CPE166 Lab 2 Part 4
By: Prof. Pang

Lab 2

Part 4: Automatic Beverage Vending Machine

The token price of a beverage is 5 cents, and our specially designed beverage vending machine only accepts tokens of 1 cent, 2 cents and 5 cents. The task of the laboratory is to design a finite state machine for the automatic beverage vending machine. The following table shows the interface signals required for this design.

Table 1. Automatic Beverage Vending Machine Interface Signals

Port Names	Port Direction	Port Size
clk	input	1 bit
reset	input	1 bit
one	input	1 bit
two	input	1 bit
five	input	1 bit
d	output	1 bit
r	output	3 bits

- When the input "reset" signal is logic high, it resets the machine to a starting state.
- When the input "one" signal is logic high, it indicates that the 1-cent token has been inserted into the vending machine.
- When the input "two" signal is logic high, it indicates that the 2-cent token has been inserted into the vending machine.
- When the input "five" signal is logic high, it indicates that the 5-cent token has been inserted into the vending machine.
- At any time, only one token can be inserted into the vending machine.
- When the output "d" signal is logic high, the vending machine returns a drink, and the output "r" signal displays the total coin token returned by the vending machine.
- When the output "d" signal is logic low, the output "r" signal should be zero.

Implement this finite state machine design in Verilog and write testbench to run simulations.

Demo Requirement

You need to demonstrate the final simulation waveform of your design to your lab instructor.

CPE166 Lab 2 Part 4
By: Prof. Pang

Lab Procedure

Step 1. Draw State Diagram

When the reset signal is logic high, the state machine goes to the first idle state. The idle state means the current received coin token is 0. The output drink is 0 and the returned out token signal r is 0.

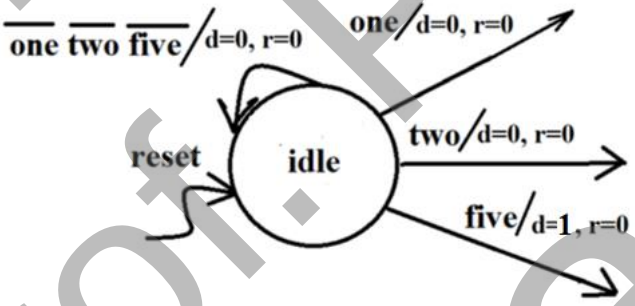


Figure 1. Idle state

Figure 1 shows the state diagram of the idle state. In the idle state, if the token is not received, the next state will still be the idle state. If a token of 1 cent is received, the "one" signal will be logic high and the finite state machine needs to enter another state. Similarly, if a token of 2 cents or a token of 5 cents is received, the finite state machine needs to enter a different state.

You can add more states to this design. In each state, you need to determine which state will become the next state based on the input signal values of "one", "two" and "five". In addition, you need to determine the output values of "d" and "r".

Step 2. Verilog Design

CPE166 Lab 2 Part 4
By: Prof. Pang

Complete the Verilog design of this project.

The Verilog design code snippet for the idle state is shown below:

```
always@(current_state or one or two or five)
begin
   case(current_state)
   idle:      if (one)   next_state =  … ;
              else if (two) next_state =   … ;
              else if (five)  next_state =   … ;
              else            next_state  =  idle;
    …       // describe more different states
     endcase
end


always@(current_state or one or two or five)
begin
   case(current_state)
   idle:     if (one)
              begin
                 d  = 0;
                 r   = 0;
             end
             else if (two)
              begin
                 d  = 0;
                  r  = 0;
             end
             else if (five)
             begin
```

```
                d = 0;
                r  = 0;
            end
            else
             begin
                d  = 0;
                r   = 0;
             end
    …       // describe more different states
     endcase
end
```

Step 3. Testbench Simulation

Write testbench to run simulations of your project.

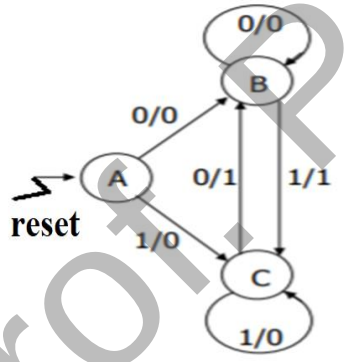
Demo Requirement

You need to demonstrate the simulation waveform to your lab instructor.

Note: Before starting this experiment, you have learned how to code a finite state machine in Verilog. This project is a design project. You need to follow the instructions above to complete the work. The following examples are for you to refresh your learning of finite state machine design.

CPE166 Lab 2 Part 4
By: Prof. Pang

Sample Example:




```
module fsm (reset, x, clk,  y, cs, ns);     //cs, and ns here are used for simulation
input       reset, x, clk;
output     y;
output  [1:0]  cs, ns;
reg          y;
reg [1:0]  cs, ns;
parameter A=2'b00,B=2'b01, C=2'b10;
always@(posedge clk or posedge reset)
begin
   if(reset) cs <= A;
   else  cs <= ns;
end
always@(cs or x)
begin
  case(cs)
    A:  y = 0;
    B:  if(x) y = 1;
        else y = 0;
    C:   if(x) y = 0;
        else y = 1;
```

CPE166 Lab 2 Part 4
By: Prof. Pang

```
    default:  y =0;
  endcase
end

always@(cs or x)
begin
    case(cs)
    A:  if(x) ns = C;
        else ns =B;
    B:  if(x) ns = C;
        else ns = B;
    C:  if(x) ns = C;
        else ns = B;
    default: ns = A;
    endcase
end
endmodule
```



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

CPE166 Lab 2 Part 4
By: Prof. Pang

```
module fsm_tb;

reg    reset, x, clk;

wire   y;

wire [1:0] cs, ns;


fsm  uut ( reset, x, clk, y, cs, ns );


initial clk = 0;

always  #10 clk=~clk;


initial begin

  $monitor($time, " ns, x=%b, cs=%b, y=%b, ns=%b", x, cs, y, ns);

  reset = 1;  x = 0;

  #25 reset = 0; x = 1;

  #40  x = 0;

  #40  x = 1;

  #40  x = 0;

  #40  x = 1;

  #40  $stop;

end


endmodule
```