# Лабораторна робота 7
## Синтез кінечного автомата

1. **Мета**: одержати теоретичні і практичні знання в розробці кінечних автоматів (FSM). Лабораторна робота служить для вивчення того, як програмувати кінечний автомат на VHDL.

**2.Теоретичні відомості**

Автомат - це послідовна логічна схема, яка виконує алгоритм керування як наперед задану послідовність його станів. Його наступний стан $S_{i+1}$ залежить від попереднього стану $S_i$ і від вхідного сигналу $X_j$. Якщо його вихідні сигнали $Y_p$ залежать тільки від його станів $S_i$, то такий автомат називається автоматом Мура. Якщо його вихідні сигнали $Y_p$ залежать як від його станів $S_i$ , так і від сигналів $X_j$, тоді це автомат Мілі. Алгоритм функціонування автомату представляється його графом станів (state diagram) або діаграмою автомата.

Вузли графа станів представляють стани автомата, а його направлені дуги представляють переходи від одного стану до іншого. В графі станів автомата Мура у вершині $S_k$ помітка $Y_P$, відділена косою "/" від помітки $S_k$, означає, що на виході $Y_P$ буде 1, коли автомат знаходиться в цьому стані. Дуга, помічена мітками вхідних сигналів, які формують булевську функцію від вхідних змінних, означає умову переходу. Така дуга може бути помічена вихідними мітками, якщо це граф станів автомата Мілі. На рис.1 показано приклад графа станів, на якому представлено виходи як автомата Мілі, так і автомата Мура.

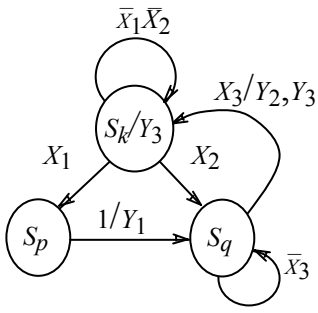


Рис.1

Якщо помічена дуга міткою $X_iX_j/Y_pY_q$, це означає, що якщо вхідні сигнали $X_i$ і $X_j$ дорівнюють 1, то виходи $Y_p$ і $Y_q$ дорівнюватимуть 1, а решта виходів видаватимуть 0, і можна по цій дузі перейти до наступного стану. Наприклад, у графа на Рис.1 стан $S_k$ залишається без змін якщо $\bar{X}_1\bar{X}_2 = 1$ і він змінюється в стан $S_k$ якщо $X_1 = 1$, з видачею вихідного сигналу $Y_1 = 1$. Щоб одержати повністю заданий граф станів, в якому наступний стан завжди конкретно заданий для кожної комбінації вхідних сигналів, необхідно дотримуватись наступних обмежень для кожного стану $S_k$:

Якщо $F_i$ і $F_j$ – деяка пара функцій, які навантажують вхідні дуги (Булевські функції) для вершини зі станом $S_k$, тоді

$$F_i \cdot F_j = 0, \ i \neq j.$$

Якщо $n$ дуг виходять з вершини $S_k$, і вони навантажені функціями (мітками) $F_1$, $F_2$,…, $F_n$, відповідно, тоді

$$F_1 \vee F_2 \vee \ldots \vee F_n = 1.$$

Перша умова забезпечує, що в будь-який момент часу виникне не більше однієї умови, що призведе до переходу в стан $S_k$. Друга умова забезпечує, що принаймні одна вихідна дуга буде задіяна, якщо автомат знаходиться в стані $S_k$ і не виникне блокування при будь-якій комбінації вхідних сигналів.

Блок-схема алгоритму або діаграма автомата є альтернативою графу станівwдіаграма. В ній стан представляють вершиною стану. Вона вміщує ім′я стану, за яким через косу лінію "/" задано необов′язковий список вихідних сигналів. Вершина-селектор, або умовна вершина представлена ромбічним символом, задає переходи згідно з результатом булевського виразу в ній. Вершина видачі вихідних сигналів у вигляді прямокутника з закругленими краями вміщує список вихідних сигналів з відповідними умовами їхньої видачі.

Нескладно перетворити граф станів в еквівалентну діаграму автомата. Діаграма, яка еквівалентна графу на Рис.1, має 3 вершини стану. Вихідний сигнал автомата Мура $Y_3$ розміщується в вершині стану $S_k$, так як він не залежить від вхідних сигналів. Виходи автомата Мілі $Y_1$, $Y_2$ видають сигнал при умові відповідних вхідних сигналів, тому вершина, що їх генерує, підключена до відповідного виходу умовної вершини, яка в свою чергу підключена до виходу вершини стану, так як ці сигнали залежать як від стану, так і від вхідного сигналу. Результуюча діаграма автомату показана на Рис.2.

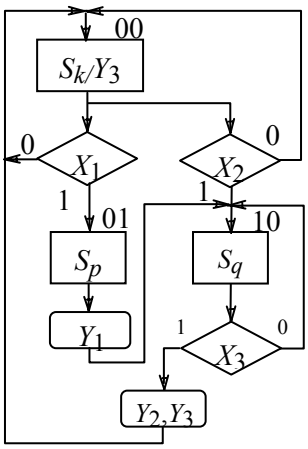

Рис.2

Схема автомата вміщує набір тригерів і логічну схему. Тригери зберігають стан автомату, тобто вони формують регістр стану. Логічна схема складається з двох частин. Одна з них генерує сигнали $D_i$, які є функціями збудження тригерів, а друга –вихідні сигнали $Y_j$.

Спочатку множині станів $S_k$ задають конкретні значення. Є кілька спосіб кодування станів. Вибір способу залежить від числа станів, чи автомат оптимізується по швидкодії, чи апаратним витратам, чи завадостійкості. Унітарне (one-hot) кодування означає, що для $n$ станів автомата вибирається $n$-розрядне слово станів, в якому біт 1 стоїть в $k$-й позиції для кодування стану $S_k$. Наприклад, автомат з графом станів на Рис. 1 матимуть кодовані стани 001, 010, 100. Це кодування використовується якщо число станів невеликеl. Воно забезпечує, як правило найбільшу швидкодію, так як функції збудження тригерів стає досить простою. Якщо граф станів вміщує довгі ланцюги вузлів, регістр станів може бути виконаний на основі регістра зсуву.

Кодування натуральними числами використовується у більшості випадків, особливо, якщо граф станів вміщує довгі ланцюжки вузлів. Для прикладу на Рис.2, таке кодування буде 00, 01 і 10. В цьому випадку регістр стану працює як лічильник. Якщо стани кодуються кодами Грея, тоді в більшості переходів тільки один біт коду стану буде змінюватись. Це служить як для мінімізації логічної схеми, так і для завадостійкості. При комбінованому кодуванні, слово стану розділене на 2 або більше полів, кожне з яких кодується своїм способом. Наприклад, якщо слово стану має 2 $n$-розрядних поля, які мають унітарне кодування, тоді це слово може закодувати до $n^2$ станів.

Після того, як побудовано граф станів або діаграма автомата, тоді автомат може бути описаний на VHDL і тоді його схема може бути синтезована компілятором. При цьому оператор case використовується щоб описати, що відбувається при кожномустані. Кожна умовна вершина відповідає одному оператору if. Наступна програма описує автомат з діаграмою на Рис.2.

```
entity authomata1 is port(C,X1,X2,X3:in bit;          Y1,Y2,Y3:out bit);
end authomata1;
architecture beh of authomata1 is
  сигнал S,D:bit_vector(0 to 1);        -- коди стану
begin
   LN:process(X1,X2,X3,S) begin          --LN model
         Y1<='0';Y2<='0';Y3<='0';  --usual вихідна станів
         case S is
         when "00" => Y3<='1';    -- наявний стан Sk
                  if X1='1' then
                        D<="01"; --наступний стан
                  elsif X2='1' then
                        D<="10";
                  else   D<="00";
                  end if;
         when "01"=> D<="10"; Y1<='1'; -- стан Sp
         when others=> if X3='1' then    -- стан Sq
                        D<="00";Y2<='1';Y3<='1';
                  else    D<="10";
                  end if;
         end case;
  end process;
 RG:process(C) begin              -- стан регістр
         if C='1' and C'event then
                  S <= D; -- update стан on rising дуга of C
         end if;
  end process;
end beh;
```

Перший процес представляє логічну схему автомата, а другий процес – описує регістр стану. Сигнали *Y1, Y2, Y3* видаються у відповідних станах і повинні бути нульовими в інших станах. Найпростіше це зробити встановленням їх в 0 на початку процесу.

4. **Варіанти завдань**

Автомат виконує керування торговим автоматом. Нехай це автомат для продажу деякого продукту, який коштує $N_C$ копійок. Тоді цей продукт видається, якщо сума монет, які кинуті користувачем, дорівнює $N_C$. Номінали монет можуть бути $V_1$**,** $V_2$, і $V_3$. Тому автомат повинен видавати вихідний сигнал OK=1 якщо $N_C = a_1V_1 + a_2V_2 + a_3V_3$, where $a_1$, $a_2$, $a_3 \in \{0,1,2,3,\ldots\}$, DR=1, якщо можна кинути монету (Drop a coin) і ERR=1, якщо набрана сума перевищує задану.

Значення $N_C$ , $V_1$**,** $V_2$, і $V_3$ вибираються з таблиці:

| № варіанту | $N_C$ | $V_1$ | $V_2$ | $V_3$ | № варіанту | $N_C$ | $V_1$ | $V_2$ | $V_3$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 20 | 5 | 10 | - | 12 | 100 | 25 | 50 | - |
| 2 | 30 | 5 | 25 | - | 13 | 110 | 10 | 50 | |
| 3 | 40 | 10 | 25 | - | 14 | 125 | 25 | 50 | |
| 4 | 50 | 10 | 25 | 50 | 15 | 125 | 25 | 50 | 100 |
| 5 | 55 | 10 | 25 | - | 16 | 150 | 25 | 50 | - |
| 6 | 60 | 10 | 25 | - | 17 | 150 | 50 | 100 | - |
| 7 | 60 | 10 | 25 | 50 | 18 | 175 | 25 | 100 | - |
| 8 | 70 | 10 | 25 | - | 19 | 200 | 50 | 100 | - |
| 9 | 70 | 10 | 50 | - | 20 | 250 | 50 | 100 | - |
| 10 | 75 | 25 | 50 | - | 21 | 300 | 50 | 100 | - |
| 11 | 75 | 10 | 25 | - | | | | | |

5. **Приклад розробки автомата**

Нехай задано автомат для продажу з параметрами: $N_C$=25, $V_1$=5, $V_2$=10.

Об′ява об′єкта автомату наступна.

```
entity authomata is
 port(CLK:in BIT; -- синхросигнал
      RST:in BIT;  -- reset
      V1:in BIT;-- кинута монета першого типу
      V2:in BIT;-- кинута монета другого типу
      DR:out BIT;-- команда: "Drop a coin"
      OK:out BIT;-- сума досягнута
      ERR: out BIT);-- сума неправильна
end authomata;
```

**Розробка графу алгоритму**

Граф станів розробляється у відповідності з алгоритмом роботи торгового автомату. Вузли графу станів формують набір рівнів або ступенів. Його перший вузол належить першому ступеню, це стан після початкового встановлення. Другий ступінь формують вершини, які фіксують подію, що перша монета $V_1$ або $V_2$ була кинутаа. Третій ступінь – це вершини, що відповідають подіям, що друга монета $V_1$ або $V_2$ була кинута і т.д. А останній ступінь зформований з вершин, що фіксують те, що потрібна сума досягнута, або ця сума - неправильна. Стани можуть мати імена, що відповідають одержаній сумі, наприклад, якщо сума 10 коп., тоді стан – це N10. Одержаний граф станів намальовано на Рис.3.

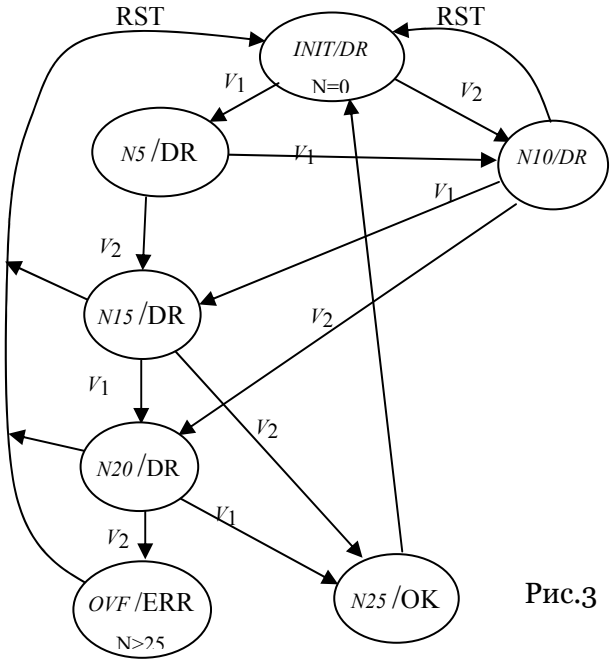


Рис.3

**Опис автомату**

Алгоритм, представлений графом станів на Рис.3 описується в наступній архітектурі.

```
architecture BEH of authomata is
  Type STATE is  (INIT, N5, N10, N15, N20, N25, OVF);--станів of the автомат
  сигнал st : STATE;
begin
  STATES:  process(CLK,RST)      -- автомат process
  begin
        if RST='1' then
              st<=INIT;
        elsif CLK='1' and CLK'event then
              case st is    --наступний автомат стан залежитьing від actual стан і сигнал V1 і V2
                    when INIT => if V1='1' then st<=N5;
                                 elsif V2='1' then st<=N10;
                                 end if;
                    when N5 => if V1='1' then st<=N10;
                                 elsif V2='1' then st<=N15;
                                 end if;
                    when N10 => if V1='1' then st<=N15;
                                 elsif V2='1' then st<=N20;
                                 end if;
                    when N15 => if V1='1' then st<=N20;
                                 elsif V2='1' then st<=N25;
                                 end if;
                    when N20 => if V1='1' then st<=N25;
                                 elsif V2='1' then st<=OVF;
                                 end if;
                    when N25 => st<=INIT;
                    when others => null; --якщо other станів – nothing to do
              end case;
        end if;
  end process;
-- вихідна сигнал
  DR<='1' when st=INIT or  st=N5 or  st=N10 or st=N15 or st=N20 else '0';
  OK<='1' when st = N25 else '0';  -- the sum is achieved
  ERR<='1' when st=OVF else '0';-- the achieved sum is false
end BEH;
```

Одержана модель автомата моделюється, використовуючи стимульовані сигнали. При цьому сигнали RST, V1 і V2 стимулюються призначеними кнопками (режим стимулювання Hotkey). Результуючі графіки показані на Рис.4, і вони показують, що автомат працює правильно.

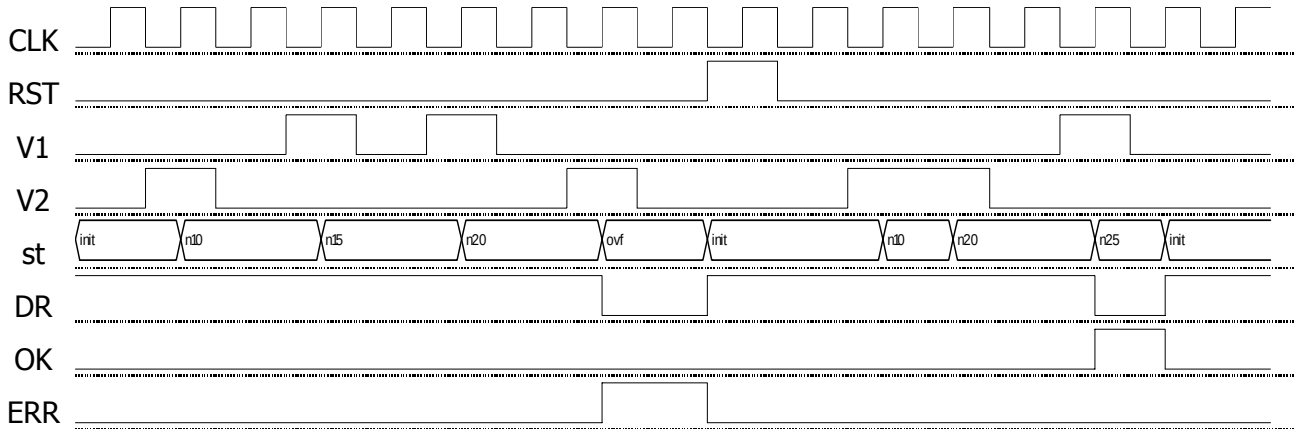


Рис.4.