LAB PROJECT

Title: LCD controller over FPGA

Roll No. : 21BEC111 (Senghani Dhairya)
21BEC132 (Vartak Aditya)

Subject: Digital System Design (2ECDE68)

Submitted To: Prof. Hardik Joshi

Introduction

16x2 LCDs are commonly used for displaying text in various electronic projects. Interfacing an FPGA with an LCD screen allows for more flexible control and integration into digital systems. This project aims to demonstrate the interface between an FPGA and a 16x2 LCD using Verilog for programming.

The interface of a 16x2 LCD with a microcontroller or other control unit involves connecting the LCD to the controller and defining the communication protocol. Here's a brief overview of the interface:

1. Physical Connections: To interface with a 16x2 LCD, you need to establish the physical connections between the LCD module and the controller. This typically includes connecting data lines (D0-D7) and control lines (RS, RW, E), as well as power and ground connections.

2. Data Lines (D0-D7): The data lines are used to transmit data between the controller and the LCD. In 4-bit mode, only D4-D7 are used, while in 8-bit mode, all data lines (D0-D7) are utilized.

3. Control Lines:

-RS (Register Select): RS is used to select whether data sent to the LCD is an instruction (RS = 0) or character data (RS = 1).

- RW (Read/Write): RW is used to indicate whether data is being written to the LCD (RW = 0) or read from the LCD (RW = 1). In most applications, RW is permanently set to write mode (0).

- E (Enable): The Enable (E) line is used to signal the LCD that valid data is present on the data lines. It typically works by sending a pulse to latch the data.

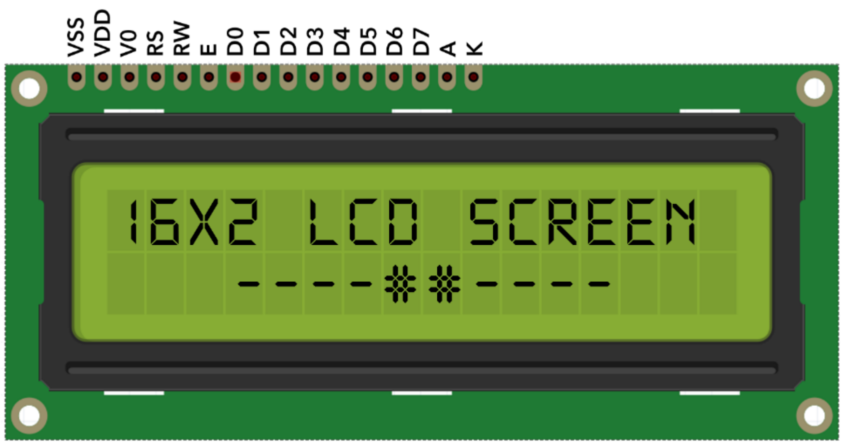
4. Initialization: Prior to using the LCD, it must be initialized by sending a sequence of commands. These commands configure display settings, cursor behavior, and other parameters.

5. Sending Data: To display characters or custom symbols on the LCD, the controller sends data to the LCD's data lines (D0-D7) along with control signals. RS is set to 1 to indicate character data. If in 4-bit mode, the data is sent in two 4-bit nibbles.

6. Backlight Control: If the LCD has a backlight, it can be controlled by an additional pin. This allows for turning the backlight on or off based on your application's requirements.

7. Custom Characters: Many 16x2 LCDs allow you to create custom characters or symbols. Custom character patterns are defined and stored in the LCD's memory, and these custom characters can be displayed in place of standard characters.

16x2 LCD pin diagram:



No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

Common commands for a 16x2 LCD display (HD44780):

1. Clear Display (0x01): Clears the screen.

2. Return Home (0x02): Moves cursor to (0,0).

3. Entry Mode Set (0x04): Configures cursor movement.

4. Display On/Off (0x08): Controls display and cursor.

5. Cursor/Display Shift (0x10): Shifts cursor or display.

6. Function Set (0x20): Sets data length, display lines, font.

7. Set CGRAM Address (0x40 + Addr): Custom character address.

8. Set DDRAM Address (0x80 + Addr): Cursor position.

9. Read Busy Flag (0x80): Read BF and address.

10. Write Data (Character Display): Send characters to display.

11. Shift Display Left (0x18): Shifts the entire display to the left.

12. Shift Display Right (0x1C): Shifts the entire display to the right.

13. Set Display to 1 Line (0x30): Sets the display to one line (useful for 1-line displays).

14. Set Display to 2 Lines (0x38): Sets the display to two lines (typical for 16x2 displays)

15. Create Custom Character (0x40 to 0x47): Defines custom characters for display.

16. Set Contrast (0x39): Used with displays that support contrast control

17. Cursor Blink (0x0D): Enables cursor blinking.

18. Cursor No Blink (0x0C): Disables cursor blinking

19. Display Shift Left (0x07): Shifts the display to the left (cursor follows).

20. Display Shift Right (0x05): Shifts the display to the right (cursor follows).

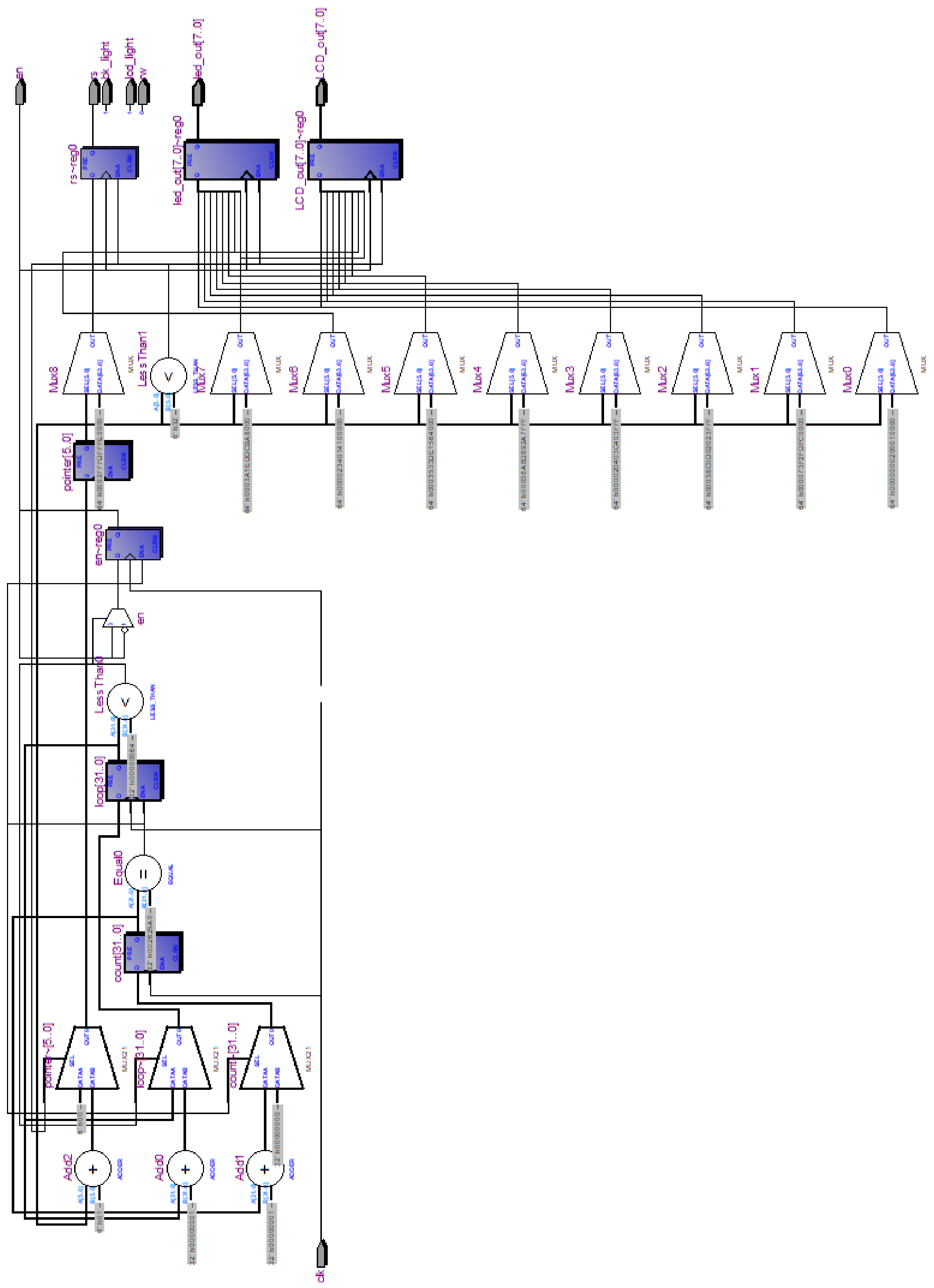
Verilog Code(Dataflow and Behavioural Modeling):

```
module LCD(
   input clk,
   output reg rs,en,
          output bk_light,lcd_light,rw,
   output reg [7:0] LCD_out,led_out
    );
assign rw=1'b0;
assign bk_light=1'b1;
assign lcd_light=1'b1;
wire [8:0]memory[0:49];
reg [5:0]pointer;
reg [31:0]count,loop;
initial pointer=6'b000000;
initial en=1'b0;
initial loop=0;

assign  memory[0]={1'b0,8'h38};
assign  memory[1]={1'b0,8'h38};
assign  memory[2]={1'b0,8'h38};
assign  memory[3]={1'b0,8'h38};
assign  memory[4]={1'b0,8'h38};
assign  memory[5]={1'b0,8'h38};
assign  memory[6]={1'b0,8'h38};
assign  memory[7]={1'b0,8'h38};
assign  memory[8]={1'b0,8'h38};
assign  memory[9]={1'b0,8'h38};
assign  memory[10]={1'b0,8'h38};
assign  memory[11]={1'b0,8'h38};
assign  memory[12]={1'b0,8'h38};
assign  memory[13]={1'b0,8'h38};
assign  memory[14]={1'b0,8'h0C};
assign  memory[15]={1'b0,8'h01};
assign  memory[16]={1'b0,8'h80};

assign  memory[17]={1'b1,"-"};
assign  memory[18]={1'b1,"D"};
assign  memory[19]={1'b1,"I"};
assign  memory[20]={1'b1,"G"};
assign  memory[21]={1'b1,"I"};
assign  memory[22]={1'b1,"T"};
assign  memory[23]={1'b1,"A"};
assign  memory[24]={1'b1,"L"};
assign  memory[25]={1'b1," "};
assign  memory[26]={1'b1,"S"};
assign  memory[27]={1'b1,"Y"};
assign  memory[28]={1'b1,"S"};
assign  memory[29]={1'b1,"T"};
assign  memory[30]={1'b1,"E"};
assign  memory[31]={1'b1,"M"};
assign  memory[32]={1'b1,"-"};

assign  memory[33]={1'b0,8'hC0};

assign  memory[34]={1'b1,"-"};
assign  memory[35]={1'b1,"-"};
assign  memory[36]={1'b1,"D"};
assign  memory[37]={1'b1,"E"};
assign  memory[38]={1'b1,"S"};
assign  memory[39]={1'b1,"I"};
assign  memory[40]={1'b1,"G"};
assign   memory[41]={1'b1,"N"};
assign   memory[42]={1'b1," "};
assign   memory[43]={1'b1,"("};
assign   memory[44]={1'b1,"D"};
assign   memory[45]={1'b1,"S"};
assign   memory[46]={1'b1,"D"};
assign   memory[47]={1'b1,")"};
assign   memory[48]={1'b1,"-"};
assign   memory[49]={1'b1,"-"};

always @(posedge clk)
begin
if(count == 2500000) begin
   count <= 0;
          if(loop<100)
          begin
   en <= ~en;
          loop<=loop+1;
          end
end else begin
   count <= count + 1;
   end
end


always @(posedge en)
begin
        if(pointer < 6'b110010)
   begin
          LCD_out=memory[pointer][7:0];
          led_out=memory[pointer][7:0];
          rs=memory[pointer][8];
          pointer = pointer + 1'b1;
          end
          else
          pointer=6'b000000;
end

endmodule
```

Code execution flow:

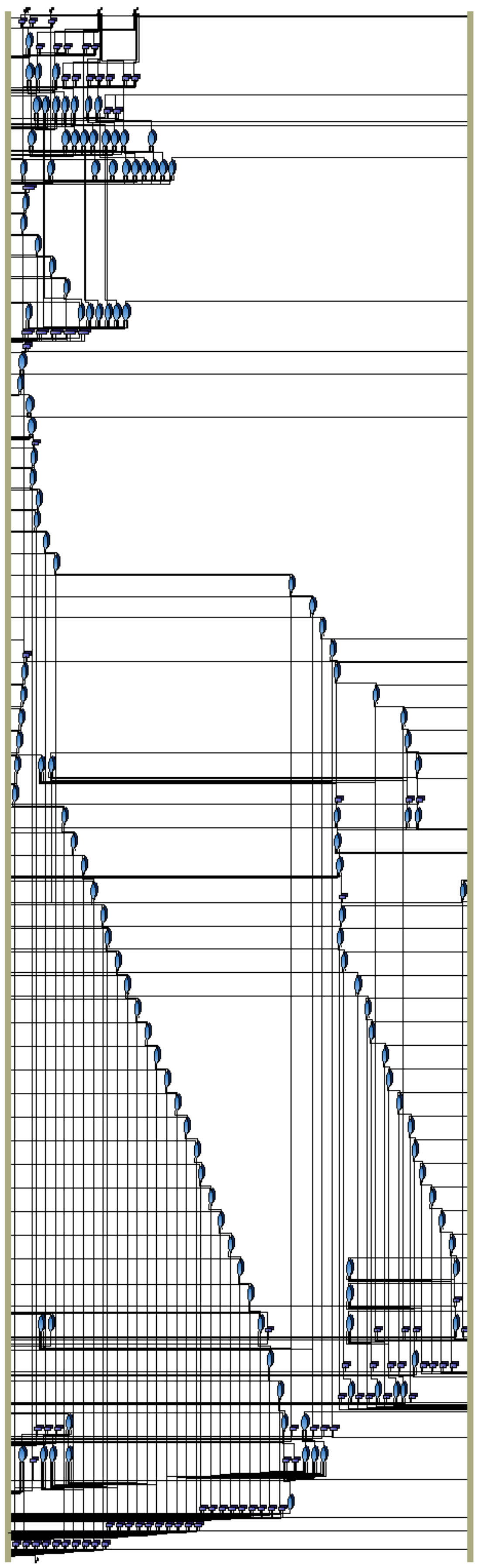
This code display ***"DIGITAL SYSTEM DESIGN (DSD)".*** In this code first we have created 49 memory elements containing commands and data(ASCII values). We used 16x2 LCD in 8 bit mode. First we have decreases the input clock frequency from 50MHz to 1 Hz using counter and assigned to variable '**en**'. At every positive edge of **en**, data on data lines get change and on negative edge of **en** data or command on data line gets implemented on LCD. One counter is used to count the transition of **en** up to 50, because there are 50 memory element. **pointer** points the memory and **bk_light** and **lcd_light** are used to turn on LCD. Here logic '0' of **rw** indicates the write operation of LCD. 'low' logic of **rs** (register select) indicates command register and 'high' logic indicates data register.

RTL view

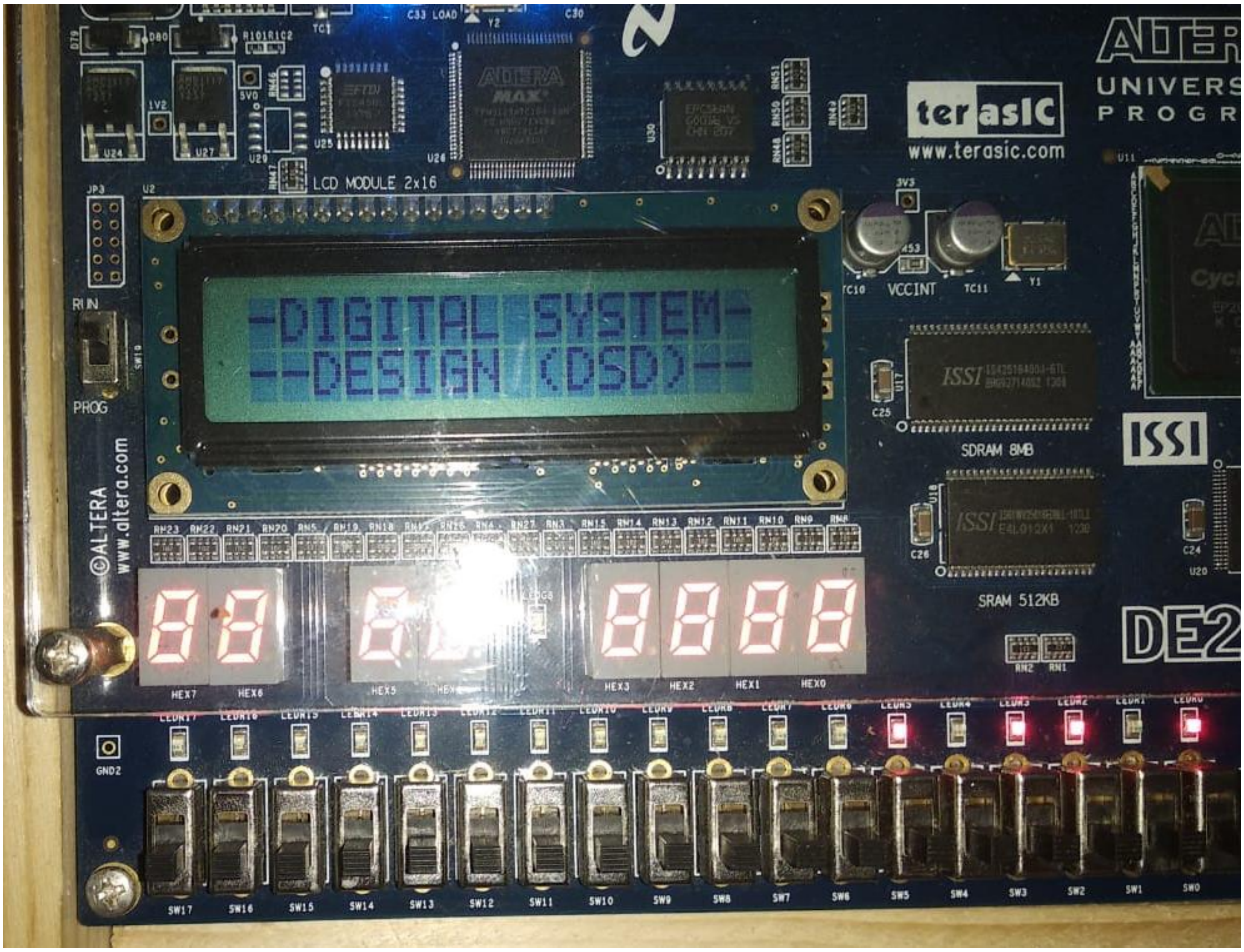


en
rs
bk_light
lcd_light
rw
led_out[7..0]
LCD_out[7..0]
rs~reg0
led_out[7..0]~reg0
LCD_out[7..0]~reg0
Mux8
LessThan1
Mux7
Mux6
Mux5
Mux4
Mux3
Mux2
Mux1
Mux0
pointer[5..0]
en~reg0
en
LessThan0
loop[31..0]
Equal0
count[31..0]
pointer~[5..0]
loop~[31..0]
count~[31..0]
Add2
Add0
Add1
clk

TTL view :



FPGA Implementation



Conclusion

LCD interfacing on FPGA using verilog is quit complex task. We have to take care of hold times of command and data registers. We can use LCD in 4 bit or 8 bit mode as per available hardware resources(we used in 8 bit mode). We learnt to handle several delays on FPGA using Verilog and implemented it successfully on ALTERA DE2 FPGA kit. This report has outlined the hardware setup, Verilog code structure, and the process of testing and implementation.