VLSI Engineering Lab (Digital)

Experiment 5 – Clock and Demux

Utkarsh Patel (18EC30048)

---

1 Objective

- Design a clock with time period 40 ns and a duty cycle of 25% by using always and initial statements. The value of clock at t = 0 should be initialized to 0
- Design a 1-to-4 DEMUX using 1-to-2 DEMUX and switch level Verilog description

2 Theory

2.1 Clock signal

A clock signal is defined as the one which synchronizes the state transitions by keeping all the registers/state elements in synchronization. In common terminology, a clock signal is a signal that is used to trigger sequential devices (flip-flops in general). By this, we mean that on the active state/edge of clock, data at input of flip-flops propagates to the output. This propagation is termed as state transition. Clock signals occupy a very important place throughout the chip design stages. Since, the state transition happens on clock transition, the entire simulations including verification, static timing analysis and gate level simulations roam around these clock signals only. If Static timing analysis can be considered as a body, then clock is its blood. Also, during physical implementation of the design, special care has to be given to the placement and routing of clock elements, otherwise the design is likely to fail. Clock elements are responsible for almost half the dynamic power consumption in the design. That is why; clock has to be given the prime importance.

There are different terms related to clock signals, described below:

1. Leading and trailing clock edge: When clock signal transitions from '0' to '1', the clock edge is termed as leading edge. Similarly, when clock signal transitions from '1' to '0', the clock edge is termed as trailing edge.

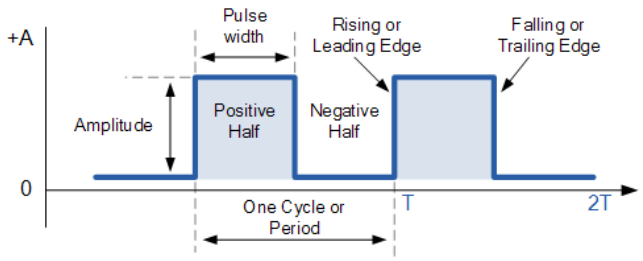

Fig. 1. Clock signal in VLSI circuits

2. Launch and capture edge: Launch edge is that edge of the clock at which data is launched by a flop. Similarly, capture edge is that edge of the clock at which data is capture by a flop.

3. Clock skew: Clock skew is defined as the difference in arrival times of clock signals at different leaf pins. Considering a set of flops, skew is the difference in the minimum and maximum arrival times of the clock signal. Global skew is the clock skew for the whole design. On the contrary, considering only a portion of the design, the skew is termed as local skew.

4. Duty cycle: Fraction of time for which clock is 'on' state for one time period is called duty cycle. From Fig. 1, it is given as:

$$\text{duty cycle} = \frac{\text{pulse width}}{\text{time period}}$$

2.2 Demultiplexer

A demultiplexer (or demux) is a device that takes a single input line and routes it to one of several digital output lines. A demultiplexer of 2^n outputs has n select lines, which are used to select which output line to send the input. A demultiplexer is also called a data distributor.

Demultiplexers can be used to implement general purpose logic. By setting the input to true, the demux behaves as a decoder.

1 to 2 demultiplexer

A 1 to 2 demultiplexer uses 1 select line (S) to determine which one of the 2 outputs (Y0, Y1) is routed from the input (D). Its simplified truth table is:

S	Y1	Y0
0	0	D
1	D	0

1 to 4 demultiplexer

A 1 to 4 multiplexer uses 2 select lines (S0, S1) to determine which one of the 4 outputs (Y0 - Y3) is routed from the input (D). Its characteristics can be described in the following simplified truth table.

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

1 to 4 DEMUX can be designed by cascading 1 to 2 DEMUX as follows:

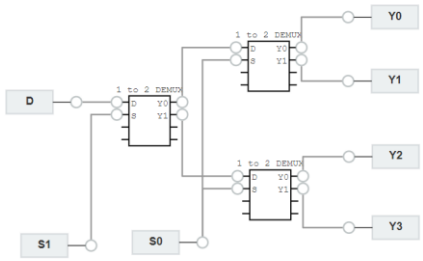

Fig. 2. Designing 1-to-4 DEMUX using 1-to-2 DEMUX

For this experiment, the 1-to-2 DEMUX module is designed using switch level Verilog description, i.e., using CMOS and NMOS.

For 1-to-2 DEMUX, the output terminals Y_0 and Y_1 are related to input terminal D and select terminal S as,

$$Y_0 = D \cdot \bar{S}; Y_1 = D \cdot S$$

This can be reduced as

$$Y_0 = D \cdot \bar{S} = \overline{\bar{D} + S}$$

$$Y_1 = D \cdot S = \overline{\bar{D} + \bar{S}}$$

Therefore, we need to implement a NOT gate and a NOR gate using switch level logic and used these gates to implement 1-to-2 DEMUX.

3 Verilog codes

3.1 Clock

```
`timescale 1ns/1ns

module clock (out);
  output out;
  reg CLK = 0;
  assign out = CLK;
  always begin
  #20
  CLK = 1; #10;
  CLK = 0; #10;
  end
endmodule

module clock_tb;
  wire OUT;
  clock c1(.out(OUT));
  initial begin
    $dumpfile("clock.vcd");
    $dumpvars;
    #400;
    $display("COMPLETED");
    $finish;
  end
endmodule
```

3.2 1-to-4 DEMUX

```
module my_not(input x, output f);
  supply1 vdd;
  supply0 gnd;
  pmos p1 (f, vdd, x);
  nmos n1 (f, gnd, x);
endmodule

module my_nor(input x, y, output f);
  supply0 gnd;
  tri1 f;
  nmos nx (f, gnd, x);
```

```
  nmos ny (f, gnd, y);
endmodule

module demux_1_2(out0,out1,s,inp);
  output out0,out1;
  input inp,s;
  wire inp_inv, s_inv;
  my_not not11 (inp, inp_inv);
  my_not not22 (s, s_inv);
  my_nor nor11 (inp_inv, s, out0);
  my_nor nor22 (inp_inv, s_inv, out1);

endmodule

module demux_1_4(out3,out2,out1,out0,s,in);

  output out0,out1,out2,out3;
  input in;
  input [1:0] s;
  wire [1:0] w;

  demux_1_2 dl1(.s(s[1]),.inp(in),.out1(w[1]),.out0(w[0]));
  demux_1_2 dl21(.s(s[0]),.inp(w[0]),.out1(out1),.out0(out0));
  demux_1_2 dl22(.s(s[0]),.inp(w[1]),.out1(out3),.out0(out2));

endmodule

module demux_1_4_tb;
reg IN;
reg [1:0] S;
wire O0,O1,O2,O3;
demux_1_4 dm(.s(S),.in(IN),.out0(O0),.out1(O1),.out2(O2),.out3(O3));

  initial begin
    $dumpfile("demux1x4_tb.vcd");
    $dumpvars;
    $monitor("Input = %b S = %b Output0 = %b Output1 = %b Output2 = %b Output3 = %b", IN, S, O0, O1,O2
, O3);
    IN = 0;
    S = 2'b00; #20;
    S = 2'b01; #20;
    S = 2'b10; #20;
    S = 2'b11; #20;
    $display("COMPLETED");
    $finish;
  end

endmodule
```

4 Output Log

4.1 1-to-4 DEMUX

```
C:\iverilog\DVLSI\exp5>iverilog demux.v

C:\iverilog\DVLSI\exp5>vvp a.out
VCD info: dumpfile demux1x4_tb.vcd opened for output.
Input = 1 S = 00 Output0 = 1 Output1 = 0 Output2 = 0 Output3 = 0
Input = 1 S = 01 Output0 = 0 Output1 = 1 Output2 = 0 Output3 = 0
Input = 1 S = 10 Output0 = 0 Output1 = 0 Output2 = 1 Output3 = 0
Input = 1 S = 11 Output0 = 0 Output1 = 0 Output2 = 0 Output3 = 1
COMPLETED
demux.v:68: $finish called at 80 (1s)
```

5 Waveforms

5.1 Clock signal

Fig. 3. The circuit generates clock signal `OUT` with time period of 40 ns. It can be observed that clock is in 'on'-state for about 10 ns for a given time period, hence having a duty cycle of 25%

5.2 1-to-4 DEMUX

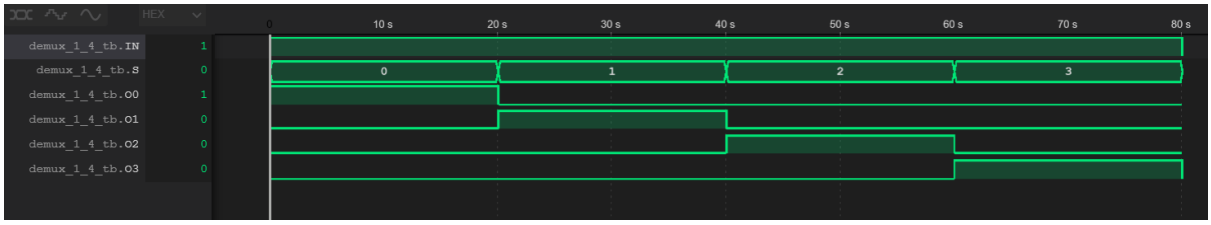

Fig. 4. The input `IN` to module is set to '1'. It can be observed that as the value of select pin `S[1:0]` is varied, input is being decoded to the corresponding output terminal. (`S[1:0]=00` for `O0`, `01` for `O1`, `10` for `O2`, `11` for `O3`)

6 Conclusion

- In this experiment, we were required to design a clock with time period of 40 ns and having a duty cycle of 25%.

- For this, for a given time period, the output of the clock is first initialized to 0. Then, after a delay of 20 ns, it is set to ‘on’ state, then a delay of 10 ns is imposed and its state is set to ‘off’ and another 10 ns delay is imposed. Hence, for a time period of 40 ns, clock is in ‘on’ state of 10 ns, and ‘off’ for 30 ns, hence having a duty cycle of 25%.

- In the next part of the experiment, 1-to-4 DEMUX was to be designed using 1-to-2 DEMUX, which in turn was to be designed using switch level Verilog description.

- For 1-to-2 DEMUX, the output terminals Y_0 and Y_1 are related to input terminal D and select terminal S as $Y_0 = D \cdot \bar{S}$ and $Y_1 = D \cdot S$. These can be reduced to nor gate logic as

$$Y_0 = D \cdot \bar{S} = \overline{\bar{D} + S}$$
$$Y_1 = D \cdot S = \overline{\bar{D} + \bar{S}}$$

- Therefore, we need to implement a NOT gate and a NOR gate using switch level logic and used these gates to implement 1-to-2 DEMUX.

- The 1-to-2 DEMUX module is then used in cascading to design 1-to-4 DEMUX as given in Fig. 2.

- The waveforms were plotted for each part.

- Fig. 3. shows that clock module circuit generates clock signal `OUT` with time period of 40 ns. It can be observed that clock is in ‘on’-state for about 10 ns for a given time period, hence having a duty cycle of 25%.

- Fig. 4. Shows that in 1-to-4 DEMUX module, the input `IN` to module is set to ‘1’. It can be observed that as the value of select pin `S[1:0]` is varied, input is being transmitted to the corresponding output terminal. (`S[1:0]=00` for `O0`, `01` for `O1`, `10` for `O2`, `11` for `O3`)