ECS 409/609 : Verilog/VHDL Sequential Logic Assignment

Instructor: Dr. Sukarn Agarwal,
Assistant Professor,
Electrical Engineering and Computer Science Department,
IISER Bhopal

September 1, 2025

Sequential Logic in Verilog

- Sequential Logic state transition is triggered by the "Clock" signal.
- Latches are sensitive to the level of signal.
- Flip Flop is sensitive to the transition of signal.

New Verilog Construct for Sequential Logic

To model a sequential logic, two new programming constructs are required:

```
1. always
2. posedge/negedage
```

The always block

The always block basic syntax are as follows:

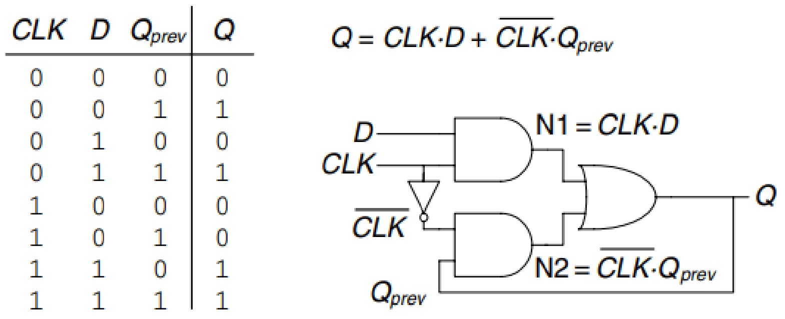
```
always @ (sensivity list)
statement;
```

whenever the event in the `sensitivity list` occurs, the `statement` is executed.

Sample Code Structure of D flip flop

```
module flop (input clk,
 input [3:0] d,
 output reg [3:0] q);
always @ (posedge clk)
q <= d;
endmodule
```

The detailed description of the above D flip flop module are as follows:



CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$

Figure 1: Improved Version of D Latch

- `posedge` defines a rising edge (transition from 0 to 1).
- Statement executed when the clock signal rises (i.e., on positive edge (`posedge`) of the clock (`clk`))
- Once the `clk` signal rises; the value of `d` is assigned to `q`.
- Note that `assign` statement is not used within an `always` block.
- Assigned variables need to be declared as `reg`.
- The name `reg` does not necessarily mean the value is a register.

With that sample coding D flip flop module in mind, implement the following Verilog programs using sequential logic construct.

Problem Statements

Level-Easy

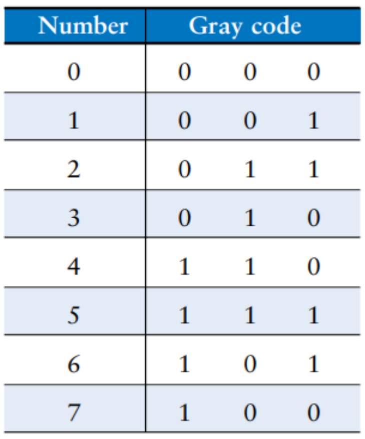
Problem 1: Implement a verilog module of an SR latch with asynchronous enable and reset.
Problem 2: Implement the Verilog module of an improved version of D latch as shown in figure 1. Specify delays of 1 ns to each gate. With your simulator, show that the latch operates correctly.

Level-Medium

Problem 3: Design a verilog module of counter that counts from 7 to 0 (e.g., 7, 6, 5, 4, 3, 2, 1, 0).
Problem 4: Implement a verilog module of UP/DOWN modulo 8 Gray Code Counter as shown in table 2, adding an Input `UP`. If `UP = 1`, the counter advances sequentially to the next number. Otherwise, `UP = 0`, the counter stays with the old value.

Level-Difficult

Problem 5: Implement a verilog module of N-bit bidirectional shift registers using D flip flop. (Hint: Use Parameterized Module to implement N).



Number	Gray code		
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Figure 2: Gray Code Counter Table

Problem 6: Implement a single port memory of address space 128 and the addressability of 16-bit in Verilog. The memory has four inputs: clock, write enable, write address register, and read address register, and two outputs: read and write data register.

Submission Format and Deadline: Submit all your source code, test bench code, and output/waveform (if applicable) in a pdf format file by the end of the day of 14-Sep-2025 (IST) at Google Classroom. Please make sure your name should appear on the gtkoutput waveform/terminal screen. Further, any copy case between the assignment(s) results in a zero mark. Note that TA may run the Plagiarism checker to check if the code is not copied from any other online source(s), if the tool found more than a 30% match percentage, it results in a zero mark.