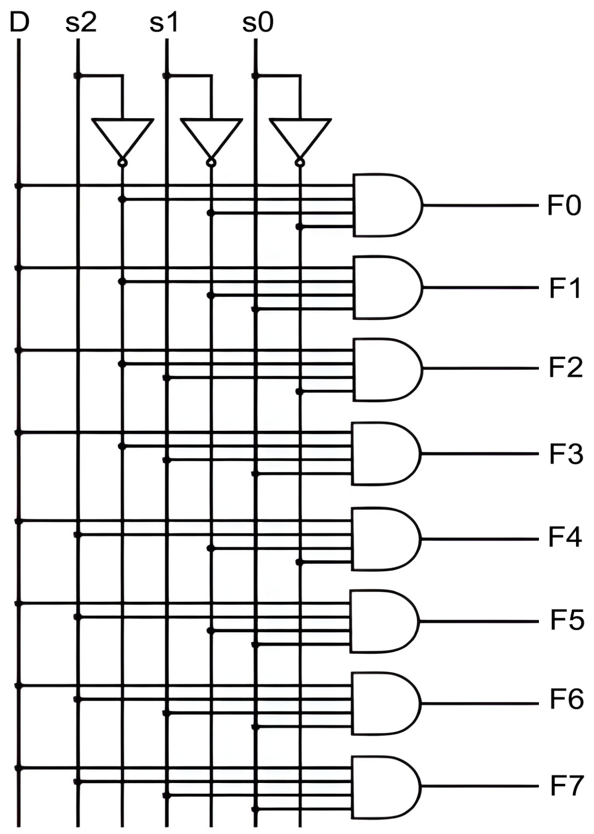
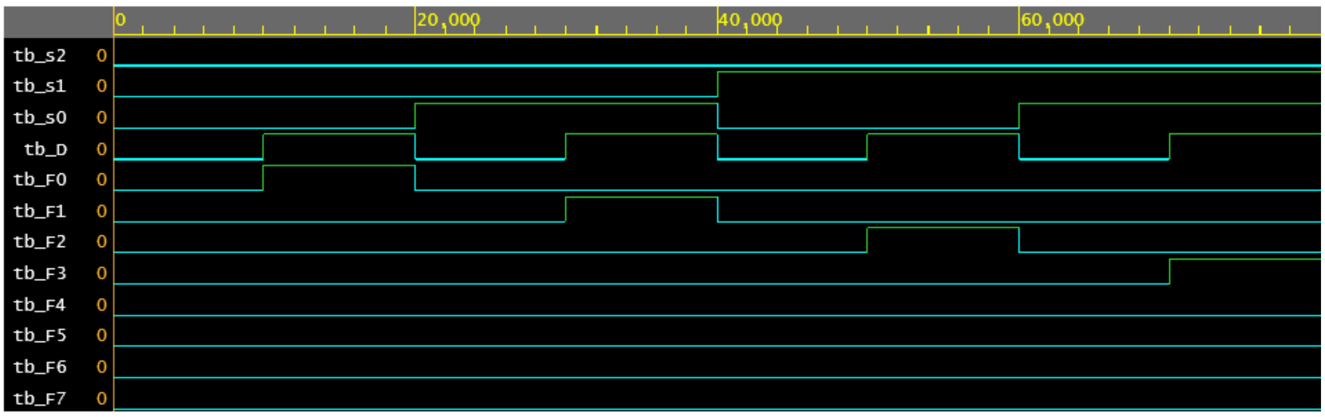
1 to 8 Demultiplexer

s2	s1	s0	D	F0	F1	F2	F3	F4	F5	F6	F7
0	0	0	0|1	**D**	0	0	0	0	0	0	0
0	0	1	0|1	0	**D**	0	0	0	0	0	0
0	1	0	0|1	0	0	**D**	0	0	0	0	0
0	1	1	0|1	0	0	0	**D**	0	0	0	0
1	0	0	0|1	0	0	0	0	**D**	0	0	0
1	0	1	0|1	0	0	0	0	0	**D**	0	0
1	1	0	0|1	0	0	0	0	0	0	**D**	0
1	1	1	0|1	0	0	0	0	0	0	0	**D**

A 1x8 demultiplexer consists of one enable input (D), three input lines (s2, s1, s0) and eight outputs AND (F0 - F7) consisting of four inputs . The values of F0 - F7 always have a value equal to the input of D (depending on the values of inputs s2,s1,s0). By comparing the simulation results with the truth table we see that you achieve the desired result. In all 3 circuits the simulation results are as follows.



Run Time: 160ns
- 0ns to 80ns:



For instance:

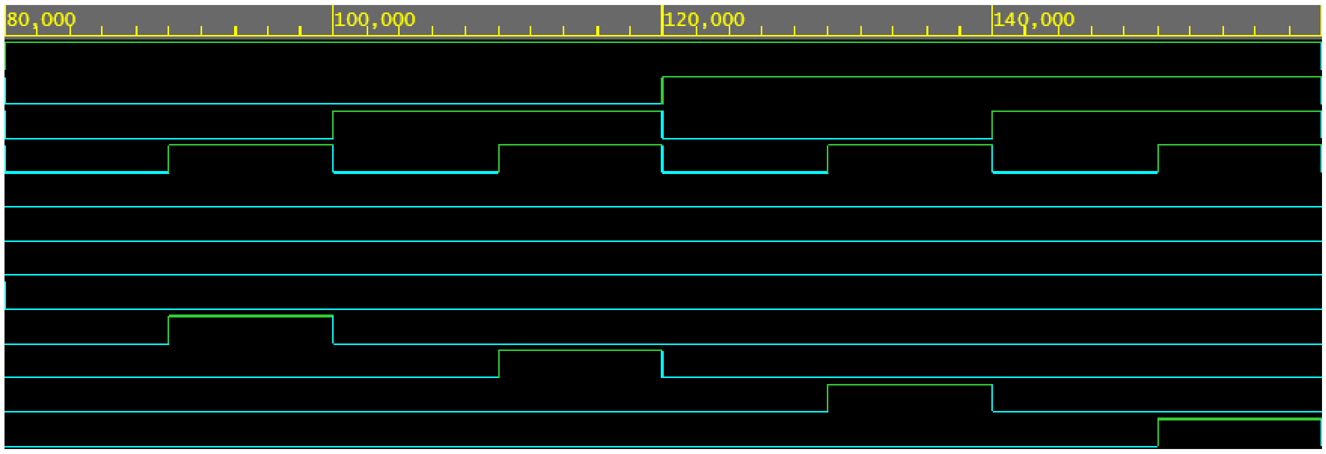
For the value of output F1:

- **20ns:** s2 = 0, s1 = 0, s0 = 1, **D = 0**, (001| 0)
  F0 = 0, **F1 = 0**, F2 = 0, F3 = 0, F4 =0, F5 = 0, F6 = 0, F7 =0
- **30ns**: s2 = 0, s1 = 0, s0 = 1, **D = 1**, (001| 1)
  F0 = 0, **F1 = 1**, F2 = 0, F3 = 0, F4 =0, F5 = 0, F6 = 0, F7 =0

For the value of output F3:

- **60ns:** s2 = 0, s1 =1, s0 = 1, **D = 0**, (011| 0)
  F0 = 0, F1 = 0, F2 = 0, **F3 = 0**, F4 =0, F5 = 0, F6 = 0, F7 =0
- **70ns:** s2 = 0, s1 = 1, s0 = 1, **D = 1**, (011| 1)
  F0 = 0, F1 = 0, F2 = 0, **F3 = 0**, F4 =1, F5 = 0, F6 = 0, F7 =0

- 80ns to 160ns:



For instance:

For the value of output F4:

- **80ns:** s2 = 1, s1 = 0, s0 = 1, **D = 0**, (100| 0)
  F0 = 0, F1 = 0, F2 = 0, F3 = 0, **F4 =0**, F5 = 0, F6 = 0, F7 =0
- **90ns:** s2 = 1, s1 = 0, s0 = 1, **D = 1**, (100| 1)
  F0 = 0, F1 = 1, F2 = 0, F3 = 0, **F4 =0**, F5 = 0, F6 = 0, F7 =0

For the value of output F7:

- **140ns:** s2 = 1, s1 =1, s1 = 1, **D = 0**, (111| 0)
  F0 = 0, F1 = 0, F2 = 0, F3 = 0, F4 =0, F5 = 0, F6 = 0, **F7 =0**
- **150ns:** s2 = 1, s1 = 1, s1 = 1, **D = 1**, (111| 1)
  F0 = 0, F1 = 0, F2 = 0, F3 = 0, F4 =1, F5 = 0, F6 = 0, **F7 =1**

Circuit Implementations:
*The testbench remains the same for all three models.

> DataFlow Model:
```
library IEEE;
use IEEE.std_logic_1164.all;

entity demux8x1 is
port (s0,s1,s2,D: in bit;
        F0,F1,F2,F3,F4,F5,F6,F7: out bit);
end demux8x1;


architecture dataflow of demux8x1 is

begin
  F0 <= not(s2)and not(s1)and not(s0)and D;
  F1 <= not(s2)and not(s1)and s0 and D;
  F2 <= not(s2)and s1 and not(s0)and D;
  F3 <= not(s2)and s1 and s0 and D;
  F4 <= s2 and not(s1)and not(s0)and D;
  F5 <= s2 and not(s1)and s0 and D;
  F6 <= s2 and s1 and not(s0)and D;
  F7 <= s2 and s1 and s0 and D;
end dataflow;
```

> Behavioural/Algorithmic Model:
```
library IEEE;
use IEEE.std_logic_1164.all;

entity demux8x1 is
port (s0,s1,s2,D: in bit;
        F0,F1,F2,F3,F4,F5,F6,F7: out bit);
end demux8x1;

architecture algorithmic of demux8x1 is
begin
        p0:process(s0,s1,s2,D)
        variable s: bit_vector (2 downto 0);
begin
        s:= s2&s1&s0;
        -----000-----
        if s = "000" then
        F0 <= D; F1 <= '0'; F2 <= '0'; F3 <= '0';
        F4 <= '0'; F5 <= '0'; F6 <= '0'; F7 <= '0';
        -----001-----
        elsif s = "001" then
        F0 <= '0'; F1 <= D; F2 <= '0'; F3 <= '0';
        F4 <= '0'; F5 <= '0'; F6 <= '0'; F7 <= '0';
        -----010-----
        elsif s = "010" then
        F0 <= '0'; F1 <= '0'; F2 <= D; F3 <= '0';
        F4 <= '0'; F5 <= '0'; F6 <= '0'; F7 <= '0';
        -----011-----
        elsif s = "011" then
        F0 <= '0'; F1 <= '0'; F2 <= '0'; F3 <= D;
        F4 <= '0'; F5 <= '0'; F6 <= '0'; F7 <= '0';
        -----100-----
        elsif  s = "100" then
        F0 <= '0'; F1 <= '0'; F2 <= '0'; F3 <= '0';
        F4 <= D; F5 <= '0'; F6 <= '0'; F7 <= '0';

        -----101-----
        elsif  s = "101" then
        F0 <= '0'; F1 <= '0'; F2 <= '0'; F3 <= '0';
        F4 <= '0'; F5 <= D; F6 <= '0'; F7 <= '0';
        -----110-----
        elsif s = "110" then
        F0 <= '0'; F1 <= '0'; F2 <= '0'; F3 <= '0';
        F4 <= '0'; F5 <= '0'; F6 <= D; F7 <= '0';
        -----111-----
        elsif s = "111" then
        F0 <= '0'; F1 <= '0'; F2 <= '0'; F3 <= '0';
        F4 <= '0'; F5 <= '0'; F6 <= '0'; F7 <= D;
        end if;
  end process;
end algorithmic;
```

> Structural Model:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity demux8x1 is
port (s0,s1,s2,D: in bit;
        F0,F1,F2,F3,F4,F5,F6,F7: out bit);
end demux8x1;

architecture structure of demux8x1 is
component and4
port(X0,X1,X2,X3:in bit; O:out bit);
end component;

component inv
port (X1: in bit; O: out bit);
end component;

signal i0,i1,i2:bit;
begin
```

```
u0: inv port map (s0,i0);
u1: inv port map (s1,i1);
u2: inv port map (s2,i2);
u3: and4 port map (i2,i1,i0,D,F0);
u4: and4 port map (i2,i1,s0,D,F1);
u5: and4 port map (i2,s1,i0,D,F2);
u6: and4 port map (i2,s1,s0,D,F3);
u7: and4 port map (s2,i1,i0,D,F4);
u8: and4 port map (s2,i1,s0,D,F5);
u9: and4 port map (s2,s1,i0,D,F6);
u10: and4 port map (s2,s1,s0,D,F7);
end structure;

entity and4 is -- Πύλη AND 4 εισόδων
port(X0,X1,X2,X3: in bit; O: out bit);
end and4;

architecture behave of and4 is
begin
O<= X0 and X1 and X2 and X3;
end behave;

entity inv is -- Αντιστροφέας
port (X1: in bit; O: out bit);
end inv;

architecture behave of inv is
begin
O<= not X1;
end behave;
```

Testbench

```
library IEEE;
use IEEE.std_logic_1164.all;

entity test_demux8x1 is
end test_demux8x1;

architecture testbench of test_demux8x1 is
component demux8x1
port (s0,s1,s2,D: in bit;
        F0,F1,F2,F3,F4,F5,F6,F7: out bit);
end component;

signal tb_D: bit; signal tb_s0: bit;
signal tb_s1: bit; signal tb_s2: bit;
signal tb_F0: bit; signal tb_F1: bit;
signal tb_F2: bit; signal tb_F3: bit;
signal tb_F4: bit; signal tb_F5: bit;
signal tb_F6: bit; signal tb_F7: bit;
```

```
begin
u0: demux8x1 port map(
D => tb_D, s0=>tb_s0, s1=>tb_s1, s2=>tb_s2,
F0=>tb_F0, F1=>tb_F1, F2=>tb_F2, F3=>tb_F3,
F4=>tb_F4, F5=>tb_F5, F6=>tb_F6, F7=>tb_F7);

process
begin
-----000-----
--  D = 0  -> F0 = 0
tb_D <= '0';
tb_s0 <= '0' ;
tb_s1 <= '0' ;
tb_s2 <= '0' ;
wait for 10 ns;
--  D = 1 -> F0 = 1
tb_D <= '1';
tb_s0 <= '0' ;
tb_s1 <= '0' ;
tb_s2 <= '0' ;
wait for 10 ns;

-----001-----
-- D = 0 -> F1 = 0
tb_D <= '0';
tb_s0 <= '1' ;
tb_s1 <= '0' ;
tb_s2 <= '0' ;
wait for 10 ns;
-- με D = 1 -> F1 = 1
tb_D <= '1';
tb_s0 <= '1' ;
tb_s1 <= '0' ;
tb_s2 <= '0' ;
wait for 10 ns;

-----010-----
--  D = 0 -> F2 = 0
tb_D <= '0';
tb_s0 <= '0' ;
tb_s1 <= '1' ;
tb_s2 <= '0' ;
wait for 10 ns;
--  D = 1 -> F2 = 1
tb_D <= '1';
tb_s0 <= '0' ;
tb_s1 <= '1' ;
tb_s2 <= '0' ;
wait for 10 ns;
```

```
-----011-----
--  D = 0 -> F3 =0
tb_D <= '0';
tb_s0 <= '1' ;
tb_s1 <= '1' ;
tb_s2 <= '0' ;
wait for 10 ns;
--  D = 1 -> F3 = 1
tb_D <= '1';
tb_s0 <= '1' ;
tb_s1 <= '1' ;
tb_s2 <= '0' ;
wait for 10 ns;

-----100-----
--  D = 0 -> F4 = 0
tb_D <= '0';
tb_s0 <= '0' ;
tb_s1 <= '0' ;
tb_s2 <= '1' ;
wait for 10 ns;
--  D = 1 -> F4 = 1
tb_D <= '1';
tb_s0 <= '0' ;
tb_s1 <= '0' ;
tb_s2 <= '1' ;
wait for 10 ns;

-----101-----
--  D = 0 -> F5 = 0
tb_D <= '0';
tb_s0 <= '1' ;
tb_s1 <= '0' ;
tb_s2 <= '1' ;
wait for 10 ns;
--  D = 1 -> F5 = 1
tb_D <= '1';
tb_s0 <= '1' ;
tb_s1 <= '0' ;
tb_s2 <= '1' ;
wait for 10 ns;

-----110-----
--  D = 0 -> F6 = 0
tb_D <= '0';
tb_s0 <= '0' ;
tb_s1 <= '1' ;
tb_s2 <= '1' ;
wait for 10 ns;
```

```
--  D = 1 -> F6 = 1
tb_D <= '1';
tb_s0 <= '0' ;
tb_s1 <= '1' ;
tb_s2 <= '1' ;
wait for 10 ns;

-----111-----
--  D = 0 -> F7 = 0
tb_D <= '0';
tb_s0 <= '1' ;
tb_s1 <= '1' ;
tb_s2 <= '1' ;
wait for 10 ns;
--  D = 1 -> F7 = 1
tb_D <= '1';
tb_s0 <= '1' ;
tb_s1 <= '1' ;
tb_s2 <= '1' ;
wait for 10 ns;
end process;
end testbench;
```