University Of Engineering and Technology, Lahore
Computer Engineering Department

Course Name: Computer Architecture	**Course Code:** CMPE-421L
Assignment Type: Lab	**Dated:** 18th September 2023
Semester: 7th	**Session:** 2020
Lab/Project/Assignment #: 3	**CLOs to be covered: CLO 1**
Lab Title: Design& Verification of FSMs in System Verilog	**Teacher Name:** Engr. Afeef Obaid

Lab Evaluation

CLO1	Understand the design and simulation of basic digital circuits with HDL and HDL simulation tools.					
Levels (Marks)	**Level1**	**Level2**	**Level3**	**Level4**	**Level5**	**Level6**
(10)						
					Total	**/10**

Rubrics for Current Lab Evaluation

Scale	**Marks**	**Level**	**Rubric**
Excellent	**9-10**	L1	Submitted all lab tasks, BONUS task, have good understanding.
Very Good	**7-8**	L2	Submitted the lab tasks but have good understanding
Good	**5-6**	L3	Submitted the lab tasks but have weak understanding.
Basic	**3-4**	L4	Submitted the lab tasks but have no understanding.
Barely Acceptable	**1-2**	L5	Submitted only one lab task.
Not Acceptable	**0**	L6	Did not attempt

Lab # 3

Lab Goals

By reading this manual, students will be able to:

- Understand the basics of finite state machines (FSMs).
- Create SV modules to implement finite state machines.
- Simulate and verify the behaviors of finite state machines.

Equipment Required

- Computer system with ModelSim or Xcelium, installed on it.

Finite State Machines

Introduction

Synchronous sequential circuits can be drawn in the forms shown in Figure 1. These forms are called finite state machines (FSMs). They get their name because a circuit with k registers can be in one of a finite number (2^k) of unique states. An FSM has M inputs, N outputs, and k bits of state. It also receives a clock and, optionally, a reset signal. An FSM consists of two blocks of combinational logic, next state logic and output logic, and a register that stores the state. On each clock edge, the FSM advances to the next state, which was computed based on the current state and inputs. There are two general classes of finite state machines, characterized by their functional specifications. In Moore machines, the outputs depend only on the current state of the machine. In Mealy machines, the outputs depend on both the current state and the current inputs. Finite state machines provide a systematic way to design synchronous sequential circuits given a functional specification. (Harris & Harris)

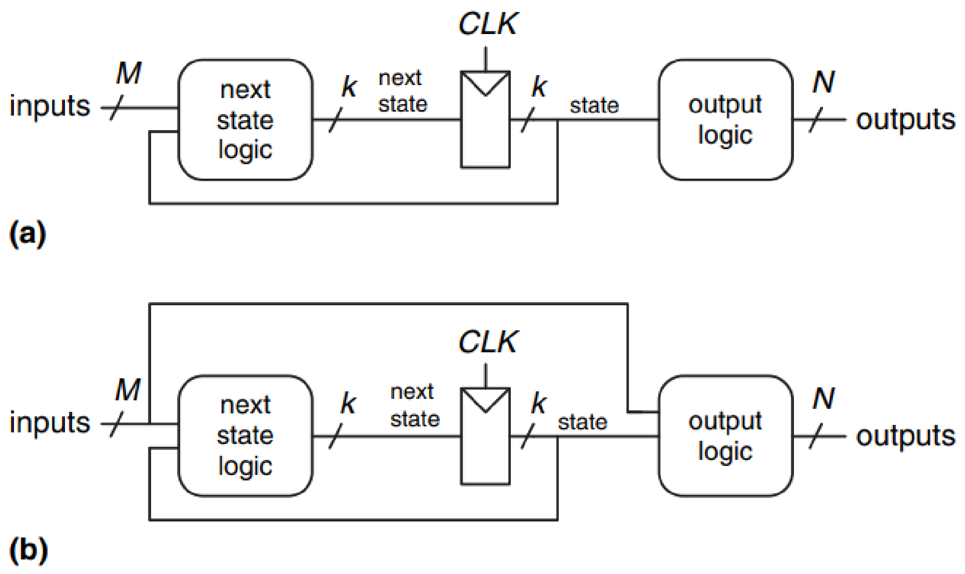

Figure 1: Finite state machines: (a) Moore machine, (b) Mealy machine

FSM Design Example

We will be consulting **section 3.4** of the textbook for a detailed FSM example. You **are advised to read up to section 3.4.2 of the textbook before coming to the lab**.

Steps to Design an FSM

Finite state machines are a powerful way to systematically design sequential circuits from a written specification.

Use the following procedure to design an FSM. .

- Identify the inputs and outputs.
- Sketch a state transition diagram.
- For a Moore machine,
    - Write a state transition table.
    - Write an output table.
- For a Mealy machine,
    - Write a combined state transition and output table.
- Select state encodings—your selection affects the hardware design.
- Write Boolean equations for the next state and output logic.
- Sketch the circuit schematic.

Translating FSMs to SV

As we have seen in the previous labs there is a beautiful correspondence between hand drawn diagrams and SV modules. Same is the case for finite state machines. Referring to figure 1 we can see that there are three distinct blocks in an FSM.

- Next stage logic – Combinational logic
- Register – Bank of flip-flops (Sequential logic)
- Output logic – Combinational logic

Describing these blocks in SV will be enough for designing our finite state machine.

FSM Design Code

```
module fsm
(
    input  logic clk,
    input  logic rst,
    input  logic inputs,
    output logic outputs
);

    always_comb
    begin
        // next state calculation logic
    end

    always_ff @(posedge clk)
    begin
        // state register
    end

    always_comb
    begin
        // ouput calculation logic
    end

endmodule
```

These three **always blocks** are enough to create behavior of a finite state machine. We just need to add next state logic, output calculation logic and settings for state register to model any state machine in SV.

Laborartory Task *(see example # 3.7 for hints)*

Convert the Moore state machine designed during the lab session to Mealy state machine. In Mealy state machine, the output combinationaly depends on the inputs so we can have the output one cycle earlier than the Moore state machine.

- Create state transition graph and state transition table for the Mealy machine.
- Update the SV design and testbench to comply with the new design requirements.
- Simulate and verify the behavior of Mealy machine.

Review Questions

Review Question 1

What is the difference between Mealy and Moore state machine?

Review Question 2

Which blocks in an FSM are combinational?

Review Question 3

What is a state transition graph and state transition table?

Note

There will be a complex engineering problem for this week, details of which will be shared during the lab.

References

Harris, D. M., & Harris, S. L. (n.d.). *Digital Design and Computer Architecture.*