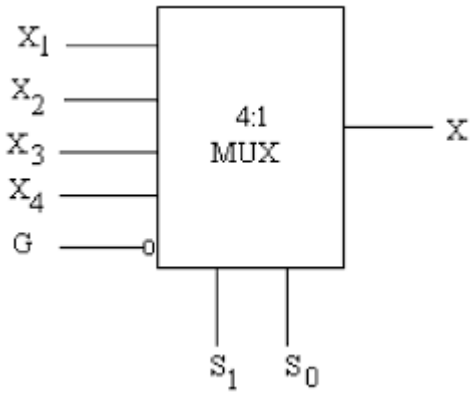
Assignment 6
Rishabh Jain
S24CSEU0324

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

A **Multiplexer (MUX)** is a combinational logic circuit that selects one input from multiple inputs and forwards it to a single output line. It acts like a digital switch, allowing one input to be transmitted at a time based on the control signals (also called **selection lines**).



Example Usage of Multiplexers:

1. **Data Selection**: Select data from different sources (e.g., ALU operations).
2. **Communication Systems**: Transmit multiple signals over a single communication line.
3. **Memory Addressing**: Select different memory locations.
4. **Arithmetic Circuits**: Use in arithmetic logic units (ALU) for operations.

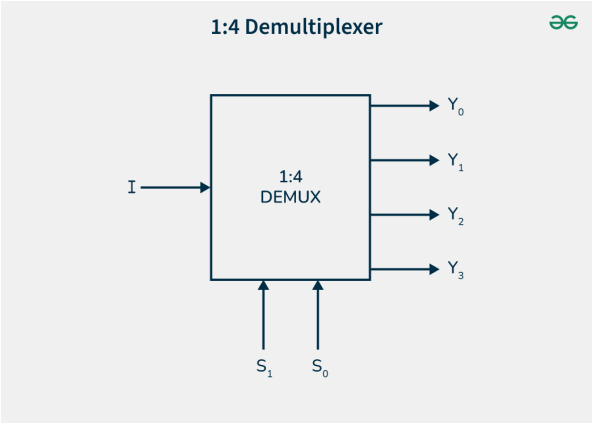
Types of Multiplexers:

1. **2-to-1 Multiplexer**: Selects between 2 inputs (1 selection line).
2. **4-to-1 Multiplexer**: Selects between 4 inputs (2 selection lines).
3. **8-to-1 Multiplexer**: Selects between 8 inputs (3 selection lines).
4. **16-to-1 Multiplexer**: Selects between 16 inputs (4 selection lines)

Q3)What is Demultiplexer? Discuss in brief with appropriate block diagram and examples.

Demultiplexer (DEMUX)

A **Demultiplexer (DEMUX)** is a combinational circuit that takes a single input and distributes it to one of several outputs based on the control signals (selection lines). It performs the reverse operation of a multiplexer (MUX).



Applications of Demultiplexers:

1. **Data Distribution**: Sending data from a single source to multiple destinations.
2. **Communication Systems**: Routing information to different communication channels.
3. **Memory Access**: Selecting specific memory addresses for reading or writing.
4. **Digital Displays**: Controlling LED segments and other display devices.

Types of Demultiplexers:

1. **1-to-2 DEMUX**: 1 input to 2 outputs (1 selection line).
2. **1-to-4 DEMUX**: 1 input to 4 outputs (2 selection lines).
3. **1-to-8 DEMUX**: 1 input to 8 outputs (3 selection lines).
4. **1-to-16 DEMUX**: 1 input to 16 outputs (4 selection lines).

Q3. Write truth table and Verilog code to implement 4:1 Multiplexer.

//Rishabh Jain: S24CSEU0324

Design file

```
module Q3(input a, input b, input c, input d, input s1, input s2, output y); assign y=(s1 == 0 && s2 == 0) ? a:

        (s1 == 0 && s2 == 1) ? b:

        (s1 == 1 && s2 == 0) ? c:

        (s1 == 1 && s2 == 1) ? d:

1'b0;
```

Endmodule

Test Bench

```
module tb_Q3;

reg a, b, c, d, s1, s2; wire y;

Q3 uut(.a(a), .b(b), .c(c), .d(d), .s1(s1), .s2(s2), .y(y)); initial begin

        $dumpfile("Q3dump.vcd");

        $dumpvars(1);

        end

initial begin

 $monitor("At Time=%0t: a=%b b=%b c=%b d=%b s1=%b s2=%b y=%b ", $time, a, b, c, d, s1, s2, y);

        end

initial begin a=0; b=0; c=1; d=0; s1 = 0; s2 = 0;

#10;

s1 = 0; s2 = 1; #10;

s1 = 1; s2 = 0; #10;

s1 = 1; s2 = 1;

#10;

        end

endmodule
```

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

//Rishabh Jain : S24CSEU0324

Design File

```
module Q4(input i, input s1, input s2, output y0, output y1, output y2, output y3); assign y0 = (s1 == 0
&& s2 == 0) ? i:

0; assign y1 = (s1 == 0 && s2 == 1) ? i:

0; assign y2 = (s1 == 1 && s2 == 0) ? i:
```

```
0; assign y3 = (s1 == 1 && s2 == 1) ? i:

0;

Endmodule
```

Test Bench

```
module tb_Q4;

reg i, s1, s2; wire y0, y1, y2, y3;

Q4 uut(.i(i), .s1(s1), .s2(s2), .y0(y0), .y1(y1), .y2(y2), .y3(y3));

initial begin

        $dumpfile("Q4dump.vcd");

        $dumpvars(1);

        end

initial begin

 $monitor("At Time=%0t: i=%b y0=%b y1=%b y2=%b s1=%b s2=%b y3=%b ", $time, i, y0, y1, y2, s1, s2, y3);

        end

initial begin

i = 1; s1 = 0; s2 = 0; #10; s1 = 0; s2 = 1; #10; s1 = 1; s2 = 0; #10;

s1 = 1; s2 = 1;

#10;

        end

Endmodule
```

5. Design a combinational circuit using 1: 8 De-Multiplexer constructed using 1:2 De-Multiplexer. Write a Verilog code in dataflow mode for the above design.

```
// Rishabh Jain S24CSEU0324

module demux_1to2(

  input D,

  input S,

  output Y0, Y1
```

```
);
  assign Y0 = (~S) & D;
  assign Y1 = S & D;
endmodule

module demux_1to8(
  input D,
  input [2:0] S,
  output [7:0] Y
);
  wire Y_upper, Y_lower;
  wire [3:0] Y_mid;

  demux_1to2 demux1(D, S[2], Y_upper, Y_lower);

  demux_1to2 demux2(Y_upper, S[1], Y_mid[0], Y_mid[1]);
  demux_1to2 demux3(Y_lower, S[1], Y_mid[2], Y_mid[3]);

  demux_1to2 demux4(Y_mid[0], S[0], Y[0], Y[1]);
  demux_1to2 demux5(Y_mid[1], S[0], Y[2], Y[3]);
  demux_1to2 demux6(Y_mid[2], S[0], Y[4], Y[5]);
  demux_1to2 demux7(Y_mid[3], S[0], Y[6], Y[7]);
endmodule

module tb_demux_1to8;
  reg D;
  reg [2:0] S;
  wire [7:0] Y;
```

```
  demux_1to8 uut(.D(D), .S(S), .Y(Y));

  initial begin
    $dumpfile("Q5dump.vcd");
    $dumpvars(0, tb_demux_1to8);

    $monitor("Time=%0t | D=%b, S=%b, Y=%b", $time, D, S, Y);

    D = 0; S = 3'b000; #10;
    D = 1; S = 3'b000; #10;
    D = 1; S = 3'b001; #10;
    D = 1; S = 3'b010; #10;
    D = 1; S = 3'b011; #10;
    D = 1; S = 3'b100; #10;
    D = 1; S = 3'b101; #10;
    D = 1; S = 3'b110; #10;
    D = 1; S = 3'b111; #10;
    D = 0; S = 3'b101; #10;

    $finish;
  end
endmodule
```