Lab Report: Attendant Call System

Lab Name: Attendant Call System

Student Name: Dumany Lombe

Student Number: 101316658

Lab Section: L4

1.0 Introduction

The Attendant Call system demonstrates a sequential circuit design implemented using Verilog. The circuit simulates an event-driven notification system, where state transitions occur based on specific inputs. Such systems are widely used in applications like elevator floor requests or automated customer assistance alerts.

The circuit operates as a state machine with three states (`S0`, `S1`, and `S2`), utilizing LEDs to indicate the current state. Inputs include a clock signal, a reset button, and switches for state control. Testing was conducted through a simulation in a Verilog testbench.

The report will describe the system's specifications, design, implementation, testing, and results, concluding with insights and potential improvements.

2.0 Specifications

The Attendant Call system is defined by the following specifications:

- **Inputs:**
  - `CLK100MHZ`: A 100 MHz clock signal for synchronizing the circuit.
  - `BTNC`: An asynchronous reset button to initialize the state.
  - `SW[1:0]`: Two switches controlling state transitions.
- **Outputs:**
  - `LED[1:0]`: LEDs indicating the state of the system. LED[0] turns on in states `S1` and `S2`, while LED[1] remains off.

Behavior:

- State `S0`: Initial state after reset.
- State `S1`: Transitioned from `S0` when `SW[1]` is pressed.
- State `S2`: Transitioned from `S1` when `SW[0]` is pressed.
- The system cycles between `S1` and `S2` when `SW[0]` is toggled.
- Returning to `S0` is possible from `S2` by toggling `SW[1]`.

Constraints:

- Simulated only; no physical implementation.
- No specific limitations on the number of logic gates.

3.0 Design

3.1 State Diagram

The state machine is defined as follows:

- States: `S0 (00)`, `S1 (01)`, `S2 (10)`.
- Transitions based on `SW[1:0]`.

3.2 Verilog Implementation

The design includes two Verilog modules:

1. **Attendant_Call**: Implements the state machine with state transitions controlled by the clock and input switches.
2. **Attendant_Call_TB**: A testbench simulating input scenarios to validate the circuit.

Design Methodology

- The state machine was implemented with a `reg` variable (`state`) and a `parameter` block defining the states.
- A synchronous `always` block was used to handle state transitions.
- The LEDs were assigned logic based on the state variable.

4.0 Implementation and Testing

4.1 Simulation Environment

- Tool: ModelSim/Vivado
- Modules: `Attendant_Call` (design), `Attendant_Call_TB` (testbench)
- Simulation duration: 120 ns

4.2 Test Scenarios

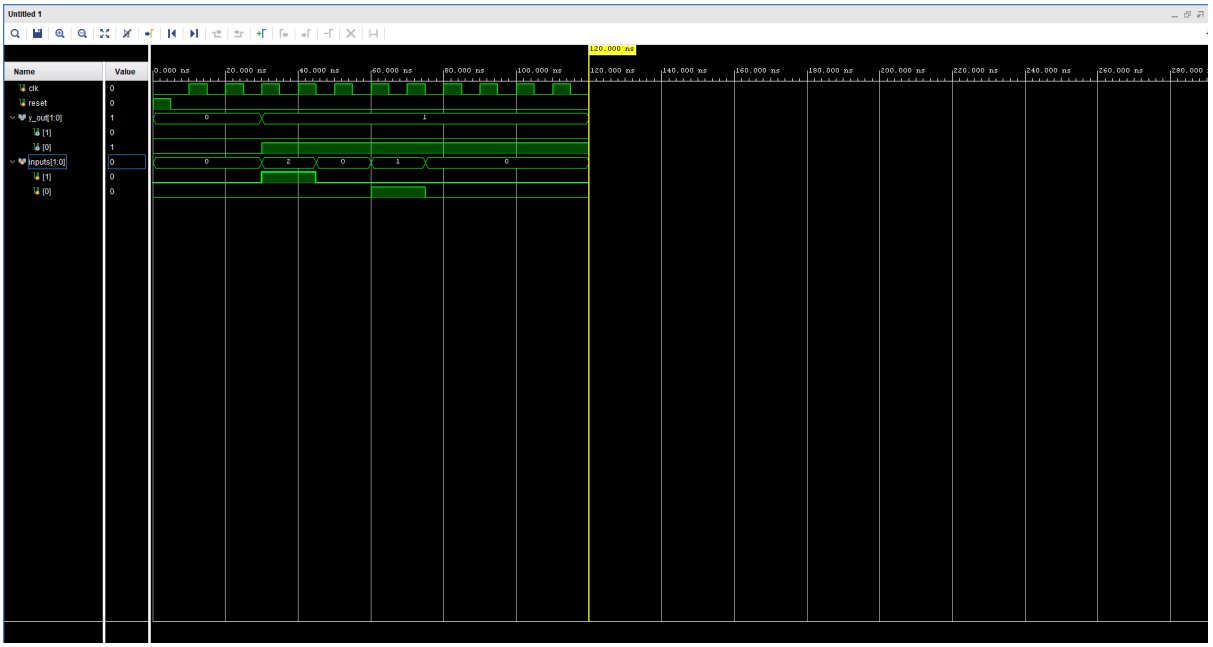
- Reset the system using `BTNC` to initialize to `S0`.
- Test state transitions by toggling `SW[1]` and `SW[0]`.
- Observe LED behavior for each state.

4.3 Results

Waveform Analysis:

- Initial state (`S0`): Both LEDs off.
- State `S1`: LED[0] on.
- State `S2`: LED[0] remains on.
- Correct transitions observed for all test cases.

Bitstream Screenshot:



5.0 Conclusion

The Attendant Call system was successfully implemented and verified through simulation, demonstrating the core principles of synchronous sequential circuits. The circuit met all specified objectives, with proper state transitions and LED indications per the design requirements. The simulation validated the functionality of the state machine, with all transitions responding accurately to the input switches and reset button. While the system effectively handled state management, areas for improvement include incorporating physical switch inputs with debouncing logic to enhance real-world applicability. Expanding the design to accommodate more complex state systems could further improve its versatility. Overall, this project provided valuable insights into state machine design and implementation, highlighting its utility in applications such as elevator controls, event management systems, and other notification-based circuits. Future design iterations could address the identified limitations and broaden its use cases.

Appendix

1. **Verilog Code:**

Attendant_Call module:

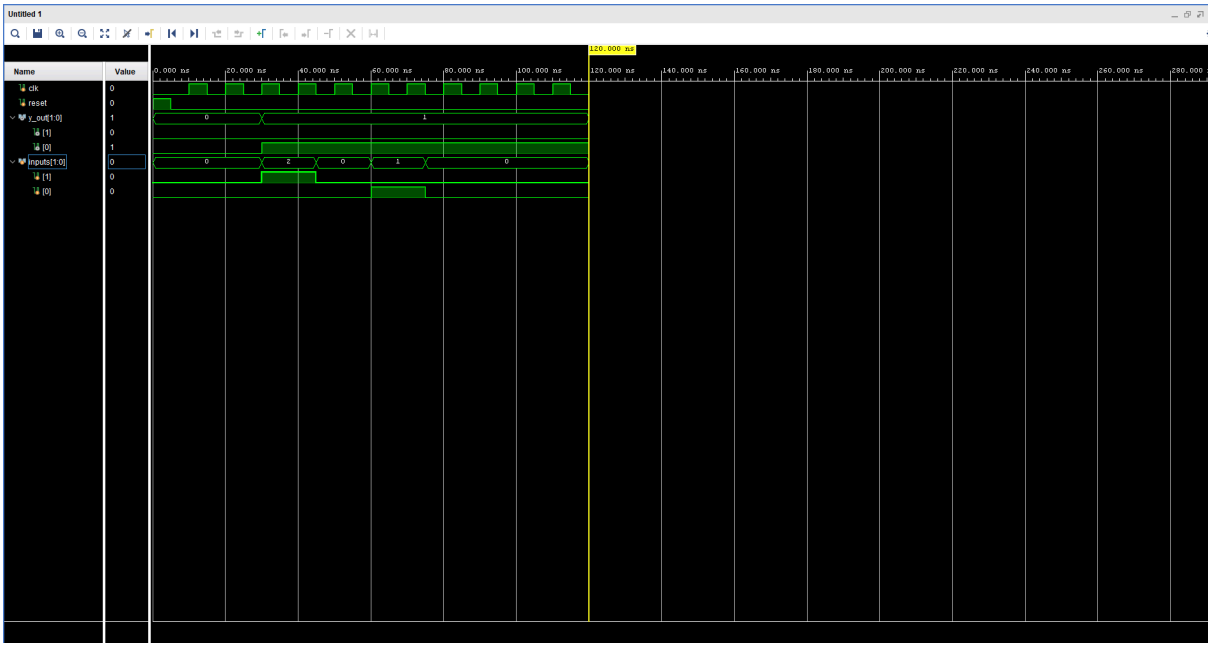
```
`timescale 1ns / 1ps

module Attendant_Call(
    input CLK100MHZ,
    input BTNC,
    output [1:0] LED,
    input [1:0] SW
    );

    reg [1:0] state;
    parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;

    always @(posedge CLK100MHZ, posedge BTNC)
        if (BTNC)
            state <= S0;
        else
            case (state)
                S0: if (SW[1]) state <= S1; else state <= S0;
                S1: if (SW[0]) state <= S2; else state <= S1;
                S2: if (SW[0]) state <= S1; else state <= S0;
            endcase

    assign LED[0] = (state == S1) || (state == S2);
    assign LED[1] = 1'b0;

endmodule
```

Attendant_Call_TB module:

```
`timescale 1ns / 1ps
```

```
module Attendant_Call_TB();

    reg clk, reset;
    wire [1:0] y_out;
    reg [1:0] inputs;

    Attendant_Call M1 (clk, reset, y_out[1:0], inputs[1:0]);

    initial #120 $finish;

    initial begin
        reset = 1;
        clk = 0;
        #5 reset = 0;
        forever #5 clk = ~clk;
    end

    initial begin
        inputs[1:0] = 2'b0;
        #15 reset = 0;
        #15 inputs[1] = 1'b1;
        #15 inputs[1] = 1'b0;
        #15 inputs[0] = 1'b1;
        #15 inputs[0] = 1'b0;
    end

endmodule
```

2. **Simulation Results:**


Untitled 1
Name
Value
clk
reset
y_out[1:0]
[1]
[0]
inputs[1:0]
[1]
[0]
0
0
1
0
1
0
0
0
120.000 ns
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
120.000 ns
140.000 ns
160.000 ns
180.000 ns
200.000 ns
220.000 ns
240.000 ns
260.000 ns
280.000 ns
0
1
0
2
0
1
0