|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AIUB | **American International University- Bangladesh (AIUB)**  **Faculty of Engineering (FE)** | | | | | | | | | |
|  | |  | | | | |  | |  | |
| **Course Name :** | | Digital Logic & Circuits Laboratory | | | | | **Course Code :** | | EEE 1204 | |
| **Semester :** | | Fall 2024-25 | | | | | **Sec :** | | G | |
| **Lab Instructor :** | | MD. ALOMGIR KABIR | | | | | **Group :** | | 07 | |
|  | |  | | | | |  | |  | |
| **Experiment No :** | | 09 | | | | | | | | |
| **Experiment Name :** | | Design and Verilog HDL Modeling of Finite State Machines (FSM) | | | | | | | | |
|  | |  | | | | |  | |  | |
| **Submitted by (NAME):** | | **Chinmoy Guha** | | | | **Student ID:** | | | **22-48056-2** | |
|  | |  | | | |  | | |  | |
| **Group Members** | | | | | **ID** | | | **Name** | | |
|  | | | | 1. | 22-48056-2 | | | CHINMOY GUHA | | |
|  | | | | 2. | 22-48067-2 | | | SUVRA CHAKRABORTY | | |
|  | | | | 3. | 22-47975-2 | | | MOHAMMAD ANSAR UDDIN | | |
|  | | | | 4. | 22-49393-3 | | | EFAZ MASHRUR AMIN | | |
|  | | | | 5. | 22-49385-3 | | | NAZMUL HASSAN NAYEEM | | |
|  | | | | 6. |  | | |  | | |
|  | | | | 7. |  | | |  | | |
|  | | | | |  | |  | |  | |
| **Performance Date :** | | | **04/01/25** | | | **Due Date :** | | | | **15/01/25** |
|  | | | | |  | | | | | |

**Marking Rubrics (to be filled by Lab Instructor)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Proficient  [6] | Good  [4] | Acceptable  [2] | Unacceptable [1] | Secured Marks |
| **Theoretical Background, Methods  & procedures sections** | All information, measures and variables are provided and explained. | All Information provided that is sufficient, but more explanation is needed. | Most information correct, but some information may be missing or inaccurate. | Much information missing and/or inaccurate. |  |
| **Results** | All of the criteria are met; results are described clearly and accurately; | Most criteria are met, but there may be some lack of clarity and/or incorrect information. | Experimental results don’t match exactly with the theoretical values and/or analysis is unclear. | Experimental results are missing or incorrect; |  |
| **Discussion** | Demonstrates thorough and sophisticated understanding. Conclusions drawn are appropriate for analyses; | Hypotheses are clearly stated, but some concluding statements not supported by data or data not well integrated. | Some hypotheses missing or misstated; conclusions not supported by data. | Conclusions don’t match hypotheses, not supported by data; no integration of data from different sources. |  |
| **General formatting** | Title page, placement of figures and figure captions, and other formatting issues all correct. | Minor errors in formatting. | Major errors and/or missing information. | Not proper style in text. |  |
| **Writing & organization** | Writing is strong and easy to understand; ideas are fully elaborated and connected; effective transitions between sentences; no typographic, spelling, or grammatical errors. | Writing is clear and easy to understand; ideas are connected; effective transitions between sentences; minor typographic, spelling, or grammatical errors. | Most of the required criteria are met, but some lack of clarity, typographic, spelling, or grammatical errors are present. | Very unclear, many errors. |  |
| Comments: |  | | | Total Marks  (Out of ): |  |

**Abstract:**

This paper presents a comprehensive overview of Finite State Machines (FSMs), covering design methodologies and Verilog HDL modeling techniques. It explores state transition diagrams, state encoding methods, and FSM implementation in Verilog, emphasizing efficient synthesis and simulation for digital circuit design. In Verilog HDL, a Finite State Machine (FSM) is a sequential logic design model used to describe systems that transition between a finite number of states based on inputs and a clock signal. An FSM typically consists of:

1. **State Encoding**: Defining and encoding the possible states.
2. **State Transition Logic**: Describing the conditions under which the system transitions from one state to another.
3. **Output Logic**: Determining the outputs based on the current state and optionally the inputs.

FSMs are commonly categorized into two types:

* **Moore FSM**: Outputs depend only on the current state.
* **Mealy FSM**: Outputs depend on both the current state and the inputs.

Verilog is used to implement FSMs by defining registers for states and using procedural blocks (e.g., always blocks) for state transitions and output logic.

**Objectives:**

 Develop the skills to design a Finite State Machine (FSM) at the gate level for a simple sequence generator.

 Utilize Verilog HDL to model, simulate, and verify the functionality of the designed FSM.

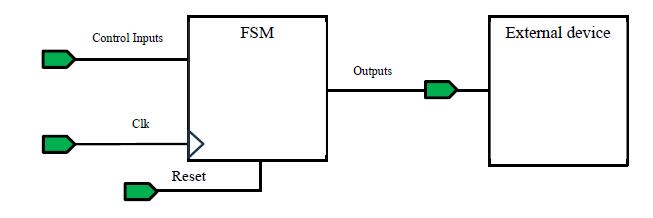
 Design and implement an FSM that operates as follows:

* Remains in an idle state generating a "Ready" output until the input signal "Go" is high.
* Upon detecting a high "Go" signal, transitions through states that produce the outputs "Red," "Green," and "Yellow" in sequence for 2 seconds, 2 seconds, and 1 second, respectively.
* Repeats the Red-Green-Yellow sequence continuously without returning to the idle state.
* To bridge the gap between theoretical knowledge of FSMs and practical Verilog HDL implementation, provide cohesive resource for students and practitioners in the field of digital circuit design.

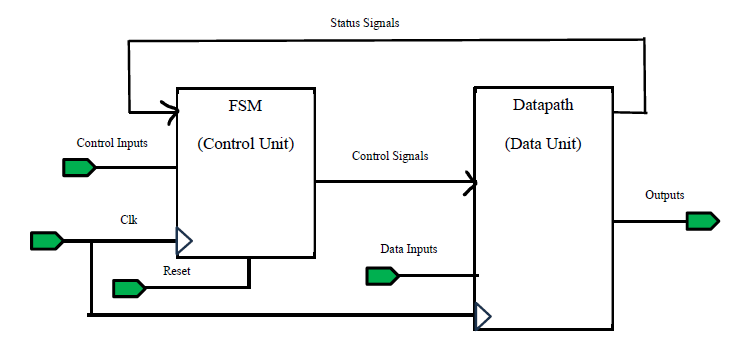
 Apply binary state encoding for the FSM design and use a 1 Hz clock frequency for timing the state transitions.

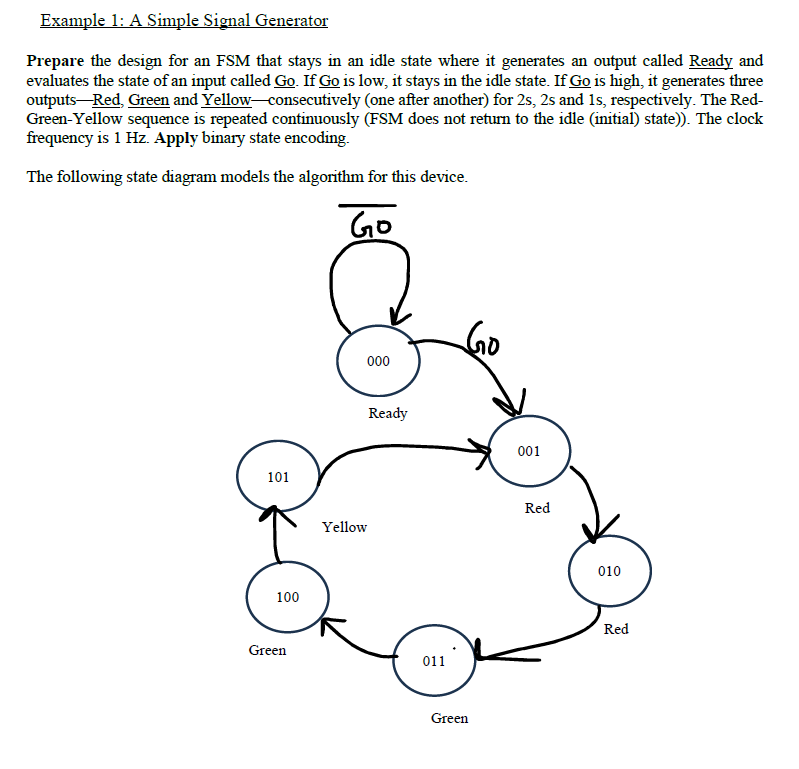
**Theory and Methodology:**

A Finite State Machine (FSM) is a computational model that transitions through a defined sequence of states to perform automated tasks. It serves as a general model for sequential logic circuits and is commonly used in digital systems to implement control units. FSMs are integral to control-dominated designs, such as signal generators (e.g., traffic light controllers) and stepper motor controllers, where they manage the operation of external devices like LEDs or motors.



In data-dominated designs, for example - microprocessors, FSMs control logical and arithmetic operations by managing computing elements like Arithmetic and Logic Units (ALUs), registers, and data-steering circuits (e.g., multiplexers). They generate control signals (e.g., ALU\_Control, select, load/enable) to coordinate these components and may read status signals from device outputs to ensure proper operation. The FSM acts as the control unit, guiding computations within the datapath to ensure they are executed accurately and in the correct sequence.





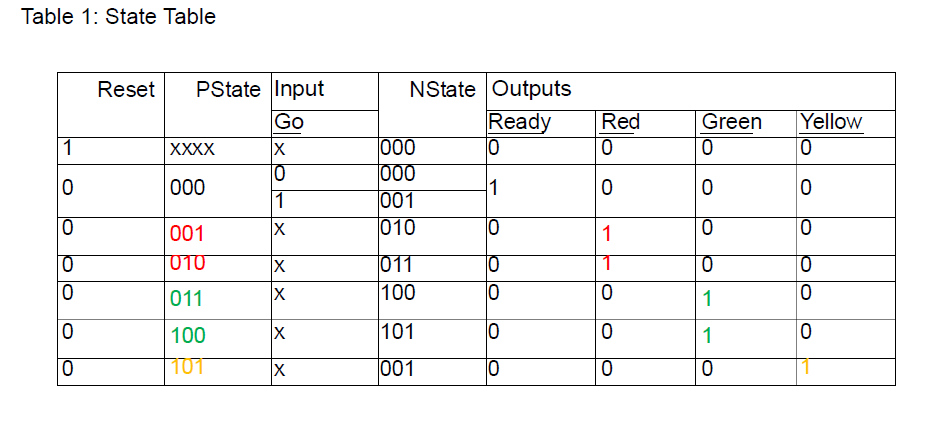
**Apparatus used:**

1) Computer with Internet Access

2) Google account

3) Access to EDA Playground

**Simulations:**



Equations for NSTATE:

Equations for Outputs:

Verilog Code:

// Code your design here

module TLC\_LAB9\_V1

(input wire CLK, RST, GO,

output wire READY, RED, GREEN, YELLOW);

reg [2:0] P\_STATE;

wire [2:0] N\_STATE;

// NSTATE equations

assign N\_STATE[0] = ~RST & ( ~P\_STATE[2] & ~P\_STATE[1] & ~P\_STATE[0] & GO

| ~P\_STATE[2] & P\_STATE[1] & ~P\_STATE[0]

| P\_STATE[2] & ~P\_STATE[1] & ~P\_STATE[0]

| P\_STATE[2] & ~P\_STATE[1] & P\_STATE[0]);

assign N\_STATE[1] = ~RST & (~P\_STATE[2] & ~P\_STATE[1] & P\_STATE[0]

| ~P\_STATE[2] & P\_STATE[1] & ~P\_STATE[0]);

assign N\_STATE[2] = ~RST & (~P\_STATE[2] & P\_STATE[1] & P\_STATE[0]

| P\_STATE[2] & ~P\_STATE[1] & ~P\_STATE[0]);

// Sequential logic to update P\_STATE

always @(posedge CLK) begin

if (RST)

P\_STATE <= 3'b000;

else

P\_STATE <= N\_STATE;

end

// Output equations

assign READY = ~P\_STATE[2] & ~P\_STATE[1] & ~P\_STATE[0];

assign RED = ~P\_STATE[2] & ~P\_STATE[1] & P\_STATE[0]

| P\_STATE[2] & ~P\_STATE[1] & P\_STATE[0];

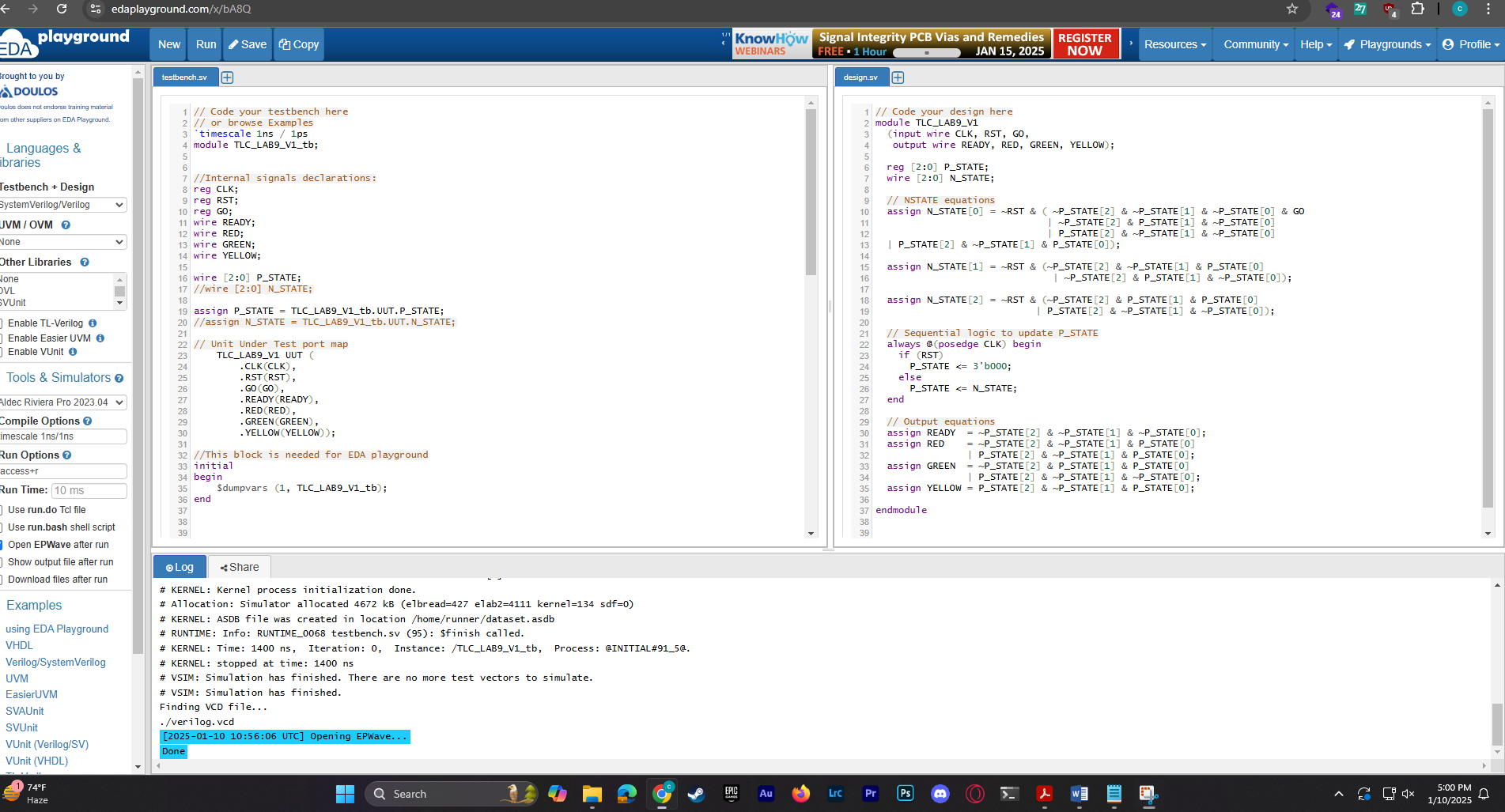
assign GREEN = ~P\_STATE[2] & P\_STATE[1] & P\_STATE[0]

| P\_STATE[2] & ~P\_STATE[1] & ~P\_STATE[0];

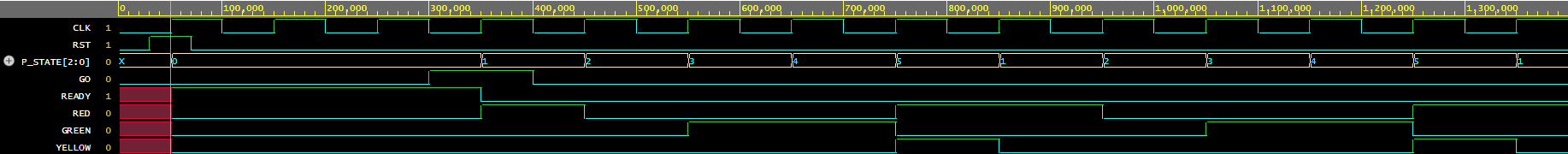
assign YELLOW = P\_STATE[2] & ~P\_STATE[1] & P\_STATE[0];

endmodule

EDA PLAYGROUND:



EDA PLAYGROUND FUNCTIONAL SIMULATION:



**Results and Discussions:**

At the start of the lab on Finite State Machines (FSMs), the focus was on understanding and implementing key concepts like states, transitions, and outputs using Verilog HDL. The basics of FSM design were reviewed, and a timing diagram was demonstrated with Verilog HDL. Simulations were performed using EDA Playground, which produced error-free results. In summary, FSMs are essential for automating tasks and controlling digital systems. They ensure proper sequencing and control in both control-focused designs, like signal generators, and data-heavy designs, such as microprocessors. By coordinating operations within datapaths and managing external devices, FSMs are vital for achieving precise control and automation in electronics and computing.

**References:**

1. Thomas L. Floyd, *Digital Fundamentals*, 9th Edition, 2006, Prentice Hall.
2. Boylestad, Robert L., *Electronic Devices and Circuit Theory*, Pearson Education, 2009.
3. Michael Ciletti, Advanced Digital Design with the Verilog HDL-2nd Edition, 2010. \
4. Doughlas J. Smith, HDL Chip Design-A Practical Guide for Designing, Synthesizing and Simulating ASICs & FPGAs using VHDL or Verilog, 1997.