AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH

Faculty of Engineering

Laboratory Report Cover Sheet

Students must complete all details except the faculty use part.

Please submit all reports to your subject supervisor or the office of the concerned faculty.

Laboratory Title : Design and Verilog HDL Modeling of Finite State Machines (FSM)

Experiment Number: 9 Due Date: 28-04-24 Semester: Spring 23-24

Subject Code:0067 Subject Name: **Digital Logic And Circuits Lab** Section: R

Course Instructor: Md. Ashiquzzaman Degree Program: CSE

Declaration and Statement of Authorship:

1. I/we hold a copy of this report, which can be produced if the original is lost/ damaged.
2. This report is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this report has been written for me/us by any other person except where such collaboration has been authorized by the lecturer/teacher concerned and is clearly acknowledged in the report.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the School for review and comparison, including review by external examiners.

Group Number:03

No.	Student Name	ID	Date
Submitted by:			
1	Tutul Majumder	23-51364-1	28-04-24
Group Members:			
2	Takbir Zaman Bhuiyan	22-49857-3	28-04-24
3	Md. Showkat Islam Sakib	22-49858-3	28-04-24
4	Kazi Imtiaz	22-49857-3	28-04-24

For faculty use only: **Total Marks:_______Marks Obtained: ________**

Faculty comments____

Title: Design and Verilog HDL Modeling of Finite State Machines (FSM)

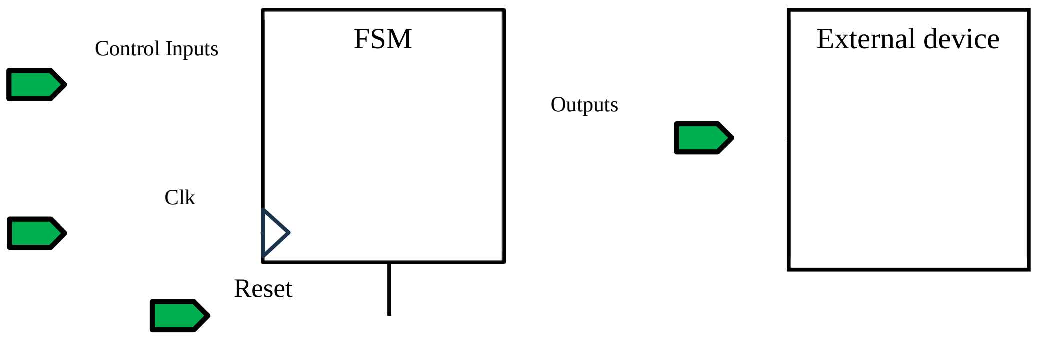
Introduction:

This lab is intended for students to acquire the skills to design a Finite State Machine (FSM) at gate-level for a simple sequence generator. Later students will apply Verilog HDL to model and simulate their FSM.

Theory and Methodology:

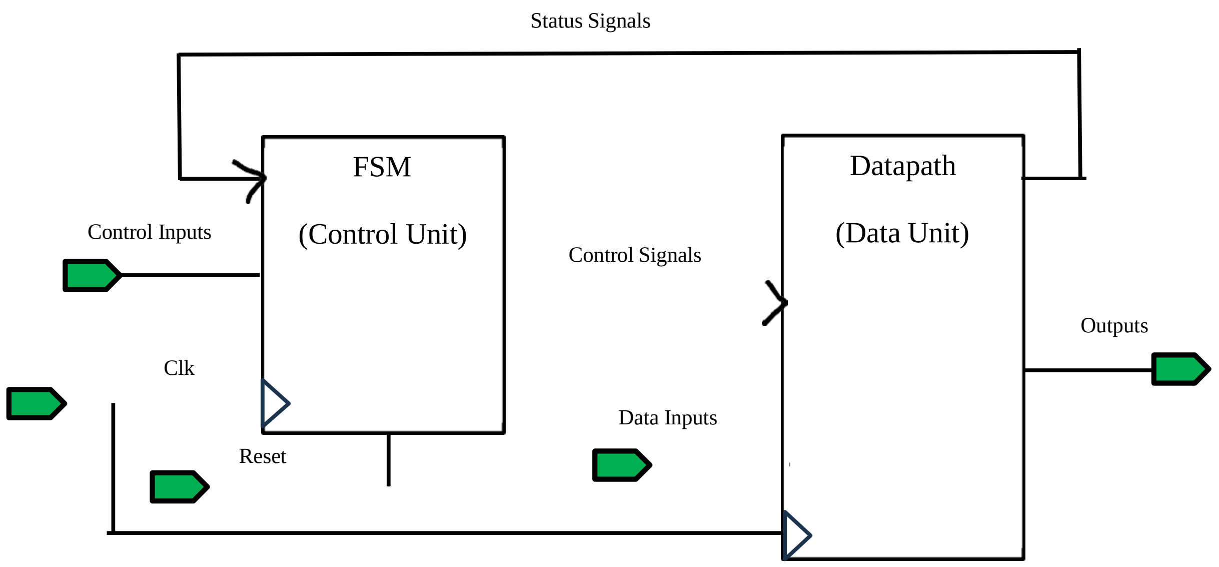
FSM is a computational model that goes through a predetermined sequence of operations in a finite number of states. The purpose of FSM is to automate computational tasks. FSMs also serve as the general model of sequential logic circuits. In digital electronic circuits, they are widely used to implement control units of digital systems.

In control-dominated designs, such as signal generators (e.g., traffic light controllers) and stepper motor controllers, FSMs control operation of an external device such as LEDs or a stepper motor.



In data-dominated designs, such as a microprocessor, FSMs control logical and arithmetic operations for the computations that the microprocessor needs to execute. In other words, it controls the operations of computing elements such as Arithmetic and Logic Units (ALU), storage devices for data such as registers and data steering circuits such as multiplexers and demultiplexers. FSM controls these operations through control signals (such as ALU_Control signals, select signals, load/enable signals for ALU, multiplexer and registers, respectively). To ensure proper control of these devices, sometimes the output states of these devices are read back by the FSM as status signals.

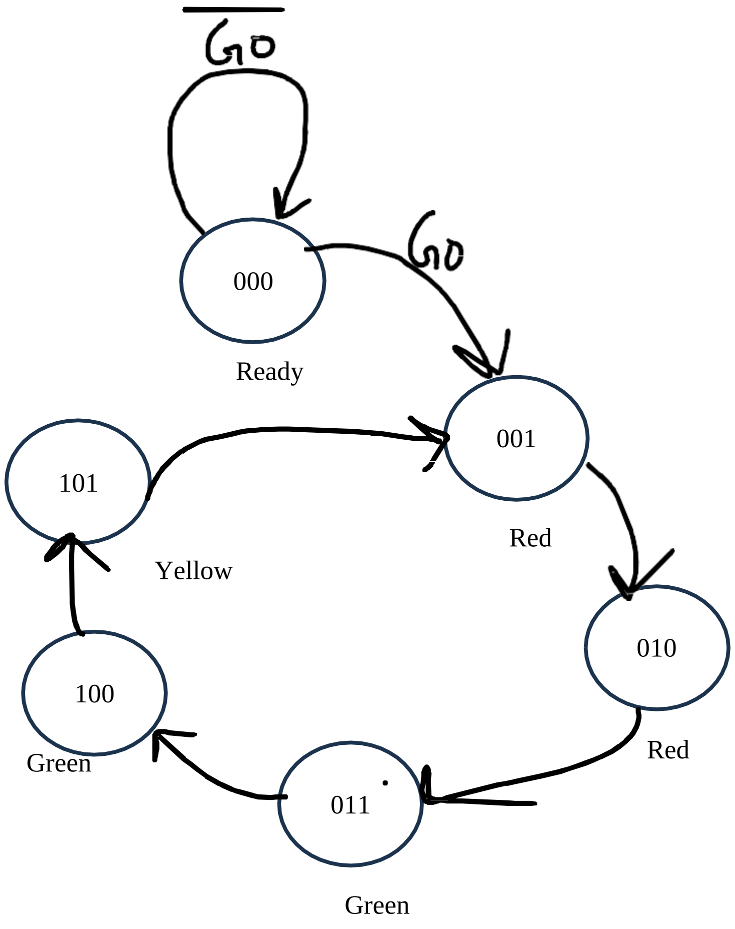
Note that the computations are done inside a datapath or data unit. The FSM serves as the control unit that guides the computations so that they are executed correctly and in the right sequence.



Example 1: A Simple Signal Generator

Prepare the design for an FSM that stays in an idle state where it generates an output called Ready and evaluates the state of an input called Go. If Go is low, it stays in the idle state. If Go is high, it generates threeoutputs—Red, Green and Yellow—consecutively (one after another) for 2s, 2s and 1s, respectively. The Red-Green-Yellow sequence is repeated continuously (FSM does not return to the idle (initial) state)). The clock frequency is 1 Hz. **Apply** binary state encoding.

The following state diagram models the algorithm for this device.



Pre-Lab Homework:

Completed the State Table for Example 1.

Apparatus:

1) Computer with Internet Access
2) Google account
3) Access to EDA Playground

Precautions:

Ensuring that my computer has active and updated antivirus software.

Experimental Procedure:

1. Logic equations for the next state signals and outputs had been developed.
2. The Verilog HDL code for the FSM had been written down within the provided template for the design module. Additionally, the complete testbench module had been provided.
3. The design module had been simulated.

Simulation and Results:

1. Show the state table.

Table 1: State Table

Reset	PState	Input	NState	Outputs			
		Go		Ready	Red	Green	Yellow
1	xxxx	x	000	0	0	0	0
0	000	0	000	1	0	0	0
		1	001				
0	001	x	010	0	1	0	0
0	010	x	011	0	1	0	0
0	011	x	100	0	0	1	0
0	100	x	101	0	0	1	0
0	101	x	001	0	0	0	1

2. Show the logic equations.

Equation for NState

000, 010, 100, 101

$NS0 = \overline{Reset}.(\overline{PS2}.\overline{PS1}.\overline{PS0}.\mathrm{Go} + \overline{\mathrm{PS2}}.\mathrm{PS1}.\overline{\mathrm{PS0}} + \mathrm{PS2}.\overline{\mathrm{PS1}}.\overline{\mathrm{PS0}} + \mathrm{PS2}.\overline{\mathrm{PS1}}.\mathrm{PS0})$

$= \overline{Reset}.(\overline{PS2}.\overline{PS1}.\overline{PS0}.\mathrm{Go} + \overline{\mathrm{PS2}}.\mathrm{PS1}.\overline{\mathrm{PS0}} + \mathrm{PS2}.\overline{\mathrm{PS1}}.(\overline{\mathrm{PS0}} +))$

$= \overline{Reset}.(\overline{PS2}.\overline{PS1}.\overline{PS0}.\mathrm{Go} + \overline{\mathrm{PS2}}.\mathrm{PS1}.\overline{\mathrm{PS0}} + \mathrm{PS2}.\overline{\mathrm{PS1}})$

$= \overline{Reset}.(\overline{PS2}.\overline{PS0}.\left(\overline{PS1}.\mathrm{Go} + \mathrm{PS1}\right) + \mathrm{PS2}.\overline{\mathrm{PS1}})$

001, 010

$NS1 = \overline{Reset}.(\overline{PS2}.\overline{PS1}.\mathrm{PS0} + \overline{\mathrm{PS2}}.\mathrm{PS1}.\overline{PS0})$

$= \overline{Reset}.(\overline{PS2}.(\overline{PS1}.\mathrm{PS0} + \mathrm{PS1}.\overline{PS0}))$

$= \overline{Reset}.\overline{PS2}.(\mathrm{PS0} \oplus \mathrm{PS1})$

011, 100

$NS2 = \overline{Reset}.(\overline{PS2}.\mathrm{PS1}.\mathrm{PS0} + \mathrm{PS2}.\overline{PS1}.\overline{PS0})$

Equation for Outputs

000

$\mathrm{Ready} = \overline{PS2}.\overline{PS1}.\overline{PS0}$

001, 010

$\mathrm{Red} = \overline{PS2}.\overline{PS1}.\mathrm{PS0} + \overline{PS2}.\mathrm{PS1}.\overline{PS0}$

$= \overline{PS2}.(\overline{PS1}.\mathrm{PS0} + \mathrm{PS1}.\overline{PS0})$

$= \overline{PS2}.(\mathrm{PS0} \oplus \mathrm{PS1})$

011, 100

$\mathrm{Green} = \overline{PS2}.\mathrm{PS1}.\mathrm{PS0} + \mathrm{PS2}.\overline{PS1}.\overline{PS0}$

101

$\mathrm{Yellow} = \mathrm{PS2}.\overline{\mathrm{PS1}}.\mathrm{PS0}$

Final Equations

$NS0 = \overline{Reset}.(\overline{PS2}.\overline{PS0}.\left(\overline{PS1}.\mathrm{Go} + \mathrm{PS1}\right) + \mathrm{PS2}.\overline{\mathrm{PS1}})$

$NS1 = \overline{Reset}.\overline{PS2}.(\mathrm{PS0} \oplus \mathrm{PS1})$

$NS2 = \overline{Reset}.(\overline{PS2}.\mathrm{PS1}.\mathrm{PS0} + \mathrm{PS2}.\overline{PS1}.\overline{PS0})$

$\mathrm{Ready} = \overline{PS2}.\overline{PS1}.\overline{PS0}$

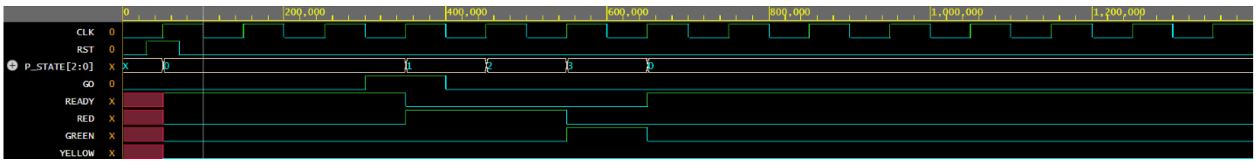
$\mathrm{Red} = \overline{PS2}.(\mathrm{PS0} \oplus \mathrm{PS1})$

$\mathrm{Green} = \overline{PS2}.\mathrm{PS1}.\mathrm{PS0} + \mathrm{PS2}.\overline{PS1}.\overline{PS0}$

$\mathrm{Yellow} = \mathrm{PS2}.\overline{\mathrm{PS1}}.\mathrm{PS0}$

3. Show the Functional Simulation from EDA Playground.

<u>Functional Simulation:</u>



Questions for report writing:

What benefits one would gain if gray or one-hot state encoding were used for the FSM?

Ans: If gray or one-hot state encoding were used for the FSM, one would gain several benefits.

One-Hot Encoding:

- **Simpler Combinational Logic:** One-hot encoding uses a flip-flop for each state, with only one flip-flop being high at a time. This simplifies the logic needed to determine the next state and outputs, making the design easier to understand and implement.
- **Faster Operation:** Due to the simpler logic, one-hot encoding can sometimes lead to faster operation of the FSM, especially in FPGAs where registers are plentiful.
- **Easier Error Detection:** Since only one bit is high in a one-hot encoded state, any errors with multiple bits flipped will be easily detectable.
- On the other hand, gray encoding, assigns state codes such that consecutive state codes differ by only one bit. This encoding technique has a Hamming distance of 1, which is always the minimum possible. This means that the number of bits that need to be changed to transition from one state to another is minimized, reducing the power consumption of the FSM.

Gray Encoding:

- **Reduced Glitches:** Gray encoding ensures that only one bit flips between adjacent states. This minimizes glitches in asynchronous circuits or circuits with short clock cycles, improving reliability.
- **Power Efficiency:** For sequential state transitions, Gray encoding can reduce dynamic power consumption due to minimal bit changes.
- **Useful in Certain Applications:** Gray encoding is particularly beneficial for applications like asynchronous FIFOs (First-In-First-Out) where glitch immunity is crucial.

Discussion:

By the time the lab on designing and Verilog HDL modeling of Finite State Machines (FSM) began, the main objective had already been to understand and implement the fundamental concepts of FSMs, such as states, transitions, and outputs, using Verilog HDL. Initially, the basics of Finite State Machines (FSMs) and their design had been discussed. The timing diagram had also been demonstrated using Verilog HDL code. For simulation, EDA Playground, an online tool for Verilog simulation, had been utilized. No errors had been found in the output.

Conclusion:

In conclusion, Finite State Machines (FSMs) play a fundamental role in automating computational tasks and controlling digital systems. Whether in control-dominated designs like signal generators or data-dominated designs like microprocessors, FSMs provide the necessary sequencing and control to ensure correct and efficient operation. By guiding operations within a datapath or data unit, FSMs serve as crucial control units, orchestrating logical and arithmetic operations as well as managing external devices. This versatility makes FSMs indispensable in various fields, from electronics to computing, where precise control and automation are essential.

Reference:

[1] Thomas L. Floyd, *Digital Fundamentals*, 9th Edition, 2006, Prentice Hall.

[2] Michael Ciletti, Advanced Digital Design with the Verilog HDL-2nd Edition, 2010.

[3] Doughlas J. Smith, HDL Chip Design-A Practical Guide for Designing, Synthesizing and Simulating ASICs & FPGAs using VHDL or Verilog, 1997.

[4] American International University–Bangladesh (AIUB) Digital Logic And Circuits Lab Manual.