CSE 436/536 Term Project:

State Machine Design and Optimization

Using Magic, SPICE, and Logical Effort

Deadline: 14/01/2025 23:59

Objective:

The objective of this project is to design, simulate, and optimize a sequential circuit based on a **Finite State Machine (FSM)**. Students will utilize Magic for layout design, Ngspice for simulation, and apply the Logical Effort method to improve the critical path delay. The project will involve creating and optimizing a small FSM, an essential component in digital design. You need a timer for that project but you will not design the timer. Instead you will control the timer with **START** output signal and read the timeout from **TIMEOUT** input signal.

Project Overview:

Students will design an **FSM** that operates as a traffic light controller for a pedestrian crossing. This project will cover state transition design, logic implementation, layout creation, timing analysis, and optimization.

Project Description:

Traffic Light Controller FSM Specification:

- **Main States:**
    1. **Green Light (G):** Pedestrian light is off, cars can go.
    2. **Yellow Light (Y):** Warning state, indicating a transition from green to red.
    3. **Red Light (R):** Pedestrian light is on, cars must stop.
- **Inputs:**
    - **CLK:** System clock.
    - **Reset (RST):** Asynchronous reset to return to the initial state.
    - **Pedestrian Request (REQ):** Request signal for pedestrian crossing.
- **Outputs:**
    - **Traffic Light Signals:** Green (G), Yellow (Y), and Red (R).
    - **Pedestrian Signal:** Walk/Stop indication.

State Transitions:

- **Initial State:** Starts in the Green (G) state.
- **START Signal:** This signal is output to a timer and it will start the timer from 0 whenever it is made 1 from 0.

- **TIMEOUT Signal:** This signal will be taken as input from the timer and it means 10 seconds have passed after the START signal is given to the timer.
- **REQ Signal:**
  - If REQ is asserted during the Green state and 10 seconds have passed since the start of the green led, the FSM transitions to Yellow on the next clock cycle.
  - From Yellow, it transitions to Red.
  - After 10sec duration in Red, it returns to Green.
- **Reset:** Returns the system to the Green state.

Project Phases:

Phase 1: FSM Design and Implementation (Magic)

1. **State Diagram and Truth Table:**
   - Design the state diagram and derive the state transition table.
   - Create a truth table for state transitions and outputs.
2. **Schematic Design:**
   - Implement the FSM using D flip-flops (DFFs) for state memory and combinational logic gates for next-state logic.
   - Use a clocked synchronous design.
   - Use Logisim for that purpose. Your design should execute in Logisim.
3. **Layout Design:**
   - Design the layout in Magic, ensuring proper placement of flip-flops and combinational logic.
   - Perform Design Rule Check (DRC) and Layout vs. Schematic (LVS) verification.
4. **SPICE Netlist Extraction:**
   - Extract the SPICE netlist from the layout for simulation.

Phase 2: Functional Verification (Ngspice)

1. **Simulation Setup:**
   - Set up the simulation in Ngspice to verify the FSM operation.
   - Apply clock, reset, and pedestrian request signals.
   - Ensure correct state transitions and output behavior for all possible input conditions.
   - Consider the best demonstration for your project and use Spice accordingly.
2. **Timing Analysis:**

- ➢ Identify the critical path by measuring propagation delays, especially between state transitions.

Phase 3: Delay Optimization using Logical Effort (BONUS)

1. **Critical Path Identification:**
    - ➢ Analyze the critical path in the FSM, focusing on state flip-flops and transition logic.
2. **Logical Effort Calculation:**
    - ➢ Compute the logical effort, electrical effort, and parasitic delay for the gates in the critical path.
    - ➢ Optimize the gate sizes to reduce the overall delay.
3. **Layout Update:**
    - ➢ Apply the optimized transistor sizes in the Magic layout.
    - ➢ Verify the updated design with DRC and LVS checks.

Phase 4: Post-Optimization Verification (Ngspice)

1. **Simulate Optimized FSM:**
    - ➢ Re-run the Ngspice simulation for the optimized circuit.
    - ➢ Verify that the FSM operates correctly and that the optimized design reduces delay.
2. **Performance Comparison:**
    - ➢ Compare timing results (critical path delays) between the original and optimized designs.
    - ➢ Discuss any trade-offs in terms of area and power consumption.

Deliverables:

1. **Project Report:**
    - ➢ **Introduction:** Overview of FSM design and the traffic light controller application.
    - ➢ **Design Process:** State diagram, truth tables, and schematic design.
    - ➢ **Simulation Results:** Waveforms showing state transitions and output verification.
    - ➢ **Optimization Analysis:** Critical path analysis, Logical Effort calculations, and optimized design details.
    - ➢ **Comparison:** Performance metrics before and after optimization.
    - ➢ **Conclusion:** Insights gained, challenges faced, and potential improvements.

2. **SPICE Files:**
   - Submit SPICE netlists for both the original and optimized designs.
3. **Magic Layout Files:**
   - Provide layout files, including DRC and LVS reports.

Evaluation Criteria:

- **Design Accuracy:** Correct state diagram implementation and FSM behavior.
- **Functional Verification:** Accurate simulation results for all state transitions.
- **Optimization Effectiveness:** Degree of delay improvement and adherence to Logical Effort methodology.
- **Report Quality:** Clear, thorough documentation with supporting diagrams and analysis.
- **Demo Performance:** It is important to demonstrate your project in a user-friendly and easy to understand way.

This project will enhance your understanding of state machine design, layout implementation, and delay optimization, preparing you for advanced VLSI challenges.