Xavier Howell
10/10/20

CPE 166

LAB 2: PROF. PANG

Table of Contents

Lab Part 1.

Procedure: The purpose of this lab is to create a 3 by 3 combinational array multiplier. In order to do it we need a half adder and a full adder.

Project Summary | myl2.3

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.1/Lab2.1.srcs/sources_1/new/myl2.3

```
`timescale 1ns/1ps
// NAME: Xavier Howell
// Project: Lab2 Part 3 Multiplier


module multi(p,x,y);
    output [5:0]p;
    input [2:0]x;
    input [2:0]y;

    wire a0,a1,a2,b0,b1,c0,c1;
    assign p[0]=(x[0] & y[0]);

ha ha0(.a(x[1] & y[0]),.b(x[0] & y[1]),.sum(p[1]),.ct(a0));
ha ha1(.a(a1),.b(x[2] & y[1]),.sum(b1),.ct(a2));
ha ha2(.a(b0),.b(x[0] & y[2]),.sum(p[2]),.ct(c0));
fa fa0(.a(x[2] & y[0]),.b(x[1] & y[1]),.sum(b0),.ct(a1),.cin(a0));
fa fa1(.a(b1),.b(x[1] & y[2]),.sum(p[3]),.ct(c1),.cin(c0));
fa fa2(.a(a2),.b(x[2] & y[2]),.sum(p[4]),.ct(p[5]),.cin(c1));

endmodule
```

Project Summary | myl2.3 | myl2.1 | **myl2.2**

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.1/Lab2.1.srcs/sources_1/new/myl2.2

```
// NAME: Xavier Howell
// Project: Lab2 Part 2 Full Adder

module fa(a,b,cin,sum,ct);
  input a,b,cin;
  output sum,ct;
  wire s,c,p;

  ha g1(.a(a), .b(b), .sum(s), .ct(c));
  ha g2(.a(cin), .b(s), .sum(sum), .ct(p));

assign ct = p|c;
endmodule
```

Project Summary | myl2.3 | **myl2.1**

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.1/Lab2.1.srcs/sources_1/new/myl2.1

```
module ha (a,b,sum,ct);
// NAME: Xavier Howell
// Project: Lab2 Part 1 Half Adder

  input a,b;
  output sum,ct;
  wire s,c,p;
  xor g1(sum,a,b);
  and g2(ct,a,b);
endmodule
```

Project Summary | myl2.3 | myl2.1 | myl2.2 | myl2.3_tb

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.1/Lab2.1.srcs/sim_1/new/myl2.3_tb

```
`timescale 1ns/1ps

module multi_tb;
    reg [2:0]x;
    reg [2:0]y;
    wire [5:0]p;

    multi mulitf(.p(p),.x(x),.y(y));

    initial
      begin

         x=3'b000; y=3'b000;
      #5 x=3'b001; y=3'b001;
      #5 x=3'b010; y=3'b010;
      #5 x=3'b011; y=3'b011;
      #5 x=3'b100; y=3'b100;
      #5 x=3'b101; y=3'b101;
      #5 x=3'b110; y=3'b110;
      #5 x=3'b111; y=3'b111;
      #5 $stop;

    end
   initial $monitor($time, "x=%d, y=%d, p=%d", x,y,p);
endmodule
```

myl2.3 | myl2.1 | myl2.2 | myl2.3_tb | Untitled 1*

40.000 ns

Name	Value	0.000 ns	5.000 ns	10.000 ns	15.000 ns	20.000 ns	25.000 ns	30.000 ns	35.000 ns
x[2:0]	7	0	1	2	3	4	5	6	7
y[2:0]	7	0	1	2	3	4	5	6	7
p[5:0]	49	0	1	4	9	16	25	36	49

Lab Part 2.

Procedure: The purpose of this lab is to create a 8 bit carry selecter adder. In order to do this we need a 4 bit ripple carry adder, a MUX or Multiplexer, and a modified version of MUX – MUXB.

Untitled 1* | csa8.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.2/Lab2P2.srcs/sources_1/new/csa8.v

```
module csa8(a, b, cin, sum, cout);
input cin;
input [7:0] a, b;
output cout;
output [7:0] sum;
wire m;
wire [1:0] c;
wire [7:0] sa, sb;


rca g1(.a(a[3:0]), .b(b[3:0]), .cin(cin), .sum(sum[3:0]), .cout(m));
rca g2(.a(a[7:4]), .b(b[7:4]), .cin(1), .sum(sb[7:4]), .cout(c[1]));
rca g3(.a(a[7:4]), .b(b[7:4]), .cin(0), .sum(sa[7:4]), .cout(c[0]));

muxb g4(.d0(sa[7:4]), .d1(sb[7:4]), .s(m), .y(sum[7:4]));

mux g5(.s(m), .d0(c[0]), .d1(c[1]), .y(cout));
endmodule


//module csa8(a,b,cin,ct,sum);
//input [7:0] a;
//input [7:0] b;
//input cin;

//output [7:0] sum;
//output ct;

//wire m,c2,c3;
//wire [7:0] sa;
//wire [7:0] sb;
```

Untitled 1* | csa8.v | rca4.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.2/Lab2P2.srcs/sources_1/new/rca4.v

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module rca(a, b, cin, sum, cout);

input [3:0] a, b;
input cin;

wire [2:0] car;

output [3:0] sum;
output cout;

fullAdder g1(.a(a[0]), .b(b[0]), .cin(cin), .sum(sum[0]), .cout(car[0]));
fullAdder g2(.a(a[1]), .b(b[1]), .cin(car[0]), .sum(sum[1]), .cout(car[1]));
fullAdder g3(.a(a[2]), .b(b[2]), .cin(car[1]), .sum(sum[2]), .cout(car[2]));
fullAdder g4(.a(a[3]), .b(b[3]), .cin(car[2]), .sum(sum[3]), .cout(cout));

endmodule


//module rca4(a,b,cin,ct,sum);
//input [3:0]a;
//input [3:0]b;
//input cin;
//output ct;
```

Untitled 1* | csa8.v | rca4.v | muxb.v | mux.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.2/Lab2P2.srcs/sources_1/new/mux.v

```
// Company:
// Engineer:
//
// Create Date: 09/22/2020 09:18:46 PM
// Design Name:
// Module Name: mux
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module mux(d1,d0,s,y);
input d1,d0,s;
output y;
reg y;

always@(d1 or d0 or s)
begin
if(s) y=d1;
else y=d0;
end

endmodule
```

Untitled 1* | csa8.v | rca4.v | muxb.v

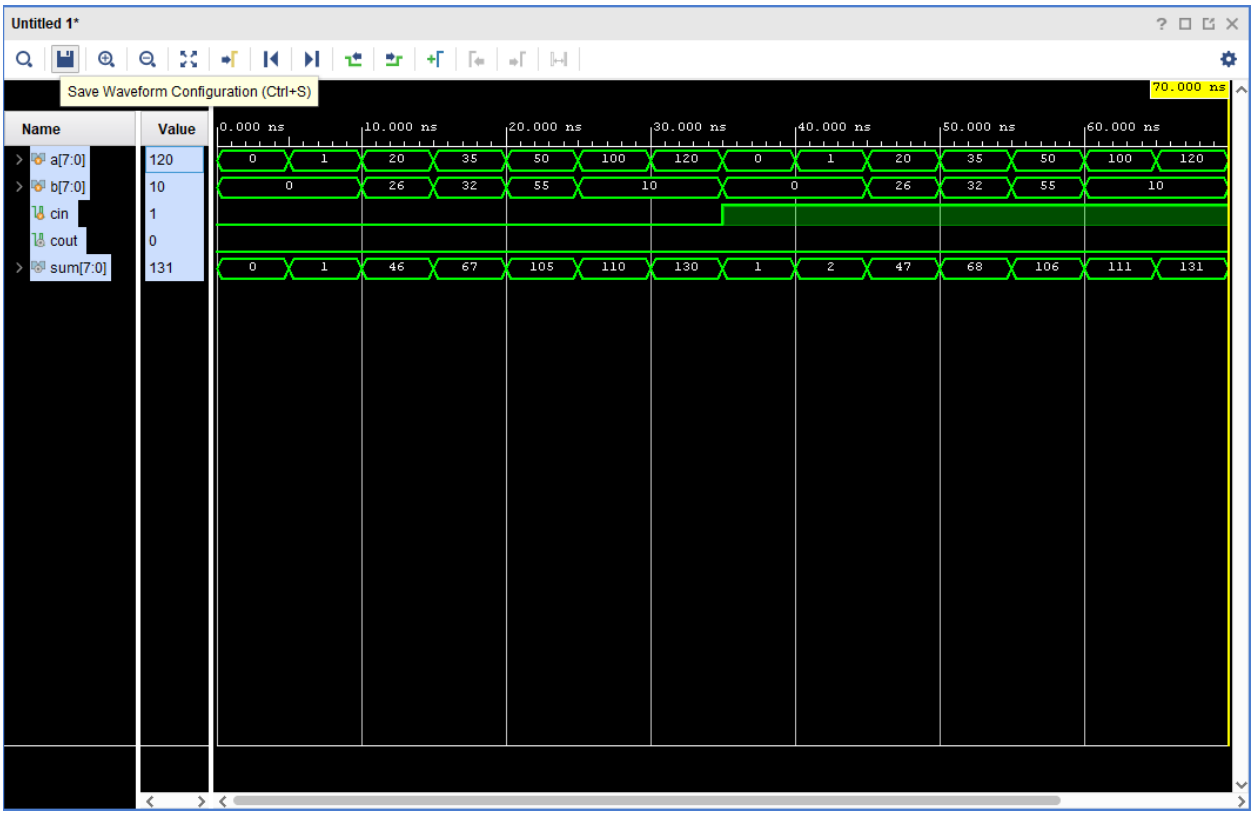
C:/Users/howel/Desktop/School/166/LAB2/Lab2P.2/Lab2P2.srcs/sources_1/new/muxb.v

```
//
// Create Date: 09/22/2020 09:57:00 PM
// Design Name:
// Module Name: muxb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module muxb(d0,d1,s,y);
input [3:0] d0;
input [3:0] d1;
input s;
output [3:0] y;
reg [3:0]y;

always@(d0 or d1 or s)
begin
if(s) y=d1;
else y =d0;
end

endmodule
```

Project Summary | csa8.v | rca4.v | muxb.v | mux.v | csa8_tb.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.2/Lab2P2.srcs/sim_1/new/csa8_tb.v

```
//reg cin;

//wire [7:0]sum;
//wire ct;

//csa8 c1(.a(a),.b(b),.cin(cin),.ct(ct),.sum(sum));
reg [7:0] a, b;
reg cin;
wire cout;
wire [7:0] sum;
csa8 u1(.a(a), .b(b), .cin(cin), .sum(sum), .cout(cout));

initial
begin
        a=0;   b=0;  cin=0;
     #5 a=1;   b=0;  cin=0;
     #5 a=20;  b=26; cin=0;
     #5 a=35;  b=32; cin=0;
     #5 a=50;  b=55; cin=0;
     #5 a=100; b=10; cin=0;
     #5 a=120; b=10; cin=0;
     #5 a=0;   b=0;  cin=1;
     #5 a=1;   b=0;  cin=1;
     #5 a=20;  b=26; cin=1;
     #5 a=35;  b=32; cin=1;
     #5 a=50;  b=55; cin=1;
     #5 a=100; b=10; cin=1;
     #5 a=120; b=10; cin=1;
     #5 $stop;
end
```


Untitled 1*
Save Waveform Configuration (Ctrl+S)
70.000 ns
Name
Value
a[7:0]
120
b[7:0]
10
cin
1
cout
0
sum[7:0]
131
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
0 1 20 35 50 100 120 0 1 20 35 50 100 120
0 26 32 55 10 0 26 32 55 10
0 1 46 67 105 110 130 1 2 47 68 106 111 131

Lab Part 3.

Procedure: The purpose of this lab is to create a two speed BCD counter. In order to do this we need a clock divider, another MUX.

Project Summary | top.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.3/Lab2P.3.srcs/sources_1/new/top.v

```
//
// Create Date: 10/04/2020 08:19:09 PM
// Design Name:
// Module Name: top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top(clk,s,y);
input clk;
input s;
output [3:0] y;
wire clkd,clky;

clkdiv10 g1(.clk1(clk), .clk2(clkd));
mux g2(.d0(clk), .d1(clkd), .s(s), .y(clky));
bcd g3(.clk(clky), .dout(y[3:0]));

endmodule
```

Project Summary | top.v | clkdiv10.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.3/Lab2P.3.srcs/sources_1/new/clkdiv10.v

Save File (Ctrl+S)

```
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module clkdiv10(clk1,clk2);
 input clk1;
 output clk2;

 parameter N = 10;
 reg[3:0] out=0;

 always @ (posedge clk1)
    begin
    if (out == N-1)
    out <= 0;
    else
    out <= out+1;
    end
    assign clk2 = (out<N/2) ?1'b0:1'b1;
endmodule
```

Project Summary | top.v | clkdiv10.v | bcd.v

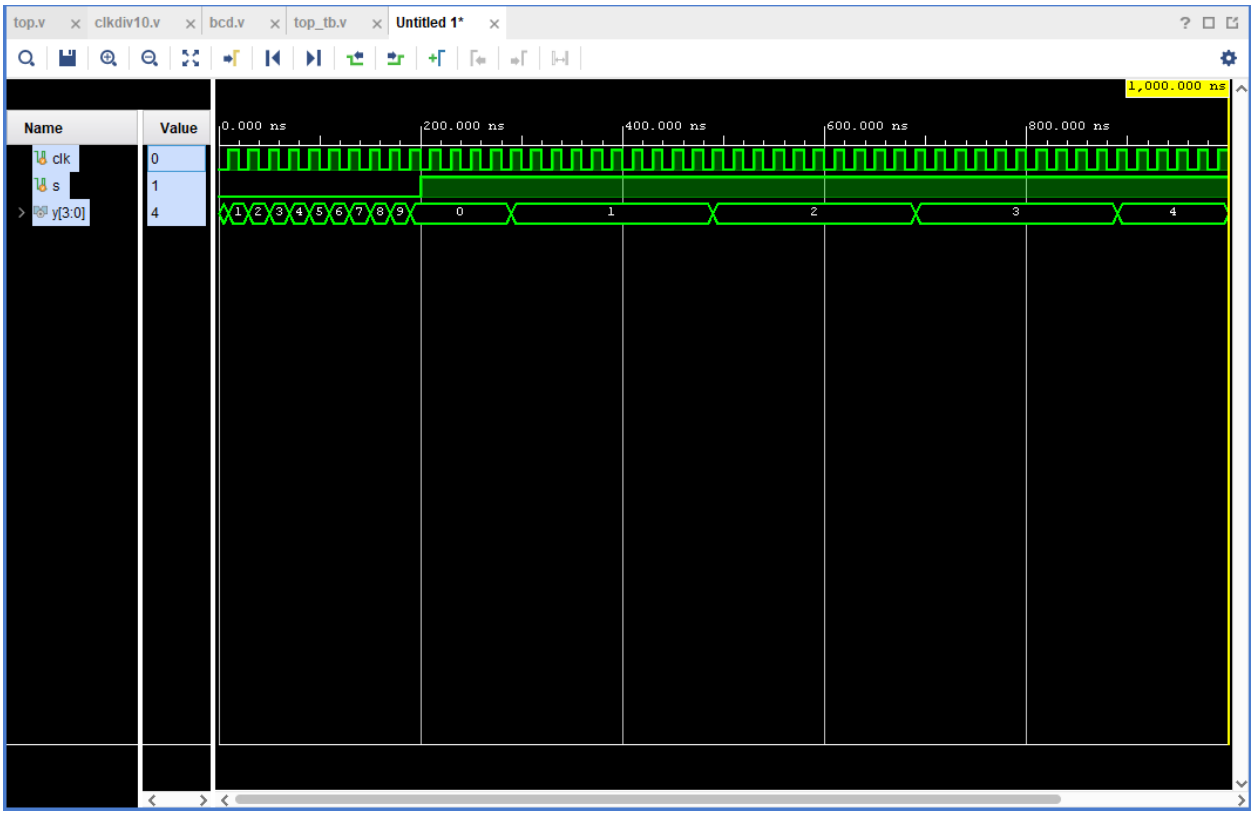
C:/Users/howel/Desktop/School/166/LAB2/Lab2P.3/Lab2P.3.srcs/sources_1/new/bcd.v

```
//
//////////////////////////////////////////////////////////////////////////////////


module bcd(clk,reset,dout);

input clk;
input reset;
output [3:0] dout;
reg [3:0] dout;

wire clk;
wire reset;

initial dout = 0;

always@(posedge (clk))
begin
    if (reset)
        dout <=0;
    else if (dout<9)
    begin
        dout<= dout + 1;
    end else if (dout==9)
    begin
    dout <=0;
    end
    end
    endmodule

```

Project Summary | top.v | clkdiv10.v | bcd.v | top_tb.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.3/Lab2P.3.srcs/sim_1/new/top_tb.v

```
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top_tb;
reg clk;
reg s;
wire [3:0] y;

top u1(.clk(clk), .s(s), .y(y));

initial
begin

   clk=1'b0;     s=1'b0;
#100 s=1'b0;
#100 s=1'b1;
end
always #10 clk = ~clk;
endmodule

```


top.v
clkdiv10.v
bcd.v
top_tb.v
Untitled 1*
Name
Value
clk
s
y[3:0]
0
1
4
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
1 2 3 4 5 6 7 8 9
0
1
2
3
4

Lab Part 4.

Procedure: The purpose of this lab is to create a automatic beverage machine. We will combine all the previous parts of this lab to create it.

Project Summary | ven.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.4/Lab2P.4.srcs/sources_1/new/ven.v

```
module ven(clock,reset,one,two,five,d,r,current_state);
input clock;
input reset;
input one,two,five;

output [3:0]r;
output d;
output [2:0]current_state;

reg [3:0]r;
reg d;
reg [2:0]current_state;
reg [2:0]next_state;

parameter idle = 3'b000;
parameter s1 = 3'b001;
parameter s2 = 3'b010;
parameter s3 = 3'b011;
parameter s4 = 3'b100;
parameter s5 = 4'b101;

always@(posedge clock or posedge reset)
begin
    if (reset)
    begin
        current_state <= idle;
    end else
    begin
        current_state <= next_state;
```

Project Summary | ven.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.4/Lab2P.4.srcs/sources_1/new/ven.v

```
    end else
    begin
        current_state <= next_state;
    end
end

always@(current_state or one or two or five)
begin
    case(current_state)
    idle:
        if(one) next_state = s1;
        else if(two) next_state = s2;
        else if(five) next_state = s5;
        else next_state = idle;

    s1:
        if(one) next_state = s2;
        else if(two) next_state = s3;
        else if(five) next_state = s5;
        else next_state = s1;

    s2:
        if(one) next_state = s3;
        else if(two) next_state = s4;
        else if(five) next_state = s5;
        else next_state = s2;

    s3:
        if(one) next_state = s4;
        else if(two) next_state = s5;
```

vsc.v

C:\Users\hoeel\Desktop\School\105L\ab2\Lab2P 4\Lab2P 4\srcs\sources_1\new\vsc.v

```
    a2:
        if(one)
        begin
            d=1;
            e=1;
        end
        else if(two)
        begin
            d=1;
            e=1;
        end
        else if(five)
        begin
            d=1;
            e=1;
        end
        else
        begin
            d=1;
            e=1;
        end

    a3:
        if(one)
        begin
            d=1;
            e=1;
        end
        else if(two)
        begin
            d=1;
            e=1;
        end
        else if(five)
        begin
            d=1;
            e=1;
        end
        else
        begin
            d=1;
            e=1;
        end

    a4:
        if(one)
        begin
            d=1;
            e=1;
        end
        else if(two)
        begin
            d=1;
            e=1;
        end
        else if(five)
        begin
            d=1;
            e=1;
        end
        else
        begin
            d=1;
            e=1;
        end

    a5:
        d=1;
        default: d=1;
    endcase
end

endmodule
```

vsc.v

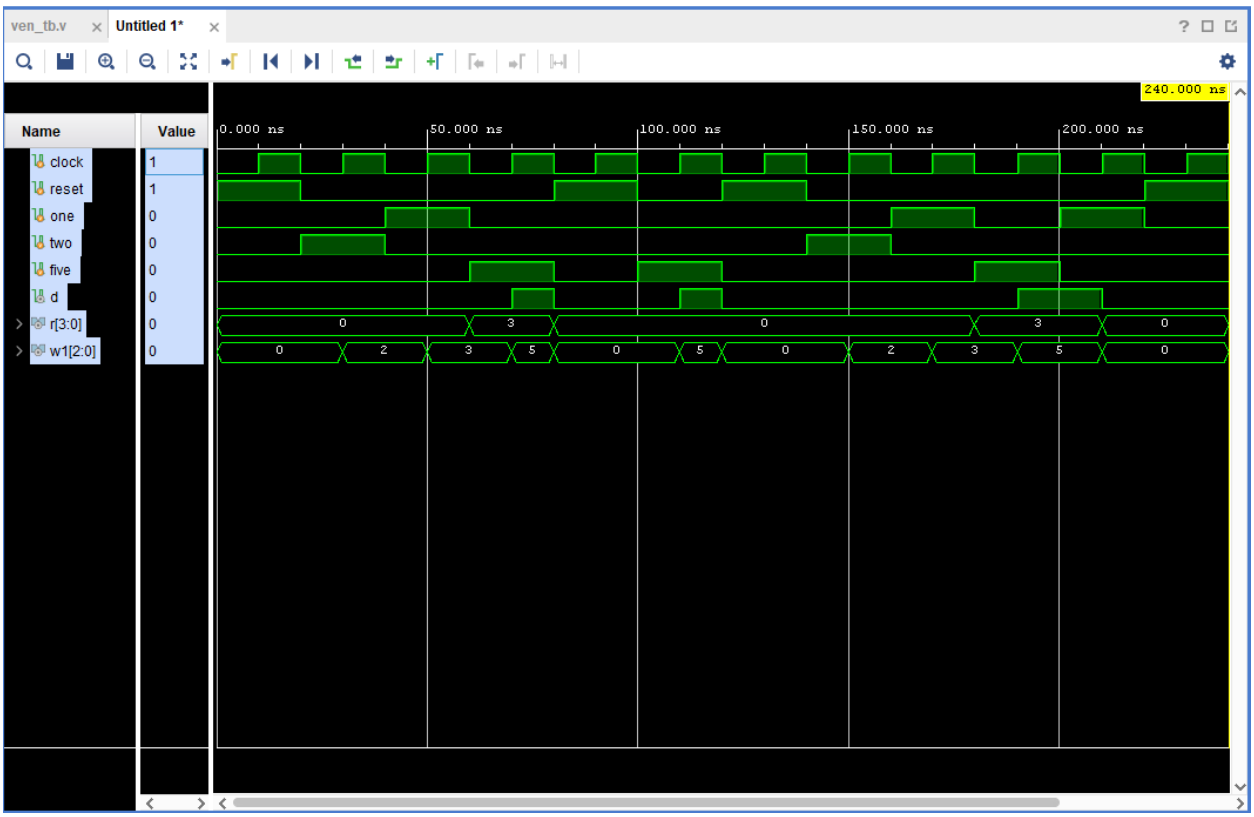
C:\Users\hoeel\Desktop\School\105L\ab2\Lab2P 4\Lab2P 4\srcs\sources_1\new\vsc.v

```
parameter a1 = 3'b111;
parameter a2 = 3'b112;
parameter a3 = 3'b113;

always@(posedge clock or posedge reset)
begin
    if (reset)
    begin
        current_state <= idle;
    end else
    begin
        current_state <= next_state;
    end
end

always@(current_state or one or two or five)
begin
    case(current_state)
    idle:
        if(one) next_state = a1;
        else if(two) next_state = a2;
        else if(five) next_state = a5;
        else next_state = idle;

    a1:
        if(one) next_state = a1;
        else if(two) next_state = a2;
        else if(five) next_state = a5;
        else next_state = a1;

    a2:
        if(one) next_state = a3;
        else if(two) next_state = a2;
        else if(five) next_state = a5;
        else next_state = a2;

    a3:
        if(one) next_state = a3;
        else if(two) next_state = a2;
        else if(five) next_state = a5;
        else next_state = a3;

    a4:
        if(one) next_state = a5;
        else if(two) next_state = a2;
        else if(five) next_state = a5;
        else next_state = a4;

    a5:
        if(one) next_state = a5;
        else if(two) next_state = a2;
        else if(five) next_state = a5;
        else next_state = a5;

    default:   next_state = idle;

    endcase
end

always@(current_state or one or two or five)
begin
    case (current_state)
    idle:
        if(one)
        begin
            d=1;
            e=1;
        end
        else if(two)
        begin
            d=1;
            e=1;
        end
        else if(five)
        begin
            d=1;
            e=1;
        end
        else
        begin
            d=1;
```

Project Summary | ven_tb.v

C:/Users/howel/Desktop/School/166/LAB2/Lab2P.4/Lab2P.4.srcs/sim_1/new/ven_tb.v

```
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module ven_tb();
reg clock,reset,one,two,five;
wire d;
wire [3:0]r;
wire [2:0]wl;

ven ven1(.clock(clock),.reset(reset),.one(one),.two(two),.five(five),.d(d),.r(r),.current_state(wl));

initial clock = 0;
always #10 clock = ~clock;

initial begin

        reset=1;    one=0;  two=0;  five=0;
    #20 reset=0;    one=0;  two=1;  five=0;
    #20 reset=0;    one=1;  two=0;  five=0;
    #20 reset=0;    one=0;  two=0;  five=1;
    #20 reset=1;    one=0;  two=0;  five=0;
    #20 reset=0;    one=0;  two=0;  five=1;
    #20 reset=1;    one=0;  two=0;  five=0;
    #20 reset=0;    one=0;  two=1;  five=0;
    #20 reset=0;    one=1;  two=0;  five=0;
    #20 reset=0;    one=0;  two=0;  five=1;
    #20 reset=0;    one=1;  two=0;  five=0;
    #20 reset=1;    one=0;  two=0;  five=0;
    #20 $stop;
end

initial $monitor($time, ". reset=%b, one=%b, two=%b, five=%b, d=%b, r=%b, wl=%b",reset,one,two,five,d,r,wl);

endmodule
```



Conclusion:

In conclusion we created a vending machine that counts credits and distributes a beverage. This vending machine is considered a finite state machine. Meaning the machine can be in any one state at a given time and essentially constantly runs ready for another input at any given time. States in the machine can be infinite and represent different functions of the machine.