як, втім, й інші, пов'язані з АЛП складові: акумулятор, регістр станів і шини. Вони разом з АЛП будуть розглянуті в останній роботі, присвяченій моделюванню найпростішої ЕОМ. Перелік функцій власне АЛП залежить від архітектури мікропроцесора й різний для машин різних типів.

У сучасних ЕОМ арифметико-логічний пристрій не є самостійним блоком схемотехніки. Воно належить до складу мікропроцесора, на якому будується комп'ютер. Проте знання структури й принципів роботи АЛУ вельми важливе для розуміння

роботи комп'ютера в цілому. Для кращого розуміння цих питань проведемо синтез арифметичного пристрою, призначеного для виконання тільки однієї операції – додавання чисел з фіксованою комою, заданих у прямому коді, із старших розрядів множника. У ході цього процесу також звернемо увагу на особливості використовування розглянутих вище основних елементів ЕОМ.

Синтез АЛП проходить у кілька етапів. Спочатку необхідно обрати метод, згідно з яким передбачається виконання операції, і скласти алгоритм відповідних дій. Виходячи з алгоритму і формату початкових даних, слід визначити набір складових елементів АЛУ. Потім вимагається визначити зв'язки між елементами, установити порядок функціонування пристрою і тимчасову діаграму керуючих сигналів, які повинні бути подані на АЛУ від пристрою управління.

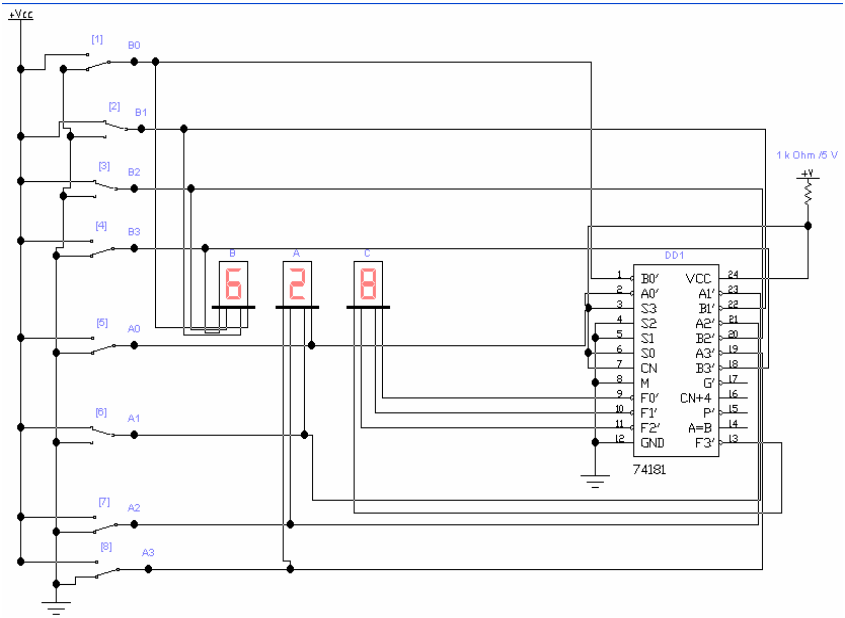


Рис. 11.1. Суматор на чотирьохрозрядному АЛУ на мікросхемі 74181

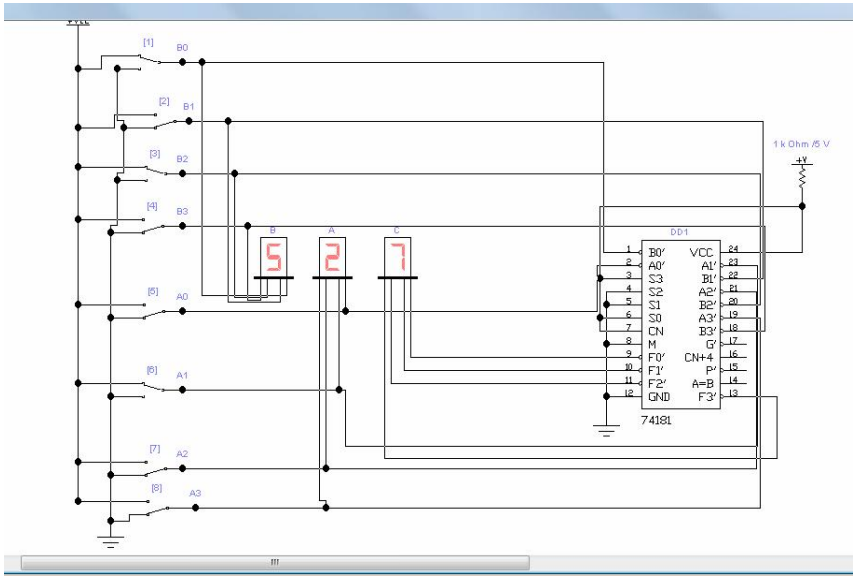


Рис. 11.2. Суматор на чотирьохрозрядному АЛУ на мікросхемі 74181

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Скласти у *MC8* та *EWB* відповідні схеми АЛУ (див.: рис. 13.1 та 13.2).
3. Провести відповідні експерименти. Заповнити протоколи та скласти звіт лабораторної роботи. Зробити висновки щодо виконаної роботи.

Запитання для самоперевірки

1. У якій послідовності виконується синтез АЛУ?
2. Яким чином отримати тимчасову діаграму роботи складових частин АЛУ?
3. За допомогою яких логічних елементів можна спростити схему в повних суматорів *MC8* та *EWB*?

Лабораторна робота № 14. Програмна реалізація автомата Мілі

Мета роботи: навчитися складати програми, які ілюструють роботу автоматів Мілі.

Загальні теоретичні відомості

Абстрактний автомат, уживаний у машині Тюрінга, сам собою не є універсальним і не зручний для практичного вживання. Другий із зазначених недоліків знімається при переході від абстрактного автомата до структурного, головна відмінність якого полягає в паралелізмі за входами і виходами. Розглянемо модель (рис. 14.1), що базується на структурному автоматі, у яку для забезпечення універсальності та практичного вживання введено формувачі вхідних дій 1, а найголовніше – об'єкти керування 2.

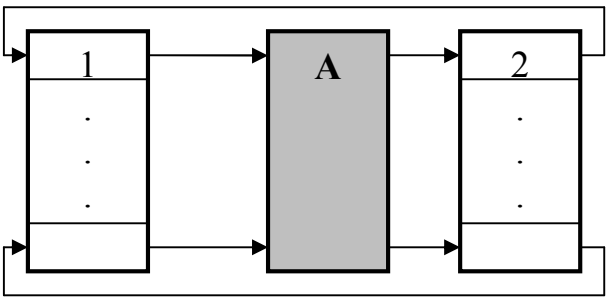


Рис. 14.1. Схема зв'язків автомата

Цю модель називатимемо „схема зв'язків автомата". У ній формувач 1 перетворить інформацію, що надходить від джерел, у вхідні дії, наприклад, шляхом порівняння значення лічильника з нулем. У цій схемі об'єктами управління можуть бути пристрої , у яких за командами автомата можливе виконання скільки завгодно складних операцій (зокрема й над пам'ятью).

У цій моделі вихідні дії формуються в певних „вузлах” простору станів, що задається графом переходів автомата. Тому процес керування об’єктами повністю задається цим графом у наочній і зрозумілій формі.

Ефективність уживання пропонованої моделі розглянемо на прикладі класичних завдань розпізнавання ланцюжків символів.

Уживання розпізнювачів є кінцевими автоматами без виходу, дозволяє, зокрема, вирішити завдання розпізнавання парності числа символів у ланцюжку. Проте навіть можливість формування вихідних ланцюжків у класичній моделі кінцевого автомата не дозволяє на її основі вирішити перераховані вище завдання. При цьому перші дві мови, що задаються контекстно-вільними граматиками, вимагають використовування моделі автомата з магазинною пам’яттю, а для третьої мови, визначуваної контекстно-залежною граматикою, необхідний перехід до лінійно-обмежених автоматів.

Кожний структурний автомат може належати до одного з таких класів:

- автомат без виходів;
- автомат Мура;
- автомат Мілі;
- змішаний автомат.

Як основний спосіб завдання автоматів використовуються графи переходів.

При цьому замість двійкових входів і виходів використовуються вхідні і вихідні дії. Вхідні дії можуть бути подіями і вхідними змінними, а вихідні дії – діями, які можуть виконуватися у вершинах, а також на дугах і петлях графа переходів. Для забезпечення універсальності вхідні змінні і дії реалізуються функціями, число і вид яких не фіксоваий, а визначається схемою зв’язків, яка містить також автомат і об'єкти керування.

У завданні як об’єкт керування постає лічильник, замінюючий магазин, і два індикатори „Пропустити” і „Відкинути”. Лічильник може бути обнулений, а його значення – збільшено або зменшено на одиницю. З виходу лічильника на вхід автомата надходить інформація про те, чи дорівнює значення лічильника нулю.

При цьому граф переходів автомата Мілі реалізується одним оператором *switch*. Якщо на всіх вхідних дугах деякої вершини цього графа наявні однакові дії, то вони можуть бути перенесені в неї. Таким чином, автомат Мілі перетвориться в більш компактний змішаний автомат, який, як і автомат Мура, реалізується за шаблоном, що містить два оператори *switch*.

На рис. 14.2 та 14.3. відповідно показано схеми зв'язків і граф переходів автомата Мілі.

Відповідно до запропонованої програмної реалізації автоматів номер події передається функції, що реалізовує автомат, як перший параметр. Другим параметром передається код чергового символу.

Подія *e0* призначена для ініціалізації автомата перед початком обробки нового виразу, і тому петлі в графах переходів, помічені цією подією, мають перший пріоритет. Подія *e1*, зазначена на рис. 14.2 не використовується у графах переходів і призначена для введення чергового символу. Вона застосовується програмі при виклику автомата в циклі.

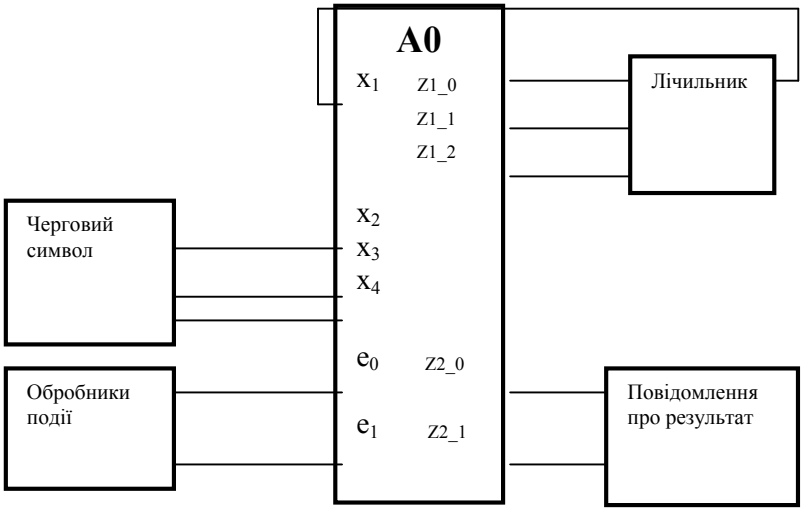


Рис. 14.2. Розпізнавання послідовностей дужок довільної глибини. Схема зв'язків автомата

У лістингу наведено програму для розпізнавання послідовностей дужок довільної глибини, що реалізовує автомат Мілі.

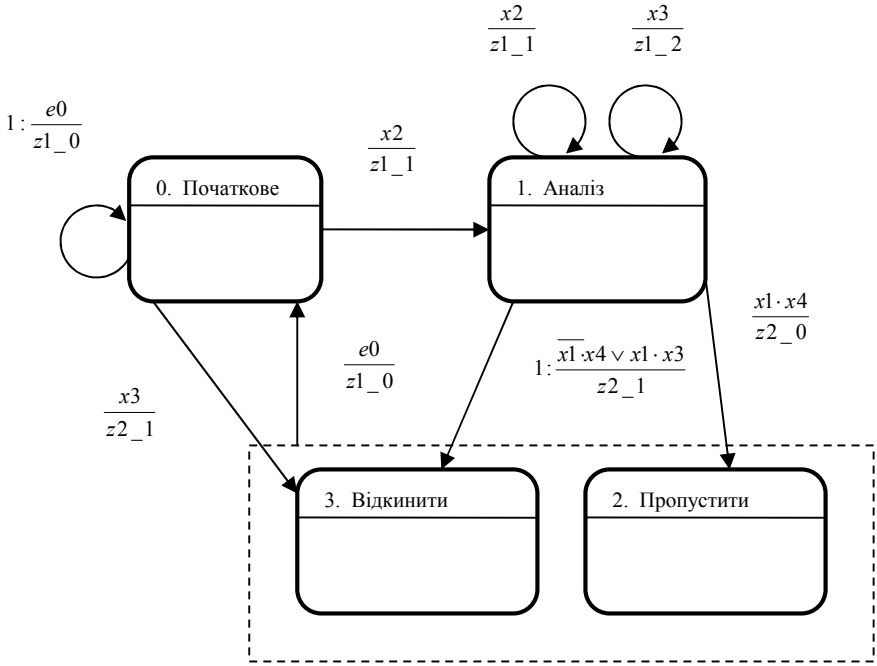


Рис. 14.3. Розпізнавання послідовностей дужок довільної глибини. Граф переходів автомата Мілі

На рис 14.2. та 14.3. відповідно набути такі позначення:

X1– значення лічильника -0;

X2 – символ ‘(’; X3 – символ ‘)’; X4 – кінець виразу;

E0 – початок обробки нового виразу (ініціалізація);

E1 – обробка наступного символу; Z1_0 – обнулювання лічильника;

Z1_1 – збільшення значення лічильника на одиницю;

Z1_2 – зменшення значення лічильника на одиницю;

Z2_0 – пропустити вираз;

Z2_1 – відкинути вираз.

Лістинг програми (реалізація автомата Мілі)
(Компілятор - C++ Builder 6)

```
#include <vcl.h>
#include <stdio.h>
#include <string.h>
#pragma hdrstop
#pragma argsused
int     y0 = 0 ;     // Стани автомата
int     i = 0 ;      // Лічильник символів
int x1()
{ return i == 0 ; }
int x2( char c )
{ return c == '(' ; }
int x3( char c )
{ return c == ')' ; }
int x4( char c )
{ return c == 0 ; }
void z1_0()
{ i = 0 ; }
void z1_1()
{ i++ ; }
void z1_2()
{ i-- ; }
void z2_0()
{ printf( "\n DOPUSTIT.\n" ) ; }
void z2_1()
{ printf( "\nOTVERGNUT.\n" ) ; }
void A0( int e, char c )
{
  switch( y0 )
  {
    case 0:
      if( x2(c) )  { z1_1() ;  y0 = 1 ; }
      else
      if( x3(c) )  { z2_1() ;  y0 = 3 ; }
      else
      if( e == 0 ) { z1_0() ; }
    break ;
    case 1:
      if( !x1() && x4(c) || x1() && x3(c) )   { z2_1() ;
y0 = 3 ; }
      else
      if( x1() && x4(c) )  { z2_0() ;  y0 = 2 ; }
      else
      if( x2(c) )          { z1_1() ; }
      else
```

```
      if( x3(c) )     { z1_2() ; }
    break ;
    case 2:
      if( e == 0 )    { z1_0() ;  y0 = 0 ; }
    break ;
    case 3:
      if( e == 0 )    { z1_0() ;   y0 = 0 ; }
    break ;
  }
}
void main()
{
  char  str[100] = "" ;
  int   j ;
  for(;;)
  {
    printf( "\n VVEDITE STROKY: " ) ;
    scanf( "%s", str ) ;
    printf( " VVEDENA STROKA: %s", str ) ;
    A0(0, 0 ) ;
    for( j = 0 ; j <= strlen(str) ; j++ )
      A0( 1, str[j] ) ;
  }
}
```

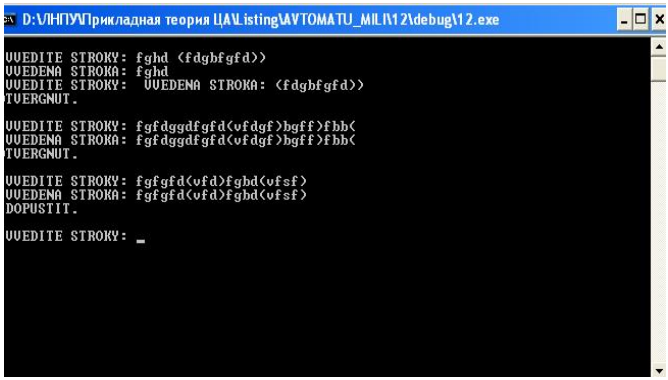


Рис. 14.4. Приклад роботи програми

Завдання

На рис. 14.5 наведено граф переходів змішаного автомата (Розпізнавання послідовностей дужок довільної глибини). Скласти програму, яка реалізує змішаний автомат.

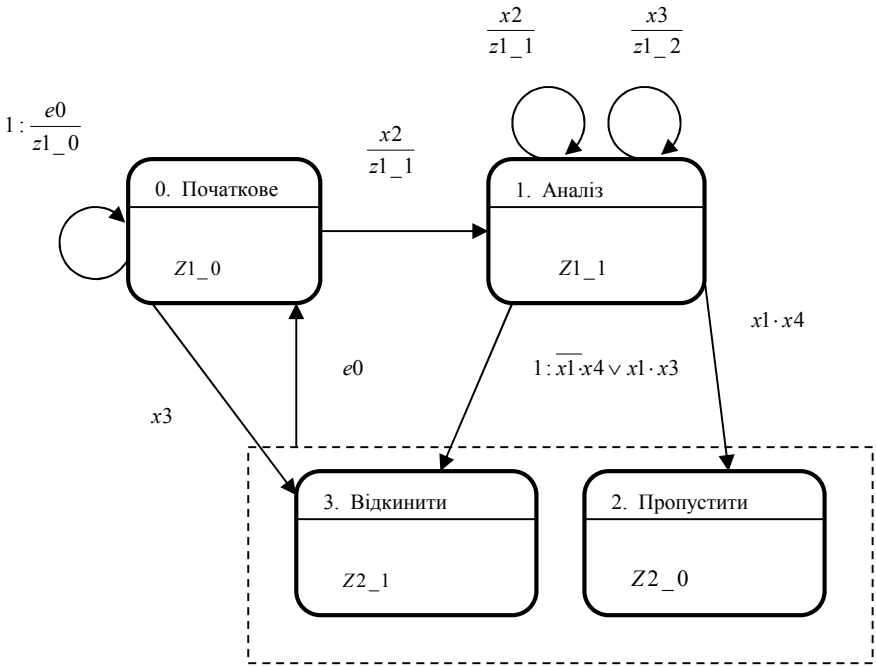


Рис. 14.5. Розпізнавання послідовностей дужок довільної глибини. Граф переходів змішаного автомата

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Скласти відповідну програму. Скласти звіт з лабораторної роботи. Зробити висновки щодо виконаної роботи.
4. Відповісти на запитання для самоперевірки.

Запитання для самоперевірки

1. У чому полягає відмінність автомата Мілі від змішаного автомата.
2. З яких етапів складається програмний синтез цифрового автомата?

Лабораторна робота № 15. Системи проектування програмованих логічних пристроїв

Мета роботи: проектування (розробка) технічної документації, що дозволяє виготовити заданий пристрій у заданих умовах.

Загальні теоретичні відомості

Проектування на основі схем програмованої логіки високої складності виконується тільки за допомогою систем автоматизованого проектування (САПР) (див.: рис. 15.1).

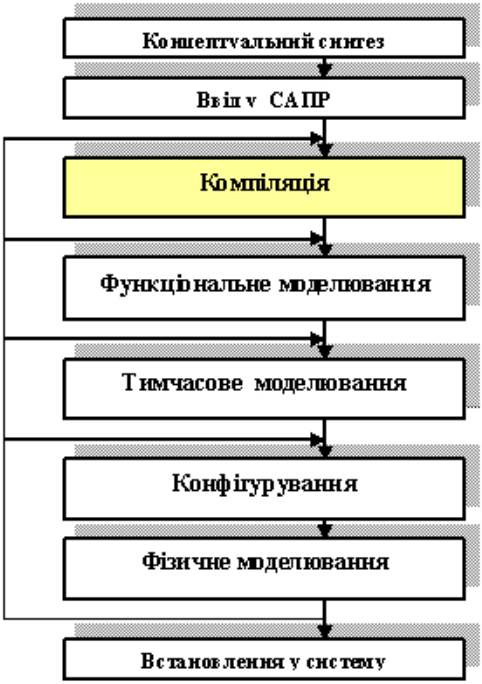


Рис. 15.1. Структура САПР ЦА

Щоб ознайомиться із прикладами простих засобів розробки ЦА, скористаємося популярним і досяжним компілятором – CUPL (Universal Compiler for Programmable Logic), який розроблений компанією Logical Devices, Inc. Він може бути використаний для програмування широкого спектра компонентів від різних виробників, а його демонстраційну версію можна безкоштовно завантажити з Internet. CUPL також підтримує багато зручних режимів уведення, що є невід'ємною межею його мови опису апаратного забезпечення.

Вхідний файл поділенний на кілька розділів. Заголовок містить деталі документації, а також інформацію, яку може використовувати для програмування необхідного пристрою. Розділи вхідної і вихідної специфікації використовуються для призначення імен сигналів на відповідних висновках програмованого пристрою. Розглянуті описи апаратної частини дозволяють зобразити проект одним з вищезгаданих відповідних методів (див.: рис. 15.2).

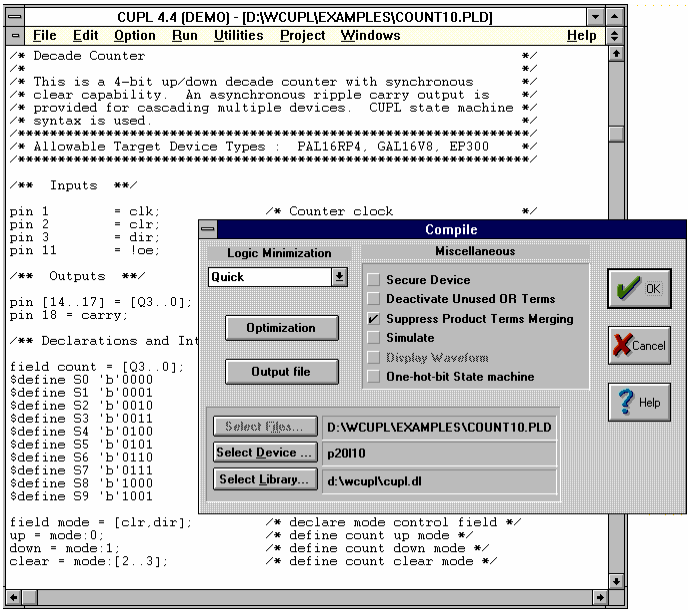


Рис. 15.2. Загальний вигляд компілятора CUPL

Логічні оператори та правильний синтаксис написання наведено в табл. 15.1.

Таблиця 15.1

Синтаксис мови CUPL для логічних операцій

Функція	**Оператор**	**Формат CUPL**	**Стандартне позначення**
І	&	A&B	$A \cdot B$
АБО	#	A#B	$A + B$
НІ	!	A!B	$\overline{A}$
Виключаюче АБО	$	A$B	$A \otimes B$

Як приклад, реалізуємо просту комбінаційну логічну схему з використанням компілятора CUPL і дуже популярного програмованого пристрою GAL 16V8 [1]. Спершу повернемося до прикладу, який розкривав процес розробки комбінаційної схеми.

Спроектуємо логічну схему з трьома входами *A, B* і *3,* виходи якої матимуть *високий* рівень, тільки якщо на більшій частині входів також наявний *високий* рівень.

Етап 1 – Складання таблиці істинності.

Згідно з умовами цього завдання, вихідний сигнал *x* повинен дорівнювати «1», якщо «1» дорівнює два або більше вхідних сигналів; для всіх інших випадків вихідний сигнал повинен дорівнювати «0» (табл. 15.2).

Таблиця 15.2

Таблиця істинності

A	B	C	x	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\rightarrow \overline{A} \cdot B \cdot C$
1	0	0	0	
1	0	1	1	$\rightarrow A \cdot \overline{B} \cdot C$

A	B	C	x	
1	1	0	1	$\rightarrow \; A \cdot B \cdot \overline{C}$
1	1	1	1	$\rightarrow \; A \cdot B \cdot C$

Вираз для вихідного сигналу в диз'юнктивній формі.

$$x = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C.$$

Лістинг програми у CUPL

```
Name combo1.pid                          ;
Partno 1                                    ;
Data June 2                                ;
Revision 06                                ;
Designer  Lahno                          ;
Company LNPU                           ;
Assembly Chapter 4                      ;
Location Chapter 4                       ;
Device G16V8                              ;
Format j                                        ;
/*      Primer                        */
/*      Vhod                        */
        pin 15A                                  ;
        pin 25B                                  ;
        pin 35C                                  ;
/*      Vihod                        */
        pin 195 X                                   ;
/*      Opisanie apparaturi                    */
/*x = A·B·C + A·B·C + A·B·C + A·B·C.*/

!A&B&C#A&!B&C#A&B&!C#A&B    ;
```

На рис. 15.3. наведено загальний вигляд схеми, отриманої за допомогою компілятора CUPL.

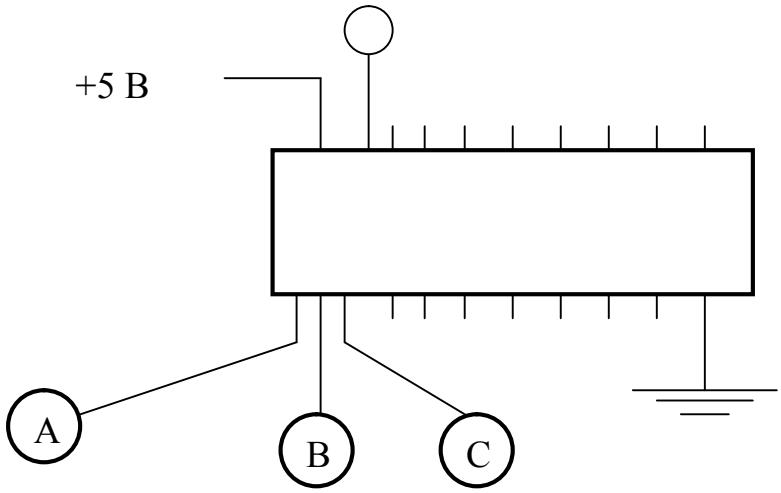


Рис. 15.3. Отримана схема

Компіляція проекту

Перед початком компіляції логічного опису проекту задаються такі параметри:

- – критерій логічної мінімізації;
- – режим оптимізації (Best for Polarity, DeMorgan, Keep XOR, P-Term Sharing);
- – тип пристрою (наприклад, PAL16, PAL20) і його найменування (наприклад, P1618, P16R4);
- – тип корпусу (наприклад, DIP або LCC).

Процес компіляції проекту розбивається на 5 етапів:

1) обробка тексту програми препроцесором;

2) перетворення описів автоматів станів, таблиць істинності, функцій, визначених розробником, у логічні рівняння;

3) виконання перетворень, одержаних на попередньому етапі рівнянь, для реалізації на пристроях обраного типу (ураховується число висновків та інші фізичні характеристики, полярність і рівень сигналів);

4) логічна мінімізація одержаної двійкової матриці;

5) перевірка можливості розміщення синтезованого проекту в обраному пристрої; генерація документації і вихідних файлів.

Мова VHDL

Мова VHDL з'явилася на початку 80-х рр. на запити організацій Міністерства оборони США. Перша її версія, призначена в основному для уніфікації описів проектів у різних відомствах, була прийнята в 1985 р. У 1987 р. мова VHDL була прийнята міжнародним інститутом IEEE (Institute Electrical and Electronic Engineers) як стандарт VHDL-87. Він використовувався, головним чином, для опису вже спроектованих систем. Використовування для задач синтезу пристроїв (робота з компіляторами) почалося з 1991 р. В 1993 р. IEEE набуває нового розширеного стандарту VHDL-93. Мова може бути використаний для проектування ЦА різних ієрархічних рівнів — від вентильного рівня представлення схем до рівня системи в цілому. У наш час вона є, мабуть, найпопулярнішою серед проектувальників цифрової апаратури. Приблизною за популярністю є мова Verilog, і практично будь-яка сучасна САПР засобів ВТ або цифрових пристроїв має у своєму складі компілятори з цими мовами (як вхідними, так і вихідними).

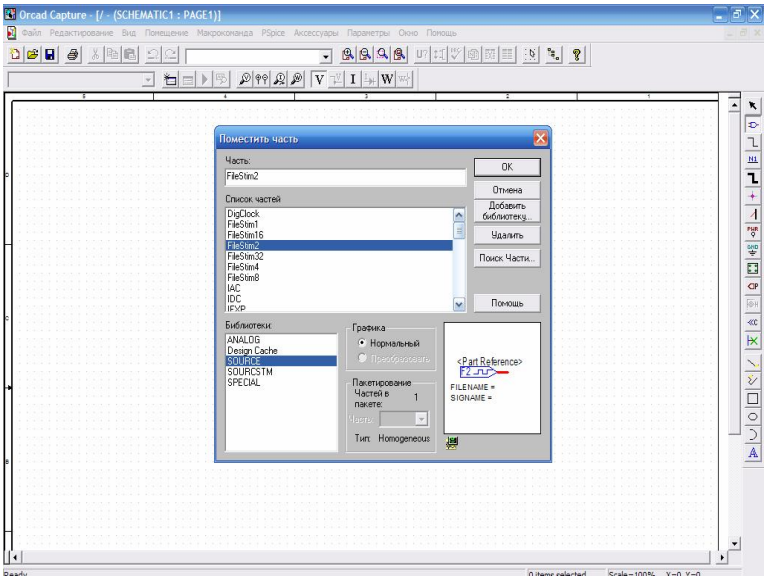
Мова VHDL є проблемно-орієнтованою мовою, її основні прикладні аспекти пов'язані з використанням як робочого інструменту для завдань опису структури і/або поведінки широкого класу цифрових пристроїв. Описи можуть використовуватися для синтезу і/або моделювання таких систем. Відповідно до призначення, мова пристосована для опису систем як з погляду їх структурної організації (з модулів з відомою поведінкою), так і з погляду поведінки або системи в цілому, або всіх її складових частин. Найбільші обмеження на набір допустимих (щодо стандарту) операторів мови мають компілятори для синтезу спроектованих пристроїв, значно менше обмежень існує у систем моделювання.

Приклад поведінкових описів елементів мовою VHDL

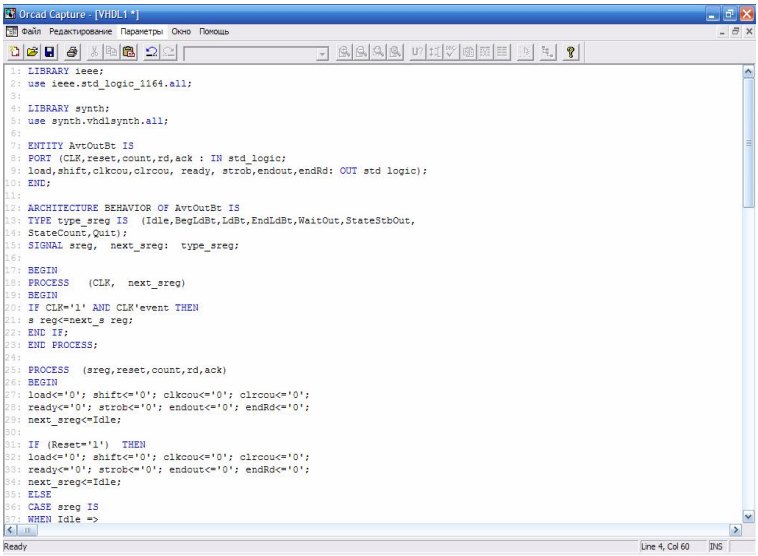
Проілюструємо поведінкові варіанти описів на найпростіших прикладах. Необхідно, наприклад, описати мовою VHDL логічний елемент, що реалізовує функцію $Z = (a \vee b)c$ під найменуванням

input3_orandl. Починаючи з розділу *Entity Declaration*, опис може мати такий вигляд (наприклад, у Orcad, див. рис. 15.4):

```
ENTITY input3_orandl IS -- entity declaration
PORT (a,b,c: IN BIT; -- port statement
Z: OUT BIT) ;
END input3_orandl;
ARCHITECTURE one OF input3_orandl IS
-- architecture "one" of entity input3_orandl
BEGIN
orand3:  PROCESS
BEGIN
IF  (c = 4')  THEN z<= a OR b
ELSE z<='0';
END IF;
WAIT ON a,b,c;
END PROCESS;
END;
```



а)



б)

Рис. 15.4. Загальний вигляд програми на VHDL у Orcad

Завдання 1

За допомогою компілятора CUPL отримати схеми для наступних виразів (див.: табл. 15.3).

Завдання 2

За допомогою мови VHDL отримати схеми для наступних виразі (див.: табл. 15.3).

Таблиця 15.3

Варіанти завдання

Варіант	**Логічний вираз**
1	$y = A \cdot (\overline{B} \cdot \overline{D} + D) + A \cdot \overline{B} \cdot C + \overline{D}.$
2	$y = \overline{A} \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot D + A \cdot \overline{B} \cdot C + \overline{D}.$
3	$y = \overline{A} \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot D + \overline{D} + A \cdot \overline{B} \cdot \overline{C}.$
4	$y = A \cdot \overline{B} \cdot C + A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A.$

Варіант	Логічний вираз
5	$y = A \cdot \overline{B} + A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A \cdot \overline{B} \cdot C.$
6	$y = A \cdot \overline{B} + A \cdot (\overline{B} \cdot \overline{D} + C) + D \cdot \overline{A} \cdot \overline{B} \cdot \overline{C}.$
7	$y = \overline{A} \cdot \overline{B} + A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A \cdot \overline{B} \cdot C + \overline{A} \cdot B.$
8	$y = \overline{A} \cdot \overline{B} \cdot D + A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C.$
9	$y = A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{D} + C.$
10	$y = \overline{C}(\overline{A} \cdot \overline{B} \cdot \overline{D} + D) + A \cdot \overline{B} \cdot C + \overline{D}.$
11	$y = A \cdot (\overline{B} \cdot \overline{D} + C) + \overline{D} \cdot (A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}).$
12	$y = A \cdot (\overline{B} \cdot \overline{D} + C) + D \cdot (A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}).$
13	$y = B \cdot (A \cdot \overline{D} + C) + \overline{D} \cdot (\overline{B} \cdot \overline{C} \cdot A + \overline{A} \cdot B \cdot C) + D.$
14	$y = A \cdot \overline{B} \cdot \overline{D} \cdot C + A \cdot (\overline{B} \cdot \overline{D} + C \cdot D) + D \cdot \overline{A} \cdot \overline{B} \cdot \overline{C}.$
15	$y = C \cdot (\overline{A} \cdot \overline{D} + B) + \overline{D} \cdot (A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}).$
16	$y = A \cdot \overline{B} \cdot \overline{D} + C \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}.$
17	$y = (\overline{C + D}) + A \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C \cdot D + A \cdot C \cdot D.$
18	$y = A \cdot B \cdot (\overline{\overline{C} \cdot D}) + A \cdot \overline{B} \cdot D + \overline{B} \cdot \overline{C} \cdot \overline{D}.$

Хід виконання роботи

1. Повторити конспект (теоретичні відомості).
2. Отримати допуск до роботи.
3. Скласти відповідну програму. Скласти звіт з лабораторної роботи. Зробити висновки щодо виконаної роботи.
4. Самостійно вивчити основні оператори мови VHDL.

Запитання для самоперевірки

1. Перелічіть основні етапи проектування цифрових пристроїв за допомогою систем автоматизованого проектування.
2. Перерахуйте мови програмування та компілятори, які можуть бути використані для автоматизації програмування цифрових систем.

СПИСОК ЛІТЕРАТУРИ

1. Будинский Я. М. Логические цепи в цифровой технике : підручник для студ. вищ. навч. закл. / Я. М. Будинский. – М. : Связь, 1977. – 164 с.
2. Вавилов Е. Н. Синтез схем электронных цифровых машин / Е. Н Вавилов., Г. П. Портной. – М. : Советское радио, 1963. – 176 с.
3. Жабін В. І. Цифрові автомати. Практикум: підручник для студ. вищ. навч. закл. / В. І. Жабін. – К.: ВЕК, 2004. – 160 с., ил.
4. Электронные вычислительные машины и системы: підручник для студ. вищ. навч. закл. / Б. М. Каган. – М.: Энергоатомиздат, 1991. – 212 с.
5. Карпов Ю. Г. Теория автоматов: підручник для студ. вищ. навч. закл. / Ю. Г. Карпов. – СПб.: Питер, 2002. – 224 с.
6. Лазарев В. Г. Синтез управляющих автоматов / В. Г. Лазарев, Е. И. Пийль. – М.: Энергия, 1978. – 246 с.
7. Лахно В. А. Прикладна теорія цифрових автоматів: метод. вказівки до лабораторних робот для студ. спец. 7.091501 „Комп'ютерні системи та мережі"; / В. А. Лахно. – Луганськ: Вид-во ДЗ „ЛНУ імені Тараса Шевченка", 2008. – 68 с.
8. Прикладная теория цифровых автоматов: підручник [для студ. вищ. навч. закл.] / А. Я. Савельєв – М.: Высшая школа, 1987. – 272 с.
9. Самофалов К. Г. Прикладная теория цифрових автоматов: підручник для студ. вищ. навч. закл. / К. Г. Самофалов, А. М. Романкевич. – К. : Вища школа, 1987. – 369 с.
10. Точчи Рональд. Цифровые системы. Теория и практика / Точчи Рональд, Дж., Уидмер, Нил, С.; Пер. с англ. – М. : Вильямс, 2004. – 1024. с.
11. Филиппов А. Г. Проектирование логических узлов ЭВМ / А. Г. Филиппов, О. С.Белкин. М. : Сов. радио, 1974. – 144 с.

12. Якубовський С. В. Цифровые и аналоговые интегральные микросхемы / С. В. Якубовский, Л. И. Нельсон, В. И. Кулешова. – М.: Радио и связь, 1990. – 496 с..
13. Emmanuel C. Ifeachar, Barrie W. Jervis. Digital Signal Processing. – М. : Вильямс, 2007. – 1222 с.
14. Floiyd L., Thomas. Digital Fundamentals. – Sixth Editional. 1997 by Prentice-Hall, Inc.
15. Short K. Microprocessor And Programmed Logic. – 2-nd Ed.- Edlewnod Cliffc : Prentice-Hall, 1987. – 515 p.

Навчально-методичне видання

ЛАХНО Валерій Анатолійович
МОГИЛЬНИЙ Геннадій Анатолійович

„ЛАБОРАТОРНИЙ ПРАКТИКУМ З ПРИКЛАДНОЇ ТЕОРІЇ ЦИФРОВИХ АВТОМАТІВ”

для студентів напряму 0915 „Комп’ютерна інженерія” спеціальності 7.091501 „Комп’ютерні системи та мережі”

За редакцією авторів
Комп’ютерний макет – Лахно В. А.
Коректор – Ширяєв Д. О.

Здано до склад. 01.11.2010 р. Підп. до друку 01.12.2010 р.
Формат 60x84 1/16. Папір офсет. Гарнітура Times New Roman.
Друк ризографічний. Ум. друк. арк. 8,02. Наклад 300 прим. Зам. № 42.

Видавець і виготовлювач
Видавництво Державного закладу
„Луганський національний університет імені Тараса Шевченка”
вул. Оборонна, 2, м. Луганськ, 91011. Тел./факс: (0642) 58-03-20
e-mail: alma-mater@list.ru
Свідоцтво суб’єкта видавничої справи ДК № 3459 від 09.04.2009 р.