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

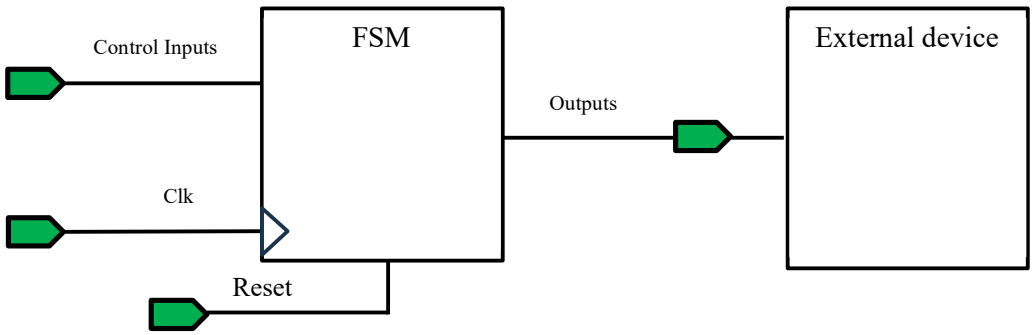
Introduction:

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:

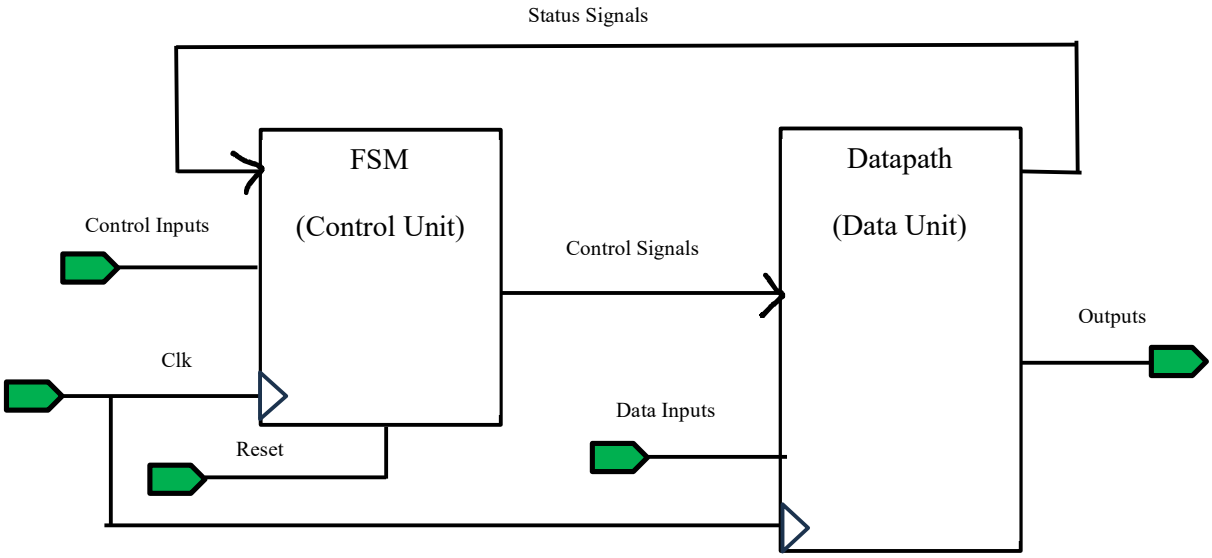
FSM is a computational model that goes through a predetermined sequence of operations in a finite number of states. The purpose of FSM is to automate computational tasks. FSMs also serve as the general model of sequential logic circuits. In digital electronic circuits, they are widely used to implement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

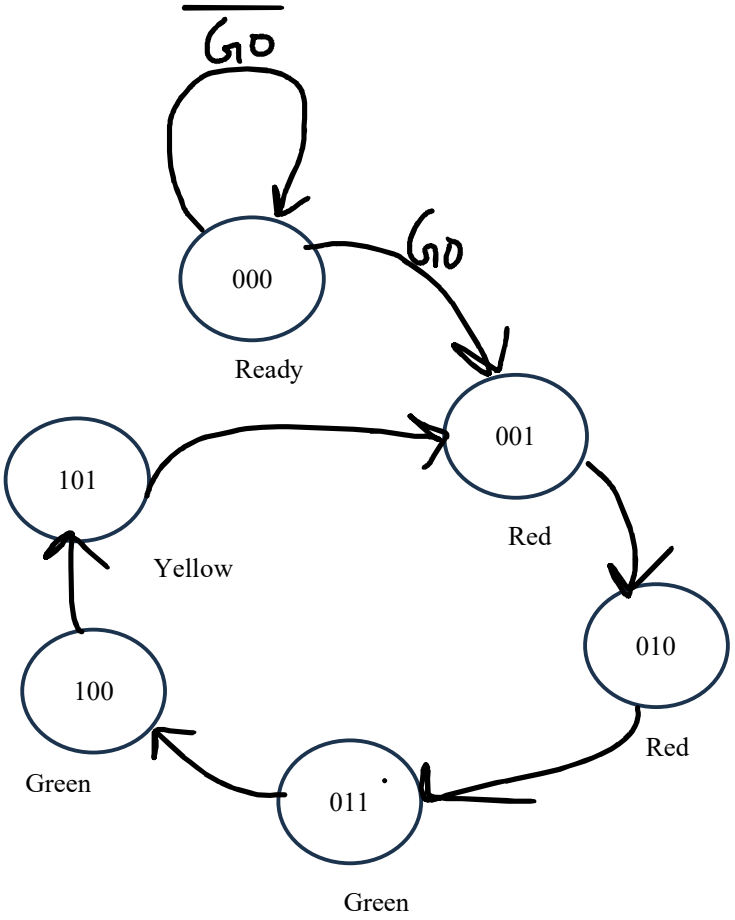
Note that the computations are done inside a datapath or data unit. The FSM serves as the control unit that guides the computations so that they are executed correctly and in the right sequence.



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 three outputs—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:

Complete the State Table for Example 1.

Apparatus:

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

Precautions:

Make sure your computer has an active and updated anti-virus.

Experimental Procedure:

1) Develop the logic equations for the next state signals and outputs.
2) Write down the Verilog HDL code for the FSM. The template for the design *module* will be provided. The complete testbench *module* will also be provided.
3) Simulate the design *module.*

Simulation and Results:

1. Show the state table.
2. Show the logic equations.
3. Show the Functional Simulation from EDA Playground.

Questions for report writing:

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

Discussion and Conclusion:

Interpret the data/findings and determine the extent to which the experiment was successful in complying with the goal that was initially set. Discuss any mistake you might have made while conducting the investigation and describe ways the study could have been improved.

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.