## ***EE311 VLSI LABORATORY***

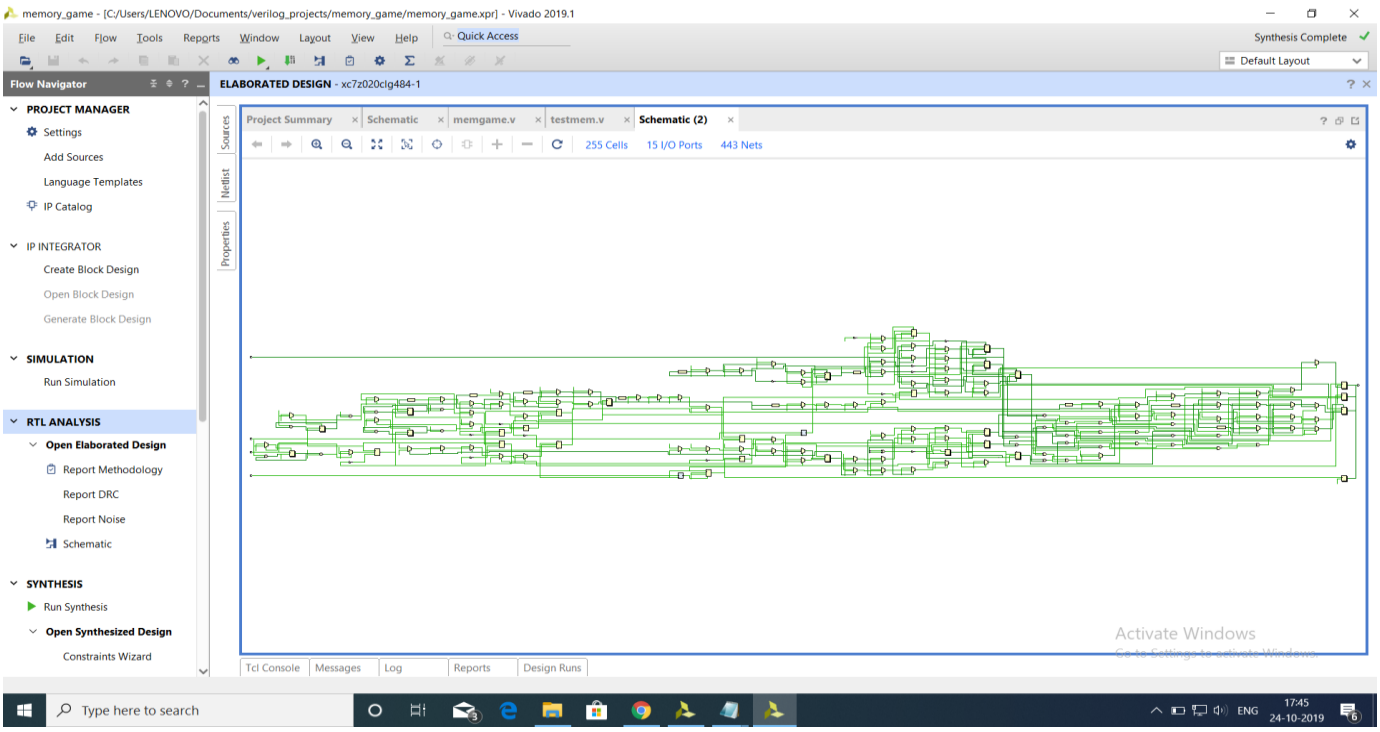
## ***ASSIGNMENT 6(Verilog)***


- **Name – Mayank Tantuway**
- **Roll No. 170102036**


**********

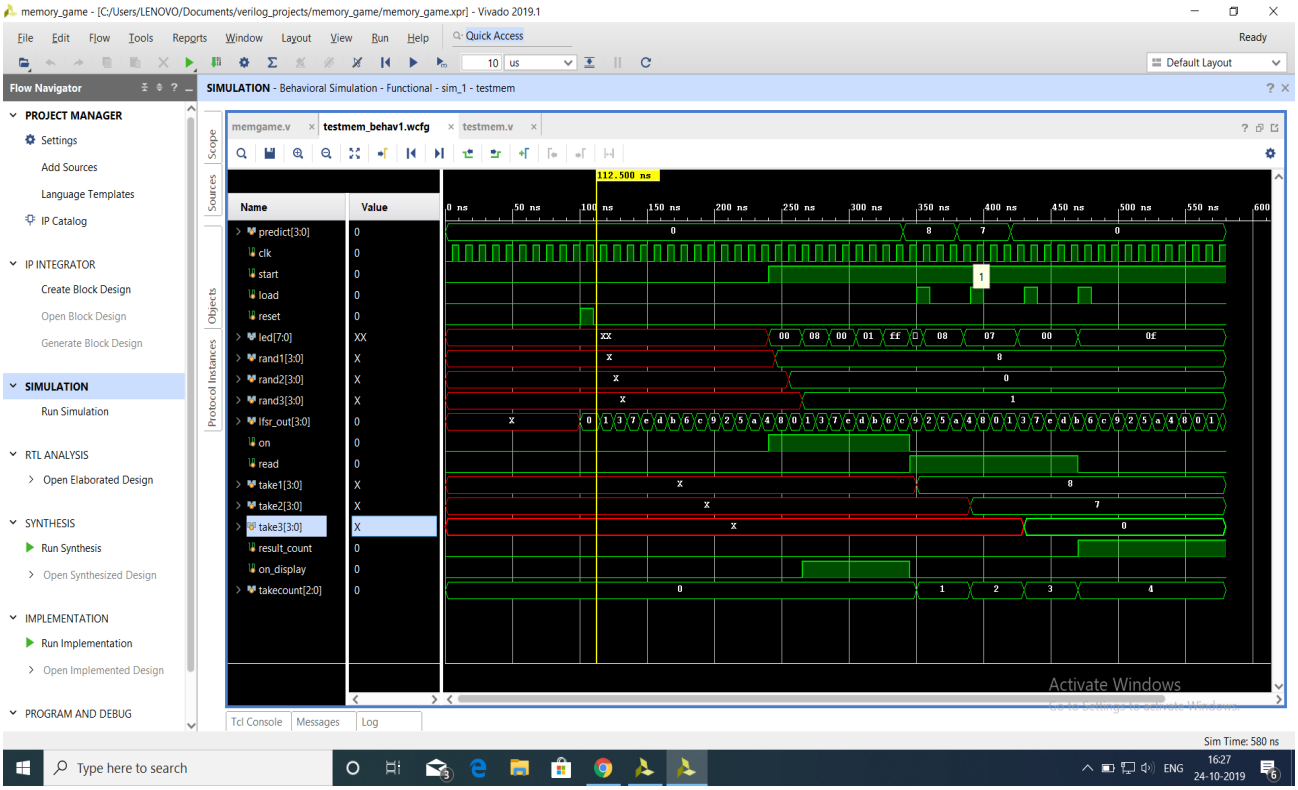
**Memory Game**: Students have to implement a game on Zedboard FPGA. The specs of the game are: i) After dumping the bit file to FPGA player will press a push button to start the game and to restart the game there should be one push button. ii) Then every 2 second interval 4-bit random number will glow (every time player plays the game random numbers should be different). iii) After showing the number all the 8 Leds should blink once to show that program is now ready to accept the input. iv) Player have to give the number using switches presenter on the FPGA board one by one. v) Finally, if all the inputs are correct with the correct sequence then board will show "WIN" by glowing all the Led (means 8'b1111_1111) & if all the inputs are correct but sequence is wrong then board will show "DRAW" by glowing alternate Leds (means 8'b1010_1010) & if entered numbers are wrong then no Leds will glow.

## ***SCHEMATICS***




*clear image available in zip folder along with the report

## SIMULATION OUTPUT



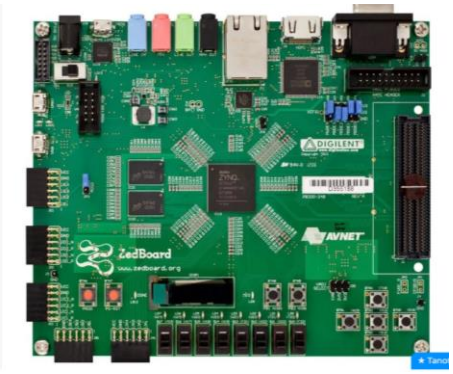

*clear image available in zip folder along with the report

PINS :-

START– Central Push Button

LOAD- Upper Push Button

INPUT – Right 4 Dip Switches




## VERILOG CODE

```
`timescale 1ns / 1ps
module memgame(
    input [3:0] predict,
    input clk,
    input startIN,
    input loadIN,

    output reg [7:0] led
    );

    reg clk_10ms =1'b0 ;
    reg clk_1s   =1'b0;
    reg clk_2s   =1'b0;
    reg [19:0]count_clk = 20'b0 ;
    reg [6:0]count_clk10ms = 7'b0;
    reg [2:0]count_reset=3'b0;
    reg reset=1'b0;
    reg [1:0]monitor_reset=2'b0;

    always@(posedge clk)
       begin
       if(count_reset<3'b111)
          begin
          count_reset=count_reset+1;
          end
       else if(monitor_reset<2'b10)
          begin
          count_reset=3'b0;
          reset=~reset;
```

```
      monitor_reset=monitor_reset+1;
      end
   else
      begin
      count_reset=3'b0;
      end
   end

//////////////////////////
/// always block to get a clock of 10ms from input 100MHZ clk
always @ (posedge clk )
   begin
   if (count_clk < 19'b1111010000100011111)   //  49,9999
      begin
         count_clk = count_clk +1 ;
      end
   else
      begin
         clk_10ms = ~clk_10ms ;
         count_clk = 20'b0;
      end
   end
always @ (posedge clk_10ms )
   begin
   if (count_clk10ms < 7'b1111111)   // clk_1s=1.28sec.
      begin
         count_clk10ms = count_clk10ms +1 ;
      end
   else
```

```
        begin
          clk_1s = ~clk_1s ;
          count_clk10ms = 7'b0;
        end
      end

  ////////////////////////////

//////////////////////
  always @(posedge clk_1s)
    begin
    clk_2s = ~clk_2s;

    end
////
  wire startOUT;
  wire load;
  debounce FIRST ( clk_10ms , startIN  , startOUT);
  debounce SECOND ( clk_10ms , loadIN  , load);

  reg start=1'b0;
  always@(posedge clk_10ms )
    if(startOUT)start=1'b1;

///
  //wire [3:0]rand = 4'b0000;
  reg on = 1'b0;
  reg result_count =1'b0;
  reg on_display = 1'b0;
```

```
 reg read =1'b0;
 reg clk_lfsr = 1'b0;
 reg [2:0]count = 3'b000;



always @(posedge start)
 begin
 on <= ~on ;
 led <= 8'b0000_0000;
 end



 always@*       // when on run at 2sec else run fast
 if(on)
    clk_lfsr=clk_2s;
  else
    begin
    clk_lfsr=clk_10ms;
    end

//lfsr i1 (rand,clk,reset);
///////// lfsr //////


reg [3:0] lfsr_out;
reg [3:0]rand1;
reg [3:0]rand2;
```

```
  reg [3:0]rand3;



  wire feedback;

 assign feedback = ~(lfsr_out[3] ^ lfsr_out[2]);
  reg [1:0]i=2'b0;
always @(posedge clk_10ms, posedge reset)
 begin
  if (reset)
    lfsr_out = 4'b0;
  else
     begin
     lfsr_out = {lfsr_out[2:0],feedback};
     if(on)
        begin
           if(i==2'b00)
           begin
              rand1=lfsr_out;
              i=i+1;
           end
           else if(i==2'b01)
           begin
              rand2=lfsr_out;
              i=i+1;
           end
           else if(i==2'b10)
           begin
              rand3=lfsr_out;
```

```
            i=i+1;
            on_display=~on_display;
          end
        end
    end
end



////////////

 always @(posedge clk_2s)
 begin
    if(on_display)
    begin
       if(count==3'b000)
        begin
        led[0] <= rand1[0];
        led[1] <= rand1[1];
        led[2] <= rand1[2];
        led[3] <= rand1[3];
        led[4] <= 1'b0;
        led[5] <= 1'b0;
        led[6] <= 1'b0;
        led[7] <= 1'b0;

        count = count +1;
        end
        else if(count==3'b001)
```

```
begin
led[0] <= rand2[0];
led[1] <= rand2[1];
led[2] <= rand2[2];
led[3] <= rand2[3];
led[4] <= 1'b0;
led[5] <= 1'b0;
led[6] <= 1'b0;
led[7] <= 1'b0;

count = count +1;
end
else if(count==3'b010)
begin
led[0] <= rand3[0];
led[1] <= rand3[1];
led[2] <= rand3[2];
led[3] <= rand3[3];
led[4] <= 1'b0;
led[5] <= 1'b0;
led[6] <= 1'b0;
led[7] <= 1'b0;

count = count +1;
end
else if(count == 3'b011)
begin
led=8'b1111_1111;
count = count +1;
```

```
            end
           else
            begin
            led = 8'b0000_0000;
            on = ~on;
            on_display = ~on_display;
            read = ~read;
            end
        end
    end

  always @(posedge clk_10ms)
    begin
       if(read)
          begin

                led[0] <= predict[0];
                led[1] <= predict[1];
                led[2] <= predict[2];
                led[3] <= predict[3];
                led[4] <= 1'b0;
                led[5] <= 1'b0;
                led[6] <= 1'b0;
                led[7] <= 1'b0;
          end

    end
reg [3:0]take1;
reg [3:0]take2;
```

```
reg [3:0]take3;
reg [2:0]takecount=3'b000;
always @(posedge read, posedge load)
   begin
      if(read)
         begin
         if(load)
            begin
             if(takecount==3'b000)
                begin
                take1=predict;
                takecount=takecount+1;
                end
             else if(takecount==3'b001)
                begin
                take2=predict;
                takecount=takecount+1;
                end
            else if(takecount==3'b010)
                begin
                take3=predict;
                takecount=takecount+1;



                end
            else if(takecount==3'b011)
                begin
                result_count=~result_count;
                takecount=takecount+1;
```

```
            read=~read;
            end

          end
        end
    end
always @(posedge clk_1s,posedge load)
    begin
      if(result_count)
      begin
        if(load)
          begin
      ////////////////////result
          if((rand1==take1)&&(rand2==take2)&&(rand3==take3))
          led<=8'b1111_1111;
          else if ((rand1==take2)&&(rand2==take1)&&(rand3==take3))
          led<=8'b1010_1010;
          else if ((rand1==take3)&&(rand2==take2)&&(rand3==take1))
          led<=8'b1010_1010;
          else if ((rand1==take1)&&(rand2==take3)&&(rand3==take2))
          led<=8'b1010_1010;
          else if ((rand1==take2)&&(rand2==take3)&&(rand3==take1))
          led<=8'b1010_1010;
          else if ((rand1==take3)&&(rand2==take1)&&(rand3==take2))
          led<=8'b1010_1010;
          else
          led<=8'b0000_1111;

            end
```

```
      end
   end
endmodule




module debounce (input clk ,
       input   PBI ,              // Push Butoon Input
       output  reg PBO  );        // Push Butoon Output

   reg  [9:0]count ;
   always @ (posedge clk or negedge PBI )
   begin
      if (PBI==0)
      begin
         count <= 10'b0 ;
      end
      else
      begin
         count = count +1 ;
      end
   end



      always @ *
   begin
      if ( count >  10'b0000000011 )    ////  30ms check
      begin     PBO <= 1'b1 ; end
```

```
      else
      begin  PBO <= 1'b0 ;end
   end
Endmodule
```

***TEST BENCH***

```
module testmem;

   reg [3:0] predict;
   reg clk;
   reg startIN;
   reg loadIN;
   reg reset;
   wire [7:0] led;

   memgame DUT ( .predict(predict), .clk(clk), .startIN(startIN), .loadIN(loadIN), .led(led));

   initial
      begin
         $monitor ($time, " predict=%d, clk=%b, startIN=%b, loadIN=%b, led=%d", predict, clk, startIN,
loadIN, led);
```

```
    $dumpfile("test.vcd");
    $dumpvars(0,DUT);
    predict = 4'b0000;
    clk = 1'b0;
    startIN = 1'b0;
    loadIN = 1'b0;



    #130 startIN = 1'b1;
    #10000 predict = 4'b1000;
    #10 loadIN =1'b1;
    #10 loadIN =1'b0;
    #20 predict = 4'b0111;
    #10 loadIN =1'b1;
    #10 loadIN =1'b0;
    #20 predict = 4'b0000;
    #10 loadIN =1'b1;
    #10 loadIN =1'b0;
    #2000
    #10 loadIN =1'b1;
    #10 loadIN =1'b0;
    #100000 $finish;



    end

always #5 clk = ~clk;
```

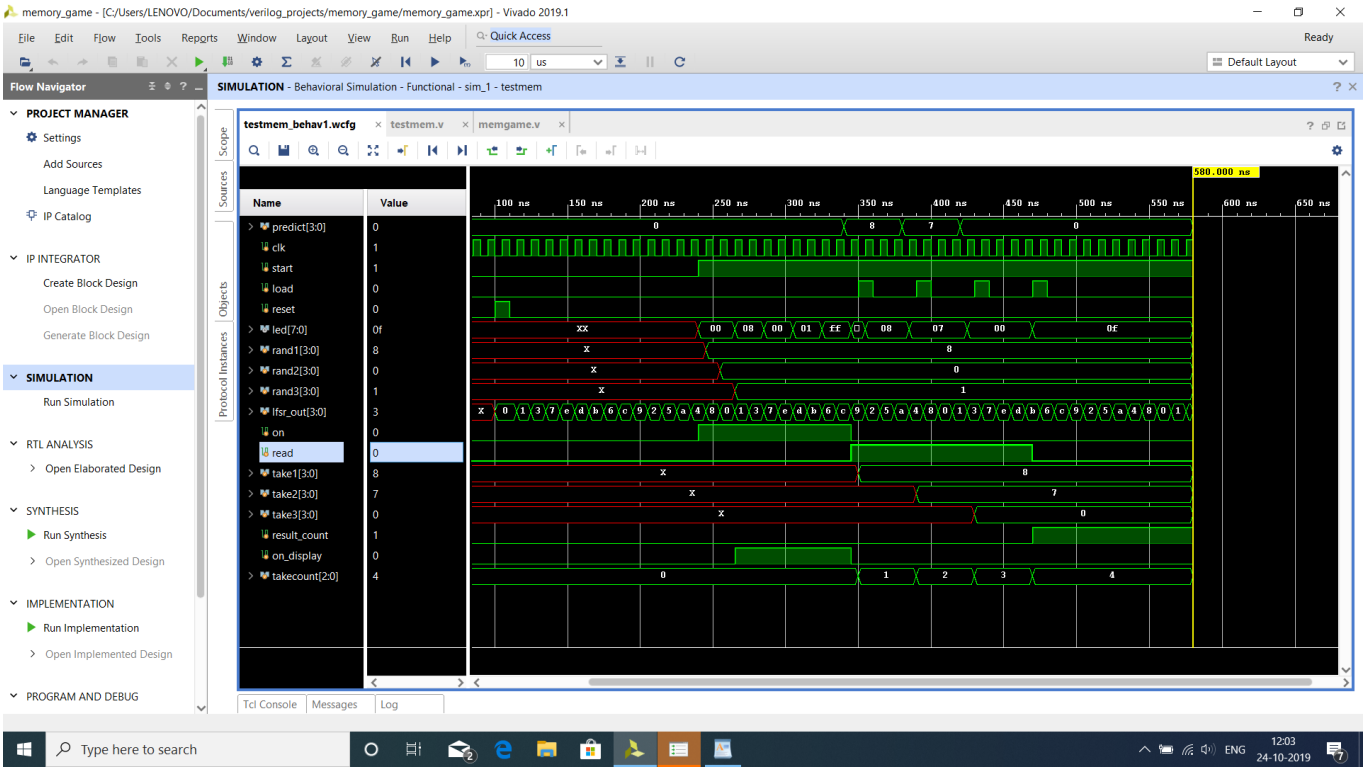
Endmodule

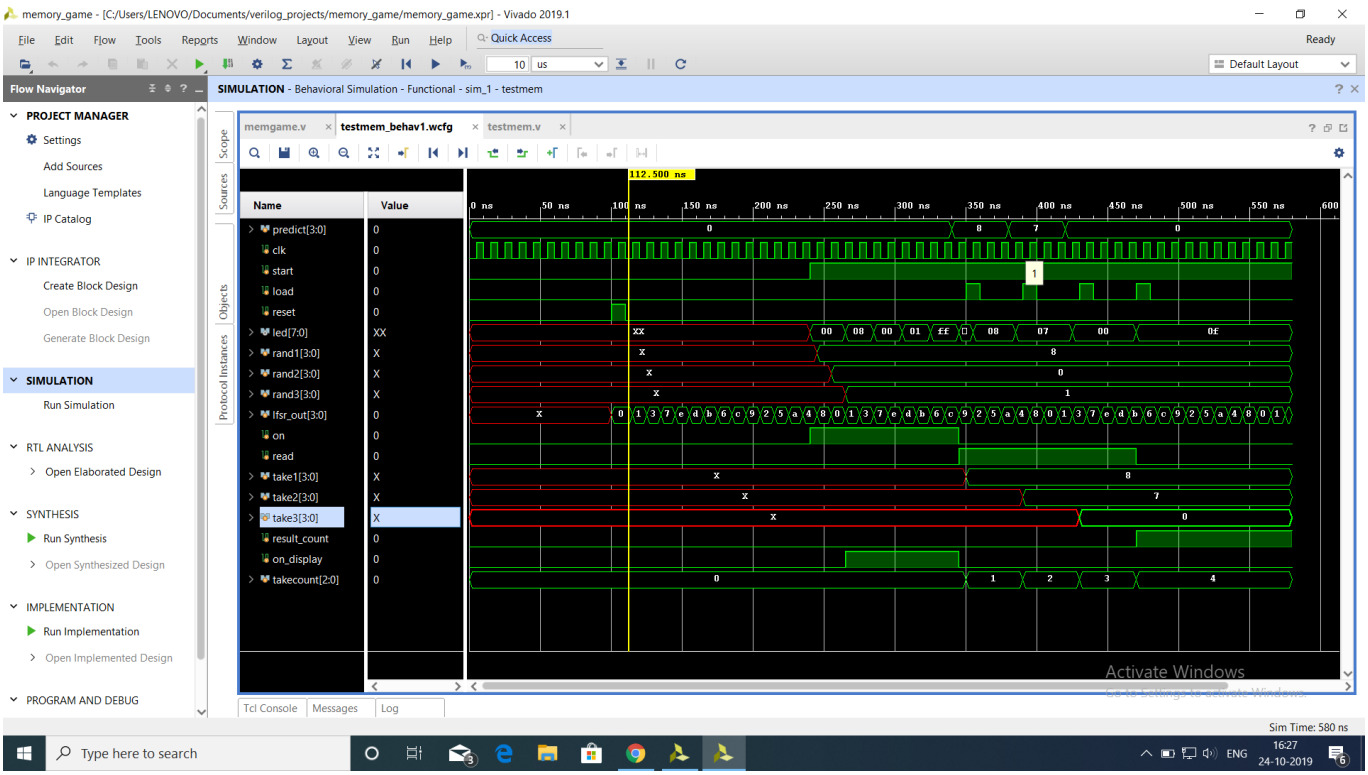
CONSTRAINT file (.XDC) (I/O PIN CONNECTION )

```
set_property IOSTANDARD LVCMOS33 [get_ports {predict[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {predict[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {predict[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {predict[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports loadIN]
set_property IOSTANDARD LVCMOS33 [get_ports startIN]
set_property PACKAGE_PIN Y9 [get_ports clk]
set_property PACKAGE_PIN P16 [get_ports startIN]
```

```
set_property PACKAGE_PIN T18 [get_ports loadIN]
set_property PACKAGE_PIN F22 [get_ports {predict[0]}]
set_property PACKAGE_PIN G22 [get_ports {predict[1]}]
set_property PACKAGE_PIN H22 [get_ports {predict[2]}]
set_property PACKAGE_PIN F21 [get_ports {predict[3]}]
set_property PACKAGE_PIN U14 [get_ports {led[7]}]
set_property PACKAGE_PIN U19 [get_ports {led[6]}]
set_property PACKAGE_PIN W22 [get_ports {led[5]}]
set_property PACKAGE_PIN V22 [get_ports {led[4]}]
set_property PACKAGE_PIN U21 [get_ports {led[3]}]
set_property PACKAGE_PIN U22 [get_ports {led[2]}]
set_property PACKAGE_PIN T21 [get_ports {led[1]}]
set_property PACKAGE_PIN T22 [get_ports {led[0]}]
```

***SIMULATION OUTPUTS***


memory_game - [C:/Users/LENOVO/Documents/verilog_projects/memory_game/memory_game.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
SIMULATION - Behavioral Simulation - Functional - sim_1 - testmem
testmem_behav1.wcfg
testmem.v
memgame.v
Scope
Sources
Objects
Protocol Instances
Name
Value
predict[3:0]
clk
start
load
reset
led[7:0]
rand1[3:0]
rand2[3:0]
rand3[3:0]
lfsr_out[3:0]
on
read
take1[3:0]
take2[3:0]
take3[3:0]
result_count
on_display
takecount[2:0]
580.000 ns
Tcl Console
Messages
Log
Type here to search
ENG
12:03
24-10-2019


memory_game - [C:/Users/LENOVO/Documents/verilog_projects/memory_game/memory_game.xpr] - Vivado 2019.1
File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
SIMULATION - Behavioral Simulation - Functional - sim_1 - testmem
memgame.v
testmem_behav1.wcfg
testmem.v
Scope
Sources
Objects
Protocol Instances
112.500 ns
Name
Value
predict[3:0]
clk
start
load
reset
led[7:0]
rand1[3:0]
rand2[3:0]
rand3[3:0]
lfsr_out[3:0]
on
read
take1[3:0]
take2[3:0]
take3[3:0]
result_count
on_display
takecount[2:0]
Tcl Console
Messages
Log
Sim Time: 580 ns
Type here to search
ENG
16:27
24-10-2019