Національний Технічний Університет України

«Київський Політехнічний Інститут»

ННК «ІПСА»

Кафедра СП

Розрахунково-графічна робота

з дисципліни

Комп'ютерне проектування

Виконала: студентка гр. ДА-41

Краштан Т. Є.

Київ - 2016

Зміст

1. Теоретичні відомості

Как известно, конечный (абстрактный) автомат определяется как совокупность объектов:

$K=\{X,Y,Q, F1,F2,q_0\}$, где

$X=\{x_1, x_2, \ldots, x_m\}$- множество входных сигналов;

$Y=\{y_1, y_2, \ldots, y_n\}$- множество выходных сигналов;

Q=$\{q_1, q_2, \ldots, q_k\}$- множество внутренних состояний, сохраняемых в памяти автомата;

F1-функция переходов;

F2- функция выходов;

q_0- начальное состояние автомата в момент времени t=0, $q_0 \in Q$.

По способу формирования выходных сигналов различаю **автоматы Мили и Мура**. Функция ***переходов*** всех автоматов одинакова и имеет вид:
Q(t)= F1[X(t),Q(t-1)].

Функция ***выходов*** определяется как:

Y(t)= F2[X(t),Q(t-1)] - для автомата Мили;

Y(t)= F2[Q(t-1)] - для автомата Мура.

Структурная схема автомата Мили показано на рис.1, а Мура – на рис.2. На рисунках функции F1– переходов и F2 – выходов реализуются комбинационными схемами, для хранения внутренних переменных Q, характеризующих состояние автомата, используется РГ – регистр памяти.

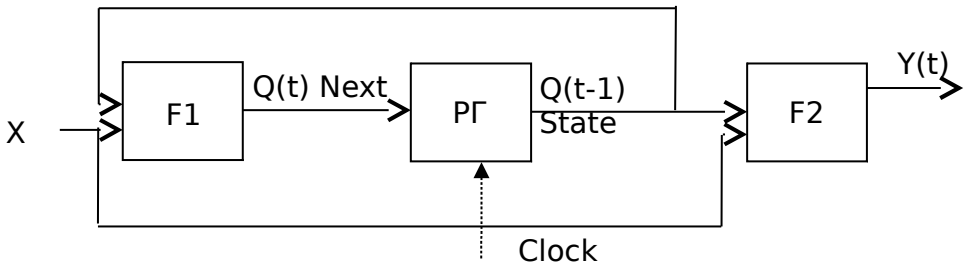


Рис.1.Автомат Мили

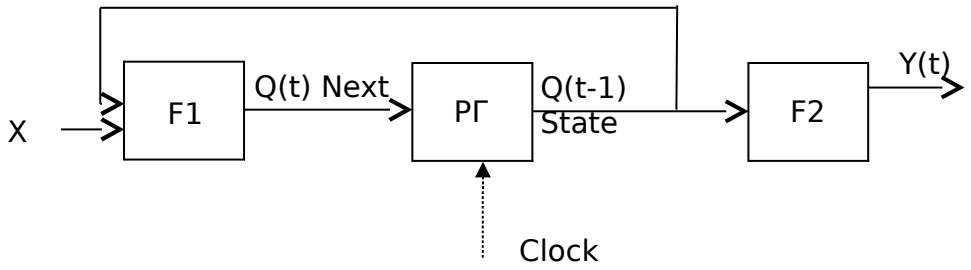


Рис.2. Автомат Мура

Частный случай автомата Мура, когда выходной сигнал Y(t) совпадает с некоторыми значениями $q_i(t-1)$, показан на рис.3.

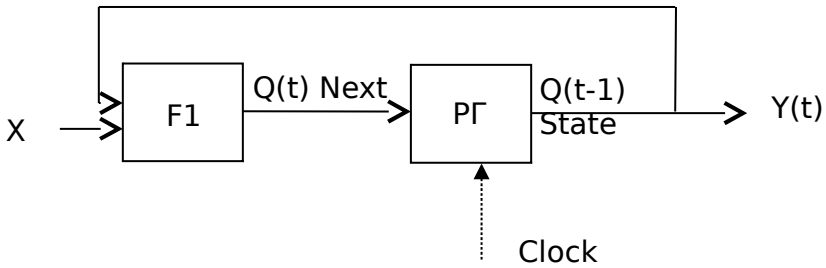


Рис.3. Автомат Мура с вырожденной функцией F2

В зависимости от организации элементов памяти различают синхронные и асинхронные автоматы. Синхронные автоматы отличаются от асинхронных только типом триггеров в РГ (регистре состояний). Смена состояний в синхронных автоматах происходит только по разрешающему сигналу Clock.

Процедура структурного синтеза автоматов предусматривает следующие этапы:

- Исходное задание автомата;
- Минимизация количества состояний автомата;
- Кодирование состояний автомата;
- Выбор элементов памяти и формирование функций выходов (F2) и возбуждения элементов памяти (F1);
- Выбор элементного базиса для реализации комбинационной части автомата и преобразование к нему полученных выражений для F1 и F2.

Существуют различные формы задания конечных автоматов. Для нашей задачи удобно воспользоваться совмещенной таблицей переходов и выходов [2]. Пример таблицы переходов и выходов частично определенного автомата Милли приведен в таб.1., а – автомата Мура в таб.2.

Таблица1.

Состояния автомата	Входные Сигналы	
	x_1	x_2
q_1	q_2/y_1	-
q_2	-	q_4/y_2
q_3	q_2/y_4	-
q_4	q_4/y_3	q_1/y_1

Таблица 2.

Состояния автомата	Выходные сигналы	Входные сигналы	
		x_1	x_2
q_1	y_1	q_2	q_3
q_2	y_1	-	q_4
q_3	y_2	q_2	-
q_4	y_3	q_4	q_1

В клетках таблиц указывается новое состояние, в которое переходит автомат из исходного под воздействием определённого входного сигнала. Прочерки в таблице в зависимости от логики работы автомата означают безразличные или запрещенные переходы. Может подразумеваться, что в данном состоянии автомата некоторые комбинации входных сигналов подаваться либо не могут, либо не должны, либо не имеют значения. В любом случае, состояние, в которое перейдёт автомат, для разработчика безразлично (т.е. может быть любым). Такие автоматы называются *не полностью определёнными*, а безразличные переходы в процессе дальнейшей минимизации количества состояний автомата могут быть легко заменены на любые определённые. Это значительно упрощает процедуру минимизации.

Для автомата Милли в таб.1 после косой черты указывается значение выходного сигнала, который будет выдать автомат в исходном состоянии под воздействием входного сигнала. Для автомата Мура каждому состоянию автомата соответствует определенный выходной сигнал. Не сложно видеть, что таблица 2. легко может быть преобразована к виду таблицы 1.(таб.3).

Таблица 3.

Состояния Автомата	Входные сигналы	
	x_1	x_2
q_1	q_2/y_1	q_3/y_2
q_2	-	q_4/y_3
q_3	q_2/y_1	-
q_4	q_4/y_3	q_1/y_1

В совмещенной таблице переходов/выходов записи для автоматов Милли и Мура будут выглядеть одинаково. Их общая структурная схема приведена на рис.4.

Этапы минимизации количества и кодирования состояний для автоматов Милли и Мура выполняются аналогично. Разница состоит только в составлении исходной таблицы на начальном этапе проектирования и синтезе функции выходов F2 – на конечном этапе.

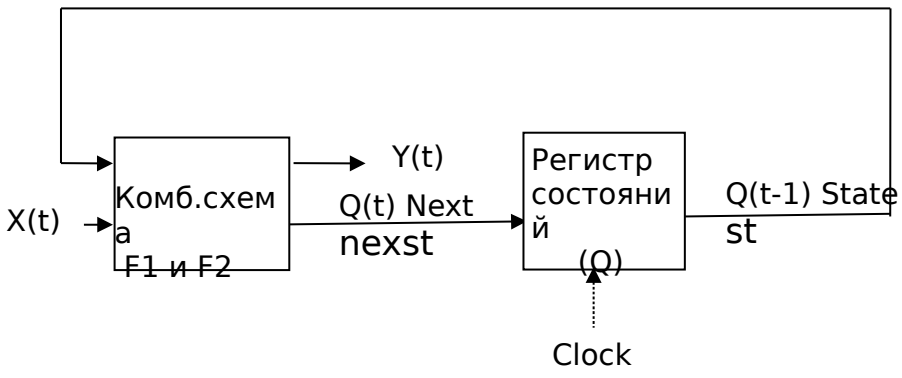


Рис.4. Обобщенное представление конечного автомата

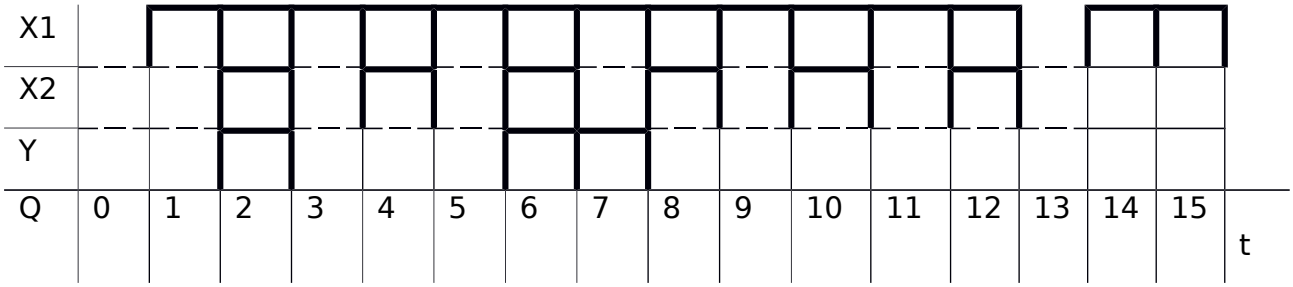
2. Побудова цифрового пристрою

2.1 Задання закону функціонування автомату і його формального опису

Варіант 11:

	Номер такта															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x1x2	00	10	11	10	11	10	11	10	11	10	11	10	11	00	10	10
y	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0

Закон функціонування автомата у формі діаграми:



Суміщена таблиця переходів та виходів:

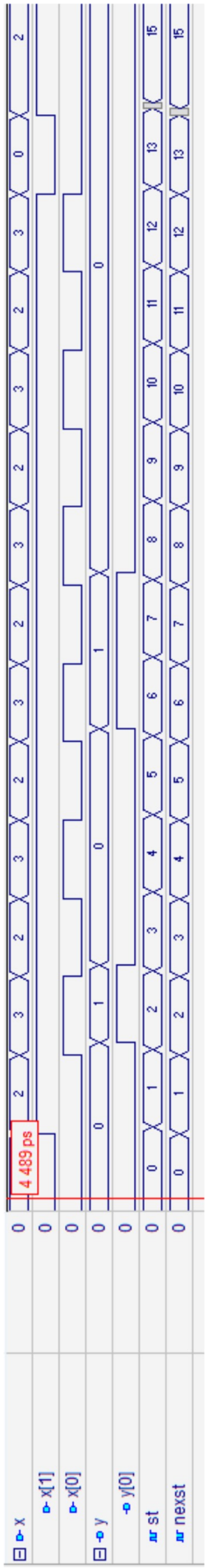
Q	Y	X1\X2			
		00	01	10	11
0	0	(0)	-	1	-
1	0	-	-	(1)	2
2	1	-	-	3	(2)
3	0	-	-	(3)	4
4	0	-	-	5	(4)
5	0	-	-	(5)	6
6	1	-	-	7	(6)
7	1	-	-	(7)	8
8	0	-	-	9	(8)
9	0	-	-	(9)	10
10	0	-	-	11	(10)
11	0	-	-	(11)	12
12	0	13	-	-	(12)
13	0	(13)	-	14	-
14	0	-	-	15	-
15	0	(!)0	-	(15)	-

Модель:

```
entity avtom is
generic (Lx: integer :=1;Ly: integer :=0; -- длина векторов вх. и вых. переменных
K: integer :=15; M: integer :=3; --количество строк и столбцов таб. перех/вых.
TZ: time :=2ns); -- введено для наглядности последовательности переключений


port(
        x: in bit_vector (Lx downto 0); --  входы: в  примере x(0)-C, a x(1)-D
         --  clk: in bit ;              синхровход ( добавить для синхронного автомата !!!)
        y: out bit_vector (Ly downto 0)  --выходы: в  примере Q
   );
   end avtom;

architecture avtom of avtom is
   type stab is array (0 to K, 0 to M) of integer;
   type outtab is array (0 to K, 0 to M) of bit_vector(Ly downto 0) ;

-- функция преобразования bit_vector в integer
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;

  constant Ust: integer :=16;--код запрещенного состояния
  constant tab_st: stab :=((0,3,1,5),(16,16,1,2),(16, 16, 3, 2), (16, 16, 3, 4),
  (16,16,5,4),(16,16,5,6),(16,16,7,6),(16,16,7,8),(16,16,9,8),(16,16,9,10),
```

```
  (16, 16, 11,10),(16,16,11,12),(13,16,16,12),(13,16,14,16),
  (16,16,15,16),(0,16,15,16)); --таблица переходов
  constant tab_y : outtab :=(("0","0","0","0"),("0","0","0","0"),
     ("1","1","1","1"),("0","0","0","0"),("0","0","0","0"),("0","0","0","0"),
     ("1","1","1","1"),("1","1","1","1"),("0","0","0","0"),("0","0","0","0"),
          ("0","0","0","0"),("0","0","0","0"),("0","0","0","0"),("0","0","0","0"),
          ("0","0","0","0"),("0","0","0","0"));--таблица выходов


  signal st, nexst: integer:=0;  --установка начального состояния автомата
  begin


  process  --комбинационная часть автомата
  begin
         wait on x, st;
  if st=Ust then null; -- из запрещенного состояния нет перехода (изменить вх. -
сигналы)
  else
         nexst<=tab_st(st,vecint(x));                        --комбинацонная схема F1
         y<=transport tab_y(st,vecint(x))after TZ;         --комбинацонная схема F2
  end if;
  end process;


  process -- память автомата (регистр состояний)
  begin
     wait on nexst;  --( добавить для асинхронного автомата, убрать для
синхронного !!!)
    --wait on clk; ( добавить для синхронного автомата, убрать для
асинхронного !!!)
  --if pozadge(clk) ='1'  then (добавить для синхронного автомата, убрать для
асинхрон. !!!)
  st<= transport nexst after TZ;
  --end if; (добавить для синхронного автомата, убрать для асинхрон. !!!)
  end process;


  end avtom;
```

Результат моделювання:



2.2 Мінімізація кількості станів автомату

Два стійких стани називаються еквівалентними якщо виконуються 3 умови:
1. Стани є стійкими при одній і тій самій комбінації вхідних сигналів. Формально стійкі стани знаходяться в одному стовпчику.
2. Значення станів виходів у еквівалентних стійких станів мають бути однаковими при однакових комбінацій вхідних сигналів. Для автомата Мура це означає що еквівалентні стани мають мати однакові вихідні сигнали.
3. Для кожного із стійких станів однаковим вхідним послідовностям мають відповідати однакові вихідні послідовності сигналів: тобто послідовності переходів не мають породжувати нееквівалентних пар станів. Формально це означає що в однакових стовпцях таблиці переходів мають бути однакові цифри або цифра і риска або різні цифри, але еквівалентні між собою.

Виписуємо сумісні між собою за умовою 1 і 2 стани у таблицю 2.

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

Перевіримо попарну сумісність станів у кожному стовпчику таб.2 на виконання умови 3. Будемо позначати «+» – сумісні стани, а «-» – несумісні.

0-13 –> 1-15 +

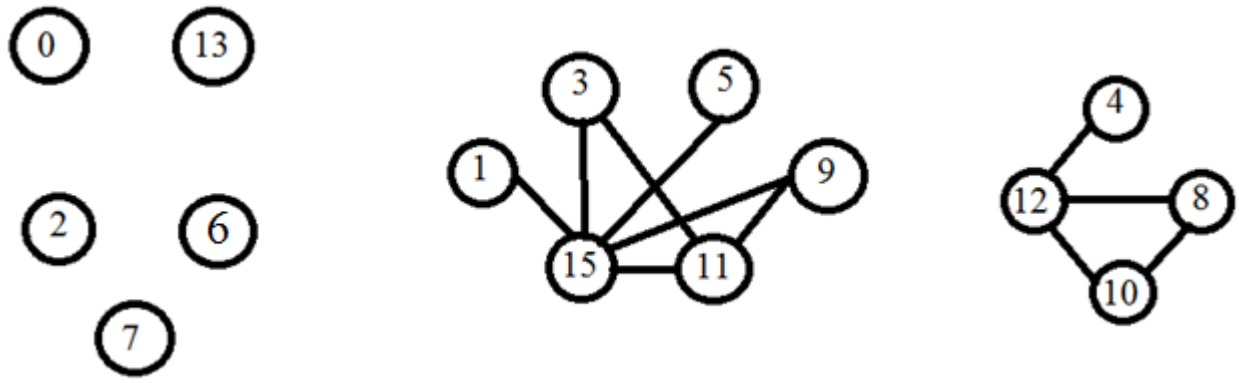
1-3 -> 2-4 –
1-5 -> 2-6 -> 3-7 –
1-9 -> 2-10 –
1-11 -> 2-12 –
1-15 +
3-5 -> 4-6 –
3-9 -> 4-10 -> 5-11 -> 6-12 –
3-11 -> 4-12 +
3-15 +
5-9 -> 6-10 –
5-11 -> 6-12 –
5-15 +
9-11 -> 10-12 +
9-15 +
11-15 +

4-8 -> 5-9 -> 6-10 –
4-10 -> 5-11 -> 6-12 –
4-12 +
8-10 -> 9-11 -> 10-12 +
8-12 +

10-12 +

2-6 -> 3-7 –

Побудуємо граф еквівалентних станів. У вершинах графа – стани, дуги з’єднують кожний стан з усіма іншими, які з ним сумісні.



Можливий варіант покриття: 9-11-15; 8-10-12; 4; 0; 13; 2; 6; 7; 1; 5; 3.

Мінімізована таблиця переходів/виходів автомату

Q	old	y	X1\X2			
			00	01	10	11
0	{9, 11, 15}	0	2	-	(0)	1
1	{8, 10, 12}		3	-	0	(1)
2	{0}		(2)	-	7	-
3	{13}		(3)	-	0	-
4	{2}	1	-	-	9	(4)
5	{6}		-	-	6	(5)
6	{7}		-	-	(6)	1
7	{1}	0	-	-	(7)	4
8	{5}		-	-	(8)	5
9	{3}		-	-	(9)	10
10	{4}		-	-	8	(10)

Об’єднаємо сумісні внутрішні стани автомата:

Q	old1	y	X1\X2			
			00	01	10	11
0	{0}	0	2	-	(0)	1
1	{1, 3}		(1)	-	0	(1)
2	{2, 7}		(2)	-	(2)	4
3	{10}		-	-	8	(3)
4	{4}	1	-	-	7	(4)
5	{5}		-	-	6	(5)
6	{6}		-	-	(6)	1
7	{9}	0	-	-	(7)	3
8	{8}		-	-	(8)	5

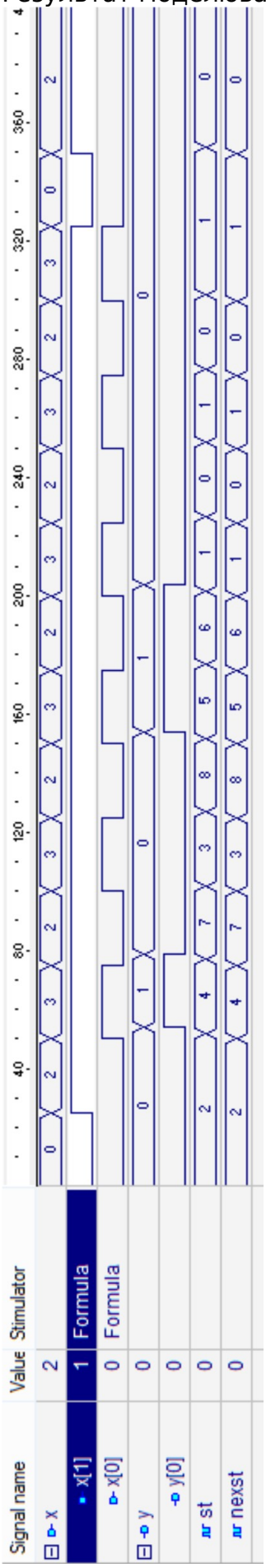
У коді VHDL для попереднього пункту замінюємо розмір і вміст таблиць, а також початковий стан:

K: integer :=8;

(…)

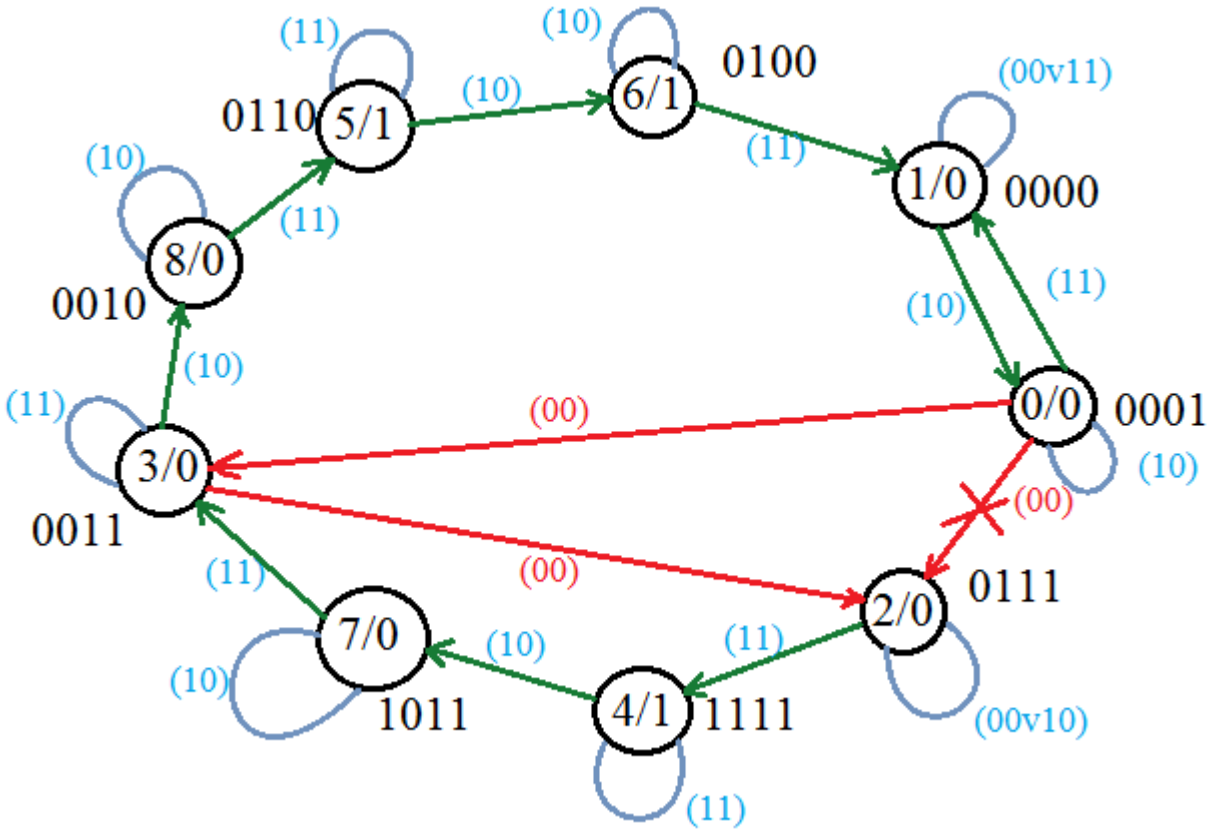
```
  constant tab_st: stab :=((2,16,0,1),(1,16,0,1),(2, 16, 2,4), (16, 16, 8,3),
  (16,16,7,4),(16,16,6,5),(16,16,6,1),(16,16,7,3),(16,16,8,5)); --таблица переходов
  constant tab_y : outtab :=(("0","0","0","0"),("0","0","0","0"),
     ("0","0","0","0"),("0","0","0","0"),("1","1","1","1"),
     ("1","1","1","1"),("1","1","1","1"),("0","0","0","0"),("0","0","0","0"));--таблица
выходов

  signal st, nexst: integer:=2;  --установка начального состояния автомата
```

Результат моделювання:



2.3 Двійкове кодування станів автомату

Граф переходів автомату:



Тут є замкнутий контур з 9 вершин, тому замінимо простий перехід (0->2) складним (0->3->2). Значення виходу при такій заміні не змінюються (0), напрямки стрілок зберігаються.

Таблиця переходів/виходів автомату, закодована двійковим кодом:

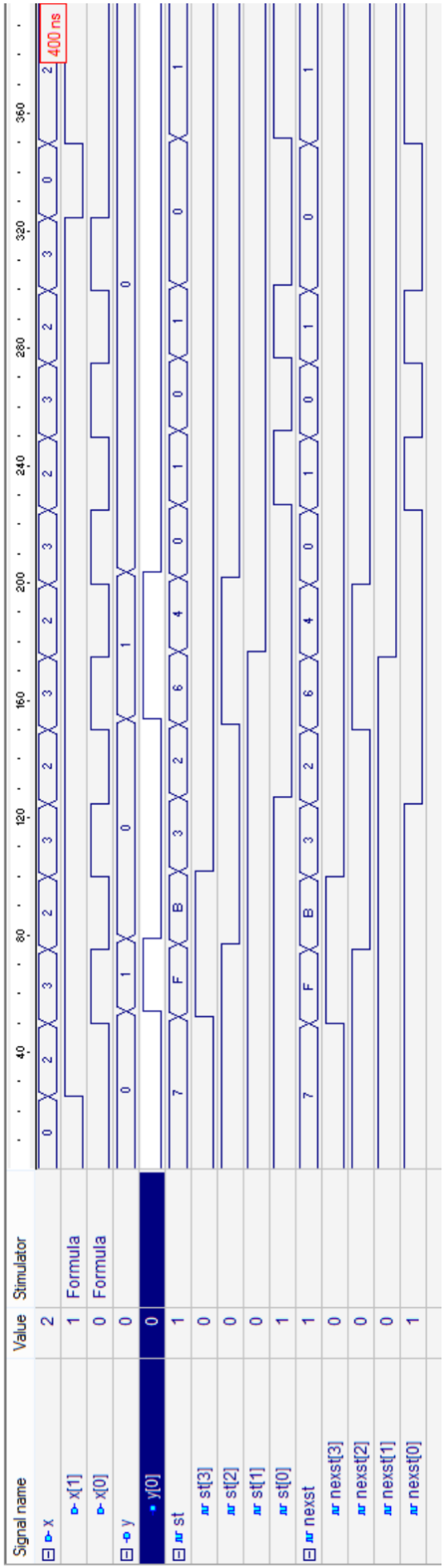
old	Q	y	X1\X2			
			00	01	10	11
0	0001	0	0011	-	(0001)	0000
1	0000		(000	-	0001	(0000)
2	0111		(011	-	(0111)	1111
3	0011		1011	-	0010	(0011)
4	1111	1	-	-	1011	(1111)
5	0110		-	-	0100	(0110)
6	0100		-	-	(0100)	0000
7	1011	0	-	-	(1011)	0011
8	0010		-	-	(0010)	0110

Програма для перевірки:

```
entity avtom is
generic (Lx: integer :=1;Ly: integer :=0; -- äëèíà âåêòîðîâ âõ. è âûõ. ïåðåìåííûõ
K: integer :=8; M: integer :=3; --êîëè÷åñòâî ñòðîê è ñòîëáöîâ òàá. ïåðåõ/âûõ.
TZ: time :=2ns); -- ââåäåíî äëÿ íàãëÿäíîñòè ïîñëåäîâàòåëüíîñòè ïåðåêþ÷åíèé

port(
        x: in bit_vector (Lx downto 0); --  âõîäû: â  ïðèìåðå õ(0)-Ñ, à õ(1)-D
        y: out bit_vector (Ly downto 0)  --âûõîäû: â  ïðèìåðå Q
   );
   end avtom;

architecture avtom of avtom is
        constant LCOD: integer :=3        ;
   type stab is array (0 to K, 0 to M) of bit_vector(LCOD downto 0);
      type codtab is array (0 to K) of bit_vector(LCOD downto 0);
   type outtab is array (0 to K, 0 to M) of bit_vector(Ly downto 0) ;

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;

function vecint1 (st: bit_vector; C:codtab) return integer is
        variable retval:integer:=0;
begin
        for i in C'length-1 downto 1 loop
                if (C(i)=st) then retval:=i;
                end if;
        end loop;
        return retval;
end vecint1;

        constant Ust: bit_vector(LCOD downto 0):="0101";--êîä çàïðåùåííîãî ñîñòîÿíèÿ
     constant Nst: bit_vector(LCOD downto 0):="0111";-- êîä íà÷àëüíîãî ñîñòîÿíèÿ
     constant C: codtab
:=("0001","0000","0111","0011","1111","0110","0100","1011","0010");

  constant tab_st: stab :=(("0011","0101","0001","0000"),
("0000","0101","0001","0000"),("0111","0101","0111","1111"),
  ("1011","0101","0010","0011"),("0101","0101","1011","1111"),
("0101","0101","0100","0110"),
  ("0101","0101","0100","0000"),("0101","0101","1011","0011"),
("0101","0101","0010","0110"));
  constant tab_y : outtab :=(("0","0","0","0"),("0","0","0","0"),
     ("0","0","0","0"),("0","0","0","0"),("1","1","1","1"),
     ("1","1","1","1"),("1","1","1","1"),("0","0","0","0"),("0","0","0","0"));--òàáëèöà
âûõîäîâ
```

```
  signal st,nexst: bit_vector(LCOD downto 0):=Nst; --óñòàíîâêà àâòîìàòà â íà÷àëüíîå
ñîñòîÿíèå

  begin

  process  --êîìáèíàöèîííàÿ ÷àñòü àâòîìàòà
  begin
        wait on x, st;
  if st=Ust then null; -- èç çàïðåùåííîãî ñîñòîÿíèÿ íåò ïåðåõîäà (èçìåíèòü âõ. - ñèãíàëû)
  else
        nexst<=tab_st(vecint1(st,C),vecint(x));                    --êîìáèíàöîííàÿ ñõåìà F1
        y<=transport tab_y(vecint1(st,C),vecint(x))after TZ;          --êîìáèíàöîííàÿ ñõåìà
F2
  end if;
  end process;

  process -- ïàìÿòü àâòîìàòà (ðåãèñòð ñîñòîÿíèé)
  begin
     wait on nexst;  --( äîáàâèòü äëÿ àñèíõðîííîãî àâòîìàòà, óáðàòü äëÿ ñèíõðîííîãî !!!)
    --wait on clk; ( äîáàâèòü äëÿ ñèíõðîííîãî àâòîìàòà, óáðàòü äëÿ àñèíõðîííîãî !!!)
  --if pozadge(clk) ='1'  then (äîáàâèòü äëÿ ñèíõðîííîãî àâòîìàòà, óáðàòü äëÿ
àñèíõðîí. !!!)

  st<= transport nexst after TZ;
  --end if; (äîáàâèòü äëÿ ñèíõðîííîãî àâòîìàòà, óáðàòü äëÿ àñèíõðîí. !!!)
  end process;

  end avtom;
```

Результат моделювання:



2.4 Етап функціонального проектування

Таблиця істинності RS-тригера

R	S	Q_{i-1}	Q
1	0	–	0
0	1	–	1
0	0	0	0
0	0	1	1
1	1	*	*

Таблиця переключень RS-тригера

Q_{i-1}	Q	R	S
0	0	*	0
0	1	0	1
1	0	1	0
1	1	0	*

Таблиця збудження автомату:

Q	y	X1\X2															
		00				01				10				11			
		R_3S_3	R_2S_2	R_1S_1	R_0S_0	R_3S_3	R_2S_2	R_1S_1	R_0S_0	R_3S_3	R_2S_2	R_1S_1	R_0S_0	R_3S_3	R_2S_2	R_1S_1	R_0S_0
000	0	*0	*0	01	0*	-	-	-	-	*0	*0	*0	0*	*0	*0	*0	10
000		*0	*0	*0	*0	-	-	-	-	*0	*0	*0	01	*0	*0	*0	*0
011		*0	0*	0*	0*	-	-	-	-	*0	0*	0*	0*	01	0*	0*	0*
001		01	*0	0*	0*	-	-	-	-	*0	*0	0*	10	*0	*0	0*	0*
111	1	-	-	-	-	-	-	-	-	0*	10	0*	0*	0*	0*	0*	0*
011		-	-	-	-	-	-	-	-	*0	0*	10	*0	*0	0*	0*	*0
010		-	-	-	-	-	-	-	-	*0	0*	*0	*0	*0	10	*0	*0
101	0	-	-	-	-	-	-	-	-	0*	*0	0*	0*	10	*0	0*	0*
001		-	-	-	-	-	-	-	-	*0	*0	0*	*0	*0	01	0*	*0

Карта Карно для функції R_3

R_3	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	*	*	-	-	*	*	*	*
001	-	0	-	-	*	*	*	*
011	-	*	-	-	*	0	*	*
010	-	-	-	-	*	-	-	*
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	0	0	-
101	-	-	-	-	-	1	0	-
100	-	-	-	-	-	-	-	-

$$R3=\acute{Q}2\,X2$$

Карта Карно для функції R_2

R_2	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	*	*	-	-	*	*	*	*
001	-	*	-	-	0	*	*	*
011	-	0	-	-	0	0	0	0
010	-	-	-	-	1	-	-	0
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	0	1	-
101	-	-	-	-	-	*	*	-
100	-	-	-	-	-	-	-	-

$$R2=Q3\acute{X}2\,vQ2\acute{Q}1X2$$

Карта Карно для функції R_1

R_1	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	*	0	-	-	*	*	*	*
001	-	0	-	-	0	0	0	0
011	-	0	-	-	0	0	0	1
010	-	-	-	-	*	-	-	*
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	0	0	-
101	-	-	-	-	-	0	0	-
100	-	-	-	-	-	-	-	-

$$R1=Q2\acute{X}2\acute{Q}0$$

Карта Карно для функції R_0

R_0	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	*	0	-	-	*	1	0	0
001	-	0	-	-	*	0	1	*
011	-	0	-	-	*	0	0	*
010	-	-	-	-	*	-	-	*
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	0	0	-
101	-	-	-	-	-	0	0	-
100	-	-	-	-	-	-	-	-

$$R0=\acute{Q}2X2\acute{Q}1v\acute{Q}3\acute{Q}2Q1X1\acute{X}2$$

Карта Карно для функції S_3

S_3	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	0	0	-	-	0	0	0	0
001	-	1	-	-	0	0	0	0
011	-	0	-	-	0	1	0	0
010	-	-	-	-	0	-	-	0
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	*	*	-
101	-	-	-	-	-	0	*	-
100	-	-	-	-	-	-	-	-

$$S3=\acute{Q}2Q1\acute{X}1 v X2Q0Q2$$

Карта Карно для функції S_2

S_2	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	0	0	-	-	0	0	0	0
001	-	0	-	-	1	0	0	0
011	-	*	-	-	*	*	*	*
010	-	-	-	-	0	-	-	*
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	*	0	-
101	-	-	-	-	-	0	0	-
100	-	-	-	-	-	-	-	-

$$S2=Q1X2\acute{Q}0$$

Карта Карно для функції S_1

S_1	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	0	1	-	-	0	0	0	0
001	-	*	-	-	*	*	*	*
011	-	*	-	-	*	*	*	0
010	-	-	-	-	0	-	-	0
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	*	*	-
101	-	-	-	-	-	*	*	-
100	-	-	-	-	-	-	-	-

$$S1=\acute{X}1Q0$$

Карта Карно для функції S_0

S_0	X1\X2 Q_0							
$Q_3Q_2Q_1$	000	001	011	010	110	111	101	100
000	0	*	-	-	0	0	*	1
001	-	*	-	-	0	*	0	0
011	-	*	-	-	0	*	*	0
010	-	-	-	-	0	-	-	0
110	-	-	-	-	-	-	-	-
111	-	-	-	-	-	*	*	-
101	-	-	-	-	-	*	*	-
100	-	-	-	-	-	-	-	-

$$S0=\acute{Q}2\acute{Q}1X1\acute{X}2$$

Карта Карно для функції Y

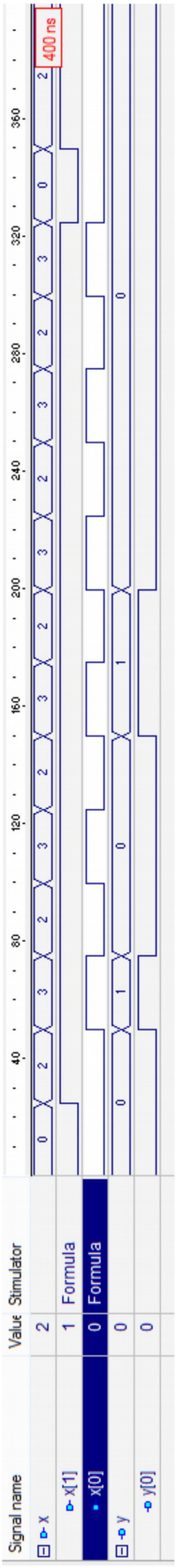
Y	Q_1Q_0			
Q_3Q_2	00	01	11	10
00	0	0	0	0
01	1	-	0	1
11	-	-	1	-
10	-	-	0	-

$$Y=Q2\acute{Q}0\vee Q3Q2$$

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity asavt_vozb is
        generic (Lx: integer :=1;Ly: integer :=0; -- äëèíà âåêòîðîâ âõ. è âûõ. ïåðåìåííûõ
                    Lst: integer :=3) ; -- äëèíà âåêòîðà ñîñòîÿíèé
         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;                -- âìåñòî nexst
signal Q:STD_LOGIC_vector(Lst downto 0):= "0111";        -- âìåñòî st c óñòàíîâêîé â
        íà÷àëüíîå ñîñòîÿíèå
 -- ßâíîå ïîèñàíèå çàïðåùåííîãî ïåðåõîäà îòñóòñòâóåò
-- Çàäàâàòü âðåìåííûå äèàãðàììû íóæíî ïðàâèëüíî.
begin
 process
  begin                          --        êîìáèíàöîííàÿ ñõåìà F1
wait on x,Q;
r3<=(not Q(2)) and x(0);
s3<=((not Q(2)) and Q(1) and (not x(1))) or  (x(0) and Q(0) and Q(2));
r2<=(Q(3) and (not x(0))) or (Q(2) and x(0) and (not Q(0)));
s2<=Q(1) and x(0) and (not Q(0));
r1<=Q(2) and (not x(0)) and (not Q(0)); --ôóíêöèè âîçáóæäåíèÿ RS- òðèããåðîâ
s1<=Q(0) and (not x(1));
r0<=((not Q(2)) and x(0) and (not Q(1))) or ((not Q(3)) and (not Q(2)) and Q(1) and
x(1) and (not x(0)));
s0<=(not Q(2)) and (not Q(1)) and x(1) and (not x(0));
end process;

  y(0)<=(Q(2) and (not Q(0))) or (Q(3) and Q(2));          --        êîìáèíàöîííàÿ ñõåìà F2
```

```
  process --ðåãèñòð ñîñòîÿíèé íà àñèíõðîííûõ RS- òðèããåðàõ
begin
wait on r0,s0,r1,s1,r2,s2,r3,s3;
Q(0)<=s0 or (not(r0) and Q(0)); -- óðàâíåíèå àñèíõðîííîãî RS-òðèããåðà
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;
```

Результат моделювання:



2.5 Етап логічного проектування

nor2.vhd:

```
entity nor2 is
	GENERIC(t : time := 2 ns);
	port (a, b: in bit; c: out bit);
end nor2;

architecture nor2 of nor2 is
signal nor_val: bit;
begin
	c <= a nor b after t;
end nor2;
```

no.vhd:

```
entity no is
	GENERIC(t : time := 2 ns);
	port (a: in bit; b: out bit);
end no;

architecture no of no is
begin
	b <= not a after t;
end no;
```

rs_trig.vhd:

```
entity rs_trig is
	port (R, S, nCLR, Rn, Sn: in bit;
		Q ,nQ: inout bit);
end rs_trig;

architecture rs_trig of rs_trig is
signal r_val, s_val, nRn,nSn, R1,R2,S1,S2, CLR_Rn,CLR_Sn: bit;
begin
	E1:	entity work.no(no) port map(Rn, nRn);
```

```
        E2: entity work.nor2(nor2) port map(nCLR, nRn, CLR_Rn);
        E3: entity work.nor2(nor2) port map(nCLR, CLR_Rn, R1);
        E4: entity work.nor2(nor2) port map(CLR_Rn, R, R2);
        E5: entity work.nor2(nor2) port map(R1, R2, r_val);

        E11: entity work.no(no) port map(Sn, nSn);
        E21: entity work.nor2(nor2) port map(nCLR, nSn, CLR_Sn);
        E31: entity work.nor2(nor2) port map(nCLR, CLR_Sn, S1);
        E41: entity work.nor2(nor2) port map(CLR_Sn, S, S2);
        E51: entity work.nor2(nor2) port map(S1, S2, s_val);

        E6: entity work.nor2(nor2) port map(r_val, nQ, Q);
        E7: entity work.nor2(nor2) port map(s_val, Q, nQ);
end rs_trig;
```

avtom.vhd:

```
entity avtom is
        port (X1, X0, nCLR, Rn3,Sn3,Rn2,Sn2,Rn1,Sn1,Rn0,Sn0: in bit;
        Q3 ,Q2,Q1,Q0: inout bit;
        Y: out bit);
end avtom;

architecture avtom of avtom is
signal R3,S3,R2,S2,R1,S1,R0,S0,nQ3,nQ2,nQ1,nQ0,nX1,nX0,
        sig1,sig2,sig3,sig4,sig5,sig6,sig7,sig8,sig9,sig10,
        sig11,sig12,sig13,sig14,sig15,sig16,sig17,sig18,sig19,sig20,
        sig21,sig22,sig23,sig24,sig25,sig26,pR0: bit;
begin
        E1: entity work.rs_trig(rs_trig) port map(R3,S3,nCLR,Rn3,Sn3, Q3);
        E2: entity work.rs_trig(rs_trig) port map(R2,S2,nCLR,Rn2,Sn2, Q2);
        E3: entity work.rs_trig(rs_trig) port map(R1,S1,nCLR,Rn1,Sn1, Q1);
        E4: entity work.rs_trig(rs_trig) port map(R0,S0,nCLR,Rn0,Sn0, Q0);
        E5:     entity work.no(no) port map(Q3, nQ3);
        E6: entity work.nor2(nor2) port map(nQ3, X0, sig1);
```

E7: entity work.nor2(nor2) port map(sig1, Q2, sig2);

E8: entity work.no(no) port map(X0, nX0);

E9: entity work.nor2(nor2) port map(Q2, nX0, R3);

E10: entity work.nor2(nor2) port map(nX0, Q1, sig3);

E11: entity work.nor2(nor2) port map(sig1, sig3, sig4);

E13: entity work.nor2(nor2) port map(sig2, sig4, R2);

E14: entity work.no(no) port map(Q1, nQ1);

E15: entity work.nor2(nor2) port map(nQ1, Q2, sig5);

E16: entity work.no(no) port map(sig5, sig6);

E17: entity work.nor2(nor2) port map(X1, sig6, sig7);

E18: entity work.nor2(nor2) port map(Q2, sig7, sig8);

E19: entity work.no(no) port map(Q0, nQ0);

E20: entity work.nor2(nor2) port map(nX0, nQ0, sig9);

E21: entity work.nor2(nor2) port map(sig7, sig9, sig10);

E22: entity work.nor2(nor2) port map(sig8, sig10, S3);

E23: entity work.nor2(nor2) port map(nQ0, X1, S1);

E24: entity work.nor2(nor2) port map(nQ1, nX0, sig11);

E25: entity work.no(no) port map(sig11, sig12);

E26: entity work.nor2(nor2) port map(sig12, Q0, S2);

E27: entity work.nor2(nor2) port map(X0, Q0, sig13);

E28: entity work.no(no) port map(sig13, sig14);

E285: entity work.no(no) port map(Q2, nQ2);

E29: entity work.nor2(nor2) port map(nQ2, sig14, R1);

E30: entity work.no(no) port map(X1, nX1);

E31: entity work.nor2(nor2) port map(nQ1, nX1, sig15);

E32: entity work.no(no) port map(sig15, sig16);

E33: entity work.nor2(nor2) port map(Q3, X0, sig17);

E34: entity work.no(no) port map(sig17, sig18);

E35: entity work.nor2(nor2) port map(sig16, sig18, sig19);

E36: entity work.nor2(nor2) port map(Q1, nX0, sig20);

E37: entity work.nor2(nor2) port map(sig19, sig20, pR0);

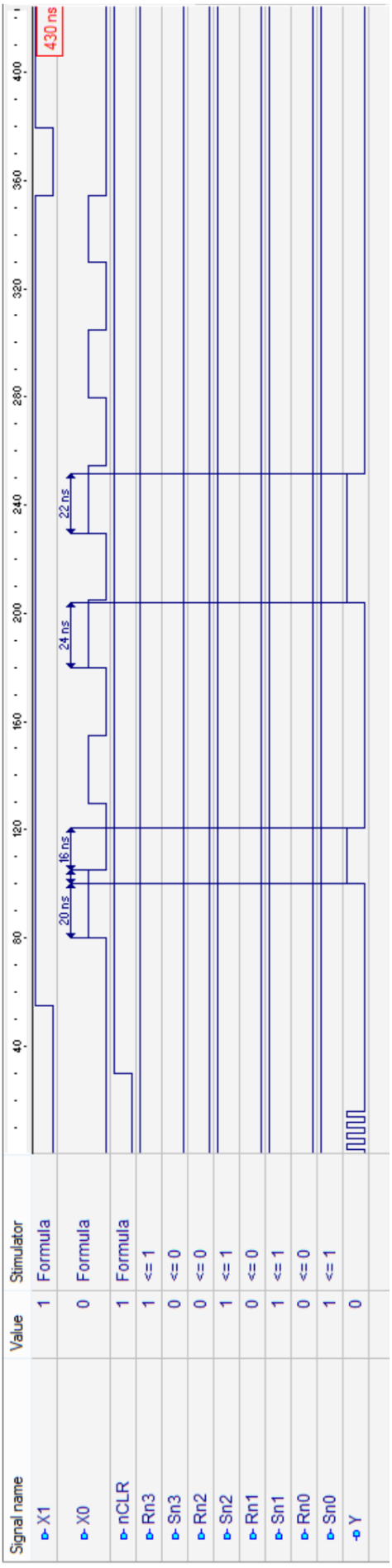
E375: entity work.nor2(nor2) port map(Q2, pR0, R0);

E38: entity work.nor2(nor2) port map(nX1, X0, sig21);

E39: entity work.nor2(nor2) port map(Q2, Q1, sig22);

```
        E40: entity work.no(no) port map(sig22, sig23);
        E41: entity work.nor2(nor2) port map(sig21, sig23, S0);
        --E42: entity work.no(no) port map(Q2, nQ2);
        E43: entity work.nor2(nor2) port map(nQ2, Q0, sig24);
        E44: entity work.nor2(nor2) port map(Q3, sig24, sig25);
        E45: entity work.nor2(nor2) port map(Q2, sig24, sig26);
        E46: entity work.nor2(nor2) port map(sig25, sig26, Y);
end avtom;
```

Результат моделювання із виміряними затримками розповсюдження сигналів від входів до виходу:



Як видно зі скріншота, максимальною є затримка у 24ns. Отже мінімальна припустима довжина такту дорівнює 24ns, а максимальна припустима частота, за якої може функціонувати пристрій дорівнює $1/24*10^{-9}$ = 41.7 МГц.

Висновки

У ході виконання даної розрахунково-графічної роботи було побудовано асинхронний цифровий автомат Мура. Використовувалося низхідне проектування, що включало кілька етапів:

- вихідне задання автомату у вигляді суміщеної таблиці вхідних/вихідних станів
- мінімізація кількості станів автомата
- кодування станів автомата
- формування функцій збудження елементів пам’яті (R – S тригерів) F1 та функції виходу F2
- Вибір елементного базису (було обрано елементи АБО-НІ) для реалізації комбінаційної частини автомата та тригерів, перетворення до нього отриманих на минулому етапі функцій

На кожному етапі проводилася верифікація результатів. По завершенні проектування було виміряно затримки проходження сигналів від входів до виходів та обчислено максимальну допустиму частоту роботи пристрою.