Lab No. 13

Behavioral Modeling of Sequential Circuits in Verilog

Objective:

In this lab students will learn how to implement

- To understand the concept of **behavioral modeling** using `always` blocks in Verilog.
- To learn how to design and simulate **D-Latch**, **D Flip-Flop**, **JK Flip-Flop**, and **T Flip-Flop** using behavioral style.
- To analyze the difference between **latches and flip-flops** and their timing behavior.
- To understand edge-sensitive and level-sensitive triggering in sequential circuits.
- To verify the functional behavior of sequential circuits through waveform simulations.

Theory

Behavioral modeling in Verilog describes a circuit in terms of **how it behaves**, rather than how it is physically connected. Instead of gate-level connections, behavioral modeling uses **procedural statements**, mainly inside always blocks, to define the logical operation of sequential circuits. This makes the code simpler, readable, and closer to high-level algorithms.

Sequential circuits such as latches and flip-flops store information and change states based on a control signal (Enable or Clock).

- A **D-Latch** is **level-sensitive**, meaning the output follows the input D only when Enable = 1.
- **Flip-flops** are **edge-triggered** devices. They change state only on the positive or negative edge of the clock signal.
- A **D Flip-Flop** transfers the value of D to Q on the active clock edge.
- A **JK Flip-Flop** allows toggle, reset, and set operations depending on the values of J and K.
- A **T Flip-Flop** toggles its state when T = 1 and holds its state when T = 0.

Using behavioral modeling, these storage elements are described with constructs like if, case, posedge, negedge, and non-blocking assignments (<=). This approach closely follows the truth tables and timing diagrams of the sequential devices, making it ideal for both learning and digital design implementation.

Date: _____ Roll Number_____
Total Marks: 20

Getting Started with an Example

Run the following code and observe its output

```
module DLatch(input Enable, D,output Q);
reg Q;
always @ (Enable or D)
if (Enable) Q <= D;
endmodule
```

Testbench Code

```
timescale 1ns/1ps

module D_Test;

   // Inputs
   reg Enable, D;

   // Output
   wire Q;

   // Instantiate the Unit Under Test (UUT)
   DLatch uut (
      .Enable(Enable),
      .D(D),
      .Q(Q)
   );

   initial begin
      $monitor("Time=%0t  Enable=%b  D=%b  |  Q=%b",
            $time, Enable, D, Q);

      // Initial values
      Enable = 0; D = 0; #10;

      // Test 1: Latch disabled (Q should not change)
      D = 1; #10;
      D = 0; #10;

      // Test 2: Latch enabled (Q follows D)
      Enable = 1; D = 1; #10;
      D = 0; #10;

      // Test 3: Disable again (Q should hold last value)
      Enable = 0; D = 1; #10;
      D = 0; #10;

      $finish;
   end

endmodule
```

Lab Exercise

Digital Logic Design Lab | Lab Manual 13
Date: _____ | Roll Number_____
Total Marks: 20

Write down behavioral level description of D Flip Flop in Verilog and generate its Testbench for verifying its operation. [5]

Now, using the same D_FF module, perform the conversion of D into JK Flip Flop using instantiation. [5]

Similarly, using the same D_FF module, perform the conversion of D into T Flip Flop using instantiation. [5]

What is the role of always statement in the behavioral modeling? [2]

Date: _____ Roll Number_____
Total Marks: 20

Conclusion [3]