VLSI Project

Sequence detector

Using Mealy State Machine

Presented by

Naga Mohitha Kummari

Rajiv Gandhi University of Knowledge technologies

Mealy State Machine

The objective of using a Mealy state machine is to design a system or circuit that can efficiently and accurately respond to inputs by producing corresponding outputs based on both the current state and the inputs. **Some specific objectives and applications of Mealy state machines include:**

- Sequential Logic Design: Mealy state machines are widely used in sequential logic design to model and implement complex behavior that depends on the current state and the inputs. They are utilized to design and control the sequencing of events in digital systems, such as controllers, processors, and data processing units.

- Protocol Implementation: Mealy state machines are often employed in the implementation of communication protocols. They can be used to define the behavior and processing of protocol-specific data packets or messages. Mealy machines are utilized to handle the interpretation, validation, and generation of protocol-specific information based on the current state and the inputs received.

- Error Detection and Correction: Mealy state machines are valuable in error detection and correction applications. They can analyze input patterns or data streams to identify errors or discrepancies and generate appropriate error detection or correction codes. Mealy machines are used in applications such as error checking in data transmission, error correction in memory systems, and fault-tolerant systems.

- Finite State Automata: Mealy state machines serve as the basis for constructing finite state automata used in pattern recognition, language processing, and automation tasks. They are utilized to recognize specific input patterns or sequences and generate outputs or trigger actions based on the recognized patterns. Mealy machines are employed in areas such as natural language processing, speech recognition, and regular expression matching.

- Control Systems: Mealy state machines can be used to design and implement control systems where the output behavior depends on the current state and the inputs. They are employed in applications such as industrial automation, robotics, and embedded systems, where the control logic needs to adapt and respond to changing input conditions.

- Overall, the objective of using Mealy state machines is to create efficient and flexible systems that can process inputs and generate outputs based on both the current state and the input values. Mealy machines provide a balance between responsiveness, memory efficiency, and flexibility, making them suitable for a wide range of applications in digital systems, communication protocols, control systems, and automation tasks.

Overview of how a Mealy state machine works:

- States: A Mealy state machine consists of a set of states, each representing a specific condition or behavior of the system. The machine starts in an initial state and transitions between states based on input signals and the current state.

- Inputs: The Mealy machine receives input signals or events that trigger state transitions and affect the output generation. Inputs can be external signals, sensor readings, button presses, or any other relevant signals to the system.

- State Transitions: The Mealy state machine transitions from one state to another based on the current state and the input signals. State transitions are typically defined using a state transition diagram or table, which specifies the conditions for moving from one state to another.

- Output Generation: In addition to state transitions, the Mealy machine generates outputs based on the current state and the input signals. The output is associated with each transition or state and is determined by the combination of the current state and the input signals.

- Timing: Mealy state machines are often synchronized with a clock signal. The state transitions and output generation typically occur on clock edges, ensuring precise timing and coordination of the system's behavior.

- Feedback: Mealy machines can have feedback loops where the output signals can also influence future state transitions or input processing. This feedback mechanism allows the machine to have memory and react to the history of inputs and outputs.

- Iteration: The Mealy state machine continues to operate in a loop, repeatedly processing inputs, transitioning between states, and generating outputs based on the current state and inputs. The machine remains in this iterative process until it reaches a terminal or final state or encounters a specific condition for termination.

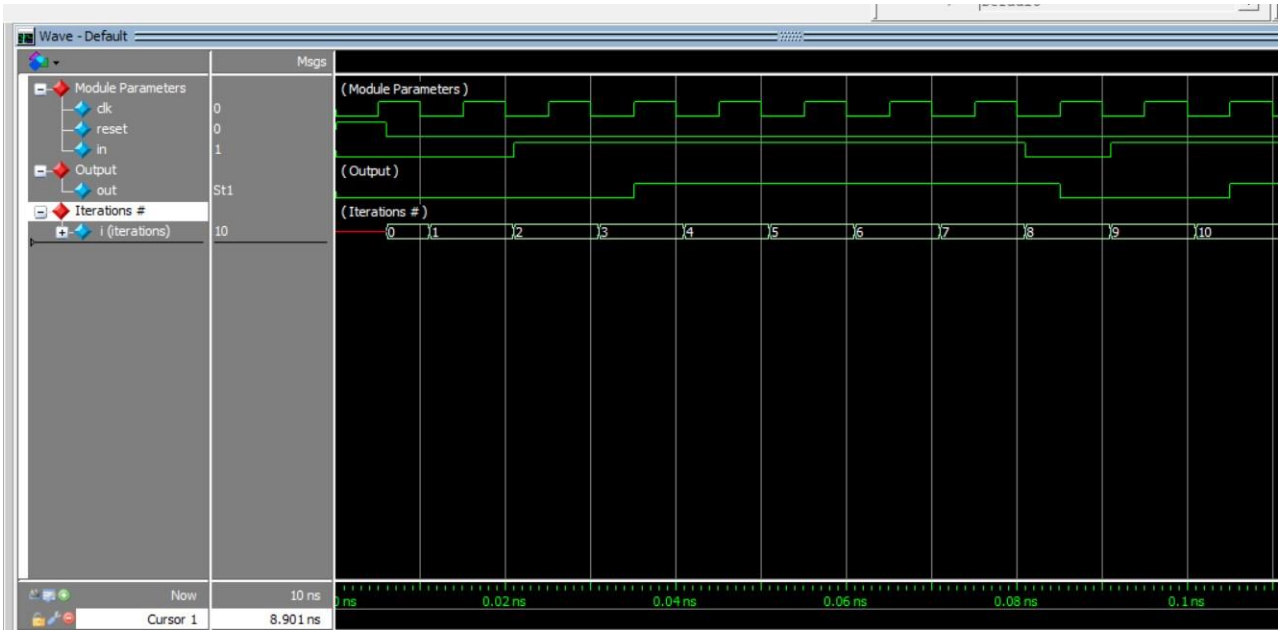
The challenges and problems we faced:

Using modelsim in the first place as we had problems in the setup.

Coming up with an idea that is both sequential and combinational.

The Simulation:

Wave:

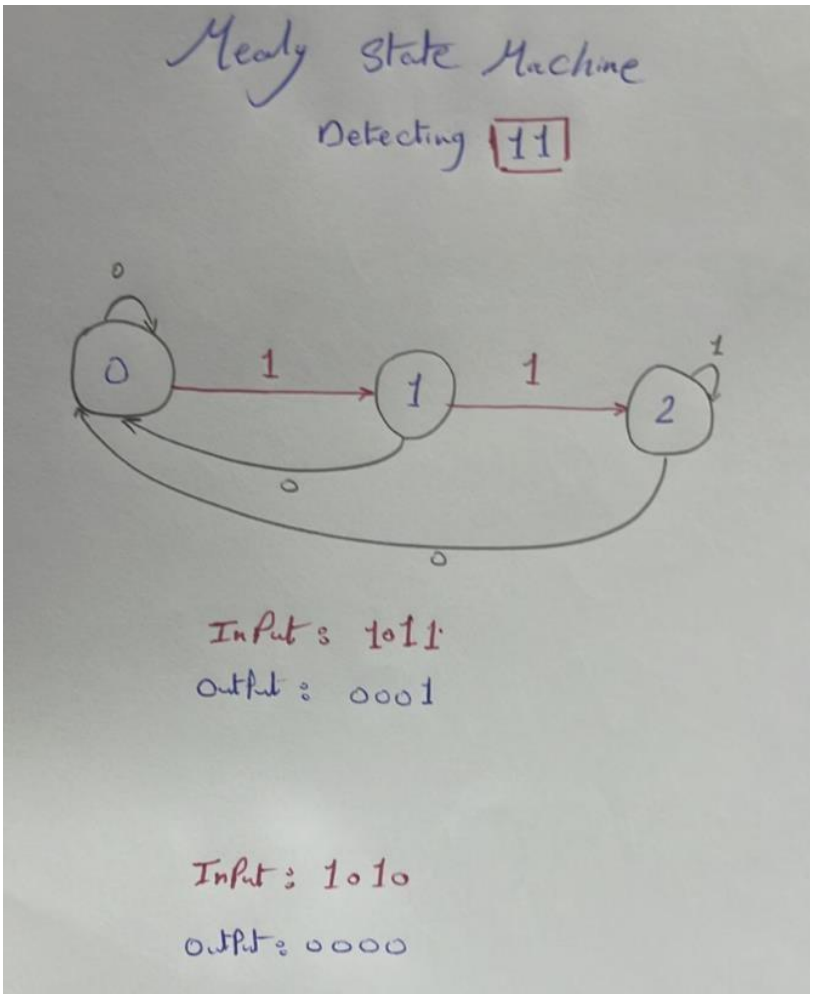


Transcript:

```
add wave -position insertpoint sim:/state_machine_mealy_tb/*
VSIM 3> run
# PASS: Sequence 11 detected i =          3
#
# PASS: Sequence 11 detected i =          4
#
# PASS: Sequence 11 detected i =          5
#
# PASS: Sequence 11 detected i =          6
#
# PASS: Sequence 11 detected i =          7
#
```

As provided it detected the 11 Sequence at iteration 3 after already reaching two1s in previous state so the current value for ‘out’ is 1 and still follows the Finite state machine that we will be explaining in next sections till reaching iteration 7 then it returns to state ‘0’

The Finite State Machine of Mealy state machine:



Here's a summary of the finite state machine's behavior:

- The FSM has three states: zero, one1, and two1s.
- The initial state is zero.The FSM takes an input signal in and generates an output signal out.
- The FSM transitions between states based on the current state and the input value.
- The state transitions and output generation are defined by the following rules:

1- From the zero state:

- If the input in is 0, it remains in the zero state and sets the output out to 0.
- If the input in is 1, it transitions to the one1 state and sets the output out to 0.

2- From the one1 state:

- If the input in is 0, it transitions back to the zero state and sets the output out to 0.
- If the input in is 1, it transitions to the two1s state and sets the output out to 0.

3- From the two1s state:

- If the input in is 0, it transitions back to the zero state and sets the output out to 0.
- If the input in is 1, it remains in two1s state and sets the output out to 1.
- The test bench provided in the code tests the FSM's functionality by applying random inputs and checking if the output out is 1, indicating the detection of a specific sequence. The test bench includes delays to control the timing of events during simulation.
- The clk signal is used to trigger state updates and output generation on the positive edge.
- The reset signal is used to initialize the state to zero.
- The state transitions and output generation are implemented using combinational always blocks that trigger whenever there is a change in the state or in signals.

The test bench “state_machine_mealy_tb()” initializes the input signals and runs a simulation for 10 iterations. In each iteration, the in signal is assigned a random value using $random, and if the output out is equal to 1, a message is displayed indicating the

detection of the "11" sequence. After the iterations, there is a delay of 50 time units, and then the simulation is finished using $finish.

In summary, the Mealy finite state machine implemented by the code transitions between states based on the input values and generates corresponding outputs. we used to detect specific input sequence “11” so our straight forward app to detect sequence “11”. Also it can be used in other apps such as:

Error Handling - Control Systems - Data Compression