An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

**Computer Engineering Department**

**Course Name: Digital design Lab 2** **Number: 10636391**

**Lab Report Grading Sheet**

| | |
|---|---|
| Instructor: Dr Ashraf Armoush | Experiment #: 5 |
| Academic Year: 2020/2021 | Experiment Name: LCD |
| Semester: Summer | |

| **Students** | |
|---|---|
| 1-Ashraf Habromman | 2-Ra'ed Khwayerh |
| Performed on: 6/7/2021 | Submitted on:13/7/2021 |

**Report's Outcomes**

| | | | | |
|---|---|---|---|---|
| ILO __ =( ) % | ILO __ =( ) % | ILO __ =( ) % | ILO __ =( ) % | ILO __ =( ) % |

| Evaluation Criterion | Grade | Points |
|---|---|---|
| **Abstract** answers of the questions: "What did you do? How did you do it? What ?"did you find | 0.5 | |
| **Introduction and Theory** Sufficient, clear and complete statement of objectives. In addition to Presents sufficiently the theoretical basis. | 1.5 | |
| **Apparatus/ Procedure** Apparatus sufficiently described to enable another experimenter to identify the equipment needed to conduct the experiment.Procedure .sufficiently described | 2 | |
| **(Experimental Results and Discussion (In-Lab Worksheet** Crisp explanation of experimental results. Comparison of theoretical predictions to experimental results, including discussion of accuracy .and error analysis in some cases | 4 | |
| **Conclusions and Recommendations** Conclusions summarize the major findings from the experimental results with adequate specificity. Recommendations appropriate in .light of conclusions. Correct grammar | 1 | |
| **Appearance** Title page is complete, page numbers applied, content is well organized, correct spelling, fonts are consistent, good visual appeal. | 1 | |
| **Total** | 10 | |

# Introduction:

In this lab, you will design and implement a driver that will be used to handle the parallel communication between the external LCD module and the ZedBoard.
The LCD consists of two lines. Each line can display up to 16 characters. To display a character, first you need to send the location address where you want the character to be displayed.Then the character code of the character to be displayed.
The LCD device has three internal regions of memory. The Data Display RAM (DD RAM), which references the data to be displayed on the screen, the Character Generator RAM (CG RAM), which stores user-defined patterns and the Character Generator ROM (CG ROM), which includes a number of predefined patterns that correspond to ASCII symbols. We will only use the DD-RAM and the CG-ROM. To reference a value in the CG-ROM.
We have a table that has the codes for characters.

# Abstract:

In this experiment, we'll learn how to design and implement a driver that will be used to handle the parallel communication between the external LCD module and the ZedBoard.
We'll do two parts. The first is displaying our name, the second is displaying a counter.

# Apparatus:

1-Vivado Design Suite HL WebPACK™ Edition.
2-ZedBoard.
3-LCD

# Procedure:

## Part A: Writing "ASHRAF_RA" on the LCD:

RA = the first two characters of RAED name.

### LCD Driver implementation:

---

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lcd_driver is
```

```
    Port ( clk : in STD_LOGIC;
           data_bus : out STD_LOGIC_VECTOR(7 downto 0);
           enable : out STD_LOGIC;
           rg_s : out STD_LOGIC;
           r_w : out STD_LOGIC);
end lcd_driver;

architecture Behavioral of lcd_driver is

signal counter : integer:=0;

begin

process(clk)
    begin
        if(clk'event and clk = '1')then
            counter<= counter + 1;
            if(counter < 2000000)then
                enable <= '0';
                rg_s <= '0';
                r_w <= '0';
            elsif(counter >= 2000000 and counter <= 2004000)then
                data_bus <= "00111000";
                if(counter >= 2000000 and counter <= 2000100)then
                    enable <= '1';
                else enable <= '0';
                end if;
            elsif(counter > 2004000 and counter <= 2008000)then
                data_bus <= "00000110";
                if(counter >= 2004000 and counter <= 2004100)then
                    enable <= '1';
                else enable <= '0';
                end if;
            elsif(counter > 2008000 and counter <= 2012000 ) then
                data_bus <= "00001100";
                if(counter >= 2008000 and counter <= 2008100)then
                    enable <= '1';
                else enable <= '0';
                end if;
            elsif(counter > 2012000 and counter <= 2176000)then
                data_bus <= "00000001"; -- finish config
                if(counter >= 2012000 and counter <= 2012100)then
                    enable <= '1';
                else enable <= '0';
                end if ;
```

```
elsif(counter > 2176000 and counter <= 2180000)then
    data_bus <= "10000000"; -- address
    if(counter >= 2176000 and counter <= 2176100)then
       enable <= '1';
    else enable <= '0';
    end if ;

elsif(counter > 2180000 and counter <= 2184000)then
    rg_s <= '1';
    data_bus <= "01000001"; -- A char
    if(counter >= 2180000 and counter <= 2180100)then
       enable <= '1';
    else enable <= '0';
    end if ;

      elsif(counter > 2184000 and counter <= 2188000)then
    rg_s <= '1';
    data_bus <= "01010011"; -- S char
    if(counter >= 2184000 and counter <= 2184100)then
       enable <= '1';
    else enable <= '0';
    end if ;

elsif(counter > 2188000 and counter <= 2192000)then
    rg_s <= '1';
    data_bus <= "01001000"; -- H char
    if(counter >= 2188000 and counter <= 2188100)then
       enable <= '1';
    else enable <= '0';
    end if ;

elsif(counter > 2192000 and counter <= 2196000)then
    rg_s <= '1';
    data_bus <= "01010010"; -- R char
    if(counter >= 2192000 and counter <= 2192100)then
       enable <= '1';
    else enable <= '0';
    end if ;

elsif(counter > 2196000 and counter <= 2200000)then
    rg_s <= '1';
    data_bus <= "01000001"; -- A char
    if(counter >= 2196000 and counter <= 2196100)then
       enable <= '1';
    else enable <= '0';
```

```
            end if ;

              elsif(counter > 2200000 and counter <= 2204000)then
            rg_s <= '1';
            data_bus <= "01000110"; -- F char
            if(counter >= 2200000 and counter <= 2200100)then
               enable <= '1';
            else enable <= '0';
            end if ;

          elsif(counter > 2204000 and counter <= 2208000)then
            rg_s <= '1';
            data_bus <= "01011111"; -- _ char
            if(counter >= 2204000 and counter <= 2204100)then
               enable <= '1';
            else enable <= '0';
            end if ;

         elsif(counter > 2208000 and counter <= 2212000)then
            rg_s <= '1';
            data_bus <= "01010010"; -- R char
            if(counter >= 2208000 and counter <= 2208100)then
               enable <= '1';
            else enable <= '0';
            end if ;

        elsif(counter > 2212000 and counter <= 2216000)then
            rg_s <= '1';
            data_bus <= "01000001"; -- A char
            if(counter >= 2212000 and counter <= 2212100)then
               enable <= '1';
            else enable <= '0';
            end if ;

        end if;
      end if;


end process;


end Behavioral;
```

Constraint file:

```
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports enable]
set_property IOSTANDARD LVCMOS33 [get_ports r_w]
set_property IOSTANDARD LVCMOS33 [get_ports rg_s]
set_property PACKAGE_PIN Y9 [get_ports clk]
set_property PACKAGE_PIN W10 [get_ports r_w]

set_property PACKAGE_PIN AA8 [get_ports {data_bus[7]}]
set_property PACKAGE_PIN AB9 [get_ports {data_bus[6]}]
set_property PACKAGE_PIN AB10 [get_ports {data_bus[5]}]
set_property PACKAGE_PIN AB11 [get_ports {data_bus[4]}]
set_property PACKAGE_PIN AA9 [get_ports {data_bus[3]}]
set_property PACKAGE_PIN Y10 [get_ports {data_bus[2]}]
set_property PACKAGE_PIN AA11 [get_ports {data_bus[1]}]
set_property PACKAGE_PIN Y11 [get_ports {data_bus[0]}]
set_property PACKAGE_PIN V9 [get_ports enable]
set_property PACKAGE_PIN V12 [get_ports rg_s]
```

## Part B: 3-digit counter

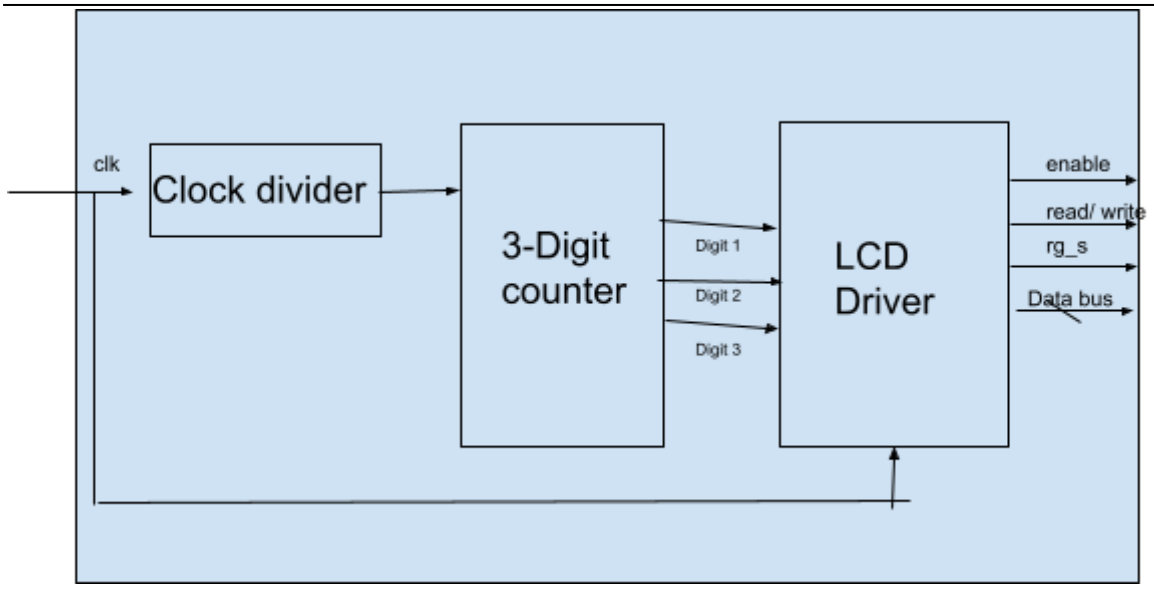


Fig1: 3-digit counter to display on LCD

## Implementation code:

### 3-Digit Counter :

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

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;

entity LCD_COUNTER is
    Port ( CLK : in STD_LOGIC;
           DIGIT0 : out STD_LOGIC_VECTOR (3 downto 0);
           DIGIT1 : out STD_LOGIC_VECTOR (3 downto 0);
           DIGIT2 : out STD_LOGIC_VECTOR (3 downto 0);
           Sign_dig : out STD_LOGIC_VECTOR (1 downto 0));
end LCD_COUNTER;

architecture Behavioral of LCD_COUNTER is
signal dig0,dig1,dig2:std_logic_vector (3 downto 0);
begin

process(clk)

begin
if(clk' event and clk='1') then
    if(dig0="1001") then
       dig0<="0000";
```

```
      if(dig1="1001")then
        dig1<="0000";
        if(dig2="1001")then
         dig2<="0000";
        else Sign_dig<="10"; dig2<=dig2+'1';
        end if;
      else Sign_dig<="01"; dig1<=dig1+'1';
      end if;
    else Sign_dig<="00"; dig0<=dig0+'1';
    end if;
end if;
end process;

end Behavioral;
```

**-- Sign_dig pin was to indicate how many digits have been updated in the count but we --do not use it!!**

Simulation part to insure that counter works fine:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity LCD_COUNTER_tb is

end LCD_COUNTER_tb;

architecture Behavioral of LCD_COUNTER_tb is

component LCD_COUNTER is
   Port ( CLK : in STD_LOGIC;
          DIGIT0 : out STD_LOGIC_VECTOR (3 downto 0);
          DIGIT1 : out STD_LOGIC_VECTOR (3 downto 0);
          DIGIT2 : out STD_LOGIC_VECTOR (3 downto 0);
          Sign_dig : out STD_LOGIC_VECTOR (1 downto 0));
end component;

signal clk :STD_LOGIC;
signal Digit0,Digit1,Digit2 : STD_LOGIC_VECTOR (3 downto 0) := "0000";
signal sign_dig : std_logic_vector(1 downto 0);
constant clk_period : time :=10 ns;
begin

uut: LCD_COUNTER port map (clk=> clk, DIGIT0=> Digit0,DIGIT1=>Digit1, DIGIT2=>Digit2,
Sign_dig=> Sign_dig );
```

```
        clk_generation : process
                begin
                clk<='0';
                wait for clk_period/2;
                clk<='1';
                wait for clk_period/2;
        end process;
end Behavioral;
```

Part of simulation output:

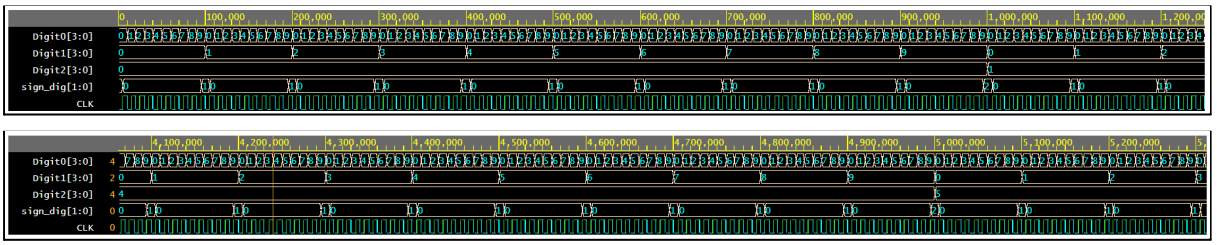


Fig1: part of the simulation

LCD Driver implementation:

---

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity LCD_Driver_counter is
    Port ( digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           Sign_dig: in STD_LOGIC_VECTOR (1 downto 0);
           clk : in STD_LOGIC;
           enable : out STD_LOGIC;
           rg_s : out STD_LOGIC;
```

```
        r_w : out STD_LOGIC;
        data_bus : out STD_LOGIC_VECTOR (7 downto 0));
end LCD_Driver_counter;

architecture Behavioral of LCD_Driver_counter is
signal counter : integer:=0;
signal counter_2 : integer:=0;
--signal flag : std_logic:='0';

begin

process(clk)
variable flag : std_logic:='0';
   begin
      if(clk'event and clk = '1')then

            counter<= counter + 1;
            if(counter < 2000000)then
               enable <= '0';
               rg_s <= '0';
               r_w <= '0';
            elsif(counter >= 2000000 and counter <= 2004000)then
               data_bus <= "00111000";
               if(counter >= 2000000 and counter <= 2000100)then
                  enable <= '1';
               else enable <= '0';
               end if;
            elsif(counter > 2004000 and counter <= 2008000)then
               data_bus <= "00000110"; -- entry mode set
               if(counter >= 2004000 and counter <= 2004100)then
                  enable <= '1';
               else enable <= '0';
               end if;
            elsif(counter > 2008000 and counter <= 2012000 ) then
               data_bus <= "00001100";
               if(counter >= 2008000 and counter <= 2008100)then
                  enable <= '1';
               else enable <= '0';
               end if;
            elsif(counter > 2012000 and counter <= 2176000)then
               data_bus <= "00000001"; -- finish config
               if(counter >= 2012000 and counter <= 2012100)then
                  enable <= '1';
               else enable <= '0';
               end if ;
```

```
            elsif(counter > 2176000 and counter <= 2180000)then
             data_bus <= "10000000"; -- address
             if(counter >= 2176000 and counter <= 2176100)then
                enable <= '1';
             else enable <= '0';
             end if ;
            elsif(counter > 2180000 and counter <= 2184000)then
                    rg_s <= '0';                -- 0 means address
                    data_bus <= "10000000";  -- address 0
                    if(counter> 2180000 and counter<= 2180100)then
                            enable <= '1';
                    else enable <= '0';
                    end if;
            elsif(counter > 2184000 and counter <= 2188000)then
                    rg_s <= '1';                -- 1 means data
                    data_bus <= ("0011" & digit2);  -- data  digit2 x00
                    if(counter > 2184000  and counter <= 2184100)then
                            enable <= '1';
                    else enable <= '0';
                    end if;
                            -- digit 1 , 0x0
            elsif(counter > 2188000 and counter <= 2188000 )then --(2188000  in the right
should -- be 2192000 )
                    rg_s <= '1';                -- 1 means data
                    data_bus <= ("0011" & digit1);  -- data  digit1 0x0
                    if(counter > 2188000  and counter <= 2188100)then
                            enable <= '1';
                    else enable <= '0';
                    end if;

             elsif(counter > 2192000 and counter <= 2196000)then
                    rg_s <= '1';                -- 1 means data
                    data_bus <= ("0011" & digit0);  -- data  digit1 0x0
                    if(counter > 2192000  and counter <= 2192100)then
                            enable <= '1';
                    else enable <= '0';
                    end if;
                    -- digit 0 , 00x
                    else counter <=2176100 ;       -- it should be 2176000.
                    end if;
             end if;


end process;
```

```
end Behavioral;
```

Top level entity implementation :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity LCD_Counter_top_level_entity is
    Port ( clk : in STD_LOGIC;
           data_bus : out STD_LOGIC_VECTOR(7 downto 0);
           enable : out STD_LOGIC;
           rg_s : out STD_LOGIC;
           r_w : out STD_LOGIC);
end LCD_Counter_top_level_entity;

architecture struct of LCD_Counter_top_level_entity is

component clock_divider
    Port ( CLK_IN : in STD_LOGIC;
           CLK_OUT : out STD_LOGIC);
end component;

component LCD_COUNTER
     Port ( CLK : in STD_LOGIC;
           DIGIT0 : out STD_LOGIC_VECTOR (3 downto 0);
           DIGIT1 : out STD_LOGIC_VECTOR (3 downto 0);
           DIGIT2 : out STD_LOGIC_VECTOR (3 downto 0);
           Sign_dig : out STD_LOGIC_VECTOR (1 downto 0));
end component;

component LCD_Driver_counter
    Port ( digit0 : in STD_LOGIC_VECTOR (3 downto 0);
           digit1 : in STD_LOGIC_VECTOR (3 downto 0);
           digit2 : in STD_LOGIC_VECTOR (3 downto 0);
           Sign_dig: in STD_LOGIC_VECTOR (1 downto 0);
           clk : in STD_LOGIC;
           enable : out STD_LOGIC;
           rg_s : out STD_LOGIC;
           r_w : out STD_LOGIC;
           data_bus : out STD_LOGIC_VECTOR (7 downto 0));
end component;
```

```
signal digit0_s, digit1_s, digit2_s : std_logic_vector(3 downto 0);
signal clk_one_sec :std_logic;
signal sign_dig_s : std_logic_vector(1 downto 0);

begin
clk_divider : clock_divider port map (CLK_IN => clk, CLK_OUT =>clk_one_sec);
counter : LCD_COUNTER port map (CLK =>clk_one_sec, DIGIT0 => digit0_s, DIGIT1 =>
digit1_s, DIGIT2 => digit2_s,Sign_dig=> Sign_dig_s);
driver : LCD_Driver_counter port map(digit0 =>digit0_s, digit1 =>digit1_s, digit2 =>digit2_s,
Sign_dig=>Sign_dig_s, clk=>clk, enable =>enable, rg_s=> rg_s, r_w=>r_w,
data_bus=>data_bus );
end struct;
```

Constraint file:

```
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {data_bus[0]}]
set_property PACKAGE_PIN AA8 [get_ports {data_bus[7]}]
set_property PACKAGE_PIN AB9 [get_ports {data_bus[6]}]
set_property PACKAGE_PIN AB10 [get_ports {data_bus[5]}]
set_property PACKAGE_PIN AB11 [get_ports {data_bus[4]}]
set_property PACKAGE_PIN AA9 [get_ports {data_bus[3]}]
set_property PACKAGE_PIN Y10 [get_ports {data_bus[2]}]
set_property PACKAGE_PIN AA11 [get_ports {data_bus[1]}]
set_property PACKAGE_PIN Y11 [get_ports {data_bus[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports enable]
set_property IOSTANDARD LVCMOS33 [get_ports r_w]
set_property IOSTANDARD LVCMOS33 [get_ports rg_s]
set_property PACKAGE_PIN Y9 [get_ports clk]
set_property PACKAGE_PIN V9 [get_ports enable]
set_property PACKAGE_PIN W10 [get_ports r_w]
set_property PACKAGE_PIN V12 [get_ports rg_s]
```

# Conclusion:

In the experiment we learned how to design and implement a driver that will be used to handle the parallel communication between the external LCD module and the ZedBoard.and we'll check our result on lcd.