2013

ECE 241 – FINAL PROJECT REPORT

YingHui Fan – 999594648

Yue Wang – 999975594

12/1/2013

ECE 241 – FINAL PROJECT

ESCAPE - BY YUE WANG, YINGHUI FAN

Table of Contents

Description of the Project

This designed project is a typing game which generates logic using DE2 board, takes in keyboard signals as input, and display corresponding images on a VGA monitor as outputs.

In this game, the main character, a cyan squid, is running away from the Pacman, the monster. In order for the squid to run forward, the player has to type in the character that is displayed on the screen, every time the player types in the correct character, the program generates a new random alphabet letter or space, and replaces the old character with this newly generated random character on the screen. The speed of the squid is never constant, the faster the player types, the faster the squid runs away from the Pacman. The squid dies either when it is caught by the Pacman or it runs out of energy (a time limit of 80 seconds), or it survived by successfully run away from the Pacman (reaches the right end of the screen).

Translating Ideas into Verilog Code

The Finite State Diagram

In order to do multiple tasks at the "same" time and to react differently depending on users' input, a unique Finite State Machine is designed to complete the job of this specific project. Figure 1 shows the Finite State Diagram.

In IDLE state, the program initializes every counter, enable signal, and registers. Whenever the user pressed ENTER key on the keyboard, the current state goes to START state.

In START state, the program checks the user has already won the game or lost the game and sets the next state as the WIN state or LOSE state. Otherwise, it keeps checking if the keyboard input matches the current alphabet letter. If it does, the program generates a new 5 bits random number, and based on the random number next state will be set as A, B, C, …, Z, or SPACE states. Also, at every 0.25 second, the next state is set to be DRAW state, which creates the animation.

In A, B, C, …, Z, and SPACE states, the program displays the corresponding character

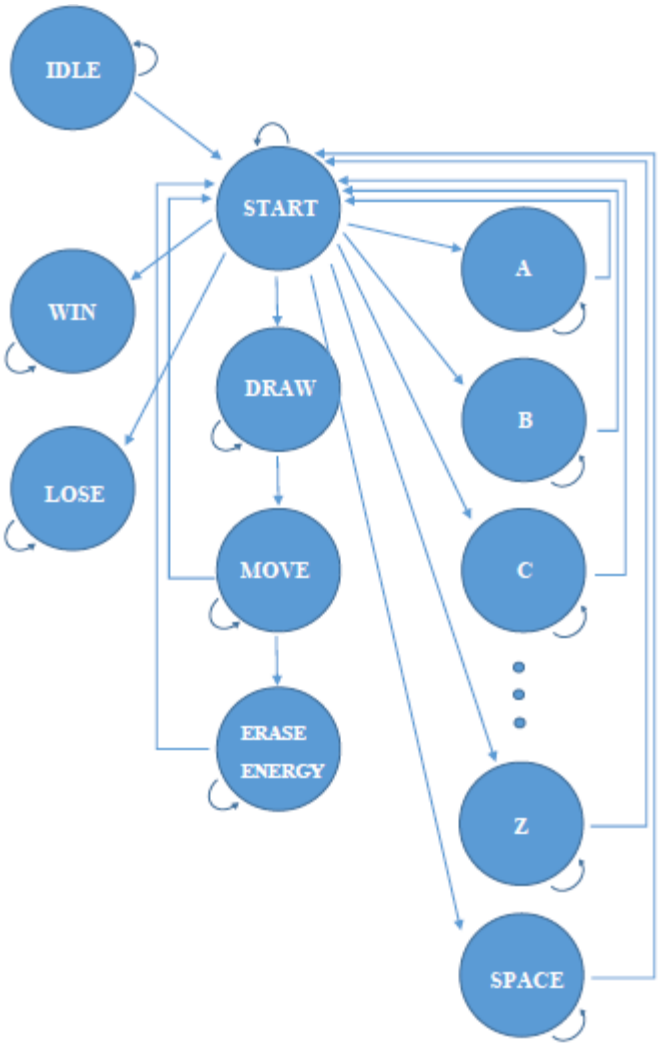

Figure 1 – Finite State Diagram of *ESCAPE*

on the VGA monitor and reset the current alphabet letter.

In the DRAW state, the program creates an animation of the road moving backwards with respect to the monster. It redraws a new dash line at every 0.25 second to create the effect.

In the MOVE state, the program redraws the main character at a new position based on performance of the player in the past 0.25 second, checks if the player wins or loses the game, and decides whether next state should be START state or ERASE ENERGY state. If the player has zero correct input, the program draws the main character at one pixel left from the previous position (the main character does not move to the respect of the road). Otherwise, depending on the number of correct alphabets the user has inputted in the past 0.25 second, the program redraws the main character at two or three pixel right from the previous position (the main character moves forward to the respect of the road). If the character's x position reaches the monster, the lose signal is enabled, or it reaches the right edge of the screen, the win signal is enabled. If the program delays 0.5 second, the next state is set to be ERASE ENERGY state, else next state is set to be START state.

In the ERASE ENERGY state, the program draws the remaining energy bar black by one column. Whenever there is no energy left, the lose signal is enabled. Then the next state is set to be START state.

In the WIN state, the program draws the you-win image on the screen.

In the LOSE state, the program draws the you-lose image on the screen.

Difficulties Met During the Project

Animation of the Dash Line

The original idea of designing the moving dash lines was to redraw the same dash line at a new position, one pixel left from the previous one, at every 0.25 second. The design was not successful, Random bugs such as some dash lines got resized and appeared to be longer the others, or after moving couple pixels left the dash line stopped moving completely, etc. This was because the whole idea was built upon an assumption that the pixel position (-1, y) would show up at position (159, y) (see Appendix B for more details).

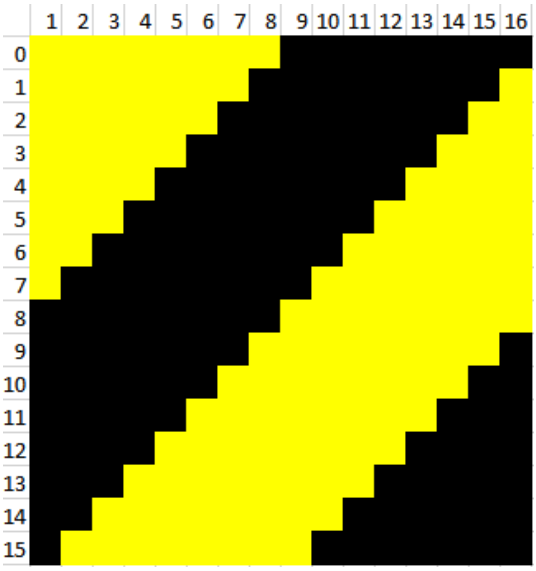

Figure 2 – Case break down for the animation of the dash line.

The solution to overcome this problem was to create a counter to count the number of pixels the dash line is shifted left. And depending on the counter, the design was divided into 15 different cases; and in each case, the program would create different sequence and display this sequence repeatedly starting from the

position (0, y) till (159, y), see figure 2 for more details.

Combining Multiple FSM

Breaking down the whole project into smaller tasks and designing smaller FSM for each task was not difficult, but combining those FSM together as a completed FSM was difficult. The reason to this was because the overall logic behind the completed FSM was missing some key elements. The only solution to this problem was to list down all possible cases and correct the sequence of events.

Generating Random Numbers

Random numbers were necessary in order to challenge the user throughout the game since he/she would not be able to guess the next letter by observing a certain pattern. Despite having the correct implementation of a Linear Feedback Shift Register (random number generator function in hardware), it proved to be impossible to have random numbers being generated. This was due to the fact that the initial seed for the LFSR was always the same. The issue was solved by simply letting the initial seed be the same every time instead of hardcoding a specific sequence that seemed random.

Appendix A – Full Demo Video

Full demo video can be found: https://www.dropbox.com/sc/lzzf9ot7jim9egb/Y6iHJdTewE

Appendix B – Original Design of the Dash Line Animation

```
if(state == draw)
begin
	if(~done[0])
	begin
		write <= 1;
		y <= 7'b1001111;
		counter_x = counter_x + 1;
		counter_pix = counter_pix + 1;
		if(counter_x < 161)
		begin
			if((counter_pix > 0) && (counter_pix < 9))
			begin
				x <= counter_x - 8'b00000001;
				color<=3'b110;
			end
			else if((counter_pix > 8) && (counter_pix < 16))
			begin
				x <= counter_x - 8'b00000001;
				color<=3'b000;
			end
			else if(counter_pix == 16)
```

```
                        begin
                                x <= counter_x - 8'b00000001;
                                color <= 3'b000;
                                counter_pix = 0;
                        end
                end
                else if(counter_x == 161)
                begin
                        counter_x = 0;
                        done[0] = 1;
                        write <= 0;
                end
        end
end
if(state == shiftleft)
begin
        shift_pix = shift_pix + 1;
        done[0]=0;
        if(shift_pix == 16)
        begin
                done[1] = 1;
                shift_pix = 0;
        end
end
```

Appendix C – Verilog Code

```
module final_project
(
        CLOCK_50,       //      On Board 50 MHz
        KEY,    //      Push Button[3:0]
        VGA_CLK,        //      VGA Clock
        VGA_HS,         //      VGA H_SYNC
        VGA_VS,         //      VGA V_SYNC
        VGA_BLANK,      //      VGA BLANK
        VGA_SYNC,       //      VGA SYNC
        VGA_R,          //      VGA Red[9:0]
        VGA_G,                  //      VGA Green[9:0]
        VGA_B,          //      VGA Blue[9:0]
        PS2_CLK,
        PS2_DAT,
        LEDG,
        LEDR
        );
        output [17:0]LEDR;
        output [5:0]LEDG;
```

```
input     CLOCK_50;    //       50 MHz
input    [3:0]    KEY;    //       Button[3:0]
output    VGA_CLK;    //       VGA Clock
output    VGA_HS;       //       VGA H_SYNC
output    VGA_VS;       //       VGA V_SYNC
output    VGA_BLANK;          //       VGA BLANK
output    VGA_SYNC;   //       VGA SYNC
output   [9:0]    VGA_R;          //       VGA Red[9:0]
output   [9:0]    VGA_G;                //       VGA Green[9:0]
output   [9:0]    VGA_B;          //       VGA Blue[9:0]
inout    PS2_CLK;
inout    PS2_DAT;

wire reset;
assign reset=~KEY[0];
wire resetn1;
assign resetn1 = 1;

// Create the color, x, y and writeEn wires that are inputs to the controller.
wire [2:0] c_out;
//wire [7:0] X;
wire [7:0] X_out;
//wire [6:0] Y;
wire [6:0] Y_out;
wire write;

datapath(CLOCK_50, reset, X_out, Y_out, c_out ,write, PS2_CLK, PS2_DAT, KEY[3], LEDG,
LEDR);

vga_adapter VGA(
  .resetn(resetn1),
  .clock(CLOCK_50),
  .colour(c_out),
  .x(X_out),
  .y(Y_out),
  .plot(write),
  /* Signals for the DAC to drive the monitor. */
  .VGA_R(VGA_R),
  .VGA_G(VGA_G),
  .VGA_B(VGA_B),
  .VGA_HS(VGA_HS),
  .VGA_VS(VGA_VS),
  .VGA_BLANK(VGA_BLANK),
  .VGA_SYNC(VGA_SYNC),
  .VGA_CLK(VGA_CLK));
 defparam VGA.RESOLUTION = "160x120";
 defparam VGA.MONOCHROME = "FALSE";
```

```
        defparam VGA.BITS_PER_COLOUR_CHANNEL = 1;
        defparam VGA.BACKGROUND_IMAGE = "display.mif";
endmodule



module datapath(input CLOCK_50, input reset, output reg [7:0]x_out, output reg[6:0]y_out, output
reg[2:0]c_out,
        output reg write, inout PS2_CLK, inout PS2_DAT, input key, output [5:0]LED, output[17:0]led);

        assign LED = state;
        assign led[4:0] = num;
        assign led[17:10] = char_x_old;

        wire [2:0]color1;
        wire [2:0]color2;
        wire [2:0]color3;
        wire [2:0]color4;
        wire [2:0]color5;
        wire [2:0]color6;
        wire [2:0]color7;
        wire [2:0]color8;
        wire [2:0]color9;
        wire [2:0]color10;
        wire [2:0]color11;
        wire [2:0]color12;
        wire [2:0]color13;
        wire [2:0]color14;
        wire [2:0]color15;
        wire [2:0]color16;
        wire [2:0]color17;
        wire [2:0]color18;
        wire [2:0]color19;
        wire [2:0]color20;
        wire [2:0]color21;
        wire [2:0]color22;
        wire [2:0]color23;
        wire [2:0]color24;
        wire [2:0]color25;
        wire [2:0]color26;
        wire [2:0]color27;
        wire [2:0]color28;
        wire [2:0]color29;
        wire [2:0]color30;
        reg [10:0]address1=0;
        reg [10:0]address2=0;
```

```
reg [10:0]address3=0;
reg [10:0]address4=0;
reg [10:0]address5=0;
reg [10:0]address6=0;
reg [10:0]address7=0;
reg [10:0]address8=0;
reg [10:0]address9=0;
reg [10:0]address10=0;
reg [10:0]address11=0;
reg [10:0]address12=0;
reg [10:0]address13=0;
reg [10:0]address14=0;
reg [10:0]address15=0;
reg [10:0]address16=0;
reg [10:0]address17=0;
reg [10:0]address18=0;
reg [10:0]address19=0;
reg [10:0]address20=0;
reg [10:0]address21=0;
reg [10:0]address22=0;
reg [10:0]address23=0;
reg [10:0]address24=0;
reg [10:0]address25=0;
reg [10:0]address26=0;
reg [10:0]address27=0;
reg [10:0]address28=0;
reg [10:0]address29=0;
reg [10:0]address30=0;
A(address1, CLOCK_50, color1);
B(address2, CLOCK_50, color2);
C(address3, CLOCK_50, color3);
D(address4, CLOCK_50, color4);
E(address5, CLOCK_50, color5);
F(address6, CLOCK_50, color6);
G(address7, CLOCK_50, color7);
H(address8, CLOCK_50, color8);
I(address9, CLOCK_50, color9);
J(address10, CLOCK_50, color10);
K(address11, CLOCK_50, color11);
L(address12, CLOCK_50, color12);
M(address13, CLOCK_50, color13);
N(address14, CLOCK_50, color14);
O(address15, CLOCK_50, color15);
P(address16, CLOCK_50, color16);
Q(address17, CLOCK_50, color17);
R(address18, CLOCK_50, color18);
S(address19, CLOCK_50, color19);
```

```
        T(address20, CLOCK_50, color20);
        U(address21, CLOCK_50, color21);
        V(address22, CLOCK_50, color22);
        W(address23, CLOCK_50, color23);
        X(address24, CLOCK_50, color24);
        Y(address25, CLOCK_50, color25);
        Z(address26, CLOCK_50, color26);
        kongge(address27, CLOCK_50, color27);
        character(address28, CLOCK_50, color28);
        youwin(address29, CLOCK_50, color29);
        youlose(address30, CLOCK_50, color30);

        wire [4:0] num;
        fibonacci_lfsr_5bit one(~new_alpha, key, num);


        wire delay_0_25s;
        delay_1_8_sec delay0(CLOCK_50, delay_0_25s);

        reg [7:0] counter_x;
        reg [3:0] counter_y;
        reg [7:0] hex_alpha;
        integer shiftPix, EngX, EngY, counterX, counterPix, countTime, alpha_x, alpha_y, char_x_new,
char_x_old, char_y, count_move,
                                counter_char_x, counter_char_y, end_x, end_y, counter_end_x,
counter_end_y;
        reg win, lose, done_draw, done_move, delay_2s, done_erase, done_alphabet_draw,
new_alpha=1, done_change_x_index, done_end;
        reg [5:0] state, next_state;
        parameter               IDLE =          6'b000000,
                                START =         6'b000001,
                                DRAW =          6'b000010,
                                move=           6'b000011,
                                eraseEnergy =   6'b000100,
                                A =             6'b000101,
                                B =             6'b000110,
                                C =             6'b000111,
                                D =             6'b001000,
                                E =             6'b001001,
                                F =             6'b001010,
                                G =             6'b001011,
                                H =             6'b001100,
                                I =             6'b001101,
                                J =             6'b001110,
                                K =             6'b001111,
                                L =             6'b010000,
                                M =             6'b010001,
```

```
                    N =             6'b010010,
                    O =             6'b010011,
                    P =             6'b010100,
                    Q =             6'b010101,
                    R =             6'b010110,
                    S =             6'b010111,
                    T =             6'b011000,
                    U =             6'b011001,
                    V =             6'b011010,
                    W =             6'b011011,
                    X =             6'b011100,
                    Y =             6'b011101,
                    Z =             6'b011110,
                    space =         6'b011111,
                    WIN=            6'b100000,
                    LOSE=           6'b100001;

always@(*)
begin
        case(state)
                IDLE: if(last_data_received==8'h5A) next_state=START;
                        else next_state=IDLE;

                START: if(win) next_state=WIN;
                        else if(lose)  next_state=LOSE;
                        else if(new_alpha & num==5'b00001) next_state=A;
                        else if(new_alpha & num==5'b00010) next_state=B;
                        else if(new_alpha & num==5'b00011) next_state=C;
                        else if(new_alpha & num==5'b00100) next_state=D;
                        else if(new_alpha & num==5'b00101) next_state=E;
                        else if(new_alpha & num==5'b00110) next_state=F;
                        else if(new_alpha & num==5'b00111) next_state=G;
                        else if(new_alpha & num==5'b01000) next_state=H;
                        else if(new_alpha & num==5'b01001) next_state=I;
                        else if(new_alpha & num==5'b01010) next_state=J;
                        else if(new_alpha & num==5'b01011) next_state=K;
                        else if(new_alpha & num==5'b01100) next_state=L;
                        else if(new_alpha & num==5'b01101) next_state=M;
                        else if(new_alpha & num==5'b01110) next_state=N;
                        else if(new_alpha & num==5'b01111) next_state=O;
                        else if(new_alpha & num==5'b10000) next_state=P;
                        else if(new_alpha & num==5'b10001) next_state=Q;
                        else if(new_alpha & num==5'b10010) next_state=R;
                        else if(new_alpha & num==5'b10011) next_state=S;
                        else if(new_alpha & num==5'b10100) next_state=T;
                        else if(new_alpha & num==5'b10101) next_state=U;
                        else if(new_alpha & num==5'b10110) next_state=V;
```

```
                    else if(new_alpha & num==5'b10111) next_state=W;
                    else if(new_alpha & num==5'b11000) next_state=X;
                    else if(new_alpha & num==5'b11001) next_state=Y;
                    else if(new_alpha & num==5'b11010) next_state=Z;
                    else if(new_alpha & num==5'b11011 | num==5'b11100 |
num==5'b11101 | num==5'b11110 |  num==5'b11111 |  num==5'b00000) next_state=space;
                    else if(~new_alpha&&delay_0_25s) next_state=DRAW;
                    else next_state=START;

                DRAW: if(done_draw) next_state=move;
                      else next_state=DRAW;

                move: if(~done_move) next_state=move;
                     else if( delay_2s & done_move) next_state=eraseEnergy;
                     else if(~delay_2s & done_move) next_state=START;

                eraseEnergy: if(done_erase) next_state=START;
                             else next_state=eraseEnergy;

                A: if(done_alphabet_draw) next_state=START;
                   else next_state=A;

                B: if(done_alphabet_draw) next_state=START;
                   else next_state=B;

                C: if(done_alphabet_draw) next_state=START;
                  else next_state=C;

                D: if(done_alphabet_draw) next_state=START;
                   else next_state=D;

                E: if(done_alphabet_draw) next_state=START;
                  else next_state=E;

                F: if(done_alphabet_draw) next_state=START;
                  else next_state=F;

                G: if(done_alphabet_draw) next_state=START;
                   else next_state=G;

                H: if(done_alphabet_draw) next_state=START;
                 else next_state=H;

                I: if(done_alphabet_draw) next_state=START;
                 else next_state=I;

                J: if(done_alphabet_draw) next_state=START;
```

```
  else next_state=J;

K: if(done_alphabet_draw) next_state=START;
    else next_state=K;

L: if(done_alphabet_draw) next_state=START;
   else next_state=L;

M: if(done_alphabet_draw) next_state=START;
    else next_state=M;

N: if(done_alphabet_draw) next_state=START;
    else next_state=N;

O: if(done_alphabet_draw) next_state=START;
    else  next_state=O;

P: if(done_alphabet_draw) next_state=START;
   else next_state=P;

Q: if(done_alphabet_draw) next_state=START;
    else  next_state=Q;

R: if(done_alphabet_draw) next_state=START;
    else next_state=R;

S: if(done_alphabet_draw) next_state=START;
   else next_state=S;

T: if(done_alphabet_draw) next_state=START;
   else next_state=T;

U: if(done_alphabet_draw) next_state=START;
    else next_state=U;

V: if(done_alphabet_draw) next_state=START;
    else next_state=V;

W: if(done_alphabet_draw) next_state=START;
     else next_state=W;

X: if(done_alphabet_draw) next_state=START;
    else next_state=X;

Y: if(done_alphabet_draw) next_state=START;
    else  next_state=Y;
```

```
                Z: if(done_alphabet_draw) next_state=START;
                    else next_state=Z;

                space: if(done_alphabet_draw) next_state=START;
                        else next_state=space;

                WIN: next_state=WIN;

                LOSE: next_state=LOSE;

                default: next_state=IDLE;

        endcase
end

always@(posedge CLOCK_50)
begin
        if(state==IDLE)
        begin
                // start
                count_move=0;
                new_alpha=1;
                // alphabet
                alpha_x=75;
                alpha_y=100;
                counter_x=8'd0;
                counter_y=4'd0;
                done_alphabet_draw=0;
                address1=0;
                address2=0;
                address3=0;
                address4=0;
                address5=0;
                address6=0;
                address7=0;
                address8=0;
                address9=0;
                address10=0;
                address11=0;
                address12=0;
                address13=0;
                address14=0;
                address15=0;
                address16=0;
                address17=0;
                address18=0;
                address19=0;
```

```
        address20=0;
        address21=0;
        address22=0;
        address23=0;
        address24=0;
        address25=0;
        address26=0;
        address27=0;
        address28=0;
        // draw dash line
        shiftPix=0;
        done_draw=0;
        counterX=0;
        counterPix=0;
        // move charactor
        char_x_old=70;
        char_x_new=0;
        char_y=57;
        done_move=0;
        countTime=0;
        done_change_x_index=0;
        // erase energy bar
        delay_2s=0;
        EngX=0;
        EngY=0;
        done_erase=0;
        // win or lose
        win=0;
        lose=0;
        end_x=50;
        end_y=25;
        counter_end_x=0;
        counter_end_y=0;
        done_end=0;
end

if(state==START)
begin
        // alphabet
        alpha_x=75;
        alpha_y=100;
        counter_x=8'd0;
        counter_y=4'd0;
        done_alphabet_draw=0;
        address1=0;
        address2=0;
        address3=0;
```

```
            address4=0;
            address5=0;
            address6=0;
            address7=0;
            address8=0;
            address9=0;
            address10=0;
            address11=0;
            address12=0;
            address13=0;
            address14=0;
            address15=0;
            address16=0;
            address17=0;
            address18=0;
            address19=0;
            address20=0;
            address21=0;
            address22=0;
            address23=0;
            address24=0;
            address25=0;
            address26=0;
            address27=0;
            address28=0;
            // draw dash line
            done_draw=0;
            counterX=0;
            counterPix=0;
            // move charactor
            char_y=57;
            done_move=0;
            done_change_x_index=0;
            // erase energy bar
            delay_2s=0;
            EngY=0;
            done_erase=0;

            if(last_data_received==hex_alpha)
            begin
                  new_alpha=1;
                  count_move=count_move+1;
            end
end

if(state==A)
begin
```

```
		hex_alpha=8'h1C;
		new_alpha=0;
		if(counter_x<10)
		begin
			if(counter_y==10)
			begin
				done_alphabet_draw=1;
				counter_x=0;
				counter_y=0;
				address1=0;
				alpha_x=75;
				alpha_y=100;
			end
			else
			begin
				write<=1;
				alpha_x=alpha_x+1;
				x_out<=alpha_x;
				y_out<=alpha_y;
				counter_x=counter_x+1;
				c_out<=color1;
				address1=address1+1;
			end
		end
		else
		begin
			alpha_x=75;
			counter_x=0;
			alpha_y=alpha_y+1;
			counter_y=counter_y+1;
			write<=0;
		end
end

if(state==B)
begin
		hex_alpha=8'h32;
		new_alpha=0;
		if(counter_x<10)
		begin
			if(counter_y==10)
			begin
				done_alphabet_draw=1;
				counter_x=0;
				counter_y=0;
				alpha_x=75;
				alpha_y=100;
```

```
                address2=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color2;
                address2=address2+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
end

if(state==C)
begin
        hex_alpha=8'h21;
        new_alpha=0;
        if(counter_x<10)
        begin
            if(counter_y==10)
            begin
                done_alphabet_draw=1;
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address3=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color3;
                address3=address3+1;
```

```
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==D)
begin
        hex_alpha=8'h23;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address4=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color4;
                        address4=address4+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end
```

```
if(state==E)
begin
	hex_alpha=8'h24;
	new_alpha=0;
	if(counter_x<10)
	begin
		if(counter_y==10)
		begin
			done_alphabet_draw=1;
			counter_x=0;
			counter_y=0;
			alpha_x=75;
			alpha_y=100;
			address5=0;
		end
		else
		begin
			write<=1;
			alpha_x=alpha_x+1;
			x_out<=alpha_x;
			y_out<=alpha_y;
			counter_x=counter_x+1;
			c_out<=color5;
			address5=address5+1;
		end
	end
	else
	begin
		alpha_x=75;
		counter_x=0;
		alpha_y=alpha_y+1;
		counter_y=counter_y+1;
		write<=0;
	end
end

if(state==F)
begin
	hex_alpha=8'h2B;
	new_alpha=0;
	if(counter_x<10)
	begin
		if(counter_y==10)
		begin
			done_alphabet_draw=1;
			counter_x=0;
```

```
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address6=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color6;
                address6=address6+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
end

if(state==G)
begin
        hex_alpha=8'h34;
        new_alpha=0;
        if(counter_x<10)
        begin
            if(counter_y==10)
            begin
                done_alphabet_draw=1;
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address7=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
```

```
                counter_x=counter_x+1;
                c_out<=color7;
                address7=address7+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
end

if(state==H)
begin
        hex_alpha=8'h33;
        new_alpha=0;
        if(counter_x<10)
        begin
            if(counter_y==10)
            begin
                done_alphabet_draw=1;
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address8=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color8;
                address8=address8+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
```

```
                write<=0;
        end
end

if(state==I)
begin
        hex_alpha=8'h43;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address9=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color9;
                        address9=address9+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==J)
begin
        hex_alpha=8'h3B;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
```

```
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address10=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color10;
                        address10=address10+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==K)
begin
        hex_alpha=8'h42;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address11=0;
                end
                else
                begin
                        write<=1;
```

```
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color11;
                address11=address11+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
end

if(state==L)
begin
        hex_alpha=8'h4B;
        new_alpha=0;
        if(counter_x<10)
        begin
            if(counter_y==10)
            begin
                done_alphabet_draw=1;
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address12=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color12;
                address12=address12+1;
            end
        end
        else
        begin
            alpha_x=75;
```

```
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==M)
begin
        hex_alpha=8'h3A;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address13=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color13;
                        address13=address13+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==N)
begin
        hex_alpha=8'h31;
        new_alpha=0;
```

```
		if(counter_x<10)
		begin
			if(counter_y==10)
			begin
				done_alphabet_draw=1;
				counter_x=0;
				counter_y=0;
				alpha_x=75;
				alpha_y=100;
				address14=0;
			end
			else
			begin
				write<=1;
				alpha_x=alpha_x+1;
				x_out<=alpha_x;
				y_out<=alpha_y;
				counter_x=counter_x+1;
				c_out<=color14;
				address14=address14+1;
			end
		end
		else
		begin
			alpha_x=75;
			counter_x=0;
			alpha_y=alpha_y+1;
			counter_y=counter_y+1;
			write<=0;
		end
end

if(state==O)
begin
		hex_alpha=8'h44;
		new_alpha=0;
		if(counter_x<10)
		begin
			if(counter_y==10)
			begin
				done_alphabet_draw=1;
				counter_x=0;
				counter_y=0;
				alpha_x=75;
				alpha_y=100;
				address15=0;
			end
```

```
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color15;
                        address15=address15+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==P)
begin
        hex_alpha=8'h4D;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address16=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color16;
                        address16=address16+1;
                end
        end
```

```
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==Q)
begin
        hex_alpha=8'h15;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address17=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color17;
                        address17=address17+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==R)
```

```
begin
	hex_alpha=8'h2D;
	new_alpha=0;
	if(counter_x<10)
	begin
		if(counter_y==10)
		begin
			done_alphabet_draw=1;
			counter_x=0;
			counter_y=0;
			alpha_x=75;
			alpha_y=100;
			address18=0;
		end
		else
		begin
			write<=1;
			alpha_x=alpha_x+1;
			x_out<=alpha_x;
			y_out<=alpha_y;
			counter_x=counter_x+1;
			c_out<=color18;
			address18=address18+1;
		end
	end
	else
	begin
		alpha_x=75;
		counter_x=0;
		alpha_y=alpha_y+1;
		counter_y=counter_y+1;
		write<=0;
	end
end

if(state==S)
begin
	hex_alpha=8'h1B;
	new_alpha=0;
	if(counter_x<10)
	begin
		if(counter_y==10)
		begin
			done_alphabet_draw=1;
			counter_x=0;
			counter_y=0;
			alpha_x=75;
```

```
                alpha_y=100;
                address19=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color19;
                address19=address19+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
end

if(state==T)
begin
        hex_alpha=8'h2C;
        new_alpha=0;
        if(counter_x<10)
        begin
            if(counter_y==10)
            begin
                done_alphabet_draw=1;
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address20=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color20;
```

```
                address20=address20+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
end

if(state==U)
begin
        hex_alpha=8'h3C;
        new_alpha=0;
        if(counter_x<10)
        begin
            if(counter_y==10)
            begin
                done_alphabet_draw=1;
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address21=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color21;
                address21=address21+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
```

```
end

if(state==V)
begin
	hex_alpha=8'h2A;
	new_alpha=0;
	if(counter_x<10)
	begin
		if(counter_y==10)
		begin
			done_alphabet_draw=1;
			counter_x=0;
			counter_y=0;
			alpha_x=75;
			alpha_y=100;
			address22=0;
		end
		else
		begin
			write<=1;
			alpha_x=alpha_x+1;
			x_out<=alpha_x;
			y_out<=alpha_y;
			counter_x=counter_x+1;
			c_out<=color22;
			address22=address22+1;
		end
	end
	else
	begin
		alpha_x=75;
		counter_x=0;
		alpha_y=alpha_y+1;
		counter_y=counter_y+1;
		write<=0;
	end
end

if(state==W)
begin
	hex_alpha=8'h1D;
	new_alpha=0;
	if(counter_x<10)
	begin
		if(counter_y==10)
		begin
			done_alphabet_draw=1;
```

```
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address23=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
                y_out<=alpha_y;
                counter_x=counter_x+1;
                c_out<=color23;
                address23=address23+1;
            end
        end
        else
        begin
            alpha_x=75;
            counter_x=0;
            alpha_y=alpha_y+1;
            counter_y=counter_y+1;
            write<=0;
        end
end

if(state==X)
begin
        hex_alpha=8'h22;
        new_alpha=0;
        if(counter_x<10)
        begin
            if(counter_y==10)
            begin
                done_alphabet_draw=1;
                counter_x=0;
                counter_y=0;
                alpha_x=75;
                alpha_y=100;
                address24=0;
            end
            else
            begin
                write<=1;
                alpha_x=alpha_x+1;
                x_out<=alpha_x;
```

```
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color24;
                        address24=address24+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==Y)
begin
        hex_alpha=8'h35;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address25=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color25;
                        address25=address25+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
```

```
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==Z)
begin
        hex_alpha=8'h1A;
        new_alpha=0;
        if(counter_x<10)
        begin
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address26=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color26;
                        address26=address26+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==space)
begin
        hex_alpha=8'h29;
        new_alpha=0;
        if(counter_x<10)
        begin
```

```
                if(counter_y==10)
                begin
                        done_alphabet_draw=1;
                        counter_x=0;
                        counter_y=0;
                        alpha_x=75;
                        alpha_y=100;
                        address27=0;
                end
                else
                begin
                        write<=1;
                        alpha_x=alpha_x+1;
                        x_out<=alpha_x;
                        y_out<=alpha_y;
                        counter_x=counter_x+1;
                        c_out<=color27;
                        address27=address27+1;
                end
        end
        else
        begin
                alpha_x=75;
                counter_x=0;
                alpha_y=alpha_y+1;
                counter_y=counter_y+1;
                write<=0;
        end
end

if(state==DRAW)
begin
        write<=1;
        y_out<=7'b1001111;
        counterX= counterX+1;
        counterPix= counterPix+1;
        if(counterX<=160)
        begin
                if(shiftPix==0)
                begin
                        if((counterPix>0)&&(counterPix<9))
                        begin
                                x_out<=counterX-1;
                                c_out<=3'b110;
                        end
                        else if((counterPix>8)&&(counterPix<16))
                        begin
```

```
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if(counterPix==16)
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
                counterPix=0;
        end
end
else if(shiftPix==1)
begin
        if((counterPix>0)&&(counterPix<8))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if((counterPix>7)&&(counterPix<16))
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if(counterPix==16)
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
                counterPix=0;
        end
end
else if(shiftPix==2)
begin
        if((counterPix>0)&&(counterPix<7))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if((counterPix>6)&&(counterPix<15))
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if(counterPix==15)
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if(counterPix==16)
```

```
            begin
                x_out<=counterX-1;
                c_out<=3'b110;
                counterPix=0;
            end
        end
        else if(shiftPix==3)
        begin
            if((counterPix>0)&&(counterPix<6))
            begin
                x_out<=counterX-1;
                c_out<=3'b110;
            end
            else if((counterPix>5)&&(counterPix<14))
            begin
                x_out<=counterX-1;
                c_out<=3'b000;
            end
            else if((counterPix>13)&&(counterPix<16))
            begin
                x_out<=counterX-1;
                c_out<=3'b110;
            end
            else if(counterPix==16)
            begin
                x_out<=counterX-1;
                c_out<=3'b110;
                counterPix=0;
            end
        end
        else if(shiftPix==4)
        begin
            if((counterPix>0)&&(counterPix<5))
            begin
                x_out<=counterX-1;
                c_out<=3'b110;
            end
            else if((counterPix>4)&&(counterPix<13))
            begin
                x_out<=counterX-1;
                c_out<=3'b000;
            end
            else if((counterPix>12)&&(counterPix<16))
            begin
                x_out<=counterX-1;
                c_out<=3'b110;
            end
```

```
        else if(counterPix==16)
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
                counterPix=0;
        end
end
else if(shiftPix==5)
begin
        if((counterPix>0)&&(counterPix<4))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if((counterPix>3)&&(counterPix<12))
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if((counterPix>11)&&(counterPix<16))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if(counterPix==16)
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
                counterPix=0;
        end
end
else if(shiftPix==6)
begin
        if((counterPix>0)&&(counterPix<3))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if((counterPix>2)&&(counterPix<11))
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if((counterPix>10)&&(counterPix<16))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
```

```
		end
		else if(counterPix==16)
		begin
			x_out<=counterX-1;
			c_out<=3'b110;
			counterPix=0;
		end
	end
	else if(shiftPix==7)
	begin
		if(counterPix==1)
		begin
			x_out<=counterX-1;
			c_out<=3'b110;
		end
		else if((counterPix>1)&&(counterPix<10))
		begin
			x_out<=counterX-1;
			c_out<=3'b000;
		end
		else if((counterPix>9)&&(counterPix<16))
		begin
			x_out<=counterX-1;
			c_out<=3'b110;
		end
		else if(counterPix==16)
		begin
			x_out<=counterX-1;
			c_out<=3'b110;
			counterPix=0;
		end
	end
	else if(shiftPix==8)
	begin
		if((counterPix>0)&&(counterPix<9))
		begin
			x_out<=counterX-1;
			c_out<=3'b000;
		end
		else if((counterPix>8)&&(counterPix<16))
		begin
			x_out<=counterX-1;
			c_out<=3'b110;
		end
		else if(counterPix==16)
		begin
			x_out<=counterX-1;
```

```
                c_out<=3'b110;
                counterPix=0;
        end
end
else if(shiftPix==9)
begin
        if((counterPix>0)&&(counterPix<8))
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if((counterPix>7)&&(counterPix<16))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if(counterPix==16)
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
                counterPix=0;
        end
end
else if(shiftPix==10)
begin
        if((counterPix>0)&&(counterPix<7))
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if((counterPix>6)&&(counterPix<15))
        begin
                x_out<=counterX-1;
                c_out<=3'b110;
        end
        else if(counterPix==15)
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
        end
        else if(counterPix==16)
        begin
                x_out<=counterX-1;
                c_out<=3'b000;
                counterPix=0;
        end
end
```

```
else if(shiftPix==11)
begin
	if((counterPix>0)&&(counterPix<6))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if((counterPix>5)&&(counterPix<14))
	begin
		x_out<=counterX-1;
		c_out<=3'b110;
	end
	else if((counterPix>13)&&(counterPix<16))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if(counterPix==16)
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
		counterPix=0;
	end
end
else if(shiftPix==12)
begin
	if((counterPix>0)&&(counterPix<5))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if((counterPix>4)&&(counterPix<13))
	begin
		x_out<=counterX-1;
		c_out<=3'b110;
	end
	else if((counterPix>12)&&(counterPix<16))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if(counterPix==16)
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
		counterPix=0;
	end
```

```
end
else if(shiftPix==13)
begin
	if((counterPix>0)&&(counterPix<4))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if((counterPix>3)&&(counterPix<12))
	begin
		x_out<=counterX-1;
		c_out<=3'b110;
	end
	else if((counterPix>11)&&(counterPix<16))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if(counterPix==16)
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
		counterPix=0;
	end
end
else if(shiftPix==14)
begin
	if((counterPix>0)&&(counterPix<3))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if((counterPix>2)&&(counterPix<11))
	begin
		x_out<=counterX-1;
		c_out<=3'b110;
	end
	else if((counterPix>10)&&(counterPix<16))
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
	end
	else if(counterPix==16)
	begin
		x_out<=counterX-1;
		c_out<=3'b000;
		counterPix=0;
```

```
                end
            end
            else if(shiftPix==15)
            begin
                if(counterPix==1)
                begin
                    x_out<=counterX-1;
                    c_out<=3'b000;
                end
                else if((counterPix>1)&&(counterPix<10))
                begin
                    x_out<=counterX-1;
                    c_out<=3'b110;
                end
                else if((counterPix>9)&&(counterPix<16))
                begin
                    x_out<=counterX-1;
                    c_out<=3'b000;
                end
                else if(counterPix==16)
                begin
                    x_out<=counterX-1;
                    c_out<=3'b000;
                    counterPix=0;
                end
            end
        end
        if(counterX==161)
        begin
            counterX=0;
            done_draw=1;
            write<=0;
            if(shiftPix==15)
            begin
                shiftPix=0;
            end
            else
            begin
                shiftPix=shiftPix+1;
            end
end
end

if(state==move)
begin
        if(count_move==0)
        begin
```

```
if(~done_change_x_index)
begin
	if(char_x_old==22)
	begin
		lose=1;
		done_change_x_index=1;
		done_move=1;
	end
	else
	begin
		done_change_x_index=1;
		char_x_old=char_x_old-1;
		char_x_new=char_x_old;
		write<=1;
	end
end
if(counter_char_x<21)
begin
	if(counter_char_y==21)
	begin
		done_move=1;
		done_change_x_index=0;
		write<=0;
		char_y=57;
		count_move=0;
		counter_char_x=0;
		counter_char_y=0;
		countTime=countTime+1;
		if(countTime==2)
		begin
			delay_2s=1;
		end

	end
	else
	begin
		char_x_new=char_x_new+1;
		x_out<=char_x_new;
		y_out<=char_y;
		counter_char_x=counter_char_x+1;
		c_out<=color28;
		address28=address28+1;
	end
end
else
begin
	char_x_new=char_x_old;
```

```
                counter_char_x=0;
                char_y=char_y+1;
                counter_char_y=counter_char_y+1;
        end
end
else if(count_move==1)
begin
        count_move=0;
        done_move=1;
        write<=0;
        countTime=countTime+1;
        if(countTime==2)
        begin
                delay_2s=1;
        end
end
else if(count_move==2)
begin
        if(~done_change_x_index)
        begin
                if(char_x_old==140)
                begin
                        win=1;
                        done_change_x_index=1;
                        done_move=1;
                end
                else
                begin
                        done_change_x_index=1;
                        char_x_old=char_x_old+2;
                        char_x_new=char_x_old;
                        write<=1;
                end
        end
        if(counter_char_x<21)
        begin
                if(counter_char_y==21)
                begin
                        count_move=0;
                        done_change_x_index=0;
                        done_move=1;
                        write<=0;
                        char_y=57;
                        counter_char_x=0;
                        counter_char_y=0;
                        countTime=countTime+1;
                        if(countTime==2)
```

```
                        begin
                                delay_2s=1;
                        end
                        if(char_x_old==22)
                        begin
                                lose=1;
                                done_change_x_index=1;
                                done_move=1;
                        end
                        else if(char_x_old==140)
                        begin
                                win=1;
                                done_change_x_index=1;
                                done_move=1;
                        end
                end
                else
                begin
                        write<=1;
                        char_x_new=char_x_new+1;
                        x_out<=char_x_new;
                        y_out<=char_y;
                        counter_char_x=counter_char_x+1;
                        c_out<=color28;
                        address28=address28+1;
                end
        end
        else
        begin
                char_x_new=char_x_old;
                counter_char_x=0;
                char_y=char_y+1;
                counter_char_y=counter_char_y+1;
                write<=0;
        end
end
else if(count_move>=3)
begin
        if(~done_change_x_index)
        begin
                if(char_x_old==140)
                begin
                        win=1;
                        done_change_x_index=1;
                        done_move=1;
                end
                else
```

```
        begin
                done_change_x_index=1;
                char_x_old=char_x_old+4;
                char_x_new=char_x_old;
                write<=1;
        end
end
if(counter_char_x<21)
begin
        if(counter_char_y==21)
        begin
                count_move=0;
                done_change_x_index=0;
                done_move=1;
                write<=0;
                char_y=57;
                counter_char_x=0;
                counter_char_y=0;
                countTime=countTime+1;
                if(countTime==2)
                begin
                        delay_2s=1;
                end
                if(char_x_old==22)
                begin
                        lose=1;
                        done_change_x_index=1;
                        done_move=1;
                end
                else if(char_x_old==140)
                begin
                        win=1;
                        done_change_x_index=1;
                        done_move=1;
                end
        end
        else
        begin
                write<=1;
                char_x_new=char_x_new+1;
                x_out<=char_x_new;
                y_out<=char_y;
                counter_char_x=counter_char_x+1;
                c_out<=color28;
                address28=address28+1;
        end
end
```

```
                else
                begin
                        char_x_new=char_x_old;
                        counter_char_x=0;
                        char_y=char_y+1;
                        counter_char_y=counter_char_y+1;
                        write<=0;
                end
        end
end

if(state==eraseEnergy)
begin
        countTime=0;
        done_move=0;
        delay_2s=0;
        write<=1;
        EngY=EngY+1;
        if(EngY==1)
        begin
                y_out<=7'b0000100;
                x_out<=EngX;
                c_out<=3'b000;
        end
        else if(EngY==2)
        begin
                y_out<=7'b0000000;
                x_out<=EngX;
                c_out<=3'b000;
        end
        else if(EngY==3)
        begin
                y_out<=7'b0000001;
                x_out<=EngX;
                c_out<=3'b000;
        end
        else if(EngY==4)
        begin
                y_out<=7'b0000010;
                x_out<=EngX;
                c_out<=3'b000;
        end
        else if(EngY==5)
        begin
                y_out<=7'b0000011;
                x_out<=EngX;
                c_out<=3'b000;
```

```
		end
		else if(EngY==6)
		begin
			EngY=0;
			EngX=EngX+1;
			done_erase=1;
		end
		if(EngX==161)
		begin
			lose=1;
		end
end

if(state==WIN)
begin
		if(~done_end)
		begin
			if(counter_end_x<78)
			begin
				if(counter_end_y==26)
				begin
					done_end=1;
				end
				else
				begin
					write<=1;
					end_x=end_x+1;
					x_out<=end_x;
					y_out<=end_y;
					counter_end_x=counter_end_x+1;
					c_out<=color29;
					address29=address29+1;
				end
			end
			else
			begin
				end_x=50;
				counter_end_x=0;
				end_y=end_y+1;
				counter_end_y=counter_end_y+1;
				write<=0;
			end
		end
end

if(state==LOSE)
begin
```

```
                // display you lost
                if(~done_end)
                begin
                        if(counter_end_x<78)
                        begin
                                if(counter_end_y==26)
                                begin
                                        done_end=1;
                                end
                                else
                                begin
                                        write<=1;
                                        end_x=end_x+1;
                                        x_out<=end_x;
                                        y_out<=end_y;
                                        counter_end_x=counter_end_x+1;
                                        c_out<=color30;
                                        address30=address30+1;
                                end
                        end
                        else
                        begin
                                end_x=50;
                                counter_end_x=0;
                                end_y=end_y+1;
                                counter_end_y=counter_end_y+1;
                                write<=0;
                        end
                end
        end

        state <= next_state;
end

wire            [7:0]   ps2_key_data;
wire                    ps2_key_pressed;
reg                     [7:0]   last_data_received=0;

always @(posedge CLOCK_50)
begin
        if (ps2_key_pressed == 1'b1)
                last_data_received <= ps2_key_data;
end

PS2_Controller PS2 (
        // Inputs
        .CLOCK_50                               (CLOCK_50),
```

```
	// Bidirectionals
	.PS2_CLK			(PS2_CLK),
	.PS2_DAT			(PS2_DAT),

	// Outputs
	.received_data		(ps2_key_data),
	.received_data_en	(ps2_key_pressed)
);

endmodule



module delay_1_8_sec(CLOCK_50, enable);
	input CLOCK_50;
	output reg enable;
	reg [31:0] count;

	always @(posedge CLOCK_50)
	begin
		if(count == 32'd19_999_999) //0.25s		//  //15000000
		begin
			count <= 0;			//32'd0;
			enable <= 1;
		end
		else
		begin
			count <= count + 1;
			enable <= 0;
		end
	end
endmodule



module fibonacci_lfsr_5bit(input clk, input rst_n,output reg [4:0] data);
	reg [4:0] data_next;

	always @(*)
	begin
	  data_next[4] = data[4]^data[1];
	  data_next[3] = data[3]^data_next[0];
	  data_next[2] = data[2]^data_next[4];
	  data_next[1] = data[1]^data_next[3];
	  data_next[0] = data[0]^data_next[2];
```

```
        end

        always @(posedge clk or negedge rst_n)
  begin
                if(!rst_n)
                        data <= 5'h1f;
                else
                        data <= data_next;
        end

endmodule
```