Vending Machine using Verilog

Ajay Sharma

3rd May 05
# Contents

1  Introduction ........................................... 2

2  Finite State Machine .................................. 3
   2.1  MOORE ............................................. 3
   2.2  MEALY ............................................. 4
   2.3  State Diagram ..................................... 4

3  Verilog Code of the Machine ......................... 6

4  Test Bench ............................................. 10

5  Output .................................................. 14

6  Explanation of Output ................................ 17
Chapter 1

Introduction

Vending Machine is a soft drink dispenser machine that dispenses drink based on the amount deposited in the machine. It accepts all the coins ie: Nickel(5 cents), Dime(10 cents), Quarter(25 cents). Till it receives 40 cents it will not dispense anything. After it has received 40 cents it will dispense a soft drink. Any amount above that will be given back as a change.

The diagram below is the block diagram of the vending machine. The inputs are the coins and the output is the drink.

![Diagram of Vending Machine]

In the next chapter we will see how we can encode the state machine in a state diagram and draw it.
Chapter 2

Finite State Machine

Any Sequential digital circuit can be converted into a state machine using state diagram. In a State machine the circuit’s output is defined in a different set of states ie. each output is a state. There is a State Register to hold the state of the machine and a nextstate logic to decode the nextstate. There is also a output register that defines the output of the machine. The nextstate logic is the sequential part of the machine and the Output and Currentstate are the Register part of the logic.

There are two types of state machines:

1. MOORE

2. MEALY

Lets see each:

2.1 MOORE

In a moore machine the output state is totally dependent on the present state. The diagram shows the information.
2.2 MEALY

In a mealy machine the output depends on the input as well as the present state.

2.3 State Diagram

The State Diagram of the Vending Machine is given here. It has the following states:

1. State 1: reset
2. State 2: Five
3. State 3: Ten
4. State 4: Fifteen
5. State 5: Twenty
6. State 6: Twenty Five

The next state is the Reset state again. The diagram is given below and is self explanatory.
What happens is whenever we get a coin we jump to the next state. So for example we get a coin from the reset state say a NICKEL, then we jump to the next state FIVE. Otherwise we stay in the same state. When we get Extra amount we come back to the reset state and the difference is given back to the user.

In the next chapter we will see the Verilog Code for the Vending Machine/State Machine.
Chapter 3

Verilog Code of the Machine

The code below is the verilog Code of the State Diagram. There are two parts to the code. The first being the Sequential Logic that decides where to go next or the change in state. The second being the one that decides the Outputs of each state. The Code is as follows:

\***********************
\***AUTHOR : AJAY SHARMA
\************************
\please see the website:
\**************************
\ http://www.csun.edu/~ags55111
\**************************
\for the entire source code of the machine

\module name
module fsm(clock,reset,coin,vend,state,change);

\these are the inputs and the outputs.
input clock;
input reset;
input [2:0]coin;

output vend;
output [2:0]state;
output [2:0]change;
i need to define the registers as change, coin and vend
reg vend;
reg [2:0]change;
wire [2:0]coin;

my coins are declared as parameters to make reading better.
parameter [2:0]NICKEL=3’b001;
parameter [2:0]DIME=3’b010;
parameter [2:0]NICKEL_DIME=3’b011;
parameter [2:0]DIME_DIME=3’b100;
parameter [2:0]QUARTER=3’b101;

MY STATES ARE ALSO PARAMETERS. I DONT WANT TO MAKE YOU READ
\IN MACHINE LANGUAGE
parameter [2:0]IDLE=3’b000;
parameter [2:0]FIVE=3’b001;
parameter [2:0]TEN=3’b010;
parameter [2:0]FIFTEEN=3’b011;
parameter [2:0]TWENTY=3’b100;
parameter [2:0]TWENTYFIVE=3’b101;

AS ALWAYS THE STATES ARE DEFINED AS REG
reg [2:0]state,next_state;

MY MACHINE WORKS ON STATE AND COIN
always @(state or coin)
begin
  next_state=0; /*VERYFIRST NEXT STATE IS GIVEN ZERO
  case(state)
IDLE: case(coin) /*THIS IS THE IDLE STATE
  NICKEL: next_state=FIVE;
    DIME: next_state=TEN;
QUARTER: next_state=TWENTYFIVE;
default: next_state=IDLE;
  endcase

FIVE: case(coin) /*THIS IS THE SECOND STATE
NICKEL: next_state=TEN;
    DIME: next_state=FIFTEEN;
QUARTER: next_state=TWENTYFIVE; //change=NICKEL
default: next_state=FIVE;
endcase

TEN: case(coin) \THIS IS THE THIRD STATE
NICKEL: next_state=FIFTEEN;
DIME: next_state=TWENTY;
QUARTER: next_state=TWENTYFIVE; //change=DIME
default: next_state=TEN;
endcase

FIFTEEN: case(coin) \THIS IS THE FOURTH STATE
NICKEL: next_state=TWENTY;
DIME: next_state=TWENTYFIVE;
QUARTER: next_state=TWENTYFIVE; //change==NICKEL_DIME
default: next_state=TWENTYFIVE;
endcase

TWENTY: case(coin) \THIS IS THE FIFTH STATE
NICKEL: next_state=TWENTYFIVE;
DIME: next_state=TWENTYFIVE; //change=NICKEL
QUARTER: next_state=TWENTYFIVE; //change==DIME_DIME
default: next_state=TWENTY;
endcase

TWENTYFIVE: next_state=IDLE; \THE NEXT STATE HERE IS THE RESET

default : next_state=IDLE;
endcase
end

always @(clock)
begin \WHENEVER I GIVE A RESET I HAVE TO MAKE THE STATE TO IDLE AND VEND TO 1
if(reset) begin
    state <= IDLE;
    vend <= 1'b0;
    // change <= 3'b000;
    end \THE CHANGE ALSO HAS TO BECOME NONE
else state <= next_state;
end
case (state) \HERE WE DECIDE THE NEXT STATE
\ALL THE STATES ARE DEFINED HERE AND THE OUTPUT IS ALSO GIVEN

IDLE: begin vend <= 1'b0; change <=3'd0; end
FIVE: begin vend <= 1'b0; if (coin==QUARTER) change <=NICKEL; else change <=3'd0; end
TEN: begin vend <= 1'b0; if (coin==QUARTER) change <=DIME; else change <= 3'd0; end
FIFTEEN : begin vend <= 1'b0; if (coin==QUARTER) change <=NICKEL_DIME; else change <=3'd0; end
TWENTY : begin vend <= 1'b0; if (coin==DIME) change <=NICKEL; else if (coin==QUARTER) change <=DIME_DIME; else change <=3'd0; end
TWENTYFIVE : begin vend <= 1'b1; change <=3'd0; end
default: state <= IDLE;
endcase
end
endmodule

In the next chapter we see the Test Bench of the machine.
We try to test the machine for the outputs. We add coins to the machine and see after which state does it give output or Softdrink.

The testbench code is given below.

\AUTHOR: AJAY SHARMA
\PLEASE VISIT: HTTP://WWW.CSUN.EDU/~AGS55111
\FOR THE ENTIRE SOURCE CODE OF THE MACHINE.

\OK
\SO I HAVE TO INCLUDE THE STATEMACHINE VERILOG CODE
\IN THIS FILE SO THAT THE I CAN RUN IT AND TEST IT HERE.
\include "fsm2.v"

\THIS IS THE HIGHEST LEVEL OF THE MODULE: THE TEST BENCH.

module test;
\THE INPUT IN THE FSM MODULE ARE REG HERE
reg clock, reset;
reg [2:0] coin;

\THE OUTPUT IN THE FSM MODULE ARE WIRES HERE
wire vend;
wire [2:0] state;
wire [2:0] change;
THE PARAMETERS AGAIN FOR THE COIN AND STATE

parameter [2:0] IDLE = 3'b000;
parameter [2:0] FIVE = 3'b001;
parameter [2:0] TEN = 3'b010;
parameter [2:0] FIFTEEN = 3'b011;
parameter [2:0] TWENTY = 3'b100;
parameter [2:0] TWENTYFIVE = 3'b101;

parameter [2:0] NICKEL = 3'b001;
parameter [2:0] DIME = 3'b010;
parameter [2:0] NICKEL_DIME = 3'b011;
parameter [2:0] DIME_DIME = 3'b100;
parameter [2:0] QUARTER = 3'b101;

I MONITOR THE TIME, DRINK, RESET, CLOCK, STATE AND CHANGE FOR CHANGES.

initial begin
  $display("Time	coin	drink	reset	clock	state	change");
  $monitor("%g	%b	%b	%b	%b	%d	%d",time,coin,vend,reset,clock,state,change);

NEW FEATURE: MY MACHINE HAS THE FACILITY TO DUMP VARIABLES SO THAT I CAN VIEW THEM USING A VCD VIEWER.
$dumpvars;
$dumpfile("file.vcd"); // Dump output file.

THIS IS WHERE THE COINS ARE ADDED.

  clock = 0;
  reset = 1; \FIRST LETS RESET THE MACHINE
  #2 reset = 0;
  coin = NICKEL; \CHECK FOR STATE 1
  #2 reset = 1; coin = 2'b00;
  #2 reset = 0;
  coin = DIME;
  \RESET AGAIN AND CHECK FOR STATE 2
  #2 reset = 1; coin = 2'b00;
  #2 reset = 0;
  \RESET AGAIN AND CHECK FOR STATE 5
coin=QUARTER;
#2 reset=1; coin=2’b00;
#2 reset=0;
//RESET AGAIN AND CHECK FOR STATE 5
coin=NICKEL;
#2 coin=NICKEL;
#2 coin=NICKEL;
#2 coin=NICKEL;
#2 coin=NICKEL;
#2 reset=1; coin=2’b00;
#2 reset=0;
//RESET AGAIN AND CHECK FOR STATE 5 AND SO ON
coin=NICKEL;
#2 coin=DIME;
#2 coin=DIME;
#2 reset=1; coin=2’b00;
#2 reset=0;
coin=NICKEL;
#2 coin=DIME;
#2 coin=QUARTER;
#2 reset=1; coin=2’b00;
#2 reset=0;
coin=NICKEL;
#2 coin=NICKEL;
#2 coin=NICKEL;
#2 coin=DIME;
#2 reset=1; coin=2’b00;
#2 reset=0;
coin=NICKEL;
#2 coin=NICKEL;
#2 coin=NICKEL;
#2 coin=DIME;
#2 reset=1; coin=2’b00;
#2 reset=0;
coin=NICKEL;
#2 coin=QUARTER;
#2 reset=1;  coin=2'b00;
#2 reset=0;
coin=NICKEL;
#2 coin=QUARTER;
#2 reset=1;  coin=2'b00;

#2 $finish;
end

\\THE CLOCK NEEDS TO TICK EVERY 2 TIME UNIT
always
#1 clock=~clock;

//always @(state)
// coin=!coin;
initial begin
if (reset)
  coin=2’b00;
end

\\THIS IS WHERE I INSTANTIATE THE MACHINE
fsm inst1(clock, reset, coin, vend, state, change);
endmodule
Chapter 5

Output

This chapter describes the output of the machine

GPLCVER_2.10c of 02/28/05 (Linux-elf).
Copyright (c) 1991-2005 Pragmatic C Software Corp.
All Rights reserved. Licensed under the GNU General Public License (GPL).
See the 'COPYING' file for details. NO WARRANTY provided.
Today is Wed May 11 03:36:14 2005.
Compiling source file "fsmttest2.v"
Highest level modules:
test

Time coin drink reset clock state change
0 000 0 1 0 0 x
1 000 0 1 1 0 0
2 001 0 0 0 0 0
3 001 0 0 1 1 0
4 000 0 1 0 0 0
5 000 0 1 1 0 0
6 010 0 0 0 0 0
7 010 0 0 1 2 0
8 000 0 1 0 0 0
9 000 0 1 1 0 0
10 101 0 0 0 0 0
11 101 0 0 1 5 0
12 000 1 1 0 0 0
13 000 0 1 1 0 0
14 001 0 0 0 0 0
15 001 0 0 1 1 0
16 001 0 0 0 2 0
17 001 0 0 1 3 0
18 001 0 0 0 4 0
19 001 0 0 1 5 0
20 001 1 0 0 0 0
21 001 0 0 1 1 0
22 001 0 0 0 2 0
23 001 0 0 1 3 0
24 000 0 1 0 0 0
25 000 0 1 1 0 0
26 001 0 0 0 0 0
27 001 0 0 1 1 0
28 010 0 0 0 2 0
29 010 0 0 1 4 0
30 010 0 0 0 5 1
31 010 1 0 1 0 0
32 000 0 1 0 0 0
33 000 0 1 1 0 0
34 001 0 0 0 0 0
35 001 0 0 1 1 0
36 010 0 0 0 2 0
37 010 0 0 1 4 0
38 101 0 0 0 5 4
39 101 1 0 1 0 0
40 000 0 1 0 0 0
41 000 0 1 1 0 0
42 001 0 0 0 0 0
43 001 0 0 1 1 0
44 001 0 0 0 2 0
45 001 0 0 1 3 0
46 001 0 0 0 4 0
47 001 0 0 1 5 0
48 010 1 0 0 0 0
49 010 0 0 1 2 0
50 000 0 1 0 0 0
51 000 0 1 1 0 0
52 001 0 0 0 0 0
53 001 0 0 1 1 0
54 001 0 0 0 2 0
55 001 0 0 1 3 0
56 001 0 0 0 4 0
57 001 0 0 1 5 0
58 001 1 0 0 0 0
59 001 0 0 1 1 0
60 010 0 0 0 2 0
61 010 0 0 1 4 0
62 000 0 1 0 0 0
63 000 0 1 1 0 0
64 001 0 0 0 0 0
65 001 0 0 1 1 0
66 001 0 0 0 2 0
67 001 0 0 1 3 0
68 101 0 0 0 4 3
69 101 0 0 1 5 4
70 000 1 1 0 0 0
71 000 0 1 1 0 0
72 001 0 0 0 0 0
73 001 0 0 1 1 0
74 101 0 0 0 2 1
75 101 0 0 1 5 2
76 000 1 1 0 0 0
77 000 0 1 1 0 0

Halted at location **fsmtest2.v(82) time 78 from call to $finish.
There were 0 error(s), 0 warning(s), and 15 inform(s).
Chapter 6

Explanation of Output

The Output is explained here. We start by adding coins. We start first with the least amount NICKEL. Then we reset the machine to see the output ie: state and the SOFTDRINK. Then we add a DIME and see the result. So on till till QUARTER is also added. Then i add a dime 5 times to see the output and it vends. ie. a softdrink is given. When i add an extra amount it gives a change also. so my machine works.

The next page has the state diagram of the machine.