Vending-Machine Controller

ENEE 245: Digital Circuits and Systems Laboratory
Lab 9

Objectives

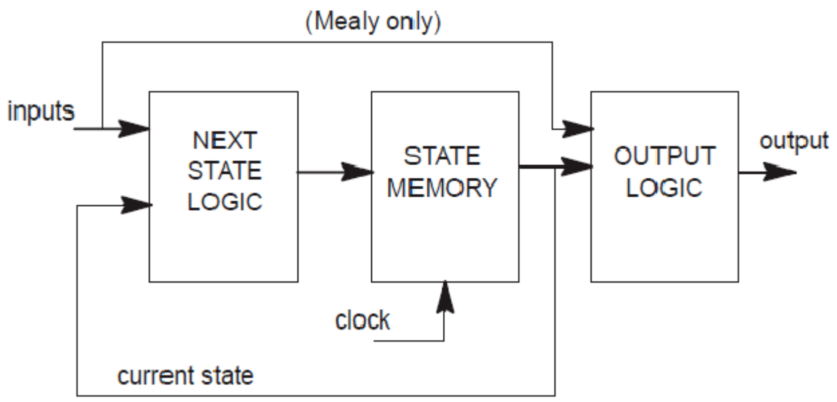
The objectives of this laboratory are the following:

- Design a vending machine controller circuit that accepts coins and product selections as inputs, and supplies requested product and cash balance.
- Display the cash balance and product cost on the 7-segment displays of the FPGA board.
- To familiarize with the design of sequential digital systems using the finite state machine (FSM) model.

This lab involves the design and implementation of a vending machine controller. The payment, dispensing, and returning change will be simulated by using the switches, buttons, and LEDs of the Nexys2 board. A user will input a product code via a keyboard interfaced with the FPGA and the system will dispense the corresponding product and change, if the price of the entered product is less than or equal to amount deposited by the user.

Finite State Machine (FSM)

A finite state machine (FSM) is an abstract description of a digital circuit in terms of (i) a collection of states and (ii) the transitions that allow the circuit to go from one state to another, based on the current input values. FSMs are very widely used in digital system designs. At the hardware level, the FSM state is kept track using a collection of flip-flops that are driven by the same clock signal (forming a synchronous sequential circuit). A combinational logic block called Next State Logic takes the state information and the system inputs as inputs and determines the next state of the circuit.



A Block Diagram of Hardware for a Finite State Machine

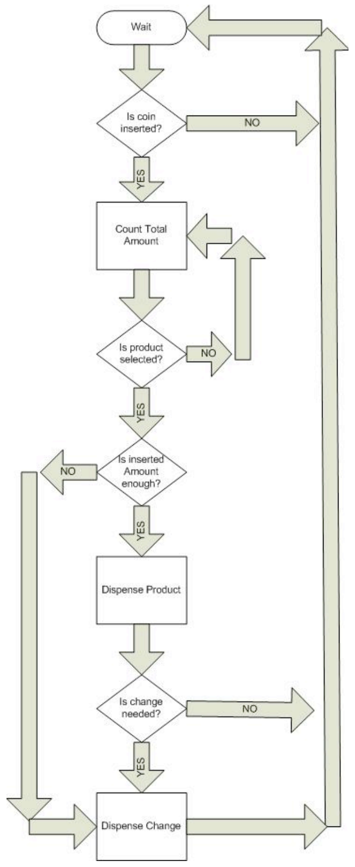
Vending Machine Controller Specification

This lab involves the design, simulation and prototyping of a vending machine controller. The completed design will be simulated in Verilog and tested by programming the Spartan 3E FPGA device of the Nexys2 board. A minimal set of assumptions would be that the machine accepts all common coins, has items with at least 4 distinct prices (all in multiples of 5 cents), and returns appropriate change. The features of the controller are summarized below:

- The vending machine has a coin-in mechanism that can accept nickels, dimes, and quarters. Assume that a pulse is generated when a coin is entered. Use BTN0, BTN1, and BTN2 for the coins. (BTN3 can be used as a master reset but note that the system should automatically reset after dispensing a product.)
- Use two of the 7-segment LED displays to display the total amount entered, which can be as high as 95 cents.
- The vending machine has four products, costing 55, 60, 65, and 70 cents. Use the slider switches (SW0 thro SW3) to select one of the products.
- Use the remaining two 7-segment LED displays to display the cost of the product selected.
- If the required amount (or more) has been entered, the machine should provide the product (turn on the corresponding LD0 thru LD3 to indicate that the product is being dispensed).

After a vending machine dispenses a product, it also releases the balance amount as change. Use two 7-digit displays to display the change given back.

The Figure below shows a flow chart depicting the working of the vending machine controller.



A Flow Chart of a Vending Machine Controller

Note that this is NOT a state machine; it simply indicates the general functionality that your system is to perform.

Verilog Implementation

In this lab, you will implement the FSM using Verilog. In Verilog, state transitions can be easily coded using the *case* structure. There are several Verilog constructs that are useful in implementing an FSM:

Always blocks

The *always* block is special for two reasons:

1. It is re-evaluated every time the value of anything on its sensitivity list changes
2. It makes it easy to do if-else logic and case statements.

Posedge CLK

The Verilog keyword *posedge* allows the variable following it to trigger the re-evaluation of the always block whose sensitivity list contains it when the value of that variable goes from 0 to 1. Similarly, *negedge* will trigger the re-evaluation when the variable value goes from 1 to 0.

When implementing an FSM in Verilog, it is often easiest to use two *always* blocks. The first *always* block will perform the sequential tasks for the machine: doing synchronous reset or updating the state. The second block will perform all the combinational logic to determine the value of the machine's output(s) and what state the machine will transition to next based on the current state and the current value of the machine's inputs. Both of these always blocks will be in the same Verilog module. Since they will be re-evaluated anytime a value changes on their sensitivity lists, the state will be updated on each positive clock edge to the computed next state and the output value(s) will change concurrently.

Pre-Lab Preparation

1. Generate a 1 Hz clock (from the 50 MHz on-board clock) for the system timing.
2. Develop Verilog codes for the vending machine controller.
3. Simulate and verify your design.

Include simulation waveforms demonstrating the correct functional operation of your vending machine. Also include schematics of your synthesized design and include a summary of the FPGA resources used.

In-Lab Procedure

Bring flash drives to store your data.

Ask the TA questions regarding any procedures about which you are uncertain.

1. Show your code and circuit to your TA.
2. Demonstrate the operation of your vending machine controller on the 7-segment LED displays.

Post-Lab Report

Write up your code, schematics, and lab procedures.

1. Suggest additional features that can make your vending machine more user-friendly.
2. What digital logic elements are generally used to represent the "states" in the finite state machine? Suggest an alternative way of implementing state information.