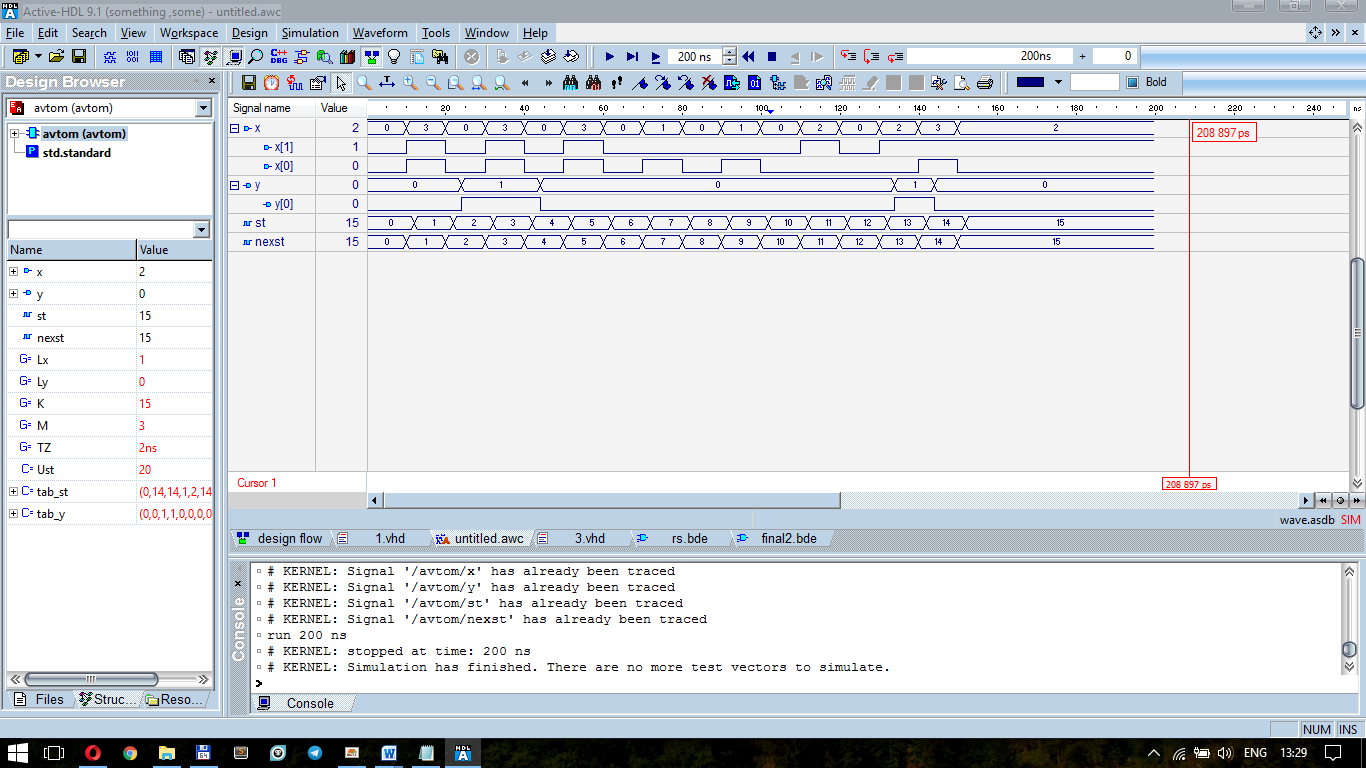
1



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

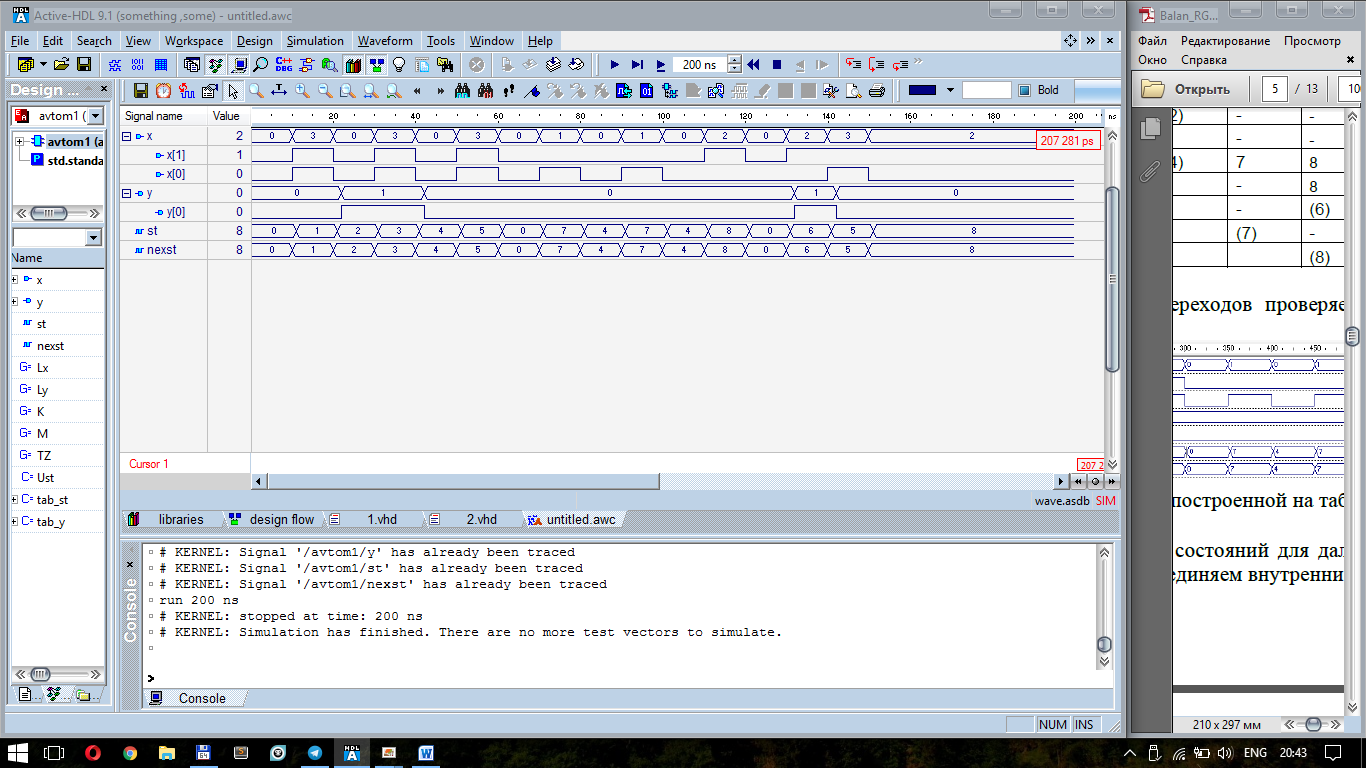
wait on nexst;

st<= transport nexst after TZ;

end process;

end avtom;

2



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

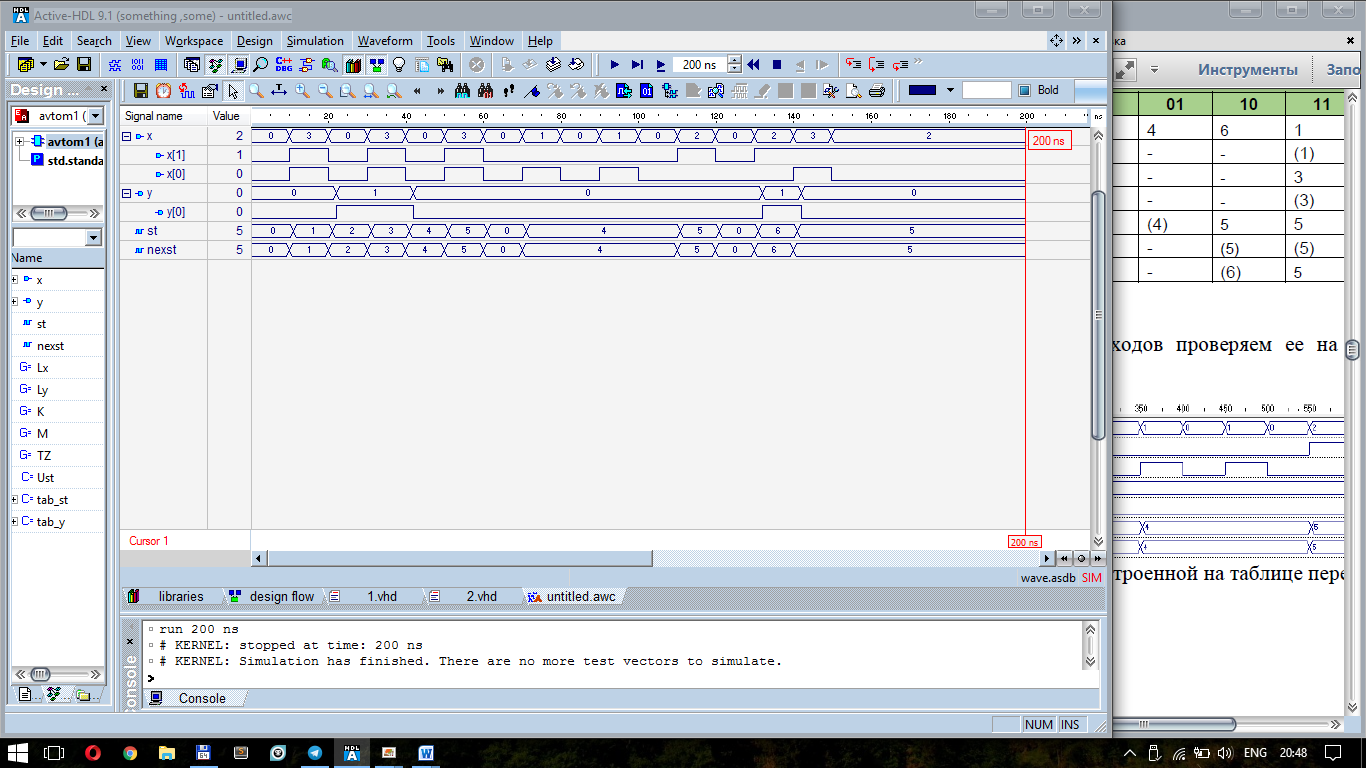
wait on nexst;

st<= transport nexst after TZ;

end process;

end avtom1;

3



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

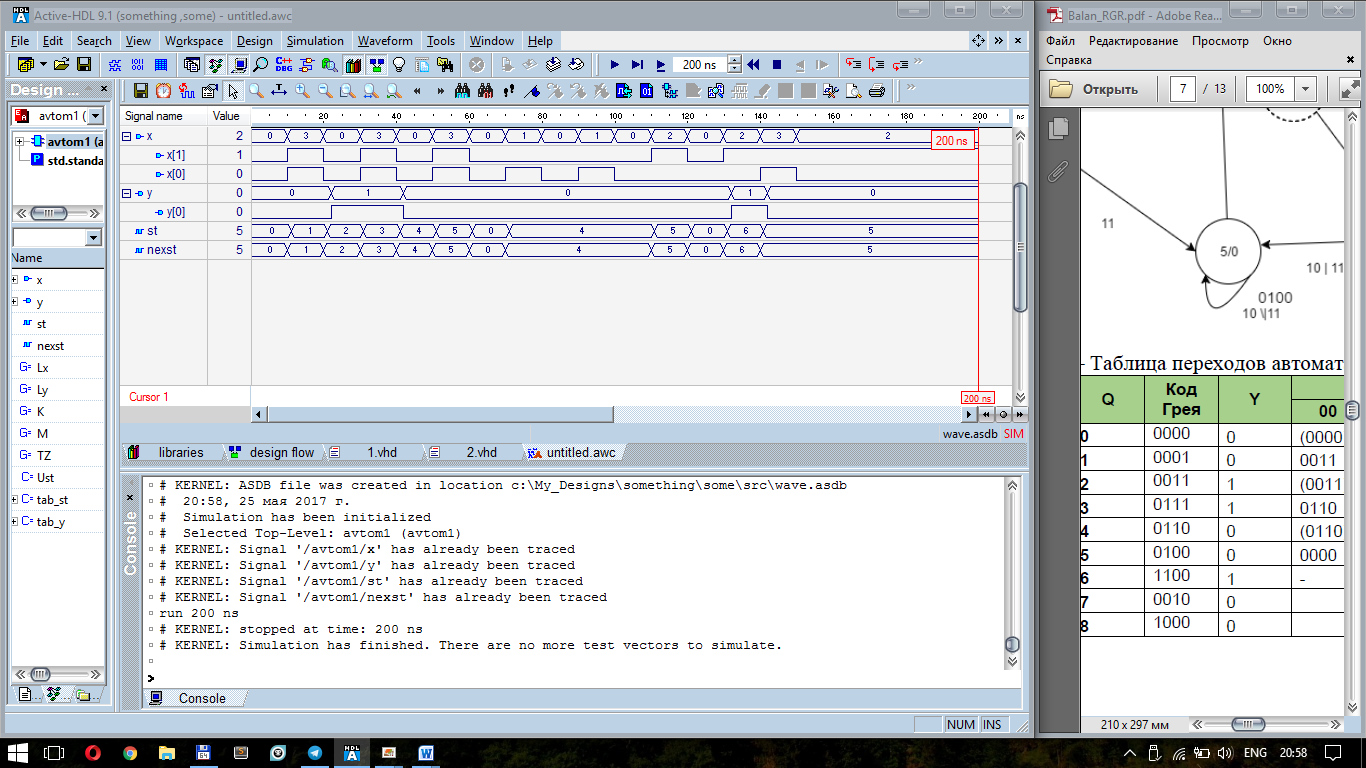
wait on nexst;

st<= transport nexst after TZ;

end process;

end avtom1;

4



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));

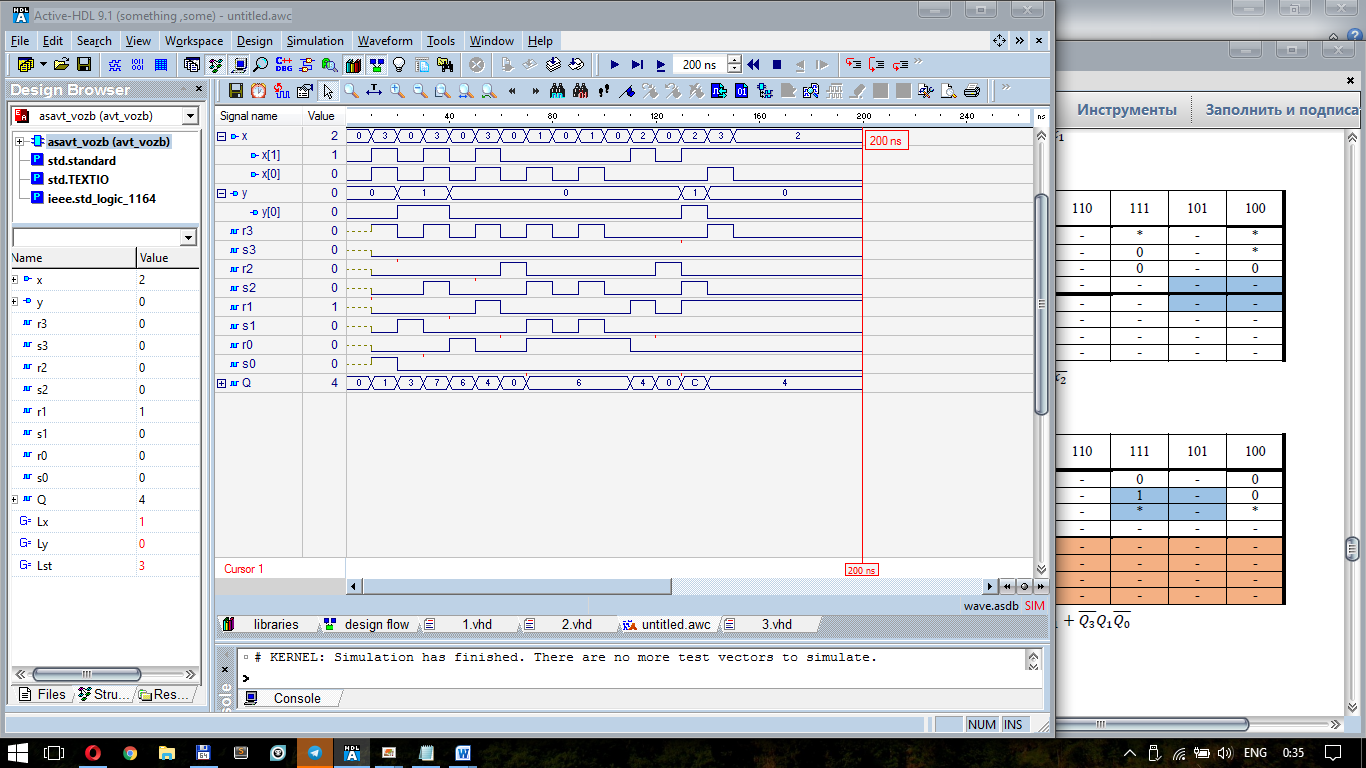
end if;

end process;

st<= transport nexst after TZ;

end avt\_code;

5



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));

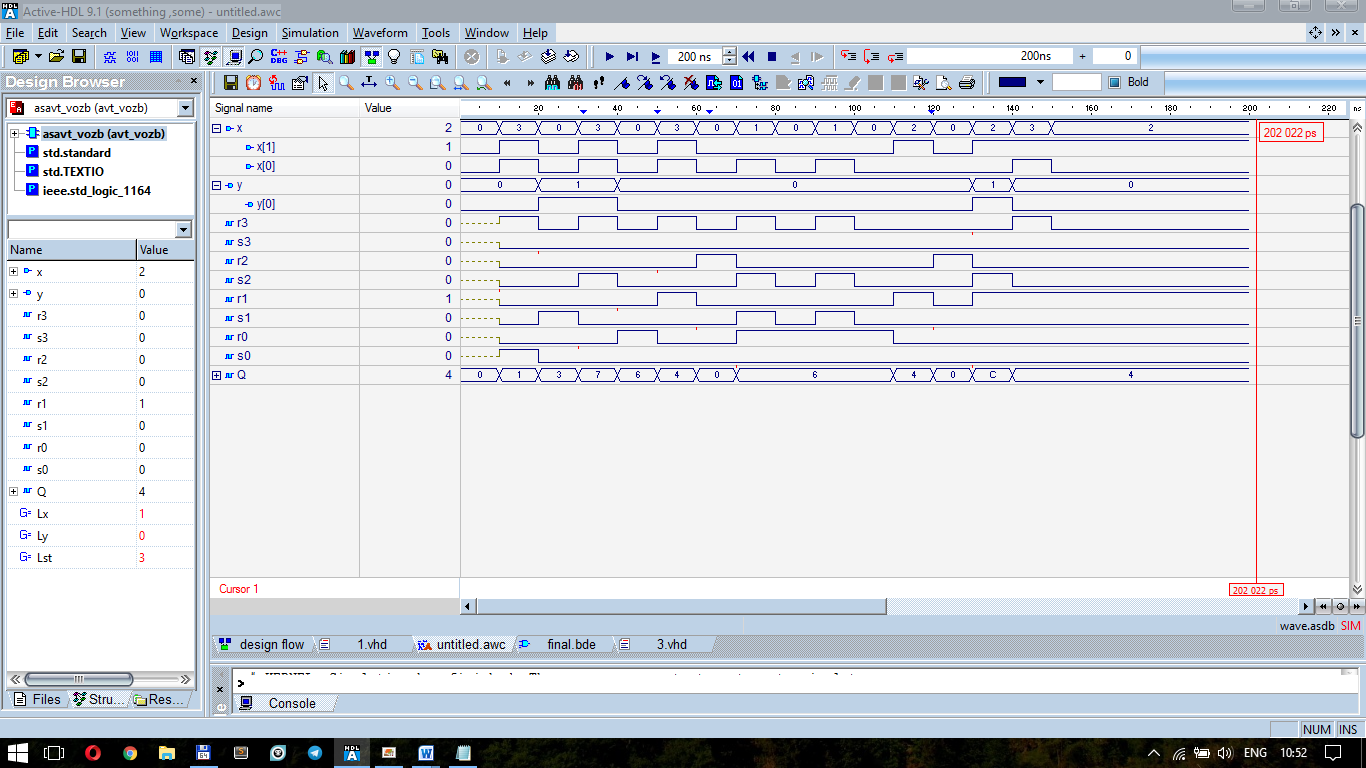
Q(2)<=s2 or (not(r2) and Q(2));

Q(3)<=s3 or (not(r3) and Q(3));

end process;

end avt\_vozb;

6 перевірка базису



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 c установкой в начальное состояние

-- Явное описание запрещенного перехода отсутствует

-- Задавать временные диаграммы нужно правильно.

begin

process

begin -- комбинацонная 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))); -- комбинацонная 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;

7

