Vending Machine Module Report

S Ujwal Niranjan

Contents

1 Introduction

This report documents the design and implementation of a vending machine module using Verilog. The module is designed to simulate a simple state machine for managing the credit, dispensing items, and providing change in a vending machine system. The module is implemented using a finite state machine (FSM) and includes input/output signals for proper functionality.

2 Design Overview

2.1 Module Description

The vending machine module operates based on a finite state machine (FSM) with three primary states:

- **s0 (Idle State):** This is the default state where the machine awaits input.
- **s1 (Partial Credit State):** This state indicates that partial credit has been inserted, but it is insufficient to dispense an item.
- **s2 (Ready State):** This state indicates that sufficient credit has been inserted, allowing the machine to dispense an item.

2.2 Inputs and Outputs

The module interacts with external systems through the following input and output signals:

- **Inputs:**
  - `clk`: Clock signal for synchronizing state transitions.
  - `rst`: Reset signal to initialize the state machine to its default state (`s0`).
  - `in`: A 2-bit input signal representing the amount of credit inserted into the vending machine.
- **Outputs:**
  - `out`: A signal that indicates whether an item is being dispensed (1) or not (0).
  - `change`: A 2-bit output signal representing the amount of change to be returned to the user.

2.3 FSM Transitions

The FSM transitions between states based on the input signals as follows:

- **From s0:**
  - Stay in **`s0`** if `in` is `00` (no credit inserted).
  - Move to **`s1`** if `in` is `01` (partial credit inserted).
  - Move to **`s2`** if `in` is `10` (sufficient credit inserted).

- **From s1:**
  - Return to **s0** with `change` 01 if `in` is 00.
  - Move to **s2** if `in` is 01 (additional partial credit inserted).
  - Dispense an item and return to **s0** if `in` is 10 (sufficient credit inserted).
- **From s2:**
  - Return to **s0** with `change` 10 if `in` is 00.
  - Dispense an item and return to **s0** if `in` is 01 or 10.

2.4 State Transition Diagram

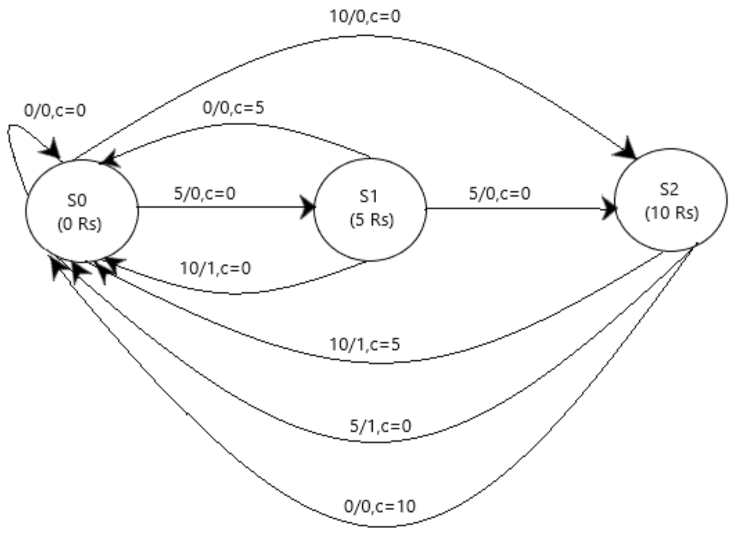

Figure 1: State Transition Diagram of the Vending Machine FSM

3 Implementation

The Verilog implementation of the vending machine module is shown below. It includes the state definitions, state transitions, and output logic.

```
module vending_machine(
    input clk,
    input rst,
    input [1:0] in,
    output reg out,
    output reg [1:0] change
);
```

```
parameter s0 = 2'b00;
parameter s1 = 2'b01;
parameter s2 = 2'b10;

reg [1:0] c_state, n_state;

always @(posedge clk) begin
    if (rst == 1) begin
        c_state <= s0;
        change <= 2'b00;
        out <= 0;
    end else begin
        c_state <= n_state;
    end
end

always @(*) begin
    case (c_state)
        s0: begin
            if (in == 2'b00) begin
                n_state = s0;
                out = 0;
                change = 2'b00;
            end else if (in == 2'b01) begin
                n_state = s1;
                out = 0;
                change = 2'b00;
            end else if (in == 2'b10) begin
                n_state = s2;
                out = 0;
                change = 2'b00;
            end else begin
                n_state = s0;
            end
        end
        s1: begin
            if (in == 2'b00) begin
                n_state = s0;
                out = 0;
                change = 2'b01;
            end else if (in == 2'b01) begin
                n_state = s2;
                out = 0;
                change = 2'b00;
            end else if (in == 2'b10) begin
                n_state = s0;
                out = 1;
                change = 2'b00;
            end else begin
                n_state = s0;
            end
        end
        s2: begin
            if (in == 2'b00) begin
                n_state = s0;
                out = 0;
                change = 2'b10;
            end else if (in == 2'b01) begin
                n_state = s0;
                out = 1;
                change = 2'b00;
            end else if (in == 2'b10) begin
                n_state = s0;
                out = 1;
                change = 2'b01;
            end else begin
                n_state = s0;
            end
        end
        default: begin
            n_state = s0;
            out = 0;
            change = 2'b00;
        end
    endcase
end

endmodule
```

Listing 1: Vending Machine Module

4 Constraints File

The constraints file defines the mapping of input and output signals to FPGA pins. Below is the content of the constraints file used in this design:

```
# Constraints for Vending Machine Module
set_property PACKAGE_PIN W5 [get_ports clk]      # Clock
   input
set_property IOSTANDARD LVCMOS33 [get_ports clk]

set_property PACKAGE_PIN U18 [get_ports rst]     # Reset
   input
set_property IOSTANDARD LVCMOS33 [get_ports rst]

set_property PACKAGE_PIN V17 [get_ports {in[0]}] # Input bit
   0
set_property IOSTANDARD LVCMOS33 [get_ports {in[0]}]

set_property PACKAGE_PIN V16 [get_ports {in[1]}] # Input bit
   1
set_property IOSTANDARD LVCMOS33 [get_ports {in[1]}]

set_property PACKAGE_PIN W4 [get_ports out]      # Output for
    item dispense
set_property IOSTANDARD LVCMOS33 [get_ports out]

set_property PACKAGE_PIN U17 [get_ports {change[0]}] # Change
    bit 0
set_property IOSTANDARD LVCMOS33 [get_ports {change[0]}]

set_property PACKAGE_PIN U16 [get_ports {change[1]}] # Change
    bit 1
set_property IOSTANDARD LVCMOS33 [get_ports {change[1]}]
```

5 Conclusion

The vending machine module demonstrates the use of a finite state machine to implement a simple credit-based system for item dispensing. The design ensures proper handling of input credits, state transitions, and output signals for dispensing and change return. This modular design can be easily extended for more complex vending machine functionalities. Future enhancements can include additional states for handling larger denominations or error handling for invalid inputs.