Implementation of Multiplexer and De-Multiplexer using VHDL

▪ OBJECTIVE:

To implement **Multiplexer and De-Multiplexer** using VHDL and verify its waveform in Xilinx.

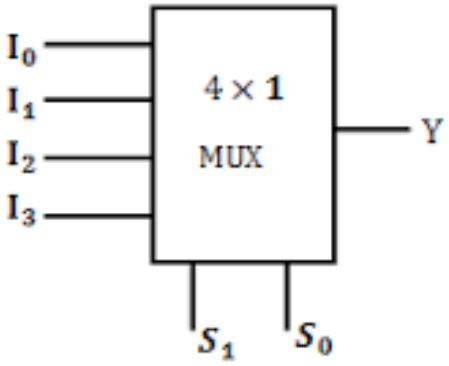
▪ INTRODUCTION:

1. Multiplexer:

A **multiplexer (MUX)** is a digital electronic device that selects one of many input signals and forwards the selected input to a single output line. It acts as a data selector. Multiplexers are widely used in digital circuits for efficient data transmission and signal routing.

4-to-1 MUX Components:

1. **Inputs** (4): The four data inputs are usually denoted as I_0, I_1, I_2, I_3.
2. **Select Lines** (2): The two select lines are denoted as S_1 and S_0.
   - The number of select lines needed is calculated by $\log_2(n)$, where n is the number of inputs. In this case, $\log_2(4) = 2$, so we need 2 select lines.
3. **Output** (1): The multiplexer has one output line, which carries the selected input signal.



Truth table

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Explanation:

- The output Y is selected based on the binary value of the select lines S_1 and S_0.
- When $S_1 = 0$ and $S_0 = 0$, the output Y is equal to input I_0.
- When $S_1 = 0$ and $S_0 = 1$, the output Y is equal to input I_1.
- When $S_1 = 1$ and $S_0 = 0$, the output Y is equal to input I_2.
- When $S_1 = 1$ and $S_0 = 1$, the output Y is equal to input I_3.

Application Area:

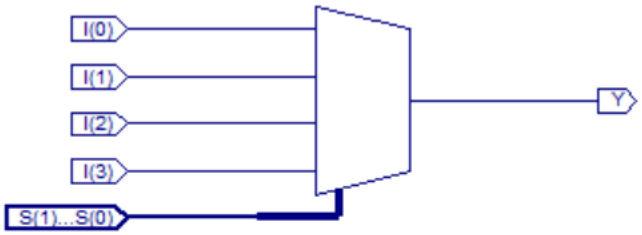
- **Data selection**: Multiplexers are commonly used in digital circuits for selecting data from different sources.
- **Communication systems**: In multiplexed transmission, multiple data streams share the same communication medium, reducing the number of lines required.
- **Bus systems**: A multiplexer can allow one of many data buses to be connected to a single output bus.

  This design allows for efficient routing and data management in digital circuits.

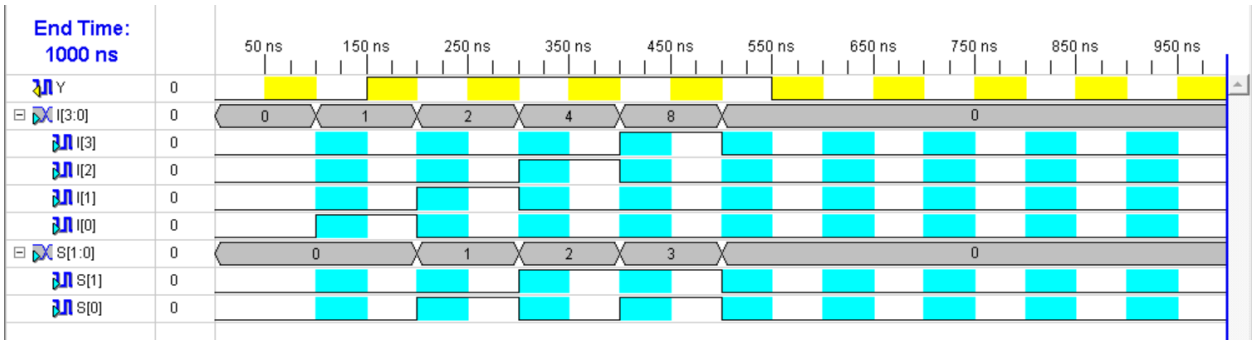
Source Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity MUx is
    Port ( I : in  STD_LOGIC_VECTOR (3 downto 0);
           S : in  STD_LOGIC_VECTOR (1 downto 0);
           Y : out  STD_LOGIC);
end MUx;

architecture Behavioral of MUx is

begin
    process (S,I)
    begin
        case (S) is
            when "00" => Y <= I(0); -- Select I0
            when "01" => Y <= I(1); -- Select I1
            when "10" => Y <= I(2); -- Select I2
            when "11" => Y <= I(3); -- Select I3
            when others => Y <= '0'; -- Default case
        end case;
    end process;
end Behavioral;
```

- **RTL (Register Transfer Level) Schematic**:

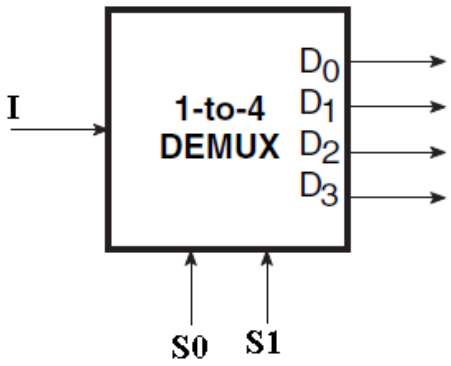


- **Test Bench Waveform:**



2. De-Mux:

A **1-to-4 De-Multiplexer (DEMUX)** is a digital circuit that takes a single input signal and routes it to one of four possible output lines based on the value of the select lines. In this case, there is **1 input** and **4 outputs**. The **select lines (S)** will determine which output line gets the signal.



I	Select		O/P			
	S0	S1	D_0	D_1	D_2	D_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Application Area:

A **Demultiplexer (DEMUX)** is used in various applications, such as:

1. **Communication Systems**: Routes signals to multiple receivers.
2. **Data Distribution**: Distributes data to multiple peripherals or devices.
3. **Memory Systems**: Routes memory addresses to specific memory blocks.
4. **Control Systems**: Directs control signals to different devices.
5. **Audio/Video Routing**: Routes audio or video signals to multiple output channels.
6. **Networking**: Routes data packets in network devices like routers.
7. **Signal Processing**: Separates signals for different processing paths.
8. **FPGA/ASIC Designs**: Manages signal routing between different modules.

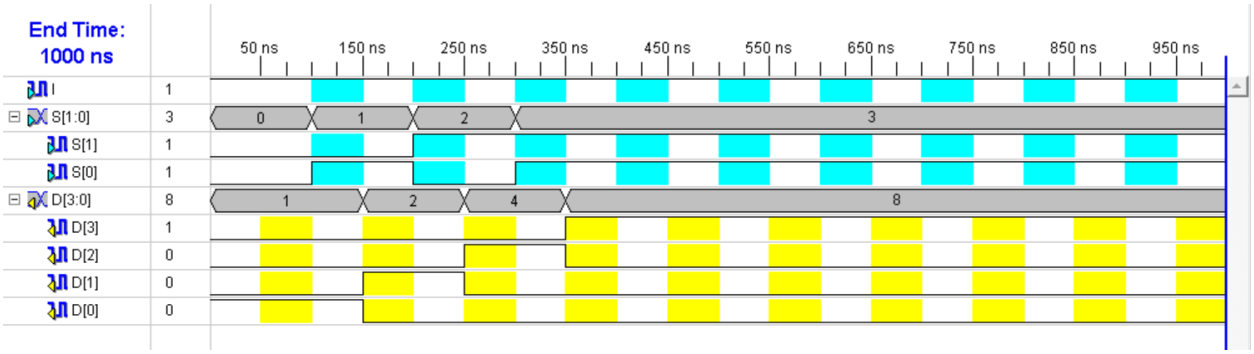
In short, DEMUX is used for **routing** a single input to multiple outputs based on select signals.

- ## Source Code:

```
entity DeMUX is
    Port ( I : in  STD_LOGIC;
           S : in  STD_LOGIC_VECTOR (1 downto 0);
           D : out  STD_LOGIC_VECTOR (3 downto 0));
end DeMUX;

architecture Behavioral of DeMUX is

begin
process (I, S)
    begin
        -- Default: All outputs are 0
        D <= "0000";
        -- Route data to the selected output based on select lines
        if (S = "00") then
            D(0) <= I;  -- Route I to D0
        elsif (S = "01") then
            D(1) <= I;  -- Route I to D1
        elsif (S = "10") then
            D(2) <= I;  -- Route I to D2
        elsif (S = "11") then
            D(3) <= I;  -- Route I to D3
        end if;
    end process;
end Behavioral;
```

- **RTL (Register Transfer Level) Schematic**:

- **Test Bench Waveform:**



▪ Discussion:

In this embedded lab, we designed and implemented a **4:1 Multiplexer (MUX)** and a **4:1 Demultiplexer (DEMUX)** using **VHDL** in **Xilinx** tools.

- **4:1 MUX**: We used select lines to route one of four inputs to a single output.
- **4:1 DEMUX**: We routed a single input to one of four outputs using select lines.

Both designs were tested using **simulation** to verify correct functionality. The lab provided hands-on experience with **VHDL syntax**, **digital logic design**, and **Xilinx tools** (Vivado/ISE).

▪ Conclusion:

This lab enhanced our understanding of data routing using MUX and DEMUX circuits and reinforced the importance of simulation and VHDL for efficient digital system design.