# De-multiplexer

## Theory: -

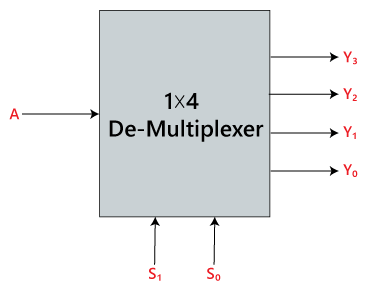
A De-multiplexer is a combinational circuit that has only 1 input line and 2N output lines. Simply, the multiplexer is a single-input and multi-output combinational circuit. The information is received from the single input lines and directed to the output line. On the basis of the values of the selection lines, the input will be connected to one of these outputs. De-multiplexer is opposite to the multiplexer.

Unlike encoder and decoder, there are n selection lines and 2n outputs. So, there is a total of 2npossible combinations of inputs. De-multiplexer is also treated as **De-mux**.

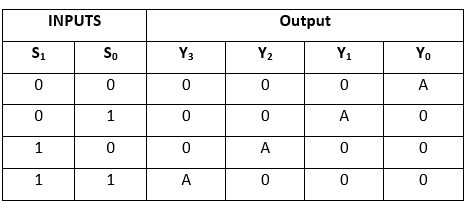
### 1×4 De-multiplexer:

In 1 to 4 De-multiplexer, there are total of four outputs, i.e., Y0, Y1, Y2, and Y3, 2 selection lines, i.e., S0 and S1 and single input, i.e., A. On the basis of the combination of inputs which are present at the selection lines S0 and S1, the input be connected to one of the outputs. The block diagram and the truth table of the 1**×**4 multiplexer are given below.

### Block Diagram:



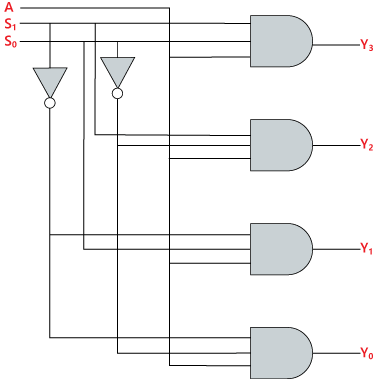
### Truth Table:



The logical expression of the term Y is as follows:

Y0=S1' S0' A  
y1=S1' S0 A  
y2=S1 S0' A  
y3=S1 S0 A

Logical circuit of the above expressions is given below:



## Source Code

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity demux\_1to4 is

port(

F : in STD\_LOGIC;

S0,S1: in STD\_LOGIC;

A,B,C,D: out STD\_LOGIC

);

end demux\_1to4;

architecture bhv of demux\_1to4 is

begin

process (F,S0,S1) is

begin

if (S0 ='0' and S1 = '0') then

A <= F;

elsif (S0 ='1' and S1 = '0') then

B <= F;

elsif (S0 ='0' and S1 = '1') then

C <= F;

else

D <= F;

end if;

end process;

end bhv;

## Testbench Code

library IEEE;

use IEEE.Std\_logic\_1164.all;

use IEEE.Numeric\_Std.all;

entity demux\_1to4\_tb is

end;

architecture bench of demux\_1to4\_tb is

component demux\_1to4

port(

F : in STD\_LOGIC;

S0,S1: in STD\_LOGIC;

A,B,C,D: out STD\_LOGIC

);

end component;

signal F: STD\_LOGIC;

signal S0,S1: STD\_LOGIC;

signal A,B,C,D: STD\_LOGIC ;

begin

uut: demux\_1to4 port map ( F => F,

S0 => S0,

S1 => S1,

A => A,

B => B,

C => C,

D => D );

stimulus: process

begin

-- Put initialisation code here

F <= '1';

S0 <= '0'; S1 <= '0';

wait for 10 ns;

S0 <= '1'; S1 <= '0';

wait for 10 ns;

S0 <= '0'; S1 <= '1';

wait for 10 ns;

S0 <= '1'; S1 <= '1';

wait for 10 ns;

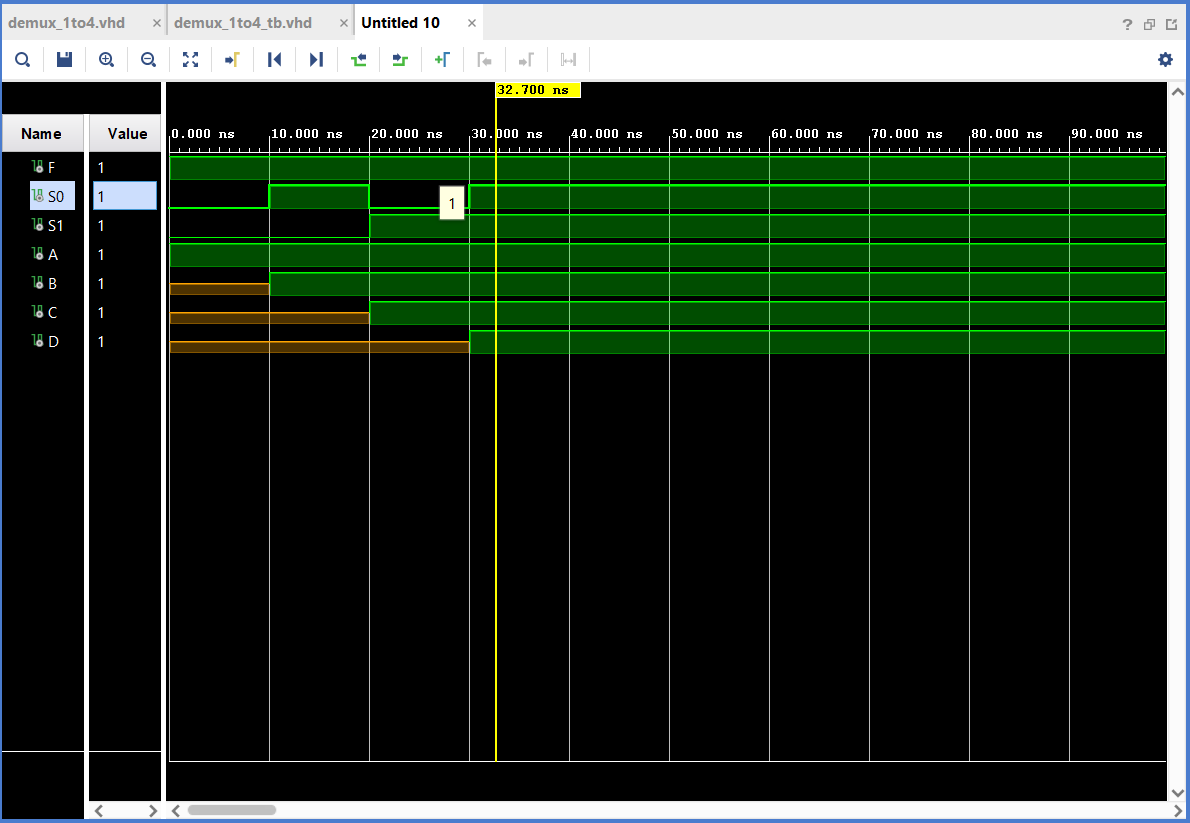
-- Put test bench stimulus code here

wait;

end process;

end;

## Observation



## Output

