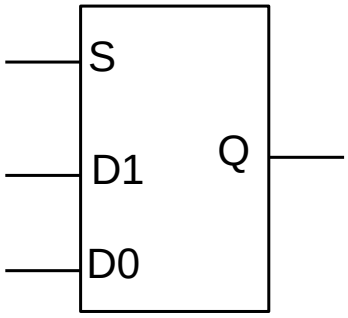
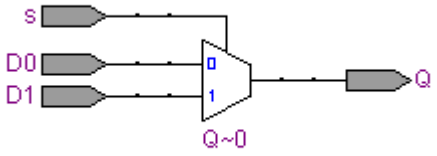
Sinteza si implementarea circuitelor logice combinationale

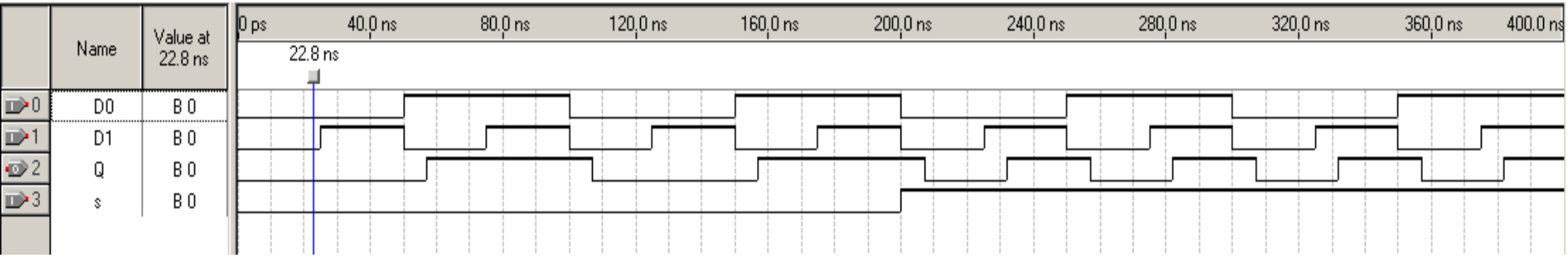
Multiplexoare

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   entity mux2_1 is
4   port (s: in std_logic;
5             D1,D0 : in std_logic;
6             Q : out std_logic);
7   end mux2_1;
8   architecture mux2_1_arch of mux2_1 is
9   begin
10  Q <= D1 when (s = '1') else D0;
11  end mux2_1_arch;
12
```

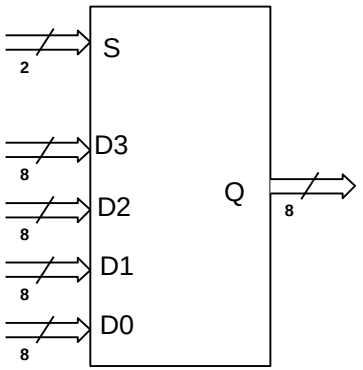
***Mux* $2 \rightarrow 1$**



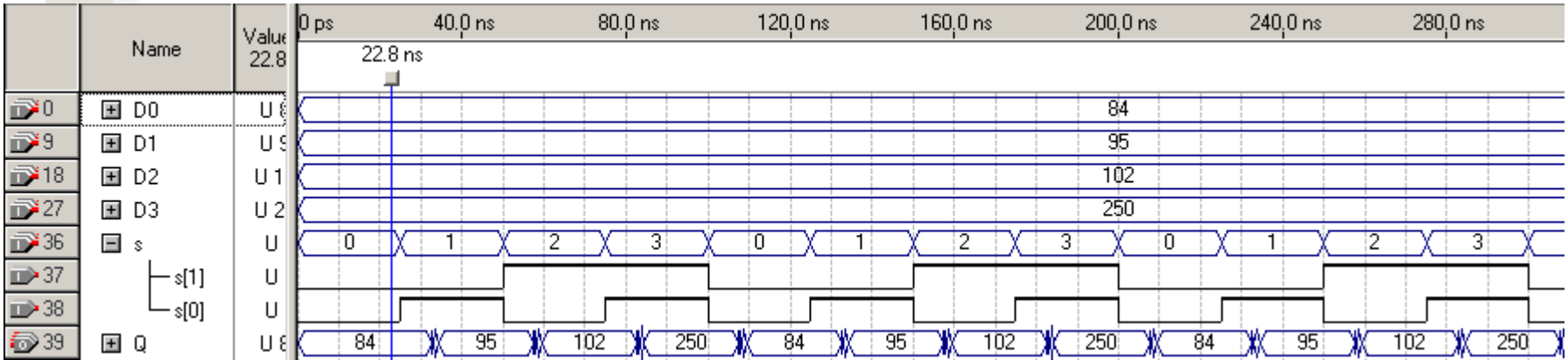


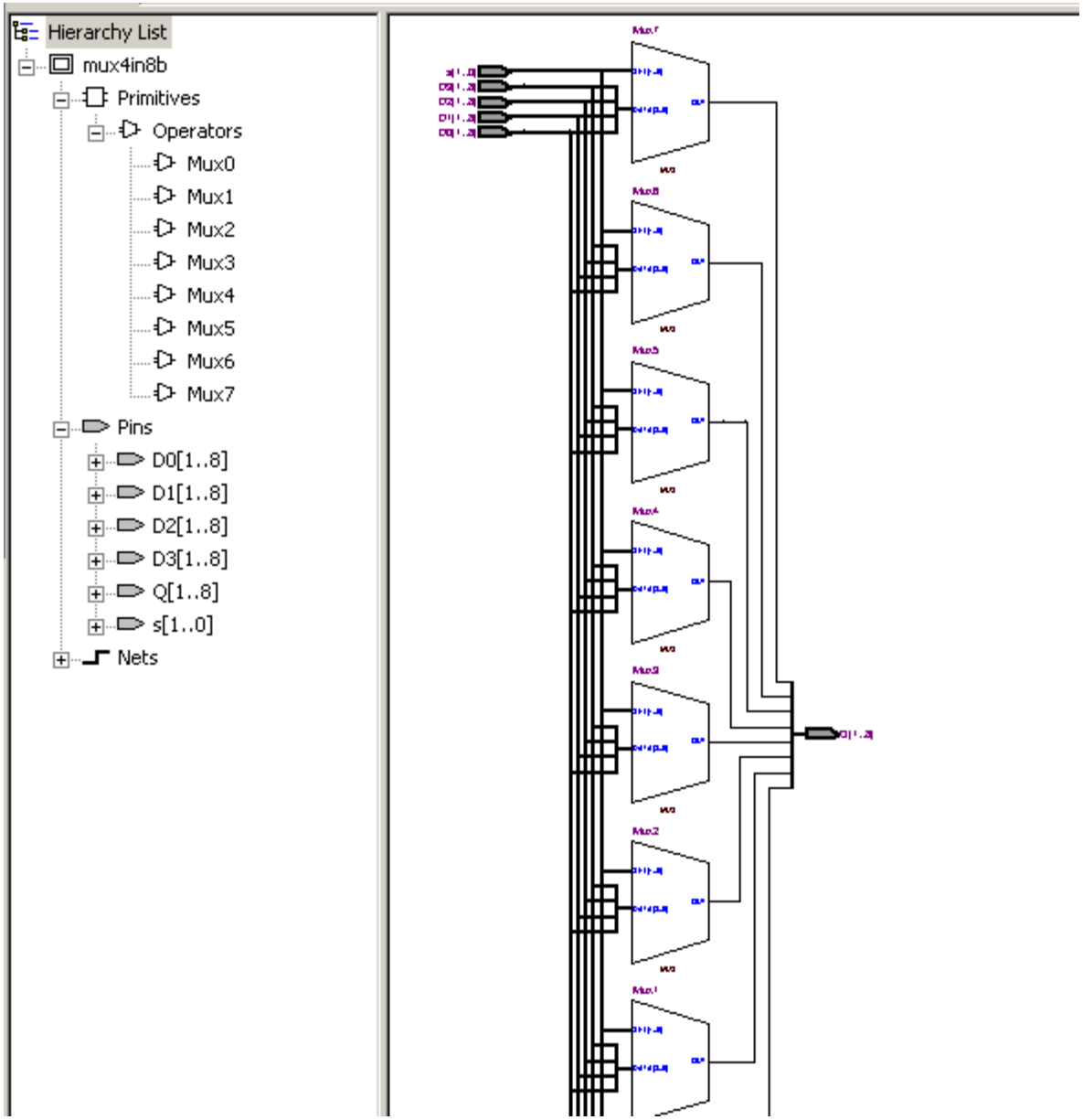


Mux $4 \rightarrow 1$ **(8*biti pe linie*)**

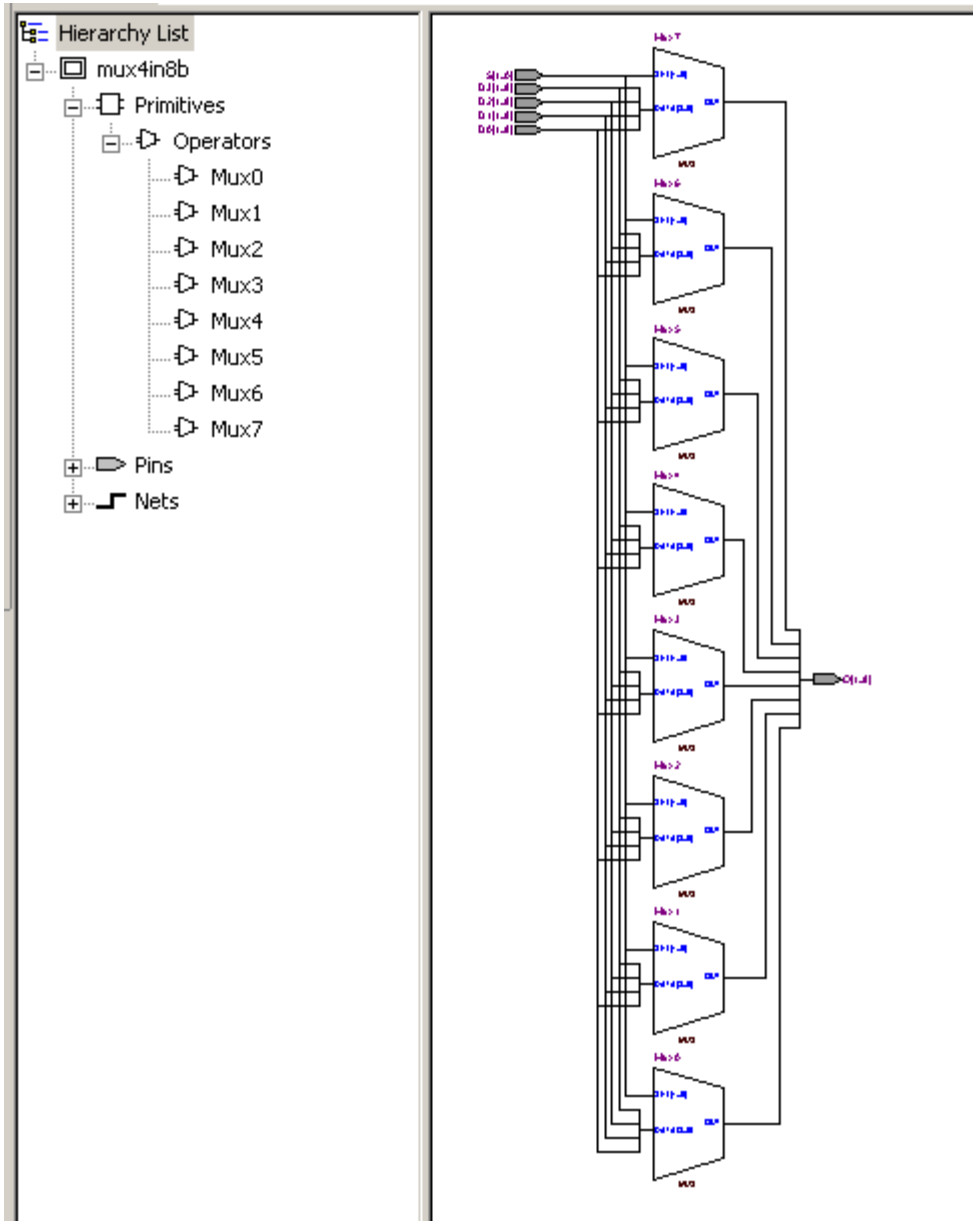



```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  entity mux4in8b is
4  port(
5     s: in std_logic_vector (1 downto 0);
6     D3, D2, D1, D0: in std_logic_vector(1 to 8);
7     Q: out std_logic_vector(1 to 8)  );
8  end mux4in8b;
9  architecture mux4in8b_arch of mux4in8b is
10   begin
11      with S select Q <=
12          D0  when "00",
13          D1  when "01",
14          D2  when "10",
15          D3  when "11",
16          (others  => 'U' ) when others;  -- creaza un vector
17                                          -- de 8 biti  din `U`, valoare
18                                          -- neinitializata
19 end mux4in8b_arch;
```



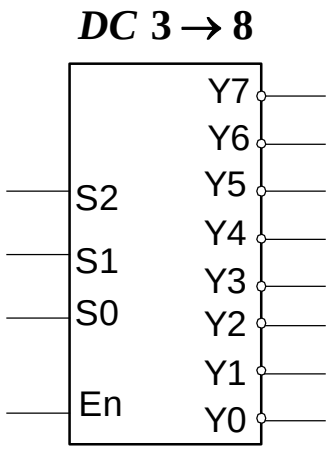

Hierarchy List
mux4in8b
Primitives
Operators
Mux0
Mux1
Mux2
Mux3
Mux4
Mux5
Mux6
Mux7
Pins
D0[1..8]
D1[1..8]
D2[1..8]
D3[1..8]
Q[1..8]
s[1..0]
Nets

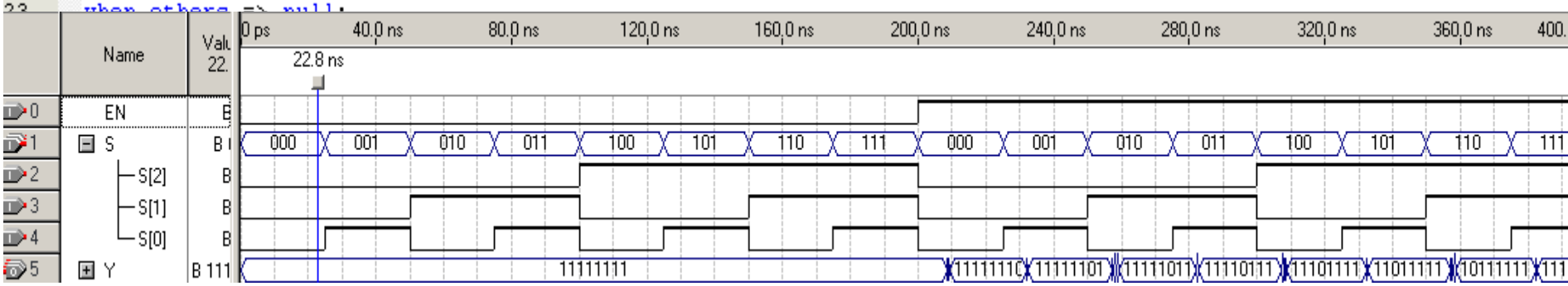
```
 1   library IEEE;
 2   use IEEE.std_logic_1164.all;
 3   entity mux4in8b is
 4   port(     S: in std_logic_vector (1 downto 0);
 5                D3, D2, D1, D0: in std_logic_vector(1 to 8);
 6                Q: out std_logic_vector(1 to 8)  );
 7   end mux4in8b;
 8   architecture mux4in8b_arch of mux4in8b is
 9     begin
10        process (S, D3, D2, D1, D0)
11          begin
12            case  S is
13            when "00"  =>  Q  <=  D0;
14            when "01"  =>  Q  <=  D1;
15            when "10"  =>  Q  <=  D2;
16            when "11"  =>  Q  <=  D3;
17            when others   =>  Q  <=  (others =>'U') ;
18          end case;
19   end process;
20   end mux4in8b_arch;
21
```

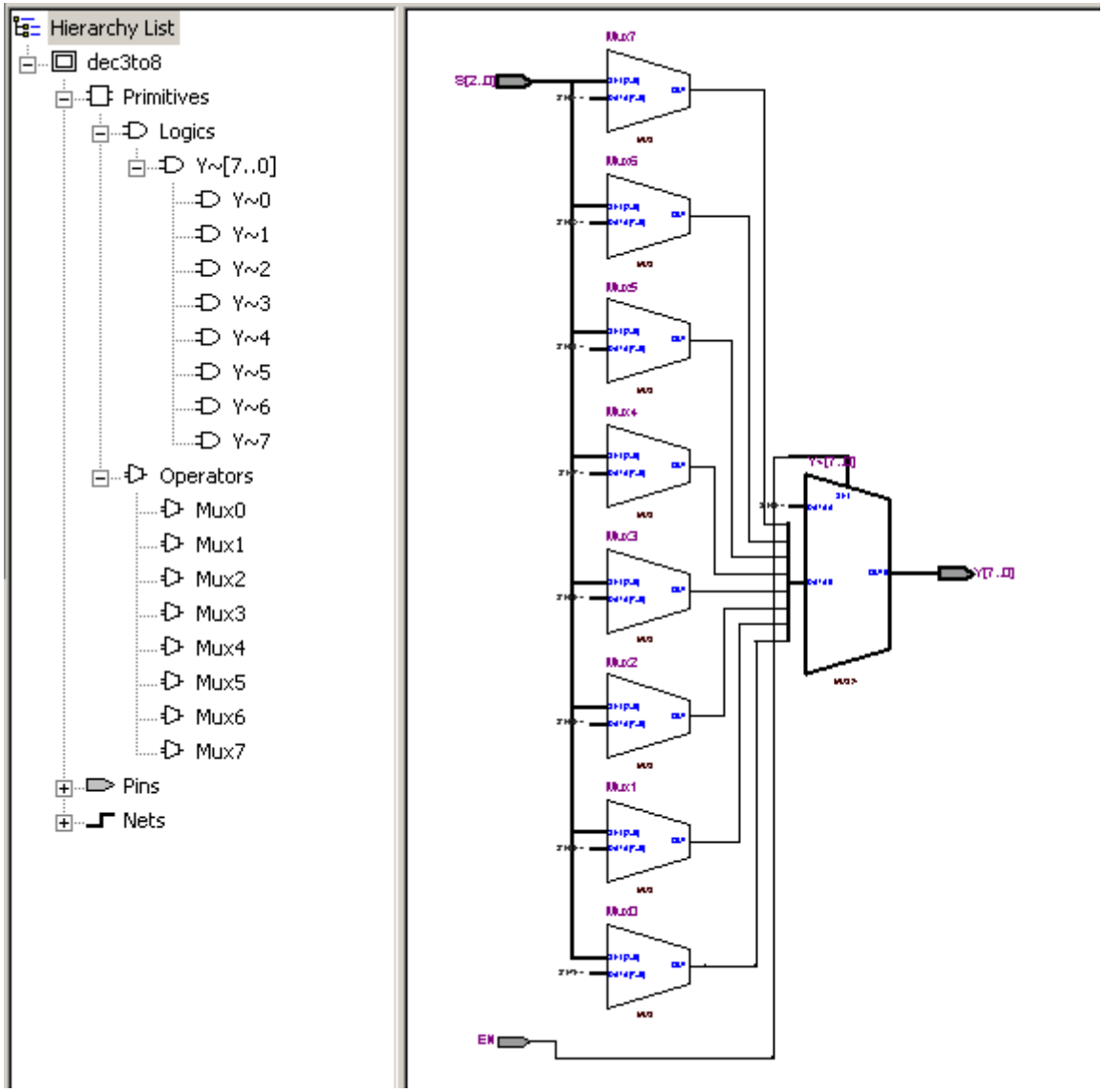

Hierarchy List
mux4in8b
Primitives
Operators
Mux0
Mux1
Mux2
Mux3
Mux4
Mux5
Mux6
Mux7
Pins
Nets

Decodificatoare

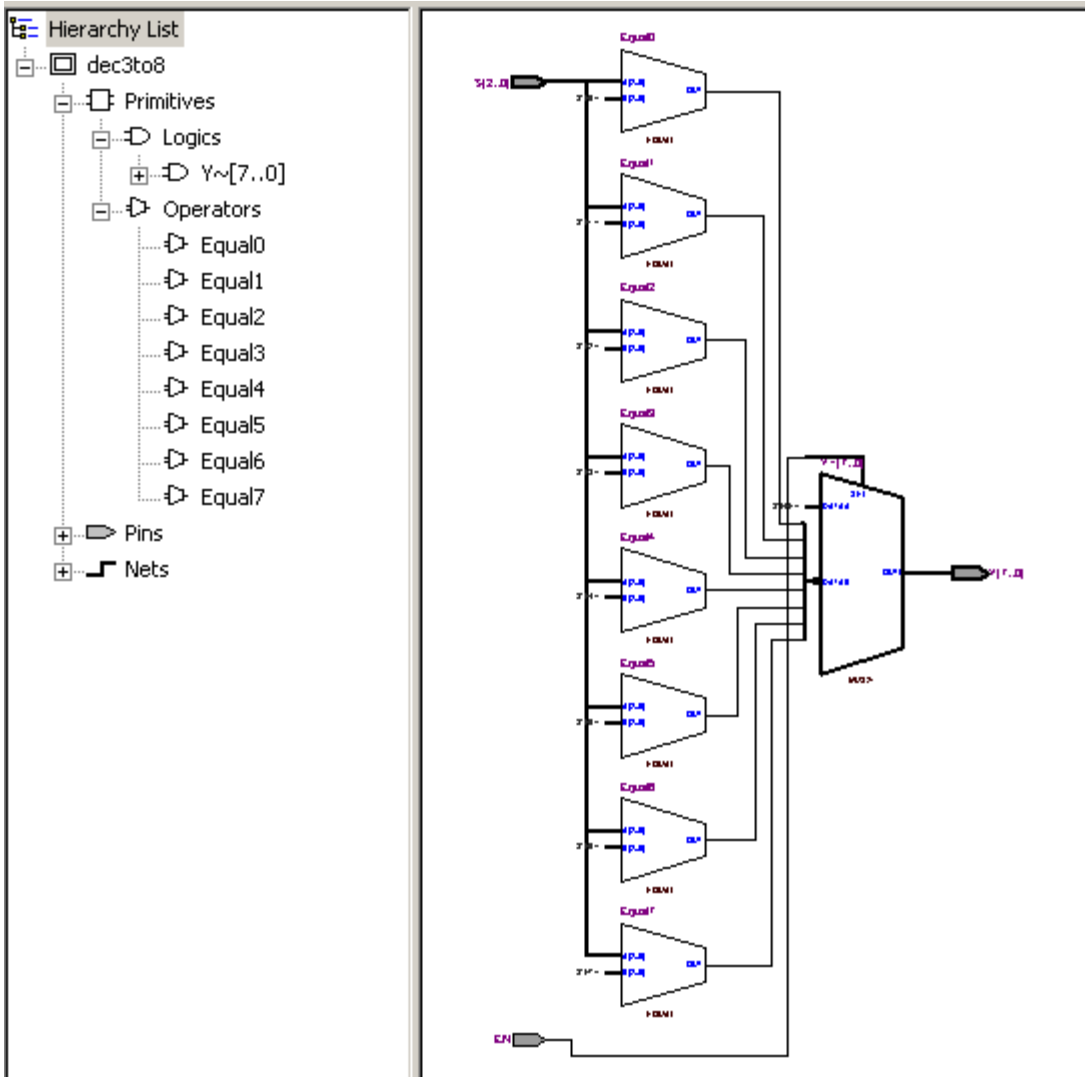
```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   entity dec3to8 is
4   port ( S: in std_logic_vector (2 downto 0); --intrari de selectare
5             EN: in std_logic; -- enable
6             Y: out std_logic_vector (7 downto 0)); -- iesirile sunt active pe zero
7   end dec3to8;
8   architecture dec3to8_arch  of  dec3to8  is
9   begin
10  process (S,EN)
11  begin
12     y <= "11111111";
13     if (en = '1') then
14        case S  is
15  when "000" => Y(0) <= '0';
16  when "001" => Y(1) <= '0';
17  when "010" => Y(2) <= '0';
18  when "011" => Y(3) <= '0';
19  when "100" => Y(4) <= '0';
20  when "101" => Y(5) <= '0';
21  when "110" => Y(6) <= '0';
22  when "111" => Y(7) <= '0';
```



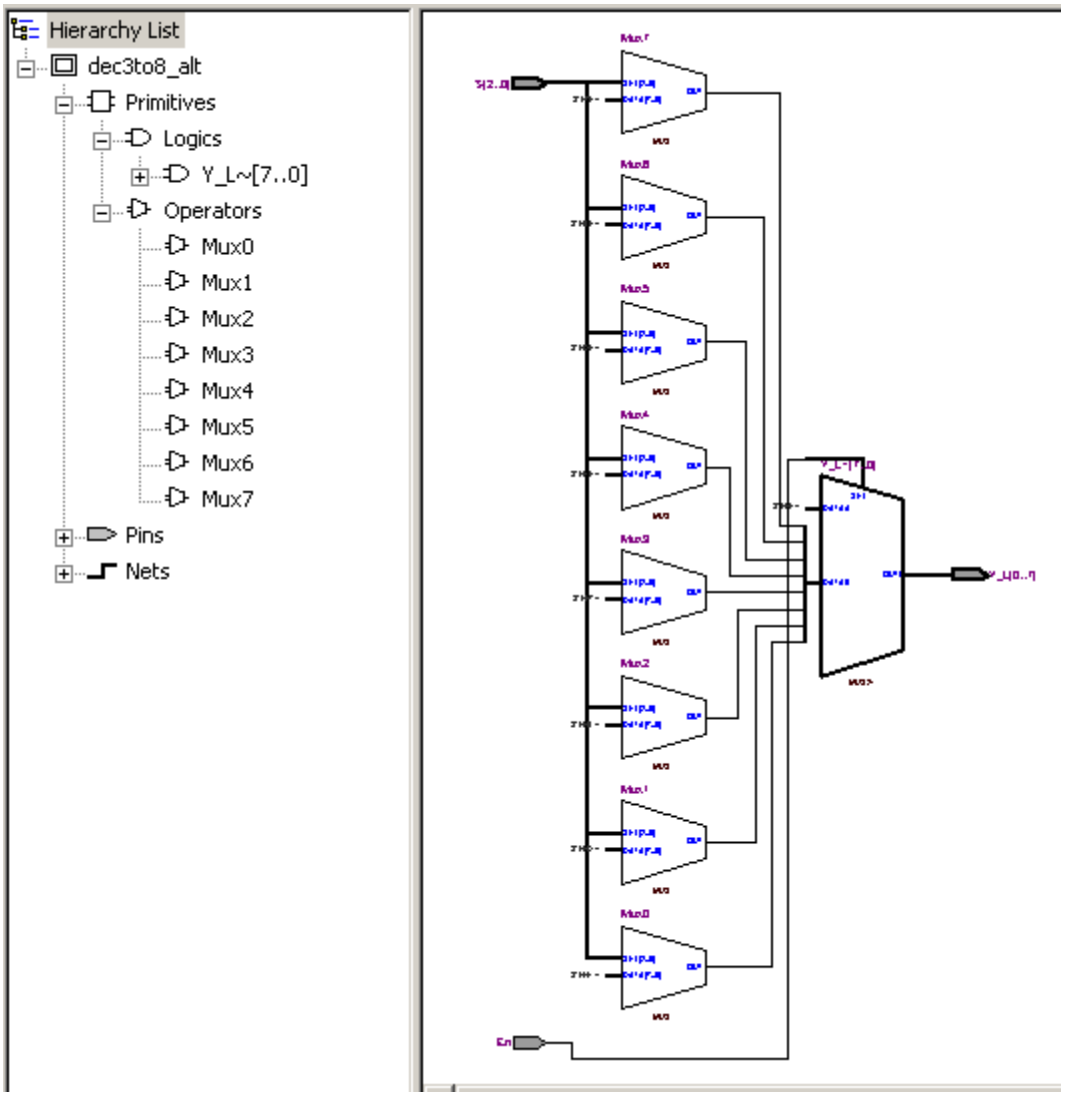



Hierarchy List
dec3to8
Primitives
Logics
Y~[7..0]
Y~0
Y~1
Y~2
Y~3
Y~4
Y~5
Y~6
Y~7
Operators
Mux0
Mux1
Mux2
Mux3
Mux4
Mux5
Mux6
Mux7
Pins
Nets
S[2..0]
EN
Y[7..0]

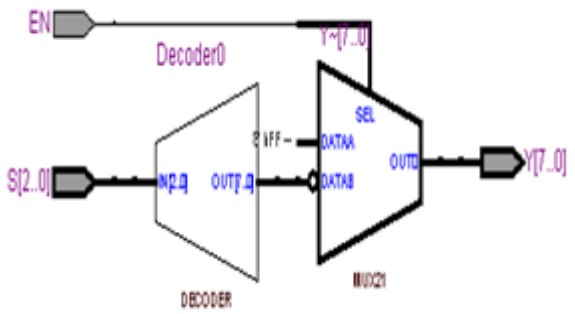
```vhdl
library IEEE;
 use IEEE.std_logic_1164.all;
USE ieee.numeric_std.ALL;
entity dec3to8 is
port ( S: in std_logic_vector (2 downto 0); --intrari de selectare
          EN: in std_logic; -- enable
          Y: out std_logic_vector (7 downto 0)); -- iesirile sunt active pe zero
end dec3to8;
architecture dec3to8_comp  of  dec3to8  is
    begin
       process (S,EN)
          variable i : INTEGER range 0  to 7;
             begin
                 Y <= "11111111";
                 if (EN = '1') then
                    for  i   in   0   to   7  loop
                    if   i = to_integer(unsigned(S))   then   Y(i)  <=  '0' ;
                  end if;
                end loop;
               end if;
       end process;
end dec3to8_comp ;

```

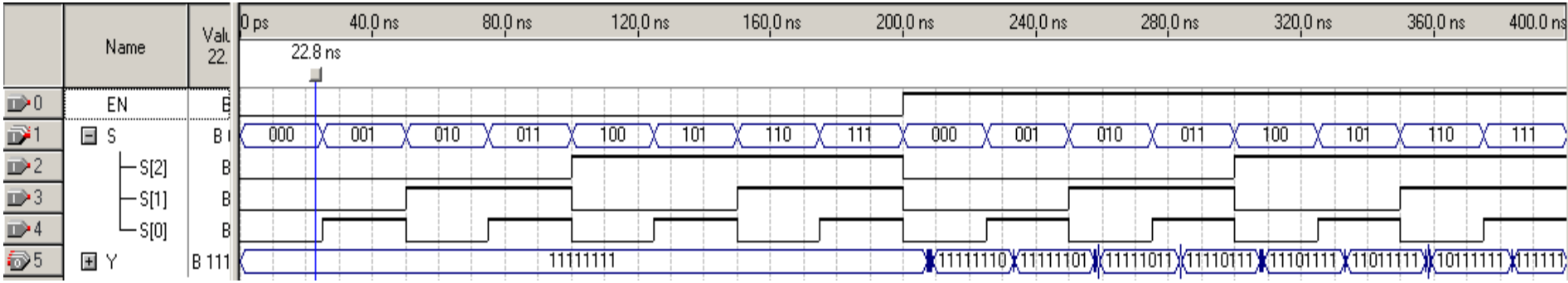

Hierarchy List
dec3to8
Primitives
Logics
Y~[7..0]
Operators
Equal0
Equal1
Equal2
Equal3
Equal4
Equal5
Equal6
Equal7
Pins
Nets

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   entity dec3to8_alt  is
4   port (S: in std_logic_vector(2 downto 0);
5             En: in std_logic;
6             Y_L: out std_logic_vector(0 to 7));
7   end dec3to8_alt;
8   architecture dec3to8_alt  of  dec3to8_alt  is
9   signal YI : STD_LOGIC_VECTOR (0 to 7);
10  begin
11    with S select YI  <=
12  "01111111"   when   "000" ,
13  "10111111"   when   "001" ,
14  "11011111"   when   "010" ,
15  "11101111"   when   "011" ,
16  "11110111"   when   "100" ,
17  "11111011"   when   "101" ,
18  "11111101"   when   "110" ,
19  "11111110"   when   "111" ,
20  "11111111"   when     others;
21      Y_L  <=  YI  when En ='1'    else "11111111";
22  end dec3to8_alt;
```

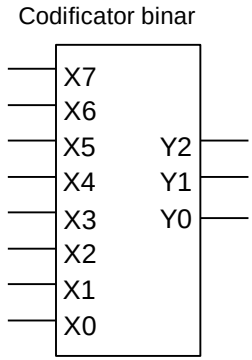

Hierarchy List
dec3to8_alt
Primitives
Logics
Y_L~[7..0]
Operators
Mux0
Mux1
Mux2
Mux3
Mux4
Mux5
Mux6
Mux7
Pins
Nets

```
library IEEE;
 use IEEE.std_logic_1164.all;
USE ieee.numeric_std.ALL;
entity dec3to8 is
port ( S: in std_logic_vector (2 downto 0); --intrari de selectare
         EN: in std_logic; -- enable
         Y: out std_logic_vector (7 downto 0)); -- iesirile sunt active pe zero
end dec3to8;
architecture dec3to8_comp  of  dec3to8  is
    begin
      process (S,EN)
         variable i :INTEGER;
           begin
        Y <= "11111111";
         if (EN = '1') then
i := to_integer(unsigned(S));
                  Y(i) <= '0';
              end if;
      end process;
end dec3to8_comp ;
```





Codificatoare în VHDL

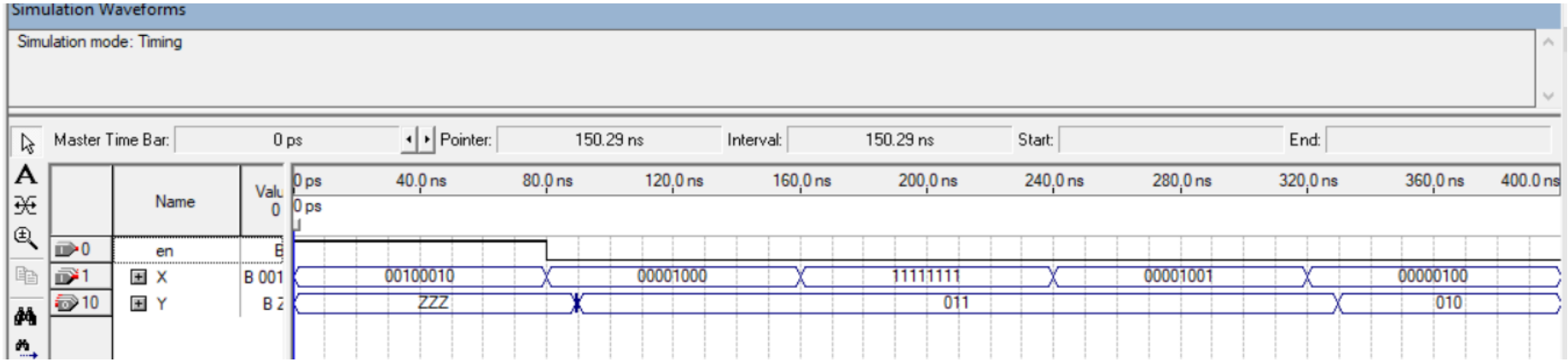


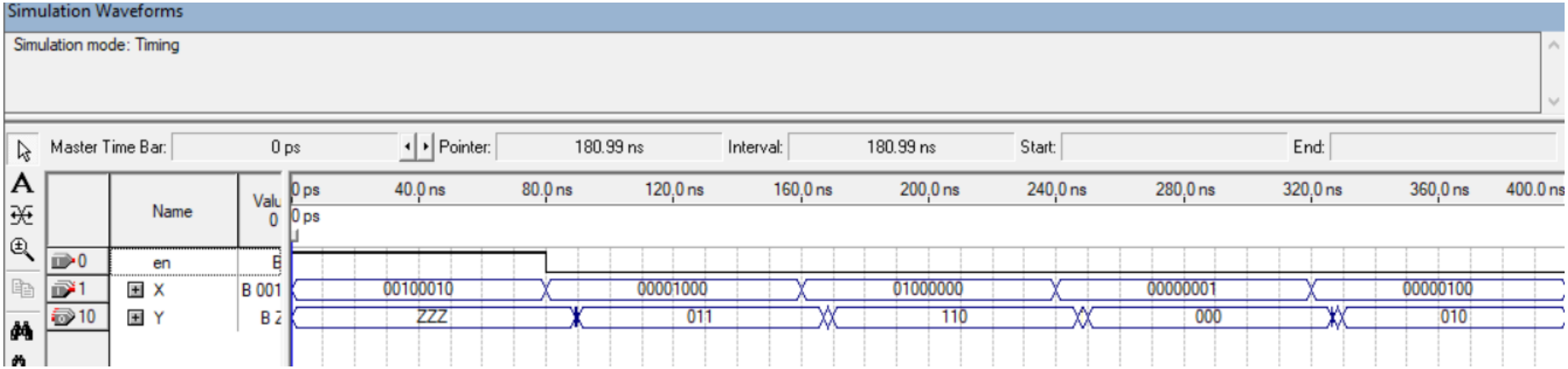
Un codificator binar cu 2^n intrări şi n ieşiri va genera la ieşire un cuvânt de cod la ieşire când se va activa o linie la intrare.

Programul VHDL al codificatorului 8→3 fără prioritate este descris comportamental, utilizând instrucțiunea process:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
 entity CD is
   port (en: in std_logic;
         X: in std_logic_vector (7 downto 0);
         Y: out std_logic_vector (2 downto 0));
  end CD;
```

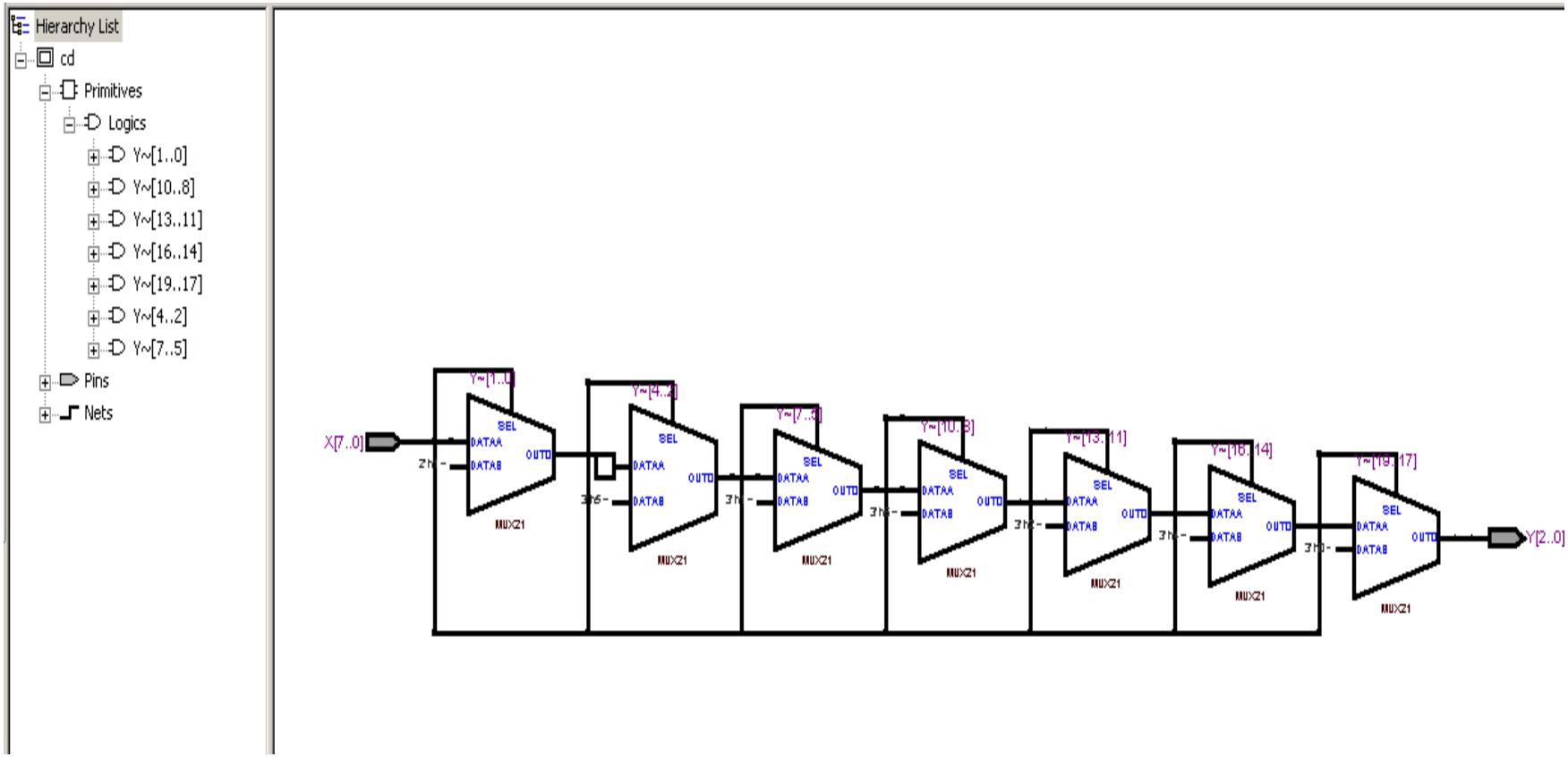
```
architecture CD_arch of CD is
  begin
    process (en, X)
      begin
        if (en='1') then Y<="ZZZ";
        else
        case (X) is
when "00000001" => Y <="000";
when "00000010" => Y <="001";
when "00000100" => Y <="010";
when "00001000" => Y <="011";
when "00010000" => Y <="100";
when "00100000" => Y <="101";
when "01000000" => Y <="110";
when "10000000" => Y <="111";
when others  => null;
        end case;
    end if;
  end process;
end;
```

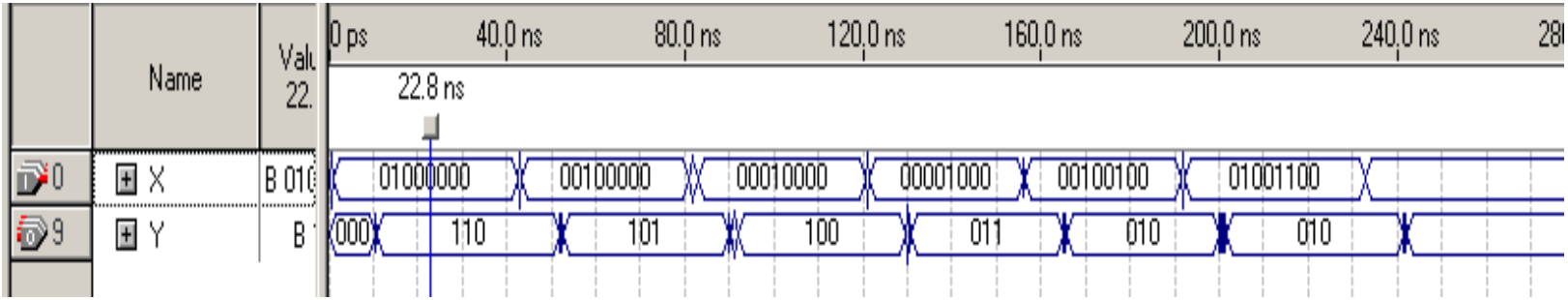

Simulation Waveforms
Simulation mode: Timing
Master Time Bar: 0 ps
Pointer: 150.29 ns
Interval: 150.29 ns
Start:
End:
Name
Valu
0
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
360.0 ns
400.0 ns
0 ps
0
en
1
X
B 001
10
Y
00100010
00001000
11111111
00001001
00000100
ZZZ
011
010


Simulation Waveforms
Simulation mode: Timing
Master Time Bar: 0 ps
Pointer: 180.99 ns
Interval: 180.99 ns
Start:
End:
Name
Valu
0
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
360.0 ns
400.0 ns
0 ps
0
en
1
X
B 001
10
Y
00100010
00001000
01000000
00000001
00000100
ZZZ
011
110
000
010

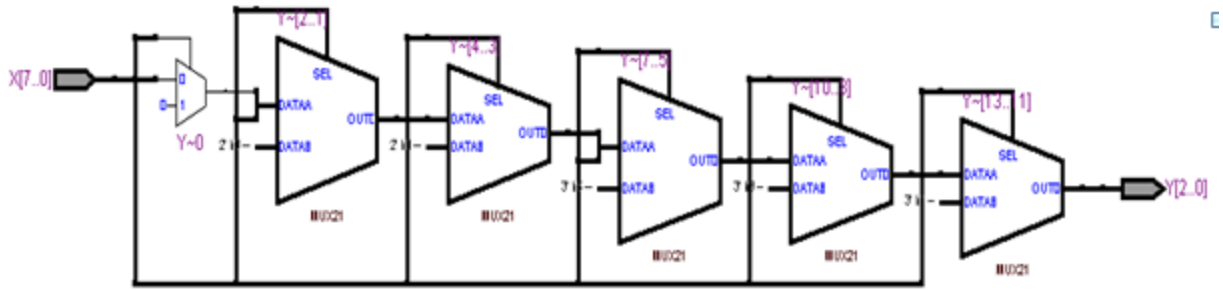
Programul VHDL al codificatorului 8→3 cu prioritate este descris utilizând fluxul de date

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   entity CD is port (
4      X : in  std_logic_vector (7 downto 0);
5      Y : out std_logic_vector (2 downto 0));
6   end CD;
7   architecture CD_arch of CD is
8   begin
9       Y <=
10          "000" when X(0)='1' else
11          "001" when X(1)='1' else
12          "010" when X(2)='1' else
13          "011" when X(3)='1' else
14          "100" when X(4)='1' else
15          "101" when X(5)='1' else
16          "110" when X(6)='1' else
17          "111" when X(7)='1' else
18          "000";
19    end CD_arch;
20
```


Hierarchy List
cd
Primitives
Logics
Y~[1..0]
Y~[10..8]
Y~[13..11]
Y~[16..14]
Y~[19..17]
Y~[4..2]
Y~[7..5]
Pins
Nets
X[7..0]
SEL
DATAA
DATAB
OUTD
MUX21
Y[2..0]


Name
Valu
22.
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
22.8 ns
X
Y
B 010
B
01000000
00100000
00010000
00001000
00100100
01001100
000
110
101
100
011
010
010

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
entity CD_PR is port (
    X : in  std_logic_vector (7 downto 0);
    Y : out std_logic_vector (2 downto 0));
end CD_PR;
architecture CD_PR_arch of CD_PR is
begin
process(X)
begin
    Y <= "000";
    if      X(7)='1' then Y <= "111";
    elsif   X(6)='1' then Y <= "110";
    elsif   X(5)='1' then Y <= "101";
    elsif   X(4)='1' then Y <= "100";
    elsif   X(3)='1' then Y <= "011";
    elsif   X(2)='1' then Y <= "010";
    elsif   X(1)='1' then Y <= "001";
    elsif   X(0)='1' then Y <= "000";
    end if;
end process;
end CD_PR_arch;

```



Sumatoare binare

Sumator cu transport succesiv pe 8 biţi.

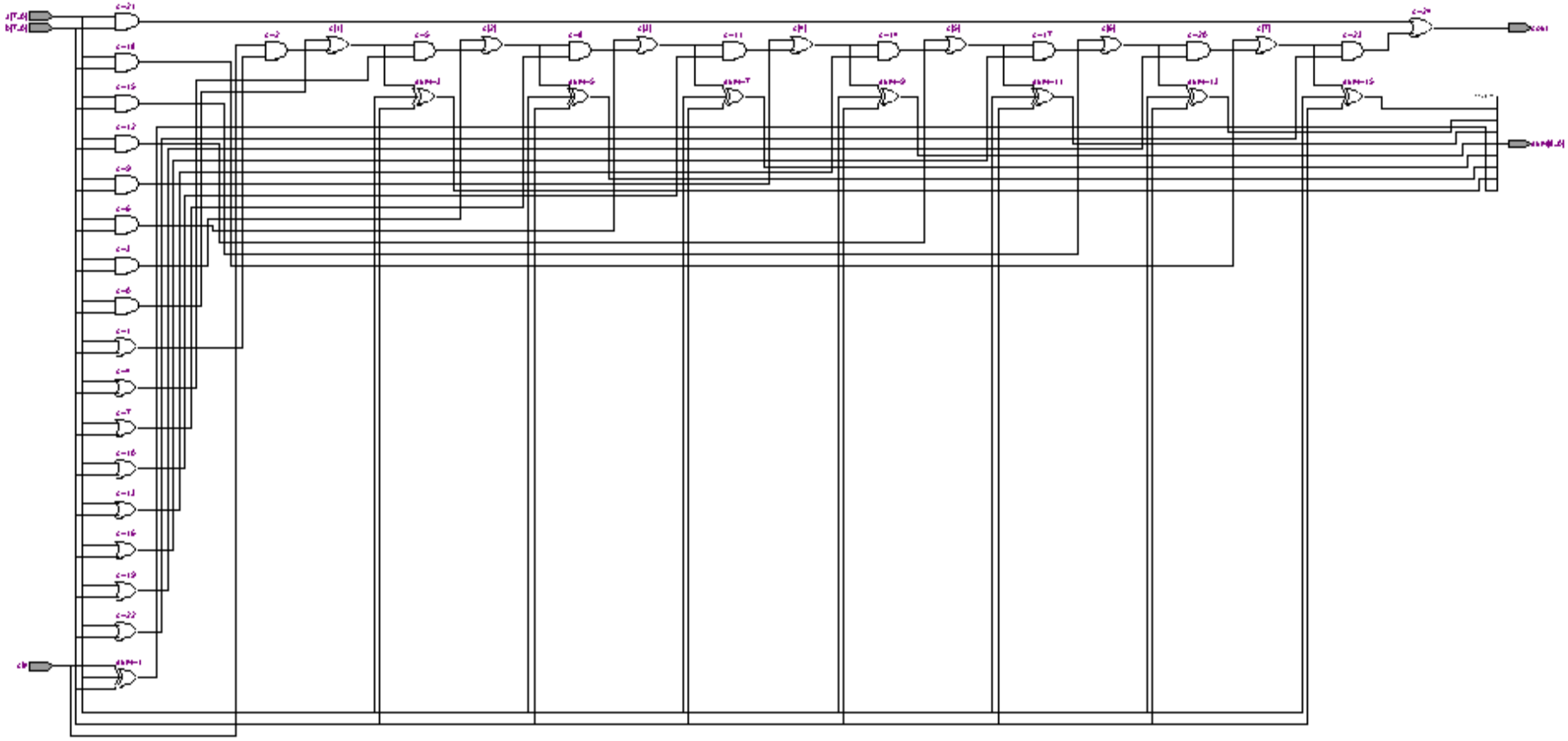
Expresiile logice pentru un sumatior complet de 1 bit:

$$S = a \oplus b \oplus c_{in}$$

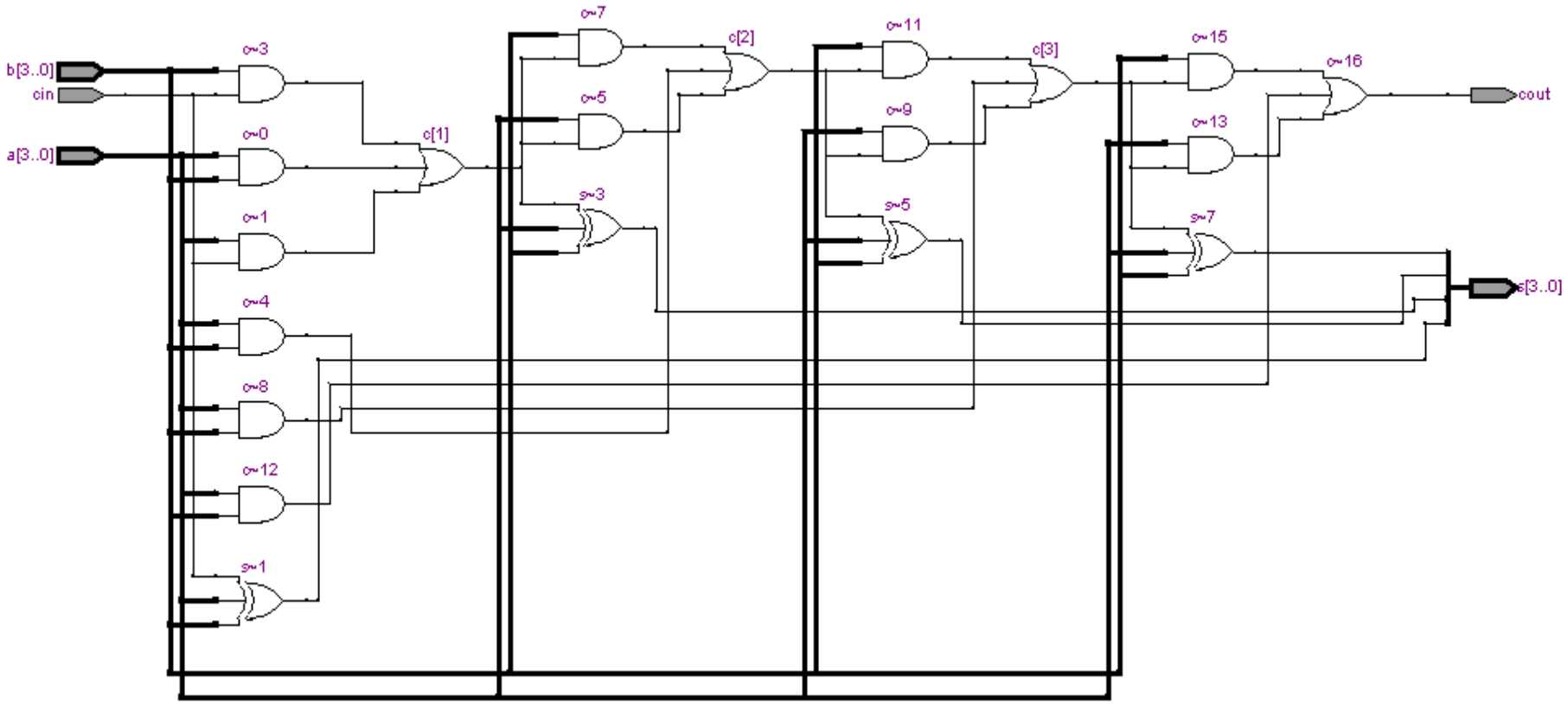
$$C_{out} = a \cdot b + c_{in} \cdot (a + b)$$

```
library IEEE;
use IEEE.std_logic_1164.all;
entity adder8 is port (
    a,b: in std_logic_vector (7 downto 0); -- semnalele de intrare
    cin: in std_logic; -- transport la intrare
    sum: out std_logic_vector(8 downto 0); -- semnale de ieşire
    cout: out std_logic); -- transport la ieşire
end adder8;
```

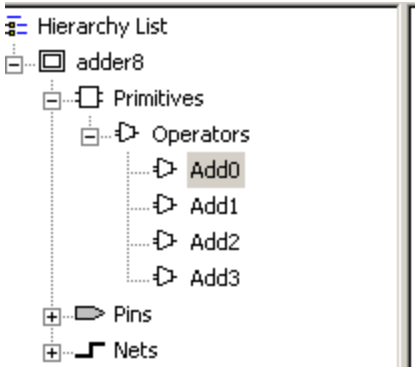
```
architecture adder8_arch of adder8 is
signal c: std_logic_vector(8 downto 0);
  begin
    process (a,b,cin,c)
      begin
        c(0) <= cin;
        for i in 0 to 7 loop
          sum(i) <= a(i) xor b(i) xor c(i);
          c(i+1) <= (a(i) and b(i)) or (c(i) and (a(i) or b(i)));
        end loop;
      cout <= c(8);
    end process;
end adder8_arch;
```



```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3  entity add_succ is
 4  generic (n: natural := 4);
 5  port (a, b: in std_logic_vector (n-1 downto 0);
 6            cin: in std_logic;
 7            s: out std_logic_vector (n-1 downto 0);
 8            cout: out std_logic);
 9  end add_succ;
10  architecture structural of add_succ is
11    signal c: std_logic_vector (n downto 0);
12      begin
13         c(0) <= cin;
14         gen: for i in 0 to n-1 generate
15                  s(i) <= a(i) xor b(i) xor c(i);
16                  c(i+1) <= (a(i) and b(i)) or (a(i) and c(i)) or
17                                       (b(i) and c(i));
18        end generate;
19             cout <= c(n);
20  end structural;
21
22
```


b[3..0]
cin
a[3..0]
c~3
c~0
c~1
c~4
c~8
c~12
s~1
c[1]
c~7
c~5
s~3
c[2]
c~11
c~9
s~5
c[3]
c~15
c~13
s~7
c~16
cout
s[3..0]

```
 1   library IEEE;
 2   use IEEE.std_logic_1164.all;
 3   use IEEE.std_logic_arith.all;
 4
 5   entity adder  is port (
 6       a,b: in unsigned (7 downto 0);
 7        c: in signed (7 downto 0);
 8        d: in std_logic_vector (7 downto 0);
 9         s: out unsigned (8 downto 0);
10         t: out signed (8 downto 0);
11         u: out signed (7 downto 0);
12         v: out std_logic_vector(8 downto 0)   );
13       end adder8;
14  architecture adder_arch of adder8 is
15    begin
16        s <= ('0'& a) + ('0' & b);
17         t <= a+c ;
18         u <= c + signed (d) ;
19         v <= c - unsigned (d) ;
20  end adder_arch;
21
22
```




```
s <= ('0'& a) + ('0' & b);
 t <= a+c ;
 u <= c + signed (d) ;
 v <= c - unsigned (d) ;
```