**Northwestern University** **Mahim Raj Gupta**
**EECS355: ASIC and FPGA** **Jenifer JacobNisha**

# FINAL PROJECT REPORT

**Objective:**
The objective of this project is to a game called “tank duel” that could be displayed on the VGA monitor and controlled by two players by one PS2 keyboard.

**Design:**
The tanks are realized as square boxes on the VGA monitor. They keep continuously moving back and forth on a horizontal line on the screen. Each tank can move with three different speeds. The direction of the tanks can be controlled using left and right keys. Both tanks shoot one bullet at a time; if it hits the opponent’s tank, it scores one point. A tank can shoot the second bullet only when the previous bullet goes out of the screen or hits the opponent. The score of both players is displayed on the LEDs. Whoever first scores 3 points wins. When a player wins, only the winner’s tank is displayed and the winner message is displayed in the LCD screen. The game resets on pressing the reset button.

**Realization:**

**VGA_top_level:** This is the top level entity that structurally connects the VGA, LED and LCD components that display the output.

**lcd :** The lcd component maps de2lcd component and displays the winner message based on a_win and b_win values.

**leddcd:** This component decodes the hexadecimal score into seven-segment display for both the tanks.

**pixelGenerator** : The provided pixelGenerator component was modified to map ps2 component to read keyboard press values, decode them accordingly and make changes in the pixel values sent to the VGA control. To indentify keyboard values, scan_readyo, scan_code and hist1 signals were compared. As the clock speed is too high for human eye perception, counters were used to reduce the speed at which the bullet moves.

**procedure:** MY package was used to store two procedures SQ and SQ_B that display the tanks and bullet respectively.

**NOTE:** Other components such as oneshot, colorROM, ps2, keyboard, VGA_SYNC and leddcd that were provided were not modified.

**VGA_top_level**

```
library IEEE;

use IEEE.std_logic_1164.all;

entityVGA_top_level is
	port(
			CLOCK_50 : in std_logic;
			RESET_N : in std_logic;
			keyboard_clk, keyboard_data: in std_logic;
			--VGA
			VGA_RED, VGA_GREEN, VGA_BLUE : out std_logic_vector(9 downto 0);
			HORIZ_SYNC, VERT_SYNC, VGA_BLANK, VGA_CLK : out std_logic;
			score_tank1_hex : out std_logic_vector(6 downto 0);
			score_tank2_hex : out std_logic_vector(6 downto 0);
			--lcd
			res_led : in std_logic;
			LCD_RS, LCD_E, LCD_ON, RESET_LED, SEC_LED : OUT STD_LOGIC;
		LCD_RW : BUFFER STD_LOGIC;
		DATA_BUS : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0)
	);
end entity VGA_top_level;

architecture structural of VGA_top_level is

componentpixelGenerator is
	port(
			clk, ROM_clk, rst_n, video_on, eof : in std_logic;
			pixel_row, pixel_column : in std_logic_vector(9 downto 0);
			keyboard_clk, keyboard_data : in std_logic;
			red_out, green_out, blue_out : out std_logic_vector(9 downto 0);
			score_tank1_hex : out std_logic_vector(6 downto 0);
			score_tank2_hex : out std_logic_vector(6 downto 0);
			tank1_win : buffer std_logic;
			tank2_win : buffer std_logic
	);
end component pixelGenerator;

component VGA_SYNC is
	port(
			clock_50Mhz : in std_logic;
			horiz_sync_out, vert_sync_out,
			video_on, pixel_clock, eof : out std_logic;
			pixel_row, pixel_column : out std_logic_vector(9 downto 0)
	);
end component VGA_SYNC;

component de2lcd IS
	PORT(reset, clk_50Mhz : IN STD_LOGIC;
		a_win, b_win : in std_logic;
		LCD_RS, LCD_E, LCD_ON, RESET_LED, SEC_LED : OUT STD_LOGIC;
		LCD_RW : BUFFER STD_LOGIC;
		DATA_BUS : INOUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END component de2lcd;

--Signals for VGA sync
signalpixel_row_int : std_logic_vector(9 downto 0);
signalpixel_column_int : std_logic_vector(9 downto 0);
signalvideo_on_int : std_logic;
signalVGA_clk_int : std_logic;
signaleof : std_logic;

--lcd
signal tank1_win, tank2_win : std_logic;

begin
```

```
---------------------------------------------------------------------------------------------

        videoGen :pixelGenerator
                port map(CLOCK_50, VGA_clk_int, RESET_N, video_on_int, eof, pixel_row_int, pixel_column_int,keyboard_clk,
keyboard_data,VGA_RED, VGA_GREEN, VGA_BLUE,score_tank1_hex,score_tank2_hex, tank1_win, tank2_win);

---------------------------------------------------------------------------------------------
--This section should not be modified in your design.  This section handles the VGA timing signals
--and outputs the current row and column.  You will need to redesign the pixelGenerator to choose
--the color value to output based on the current position
        videoSync : VGA_SYNC
                port map(CLOCK_50, HORIZ_SYNC, VERT_SYNC, video_on_int, VGA_clk_int, eof, pixel_row_int, pixel_column_int);

        VGA_BLANK <= video_on_int;

        VGA_CLK <= VGA_clk_int;

---------------------------------------------------------------------------------------------

lcd_map : de2lcd
        port map(res_led, cloCK_50, tank1_win, tank2_win, LCD_RS, LCD_E, LCD_ON, RESET_LED, SEC_LED, LCD_RW, DATA_BUS);

end architecture structural;
```

**procedure.vhd**

```
libraryieee;
use ieee.std_logic_1164.all;
useieee.numeric_std.all;

PACKAGE MY IS
PROCEDURE SQ(SIGNAL Xcur,Ycur,Xpos,Ypos: IN INTEGER;SIGNAL colorAddr:OUT STD_LOGIC_VECTOR(2 downto 0);SIGNAL DRAW: OUT STD_LOGIC);
PROCEDURE SQ_B(SIGNAL Xcur_B,Ycur_B,Xpos_B,Ypos_B: IN INTEGER;SIGNAL colorAddr_B:OUT STD_LOGIC_VECTOR(2 downto 0);SIGNAL DRAW_B: OUT
STD_LOGIC);
END MY;

PACKAGE BODY MY IS
PROCEDURE SQ(SIGNAL Xcur,Ycur,Xpos,Ypos: IN INTEGER;SIGNAL colorAddr:OUT STD_LOGIC_VECTOR(2 downto 0);SIGNAL DRAW: OUT STD_LOGIC) IS
BEGIN
IF(Xcur>Xpos AND Xcur<(Xpos+50) AND Ycur>Ypos AND Ycur<(Ypos+50))THEN
colorAddr<="111";
  DRAW<='1';
  ELSE
  DRAW<='0';
END IF;
END SQ;

PROCEDURE SQ_B(SIGNAL Xcur_B,Ycur_B,Xpos_B,Ypos_B: IN INTEGER;SIGNAL colorAddr_B:OUT STD_LOGIC_VECTOR(2 downto 0);SIGNAL DRAW_B: OUT
STD_LOGIC) IS
BEGIN
IF(Xcur_B>Xpos_B AND Xcur_B<(Xpos_B+15) AND Ycur_B>Ypos_B AND Ycur_B<(Ypos_B+15))THEN
colorAddr_B<="000";
  DRAW_B<='1';
  ELSE
  DRAW_B<='0';
END IF;
END SQ_B;

END MY;
```

**pixelGenerator.vhd**

```
library IEEE;
use IEEE.std_logic_1164.all;
useIEEE.numeric_std.all;
usework.my.all;
```

```
entitypixelGenerator is
	port(
			clk, ROM_clk, rst_n, video_on, eof		: in std_logic;
			pixel_row, pixel_column		: in std_logic_vector(9 downto 0);
			keyboard_clk, keyboard_data		: in std_logic;
			red_out, green_out, blue_out		: out std_logic_vector(9 downto 0);
			score_tank1_hex : out std_logic_vector(6 downto 0);
			score_tank2_hex : out std_logic_vector(6 downto 0);
			tank1_win : buffer std_logic;
			tank2_win : buffer std_logic
		);
end entity pixelGenerator;

architecture behavioral of pixelGenerator is

constantcolor_red		: std_logic_vector(2 downto 0) := "000";
constantcolor_green		: std_logic_vector(2 downto 0) := "001";
constantcolor_blue	: std_logic_vector(2 downto 0) := "010";
constantcolor_yellow		: std_logic_vector(2 downto 0) := "011";
constantcolor_magenta		: std_logic_vector(2 downto 0) := "100";
constantcolor_cyan : std_logic_vector(2 downto 0) := "101";
constantcolor_black		: std_logic_vector(2 downto 0) := "110";
constantcolor_white		: std_logic_vector(2 downto 0) := "111";

componentcolorROM is
	port
	(
		address		: in std_logic_vector (2 downto 0);
		clock		: in std_logic  := '1';
		q			: out std_logic_vector (29 downto 0)
	);
end component colorROM;

component ps2 is
	port(	keyboard_clk, keyboard_data, clock_50MHz ,
			reset : in std_logic;--, read : in std_logic;
			scan_code : out std_logic_vector( 7 downto 0 );
			scan_readyo : out std_logic;
			hist3 : out std_logic_vector(7 downto 0);
			hist2 : out std_logic_vector(7 downto 0);
			hist1 : out std_logic_vector(7 downto 0);
			hist0 : out std_logic_vector(7 downto 0)
		);
end component ps2;
componentleddcd is
	port(
		data_in : in std_logic_vector(3 downto 0);
		segments_out : out std_logic_vector(6 downto 0)
		);
end component leddcd;

signalcolorAddress : std_logic_vector (2 downto 0);
signal colorAddr1 : std_logic_vector (2 downto 0);
signal colorAddr2 : std_logic_vector (2 downto 0);
signalcolorAddr_B : std_logic_vector (2 downto 0);
signal color	: std_logic_vector (29 downto 0);
signal DRAW1		: std_logic;
signal DRAW2		: std_logic;
signal DRAW_B		: std_logic;
signal SQ_X1 : integer := 10;
signal SQ_Y1 : integer := 250;
signal SQ_X2 : integer := 420;
signal SQ_Y2 : integer := 250;
signal SQ_B_X3 : natural := 420;
signal SQ_B_Y3 : natural	:= 0;
signal SQ_B_X4 : natural  := 10;
signal SQ_B_Y4 : natural := 0;
```

```
signalpixel_row_int, pixel_column_int : natural;
signalscan_code_signal : std_logic_vector(7 downto 0);
signalscan_readyo_signal : std_logic;
signal hist3_signal : std_logic_vector(7 downto 0);
signal hist2_signal : std_logic_vector(7 downto 0);
signal hist1_signal : std_logic_vector(7 downto 0);
signal hist0_signal : std_logic_vector(7 downto 0);
signal colorAddr_B1 : std_logic_vector (2 downto 0);
signal DRAW_B1 : std_logic;
signal count_tank1 : natural :=5000000;
signal count_tank2 : natural :=5000000;
signal count_tank1_mov : natural :=5000000;
signal count_tank2_mov : natural :=5000000;
signal shoot_done_tank2 : integer:=0;
--signal lock_tank2 : integer :=0;
signal shoot_done_tank1 : integer:=0;
--signal lock_tank1 : integer :=0;
signal score_tank1 : std_logic_vector(3 downto 0);
signal score_tank2 : std_logic_vector(3 downto 0);
signal score_tank1_int : natural range 0 to 3:=0;
signal score_tank2_int : natural range 0 to 3:=0;
signalcount_delay : natural:=0;
signal direc_ltor_tank2: std_logic;
signal direc_ltor_tank1: std_logic;
signalcount_win : integer := 0;
begin
--------------------------------------------------------------------------------------------
		red_out<= color(29 downto 20);
		green_out<= color(19 downto 10);
		blue_out<= color(9 downto 0);

		pixel_row_int<= to_integer(unsigned(pixel_row));
		pixel_column_int<= to_integer(unsigned(pixel_column));

--------------------------------------------------------------------------------------------

		colors :colorROM
			port map(colorAddress, ROM_clk, color);

--------------------------------------------------------------------------------------------
leddcd_map_tank1 :leddcd port map(score_tank1,score_tank1_hex);
leddcd_map_tank2 :leddcd port map(score_tank2,score_tank2_hex);
keyboard_map : ps2 port map(keyboard_clk, keyboard_data, clk
,rst_n,scan_code_signal,scan_readyo_signal,hist3_signal,hist2_signal,hist1_signal,hist0_signal);
SQ(pixel_row_int,pixel_column_int,SQ_X1,SQ_Y1,colorAddr1,DRAW1); --tank1
SQ(pixel_row_int,pixel_column_int,SQ_X2,SQ_Y2,colorAddr2,DRAW2);		--tank2
SQ_B(pixel_row_int,pixel_column_int,SQ_B_X3,SQ_B_Y3,colorAddr_B,DRAW_B);--bullet for tank 2
SQ_B(pixel_row_int,pixel_column_int,SQ_B_X4,SQ_B_Y4,colorAddr_B1,DRAW_B1); -- bullet for tank 1

score_tank1<=std_logic_vector(to_signed(score_tank1_int,4));
score_tank2<=std_logic_vector(to_signed(score_tank2_int,4));

-------------------------------------------PROCESS-------------------------------------------
pixelDraw : process(clk, rst_n) is
variable speed_1 : integer:=0;
variable speed_2 : integer:=0;
variable tank1_init : integer;
variable tank1final : integer;
variable tank2_init : integer;
variable tank2final : integer;
variable tank1_bullet_int: integer;
variable tank1_bullet_final: integer;
variable tank2_bullet_int: integer;
variable tank2_bullet_final: integer;

begin
```

```
if(rising_edge(clk)) then

                --------- Synchronous Reset ----------------------
                if(rst_n='0') then
                        SQ_X1<=10;
                        SQ_Y1<=250;
                        SQ_X2<=420;
                        SQ_Y2<=250;
                        score_tank1_int<=0;
                        score_tank2_int<=0;
                        speed_1:=0;
                        speed_2:=0;
                end if;

                if(shoot_done_tank2=1) then
                        if (DRAW_B='1') then
                                colorAddress<= color_black;
                        end if;
                end if;
                if(shoot_done_tank1=1) then
                        if (DRAW_B1='1') then
                                colorAddress<= color_black;
                        end if;
                end if;

                if (DRAW1='1') then
                        colorAddress<=color_green;
                end if;
                if (DRAW2='1') then
                        colorAddress<=color_yellow;
                end if;

                IF(DRAW1='0' AND DRAW2='0' AND DRAW_B='0' AND DRAW_B1='0')THEN
                        colorAddress<= color_blue;
                END IF;


------------------------------Bullet logic----------------------------------------

                if (scan_readyo_signal='1' and hist1_signal="11110000") then
                        if(scan_code_signal=x"1D") then -- press w
                                shoot_done_tank2<=1;
                                if(shoot_done_tank2=0) then
                                SQ_B_Y3<=SQ_Y2+15;
                                end if;
                                --lock_tank2<=1;
                        end if;
                end if;

                if (scan_readyo_signal='1' and hist1_signal="11110000") then
                        if(scan_code_signal=x"73") then -- press num 5
                                shoot_done_tank1<=1;
                                if(shoot_done_tank1=0) then
                                SQ_B_Y4<=SQ_Y1+15;
                                end if;
                                --lock_tank1<=1;
                        end if;
                end if;

                if(shoot_done_tank2=1) then

                                count_tank1<=count_tank1+1;
                                if (count_tank1 = 5000000) then
                                        SQ_B_X3<=SQ_B_X3-30;
                                        count_tank1<=0;
                                        if(SQ_B_X3<30)then
```

```
                                        shoot_done_tank2<=0;
                                        --lock_tank2<=0;
                                        SQ_B_X3<=445;
                                end if;
                        end if;
        end if;

        if(shoot_done_tank1=1) then                     -- when num 5 is pressed

                        count_tank2<=count_tank2+1;
                        if (count_tank2 = 5000000) then
                                SQ_B_X4<=SQ_B_X4+30;
                                count_tank2<=0;
                                if(SQ_B_X4>450)then
                                        shoot_done_tank1<=0;
                                        --lock_tank1<=0;
                                        SQ_B_X4<=35;
                                end if;
                        end if;
        end if;
        --------------------------------Speed Logic-------------------------------------
        if (scan_readyo_signal='1' and hist1_signal="11110000") then
                if(scan_code_signal="00010110") then-- press key 1
                        speed_2:=1;
                end if;
                if(scan_code_signal= x"1E") then-- press key 2
                        speed_2:=2;
                end if;
                if(scan_code_signal="00100110") then -- press key 3
                        speed_2:=3;
                end if;

                if(scan_code_signal=x"6C") then -- press num 7
                        speed_1:=1;
                end if;
                if(scan_code_signal= x"75") then -- press num 8
                        speed_1:=2;
                end if;
                if(scan_code_signal=x"7D") then -- press num 9
                        speed_1:=3;
                end if;
        end if;


        --------------------------------Tanks logic-----------------------------------------
-------------------------------------------------------------------------------------------
        if (scan_readyo_signal='1' and hist1_signal="11110000") then
                --if(lock_tank2=0) then
                        if(scan_code_signal=x"23") then --press D
                                        direc_ltor_tank2<='1';
                        end if;
                        if(scan_code_signal=x"1C") then --press A
                                        direc_ltor_tank2<='0';
                        end if;
                --end if;
        end if;
        if (scan_readyo_signal='1' and hist1_signal="11110000") then
                        if(scan_code_signal=x"7A") then --press num 3
                                        direc_ltor_tank1<='1';
                        end if;
                        if(scan_code_signal=x"69") then --press num 1
                                        direc_ltor_tank1<='0';
                        end if;
        end if;

        if(direc_ltor_tank2='1') then
                count_tank2_mov<=count_tank2_mov+1;
```

```
                if (count_tank2_mov = 5000000) then
                        SQ_Y2<=SQ_Y2+(10*speed_2);
                        count_tank2_mov<=0;
                end if;
        end if;
        if(direc_ltor_tank2='0') then
        count_tank2_mov<=count_tank2_mov+1;
                if (count_tank2_mov = 5000000) then
                        SQ_Y2<=SQ_Y2-(10*speed_2);
                        count_tank2_mov<=0;
                end if;
        end if;
        if(direc_ltor_tank1='1') then
                count_tank1_mov<=count_tank1_mov+1;
                if (count_tank1_mov = 5000000) then
                        SQ_Y1<=SQ_Y1+(10*speed_1);
                        count_tank1_mov<=0;
                end if;
        end if;
        if(direc_ltor_tank1='0') then
                count_tank1_mov<=count_tank1_mov+1;
                if (count_tank1_mov = 5000000) then
                        SQ_Y1<=SQ_Y1-(10*speed_1);
                        count_tank1_mov<=0;
                end if;
        end if;

        ---to bounce on edges--
        if(SQ_Y2>=600)then
                direc_ltor_tank2<='0';
        elsif(SQ_Y2<=10)then
                direc_ltor_tank2<='1';
        end if;
        if(SQ_Y1>=600)then
                direc_ltor_tank1<='0';
        elsif(SQ_Y1<=10)then
                direc_ltor_tank1<='1';
        end if;

--shooting logic--
                if(shoot_done_tank2=1 and SQ_B_X3<30) then
                        tank1_init:=SQ_Y1;
                        tank1final:=SQ_Y1+50;
                        tank2_bullet_int:=SQ_B_Y3;
                        tank2_bullet_final:=SQ_B_Y3+15;
                        if(tank2_bullet_final>tank1_init and tank2_bullet_int<tank1final) then
                                score_tank2_int<=score_tank2_int + 1;
                        end if;
                end if;

                if(shoot_done_tank1=1 and SQ_B_X4>450) then
                        tank2_init:=SQ_Y2;
                        tank2final:=SQ_Y2+50;
                        tank1_bullet_int:=SQ_B_Y4;
                        tank1_bullet_final:=SQ_B_Y4+15;
                        if(tank1_bullet_final>tank2_init and tank1_bullet_int<tank2final) then
                                score_tank1_int<=score_tank1_int + 1;
                        end if;
                end if;
                if score_tank2_int = 3 then
                        tank2_win <= '1';
                        count_win<=count_win+1;
                        if (count_win = 5000000) then
                                if tank2_win = '1' then
                                        SQ_X1 <= 700;
                                        SQ_Y1 <= 700;
                                        SQ_X2 <= 240;
```

```
                                        SQ_Y2 <= 320;
                                        SQ_B_X3<=700;
                                        --SQ_B_X4<=700;
                                        SQ_B_Y3<=700;
                                        --SQ_B_Y4<=700;
                                end if;
                                count_win<=0;
                        end if;
                else
                        tank2_win <= '0';
                end if;
                if score_tank1_int =3 then
                        tank1_win <= '1';
                        count_win<=count_win+1;
                        if (count_win = 5000000) then
                                if tank1_win = '1' then
                                        SQ_X2 <= 700;
                                        SQ_Y2 <= 700;
                                        SQ_X1 <= 240;
                                        SQ_Y1 <= 320;
                                        --SQ_B_X3<=700;
                                        SQ_B_X4<=700;
                                        --SQ_B_Y3<=700;
                                        SQ_B_Y4<=700;
                                end if;
                                count_win<=0;
                        end if;
                else
                        tank1_win <= '0';
                end if;


end if;

end process pixelDraw;

-------------------------------------------------------------------------------

end architecture behavioral;
```

**de2lcd.vhd**

```
LIBRARY IEEE;
USE  IEEE.STD_LOGIC_1164.all;
USE  IEEE.STD_LOGIC_ARITH.all;
USE  IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY de2lcd IS
        PORT(reset, clk_50Mhz                                   : IN        STD_LOGIC;
                a_win, b_win : in std_logic;
                 LCD_RS, LCD_E, LCD_ON, RESET_LED, SEC_LED      : OUT       STD_LOGIC;
                 LCD_RW                                         : BUFFER STD_LOGIC;
                 DATA_BUS                                       : INOUT   STD_LOGIC_VECTOR(7 DOWNTO 0));
END de2lcd;

ARCHITECTURE a OF de2lcd IS
        TYPE STATE_TYPE IS (HOLD, FUNC_SET, DISPLAY_ON, MODE_SET, WRITE_CHAR1,
        WRITE_CHAR2,WRITE_CHAR3,WRITE_CHAR4,WRITE_CHAR5,WRITE_CHAR6,WRITE_CHAR7,
        WRITE_CHAR8, WRITE_CHAR9, WRITE_CHAR10, RETURN_HOME, TOGGLE_E, RESET1, RESET2,
        RESET3, DISPLAY_OFF, DISPLAY_CLEAR);
        SIGNAL state, next_command: STATE_TYPE;
        SIGNAL DATA_BUS_VALUE: STD_LOGIC_VECTOR(7 DOWNTO 0);
        SIGNAL CLK_COUNT_400HZ: STD_LOGIC_VECTOR(19 DOWNTO 0);
        SIGNAL CLK_400HZ : STD_LOGIC;

BEGIN
```

```
	RESET_LED <= NOT RESET;
-- BIDIRECTIONAL TRI STATE LCD DATA BUS
	DATA_BUS <= DATA_BUS_VALUE WHEN LCD_RW = '0' ELSE "ZZZZZZZZ";

-- clock period adjustment for timing on lcd state diagram
	PROCESS
	BEGIN

	 WAIT UNTIL CLK_50MHZ'EVENT AND CLK_50MHZ = '1';
		IF RESET = '0' THEN
		 CLK_COUNT_400HZ <= X"00000";
		 CLK_400HZ <= '0';
		ELSE
				IF CLK_COUNT_400HZ < X"0F424" THEN
				 CLK_COUNT_400HZ <= CLK_COUNT_400HZ + 1;
				ELSE
			 CLK_COUNT_400HZ <= X"00000";
				 CLK_400HZ <= NOT CLK_400HZ;
				END IF;
		END IF;
	END PROCESS;
--sensitive to new clock
	PROCESS (CLK_400HZ, reset)
	variableinit: std_logic:='0';
	BEGIN
		LCD_ON <= '1';
		IF init = '0' THEN
			init := '1';
			state<= RESET1;
			DATA_BUS_VALUE <= X"38";
			next_command<= RESET2;
			LCD_E <= '1';
			LCD_RS <= '0';
			LCD_RW <= '0';

		ELSIF CLK_400HZ'EVENT AND CLK_400HZ = '1' THEN

			CASE state IS
-- Set Function to 8-bit transfer and 2 line display with 5x8 Font size
-- see Hitachi HD44780 family data sheet for LCD command and timing details
				WHEN RESET1 =>
					LCD_E <= '1';
					LCD_RS <= '0';
					LCD_RW <= '0';
					DATA_BUS_VALUE <= X"38";
					state<= TOGGLE_E;
					next_command<= RESET2;
				WHEN RESET2 =>
					LCD_E <= '1';
					LCD_RS <= '0';
					LCD_RW <= '0';
					DATA_BUS_VALUE <= X"38";
					state<= TOGGLE_E;
					next_command<= RESET3;
				WHEN RESET3 =>
					LCD_E <= '1';
					LCD_RS <= '0';
					LCD_RW <= '0';
					DATA_BUS_VALUE <= X"38";
					state<= TOGGLE_E;
					next_command<= FUNC_SET;
				WHEN FUNC_SET =>
					LCD_E <= '1';
					LCD_RS <= '0';
					LCD_RW <= '0';
					DATA_BUS_VALUE <= X"38";
					state<= TOGGLE_E;
```

```
                                        next_command<= DISPLAY_OFF;
-- Turn off Display and Turn off cursor
                                WHEN DISPLAY_OFF =>
                                        LCD_E <= '1';
                                        LCD_RS <= '0';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"08";
                                        state<= TOGGLE_E;
                                        next_command<= DISPLAY_CLEAR;
-- Turn on Display and Turn off cursor
                                WHEN DISPLAY_CLEAR =>
                                        LCD_E <= '1';
                                        LCD_RS <= '0';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"01";
                                        state<= TOGGLE_E;
                                        next_command<= DISPLAY_ON;
-- Turn on Display and Turn off cursor
                                WHEN DISPLAY_ON =>
                                        LCD_E <= '1';
                                        LCD_RS <= '0';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"0C";
                                        state<= TOGGLE_E;
                                        next_command<= MODE_SET;
-- Set write mode to auto increment address and move cursor to the right
                                WHEN MODE_SET =>
                                        LCD_E <= '1';
                                        LCD_RS <= '0';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"06";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR1;
-- Write ASCII hex character in first LCD character location
                                WHEN WRITE_CHAR1 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"57";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR2;
-- Write ASCII hex character in second LCD character location
                                WHEN WRITE_CHAR2 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"49";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR3;
-- Write ASCII hex character in third LCD character location
                                WHEN WRITE_CHAR3 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"4E";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR4;
-- Write ASCII hex character in fourth LCD character location
                                WHEN WRITE_CHAR4 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"4E";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR5;
-- Write ASCII hex character in fifth LCD character location
                                WHEN WRITE_CHAR5 =>
```

```
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"45";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR6;
-- Write ASCII hex character in sixth LCD character location
                                WHEN WRITE_CHAR6 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"52";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR7;
-- Write ASCII hex character in seventh LCD character location
                                WHEN WRITE_CHAR7 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"20";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR8;
-- Write ASCII hex character in eighth LCD character location
                                WHEN WRITE_CHAR8 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"2D";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR9;
                                WHEN WRITE_CHAR9 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"20";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR10;
                                WHEN WRITE_CHAR10 =>
                                        LCD_E <= '1';
                                        LCD_RS <= '1';
                                        LCD_RW <= '0';
                                        ifa_win = '1' then
                                                DATA_BUS_VALUE <= X"41";
                                        elsifb_win = '1' then
                                                DATA_BUS_VALUE <= X"42";
                                        else
                                                DATA_BUS_VALUE <= X"20";
                                        end if;
                                        state<= TOGGLE_E;
                                        next_command<= RETURN_HOME;

-- Return write address to first character postion
                                WHEN RETURN_HOME =>
                                        LCD_E <= '1';
                                        LCD_RS <= '0';
                                        LCD_RW <= '0';
                                        DATA_BUS_VALUE <= X"80";
                                        state<= TOGGLE_E;
                                        next_command<= WRITE_CHAR1;
-- The next two states occur at the end of each command to the LCD
-- Toggle E line - falling edge loads inst/data to LCD controller
                                WHEN TOGGLE_E =>
                                        LCD_E <= '0';
                                        state<= HOLD;
-- Hold LCD inst/data valid after falling edge of E line
                                WHEN HOLD =>
                                        state<= next_command;
```

```
                END CASE;
            END IF;
        END PROCESS;
END a;
```