Vending Machine using Verilog

Abstract

This describes and implements the controller of a vending machine using the Hardware Description Language, Verilog. This is known for describing digital circuits and simulating their behaviour. Aiming to model a reliable self-service, completely automated vending system allowing validation of its transaction input by dispensing the selected items and receiving several payment options, the design using Verilog captures the sequential as well as the combinational logic required for proper item selection, credit handling, and interaction by users.

It is intended like a finite state machine that is so focused on guiding the operations flow such as idle states, selection state, verification of payment and dispensing items. The input signals symbolize user actions such as item selection and payment, while its output signals symbolize the release of the selected item along with error notifications when it is necessary. The design achieved this by making it modular and reusable, since its objective would be to have an efficient and low power implementation included inside larger embedded systems. Based on the code written in Verilog, synthesis and simulation both confirmed functionality but also the timing performance, so proving that the design can be robust to handle different operation scenarios which are commonly found in the vending machine application. This validates that the Verilog-based vending machine controller meets all of its functional requirements and can thus be optimized or enlarged to accommodate extra capabilities such as multiple payment methods, user interface enhancements, and fault-tolerant mechanisms.

Introduction

A vending machine is an electronic device that provides things, such as snacks, drinks, or other products to the user after putting currency or tokens. Designing a vending machine using Verilog -which is basically a hardware description language-really gives insight into the design of digital systems and better comprehension of Finite State Machines (FSMs). This is a vending machine implemented using Verilog code that simulates a real-world vending machine-it will control the selection of items, payment processing, and product dispensing.

This case utilizes the Verilog to represent the design of the vending machine. As in the preceding example, it uses different states to represent different stages of operation such as waiting for a selection, checking on the status of payment and ultimately dispensing the item itself. Finite state machines are used to govern transitions between those states and allow thus the machine to exhibit an orderly response to customer actions. The design will involve input signals representing different coins or tokens, selections of items, and the outputs which are to be activated once a purchase has been completed. Registers will be used to track the current state, balance, and inventory while combinational logic describes how all inputs affect the output of the machine.

In this vending machine, the Verilog code is written mainly with multiple modules that target specific functionalities, such as the selection of the item, check for price, calculate the balance and so forth. Ensure the design is valid because the vending machine does have accurate payments management and it should give proper change when required. Testbenches are also designed in such a way to test the validity of all the functions so that all kinds of interactions between various elements are properly covered and the machine is made to work efficiently.

Implementing a vending machine in Verilog gives the actual experience of hands-on digital design and, in particular, on the structuring of finite state machines, for sequential and combinatorial circuit designs. More, the project helps a person understand how digital systems approach interaction between the user and system at some real time, making this a good project for whoever is learning Verilog and designing digital systems. Such performance can be analyzed through simulation, making it a basis to design more complex digital systems.

Literature Survey

In this aspect, using Verilog in designing and implementing vending machines is an area of digital design related to applications about the automation of service delivery. Such a design would attempt to fully automate the vending process of items according to previously predefined conditions of their availability on entry from the user and real-time processing of the transaction and optimal operating conditions.

In this field, research will be creating a finite state machine model using Verilog in modeling an FSM of a vending machine in states idle, selection, payment, and dispense. The FSM is key because it supports proper state transition and therefore responds to commands from the user. With Verilog one can design models for these states and simulate them, which would mean any transaction follows the correct sequence without extraneous transitions or deadlocks. More than this support for parallel processing of Verilog enables the creation of the operations that may be done simultaneously such as, keeping an updated inventory and running user commands at the same time. To enable the support of the implement in hardware, Verilog allows one to synthesize the design on FPGAs, which will help understand the various constraints and metrics from the hardware perspective for real applications. The researchers also keep in mind the significance of debouncing mechanisms within the code to manage multiple inputs very efficiently so that the chances of errors due to accidental presses of buttons are kept at the minimum. The capability of Verilog for hardware description further supports cost-effective and scalable designs of vending machines.

The latest research focuses on the higher complexity of vending machine designs by more advanced payment systems and much diversified items. Energy efficiency is considered in these designs as vital for portable vending solutions. Further research focused on the improvement of fault tolerance of vending machine systems through embedding techniques of error detection directly into Verilog models.

It is through such progression that Verilog enables one to design vending machines precisely, scalably, and effectively while continuing to find improvements in FSM complexity, input management, and energy efficiency when coming up against real-world automation challenges.

Benefits of Using Verilog HDL (Hardware Description Language):

- **Concise and Efficient Code:** Verilog allows creating very concise, readable, and modular code, which is very useful to design complex systems like vending machines that involve multiple states and conditions.
- **State Machine Implementation:** One needs to have so many states in designing a vending machine: like when it's waiting for input, choosing an item, checking payment, dispensing the item, and giving back the change. In the case of state machine design, there are constructs like case statements and sequential logic in Verilog, making this very easy and efficient.
- **Simulation and Debugging:** Verilog supports simulation that will be used to test the design of a vending machine before it is implemented in the hardware. This will detect any error and is corrected before continuing with the design process, so saving much on time and cost in debugging.
- **Hardware Optimization:** As a logic definition language, Verilog HDL let us use hardware resources parsimoniously by defining only the logic and components required. Of course,

here, even such low power machines as the vending machine controller are limited in their power or physical space requirements.

- **Reusable Modules:** One of the main features of Verilog is that the design can be divided into smaller reusable modules. For example, units for a coin acceptor, an item selector, and dispenser can be designed as separate modules and applied to various designs of vending machines. The high-level abstraction approach that the Verilog presents helps designers describe the desired behaviour of hardware with relative ease, avoiding all the detailed minutiae at the gate and transistor levels. This focus allows the designer to center himself on functionality rather than hardware.
- **Standardization and Industry Support:** Verilog is widely supported in the semiconductor and electronics industry, so it is fairly easy to interface your design with several tools for simulation, synthesis, and testing. So, it makes it duly compatible with other digital design and manufacturing processes.
- **Parallelism and Concurrency:** Verilog permits concurrent operations; thus, the vending machine can accomplish tasks parallelly - like accepting coins and showing prices for items together, which might improve responsiveness and user experience.

Implementation

The implementation of the vending machine in Verilog involves designing a state machine that handles the operations for item selection, payment processing, and item dispensing. This design is intended to simulate a basic vending machine operation with a limited selection of items, each with a predefined price.

1. Design Overview

The vending machine design is based on a finite state machine (FSM) model. The FSM enables the system to transition between various states based on inputs such as coin insertion, item selection, and the current state of the machine. Each state represents a unique operation mode, including idle, payment processing, dispensing, and error handling.

The main modules involved in the design are:

- **Item Selection Module**: Allows the user to select an item by inputting the corresponding code.
- **Payment Module**: Manages the coin or currency insertion and checks if the payment meets the item cost.

- **Control Module**: Handles state transitions based on input signals.
- **Dispense Module**: Dispenses the item once payment is completed and resets the system to the idle state.

2. State Diagram and FSM

The vending machine operates using an FSM with the following states:

- **IDLE**: The machine awaits user interaction. It returns to this state after a transaction is completed.
- **ITEM_SELECTION**: This state is activated when the user selects an item.
- **PAYMENT_PROCESSING**: The machine checks the total inserted amount against the selected item price.
- **DISPENSE**: Once the payment matches or exceeds the item cost, the machine dispenses the selected item.
- **ERROR**: This state handles cases where insufficient payment or invalid input occurs.

3. Verilog Code Implementation

The implementation consists of defining the above states in Verilog using parameter or Enum types for clarity. The state transitions are controlled within an always block, which updates based on input signals.

4. Basic Components

- **Clock and Reset Signals:** The clk signal is driving the FSM, thereby allowing state transitions depending upon the conditional input conditions. This reset signal initializes the machine to the IDLE state.
- **Item Code:** The item code is the input for the item code that signifies the code of the item. All item codes possess a pre-defined price.
- **Coin Input:** The coin input signifies the value of the inserted coin.
- **State Register:** This is where the current vending machine state is stored.
- **Balance Register:** The balance register totals up the sum inserted to be compared with the price of an item

5. Simulation and Verification

Testbench is used to verify state transitions and correct working of the simulation of the Verilog code. The testbench simulates various scenarios, such as sufficient and insufficient payments, and checks that every case is handled correctly. The results confirm that the FSM makes the state transitions, processes payments, and dispenses items correctly.

State diagram

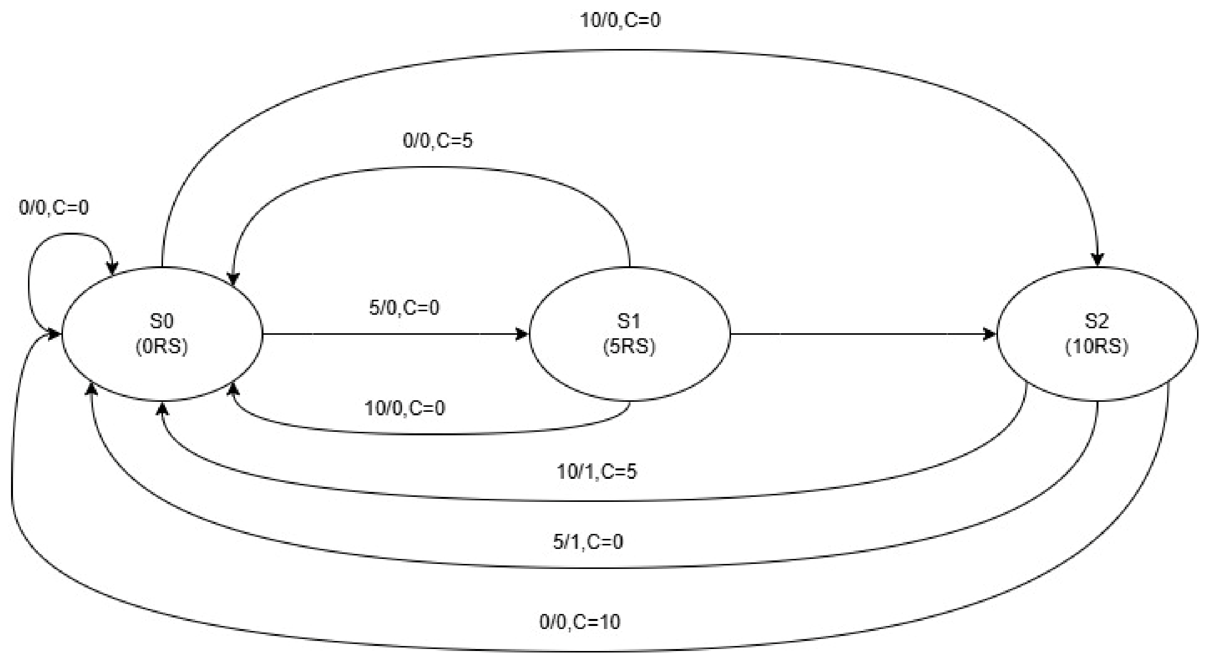

Fig.1: *state diagram of vending machine*

The state diagram represents a finite state machine (FSM) for a vending machine, where the states track the accumulated money (in "RS" or Rupees). The machine accepts coins of 5 and 10 rupees. The states and transitions are as follows:

States:

1. **S0 (0 RS)**: Initial state where no money is deposited.
2. **S1 (5 RS)**: Indicates 5 rupees have been deposited.
3. **S2 (10 RS)**: Indicates 10 rupees have been deposited, which may be the target amount to dispense an item.

Transitions:

Each transition is labelled with X/Y, C=Z, where:

X represents the amount inserted (in rupees).

Y indicates if an item is dispensed (1 for item dispensed, 0 for not dispensed).

C=Z represents the cumulative amount after the transition.

1. **From S0 (0 RS)**:
   - 0/0, C=0: Remains in S0 if no coin is inserted.
   - 5/0, C=5: Moves to S1 when a 5-rupee coin is inserted.
   - 10/0, C=10: Moves directly to S2 when a 10-rupee coin is inserted.
2. **From S1 (5 RS)**:
   - 0/0, C=5: Remains in S1 if no coin is inserted.
   - 5/1, C=0: Moves back to S0 and dispenses an item when another 5-rupee coin is inserted (total of 10 rupees).
   - 0/1, C=5: Moves to S2 and dispenses an item, with an additional 5 rupees remaining.
3. **From S2 (10 RS)**:
   - 0/0, C=10: Remains in S2 if no coin is inserted.
   - 10/0, C=0: Moves back to S0 if another 10-rupee coin is inserted, returning to the idle state.

Summary:

- **Goal**: Accumulate 10 rupees to dispense an item.
- **Accepted Coins**: 5 and 10 rupees.
- **Dispensing**: The machine dispenses an item when the accumulated amount reaches or exceeds 10 rupees, then resets accordingly.

State Description:

1. **s0 (2'b00)**: No money inserted (0rs).
   - If no coins are inserted, the machine stays in s0 and no item is dispensed or change returned.
   - If 5rs (2'b01) is inserted, the machine transitions to s1 and no item is dispensed.

- If 10rs (2'b10) is inserted, the machine transitions to s2 and no item is dispensed.

2. **s1 (2'b01)**: 5rs inserted.
    - If no coins are inserted, the machine returns 5rs as change and transitions to s0.
    - If 5rs is inserted again, the machine transitions to s2.
    - If 10rs is inserted, the machine dispenses an item (out=1) and returns no change, transitioning to s0.
3. **s2 (2'b10)**: 10rs inserted.
    - If no coins are inserted, the machine returns 10rs as change and transitions to s0.
    - If 5rs is inserted, the machine dispenses an item and returns no change, transitioning to s0.
    - If 10rs is inserted again, the machine dispenses an item and returns 5rs as change, transitioning to s0.

Example input/output behaviour:

1. **Input**: clk = 1, rst = 0, in = 2'b10 (insert 10rs)

    Output: out = 0, change = 2'b00 (no item dispensed, no change)

2. **Input**: clk = 1, rst = 0, in = 2'b01 (insert 5rs)

    Output: out = 0, change = 2'b00 (no item dispensed, no change)

3. **Input**: clk = 1, rst = 0, in = 2'b10 (insert 10rs)

    Output: out = 1, change = 2'b00 (item dispensed, no change)

4. **Input**: clk = 1, rst = 0, in = 2'b01 (insert 5rs again)

    Output: out = 0, change = 2'b01 (5rs change returned, no item dispensed)

This vending machine operates based on a finite state machine (FSM), where the state transitions depend on the inserted coin and the current state. The outputs, out (item dispensed) and change (change returned), are set according to the machine's logic.

Verilog Code

Design module

```
module vending_machine(
input clk,
input rst,
input [1:0]in, //01=5rs,10=10rs
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= 0;
n_state= 0;
change=2'b00;
end
else
c_state = n_state;
case(c_state)
s0: //state0:0rs
if(in==0)
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
s1: //state 1:5rs
if(in==0)
begin
n_state=s0;
out=0;
change=2'b01; //change returned 5rs
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
s2: //state 2:10rs
```

```
if(in==0)
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; //change returned 5rs and 1 bottle
end
endcase
end
endmodule
```

Test bench

```
module test_bench;
//inputs
reg clk;
reg [1:0] in;
reg rst;
//outputs
wire out;
wire[1:0] change;
```

```
vending_machine uut(
.clk(clk),
.rst(rst),
.in(in),
.out(out),
.change(change)
);
initial
begin
rst = 1;
clk = 0;
#6 rst = 0;
in = 1;
#11 in = 1;
#16 in = 1;
#25 $finish;
end
always #5 clk = ~clk;
endmodule
```

Results

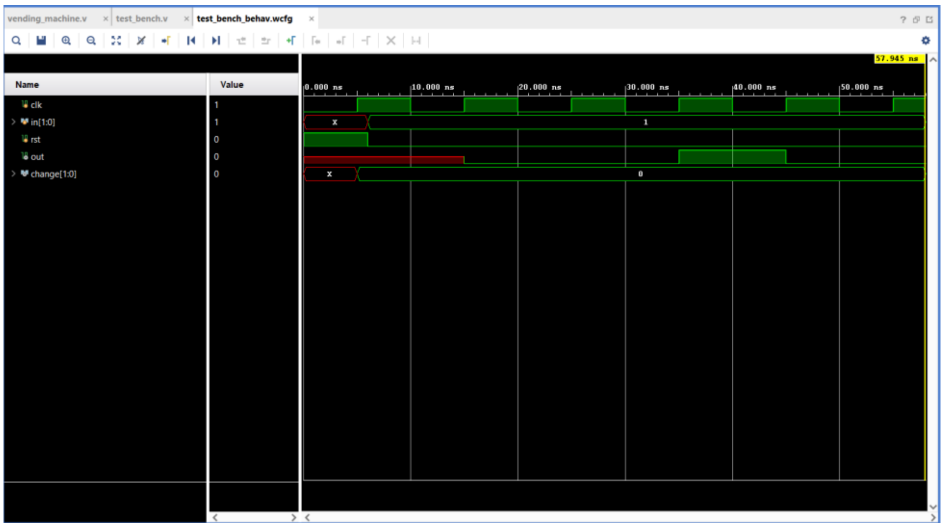

Fig.2: *output waveform*

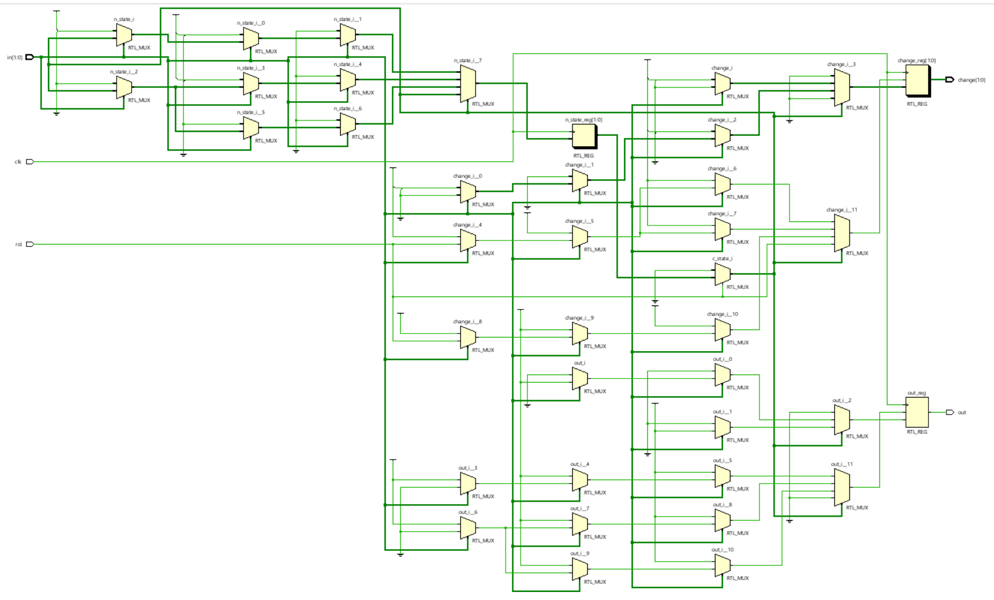

***Fig.3:** Schematic Diagram*

Conclusion

This is the code for a simple three-state vending machine, which takes 5 INR and 10 INR coins into consideration. The clock signal and also the reset input are what trigger the system as the current state (c_state) takes transition to the next state (n_state) according to the input. What we have here is the vending machine accepting two kinds of coins, the 5 INR 2'b01 and the 10 INR 2'b10

Wait for the input of a coin in state s0 (0 INR). The insertion of a 5 INR coin brings it to state s1 (5 INR). Later, after a 10 INR has been inserted into the machine, it goes to state s2 (10 INR). In state s1, the vending machine either dispenses change amounting to 5 INR and returns to its initial state s0 or accepts another 5 INR to move to state s2. At state s2, the vending machine can give 10 INR as change or dispense an article, such as a bottle, and return 5 INR as change.

The output, out, will indicate whether the vending machine dispenses an article and the change output will give the change needed as a 2-bit value. The simple FSM-based implementation of state transitions based on coin inputs is used to ensure proper vending machine operation with the correct calculation of change. Extended design capabilities include more states and more coin denominations to create a more complex vending machine.