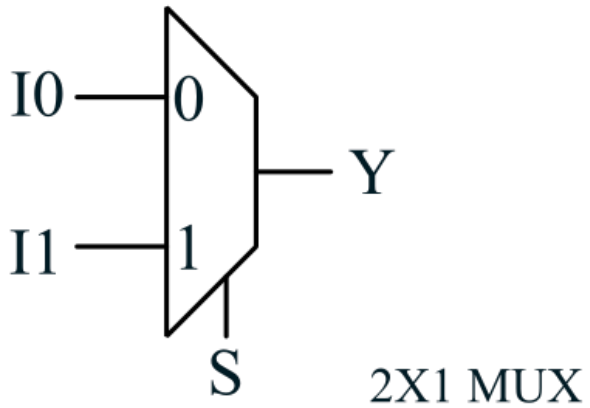
2 x 1 MUX

In 2×1 multiplexer, there are only two inputs, i.e., A_0 and A_1, 1 selection line, i.e., S_0 and single outputs, i.e., Y. On the basis of the combination of inputs which are present at the selection line S^0, one of these 2 inputs will be connected to the output. The block diagram and the truth table of the 2×1 multiplexer are given below.



2X1 MUX

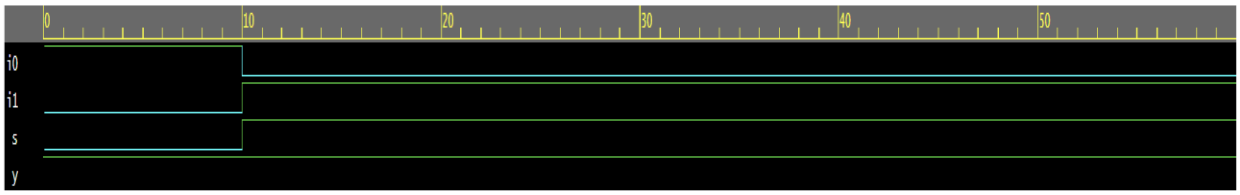
RTL CODE:

```
module mux(input i0,input i1,input s,output y);
  assign y=s?i1:i0;
endmodule
```

TESTBENCH

```
module test;
  reg i0,i1,s;
  wire y;
  mux a1(i0,i1,s,y);
  initial begin
```

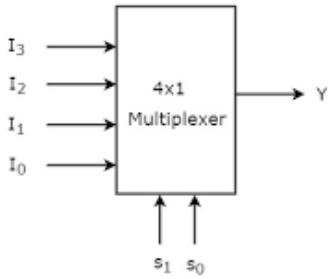
```
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
  initial begin
    i0=1'b1;i1=1'b0;s=1'b0;
    #10 i0=1'b0;i1=1'b1;s=1'b1;
  end
  initial begin
    #10 $finish();
  end
endmodule
```



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

4X1 MUX

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & s_0 and one output Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.



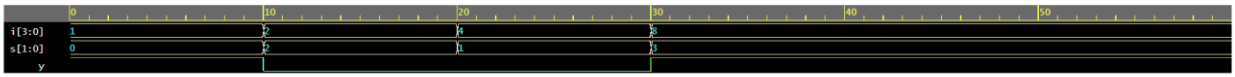
RTL CODE:

```
module mux(input [3:0]i,input[1:0]s,output reg y);
  always @( * )
    begin
      case(s)
        2'b00: y=i[0];
        2'b01: y=i[1];
        2'b10: y=i[2];
        2'b11: y=i[3];
      endcase
    end
endmodule
```

TESTBENCH:

```
module test;
  reg [3:0]i;
  reg [1:0]s;
  wire y;
  mux a1(i,s,y);
```

```
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end
  initial begin
    s[0]=1'b0;s[1]=1'b0;i[0]=1'b1;i[1]=1'b0;i[2]=1'b0;i[3]=1'b0;
    #10 s[0]=1'b0;s[1]=1'b1;i[0]=1'b0;i[1]=1'b1;i[2]=1'b0;i[3]=1'b0;
    #10 s[0]=1'b1;s[1]=1'b0;i[0]=1'b0;i[1]=1'b0;i[2]=1'b1;i[3]=1'b0;
    #10 s[0]=1'b1;s[1]=1'b1;i[0]=1'b0;i[1]=1'b0;i[2]=1'b0;i[3]=1'b1;
  end
  initial begin
    #60 $finish();
  end
endmodule
```



Note: To revert to EPWave opening in a new browser window, set that option on your user page.