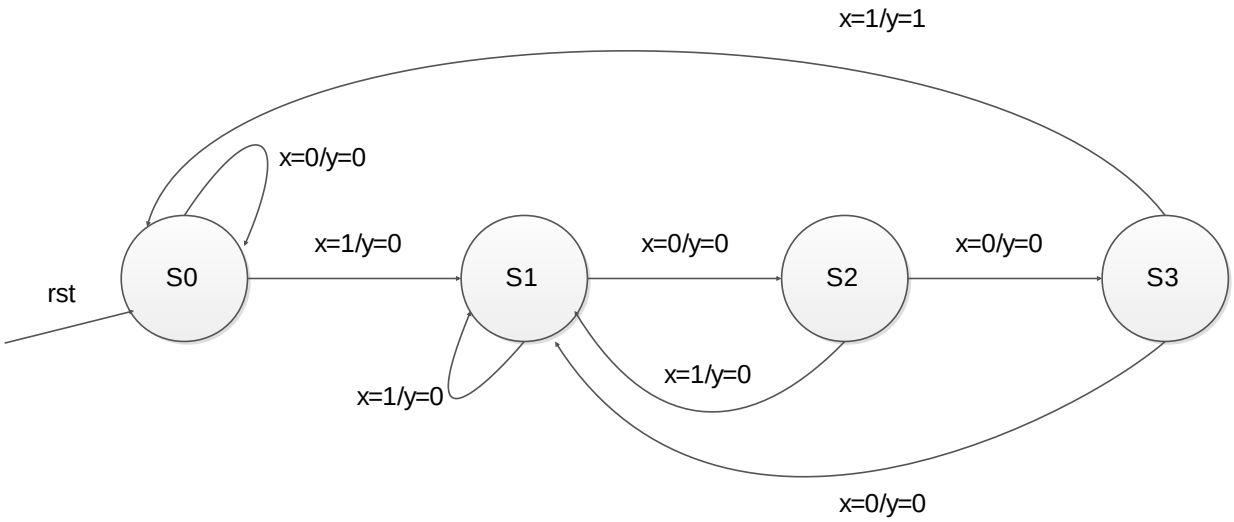
1. Să se efectuieze sinteza unui detector de secvență. Detectorul generează la ieșire valoarea 1 logic doar atunci când la intrare este aplicată secvența 1001.

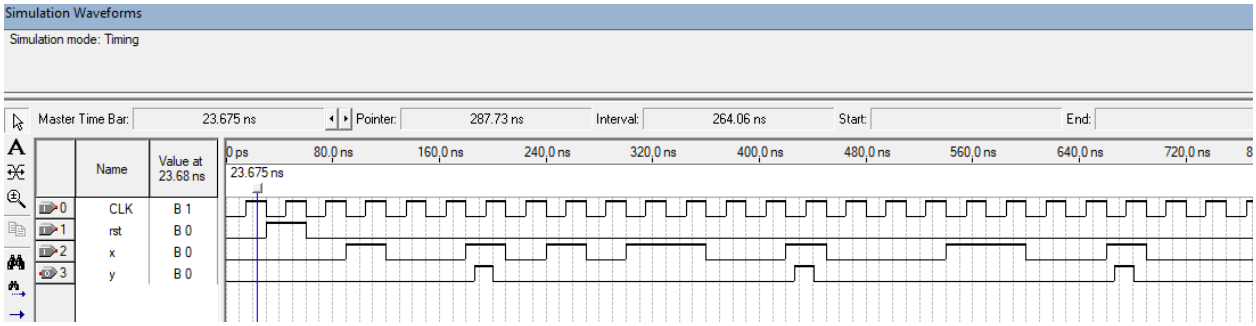
0110111101001

Diagrama de stare pentru automatul Mealy

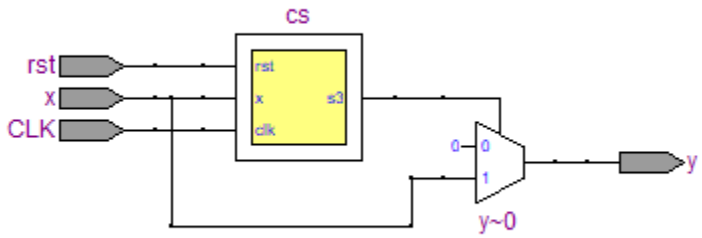



```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity secventa is
Port (CLK, rst,x: in std_logic;
            y: out std_logic);
End entity secventa;
Architecture secventa_arch of secventa is
Type State is (s0, s1, s2, s3);
Signal cs, ns: state;
Begin
Proc1: process (CLK, rst) is
begin
If (rising_edge(clk)) then
If (rst='1') then cs<=s0;  else cs<=ns;
end if;
end if;
end process Proc1;
```

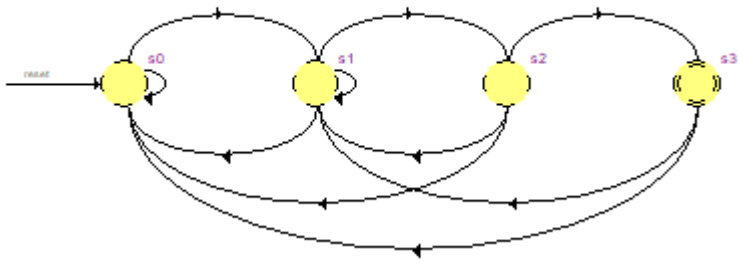
```
Proc2: process (cs, x) is
begin
y<='0';
case cs is
when s0 =>  if (x='1')  then ns<=s1; else ns<=s0; end if;
when s1=>  if (x='0')  then ns<=s2; else ns<=s1; end if;
when s2 =>  if (x='0')  then ns<=s3; else ns<=s1; end if;
when s3 =>  if (x='1')  then ns<=s0; y<='1'; else ns<=s1; end if;
end case;
end process proc2;
end architecture secventa_arch;
```



RTL Viewer

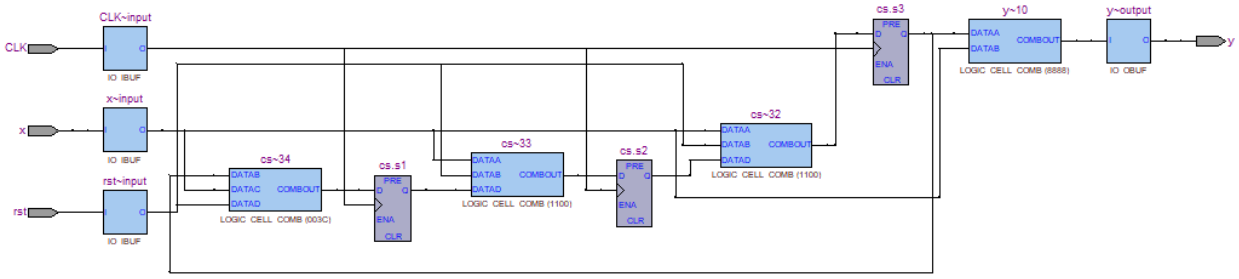


State machine Viewer



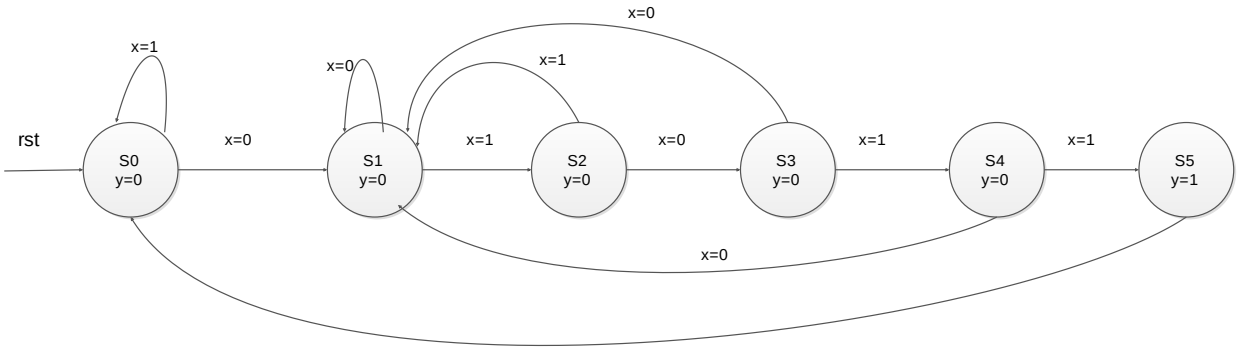
	Source State	Destination State	Condition	
1	s0	s0	(!x) + (x).(rst)	
2	s0	s1	(x).(!rst)	
3	s1	s0	(rst)	
4	s1	s1	(x).(!rst)	
5	s1	s2	(!x).(!rst)	
6	s2	s0	(rst)	
7	s2	s1	(x).(!rst)	
8	s2	s3	(!x).(!rst)	
9	s3	s0	(!x).(rst) + (x)	
10	s3	s1	(!x).(!rst)	

Technology Map Viewer



2. Să se efectuieze sinteza unui detector de secvență. Detectorul generează la ieșire valoarea 1 logic doar atunci când la intrare este aplicată secvența 01011.

Diagrama de stare pentru automatul Moore

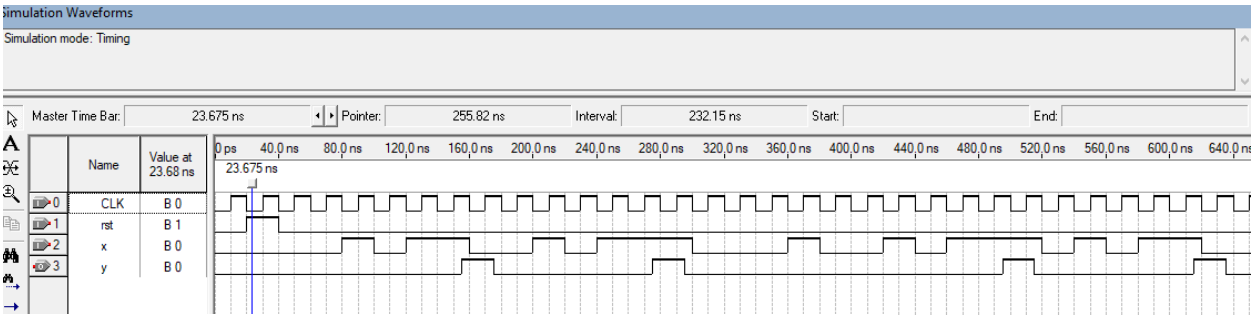


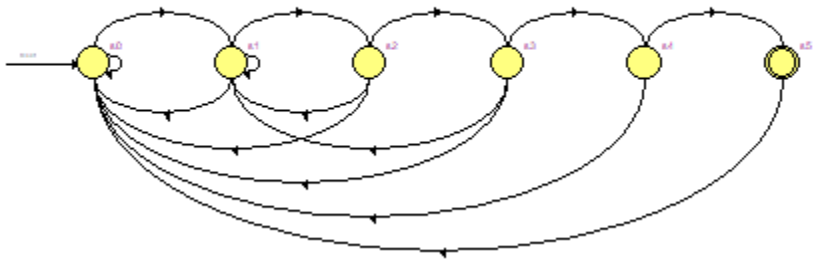
Codul VHDL pentru automatul Moore descris cu 2 procese.

```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity secventa2 is
Port (CLK, rst,x: in std_logic;
            Z: out std_logic);
End entity secventa2;
Architecture secventa2_arch of secventa2 is
Type State is (s0, s1, s2, s3, s4, s5);
Signal cs, ns: state;
Begin
Proc1: process (CLK, rst) is
begin
If (rising_edge(clk)) then
If (rst='1') then cs<=s0;  else cs<=ns;
end if;
end if;
end process Proc1;
```

```
Proc2: process (cs, x) is
```

```
begin
y<='0';
case cs is
when s0 =>  if (x='0')  then ns<=s1; else ns<=s0; end if;
when s1=>  if (x='1')  then ns<=s2; else ns<=s1; end if;
when s2 =>  if (x='0')  then ns<=s3; else ns<=s1; end if;
when s3 =>  if (x='1')  then ns<=s4; else ns<=s1; end if;
when s4 =>  if (x='1')  then ns<=s5; else ns<=s1; end if;
when s5 =>  y <='1'; ns <= s0;
end case;
end process Proc2;
end architecture secventa2_arch;
```





	Source State	Destination State	Condition
1	s0	s0	(!x).(rst) + (x)
2	s0	s1	(!x).(!rst)
3	s1	s0	(rst)
4	s1	s1	(!x).(!rst)
5	s1	s2	(x).(!rst)
6	s2	s0	(rst)
7	s2	s1	(x).(!rst)
8	s2	s3	(!x).(!rst)
9	s3	s0	(rst)
10	s3	s1	(!x).(!rst)
11	s3	s4	(x).(!rst)
12	s4	s0	(!x) + (x).(rst)