МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА СИСТЕМНОГО ПРОЕКТУВАННЯ

Розрахунково-графічна робота з курсу: «Технології комп'ютерного проектування»

Виконала:
Студентка ІІІ курсу
Групи ДА-42
Балан Катерина

Київ – 2017

Задание

Спроектировать цифровое устройство в виде асинхронного автомата Мура с проведением анализа результатов, получаемых после каждого этапа проектирования.

Процедура структурного синтеза автоматов предусматривает следующие этапы:

- Исходное задание автомата в виде совмещенной таблицы переходов-выходов;
- Минимизация количества состояний автомата;
- Кодирование состояний автомата;
- Выбор элементов памяти и формирование функций возбуждения элементов памяти (F1) и функций выходов (F2). В качестве элементов памяти необходимо использовать асинхронные R – S триггеры.
- Выбор элементного базиса (ИЛИ – НЕ или И – НЕ) для реализации комбинационной части автомата и преобразование к нему полученных выражений для F1 и F2.

Таблица 1. – Вариант задания (группа ДА-42)

Номер варианта задания	Последовательность входных сигналов X2X1	Последовательность выходных сигналов Y
1	1	4

Таблица 2. – Входные сигналы X_2X_1

Номер варианта	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	00	11	00	11	00	11	00	01	00	01	00	10	00	10	11	10

Таблица 3. – Сигнал выхода Y

Номер варианта	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0

Таблица 4. – Начальное задание для проектирования

Сигнал	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X2	0	1	0	1	0	1	0	0	0	0	0	1	0	1	1	1
X1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	1	0
Y	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0

Определение начальных условий

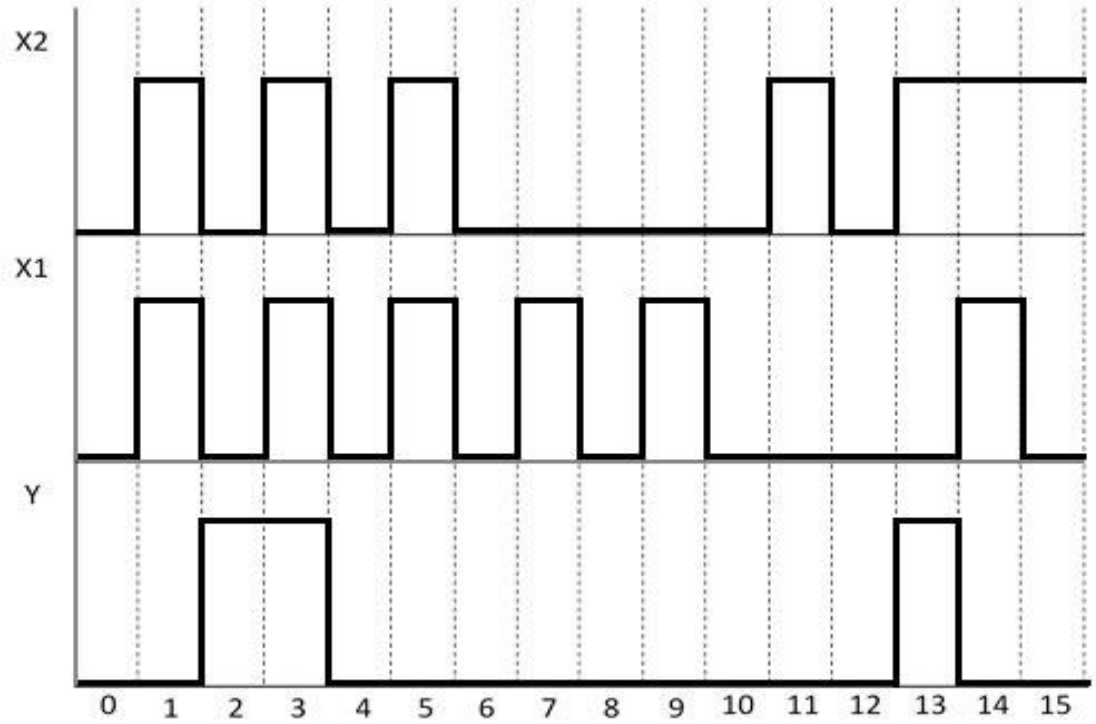


Рисунок 1 – Диаграмма работы устройства

X2 = 0 0 ns, 1 10 ns, 0 20 ns, 1 30 ns, 0 40 ns, 1 50 ns, 0 60 ns, 1 110 ns, 0 120 ns, 1 130 ns
X1 = 0 0 ns, 1 10 ns, 0 20 ns, 1 30 ns, 0 40 ns, 1 50 ns, 0 60 ns, 1 70 ns, 0 80 ns, 1 90 ns, 0 100 ns, 1 140 ns, 0 150 ns

Таблица 5. – Таблица переходов автомата

Q	Y	X2X1			
		00	01	10	11
0	0	(0)	-	-	1
1	0	2	-	-	(1)
2	1	(2)	-	-	3
3	1	4	-	-	(3)
4	0	(4)	-	-	5
5	0	6	-	-	(5)
6	0	(6)	7	-	-
7	0	8	(7)	-	-
8	0	(8)	9	-	-
9	0	10	(9)	-	-
10	0	(10)	-	11	-
11	0	12	-	(11)	-
12	0	(12)	-	13	-
13	1	-	-	(13)	14
14	0	-	-	15	(14)
15	0	0	-	(15)	-

```
entity avtom is
        generic (Lx: integer :=1;Ly: integer :=0;
        K: integer :=15; M: integer :=3;
        TZ: time :=2ns);

         port(
         x: in bit_vector (Lx downto 0);
         y: out bit_vector (Ly downto 0)
            );
  end avtom;

        architecture avtom of avtom is
        function vecint (vec1: bit_vector)
return integer is
variable retval:integer:=0;
begin
        for i in vec1'length-1 downto 1 loop
                if (vec1(i)='1') then retval:=(retval+1)*2;
                else  retval:= retval*2; end if;
        end loop;
        if vec1(0)='1' then retval:=retval+1;
          else null; end if;
        return retval;
        end vecint;

   type stab is array (0 to K, 0 to M) of integer;

 constant Ust: integer :=20;
  constant tab_st: stab :=((0,20,20,1),(2,20,20,1),
  (2,20,20,3),(4,20,20,3),(4,20,20,5),(6,20,20,5),
   (6,7,20,20),(8,7,20,20),
   (8,9,20,20),(10,9,20,20),(10,20,11,20),
   (12,20,11,20),(12,20,13,20),(20,20,13,14),
   (20,20,15,14),(0,20,15,20)
);
type outtab is array (0 to 15) of bit_vector(0 downto 0);
  constant tab_y : outtab :=("0","0","1","1","0","0","0","0",
"0","0","0","0","0","1","0","0");

  signal st, nexst: integer:=0;
  begin

  process
  begin
        wait on x, st;
  if st=Ust then null;
  else
        nexst<=tab_st(st,vecint(x));
        y<=transport tab_y(st)after TZ;
  end if;
  end process;

  process
  begin
     wait on nexst;

  st<= transport nexst after TZ;
  end process;

  end avtom;
```

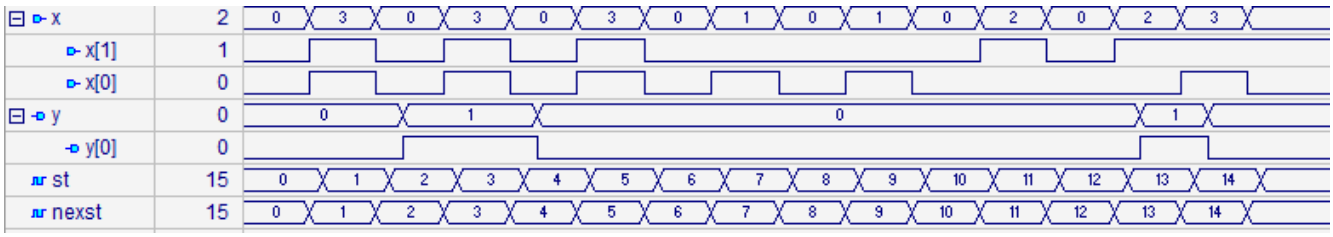


Рисунок 2 – Работа модели, построенной на исходной таблице переходов

Минимизация количества состояний автомата

Определяем эквивалентные устойчивые состояния

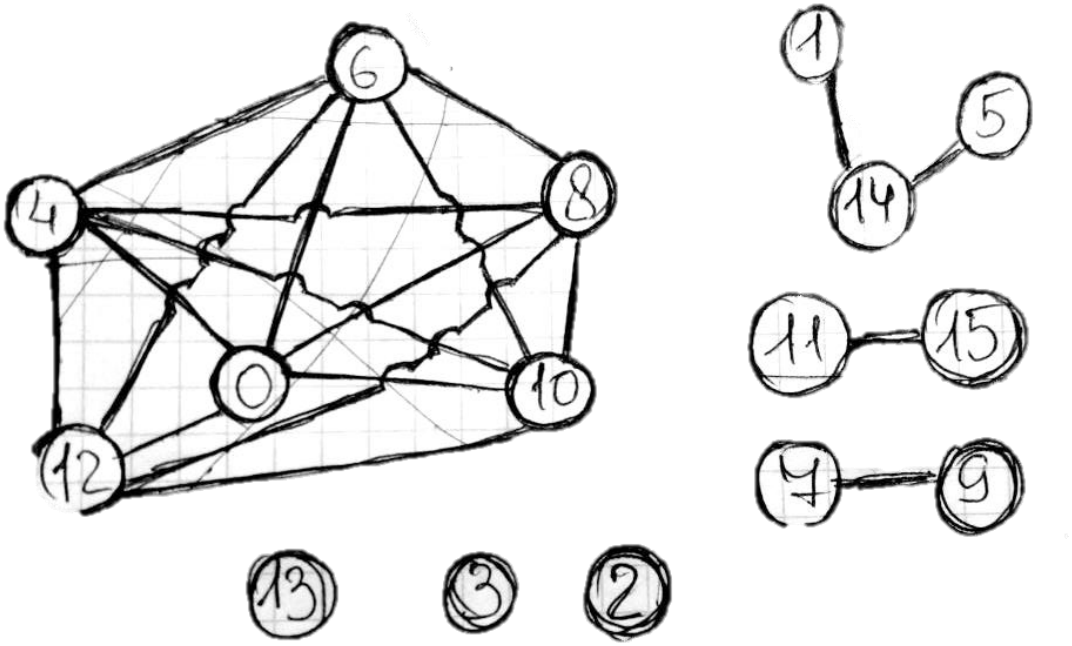
По 1-му и 2-му условиям:

- 0, 4, 6, 8, 10, 12
- 7, 9
- 11, 15
- 1, 5, 14
- 2
- 13
- 3

По 3-му условию:

0-4 => 1-5 => 2-6 -
0-6 => +
0-8 => +
0-10 => +
0-12 => +
4-6 => +
4-8 => +
4-10 => +
4-12 => +
6-8 => 7-9 => 8-10 => +
6-10 => +
6-12 => +
8-10 => +
8-12 => +
10-12 => 11-13 => -
7-9 => 8-10 => +
11-15 => 12-0 => +
1-5 => 2-6 => -
1-14 => +
5-14 => +

Определяем минимальное покрытие состояний автомата



Исходя из анализа полученного графа переписываем таблицу переходов автомата.

Таблица 6. – Таблица переходов автомата после соединения эквив. сост.

Q	OLD	Y	X2X1			
			00	01	10	11
0	0-6-12	0	(0)	7	6	1
1	1	0	2	-	-	(1)
2	2	1	(2)	-	-	3
3	3	1	4	-	-	(3)
4	4-8-10	0	(4)	7	8	5
5	5-14	0	0	-	8	(5)
6	13	1	-	-	(6)	5
7	7-9	0	4	(7)	-	-
8	11-15	0	0		(8)	-

После построения таблицы переходов проверяем ее на соответствие начальным условиям.

```
entity avtom1 is
  generic (Lx: integer :=1;Ly: integer :=0;
  K: integer :=8; M: integer :=3;
  TZ: time :=1ns);

   port(
   x: in bit_vector (Lx downto 0);
   y: out bit_vector (Ly downto 0)
      );
    end avtom1;

  architecture avtom1 of avtom1 is
  function vecint (vec1: bit_vector)
return integer is
variable retval:integer:=0;
begin
  for i in vec1'length-1 downto 1 loop
    if (vec1(i)='1') then retval:=(retval+1)*2;
    else  retval:= retval*2; end if;
  end loop;
  if vec1(0)='1' then retval:=retval+1;
     else null; end if;
  return retval;
  end vecint;

     type stab is array (0 to K, 0 to M) of integer;

  constant Ust: integer :=16;
   constant tab_st: stab :=((0,7,6,1),(2,20,20,1),
   (2,20,20,3),(4,20,20,3),(4,7,8,5),(0,20,8,5),
   (20,20,6,5),(4,7,20,20),
   (0,20,8,20)
);
type outtab is array (0 to 8) of bit_vector(0 downto 0);
  constant tab_y : outtab :=("0","0","1","1","0","0","1","0","0");

  signal st, nexst: integer:=0;
  begin

  process
  begin
        wait on x, st;
  if st=Ust then null;
  else
        nexst<=tab_st(st,vecint(x));
        y<=transport tab_y(st)after TZ;
  end if;
  end process;

  process
  begin
     wait on nexst;

  st<= transport nexst after TZ;
  end process;

  end avtom1;
```

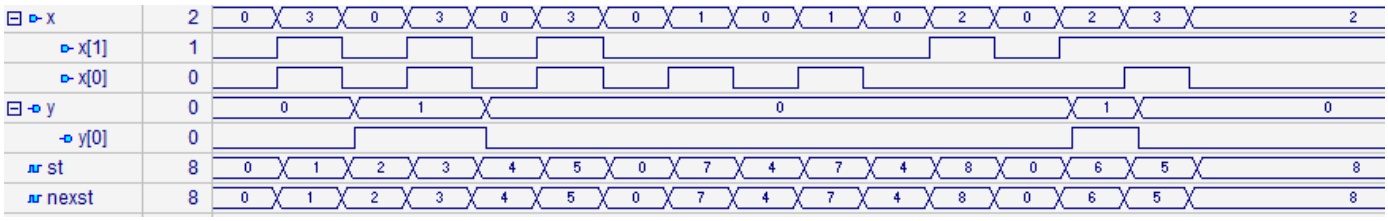


Рисунок 3 – Работа модели, построенной на таблице переходов

После объединения устойчивых состояний для дальнейшего уменьшения количества состояний в таблице объединяем внутренние состояния.

Таблица 7. – Таблица переходов автомата после объединения внутр. состояний

Q	OLD	Y	X2X1			
			00	01	10	11
0	0	0	(0)	4	6	1
1	1	0	2	-	-	(1)
2	2	1	(2)	-	-	3
3	3	1	4	-	-	(3)
4	4-7	0	(4)	(4)	5	5
5	5-8	0	0	-	(5)	(5)
6	6	1	-	-	(6)	5

После построения таблицы переходов проверяем ее на соответствие начальным условиям.

```
entity avtom1 is
  generic (Lx: integer :=1;Ly: integer :=0;
  K: integer :=6; M: integer :=3;
  TZ: time :=1ns);

   port(
   x: in bit_vector (Lx downto 0);
   y: out bit_vector (Ly downto 0)
       );
    end avtom1;

  architecture avtom1 of avtom1 is
  function vecint (vec1: bit_vector)
return integer is
variable retval:integer:=0;
begin
  for i in vec1'length-1 downto 1 loop
    if (vec1(i)='1') then retval:=(retval+1)*2;
    else  retval:= retval*2; end if;
  end loop;
  if vec1(0)='1' then retval:=retval+1;
     else null; end if;
  return retval;
  end vecint;

     type stab is array (0 to K, 0 to M) of integer;

  constant Ust: integer :=20;

   constant tab_st: stab :=((0,4,6,1),(2,20,20,1),
   (2,20,20,3),(4,20,20,3),(4,4,5,5),(0,20,5,5),
   (20,20,6,5)
);
type outtab is array (0 to 6) of bit_vector(0 downto 0);
  constant tab_y : outtab :=("0","0","1","1","0","0","1");

  signal st, nexst: integer:=0;
  begin

  process
  begin
        wait on x, st;
  if st=Ust then null;
  else
        nexst<=tab_st(st,vecint(x));
        y<=transport tab_y(st)after TZ;
  end if;
  end process;

  process
  begin
     wait on nexst;

  st<= transport nexst after TZ;
  end process;

  end avtom1;
```

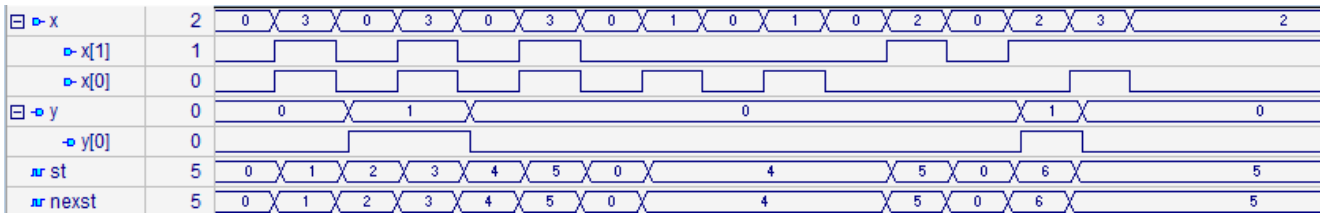


Рисунок 3 – Работа модели, построенной на таблице переходов

Кодирование состояний автомата

Так как состояния автомата будут храниться в триггерах, они должны быть закодированы двоичным кодом. Для асинхронного автомата метод кодирования должен обеспечивать отсутствие критических соревнований триггеров регистра состояний в течении осуществления переходов автомата из одного состояния в другое. Самый простой способ кодирования – код Грея. Для осуществления кодирования нужно сделать превращения таблицы переходов автомата, так как автомат на данном этапе не отвечает требованиям, которые должны осуществляться для соседнего кодирования: существуют замкнутые контуры с непарным количеством состояний. В качестве решения было применено добавление состояний.

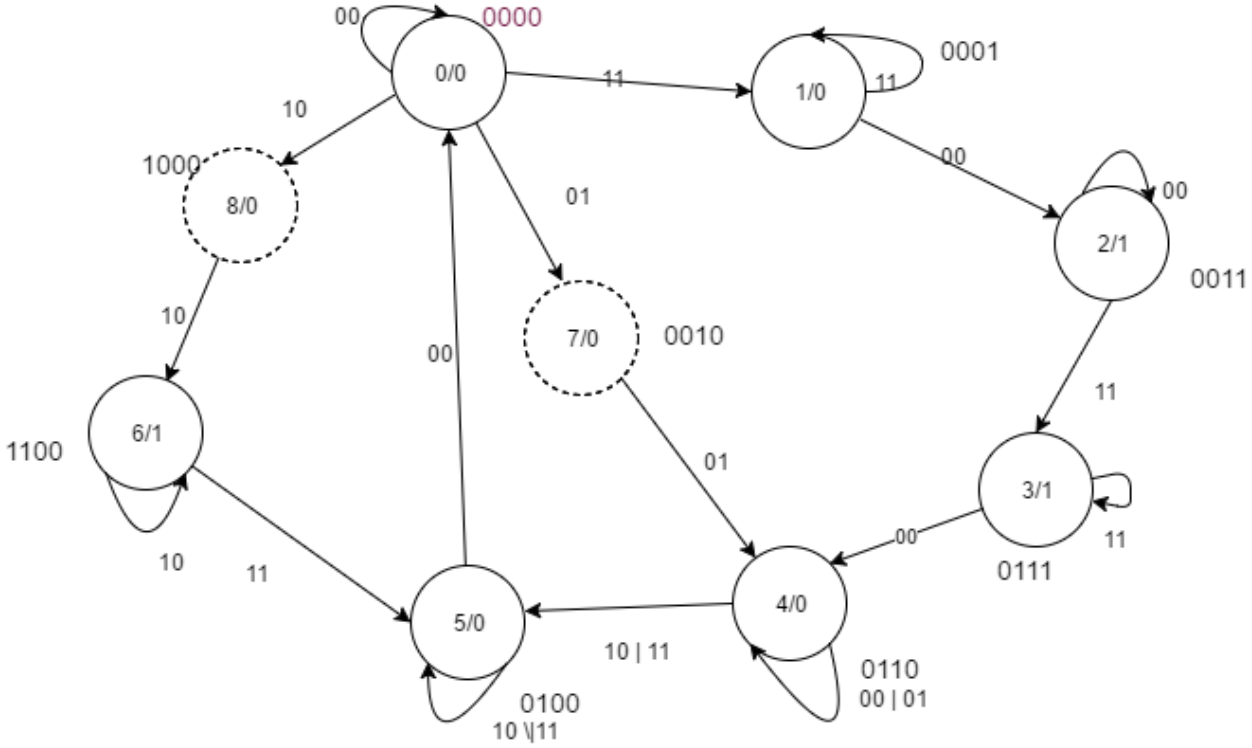


Таблица 8. – Таблица переходов автомата после кодирования

Q	Код Грея	Y	X2X1			
			00	01	10	11
0	0000	0	(0000)	0010	1000	0001
1	0001	0	0011	-	-	(0001)
2	0011	1	(0011)	-	-	0111
3	0111	1	0110	-	-	(0111)
4	0110	0	(0110)	(0110)	0100	0100
5	0100	0	0000	-	(0100)	(0100)
6	1100	1	-	-	(1100)	0100
7	0010	0		0110		
8	1000	0			1100	

После построения таблицы переходов проверяем ее на соответствие начальным условиям.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity avt_code is
    generic (Lin: integer :=1;
    K: integer :=9; M: integer :=3;
    TZ: time :=1ns);

     port(
     x: in bit_vector (Lin downto 0);
     y : inout bit_vector(0 downto 0)
         );
end avt_code;

architecture avt_code of avt_code is

function vecint (vec1: bit_vector)
return integer is
variable retval:integer:=0;
begin
    for i in vec1'length-1 downto 1 loop
        if (vec1(i)='1') then retval:=(retval+1)*2;
        else  retval:= retval*2; end if;
    end loop;
    if vec1(0)='1' then retval:=retval+1;
      else null; end if;
    return retval;
    end vecint;

type stab is array (0 to K, 0 to M) of bit_vector(3 downto 0);

constant tab_st: stab :=(
("0000","0010","1000","0001"),
("0011","1111","1111","0001"),
("0011","1111","1111","0111"),
("0110","1111","1111","0111"),
("0110","0110","0100","0100"),
("0000","1111","0100","0100"),
("1111","1111","1100","0100"),
("1111","0110","1111","1111"),
("1111","1111","1100","1111")
);

type outtab is array (0 to 8) of bit_vector(0 downto 0);
  constant tab_y : outtab :=("0","0","1","1","0","0","1","0","0");

signal st,nexst: bit_vector(3 downto 0):="0000";
begin
 process

 begin
wait on x, st;
if st="1111" then null;
else
nexst<=tab_st(vecint(st),vecint(x));
                                        y<= transport tab_y(vecint(st));

end if;
  end process;
st<= transport nexst after TZ;
end avt_code;
```

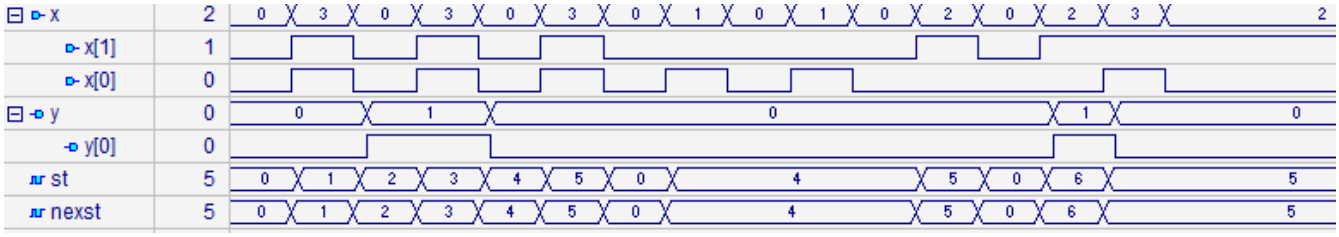


Рисунок 4 – Работа модели, построенной на таблице переходов

Функциональное проектирование

Функции возбуждения элементов памяти зависят от таблицы переходов устройства, выбранного в качестве единицы хранения – асинхронного RS-триггера.

Таблица 9. – Таблица переходов асинхронного RS-триггера.

q(n)	q(n+1)	R(n)	S(n)
0	0	*	0
0	1	0	1
1	0	1	0
1	1	0	*

Переписываем таблицу переходов автомата, чтобы указать какие сигналы должны подаваться на входы триггеров, чтобы переключать их под действием входных сигналов в соответствии с условиями работы.

Таблица 10. – Таблица переходов с учетом сигналов входных воздействий.

№	Код Грея	X1X2																													
		00								01								11								10							
		R3	S3	R2	S2	R1	S1	R0	S0	R3	S3	R2	S2	R1	S1	R0	S0	R3	S3	R2	S2	R1	S1	R0	S0	R3	S3	R2	S2	R1	S1	R0	S0
0	0000	*	0	*	0	*	0	*	0	*	0	*	0	0	1	*	0	*	0	*	0	*	0	0	1	0	1	*	0	*	0	*	0
1	0001	*	0	*	0	0	1	0	*	-	-	-	-	-	-	-	-	*	0	*	0	*	0	0	*	-	-	-	-	-	-	-	-
2	0011	*	0	*	0	0	*	0	*	-	-	-	-	-	-	-	-	*	0	0	1	0	*	0	*	-	-	-	-	-	-	-	-
3	0111	*	0	0	*	0	*	1	0	-	-	-	-	-	-	-	-	*	0	0	*	0	*	0	*	-	-	-	-	-	-	-	-
4	0110	*	0	0	*	0	*	*	0	*	0	0	*	0	*	*	0	*	0	0	*	1	0	*	0	*	0	0	*	1	0	*	0
5	0100	*	0	1	0	*	0	*	0	-	-	-	-	-	-	-	-	*	0	0	*	*	0	*	0	*	0	0	*	*	0	*	0
6	1100	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	0	*	*	0	*	0	0	*	0	*	*	0	*	0
7	0010	-	-	-	-	-	-	-	-	*	0	0	1	0	*	*	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8	1000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	*	0	1	*	0	*	0

Чтобы однозначно определить и минимизировать функции возбуждения элементов памяти и выхода Y, строим карты Карно.

Таблица 11. – Карта Карно для R3

$R3 = x_1$

q0x2x1 / q3q2q1	000	001	011	010	110	111	101	100
000	*	*	*	0	-	*	-	*
001	-	*	-	-	-	*	-	*
011	*	*	*	*	-	*	-	*
010	*	-	*	*	-	-	-	-
110	-	-	1	0	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	0	-	-	-	-

q0x2x1 q3q2q1	000	001	011	010	110	111	101	100
000	0	0	0	1	-	0	-	0
001	-	0	-	-	-	0	-	0
011	0	0	0	0	-	0	-	0
010	0	-	0	0	-	-	-	-
110	-	-	0	*	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	*	-	-	-	-

$$S3 = \overline{Q_2} x_2 \overline{x_1}$$

Таблица 12. – Карта Карно для R2

q0x2x1 q3q2q1	000	001	011	010	110	111	101	100
000	*	*	*	*	-	*	-	*
001	-	0	-	-	-	0	-	*
011	0	0	0	0	-	0	-	0
010	1	-	0	0	-	-	-	-
110	-	-	0	0	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	0	-	-	-	-

$$R2 = \overline{Q_1} \overline{x_2}$$

Таблица 13. – Карта Карно для S2

q0x2x1 q3q2q1	000	001	011	010	110	111	101	100
000	0	0	0	0	-	0	-	0
001	-	1	-	-	-	1	-	0
011	*	*	*	*	-	*	-	*
010	0	-	*	*	-	-	-	-
110	-	-	*	*	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	1	-	-	-	-

$$S2 = Q_3 + Q_0 x_1$$

Таблица 14. – Карта Карно для R1

q0x2x1 q3q2q1	000	001	011	010	110	111	101	100
000	*	0	*	*	-	*	-	0
001	-	0	-	-	-	0	-	0
011	0	0	1	1	-	0	-	0
010	*	-	*	*	-	-	-	-
110	-	-	*	*	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	*	-	-	-	-

$$R1 = \overline{Q_0} x_2$$

Таблица 15. – Карта Карно для S1

q0x2x1 q3q2q1	000	001	011	010	110	111	101	100
000	0	1	0	0	-	0	-	1
001	-	*	-	-	-	*	-	*
011	*	*	0	0	-	*	-	*
010	0	-	0	0	-	-	-	-
110	-	-	0	0	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	0	-	-	-	-

$$S1 = \overline{x_2} x_1 + Q_0 \overline{x_2}$$

Таблица 16. – Карта Карно для R0

q0x2x1 q3q2q1	000	001	011	010	110	111	101	100
000	*	*	0	*	-	0	-	0
001	-	*	-	-	-	0	-	0
011	*	*	*	*	-	0	-	1
010	*	-	*	*	-	-	-	-
110	-	-	*	*	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	*	-	-	-	-

$$R0 = \overline{x_2} Q_2$$

Таблица 17. – Карта Карно для S0

q0x1x0 / q3q2q1	000	001	011	010	110	111	101	100
000	0	0	1	0	-	*	-	*
001	-	0	-	-	-	*	-	*
011	0	0	0	0	-	*	-	0
010	0	-	0	0	-	-	-	-
110	-	-	0	0	-	-	-	-
111	-	-	-	-	-	-	-	-
101	-	-	-	-	-	-	-	-
100	-	-	-	0	-	-	-	-

$$S0 = \overline{Q_2} x_2 x_1$$

Таблица 18. – Карта Карно для Y

q1q0 / q3q2	00	01	11	10
00	0	0	1	0
01	0	-	1	0
11	1	-	-	-
10	0	-	-	-

$$Y = Q_3 Q_2 + Q_1 Q_0$$

После построения функций возбуждения проверяем их на соответствие входным условиям.

```
library IEEE;
use IEEE.Std_Logic_1164.all;
entity asavt_vozb is
generic (Lx: integer :=1;Ly: integer :=0;
Lst: integer :=3) ;
port(
x: in std_logic_vector (Lx downto 0);
y: out STD_LOGIC_vector (Ly downto 0)
);
end asavt_vozb;
architecture avt_vozb of asavt_vozb is
signal r3,s3,r2,s2,r1,s1,r0,s0:STD_LOGIC;
signal Q:STD_LOGIC_vector(Lst downto 0):= "0000";
begin
process
begin
wait on x,Q;
        r3<= x(0) after 0ns;
        s3<= (not Q(2)) and x(1) and (not x(0)) after 0ns;
        r2<= (not Q(1)) and (not x(1)) after 0ns;
        s2<= Q(3) or (Q(1) and x(0));
        --s2<= Q(3) or (not Q(3) and Q(1) and x(0));
        --s2<= Q(3) or (not Q(3) and Q(1) and Q(0) and
x(0)) or ( not Q(3) and Q(1) and not Q(0)) after 0ns;
        r1<= (not Q(0)) and x(1) after 0ns;
        s1<= (Q(0) and (not x(1))) or ((not x(1)) and x(0))
after 0ns;
        r0<= (not x(1)) and Q(2) after 0ns;
        s0<= (not Q(2)) and x(1) and x(0) after 0ns;
end process;
y(0)<=(Q(1) and Q(0)) or (Q(3) and Q(2));

process
begin
wait on r0,s0,r1,s1,r2,s2,r3,s3;
Q(0)<=s0 or (not(r0) and Q(0));
Q(1)<=s1 or (not(r1) and Q(1));
Q(2)<=s2 or (not(r2) and Q(2));
Q(3)<=s3 or (not(r3) and Q(3));
end process;
end avt_vozb;
```

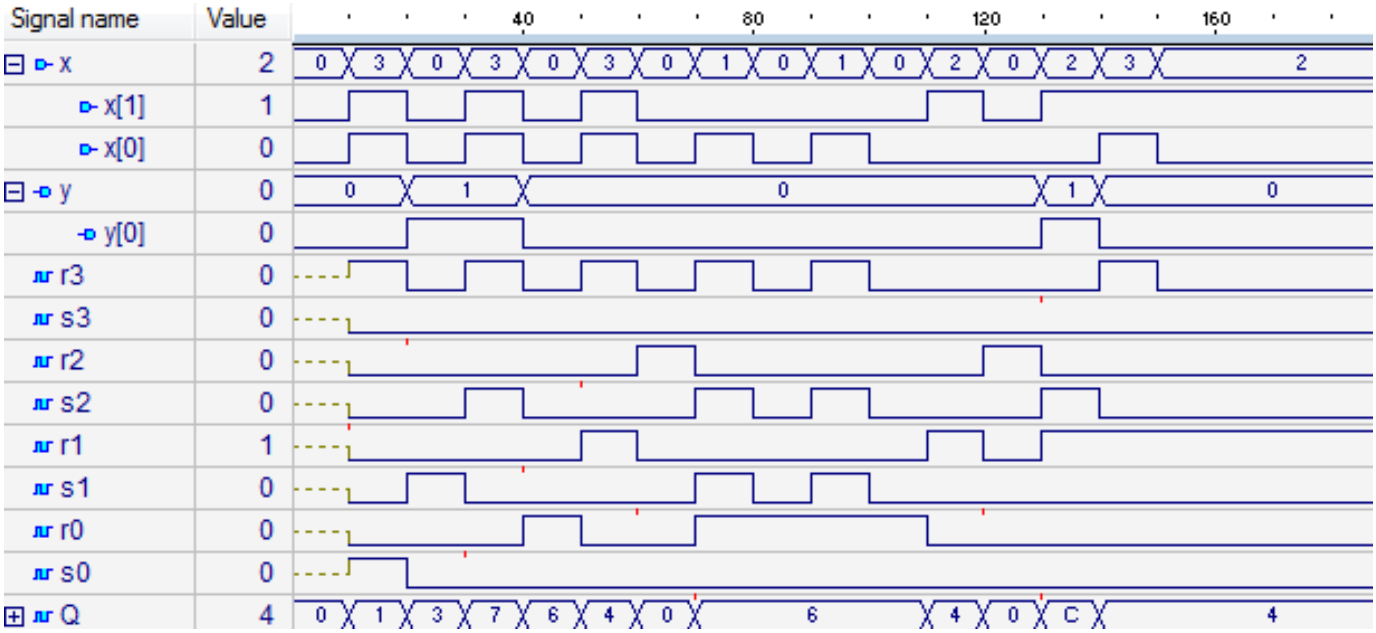


Рисунок 5 – Работа модели

Реализация комбинационной части автомата

На завершающем этапе проектирования можно преобразовать уравнения функций F1 и F2 к нужному элементному базису , взять триггеры с дополнительными входами начальной установки и задать элементам задержки сигналов. На основании этой информации несложно начертить структурную схему автомата, состоящую из перечисленных компонентов, и построить структурную модель. При этом, модель триггера может быть также представлена вентильной структурой. Тогда модель автомата будет соответствовать вентильному (логическому) уровню представления схемы. Если для описания триггера использовать функциональную модель, то получим функционально-логическую модель автомата.

Выбираем ИЛИ-НЕ (nor) базис для реализации автомата

Переход к базису или-нет

$$R_3 = X_1$$
$$S_3 = \bar{Q}_2 X_2 \bar{X}_1 = nor(Q_2, \bar{X}_2, X_1)$$
$$R_2 = \bar{Q}_1 \bar{X}_2 = nor(Q_1, X_2)$$
$$S_2 = Q_3 + Q_1 X_1 = \overline{nor(Q_3, nor(\bar{Q}_1, \bar{X}_1))}$$
$$R_1 = \bar{Q}_0 X_2 = nor(Q_0, \bar{X}_2)$$
$$S_1 = \bar{X}_2 X_1 + Q_0 \bar{X}_2 = \overline{nor(nor(X_2, \bar{X}_1), nor(\bar{Q}_0, X_2))}$$
$$R_0 = \bar{X}_2 Q_2 = nor(X_2, \bar{Q}_2)$$
$$S_0 = \bar{Q}_2 X_2 X_1 = nor(Q_2, \bar{X}_2, \bar{X}_1)$$
$$y = Q_3 Q_2 + Q_1 Q_0 = \overline{nor(nor(\bar{Q}_3, \bar{Q}_2), nor(\bar{Q}_1, \bar{Q}_0))}$$

Проверка модели:

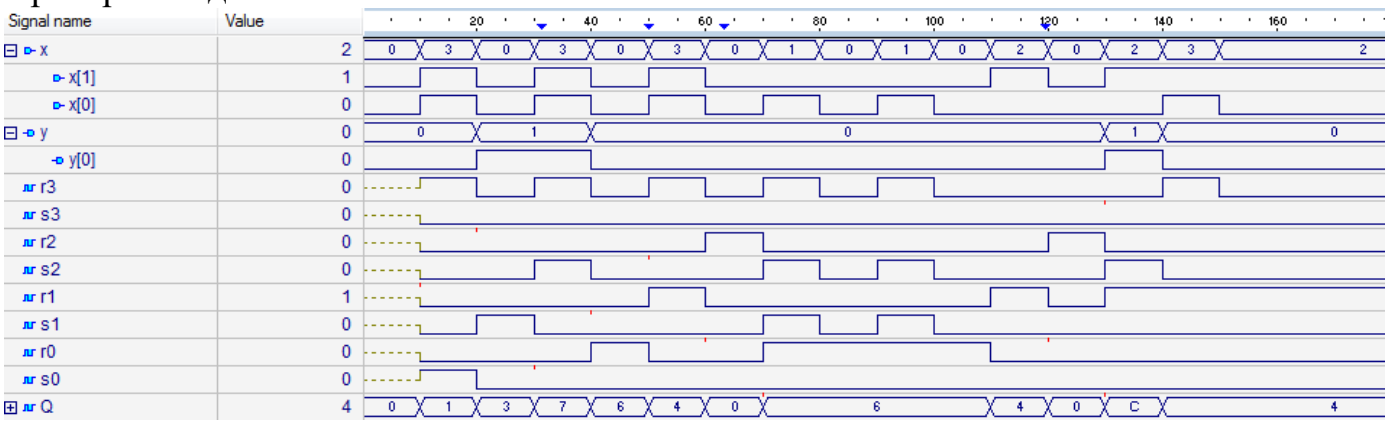


Рисунок 6

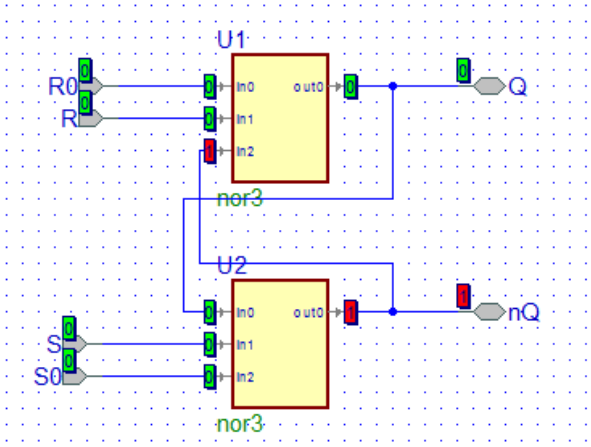
```
library IEEE;
use IEEE.Std_Logic_1164.all;
entity asavt_vozb is
generic (Lx: integer :=1;Ly: integer :=0; -- длина векторов вх. и вых.
переменных
Lst: integer :=3) ; -- длина вектора состояний
port(
x: in std_logic_vector (Lx downto 0);
y: out STD_LOGIC_vector (Ly downto 0)
);
end asavt_vozb;
architecture avt_vozb of asavt_vozb is
signal r3,s3,r2,s2,r1,s1,r0,s0:STD_LOGIC; -- вместо nexst
signal Q:STD_LOGIC_vector(Lst downto 0):= "0000"; -- вместо st с
установкой в начальное состояние
-- Явное описание запрещенного перехода отсутствует
-- Задавать временные диаграммы нужно правильно.
begin
process
begin -- комбинационная схема F1
wait on x,Q;
        r3<= x(0) after 0ns;
        s3<= not (Q(2) or not x(1) or x(0)) after 0ns;
        r2<= not (Q(1) or x(1)) after 0ns;
        s2<= (Q(3) or not (not Q(1) or not x(0))) after 0ns;
        r1<= not (Q(0) or not x(1)) after 0ns; --функции возбуждения
RS- триггеров
        s1<= (not (x(1) or not x(0)) or not (not Q(0) or x(1))) after 0ns;
        r0<= not (x(1) or not Q(2)) after 0ns;
        s0<= not (Q(2) or not x(1) or  not x(0)) after 0ns;
end process;
y(0)<= ( not ( not Q(1) or not Q(0)) or not ( not Q(3) or not Q(2))); --
комбинационная схема F2

process --регистр состояний на асинхронных RS- триггерах
begin
wait on r0,s0,r1,s1,r2,s2,r3,s3;
Q(0)<=s0 or (not(r0) and Q(0)); -- уравнение асинхронного RS-триггера
Q(1)<=s1 or (not(r1) and Q(1));
Q(2)<=s2 or (not(r2) and Q(2));
Q(3)<=s3 or (not(r3) and Q(3));
end process;
end avt_vozb;
```

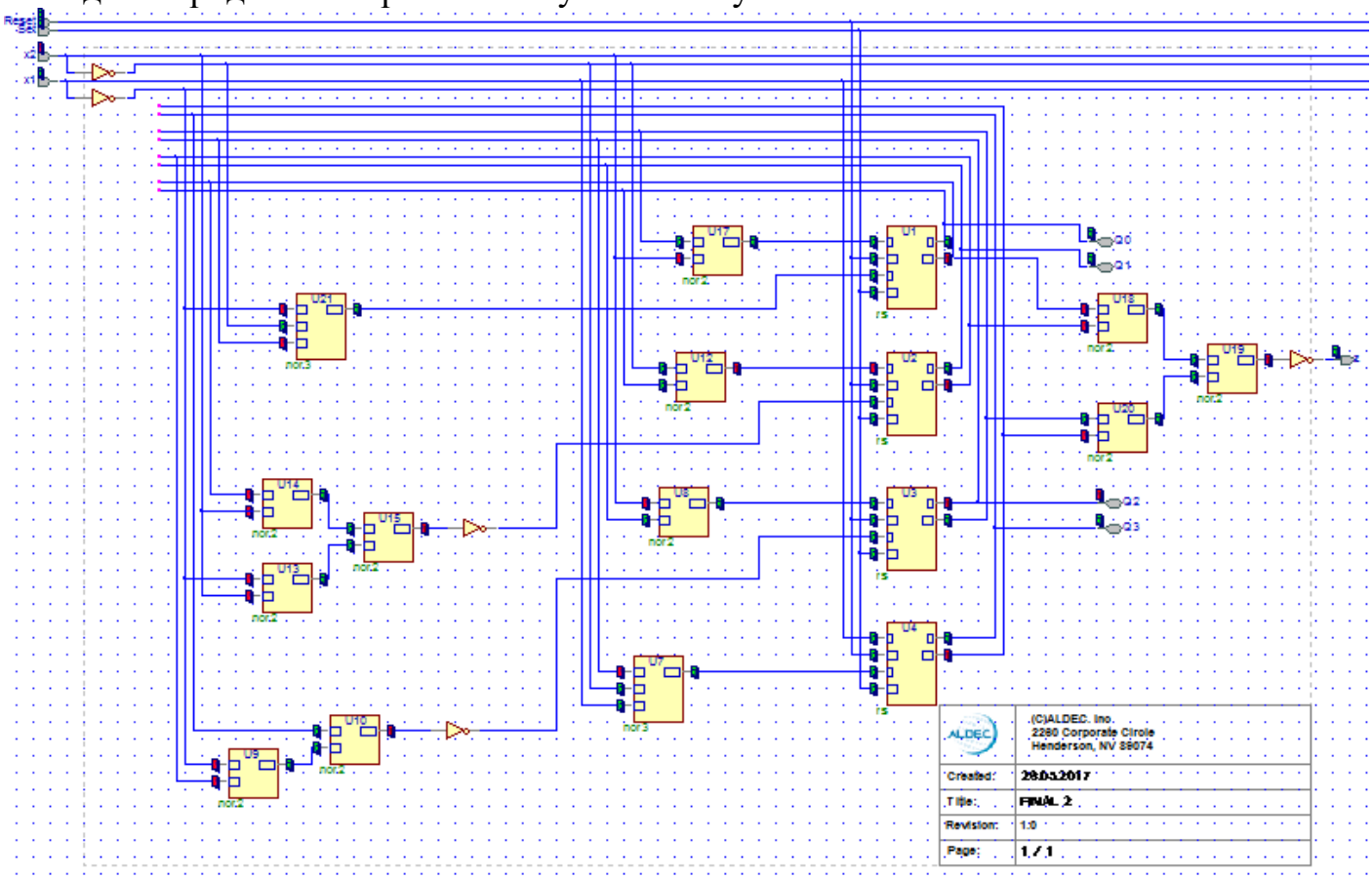
Реализуем нужные элементы:

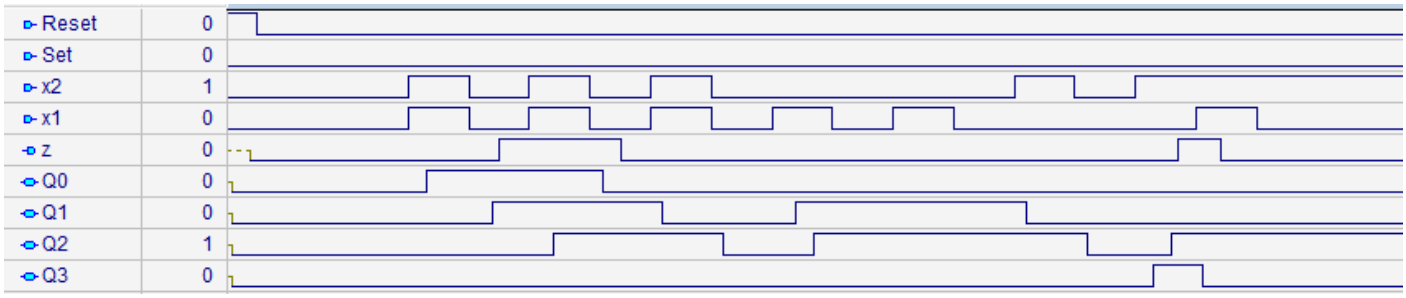
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity nor2 is
        port(
                in0 : in STD_LOGIC;
                in1 : in STD_LOGIC;
                out0 : out STD_LOGIC
            );
end nor2;
architecture nor2 of nor2 is
begin
        out0 <= transport(not(in0 or in1)) after
2ns;
end nor2;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity nor3 is
        port(
                in0 : in STD_LOGIC;
                in1 : in STD_LOGIC;
                in2 : in STD_LOGIC;
                out0 : out STD_LOGIC
            );
end nor3;
architecture nor3 of nor3 is
begin
        out0 <= transport(not(in0 or in1 or in2))
after 2ns;
end nor3;
```



Исходя из проделанной работы получаем схему автомата:





Максимальною є затримка у 14ns. Отже мінімальна припустима довжина такту дорівнює 14ns, а максимальна припустима частота, за якої може функціонувати пристрій дорівнює $1/14*10^{-9}$ = 71.4 МГц.

Висновок

В данной расчетно-графической работе я получила навык построения асинхронного автомата Мура. В процессе построения были выполнены минимизация количества состояний автомата, кодирование его состояний и получение формул F_1 и F_2.

На каждом этапе построения выполнялись проверки правильности расчетов. По результатам, полученным на всех этапах, можем сказать, что построение автомата проходило без ошибок.