

Department of Computer Science

Institute of Business Administration, Karachi

Lab #9: Moore-Based Sequence Detector Design and Circuit Implementation

Digital Logic Design

October 10, 2024

### Part 1: Introduction to a Sequence Detector (e.g., 101 Sequence)

**Objective**: Learn the basics of sequence detectors by examining a Verilog code example for a 101 sequence detector.

1. **Theory**:
   * A sequence detector is an FSM that checks for a specific bit pattern in a continuous input stream.
   * This lab begins with the detection of the sequence 101 using a state machine.
2. **Example Code**:
   * Analyze the provided 101 sequence detector code.  
     A screenshot of a computer program

     Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

* + Here is the code link as well: https://edaplayground.com/x/wr4w
  + Notice how states are used to track the bit pattern (1, 0, 1) and how transitions occur based on the input bitstream.

1. **Code Explanation**:
   * Your instructor will insert code here. Focus on understanding the state encoding and the transition logic.
2. **Moore Machine Concept**:
   * For this lab, you'll implement a Moore machine, where the output detected depends solely on the current state and not on the input.

### Part 2: Designing a Custom Sequence Detector for a Calculated Sequence

**Steps**:

1. **Calculate Sequence**:
   * Determine the highest and lowest digit in your student ID, then find their sum.
   * This sum, converted to binary, will be your target sequence to detect in the bitstream.
   * For example, if the sum is 8, you’ll be looking to detect the sequence 1000.
2. **State Diagram and State Table**:
   * Draw a state diagram representing each part of your sequence.
   * Create a state table to document each state’s behavior, showing the transitions for each possible input.
3. **Verilog Implementation**:
   * Write Verilog code for the Moore-based sequence detector.
   * Ensure that the output detected goes high when the FSM reaches the final state representing the complete sequence.
   * Base your structure on the 101 example but customize the state transitions and output to match your sequence.

### Part 3: Logisim Circuit Design Using D Flip-Flops

**Objective**: Implement the state machine in Logisim using D flip-flops to simulate each state in hardware.

1. **Circuit Setup**:
   * **D Flip-Flops**: Use D flip-flops to store the current state of the FSM.
   * **Logic Gates**: Use logic gates to implement the transition conditions for each state based on the input.
   * **Output**: Connect logic to ensure that the detected output only goes high in the final state.
2. **State Encoding**:
   * Assign binary codes to each state in your FSM.
   * Use the D flip-flops to store and transition between these states.
3. **Simulation and Verification**:
   * Simulate your design in Logisim to verify correct state transitions and output detection.
   * Attach screenshots of the Logisim circuit, state diagram, and state table.
4. **Documentation**:
   * Capture and submit screenshots of your Logisim circuit, state diagram, and state table.
   * Label each state and ensure that your circuit’s transitions align with the state transitions outlined in your Verilog code.

### Submission Requirements

1. **Verilog Code**: Submit your Verilog code for the sequence detector of lowest and highest digit of your ID.
2. **State Diagram and State Table**: Include a state diagram and state table showing each state, transition conditions, and outputs.
3. **Logisim Circuit**: Attach screenshots of your working Logisim circuit, showing D flip-flops, logic gates, and detected output.
4. **Explanation**: Write a brief explanation of how your FSM works, highlighting the differences between Moore and Mealy machines.