DIGITAL DESIGN (CSET-105)

Assignment – Mux-Demux

NAME:- Aryan Gahlot

Enrolment no. :- S24CSEU0701

BATCH NO. :- 43

Q1) What is multiplexer? Discuss in brief with appropriate block diagram and examples.

Answer:

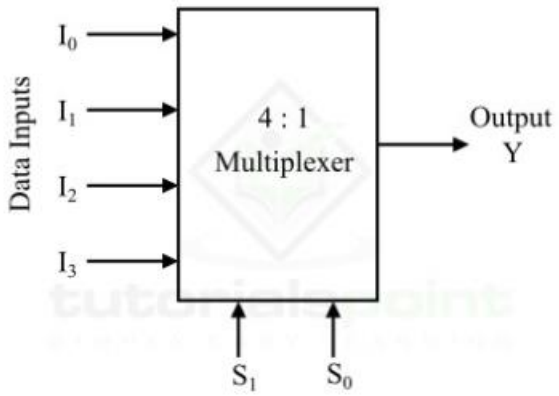
A **Multiplexer (MUX)** is a combinational circuit that selects one of the multiple input signals and forwards the selected input to a single output line. It acts as a data selector, allowing multiple signals to share a single transmission line.

Block Diagram of a Multiplexer

A general **n-to-1** multiplexer has:

A general **n-to-1** multiplexer has:

- **n input lines** $(I_0, I_1, I_2, \ldots, I_{n-1})$
- **m selection lines** $(S_0, S_1, \ldots, S_{m-1})$ where $m = \log_2 n$
- **1 output line (Y)**



S1	S0	Y(Output)
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Examples of Multiplexers

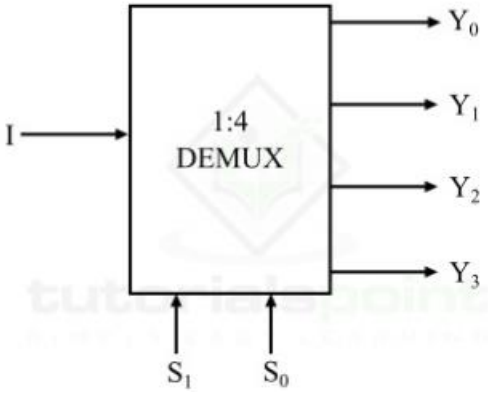
1. **Communication Systems**: Used to transmit multiple signals over a single communication channel.
2. **Data Routing**: Used in computer networks to switch between different data sources.
3. **Arithmetic and Logic Units (ALU)**: Used in processors to select operations.
4. **Memory Access**: Helps in selecting data from different memory locations.

Q2) What is Demultiplexer? Discuss in brief with appropriate block diagram and examples.

Answer:

A **Demultiplexer (DEMUX)** is a combinational circuit that takes a single input and directs it to one of the multiple output lines based on the selection inputs. It performs the reverse operation of a **Multiplexer (MUX)** and is often called a **data distributor**.

- 1 input line (D)
- m selection lines ($S_0, S_1, \ldots, S_{m-1}$) where $m = \log_2 n$
- n output lines ($Y_0, Y_1, Y_2, \ldots, Y_{n-1}$)



S1	S0	Output
0	0	Y0
0	1	Y1
1	0	Y2
1	1	Y3

Examples of Demultiplexers

1. **Communication Systems**: Used to route a single incoming signal to multiple destinations.
2. **Memory Selection**: Used in RAM for selecting specific memory locations.
3. **Arithmetic Circuits**: Used in ALUs to distribute results to different registers.
4. **Serial to Parallel Data Conversion**: Used to convert serial data (one input) into parallel outputs.

Q3)Write truth table and Verilog code to implement 2:1 Multiplexer.

Answer:

```
[2025-02-24 11:28:56 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile multiplexer_2to1.vcd opened for output.
Io I1 S | Y
-------------
0  0  0 | 0
1  0  0 | 1
0  1  0 | 0
1  1  0 | 1
0  0  1 | 0
1  0  1 | 0
0  1  1 | 1
1  1  1 | 1
testbench.sv:29: $finish called at 80 (1s)
Finding VCD file...
./multiplexer_2to1.vcd
[2025-02-24 11:28:57 UTC] Opening EPWave...
Done
```

Design:

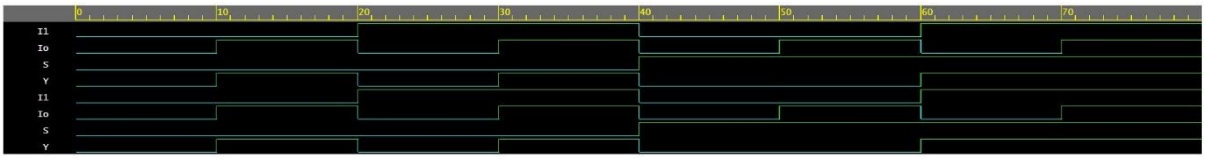
```
module multiplexer_2to1 (input wire Io, input wire I1, input wire S,output wire Y);

assign Y = (~S&Io)|(S&I1);

endmodule
```

Testbench:

```
module tb_multiplexer_2to1;


   reg Io;

   reg I1;

   reg S;

   wire Y;


 multiplexer_2to1 uut (.Io(Io), .I1(I1), .S(S), .Y(Y));
```

```
  initial begin
    $dumpfile("multiplexer_2to1.vcd");
    $dumpvars(0, tb_multiplexer_2to1);

      $display("Io I1 S | Y");
      $display("-------------");
      $monitor("%b  %b  %b | %b", Io, I1, S, Y);

      S = 0; Io = 0; I1 = 0; #10;
      S = 0; Io = 1; I1 = 0; #10;
      S = 0; Io = 0; I1 = 1; #10;
      S = 0; Io = 1; I1 = 1; #10;

      S = 1; Io = 0; I1 = 0; #10;
      S = 1; Io = 1; I1 = 0; #10;
      S = 1; Io = 0; I1 = 1; #10;
      S = 1; Io = 1; I1 = 1; #10;

      $finish;
  end

endmodule
```

Waveform:



Q4) Write truth table and Verilog code to implement 1:4 Demultiplexer.

Answer:

```
[2025-02-24 11:36:08 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile demultiplexer_4to1.vcd opened for output.
 I  S1 S0 | Y0 Y1 Y2 Y3
-----------------------
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
0  0  0  | 0  0  0  0
0  0  1  | 0  0  0  0
0  1  0  | 0  0  0  0
0  1  1  | 0  0  0  0
testbench.sv:37: $finish called at 80 (1s)
Finding VCD file...
./demultiplexer_4to1.vcd
[2025-02-24 11:36:09 UTC] Opening EPWave...
Done
```

Design:

```
module demultiplexer_4to1 (input wire I,input wire S0,input wire S1,output wire Y0,
   output wire Y1,output wire Y2,output wire Y3);


   assign Y0 = (~S1 & ~S0 & I);
   assign Y1 = (~S1 &  S0 & I);
   assign Y2 = ( S1 & ~S0 & I);
   assign Y3 = ( S1 &  S0 & I);

endmodule
```

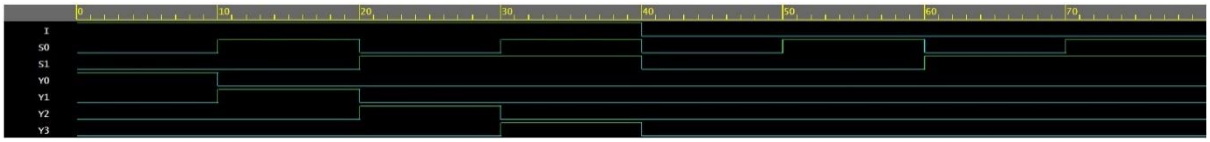
Testbench:

```
module tb_demultiplexer_4to1;
```

```
reg I;
reg S0;
reg S1;
wire Y0, Y1, Y2, Y3;

demultiplexer_4to1 uut (
    .I(I),
    .S0(S0),
    .S1(S1),
    .Y0(Y0),
    .Y1(Y1),
    .Y2(Y2),
    .Y3(Y3)
);

initial begin
    $dumpfile("demultiplexer_4to1.vcd");
    $dumpvars(0, tb_demultiplexer_4to1);

    $display(" I  S1 S0 | Y0 Y1 Y2 Y3");
    $display("----------------------");
    $monitor("%b  %b  %b  | %b  %b  %b  %b", I, S1, S0, Y0, Y1, Y2, Y3);
```

```
        I = 1; S1 = 0; S0 = 0; #10;

        I = 1; S1 = 0; S0 = 1; #10;

        I = 1; S1 = 1; S0 = 0; #10;

        I = 1; S1 = 1; S0 = 1; #10;


        I = 0; S1 = 0; S0 = 0; #10;

        I = 0; S1 = 0; S0 = 1; #10;

        I = 0; S1 = 1; S0 = 0; #10;

        I = 0; S1 = 1; S0 = 1; #10;


        $finish;

    end


endmodule
```

Waveform:



Q 5) Implement the Boolean function F(A, B, C) = Σm(3, 5, 6, 7) with a 4:1 multiplexer. Write a Verilog code for the above design

Answer:

A	B	C	F(A,B,C)
0	0	0	0

0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Design:

```
module mux4to1 (
   input wire A, B, C,
   output wire F
);
   wire S1, S0;
   assign S1 = A;
   assign S0 = B;

   wire I0, I1, I2, I3;
   assign I0 = 0;
   assign I1 = C;
   assign I2 = C;
   assign I3 = 1;

   assign F = (~S1 & ~S0 & I0) |
         (~S1 & S0 & I1)  |
         (S1 & ~S0 & I2)  |
         (S1 & S0 & I3);
endmodule
```

Testbench:

```
module tb_mux4to1;
   reg A, B, C;
   wire F;

   mux4to1 uut (.A(A), .B(B), .C(C), .F(F));

   initial begin
      $monitor("A=%b, B=%b, C=%b -> F=%b", A, B, C, F);

      A = 0; B = 0; C = 0; #10;
      A = 0; B = 0; C = 1; #10;
      A = 0; B = 1; C = 0; #10;
      A = 0; B = 1; C = 1; #10;
      A = 1; B = 0; C = 0; #10;
      A = 1; B = 0; C = 1; #10;
      A = 1; B = 1; C = 0; #10;
      A = 1; B = 1; C = 1; #10;

      $finish;
   end
endmodule
```