King Saud University

College of Computer and Information Sciences

Department of Computer Engineering

CEN 317 : Logic Design Laboratory-II

Second Semester 2017-2018

Project

Whack A Mole Game

Authors

Name	ID	Section
Ahmad Abdo AlOmari	**435107401**	**20663**
Mubarak Salem Bin Madhi	**435107556**	**20663**

Instructor: Eng. Salman Ali Al-Eid

- # Basic Concepts

Whack A Mole Game that game plays by one player. The main idea of game is following the ON light by push buttons. We have four lights and four push buttons one for each light. That lights are randomly and fast light up, it will then eventually turn off, the player should be following the lights by using push buttons before the next light is show (every time only one light is ON). The player should be following the lights (one by one) nine times to win, otherwise will be lose.

we will design and implement Whack A Mole Game by using The Altera DE1 board (FPGA) written in Verilog. In that system We will using Finite State Machine FSM diagram tool in Active-HDL. Also we will using two 7-segments: the first one to show the number of true tries, and the second one to show the number of all tries.

Part I – Finite State Machine FSM :-

In this part, we will Design FSM for system with 19 states. that is shown in Figure 1.

We have four inputs (b_0, b_1, b_2, b_3) selecting by push buttons these are active LOW. The other input num that is take random values from (random module) see part IV, the reason to make it, we need play the game with random values. The input true comes from (counter module) see part III, to know how many times is correct, similar with the input tries comes from (counter module) see part III, but use it to know how many times is done. The input timer to know if time between each two lights is end or not, if timer is 1 mean the time is up and the next light should ON, but if timer is 0 mean we stay in state until push button or time is up. We have output resTimer using as reset to (timer module) see part V, The two outputs countTrue and countTries using to counting number of true time or counting number of all time by send signal to (counter module) see part III. The output numclk using as clock to (random module) see part IV, if numclk is one the FSM will assign a new random value to num.

The output green is green light will be light up, it will then eventually turn off if player win, we do this by keep moving between state 16 and state 17 unit reset the game. If player miss at least one light or select at least one wrong the output red will active, but should wait or finish all the rest lights to the output red is show. Also that FSM has reset input to start the game if reset is active high. Finally, we need a clock for a FSM that will be generates in part II.

- Now, draw FSM of state by using Active-HDL that is show in Figure 1.

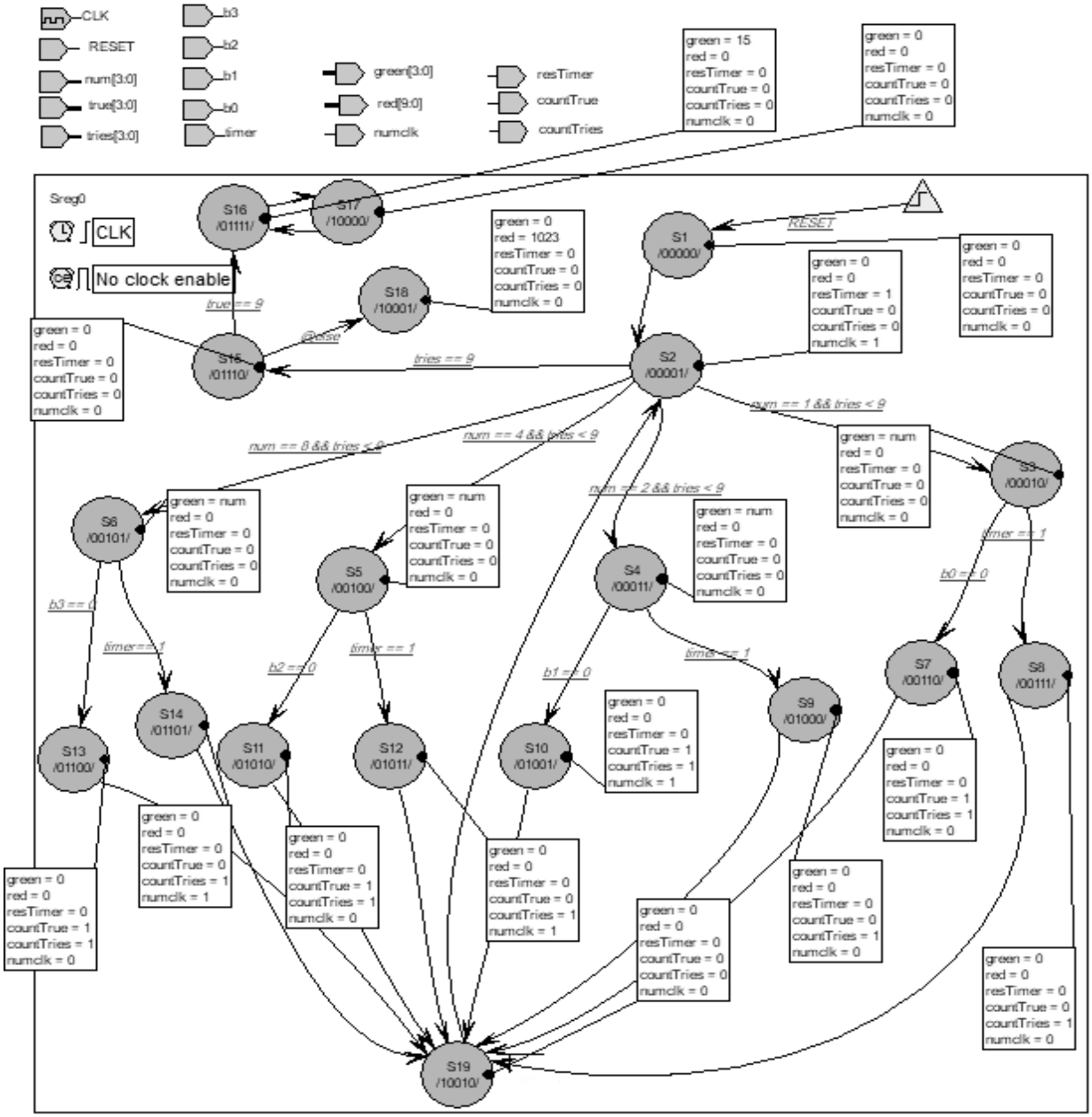

Figure 1: The FSM of a state

Part II – Clock Divider :-

The Altera DE1 board has main input clock with 50 MHz, we can't see the effect of clock because that is very fast. So, the clock divider that generates the 1Hz when its input clock is 50 MHz. This output is then used in the design of a real-time Clock.

In this part, we will Design the clock divider that generates the 1Hz when its input clock is 50 MHz. We need the clock is very fast so, the output of the clock divider is low for 7125000 clock periods and high for the second 7125000 clock period. The Verilog code of a clock divider is shown in Figure 3.

```
module clk2 (clock , Oclock);
    input clock ;
    output reg Oclock = 0 ;
    reg [25:0] counter = 0 ;
    always@(posedge clock) begin
        counter <= counter +1 ;
        if(counter == 7125000)begin
            counter <= 0 ;
            Oclock <= !Oclock ;
        end
    end
endmodule
```

Figure 3: The Verilog code for a clock divider

Part III – Counter :-

In this part, we will Design counter and using 7-segments to display the number. For design 7-segement we have one 4-bit input, and one 7-bit output. we will use two 7-segments: the first one to show the number of true tries, and the second one to show the number of all tries. The Verilog code of a 7-segment active LOW is shown in Figure 4.

We have one 1-bit input inp and one 4-bit outputs. The all initial values are zero. At every clock pulses increments by 1 up to 9. Also that counter has reset input to start counting from zero if reset is active high. Also we will use the clock divider that generated in part II as clock.

We will use this counter twice: the first one to count the number of true tries, and the second one to count the number of all tries.

```
module seven_segment4(in,out);
   input [3:0] in;
   output reg [6:0] out;

   always@(in)begin
       case(in)
           4'b0000: out = 7'b1000000;
           4'b0001: out = 7'b1111001;
           4'b0010: out = 7'b0100100;
           4'b0011: out = 7'b0110000;
           4'b0100: out = 7'b0011001;
           4'b0101: out = 7'b0010010;
           4'b0110: out = 7'b0000010;
           4'b0111: out = 7'b1111000;
           4'b1000: out = 7'b0000000;
           4'b1001: out = 7'b0010000;
           default out = 7'b0111111;
       endcase
   end
endmodule
```

Figure 4: The Verilog code for a 7-segment active LOW

- The Verilog code for counter is show in Figure 5.

```
module counter (clk ,reset , inp, out);
 input inp , reset , clk ;
 output reg[3:0] out ;
 initial out = 0 ;
     always@(posedge clk ,posedge reset )
         begin
             if(reset)
             out <= 0 ;
         else
             begin
                 if(inp == 1)
                     out <= out + 1 ;
                 else
                     out <= out ;
         end
         end
 endmodule
```

Figure 5: The Verilog code for a counter

Part IV – Random :-

The game should be show random values for each time the player will play.

In this part, we will Design the random module that send (at every clk) the random values to FSM to show on the lights. We know only and only one light should be active at every tries and we have four lights so, we have 0001, 0010, 0100 and 1000, we can shift left those numbers and make sure the outputs of that module will be one of these numbers. We use the output numclk from FSM as clk in this part. The Verilog code of a random is shown in Figure 6.

```
module random(clk, i, out);
   input clk ;
   output reg [2:0]i;
   output reg [3:0]out;

   initial  out = 1 ;

   initial i = 0;
   always@(posedge clk )
       begin

           for(i = 0 ; i < 4 ; i = i+1)
               begin
                   out <= out << 1 ;

                   if(out == 8)
                       out <= 1;
                   end
       end

endmodule
```

Figure 6: The Verilog code for a random

Part V – Timer :-

In this part, we will Design the timer that to know the time end before the next light is ON. The output resTimer from FSM use here as reset. Also we will use the clock divider that generated in part II as clock. The Verilog code of a timer is shown in Figure 7.

```
module timer(clk, reset , out );
    input clk , reset ;
    output reg out ;
    always@(posedge clk , posedge reset)
        begin
            if(reset)
                out = 0 ;
            else
                out = 1 ;
        end
    endmodule
```

Figure 7: The Verilog code for a timer

- Now, draw Whack A Mole Game block diagram using Active-HDL that is show in Figure 8, and Instantiation all modules in Whack A Mole Game module. The Verilog code for Whack A Mole Game is show in Figure 9.

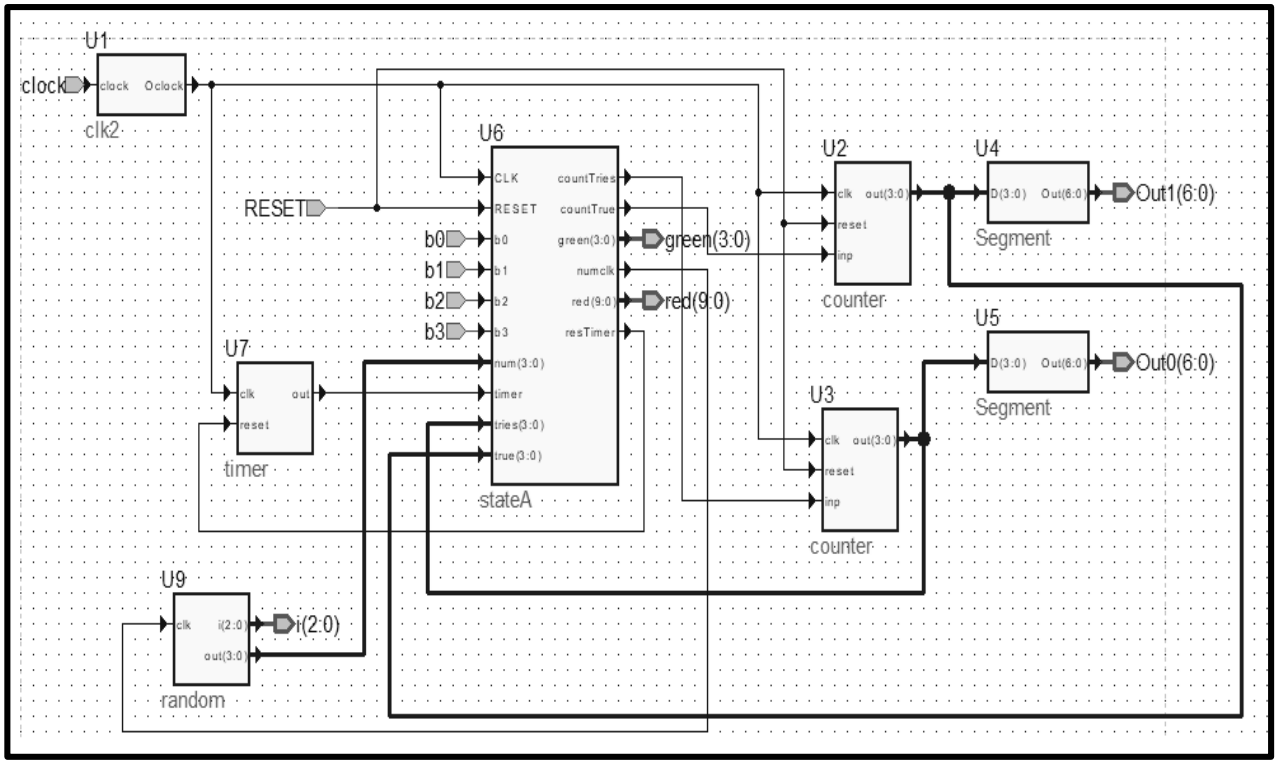

Figure 8: The Block diagram for Whack A Mole Game

```
`timescale 1ps / 1ps

module projectXXX (RESET,b0,b1,b2,b3,clock,Out0,Out1,
    green,i,red) ;

    input clock,RESET,b0,b1,b2,b3;
    wire clock,RESET,b0,b1,b2,b3;
    output [6:0] Out0,Out1;
    wire [6:0] Out0,Out1;
    output [3:0] green;
    wire [3:0] green;
    output [2:0] i;
    wire [2:0] i;
    output [9:0] red;
    wire [9:0] red;
    wire NET500,NET573,NET588,NET639,NET683,NET714;
    wire [3:0] BUS103,BUS212,BUS518;

    clk2 U1(.Oclock(NET714),.clock(clock));

    counter U2(.clk(NET714),.inp(NET500),.out(BUS518),.reset(RESET));
    counter U3(.clk(NET714),.inp(NET573),.out(BUS103),.reset(RESET));

    Segment U4(.D(BUS518),.Out(Out1));
    Segment U5(.D(BUS103),.Out(Out0));

    stateA U6(.CLK(NET714),.RESET(RESET),.b0(b0),.b1(b1),.b2(b2),
        .b3(b3),.countTries(NET573),.countTrue(NET500),.green(green),
        .num(BUS212),.numclk(NET639),.red(red),.resTimer(NET683),
        .timer(NET588),.tries(BUS103),.true(BUS518));

    timer U7(.clk(NET714),.out(NET588),.reset(NET683));

    random U9(.clk(NET639),.i(i),.out(BUS212));

endmodule
```

Figure 9: the Verilog code for Whack A Mole Game

- The Simulation for Whack A Mole Game. That is show in Figure 10.

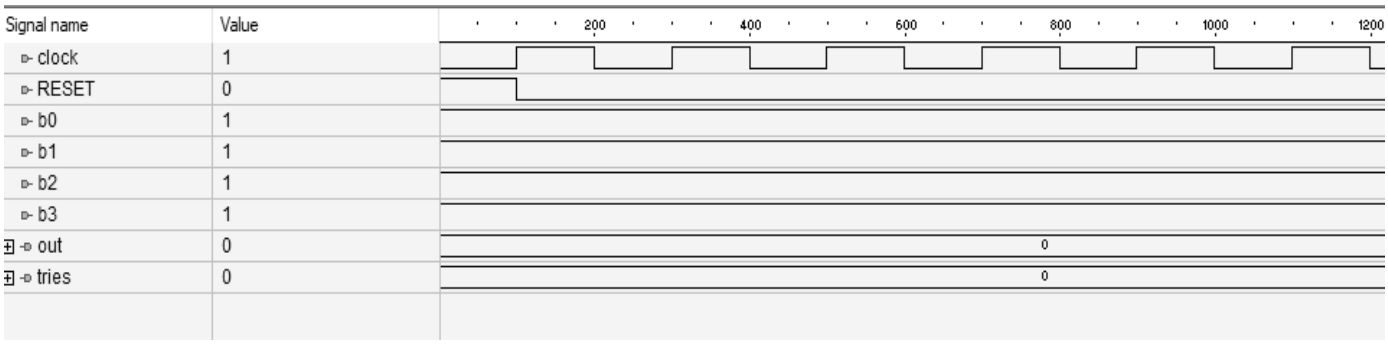

Figure 10: The Simulation for Whack A Mole Game

NOTE: If we want to cache the win situation in board without following the lights, we just push all push buttons at same time.

THE END