ECS 409/609 : Verilog/VHDL
Finite State Machine Assignment

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

September 2, 2025

Finite State Machine in Verilog

Each FSM consist of three parts, as shown in Figure 1:

- Next State Logic
- State Register
- Output Logic

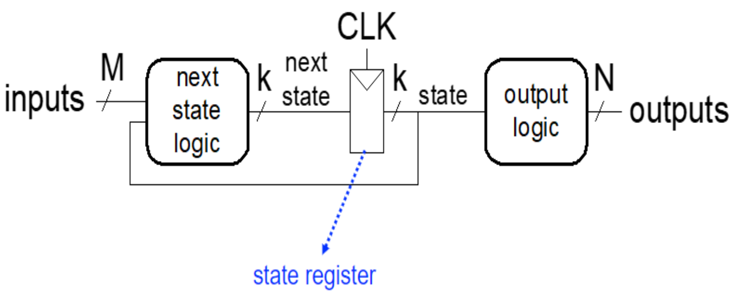

Figure 1: FSM Schematic

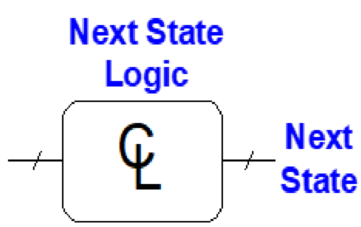

Figure 2: Next State Logic

Combinational Circuits

Next State Logic: Determine what the next state wll be, as shown in Figure 2.

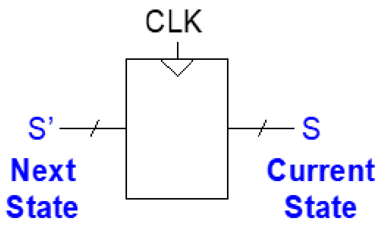

Figure 3: State Register

Sequential Circuits

State Register: Store the current state and load the next state at the clock edge, as shown in Figure 3.

Output Logic

Store the output, as shown in Figure 4.

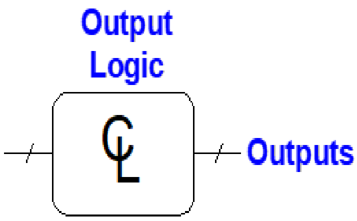

Figure 4: Output Logic

With these component in mind, implement the following FSM based Verilog programs.

Problem Statements

Level-Easy

Problem 1: Implement a Verilog based Moore machine as shown in Figure 5.
Problem 2: Implement the Verilog based Mealy machine as given in Figure 6.

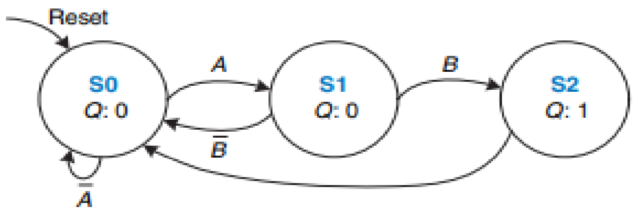

Figure 5: Moore Machine Diagram For Problem 1

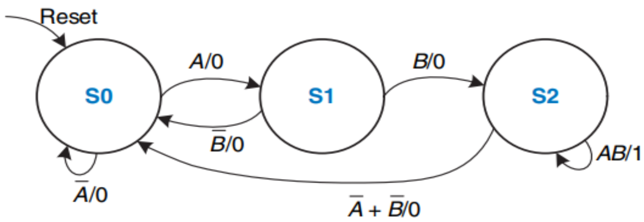

Figure 6: Mealy Machine Diagram For Problem 2

Level-Medium

Problem 3: Implement an FSM in verilog that detects the input sequence `1101` on input variable x. The Verilog code should detect the desired input sequence every time it occurs, even if embedded in a sequence of bits. When the Verilog code detects the desired input sequence and output, Z should be 1; otherwise, Z should be zero. On resetting the state machine, your verilog code should return to the initial state S0.

Level-Difficult

Problem 4: You are given two one-bit input signals (P_A and P_B) and one-bit output signal (O) for the following modular equation `2N(P_{A}) + N(P_{B}) = 1 (mod 4)`. In this modular equation, `N(P_{A})` and `N(P_{B})` represent the total number of times the inputs `P_{A}` and `P_{B}` are high (i.e., logic 1) at each positive clock edge, respectively. The one-bit output signal, `O`, is set to 1 when the modular equation is satisfied (i.e., `2N(P_{A}) + N(P_{B}) = 1 (mod 4)`), and 0 otherwise. An example that sets `O = 1` at the end of third cycle would be:

- (1_{st} cycle) `P_{A} = 0` (`N(P_A) = 0`), `P_{B} = 0` (`N(P_B) = 0`), `2N(P_{A}) + N(P_{B}) = 0 (mod 4)` $\rightarrow$ `O = 0`
- (2_{nd} cycle) `P_{A} = 1` (`N(P_A) = 1`), `P_{B} = 1` (`N(P_B) = 1`), `2N(P_{A}) + N(P_{B}) = 3 (mod 4)` $\rightarrow$ `O = 0`
- (3_{rd} cycle) `P_{A} = 1` (`N(P_A) = 2`), `P_{B} = 0` (`N(P_B) = 1`), `2N(P_{A}) + N(P_{B}) = 5 (mod 4)` $\rightarrow$ `O = 1`
- (4_{th} cycle) `P_{A} = 0` (`N(P_A) = 2`), `P_{B} = 1` (`N(P_B) = 2`), `2N(P_{A}) + N(P_{B}) = 6 (mod 4)` $\rightarrow$ `O = 0`

Considering this state diagram implementation, write the Verilog code based on the above description.

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 18-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.