НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І. СІКОРСЬКОГО»

ННК «ІПСА»

**РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА**

З КУРСУ

«ТЕХНОЛОГІЇ КОМП’ЮТЕРНОГО ПРОЕКТУВАННЯ»

ТЕМА: МОДЕЛЮВАННЯ ЦИФРОВИХ АВТОМАТІВ

Виконав студент групи ДА-42

Стогнієнко В.В.

КИЇВ 2017

Варіант 21 (9, 5)

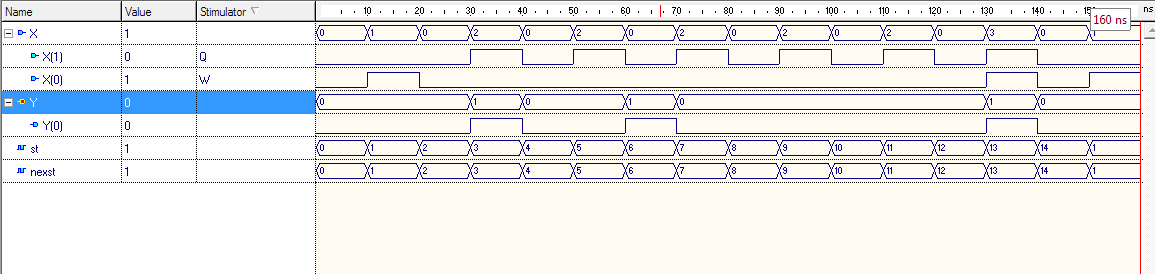
Вхідні і вихідні сигнали:

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

Таблиця переходів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | Y | 00 | 01 | 10 | 11 |
| 0 | 0 | (0) | 1 |  |  |
| 1 | 0 | 2 | (1) |  |  |
| 2 | 0 | (2) |  | 3 |  |
| 3 | 1 | 4 |  | (3) |  |
| 4 | 0 | (4) |  | 5 |  |
| 5 | 0 | 6 |  | (5) |  |
| 6 | 1 | (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 | 14 |  |  | (13) |
| 14 | 0 | (14) | 15 |  |  |
| 15 | 0 | 0 | 15 |  |  |

Перевіримо на правильність програмно:



library IEEE;ц

use IEEE.STD\_LOGIC\_1164.all;

entity a1ent is

generic (TZ: time :=0ns);

port(

X: in bit\_vector (1 downto 0);

Y: out bit\_vector(0 downto 0)

);

end a1ent;

architecture a1arc of a1ent is

-- bit\_vector to 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;

type stab is array (0 to 15, 0 to 3) of integer;

constant tab\_st: stab :=(

(0,1,16,16)

,(2,1,16,16)

,(2,16,3,16)

,(4,16,3,16)

,(4,16,5,16)

,(6,16,5,16)

,(6,16,7,16)

,(8,16,7,16)

,(8,16,9,16)

,(10,16,9,16)

,(10,16,11,16)

,(12,16,11,16)

,(12,16,16,13)

,(14,16,16,13)

,(14,15,16,16)

,(0,15,16,16)

);

type outtab is array (0 to 15) of bit\_vector(0 downto 0);

constant tab\_y : outtab :=("0","0","0","1","0","0","1","0", "0","0","0","0","0","1","0","0");

signal st,nexst: integer:=0;

begin

process begin

wait on X, st;

if st=16 then null;

else

nexst<=tab\_st(st,vecint(X));

Y<=transport tab\_y(st)after TZ;

end if;

end process;

st<= transport nexst after TZ;

end a1arc;

Розіб’ємо на групи і перевіримо на еквівалентність для подальшої мінімізації:

0, 2, 4, 6, 8, 10, 12, 14

1, 15

3, 5, 7, 9, 11

13

З однаковими значеннями виходів:

0, 2, 4, 8, 10, 12, 14

1, 15

6

5, 7, 9, 11

3

13

0,2+

0,4+

0,8+

0,10+

0,12+

0,14+

2,4-

2,8-

2,10-

2,12-

2,14+

4,8-

4,10-

4,12+

4,14+

5,7-

5,9-

5,11-

8,10+

8,12+

8,14+

10,12+

10,14+

12,14+

5,7-

5,9-

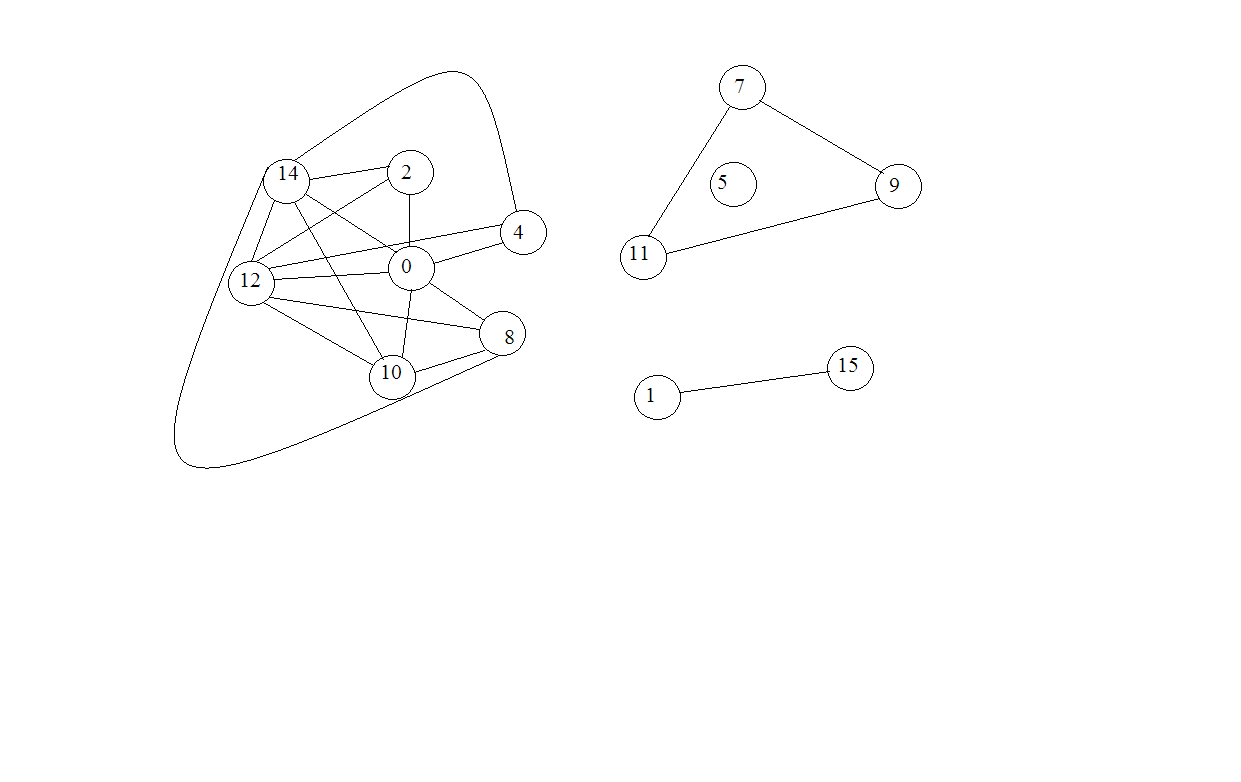
5,11-

7,9+

7,11+

9,11+

Перевіривши на еквівалентність побудуємо граф переходів:



Об’їєднаємо такі стани: 0,2,14 1,15 7,9,11 8,10,12

Далі, побудуємо спрощену таблицю переходів:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q | Q0 | 00 | 01 | 10 | 11 | Y |
| 0 | 0,2,14 | (0) | 1 | 2 |  | 0 |
| 1 | 1,15 | 0 | (1) |  |  | 0 |
| 2 | 3 | 3 |  | (2) |  | 1 |
| 3 | 4 | (3) |  | 4 |  | 0 |
| 4 | 5 | 5 |  | (4) |  | 0 |
| 5 | 6 | (5) |  | 6 |  | 1 |
| 6 | 7,9,11 | 7 |  | (6) |  | 0 |
| 7 | 8,10,12 | (7) |  | 6 | 8 | 0 |
| 8 | 13 | 0 |  |  | (8) | 1 |

Мінімізована таблиця переходів:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q1 | Q | 00 | 01 | 10 | 11 | Y |
| 0 | 0,1 | (0) | (0) | 1 |  | 0 |
| 1 | 2 | 2 |  | (1) |  | 1 |
| 2 | 3 | (2) |  | 3 |  | 0 |
| 3 | 4 | 4 |  | (3) |  | 0 |
| 4 | 5 | (4) |  | 5 |  | 1 |
| 5 | 6,7 | (5) |  | (5) | 6 | 0 |
| 6 | 8 | 0 |  |  | (6) | 1 |

Перевіримо остаточну мінімізовану таблицю програмно:

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity avtomat is

generic (

TZ: time :=0ns);

port(

x : in STD\_LOGIC\_VECTOR(1 downto 0);

y : inout STD\_LOGIC

);

end avtomat;

--}} End of automatically maintained section

architecture avtomat of avtomat is

function vecint (vec1: std\_logic\_vector)

return integer is

variable retval:integer:=0;

begin

for i in vec1'length-1 downto 0 loop

if (vec1(i)='1') then retval:=retval\*2+1;

else retval:= retval\*2; end if;

end loop;

return retval;

end vecint;

type stab is array (0 to 6, 0 to 3) of integer;

type outtab is array (0 to 6) of std\_logic;

constant tab\_st: stab :=(

(0,0,1,20)

,(2,20,1,20)

,(2,20,3,20)

,(4,20,3,20)

,(4,20,5,20)

,(5,20,5,6)

,(0,20,20,6));

Constant tab\_z : outtab :=('0','1','0','0','1','0','1');

signal st,nexst: integer:=0;

begin

process

begin

y<=tab\_z(st);

wait on x, st;

if st = 20 then null;

else

nexst<=tab\_st(st,vecint(x));

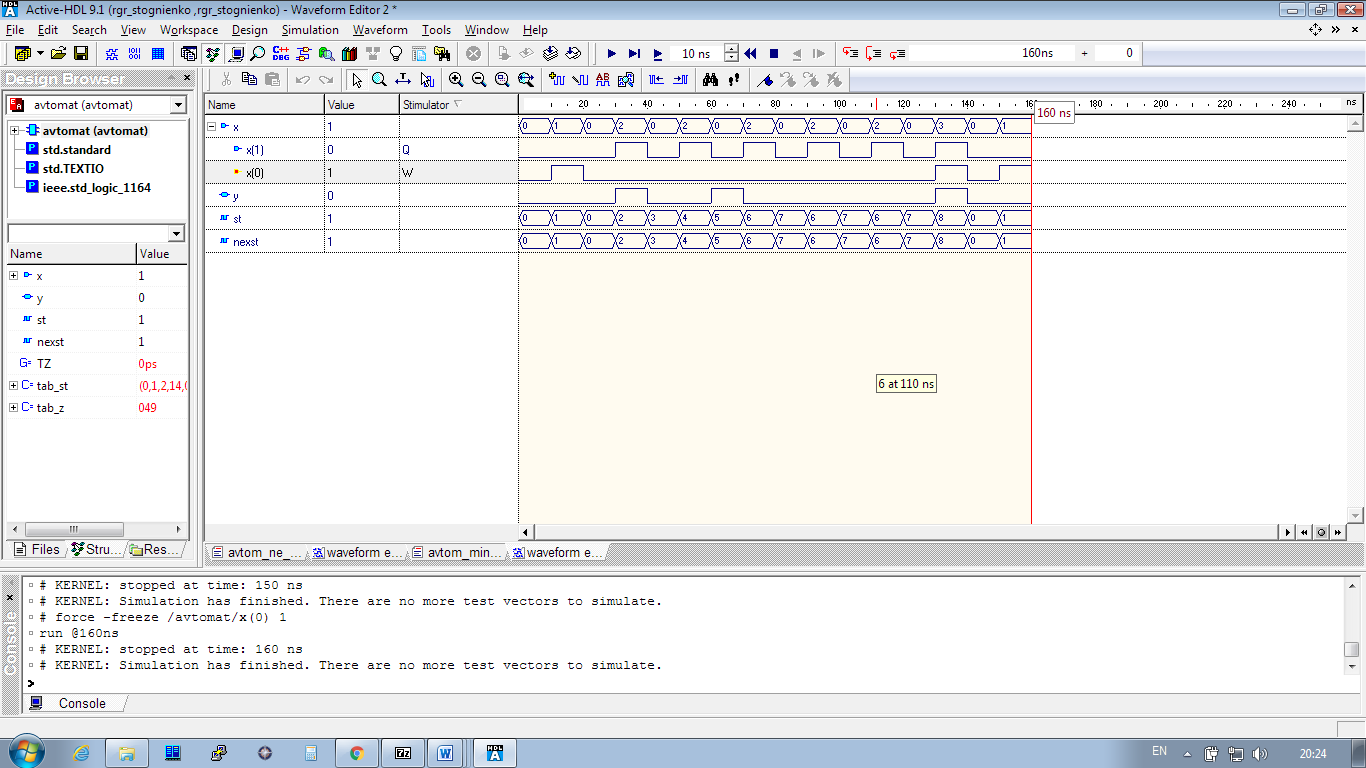
wait for TZ;

end if;

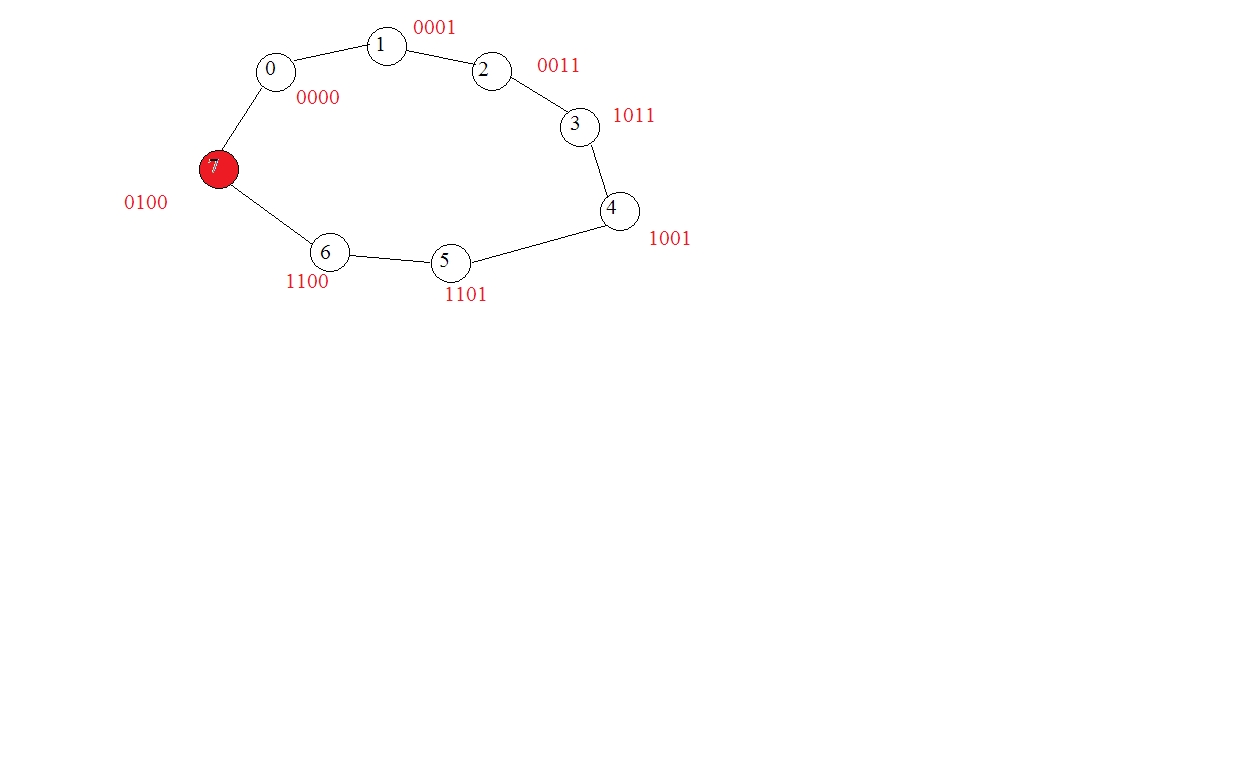
end process;

st<= transport nexst after TZ;

end avtomat;

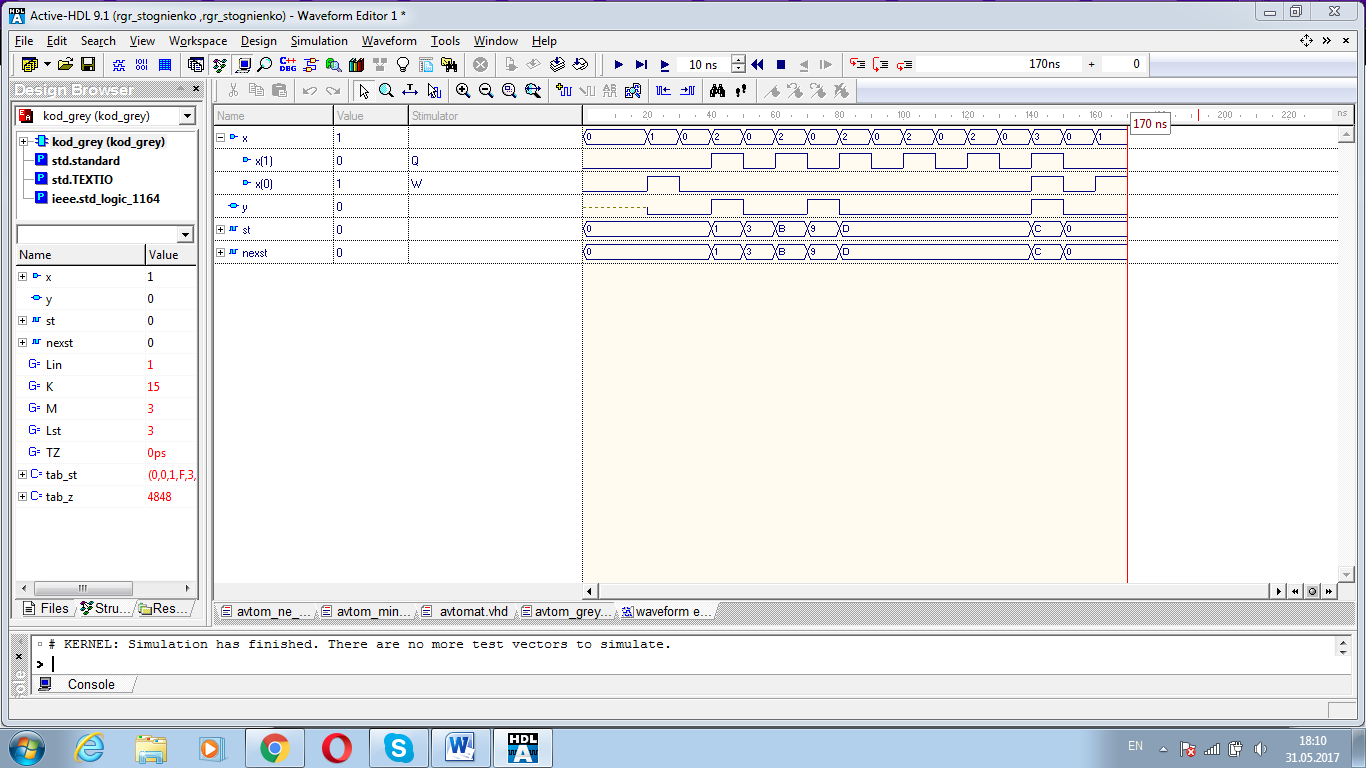


Далі побудуємо граф станів автомату і закодуємо стани кодом Грея для упередження гонки сигналів. Оскільки граф мав непарний цикл то додали 7 стан:



Побудуємо таблицю закодованих станів переходів і також перевіримо її програмно

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 00 | 01 | 10 | 11 | Y |
| 0000 | (0000) | (0000) | 0001 |  | 0 |
| 0001 | 0011 |  | (0001) |  | 1 |
| 0011 | (0011) |  | 1011 |  | 0 |
| 1011 | 1001 |  | (1011) |  | 0 |
| 1001 | (1001) |  | 1101 |  | 1 |
| 1101 | 1101 |  | (1101 | 1100 | 0 |
| 1100 | 0100 |  |  | (1100) | 1 |
| 0100 | 0000 |  |  |  | 1 |



library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity kod\_grey is

generic (Lin: integer :=1;

K: integer :=15; M: integer :=3;

Lst : integer := 3;

TZ: time :=0ns);

port(x: in bit\_vector (Lin downto 0);y : inout STD\_LOGIC);

end kod\_grey;

architecture kod\_grey of kod\_grey 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(Lst downto 0);

type outtab is array (0 to K) of std\_logic;

constant tab\_st: stab :=(

("0000","0000","0001","1111"), --0000-0

("0011","1111","0001","1111"), --0001-6

("1111","1111","1111","1111"), --0010-8

("0011","1111","1011","1111"), --0011-5

("0000","1111","1111","1111"), --0100-9

("1111","1111","1111","1111"), --0101-2

("1111","1111","1111","1111"), --0110-3

("1111","1111","1111","1111"), --0111-4

("1111","1111","1111","1111"), --1000

("1001","1111","1101","1111"), --1001

("1111","1111","1111","1111"), --1010

("1001","1111","1011","1111"), --1011

("0100","1111","1111","1100"), --1100-1

("1101","1111","1101","1100"), --1101

("1111","1111","1111","1111"), --1110-7

("1111","1111","1111","1111")); --1111

constant tab\_z : outtab :=('0', '1', '0', '0', '1', '0','0','0','0','1','0','0','1','0','0','0');

signal st,nexst: bit\_vector(Lst downto 0):="0000";

begin

process

begin

wait on x, st;

if st="1111" then null;

else

nexst<=tab\_st(vecint(st),vecint(x));

y<=tab\_z(vecint(st));

end if;

end process;

st<= transport nexst after TZ;

end kod\_grey;

Побудуємо таблицю функцій переходів.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 00 | 01 | 10 | 11 | Y |
| 0000 | \*0\*0\*0\*0 | \*0\*0\*0\*0 | \*0\*0\*001 |  | 0 |
| 0001 | \*0\*0010\* |  | \*0\*0\*00\* |  | 1 |
| 0011 | \*0\*00\*0\* |  | 01\*00\*0\* |  | 0 |
| 1011 | 0\*\*0100\* |  | 0\*\*00\*0\* |  | 0 |
| 1001 | 0\*\*0\*00\* |  | 0\*01\*00\* |  | 1 |
| 1101 | 0\*0\*\*00\* |  | 0\*0\*\*00\* | 0\*0\*\*010 | 0 |
| 1100 | 100\*\*0\*0 |  |  | 0\*0\*\*0\*0 | 1 |
| 0100 | \*010\*0\*0 |  |  |  | 1 |

Мімізуємо функції переходів картами Карно:

R3=not q(0) and not x(1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 |  |  |  |  |  |  |  |  |
| 001 |  |  |  |  | 0 |  |  |  |
| 011 |  |  |  |  |  |  |  |  |
| 010 |  |  |  |  |  |  |  |  |
| 110 | 1 |  | 0 |  | 0 | 0 |  | 0 |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  | 0 |  |  | 0 |
| 100 |  |  |  |  | 0 |  |  | 0 |

S3=q(1) and x(1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 | 0 | 0 |  | 0 | 0 |  |  | 0 |
| 001 |  |  |  |  | 1 |  |  | 0 |
| 011 |  |  |  |  |  |  |  |  |
| 010 | 0 |  |  |  |  |  |  |  |
| 110 | 0 |  |  |  |  |  |  |  |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  |  |  |  |  |
| 100 |  |  |  |  |  |  |  |  |

R2=not q(3)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 |  |  |  |  |  |  |  |  |
| 001 |  |  |  |  |  |  |  |  |
| 011 |  |  |  |  |  |  |  |  |
| 010 | 1 |  |  |  |  |  |  |  |
| 110 | 0 |  | 0 |  | 0 | 0 |  | 0 |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  |  |  |  |  |
| 100 |  |  |  |  | 0 |  |  |  |

R1=q(3) and not x(1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 |  |  |  |  |  |  |  | 0 |
| 001 |  |  |  |  | 0 |  |  | 0 |
| 011 |  |  |  |  |  |  |  |  |
| 010 |  |  |  |  |  |  |  |  |
| 110 |  |  |  |  |  |  |  |  |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  | 0 |  |  | 1 |
| 100 |  |  |  |  |  |  |  |  |

S2=x(1) and q(3) and not q(1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 | 0 | 0 |  | 0 | 0 |  |  | 0 |
| 001 |  |  |  |  | 0 |  |  | 0 |
| 011 |  |  |  |  |  |  |  |  |
| 010 |  |  |  |  |  |  |  |  |
| 110 |  |  |  |  |  |  |  |  |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  | 0 |  |  | 0 |
| 100 |  |  |  |  | 1 |  |  | 0 |

S1=not q(3) and q(0) and not x(1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 | 0 | 0 |  | 0 | 0 |  |  | 1 |
| 001 |  |  |  |  |  |  |  |  |
| 011 |  |  |  |  |  |  |  |  |
| 010 | 0 |  |  |  |  |  |  |  |
| 110 | 0 |  | 0 |  | 0 | 0 |  | 0 |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  |  |  |  | 0 |
| 100 |  |  |  |  | 0 |  |  | 0 |

R0=x0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 |  |  |  | 0 | 0 |  |  | 0 |
| 001 |  |  |  |  | 0 |  |  | 0 |
| 011 |  |  |  |  |  |  |  |  |
| 010 |  |  |  |  |  |  |  |  |
| 110 |  |  |  |  | 0 | 1 |  | 0 |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  | 0 |  |  | 0 |
| 100 |  |  |  |  | 0 |  |  | 0 |

S0=not x(0) and x(1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 000 | 0 | 0 |  | 1 |  |  |  |  |
| 001 |  |  |  |  |  |  |  |  |
| 011 |  |  |  |  |  |  |  |  |
| 010 | 0 |  |  |  |  |  |  |  |
| 110 | 0 |  | 0 |  |  | 0 |  |  |
| 111 |  |  |  |  |  |  |  |  |
| 101 |  |  |  |  |  |  |  |  |
| 100 |  |  |  |  |  |  |  |  |

Y=q(2) and not q(0) or not q(2) and not q(1) and q(0)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 |  |
| 01 | 1 |  |  |  |
| 11 | 1 | 0 |  |  |
| 10 |  | 1 | 0 |  |

Тепер перевіримо програмно отримані результати:

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity karno is

generic (Lin: integer :=1; Lst : integer := 3);

port(x: in STD\_LOGIC\_vector (Lin downto 0); z: out STD\_LOGIC);

end karno;

architecture karno of karno is

signal r3,s3,r2,s2,r1,s1,r0,s0 : STD\_LOGIC := '0';

signal q:STD\_LOGIC\_vector(Lst downto 0):="0000";

begin

process --F1

begin

wait for 2ns;

r3 <= (not q(0)) and (not x(1));

s3 <= q(1) and x(1);

r2 <= not q(3);

s2 <= x(1) and q(3) and not q(1);

r1 <= q(3) and (not x(1));

s1 <= (not q(3)) and q(0) and (not x(1));

r0 <= x(0);

s0 <= x(1) and (not q(3));

end process;

z<= (q(2) and (not q(0))) or ((not q(2)) and (not q(1)) and q(0)); -- F2

--memory register (RS-latches)

process

begin

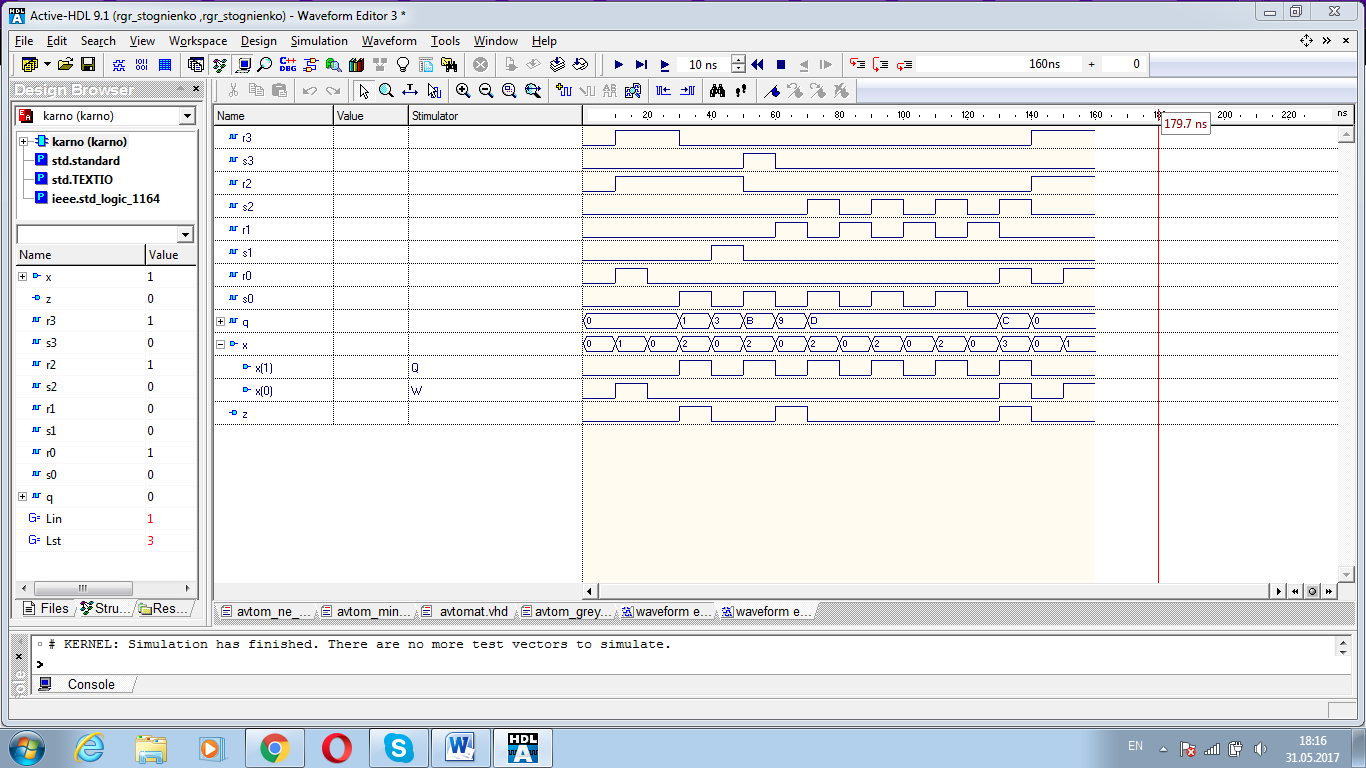
wait on r0,s0,r1,s1,r2,s2,r3,s3;

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 karno;



Як бачимо кінцевий результат співпадає з результатами таблиці переходів, отже мінімізація виконана вірно.

Висновок: В ході розрахунково графічної роботи був побудований цифровий автомат. Спочатку було побудовано таблицю переходів, мінімізовано її, і за допомогою функцій переходів, які було мінімізовано картами Карно, так як автомат був асинхронний то функція переходів була вибрана як функція асинхроного RSтригера. Кінцевий результат співпадає з початковим, отже робота виконана вірно.