ELEX 7760: DIGITAL SYSTEM DESIGN

WHAC-A-MOLE

Project Report

Richard Lim, Kyle Yoon

4-13-2018

ABSTRACT

Whac – A – Mole is a simple game that everyone can play; you have to hit as many moles as you can with a hammer. We will implement this game as our project for this course, ELEX 7660. For this project we will use an Intel DE0 Nano FPGA board. The moles will appear on the display and the player hit the right keypad to score. The game will have multiple levels; each round will have 30 seconds time limit. Through this project we are expecting to use our knowledge to check what we have learnt from the course and to relieve some stress by playing this game.

TABLE OF CONTENTS

INTRODUCTIOIN

After watching the demo of SIMON game during the lecture, we decided to implement something like that. We did some research on other projects and one of them attracted us, the Whac-a-Mole game. We chose it not only it looks fun but also thought of using our knowledge about FPGA with system Verilog structures we learn through the course. The operation we expected was a player catch the moles, randomly generated on the led matrix, using the keypad. There are 4 levels, for each difficulty the game will start with different boarder color; easy-red, normal-pink, hard-light blue, extreme-yellow. The remaining time is showing on the right side of the led matrix. Once the time is over you can select difficulty button to start over.

OBJECTIVE

The objectives of this project are to:

- Implement FPGA SPI to send data to led matrix
- Connect keypad to function as a hammer and reset
- Interface clock signals, and voltage level to meet the specification for led matrix
- Get familiar with FPGA running by Verilog
- Connect touch screen to FPGA

BACKGROUNDS

The most fundamental and important part of the project was timing issues. Using main 50 MHz clock to generate required clock frequencies for led data sending, mole generating, keypad indicating, and every sequence we needed in specific timing. For keypad we used lab2 matrix keypad decoder and adjust some code to fit our purpose of mole detector and level selector. To light up the led, 32-bit array was required to store data and sent to the buffer. Once data is sent to device using SPI system the output enable is activated to light it up. Time count and score count is also set to show appropriate values decoded by each number element.

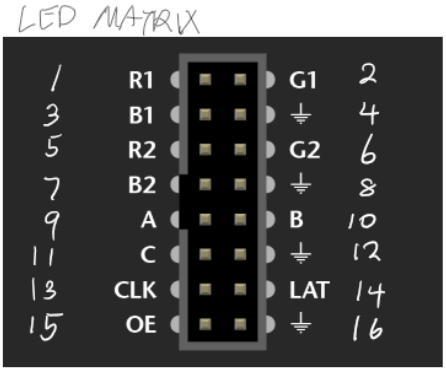

Figure 1. 16X32 LED matrix pin assignment

For the hardware part, there are 16 pins assigned to the led matrix, as of Figure 1, row select (a, b, c), column select (r1, g1, b1, r2, g2, b2), latch, clock, OE, GNDs. Once row and column selected, latch occurs to send data to the buffer than OE activated to turn on the leds where activated. The led matrix need 5v logic, so step up from 3.3V to 5V was implemented with boost converter. However, try to avoid high voltage, the source voltage of 5V is coming out of FPGA board.

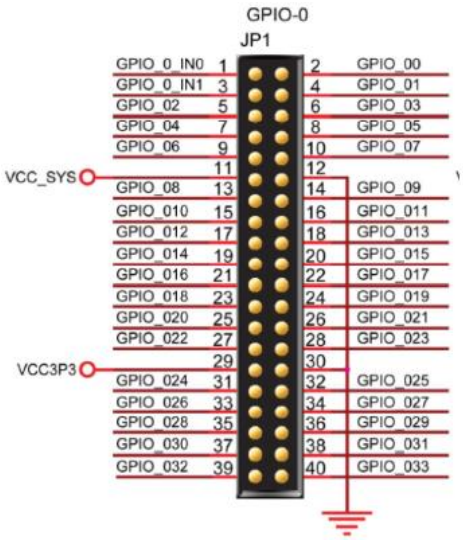

Figure 2. GPIO output from FPGA

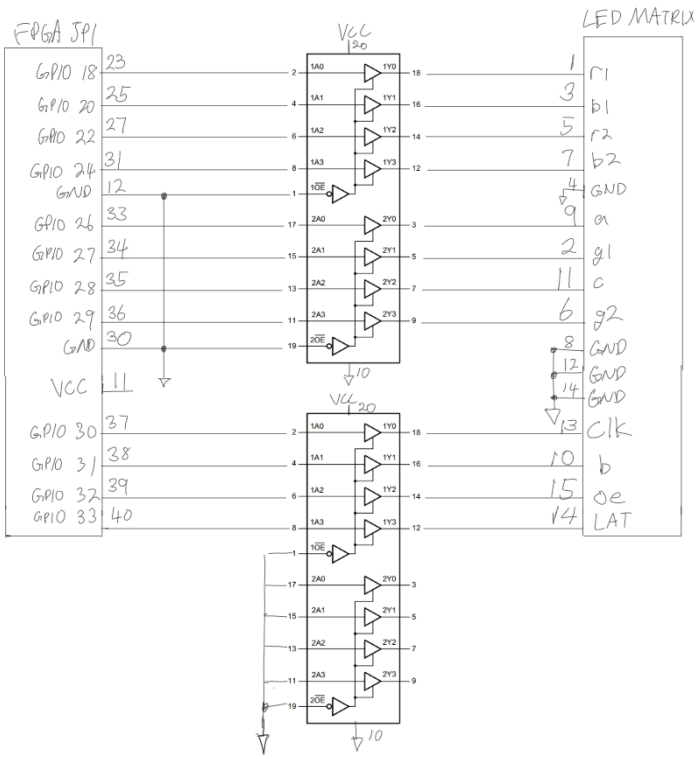

Figure 3. Boost converter to feed 5V

The output pin outs are assigned as Figure 2, and Figure 3. We used 74HC244 for 8bit octal buffer using wires and a breadboard for connection.

CHANGES TO PROPOSAL

There were some points that we changed our first decision. First one is the speaker, we have decided not to use it due to its complexity. And we tried to implement touch screen, but we had hard time on ADC conversion and time shortage lead us not to include it on this project.

The results after adjusting the design look like the following Figures.

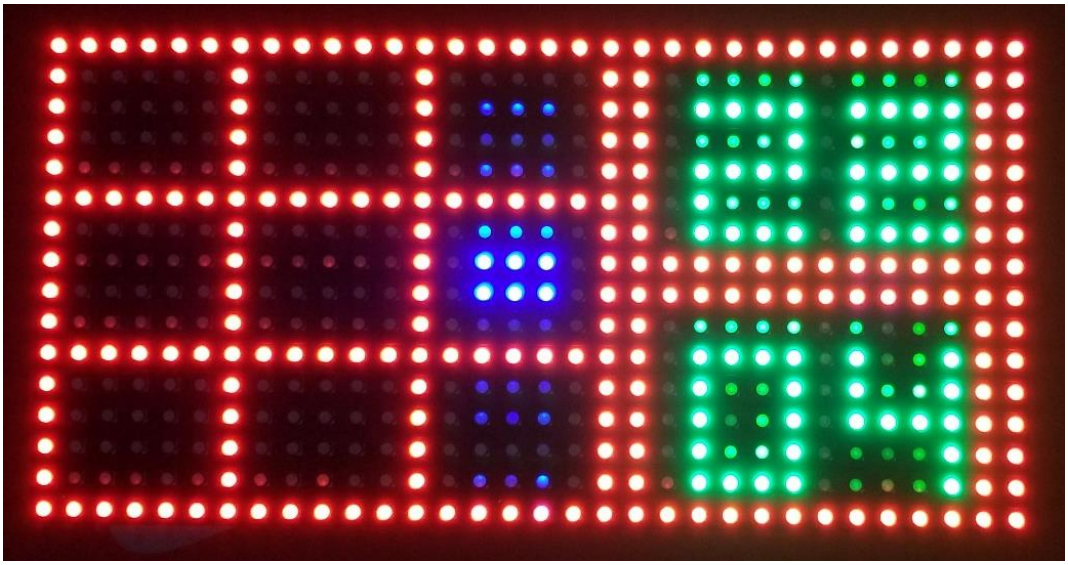

Figure 4. 16×32 LED matrix, 30 second counter at top right, score at bottom right

Figure 5. Keypad, numbers for mole, 4th column for level, 4th row for reset

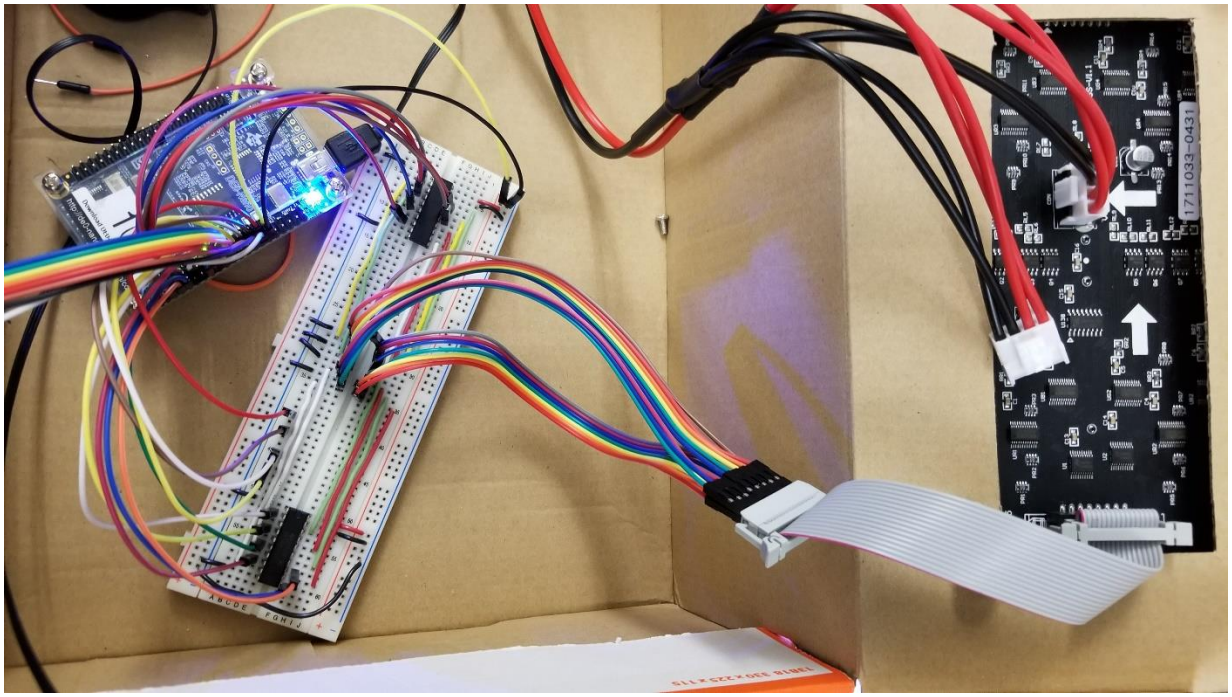

Figure 6. Wiring between FPGA and LED matrix through buffer

Flow Status	Successful - Wed Apr 11 19:10:52 2018
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	proj_01
Top-level Entity Name	lab2
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	775 / 22,320 (3 %)
Total registers	125
Total pins	31 / 154 (20 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)

Figure 7. Flow summery of the project

DESIGN APPROACH

Game clock Module

This module is the 30 second timer of the game. It takes the 50 MHz clock from the FPGA and converts it to a 1 Hz clock. After 30 converted clock cycles, it stops the game by giving start logic 0.

Mole module

This is the game module of the project; it decides if the user input is matched with the random number generated by a linear-feedback shift register method and controls the difficulties of the game.

Generating a random number:

This is done by a linear-feedback shift register method, a shift register whose input bit is a linear function of its previous state. It only generates once when the game clock is low and the game clock counter equals to game clock counter - 1.

Matching the user input with the generated random number:

When the game clock is high, it checks if the user input is matched with the random number. If it hits, it sets a flag to prevent multiple hits on one random number. The flag gets reset when the game clock is low.

Controlling difficulties of the game:

There are four different difficulties in this game and are controlled by the a, b, c and d button on the keypad; those buttons reset the game when pressed, so that the user can start the game fresh with the new level. This is simply done by dividing the FPGA's 50 MHz by a different number; smaller the number you divide, the faster the random number is generated.

LED control module

To turn on the LED Matrix, you need to set OE high and LAT low; OE is active low, and LAT is active high. Then you must clock your 32-bit data at maximum of 25 MHz into the serial data input pins, A, B, and C. After you have clocked your data, you must turn off the clock and set latch with set up time of 100 ns and hold time for 20 ns. Then you set LAT low and set OE pin low for maximum of 150 ns, hold it for maximum of 200 ns and set high. See the data sheet in the reference for more details.

colseq and kpdecode module

We recycled these two modules from ELEX7660 lab2. Basically, colseq has an algorithm that outputs a column that the user has pressed and the kpdecode takes that and the selected row and outputs a number that the user has pressed.

CONCLUSION

We implemented the game, Whac-A-Mole by a keypad, a DEo Nano FPGA and a 32 x 16 RGB display. The purpose of the game was to destress by whacking the moles, but we got more stress making the game. If we had more time, we could have done more work to improve this project. For example, instead of the keypad, we could have used a resistive touchpad and used a random seed for generating the moles. Overall, this project was successfully implemented the game Whac-A-Mole.

APPENDIX

Top level module

```
module whac (output logic [3:0] kpc,  // column select, active-low
                    input logic  reset_n, CLOCK_50,
                    input logic [3:0] kpr,
                    output logic clock,
                    output logic lat,
                    output logic a,b,c,r1,r2,b1, b2, g1, g2, oe, sclk,
                    output logic unsigned [3:0] rand_num,
                    output logic [3:0] hit_count );
   logic clk, start ;
   logic kphit ;
   logic [4:0] num ;
   logic hit ;
   logic [3:0] hit_num ;
   logic [7:0] gametimer ;
   logic [1:0] level ;
   assign clock = CLOCK_50;
   gameclock gameclock_0 ( .clock, .reset_n, .gametimer, .num, .start );
   colseq colseq_0 ( .kpr, .clk, .reset_n, .kpc);
   kpdecode kpdecode_0 ( .kpr, .kpc, .kphit, .num );
   led_ctrl led_ctrl_0 (.gametimer, .level, .start, .clock, .lat, .sclk, .r1,
                        .r2, .b1, .b2, .g1, .g2, .a, .b, .c, .reset_n, .oe,
                        .rand_num, .hit, .hit_count, .num ) ;
   mole mole_0 (.num, .clock, .reset_n, .hit, .rand_num, .hit_num, .hit_count, .level) ;
endmodule
```

Game clock Module

```
module gameclock
#( N = 50_000_000 )
  (input logic clock, reset_n,
   input logic [4:0] num,
   output logic [7:0] gametimer,
   output logic start);
   logic [7:0] gametimer_next ;
   logic [31:0] n, n_next ;
   logic start_next ;
   always_comb begin
   n_next = n ;
   gametimer_next = gametimer ;
   start_next = start ;
      if (!reset_n || num == 4'ha|| num == 4'hb|| num == 4'hc|| num == 4'hd) begin
         gametimer_next = 30 ;
         n_next = N - 1 ;
         start_next = 1 ;
      end
      else begin
         if ( n )
            n_next = n - 1 ;
         else
            n_next = N - 1 ;
         if ( n == N - N/2 )
            gametimer_next = gametimer - 1 ;
         if ( !gametimer && !n )
            start_next = 0 ;
      end
   end
   always_ff @( posedge clock ) begin
      n <= n_next ;
      gametimer <= gametimer_next;
      start <= start_next ;
   end
endmodule
```

Column Select Module

```
module colseq ( output logic [3:0] kpc,
                input logic [3:0] kpr,
                input logic clk, reset_n );
   logic [3:0] kpc_next ;
   always_comb begin
      kpc_next = kpc ;
      if ( !reset_n )
         kpc_next = 4'b0111;
      else begin
         if ( kpr == 4'b1111 ) begin
            case ( kpc )
               4'b1011: kpc_next = 4'b0111;
               4'b1101: kpc_next = 4'b1011;
               4'b1110: kpc_next = 4'b1101;
               4'b0111: kpc_next = 4'b1110;
               default: kpc_next = 4'b0111;
            endcase
         end
         else if ( kpr == 4'b0111)
            kpc_next = kpc;
         else if ( kpr == 4'b1011)
            kpc_next = kpc;
         else if ( kpr == 4'b1101)
            kpc_next = kpc;
         else if ( kpr == 4'b1110)
            kpc_next = kpc;
         else
            kpc_next = 4'b0111;
      end
   end
   always_ff@(posedge clk)
      kpc <= kpc_next;
endmodule
```

Decode Module

```
module kpdecode (input logic [3:0] kpr, kpc, output logic kphit, output logic [4:0] num);
  always_comb begin
    if ((kpr == 4'b0111) && (kpc == 4'b0111)) begin
       kphit = 1;
       num = 4'd1;
    end else if (kpr == 4'b0111 && (kpc == 4'b1011)) begin
       kphit = 1;
       num = 4'd2;
    end else if (kpr == 4'b0111 && (kpc == 4'b1101)) begin
       kphit = 1;
       num = 4'd3;
    end else if (kpr == 4'b0111 && (kpc == 4'b1110)) begin
       kphit = 1;
       num = 4'ha;
    end else if ((kpr ==4'b1011) && (kpc == 4'b0111)) begin
       kphit = 1;
       num = 4'd4;
    end else if ((kpr ==4'b1011) && (kpc == 4'b1011)) begin
       kphit = 1;
       num = 4'd5;
    end else if ((kpr ==4'b1011) && (kpc == 4'b1101)) begin
       kphit = 1;
       num = 4'd6;
    end else if ((kpr ==4'b1011) && (kpc ==4'b1110)) begin
       kphit = 1;
       num = 4'hb;
    end else if ((kpr ==4'b1101) && (kpc == 4'b0111)) begin
       kphit = 1;
       num = 4'd7;
    end else if ((kpr ==4'b1101) && (kpc == 4'b1011)) begin
       kphit = 1;
       num = 4'd8;
    end else if ((kpr ==4'b1101) && (kpc == 4'b1101)) begin
       kphit = 1;
       num = 4'd9;
    end else if ((kpr ==4'b1101) && (kpc ==4'b1110)) begin
       kphit = 1;
       num = 4'hc;
    end else if ((kpr ==4'b1110) && (kpc == 4'b0111)) begin
       kphit = 1;
       num = 4'he; // C
    end else if ((kpr ==4'b1110) && (kpc == 4'b1011)) begin
       kphit = 1;
       num = 4'd0;
    end else if ((kpr ==4'b1110) && (kpc == 4'b1101)) begin
       kphit = 1;
       num = 5'h11111;
    end else if ((kpr ==4'b1110) && (kpc ==4'b1110)) begin
       kphit = 1;
       num = 4'hd;
    end
    else begin
       num = 5'b11111 ;
       kphit = 0 ;
    end
  end
endmodule
```

Game Module

```
module mole ( input logic [4:0] num,
   input logic clock, reset_n, start,
   output logic hit,
   output logic unsigned [3:0] rand_num,
   output logic [3:0] hit_num,
   output logic [3:0] hit_count,
   output logic [1:0] level) ;
   logic [31:0] moletime, moletime_next ;
   logic sclk, sclk_next, start_next;
   logic [31:0] n, n_next ;
   logic hit_next ;
   logic [19:0] i, i_next ;
   logic unsigned [3:0] rand_num_next ;
   logic [3:0] state, state_next ;
   logic [3:0] hit_num_next ;
   logic [3:0] hit_count_next ;
   logic [1:0] level_next ;
   logic feedback, feedback_next;
    always_comb begin
      sclk_next = sclk ;
      n_next = n ;
      i_next = i ;
      rand_num_next = rand_num ;
      hit_next = hit ;
      state_next = state ;
      hit_num_next = hit_num ;
      hit_count_next = hit_count ;
      feedback_next = feedback ;
      moletime_next = moletime ;
      level_next = level ;
      if (!reset_n ) begin
         level_next = 0 ;
         rand_num_next = '0 ;
         n_next = 50_000_000 - 1 ;
         state_next = 0 ;
         hit_next = 0 ;
         hit_count_next = 0;
         start_next = 1 ;
         moletime_next = 50_000_000 ;
      end
      else if ( num == 4'ha ) begin
         level_next = 0 ;
         rand_num_next = '0 ;
         n_next = 50_000_000 - 1 ;
         state_next = 0 ;
         hit_next = 0 ;
         hit_count_next = 0;
         start_next = 1 ;
         moletime_next = 50_000_000 ;
      end
      else if ( num == 4'hb ) begin
         level_next = 1 ;
         rand_num_next = 4'b0010 ;
         n_next = 40_000_000 - 1 ;
         state_next = 0 ;
         hit_next = 0 ;
         hit_count_next = 0;
         start_next = 1 ;
         moletime_next = 40_000_000 ;
      end
      else if ( num == 4'hc ) begin
         level_next = 2 ;
         rand_num_next = 4'b1100 ;
         n_next = 30_000_000 - 1 ;
         state_next = 0 ;
         hit_next = 0 ;
         hit_count_next = 0 ;
         start_next = 1 ;
```

```
            moletime_next = 30_000_000 ;
        end
        else if ( num == 4'hd ) begin
            level_next = 3 ;
            rand_num_next = 4'b1010 ;
            n_next = 20_000_000 - 1 ;
            state_next = 0 ;
            hit_next = 0 ;
            hit_count_next = 0 ;
            start_next = 1 ;
            moletime_next = 20_000_000 ;
        end
        else begin
            if ( n )
                n_next = n - 1 ;
            else
                n_next = moletime - 1 ;
            if ( n == moletime-1 )
                rand_num_next = {rand_num[2:0], feedback} ;
            if ( n >= moletime - moletime/2 ) begin
                sclk_next = 0 ;
                feedback_next = ~(rand_num[3] ^ rand_num[2]);
                hit_next = 0 ;
            end
            else begin
                sclk_next = 1 ;
                if (!hit && (num == rand_num) ) begin
                        hit_next = 1 ;
                        hit_count_next = hit_count + 1 ;
                end
            end
        end
    end
    always_ff @(posedge clock) begin
        moletime <= moletime_next ;
        sclk <= sclk_next ;
        n <= n_next ;
        i <= i_next ;
        rand_num <= rand_num_next ;
        hit <= hit_next ;
        state <= state_next ;
        hit_num <= hit_num_next ;
        hit_count <= hit_count_next ;
        feedback <= feedback_next ;
        level <= level_next ;
    end
endmodule
```

LED Control Module

```
//  led led_0 (.clk, .lat, .oe, .sclk, .r1 ) ;
module led_ctrl
  #( N=8 )
   ( output logic lat, oe, sclk, r1, r2, b1, b2, g1, g2, a, b, c,
     input logic clock, reset_n, hit, start,
     input logic [3:0] rand_num,
     input logic [3:0] hit_count,
     input logic [4:0] num,
     input logic [7:0] gametimer,
     input logic [1:0] level) ;
     logic [31:0] teduri_reg1 [7:0];
     logic [31:0] teduri_reg2 [7:0];
     logic [31:0] key1_reg [7:0] ;
     logic [31:0] key2_reg [7:0] ;
     logic [31:0] key3_reg [7:0] ;
     logic [31:0] key4_reg_b1 [7:0] ;
     logic [31:0] key5_reg_b1 [7:0] ;
     logic [31:0] key6_reg_b1 [7:0] ;
     logic [31:0] key4_reg_b2 [7:0] ;
     logic [31:0] key5_reg_b2 [7:0] ;
     logic [31:0] key6_reg_b2 [7:0] ;
     logic [31:0] key7_reg [7:0] ;
     logic [31:0] key8_reg [7:0] ;
     logic [31:0] key9_reg [7:0] ;
     logic [31:0] score0_reg [7:0] ;
     logic [31:0] score1_reg [7:0] ;
     logic [31:0] score2_reg [7:0] ;
     logic [31:0] score3_reg [7:0] ;
     logic [31:0] score4_reg [7:0] ;
     logic [31:0] score5_reg [7:0] ;
     logic [31:0] score6_reg [7:0] ;
     logic [31:0] score7_reg [7:0] ;
     logic [31:0] score8_reg [7:0] ;
     logic [31:0] score9_reg [7:0] ;
     logic [31:0] score10_reg [7:0] ;
     logic [31:0] score11_reg [7:0] ;
     logic [31:0] score12_reg [7:0] ;
     logic [31:0] score13_reg [7:0] ;
     logic [31:0] score14_reg [7:0] ;
     logic [31:0] score15_reg [7:0] ;
     logic [31:0] score16_reg [7:0] ;
     logic [31:0] score17_reg [7:0] ;
     logic [31:0] score18_reg [7:0] ;
     logic [31:0] score19_reg [7:0] ;
     logic [31:0] r1_reg = '0 ;
     logic [31:0] r2_reg = '0 ;
     logic [31:0] b1_reg = '0 ;
     logic [31:0] b2_reg = '0 ;
     logic [31:0] g1_reg = '0 ;
     logic [31:0] g2_reg = '0 ;
     logic [2:0] row_select, row_select_next ;
     logic [3:0] n, n_next ;
     logic [5:0] i, i_next ;
     logic [2:0] state, state_next ;
     logic r1_next, r2_next,b1_next, b2_next, g1_next, g2_next ;
     logic sclk_next, lat_next, oe_next ;
     logic a_next, b_next, c_next ;
     always_comb begin
        teduri_reg1 [0] = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
        teduri_reg1 [1] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
        teduri_reg1 [2] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
        teduri_reg1 [3] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
        teduri_reg1 [4] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
        teduri_reg1 [5] = 32'b1111_1111_1111_1111_1111_0000_0000_0011;
        teduri_reg1 [6] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
        teduri_reg1 [7] = 32'b1000_0010_0000_1000_0011_1111_1111_1111;
        teduri_reg2 [0] = 32'b1000_0010_0000_1000_0011_1111_1111_1111;
        teduri_reg2 [1] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
```

```
teduri_reg2 [2] = 32'b1111_1111_1111_1111_1111_0000_0000_0011;
teduri_reg2 [3] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
teduri_reg2 [4] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
teduri_reg2 [5] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
teduri_reg2 [6] = 32'b1000_0010_0000_1000_0011_0000_0000_0011;
teduri_reg2 [7] = 32'b1111_1111_1111_1111_1111_1111_1111_1111;
key1_reg [0] = '0 ;
key1_reg [1] = '0 ;
key1_reg [2] = 32'b0011_1000_0000_0000_0000_0000_0000_0000 ;
key1_reg [3] = 32'b0011_1000_0000_0000_0000_0000_0000_0000 ;
key1_reg [4] = '0 ;
key1_reg [5] = '0 ;
key1_reg [6] = '0 ;
key1_reg [7] = '0 ;
key2_reg [0] = '0 ;
key2_reg [1] = '0 ;
key2_reg [2] = 32'b0000_0000_1110_0000_0000_0000_0000_0000 ;
key2_reg [3] = 32'b0000_0000_1110_0000_0000_0000_0000_0000 ;
key2_reg [4] = '0 ;
key2_reg [5] = '0 ;
key2_reg [6] = '0 ;
key2_reg [7] = '0 ;
key3_reg [0] = '0 ;
key3_reg [1] = '0 ;
key3_reg [2] = 32'b0000_0000_0000_0011_1000_0000_0000_0000 ;
key3_reg [3] = 32'b0000_0000_0000_0011_1000_0000_0000_0000 ;
key3_reg [4] = '0 ;
key3_reg [5] = '0 ;
key3_reg [6] = '0 ;
key3_reg [7] = '0 ;
key4_reg_b1 [0] = '0 ;
key4_reg_b1 [1] = '0 ;
key4_reg_b1 [2] = '0 ;
key4_reg_b1 [3] = '0 ;
key4_reg_b1 [4] = '0 ;
key4_reg_b1 [5] = '0 ;
key4_reg_b1 [6] = '0 ;
key4_reg_b1 [7] = 32'b0011_1000_0000_0000_0000_0000_0000_0000 ;
key4_reg_b2 [0] = 32'b0011_1000_0000_0000_0000_0000_0000_0000 ;
key4_reg_b2 [1] = '0 ;
key4_reg_b2 [2] = '0 ;
key4_reg_b2 [3] = '0 ;
key4_reg_b2 [4] = '0 ;
key4_reg_b2 [5] = '0 ;
key4_reg_b2 [6] = '0 ;
key4_reg_b2 [7] = '0 ;
key5_reg_b1 [0] = '0 ;
key5_reg_b1 [1] = '0 ;
key5_reg_b1 [2] = '0 ;
key5_reg_b1 [3] = '0 ;
key5_reg_b1 [4] = '0 ;
key5_reg_b1 [5] = '0 ;
key5_reg_b1 [6] = '0 ;
key5_reg_b1 [7] = 32'b0000_0000_1110_0000_0000_0000_0000_0000 ;
key5_reg_b2 [0] = 32'b0000_0000_1110_0000_0000_0000_0000_0000 ;
key5_reg_b2 [1] = '0 ;
key5_reg_b2 [2] = '0 ;
key5_reg_b2 [3] = '0 ;
key5_reg_b2 [4] = '0 ;
key5_reg_b2 [5] = '0 ;
key5_reg_b2 [6] = '0 ;
key5_reg_b2 [7] = '0 ;
key6_reg_b1 [0] = '0 ;
key6_reg_b1 [1] = '0 ;
key6_reg_b1 [2] = '0 ;
key6_reg_b1 [3] = '0 ;
key6_reg_b1 [4] = '0 ;
key6_reg_b1 [5] = '0 ;
key6_reg_b1 [6] = '0 ;
key6_reg_b1 [7] = 32'b0000_0000_0000_0011_1000_0000_0000_0000 ;
key6_reg_b2 [0] = 32'b0000_0000_0000_0011_1000_0000_0000_0000 ;
```

```
key6_reg_b2 [1] = '0 ;
key6_reg_b2 [2] = '0 ;
key6_reg_b2 [3] = '0 ;
key6_reg_b2 [4] = '0 ;
key6_reg_b2 [5] = '0 ;
key6_reg_b2 [6] = '0 ;
key6_reg_b2 [7] = '0 ;
key7_reg [0] = '0 ;
key7_reg [1] = '0 ;
key7_reg [2] = '0 ;
key7_reg [3] = '0 ;
key7_reg [4] = 32'b0011_1000_0000_0000_0000_0000_0000_0000 ;
key7_reg [5] = 32'b0011_1000_0000_0000_0000_0000_0000_0000 ;
key7_reg [6] = '0 ;
key7_reg [7] = '0 ;
key8_reg [0] = '0 ;
key8_reg [1] = '0 ;
key8_reg [2] = '0 ;
key8_reg [3] = '0 ;
key8_reg [4] = 32'b0000_0000_1110_0000_0000_0000_0000_0000 ;
key8_reg [5] = 32'b0000_0000_1110_0000_0000_0000_0000_0000 ;
key8_reg [6] = '0 ;
key8_reg [7] = '0 ;
key9_reg [0] = '0 ;
key9_reg [1] = '0 ;
key9_reg [2] = '0 ;
key9_reg [3] = '0 ;
key9_reg [4] = 32'b0000_0000_0000_0011_1000_0000_0000_0000 ;
key9_reg [5] = 32'b0000_0000_0000_0011_1000_0000_0000_0000 ;
key9_reg [6] = '0 ;
key9_reg [7] = '0 ;
score0_reg [0] = '0;
score0_reg [1] = '0;
score0_reg [7] = '0;
score0_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score0_reg [3] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score0_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score0_reg [5] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score0_reg [6] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score1_reg [0] = '0;
score1_reg [1] = '0;
score1_reg [7] = '0;
score1_reg [2] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score1_reg [3] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score1_reg [4] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score1_reg [5] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score1_reg [6] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score2_reg [0] = '0;
score2_reg [1] = '0;
score2_reg [7] = '0;
score2_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score2_reg [3] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score2_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score2_reg [5] = 32'b0000_0000_0000_0000_0000_000000_1000_00;
score2_reg [6] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score3_reg [0] = '0;
score3_reg [1] = '0;
score3_reg [7] = '0;
score3_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score3_reg [3] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score3_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score3_reg [5] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score3_reg [6] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score4_reg [0] = '0;
score4_reg [1] = '0;
score4_reg [7] = '0;
score4_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score4_reg [3] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score4_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score4_reg [5] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score4_reg [6] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
```

```
score5_reg [0] = '0;
score5_reg [1] = '0;
score5_reg [7] = '0;
score5_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score5_reg [3] = 32'b0000_0000_0000_0000_0000_000000_1000_00;
score5_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score5_reg [5] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score5_reg [6] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score6_reg [0] = '0;
score6_reg [1] = '0;
score6_reg [7] = '0;
score6_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1000_00;
score6_reg [3] = 32'b0000_0000_0000_0000_0000_000000_1000_00;
score6_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score6_reg [5] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score6_reg [6] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score7_reg [0] = '0;
score7_reg [1] = '0;
score7_reg [7] = '0;
score7_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score7_reg [3] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score7_reg [4] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score7_reg [5] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score7_reg [6] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score8_reg [0] = '0;
score8_reg [1] = '0;
score8_reg [7] = '0;
score8_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score8_reg [3] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score8_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score8_reg [5] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score8_reg [6] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score9_reg [0] = '0;
score9_reg [1] = '0;
score9_reg [7] = '0;
score9_reg [2] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score9_reg [3] = 32'b0000_0000_0000_0000_0000_000000_1001_00;
score9_reg [4] = 32'b0000_0000_0000_0000_0000_000000_1111_00;
score9_reg [5] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score9_reg [6] = 32'b0000_0000_0000_0000_0000_000000_0001_00;
score10_reg [0] = '0;
score10_reg [1] = '0;
score10_reg [7] = '0;
score10_reg [2] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score10_reg [3] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score10_reg [4] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score10_reg [5] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score10_reg [6] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score11_reg [0] = '0;
score11_reg [1] = '0;
score11_reg [7] = '0;
score11_reg [2] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score11_reg [3] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score11_reg [4] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score11_reg [5] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score11_reg [6] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score12_reg [0] = '0;
score12_reg [1] = '0;
score12_reg [7] = '0;
score12_reg [2] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score12_reg [3] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score12_reg [4] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score12_reg [5] = 32'b0000_0000_0000_0000_0000_0_1000_0000000;
score12_reg [6] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score13_reg [0] = '0;
score13_reg [1] = '0;
score13_reg [7] = '0;
score13_reg [2] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score13_reg [3] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score13_reg [4] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score13_reg [5] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
```

```
score13_reg [6] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score14_reg [0] = '0;
score14_reg [1] = '0;
score14_reg [7] = '0;
score14_reg [2] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score14_reg [3] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score14_reg [4] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score14_reg [5] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score14_reg [6] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score15_reg [0] = '0;
score15_reg [1] = '0;
score15_reg [7] = '0;
score15_reg [2] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score15_reg [3] = 32'b0000_0000_0000_0000_0000_0_1000_0000000;
score15_reg [4] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score15_reg [5] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score15_reg [6] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score16_reg [0] = '0;
score16_reg [1] = '0;
score16_reg [7] = '0;
score16_reg [2] = 32'b0000_0000_0000_0000_0000_0_1000_0000000;
score16_reg [3] = 32'b0000_0000_0000_0000_0000_0_1000_0000000;
score16_reg [4] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score16_reg [5] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score16_reg [6] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score17_reg [0] = '0;
score17_reg [1] = '0;
score17_reg [7] = '0;
score17_reg [2] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score17_reg [3] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score17_reg [4] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score17_reg [5] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score17_reg [6] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score18_reg [0] = '0;
score18_reg [1] = '0;
score18_reg [7] = '0;
score18_reg [2] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score18_reg [3] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score18_reg [4] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score18_reg [5] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score18_reg [6] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score19_reg [0] = '0;
score19_reg [1] = '0;
score19_reg [7] = '0;
score19_reg [2] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score19_reg [3] = 32'b0000_0000_0000_0000_0000_0_1001_0000000;
score19_reg [4] = 32'b0000_0000_0000_0000_0000_0_1111_0000000;
score19_reg [5] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
score19_reg [6] = 32'b0000_0000_0000_0000_0000_0_0001_0000000;
r1_next = r1 ;
r2_next = r2 ;
b1_next = b1 ;
b2_next = b2 ;
g1_next = g1 ;
g2_next = g2 ;
a_next = a ;
b_next = b ;
c_next = c ;
row_select_next = row_select ;
sclk_next = sclk ;
n_next = n ;
i_next = i ;
lat_next = lat ;
oe_next = oe ;
state_next = state ;
if (!reset_n || num == 4'ha|| num == 4'hb|| num == 4'hc|| num == 4'hd) begin
    oe_next = 1 ;
    lat_next = 0 ;
    n_next = N - 1 ;
    i_next = 31 ;
    state_next = 0 ;
```

```
    row_select_next = 3'b000;
    a_next = row_select[0] ;
    b_next = row_select[1] ;
    c_next = row_select[2] ;
end
else if ( reset_n ) begin
    if ( n ) // 3 2 1 0
        n_next = n - 1 ; // decrement
    else if ( !n ) begin
        n_next = N - 1 ; // reset to 8
        i_next = i - 1 ;
    end
    if ( state == 0 ) begin // read data from reg
                            // t setup= 60 ns
                            // t hold = 20 ns
                            // min clock pulse width = 50 ns
        if ( n >= N - N/2 )
            sclk_next= 0;
        else
            sclk_next = 1 ;
        if ( i >= 0 && n == 5  ) begin
            if (level == 0 ) begin
            r1_next = teduri_reg1[row_select][i] ;
            r2_next = teduri_reg2[row_select][i] ;
            b1_next = '0 ;
            b2_next = '0 ;
            g1_next = '0 ;
            g2_next = '0 ;
            if ( start ) begin
                if ( rand_num == 1 )                // blue mole
                    b1_next = key1_reg[row_select][i] ;
                else if ( rand_num == 2 )
                    b1_next = key2_reg[row_select][i] ;
                else if ( rand_num == 3 )
                    b1_next = key3_reg[row_select][i] ;
                else if ( rand_num == 4 ) begin
                    b1_next = key4_reg_b1[row_select][i] ;
                    b2_next = key4_reg_b2[row_select][i] ;
                end
                else if ( rand_num == 5 ) begin
                    b1_next = key5_reg_b1[row_select][i] ;
                    b2_next = key5_reg_b2[row_select][i] ;
                end
                else if ( rand_num == 6 ) begin
                    b1_next = key6_reg_b1[row_select][i] ;
                    b2_next = key6_reg_b2[row_select][i] ;
                end
                else if ( rand_num == 7 )
                    b2_next = key7_reg[row_select][i] ;
                else if ( rand_num == 8 )
                    b2_next = key8_reg[row_select][i] ;
                else if ( rand_num == 9 )
                    b2_next = key9_reg[row_select][i] ;
                end
                else begin
                    b1_next = '0 ;
                    b2_next = '0 ;
                end
                if (hit_count == 0 )
                    g2_next = score0_reg[row_select][i]+ score10_reg[row_select][i] ;
                else if (hit_count == 1)
                    g2_next = score1_reg[row_select][i]+ score10_reg[row_select][i] ;
                else if (hit_count == 2)
                    g2_next = score2_reg[row_select][i]+ score10_reg[row_select][i] ;
                else if (hit_count == 3)
                    g2_next = score3_reg[row_select][i]+ score10_reg[row_select][i] ;
                else if (hit_count == 4)
                    g2_next = score4_reg[row_select][i]+ score10_reg[row_select][i] ;
                else if (hit_count == 5)
                    g2_next = score5_reg[row_select][i]+ score10_reg[row_select][i] ;
                else if (hit_count == 6)
```

```
    g2_next = score6_reg[row_select][i]+ score10_reg[row_select][i] ;
else if (hit_count == 7)
    g2_next = score7_reg[row_select][i]+ score10_reg[row_select][i] ;
else if (hit_count == 8)
    g2_next = score8_reg[row_select][i]+ score10_reg[row_select][i] ;
else if (hit_count == 9)
    g2_next = score9_reg[row_select][i]+ score10_reg[row_select][i] ;
else if (hit_count == 10)
    g2_next = score0_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 11)
    g2_next = score1_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 12)
    g2_next = score2_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 13)
    g2_next = score3_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 14)
    g2_next = score4_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 15)
    g2_next = score5_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 16)
    g2_next = score6_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 17)
    g2_next = score7_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 18)
    g2_next = score8_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 19)
    g2_next = score9_reg[row_select][i]+ score11_reg[row_select][i] ;
else if (hit_count == 20)
    g2_next = score0_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 21)
    g2_next = score1_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 22)
    g2_next = score2_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 23)
    g2_next = score3_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 24)
    g2_next = score4_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 25)
    g2_next = score5_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 26)
    g2_next = score6_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 27)
    g2_next = score7_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 28)
    g2_next = score8_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 29)
    g2_next = score9_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 30)
if ( gametimer <= 9 ) begin
    if ( gametimer == 0 )
        g1_next = score0_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 1)
        g1_next = score1_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 2)
        g1_next = score2_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 3)
        g1_next = score3_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 4)
        g1_next = score4_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 5)
        g1_next = score5_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 6)
        g1_next = score6_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 7)
        g1_next = score7_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 8)
        g1_next = score8_reg[row_select][i] + score10_reg[row_select][i];
    else if (gametimer == 9)
        g1_next = score9_reg[row_select][i] + score10_reg[row_select][i];
end
else if ( gametimer >= 10 && gametimer <= 19 ) begin
```

```
        if ( gametimer == 10 )
            g1_next = score0_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 11)
            g1_next = score1_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 12)
            g1_next = score2_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 13)
            g1_next = score3_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 14)
            g1_next = score4_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 15)
            g1_next = score5_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 16)
            g1_next = score6_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 17)
            g1_next = score7_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 18)
            g1_next = score8_reg[row_select][i] + score11_reg[row_select][i];
        else if (gametimer == 19)
            g1_next = score9_reg[row_select][i] + score11_reg[row_select][i];
    end
    else if ( gametimer >= 20 && gametimer <= 29 ) begin
        if ( gametimer == 20 )
            g1_next = score0_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 21)
            g1_next = score1_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 22)
            g1_next = score2_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 23)
            g1_next = score3_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 24)
            g1_next = score4_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 25)
            g1_next = score5_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 26)
            g1_next = score6_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 27)
            g1_next = score7_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 28)
            g1_next = score8_reg[row_select][i] + score12_reg[row_select][i];
        else if (gametimer == 29)
            g1_next = score9_reg[row_select][i] + score12_reg[row_select][i];
    end
    else if ( gametimer >= 30 && gametimer <= 39 ) begin
        if ( gametimer == 30 )
            g1_next = score0_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 31)
            g1_next = score1_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 32)
            g1_next = score2_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 33)
            g1_next = score3_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 34)
            g1_next = score4_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 35)
            g1_next = score5_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 36)
            g1_next = score6_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 37)
            g1_next = score7_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 38)
            g1_next = score8_reg[row_select][i] + score13_reg[row_select][i];
        else if (gametimer == 39)
            g1_next = score9_reg[row_select][i] + score13_reg[row_select][i];
    end
end
else if ( level == 1 ) begin
    r1_next = teduri_reg1[row_select][i] ;
    r2_next = teduri_reg2[row_select][i] ;
    b1_next = teduri_reg1[row_select][i] ;
    b2_next = teduri_reg2[row_select][i] ;
```

```
if ( start ) begin
   if ( rand_num == 1 )                    // blue mole
      b1_next = key1_reg[row_select][i]+teduri_reg1[row_select][i] ;
   else if ( rand_num == 2 )
      b1_next = key2_reg[row_select][i]+teduri_reg1[row_select][i] ;
   else if ( rand_num == 3 )
      b1_next = key3_reg[row_select][i]+teduri_reg1[row_select][i] ;
   else if ( rand_num == 4 ) begin
      b1_next = key4_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
      b2_next = key4_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
   end
   else if ( rand_num == 5 ) begin
      b1_next = key5_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
      b2_next = key5_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
   end
   else if ( rand_num == 6 ) begin
      b1_next = key6_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
      b2_next = key6_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
   end
   else if ( rand_num == 7 )
      b2_next = key7_reg[row_select][i]+teduri_reg2[row_select][i] ;
   else if ( rand_num == 8 )
      b2_next = key8_reg[row_select][i]+teduri_reg2[row_select][i] ;
   else if ( rand_num == 9 )
      b2_next = key9_reg[row_select][i]+teduri_reg2[row_select][i] ;
   if (hit_count == 0 )
      g2_next = score0_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 1)
      g2_next = score1_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 2)
      g2_next = score2_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 3)
      g2_next = score3_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 4)
      g2_next = score4_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 5)
      g2_next = score5_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 6)
      g2_next = score6_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 7)
      g2_next = score7_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 8)
      g2_next = score8_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 9)
      g2_next = score9_reg[row_select][i]+ score10_reg[row_select][i] ;
   else if (hit_count == 10)
      g2_next = score0_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 11)
      g2_next = score1_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 12)
      g2_next = score2_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 13)
      g2_next = score3_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 14)
      g2_next = score4_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 15)
      g2_next = score5_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 16)
      g2_next = score6_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 17)
      g2_next = score7_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 18)
      g2_next = score8_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 19)
      g2_next = score9_reg[row_select][i]+ score11_reg[row_select][i] ;
   else if (hit_count == 20)
      g2_next = score0_reg[row_select][i]+ score12_reg[row_select][i] ;
   else if (hit_count == 21)
      g2_next = score1_reg[row_select][i]+ score12_reg[row_select][i] ;
   else if (hit_count == 22)
      g2_next = score2_reg[row_select][i]+ score12_reg[row_select][i] ;
```

```
else if (hit_count == 23)
   g2_next = score3_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 24)
   g2_next = score4_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 25)
   g2_next = score5_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 26)
   g2_next = score6_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 27)
   g2_next = score7_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 28)
   g2_next = score8_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 29)
   g2_next = score9_reg[row_select][i]+ score12_reg[row_select][i] ;
else if (hit_count == 30)
   g2_next = score0_reg[row_select][i]+ score13_reg[row_select][i] ;
if ( gametimer <= 9 ) begin
   if ( gametimer == 0 )
      g1_next = score0_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 1)
      g1_next = score1_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 2)
      g1_next = score2_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 3)
      g1_next = score3_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 4)
      g1_next = score4_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 5)
      g1_next = score5_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 6)
      g1_next = score6_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 7)
      g1_next = score7_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 8)
      g1_next = score8_reg[row_select][i] + score10_reg[row_select][i];
   else if (gametimer == 9)
      g1_next = score9_reg[row_select][i] + score10_reg[row_select][i];
end
else if ( gametimer >= 10 && gametimer <= 19 ) begin
   if ( gametimer == 10 )
      g1_next = score0_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 11)
      g1_next = score1_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 12)
      g1_next = score2_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 13)
      g1_next = score3_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 14)
      g1_next = score4_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 15)
      g1_next = score5_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 16)
      g1_next = score6_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 17)
      g1_next = score7_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 18)
      g1_next = score8_reg[row_select][i] + score11_reg[row_select][i];
   else if (gametimer == 19)
      g1_next = score9_reg[row_select][i] + score11_reg[row_select][i];
end
else if ( gametimer >= 20 && gametimer <= 29 ) begin
   if ( gametimer == 20 )
      g1_next = score0_reg[row_select][i] + score12_reg[row_select][i];
   else if (gametimer == 21)
      g1_next = score1_reg[row_select][i] + score12_reg[row_select][i];
   else if (gametimer == 22)
      g1_next = score2_reg[row_select][i] + score12_reg[row_select][i];
   else if (gametimer == 23)
      g1_next = score3_reg[row_select][i] + score12_reg[row_select][i];
   else if (gametimer == 24)
      g1_next = score4_reg[row_select][i] + score12_reg[row_select][i];
```

```
                    else if (gametimer == 25)
                        g1_next = score5_reg[row_select][i] + score12_reg[row_select][i];
                    else if (gametimer == 26)
                        g1_next = score6_reg[row_select][i] + score12_reg[row_select][i];
                    else if (gametimer == 27)
                        g1_next = score7_reg[row_select][i] + score12_reg[row_select][i];
                    else if (gametimer == 28)
                        g1_next = score8_reg[row_select][i] + score12_reg[row_select][i];
                    else if (gametimer == 29)
                        g1_next = score9_reg[row_select][i] + score12_reg[row_select][i];
                end
                else if ( gametimer >= 30 && gametimer <= 39 ) begin
                    if ( gametimer == 30 )
                        g1_next = score0_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 31)
                        g1_next = score1_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 32)
                        g1_next = score2_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 33)
                        g1_next = score3_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 34)
                        g1_next = score4_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 35)
                        g1_next = score5_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 36)
                        g1_next = score6_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 37)
                        g1_next = score7_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 38)
                        g1_next = score8_reg[row_select][i] + score13_reg[row_select][i];
                    else if (gametimer == 39)
                        g1_next = score9_reg[row_select][i] + score13_reg[row_select][i];
                end
            end
        end
        else if ( level == 2 ) begin
            b1_next = teduri_reg1[row_select][i] ;
            b2_next = teduri_reg2[row_select][i] ;
            g1_next = teduri_reg1[row_select][i] ;
            g2_next = teduri_reg2[row_select][i] ;
            r1_next = '0;
            r2_next = '0;
            if ( start ) begin
                if ( rand_num == 1 )                    // blue mole
                    b1_next = key1_reg[row_select][i]+teduri_reg1[row_select][i] ;
                else if ( rand_num == 2 )
                    b1_next = key2_reg[row_select][i]+teduri_reg1[row_select][i] ;
                else if ( rand_num == 3 )
                    b1_next = key3_reg[row_select][i]+teduri_reg1[row_select][i] ;
                else if ( rand_num == 4 ) begin
                    b1_next = key4_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
                    b2_next = key4_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
                end
                else if ( rand_num == 5 ) begin
                    b1_next = key5_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
                    b2_next = key5_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
                end
                else if ( rand_num == 6 ) begin
                    b1_next = key6_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
                    b2_next = key6_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
                end
                else if ( rand_num == 7 )
                    b2_next = key7_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if ( rand_num == 8 )
                    b2_next = key8_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if ( rand_num == 9 )
                    b2_next = key9_reg[row_select][i]+teduri_reg2[row_select][i] ;
                if (hit_count == 0 )
                    g2_next = score0_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 1)
```

```
                    g2_next = score1_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 2)
                    g2_next = score2_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 3)
                    g2_next = score3_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 4)
                    g2_next = score4_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 5)
                    g2_next = score5_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 6)
                    g2_next = score6_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 7)
                    g2_next = score7_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 8)
                    g2_next = score8_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 9)
                    g2_next = score9_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 10)
                    g2_next = score0_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 11)
                    g2_next = score1_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 12)
                    g2_next = score2_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 13)
                    g2_next = score3_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 14)
                    g2_next = score4_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 15)
                    g2_next = score5_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 16)
                    g2_next = score6_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 17)
                    g2_next = score7_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 18)
                    g2_next = score8_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 19)
                    g2_next = score9_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 20)
                    g2_next = score0_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 21)
                    g2_next = score1_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 22)
                    g2_next = score2_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 23)
                    g2_next = score3_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 24)
                    g2_next = score4_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
```

```
                    else if (hit_count == 25)
                        g2_next = score5_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                    else if (hit_count == 26)
                        g2_next = score6_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                    else if (hit_count == 27)
                        g2_next = score7_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                    else if (hit_count == 28)
                        g2_next = score8_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                    else if (hit_count == 29)
                        g2_next = score9_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                    else if (hit_count == 30)
                        g2_next = score0_reg[row_select][i]+
score13_reg[row_select][i]+teduri_reg2[row_select][i] ;
                    if ( gametimer <= 9 ) begin
                        if ( gametimer == 0 )
                            g1_next = score0_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 1)
                            g1_next = score1_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 2)
                            g1_next = score2_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 3)
                            g1_next = score3_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 4)
                            g1_next = score4_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 5)
                            g1_next = score5_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 6)
                            g1_next = score6_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 7)
                            g1_next = score7_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 8)
                            g1_next = score8_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 9)
                            g1_next = score9_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    end
                    else if ( gametimer >= 10 && gametimer <= 19 ) begin
                        if ( gametimer == 10 )
                            g1_next = score0_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 11)
                            g1_next = score1_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 12)
                            g1_next = score2_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 13)
                            g1_next = score3_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 14)
                            g1_next = score4_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 15)
                            g1_next = score5_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                        else if (gametimer == 16)
```

```
                        g1_next = score6_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 17)
                        g1_next = score7_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 18)
                        g1_next = score8_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 19)
                        g1_next = score9_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                end
                else if ( gametimer >= 20 && gametimer <= 29 ) begin
                    if ( gametimer == 20 )
                        g1_next = score0_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 21)
                        g1_next = score1_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 22)
                        g1_next = score2_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 23)
                        g1_next = score3_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 24)
                        g1_next = score4_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 25)
                        g1_next = score5_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 26)
                        g1_next = score6_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 27)
                        g1_next = score7_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 28)
                        g1_next = score8_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 29)
                        g1_next = score9_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                end
                else if ( gametimer >= 30 && gametimer <= 39 ) begin
                    if ( gametimer == 30 )
                        g1_next = score0_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 31)
                        g1_next = score1_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 32)
                        g1_next = score2_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 33)
                        g1_next = score3_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 34)
                        g1_next = score4_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 35)
                        g1_next = score5_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 36)
                        g1_next = score6_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 37)
                        g1_next = score7_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 38)
```

```
                        g1_next = score8_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 39)
                        g1_next = score9_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                end
            end
        end
        else if ( level == 3 ) begin
            r1_next = teduri_reg1[row_select][i] ;
            r2_next = teduri_reg2[row_select][i] ;
            b1_next = teduri_reg1[row_select][i] ;
            b2_next = teduri_reg2[row_select][i] ;
            g1_next = teduri_reg1[row_select][i] ;
            g2_next = teduri_reg2[row_select][i] ;
            if ( start ) begin
                if ( rand_num == 1 )                // blue mole
                    b1_next = key1_reg[row_select][i]+teduri_reg1[row_select][i] ;
                else if ( rand_num == 2 )
                    b1_next = key2_reg[row_select][i]+teduri_reg1[row_select][i] ;
                else if ( rand_num == 3 )
                    b1_next = key3_reg[row_select][i]+teduri_reg1[row_select][i] ;
                else if ( rand_num == 4 ) begin
                    b1_next = key4_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
                    b2_next = key4_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
                end
                else if ( rand_num == 5 ) begin
                    b1_next = key5_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
                    b2_next = key5_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
                end
                else if ( rand_num == 6 ) begin
                    b1_next = key6_reg_b1[row_select][i]+teduri_reg1[row_select][i] ;
                    b2_next = key6_reg_b2[row_select][i]+teduri_reg2[row_select][i] ;
                end
                else if ( rand_num == 7 )
                    b2_next = key7_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if ( rand_num == 8 )
                    b2_next = key8_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if ( rand_num == 9 )
                    b2_next = key9_reg[row_select][i]+teduri_reg2[row_select][i] ;
                if (hit_count == 0 )
                    g2_next = score0_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 1)
                    g2_next = score1_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 2)
                    g2_next = score2_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 3)
                    g2_next = score3_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 4)
                    g2_next = score4_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 5)
                    g2_next = score5_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 6)
                    g2_next = score6_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 7)
                    g2_next = score7_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 8)
                    g2_next = score8_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 9)
                    g2_next = score9_reg[row_select][i]+
score10_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 10)
```

```
                    g2_next = score0_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 11)
                    g2_next = score1_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 12)
                    g2_next = score2_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 13)
                    g2_next = score3_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 14)
                    g2_next = score4_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 15)
                    g2_next = score5_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 16)
                    g2_next = score6_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 17)
                    g2_next = score7_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 18)
                    g2_next = score8_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 19)
                    g2_next = score9_reg[row_select][i]+
score11_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 20)
                    g2_next = score0_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 21)
                    g2_next = score1_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 22)
                    g2_next = score2_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 23)
                    g2_next = score3_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 24)
                    g2_next = score4_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 25)
                    g2_next = score5_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 26)
                    g2_next = score6_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 27)
                    g2_next = score7_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 28)
                    g2_next = score8_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 29)
                    g2_next = score9_reg[row_select][i]+
score12_reg[row_select][i]+teduri_reg2[row_select][i] ;
                else if (hit_count == 30)
                    g2_next = score0_reg[row_select][i]+
score13_reg[row_select][i]+teduri_reg2[row_select][i] ;
                if ( gametimer <= 9 ) begin
                    if ( gametimer == 0 )
                        g1_next = score0_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 1)
                        g1_next = score1_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 2)
```

```
                        g1_next = score2_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 3)
                        g1_next = score3_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 4)
                        g1_next = score4_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 5)
                        g1_next = score5_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 6)
                        g1_next = score6_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 7)
                        g1_next = score7_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 8)
                        g1_next = score8_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 9)
                        g1_next = score9_reg[row_select][i] +
score10_reg[row_select][i]+teduri_reg1[row_select][i];
                end
                else if ( gametimer >= 10 && gametimer <= 19 ) begin
                    if ( gametimer == 10 )
                        g1_next = score0_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 11)
                        g1_next = score1_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 12)
                        g1_next = score2_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 13)
                        g1_next = score3_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 14)
                        g1_next = score4_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 15)
                        g1_next = score5_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 16)
                        g1_next = score6_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 17)
                        g1_next = score7_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 18)
                        g1_next = score8_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 19)
                        g1_next = score9_reg[row_select][i] +
score11_reg[row_select][i]+teduri_reg1[row_select][i];
                end
                else if ( gametimer >= 20 && gametimer <= 29 ) begin
                    if ( gametimer == 20 )
                        g1_next = score0_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 21)
                        g1_next = score1_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 22)
                        g1_next = score2_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 23)
                        g1_next = score3_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                    else if (gametimer == 24)
```

```
                        g1_next = score4_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 25)
                        g1_next = score5_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 26)
                        g1_next = score6_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 27)
                        g1_next = score7_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 28)
                        g1_next = score8_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 29)
                        g1_next = score9_reg[row_select][i] +
score12_reg[row_select][i]+teduri_reg1[row_select][i];
                  end
                  else if ( gametimer >= 30 && gametimer <= 39 ) begin
                     if ( gametimer == 30 )
                        g1_next = score0_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 31)
                        g1_next = score1_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 32)
                        g1_next = score2_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 33)
                        g1_next = score3_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 34)
                        g1_next = score4_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 35)
                        g1_next = score5_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 36)
                        g1_next = score6_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 37)
                        g1_next = score7_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 38)
                        g1_next = score8_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                     else if (gametimer == 39)
                        g1_next = score9_reg[row_select][i] +
score13_reg[row_select][i]+teduri_reg1[row_select][i];
                  end
               end
            end
            if ( !n && !i ) begin
               state_next = 1 ;
               i_next = 6 ;
               oe_next = 1 ;
               lat_next = 0 ;
               a_next = row_select[0] ;
               b_next = row_select[1] ;
               c_next = row_select[2] ;
            end
         end // if ( state == 0 )
         else if ( state == 1 ) begin // latch
                                      // hold time, t setup low = 100 ns
                                      // holdtime for data = 20 ns
                                      // setup time for data = 60 ns
                                   // latch pulse width = 100 ns
            sclk_next= 0 ;
            if ( i >= 3 )
               lat_next = 1 ;
```

```
            if ( !n && !i ) begin
               state_next = 2 ;
               i_next =  10;
               lat_next = 0 ;
               oe_next = 1 ;
            end
         end // else if ( state == 1 )
         else if ( state == 2 ) begin // oe    i = 160 ns
                                      // tpLH typ = 120 ns and max = 150 ns
                                      // tpHL typ = 70 ns and  max = 100 ns
                                      // output current rise time = 100 ns max = 200 ns    3 i cycle on
                                      // output current fall time = 60 ns  max = 120 ns
           sclk_next= 0 ;
            if ( i >= 5 )
               oe_next = 0 ;
            if ( !n && !i ) begin
               state_next = 3 ;
               i_next = 2 ;
               lat_next = 0 ;
               oe_next = 1 ;
            end
         end // else if ( state == 2 )
         else if ( state == 3 ) begin
            if (!i) begin
               state_next = 0 ;
               i_next = 31 ;
               n_next = N - 1 ;
               if (row_select <= 6)
                  row_select_next = row_select +1  ;
               else
                  row_select_next = 0 ;
            end
         end
      end // else if ( reset_n )
   end // always_comb
   always_ff @ ( posedge clock ) begin
      n <= n_next ;
      sclk <= sclk_next ;
      state <= state_next ;
      oe <= oe_next ;
      lat <= lat_next ;
      i <= i_next ;
      a <= a_next ;
      b <= b_next ;
      c <= c_next ;
      r1 <= r1_next ;
      r2 <= r2_next ;
      b1 <= b1_next ;
      b2 <= b2_next ;
      g1 <= g1_next ;
      g2 <= g2_next ;
      row_select <= row_select_next ;
   end
endmodule
```

REFERENCE

[1] G. Charitos, A. Bagri, A. Agrawal and J. Sharma, "CSEE 4840 Embedded Systems "Whac-A-Mole" Project Report," 2016.

[2] "74HC244; 74HCT244," Nexperia B.V., 26 February 2016. [Online]. Available: https://assets.nexperia.com/documents/data-sheet/74HC_HCT244.pdf.

[3] P. Burgess, "32x16 and 32x32 RGB LED Matrix Overview," Adafruit, [Online]. Available: https://learn.adafruit.com/32x16-32x32-rgb-led-matrix/overview.

[4] L. Ada, "FPGA FGB Matrix Overview," Adafruit, [Online]. Available: https://learn.adafruit.com/fpga-rgb-matrix/overview.

[5] "Adafruit FGB LED Matrix," Ray's Logic, [Online]. Available: http://www.rayslogic.com/propeller/Programming/AdafruitRGB/AdafruitRGB.htm.

[6] "16-bit Constant Current LED Sink Driver," Macroblock, [Online]. Available: http://www.rayslogic.com/propeller/Programming/AdafruitRGB/MBI5026.pdf.

[7] Toni T800, "Tutorial: Driving 16x32 RGB LED matrix with DE0-Nano-SoC FPGA board," 28 5 2015. [Online]. Available: https://www.youtube.com/watch?v=_ML6TSaRPLo.