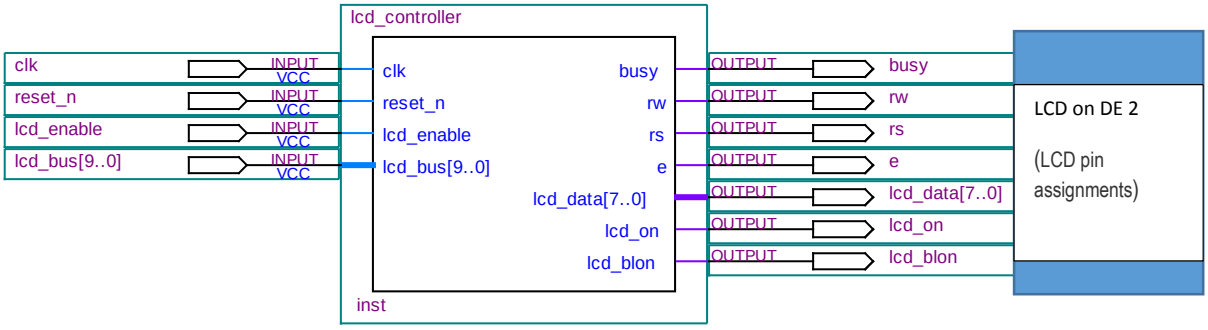
LCD controller and User Logic in VHDL and Programming a FPGAs

The LCD module (HD44780) at DE2 board has built-in fonts and can be used to display text by sending appropriate commands to the display controller. The controller manages the initialization and data flow to HD44780 compatible 8-bit interface character LCD modules. Figure 1 depicts the LCD controller implemented to interface between an LCD module and a user's custom logic.



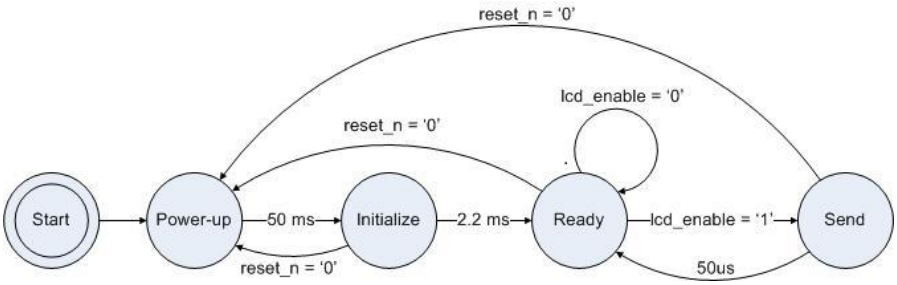
LCD controller's interface and LCD Controller I/O Description

I/O Name	Width	Mode	Description	Interface
clk	1	input	Clock for LCD controller. Default set for 50MHz. If a different frequency is desired, change the constant freq in the architecture declarations to reflect the new frequency in MHz.	system clock (50MHz)
reset_n	1	input	Active low synchronous reset pin. This pin must be set high to implement the LCD controller. Setting the pin low for one or more clock cycles restarts the LCD controller state machine.	user logic
lcd_enable	1	input	Data latch for LCD controller. H: initiates a transaction using the data currently on the lcd_bus, L: no transaction is initiated and any data on lcd_bus is ignored	user logic
lcd_bus	10	input	Data/instructions to be sent to the LCD module. The MSB is the rs signal, followed by the rw signal. The other 8 bits are the data bits. The LSB on the bus corresponds to the least significant data bit.	user logic
busy	1	output	Feedback on the state of the LCD controller. H: the controller is busy initializing or conducting a transaction with the LCD module, any instructions/data sent will be ignored, L: the controller is idle and ready to accept commands for a transaction	user logic
rs	1	output	LCD module Register Select Signal; H: sending data, L: sending instructions	LCD pin 4
rw	1	output	LCD module Read/Write Select Signal; H: Read, L: Write	LCD pin 5
e	1	output	LCD module enable signal	LCD pin 6
lcd_data	8	bidir	Data bus to the LCD module / busy signal from the LCD	LCD pins 7-14

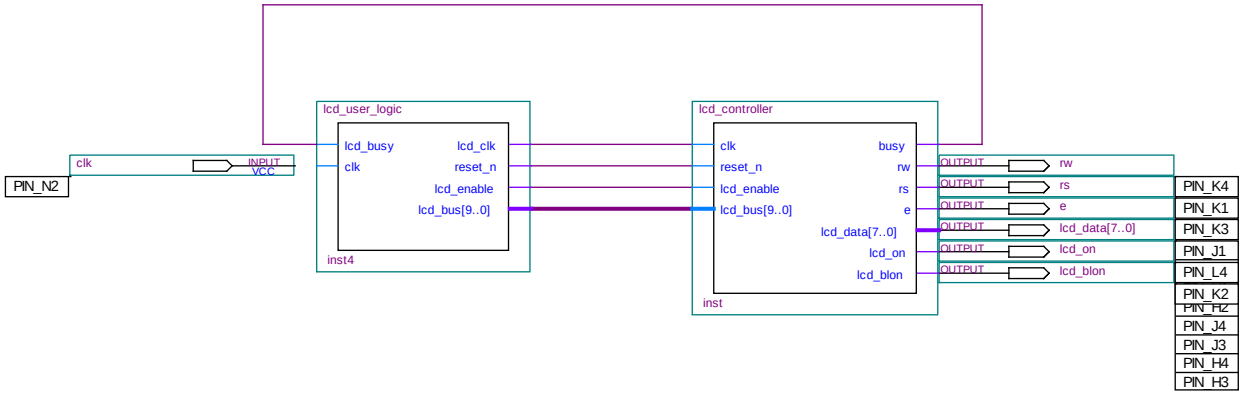
Pin assignments for the LCD

Signal Name	FPGA Pin No.	Description
LCD_DATA[0]	PIN_J1	LCD Data[0]
LCD_DATA[1]	PIN_J2	LCD Data[1]
LCD_DATA[2]	PIN_H1	LCD Data[2]
LCD_DATA[3]	PIN_H2	LCD Data[3]
LCD_DATA[4]	PIN_J4	LCD Data[4]
LCD_DATA[5]	PIN_J3	LCD Data[5]
LCD_DATA[6]	PIN_H4	LCD Data[6]
LCD_DATA[7]	PIN_H3	LCD Data[7]
LCD_RW	PIN_K4	LCD Read/Write Select, 0 = Write, 1 = Read
LCD_EN	PIN_K3	LCD Enable
LCD_RS	PIN_K1	LCD Command/Data Select, 0 = Command, 1 = Data
LCD_ON	PIN_L4	LCD Power ON/OFF
LCD_BLON	PIN_K2	LCD Back Light ON/OFF

LCD Controller State Machine



Developing lcd_user_logic.vhd to send "***123456789AB***" to a HD44780 compatible 8-bit interface character LCD by using the module of lcd_controller.vhd component.



The vhdl source code for above two componets:

1) lcd_user_logic.vhd
2) lcd_controller.vhd

HD44780U

Table 4 Correspondence between Character Codes and Character Patterns (ROM Code: A02)

Lower 4 Bits \ Upper 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)															
xxxx0001	(2)															
xxxx0010	(3)															
xxxx0011	(4)															
xxxx0100	(5)															
xxxx0101	(6)															
xxxx0110	(7)															
xxxx0111	(8)															
xxxx1000	(1)															
xxxx1001	(2)															
xxxx1010	(3)															
xxxx1011	(4)															
xxxx1100	(5)															
xxxx1101	(6)															
xxxx1110	(7)															
xxxx1111	(8)															

Reference:

https://eewiki.net/pages/viewpage.action?pageId=4096079

DE2 User Manual

HD44780(LCD-II) Dot Matrix Liquid Crystal Display Controller/Driver

```
1  ------------------------------------------------------------------
   -----------
2  --
3  --   FileName:         lcd_controller.vhd
4  --   CLOCK FREQUENCY: you may change system clock frequency,
5  --   LCD INITIALIZATION SETTINGS: to change, comment/uncomment lines:
6  --
7  --   Function Set
8  --      2-line mode, display on             Line 85    lcd_data <=
   "00111100";
9  --      1-line mode, display on             Line 86    lcd_data <=
   "00110100";
10 --      1-line mode, display off            Line 87    lcd_data <=
   "00110000";
11 --      2-line mode, display off            Line 88    lcd_data <=
   "00111000";
12 --   Display ON/OFF
13 --      display on, cursor off, blink off   Line 96   lcd_data <=
   "00001100";
14 --      display on, cursor off, blink on    Line 97   lcd_data <=
   "00001101";
15 --      display on, cursor on, blink off    Line 98   lcd_data <=
   "00001110";
16 --      display on, cursor on, blink on     Line 99   lcd_data <=
   "00001111";
17 --      display off, cursor off, blink off  Line 100   lcd_data <=
   "00001000";
18 --      display off, cursor off, blink on   Line 101   lcd_data <=
   "00001001";
19 --      display off, cursor on, blink off   Line 102   lcd_data <=
   "00001010";
20 --      display off, cursor on, blink on    Line 103   lcd_data <=
   "00001011";
21 --   Entry Mode Set
22 --      increment mode, entire shift off    Line 119   lcd_data <=
   "00000110";
23 --      increment mode, entire shift on     Line 120   lcd_data <=
   "00000111";
24 --      decrement mode, entire shift off    Line 121   lcd_data <=
   "00000100";
25 --      decrement mode, entire shift on     Line 122   lcd_data <=
   "00000101";
26 --
27 ------------------------------------------------------------------
   -----------
28
29 LIBRARY ieee;
30 USE ieee.std_logic_1164.ALL;
31 USE IEEE.STD_LOGIC_ARITH.ALL;
32 USE IEEE.STD_LOGIC_UNSIGNED.ALL;
33
34 ENTITY lcd_controller IS
35   PORT(
```

```
36       clk         : IN    STD_LOGIC;  --system clock
37       reset_n     : IN    STD_LOGIC;  --active low reinitializes lcd
38       lcd_enable  : IN    STD_LOGIC;  --latches data into lcd controller
39       lcd_bus     : IN    STD_LOGIC_VECTOR(9 DOWNTO 0);  --data and
     control signals
40       busy        : OUT   STD_LOGIC := '1';  --lcd controller
     busy/idle feedback
41       rw, rs, e   : OUT   STD_LOGIC;  --read/write, setup/data, and
     enable for lcd
42       lcd_data    : OUT   STD_LOGIC_VECTOR(7 DOWNTO 0); --data signals
     for lcd
43       lcd_on : OUT std_logic; --LCD Power ON/OFF
44       lcd_blon : OUT std_logic); --LCD Back Light ON/OFF
45   END lcd_controller;
46
47   ARCHITECTURE controller OF lcd_controller IS
48     --state machine
49     TYPE CONTROL IS(power_up, initialize, ready, send);
50     SIGNAL    state      : CONTROL;
51      CONSTANT  freq       : INTEGER := 50; --system clock frequency
     in MHz
52
53   BEGIN
54       lcd_on <= '1'; --LCD Power ON
55       lcd_blon<='1'; --LCD Back Light ON
56
57     PROCESS(clk)
58       VARIABLE clk_count : INTEGER := 0; --event counter for timing
59     BEGIN
60     IF(clk'EVENT and clk = '1') THEN
61
62         CASE state IS
63
64           --wait 50 ms to ensure Vdd has risen and required LCD wait
     is met
65           WHEN power_up =>
66             busy <= '1';
67             IF(clk_count < (50000 * freq)) THEN    --wait 50 ms
68               clk_count := clk_count + 1;
69               state <= power_up;
70             ELSE                                   --power-up complete
71               clk_count := 0;
72               rs <= '0';
73               rw <= '0';
74               lcd_data <= "00110000";  -- Function Set: 1-line mode,
     display off  lcd_data <= "00110000";
75               state <= initialize;
76             END IF;
77
78           --cycle through initialization sequence
79           WHEN initialize =>
80             busy <= '1';
81             clk_count := clk_count + 1;
```

```
82              IF(clk_count < (10 * freq)) THEN       --function set
83               -- lcd_data <= "00111100";      --2-line mode, display on
84                lcd_data <= "00110100";    --1-line mode, display on
85                --lcd_data <= "00110000";    --1-line mdoe, display off
86                --lcd_data <= "00111000";    --2-line mode, display off
87                e <= '1';
88                state <= initialize;
89              ELSIF(clk_count < (60 * freq)) THEN    --wait 50 us
90                lcd_data <= "00000000";
91                e <= '0';
92                state <= initialize;
93              ELSIF(clk_count < (70 * freq)) THEN    --display on/off
    control
94                --lcd_data <= "00001100";      --display on, cursor
    off, blink off
95                lcd_data <= "00001101";    --display on, cursor off,
    blink on
96                --lcd_data <= "00001110";    --display on, cursor on,
    blink off
97                --lcd_data <= "00001111";    --display on, cursor on,
    blink on
98                --lcd_data <= "00001000";    --display off, cursor off,
    blink off
99                --lcd_data <= "00001001";    --display off, cursor off,
    blink on
100               --lcd_data <= "00001010";    --display off, cursor on,
    blink off
101               --lcd_data <= "00001011";    --display off, cursor on,
    blink on
102               e <= '1';
103               state <= initialize;
104             ELSIF(clk_count < (120 * freq)) THEN   --wait 50 us
105               lcd_data <= "00000000";
106               e <= '0';
107               state <= initialize;
108             ELSIF(clk_count < (130 * freq)) THEN   --display clear
109               lcd_data <= "00000001";
110               e <= '1';
111               state <= initialize;
112             ELSIF(clk_count < (2130 * freq)) THEN  --wait 2 ms
113               lcd_data <= "00000000";
114               e <= '0';
115               state <= initialize;
116             ELSIF(clk_count < (2140 * freq)) THEN  --entry mode set
117               lcd_data <= "00000110";      --increment mode, entire
    shift off
118               --lcd_data <= "00000111";    --increment mode, entire
    shift on
119               --lcd_data <= "00000100";    --decrement mode, entire
    shift off
120               --lcd_data <= "00000101";    --decrement mode, entire
    shift on
121               e <= '1';
```

```
122               state <= initialize;
123             ELSIF(clk_count < (2200 * freq)) THEN  --wait 60 us
124               lcd_data <= "00000000";
125               e <= '0';
126               state <= initialize;
127             ELSE                                   --initialization
    complete
128               clk_count := 0;
129               busy <= '0';
130               state <= ready;
131             END IF;
132
133           --wait for the enable signal and then latch in the
    instruction
134           WHEN ready =>
135             IF(lcd_enable = '1') THEN
136               busy <= '1';
137               rs <= lcd_bus(9);
138               --rs<= lcd_rs;
139               rw <= lcd_bus(8);
140               --rw <= lcd_rw;
141               lcd_data <= lcd_bus(7 DOWNTO 0);
142               --lcd_data <= lcd_bus;
143               clk_count := 0;
144               state <= send;
145             ELSE
146               busy <= '0';
147               rs <= '0';
148               rw <= '0';
149               lcd_data <= "00000000";
150               clk_count := 0;
151               state <= ready;
152             END IF;
153
154           --send instruction to lcd
155           WHEN send =>
156           busy <= '1';
157           IF(clk_count < (50 * freq)) THEN  --do not exit for 50us
158              busy <= '1';
159              IF(clk_count < freq) THEN      --negative enable
160               e <= '0';
161              ELSIF(clk_count < (14 * freq)) THEN  --positive enable
    half-cycle
162               e <= '1';
163              ELSIF(clk_count < (27 * freq)) THEN  --negative enable
    half-cycle
164               e <= '0';
165              END IF;
166              clk_count := clk_count + 1;
167              state <= send;
168           ELSE
169             clk_count := 0;
170             state <= ready;
```

```
171           END IF;
172
173         END CASE;
174
175         --reset
176         IF(reset_n = '0') THEN
177             state <= power_up;
178         END IF;
179
180       END IF;
181     END PROCESS;
182   END controller;
183
```

```
1  ------------------------------------------------------------------
   -----------
2  --
3  --   FileName:          lcd_user_logic.vhd
4  --   Prints "123456789AB" on a HD44780 compatible 8-bit interface
   character LCD
5  --   module using the lcd_controller.vhd component.
6  --
7  ------------------------------------------------------------------
   -----------
8
9  LIBRARY ieee;
10 USE ieee.std_logic_1164.ALL;
11 USE IEEE.STD_LOGIC_ARITH.ALL;
12 USE IEEE.STD_LOGIC_UNSIGNED.ALL;
13
14 ENTITY lcd_user_logic IS
15   PORT(
16       lcd_busy  : IN  STD_LOGIC;  --lcd controller busy/idle feedback
17       clk       : IN  STD_LOGIC;  --system clock
18       lcd_clk    : OUT  STD_LOGIC;
19       reset_n    : OUT  STD_LOGIC;
20       lcd_enable :  buffer STD_LOGIC;  --lcd enable received from
   lcd controller
21       lcd_bus    : OUT  STD_LOGIC_VECTOR(9 DOWNTO 0)); --data and
   control signals
22                                                        --The MSB
   is the rs signal, followed by the rw signal.
23                                                        -- The
   other 8 bits are the data bits.
24 END lcd_user_logic;
25
26 ARCHITECTURE behavior OF lcd_user_logic IS
27
28 BEGIN
29
30   PROCESS(clk)
31     VARIABLE char  :  INTEGER RANGE 0 TO 12 := 0;
32   BEGIN
33     IF(clk'EVENT AND clk = '1') THEN
34       IF(lcd_busy = '0' AND lcd_enable = '0') THEN
35         lcd_enable <= '1';
36         IF(char < 12) THEN
37           char := char + 1;
38         END IF;
39         CASE char IS
40           WHEN 1 => lcd_bus <= "1000110001";
41           WHEN 2 => lcd_bus <= "1000110010";
42           WHEN 3 => lcd_bus <= "1000110011";
43           WHEN 4 => lcd_bus <= "1000110100";
44           WHEN 5 => lcd_bus <= "1000110101";
45           WHEN 6 => lcd_bus <= "1000110110";
46           WHEN 7 => lcd_bus <= "1000110111";
```

```
47              WHEN 8 => lcd_bus <= "1000111000";
48              WHEN 9 => lcd_bus <= "1000111001";
49              WHEN 10 => lcd_bus<= "1001000001";
50              WHEN 11 => lcd_bus<= "1001000010";
51              WHEN OTHERS => lcd_enable <= '0';
52            END CASE;
53          ELSE
54            lcd_enable <= '0';
55          END IF;
56        END IF;
57      END PROCESS;
58
59      reset_n <= '1';
60      lcd_clk <= clk;
61    END behavior;
62
```