An-Najah National University
Department of Computer Engineering
Digital design 2 LAB -10636391-
Second Semester 2020/2021

جامعة النجاح الوطنية
قسم هندسة الحاسوب
مختبر تصميم دوائر الكترونية 2 - 10636391-
الفصل الثاني 2021/2020

Instructor: Anas Toma

NAME	ID NUM	SEC
1- أحمد عبد الله جبر القرم	11819195	3/8:00-11:00
-	-	-

Experiment 5: LCD Driver

Introduction :

In this lab , I will design and implement a driver that will handle the parallel communication between LCD and the ZedBoard.

1- PmodCLP :

The Character LCD with Parallel Interface module from Digilent is a 16×2 character LCD to let the system boards display up to 32 different characters.

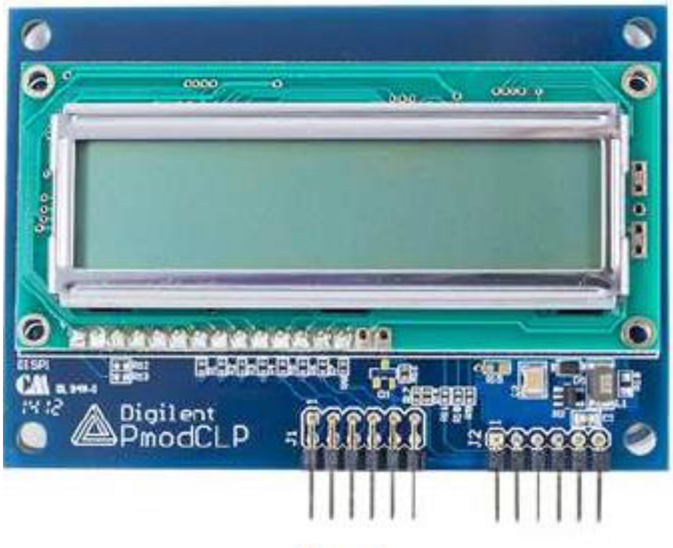


Figure 1

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, the value in the figure 1 needs to be written into the DD-RAM. For example, the character ‘S’ from the CGROM would have the value “01010011”.

These values are written to the location address of the LCD. The address of DD-RAM location is first sent to LCD then the code of the symbol to be displayed. The codes for characters are as follows:

Figure 1: **LCD Character Set**

The LCD physical locations correspond to addresses in the DD-RAM as follows:

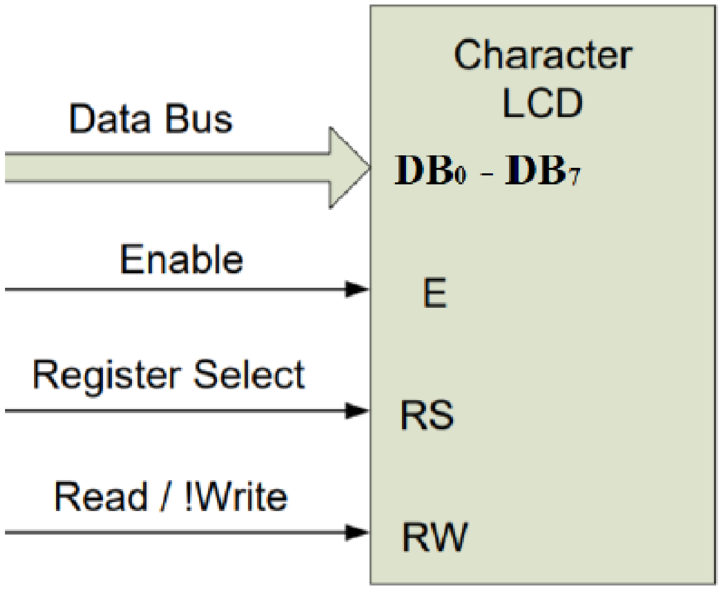
Character Display Addresses / **Undisplayed Addresses**

1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	…	27
2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	…	67
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**…**	**40**

DD RAM Hexadecimal Addresses (No Display Shifting)

2- **Interfacing with Pmod :**

The PmodCLP utilizes a Samsung KS0066 LCD controller to display information to a 16x2 LCD panel which has both kinds of interface types: 4-bit bus and 8-bit bus. The PmodCLP module has 8 bits interface as shown in the next figure.



Pmod CLP module should be connected with two Digilent Pmod™ compatible headers (2x6). Therefore, it can be connected to JA and JB connectors from the ZedBoard. The pinout for this module is described in the following table.

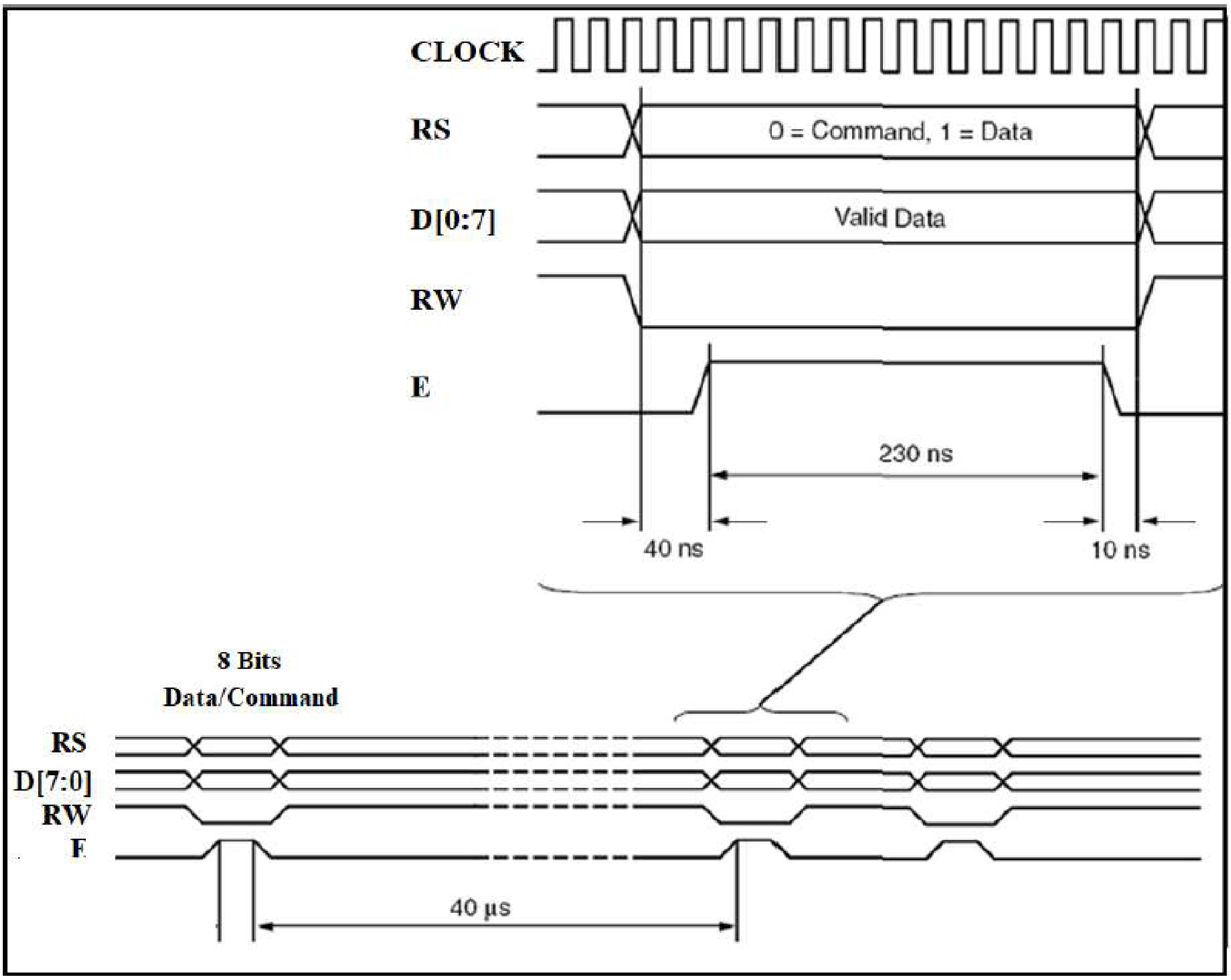
Header J1 – Top Half			Header J1 – Bottom Half		
Pin	**Signal**	**Description**	**Pin**	**Signal**	**Description**
1	DB0	Data Bit 0	7	DB4	Data Bit 4
2	DB1	Data Bit 1	8	DB5	Data Bit 5
3	DB2	Data Bit 2	9	DB6	Data Bit 6
4	DB3	Data Bit 3	10	DB7	Data Bit 7
5	GND		11	GND	
6	VCC		12	VCC	

<table>
<tr><th colspan="3">Header J2 – Top Half</th><th></th><th colspan="3">Header J2 – Bottom Half</th></tr>
<tr><th>Pin</th><th>Signal</th><th>Description</th><th></th><th>Pin</th><th>Signal</th><th>Description</th></tr>
<tr><td>1</td><td colspan="2" rowspan="6">Not used</td><td></td><td>7</td><td>RS</td><td>Register select</td></tr>
<tr><td>2</td><td></td><td>8</td><td>RW</td><td>Read/Write</td></tr>
<tr><td>3</td><td></td><td>9</td><td>E</td><td>Enable</td></tr>
<tr><td>4</td><td></td><td>10</td><td>NC</td><td>Not connected</td></tr>
<tr><td>5</td><td></td><td>11</td><td>GND</td><td></td></tr>
<tr><td>6</td><td></td><td>12</td><td>VCC</td><td></td></tr>
</table>

3- **Timing Requirements:**

The Pmod CLP module communicates with the host board via the GPIO protocol. This particular module requires specific timings in order to program the LCD correctly. Because the Pmod CLP module has 8 bits interface, any 8-bit data/command can be sent

in a single transition. The general timing diagram is shown below.



In case of two successive commands, they have to be separated with 40 µ s delay (4000 clock cycles).

From the above diagram, it is important to notice the following:

- Setup time (time for the outputs to stabilize) is 40ns (4 clock cycles).
- The hold time (time to assert the enable (E) pin) is 230ns (23 clock cycles).
- The fall time (time to allow the outputs to stabilize) is 10ns (1 clock cycle).

You can simplify the process as follows:

1. Select the correct value for RS, RW, and Data bus
2. Set E to 1
3. At least 230ns must elapse. (Use 1 µ s : 100 Clock cycles)
4. Set E to 0

Objectives:

- Lern how to display my name in the LCD
- Lern how to display counter in the LCD

Tools used in Lap :

1- Computer lap.

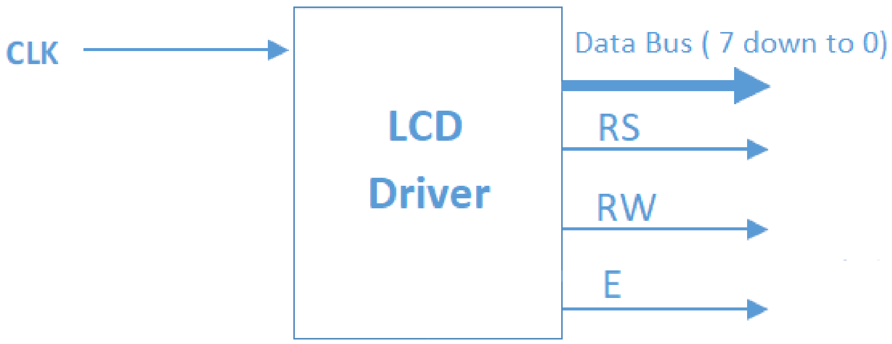
2- Vivado software .

3- Zboard from Xilinx.

4- VHDL

5- PmodCLP

6- two Digilent Pmod™ compatible headers (2x6)



Procedure :

Part 1 : Display my full Name (with shifted display)

☒ **Startup:**

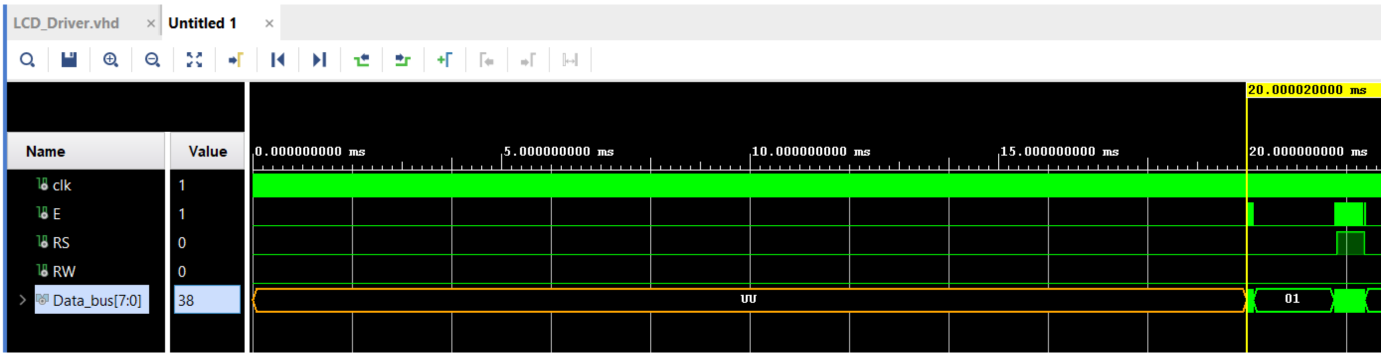
After the power on I wait for 20 ms (2 000 000 clock cycles at 100 MHz) before start configuration Code and

through this duration RS , RW and E was in logic '0' as we see :

VHDL code :

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity LCD_Driver is
    Port (
            clk : in STD_LOGIC;
            E : out STD_LOGIC;
            RS : out STD_LOGIC;
            RW : out STD_LOGIC;
            Data_bus : out STD_LOGIC_VECTOR (7 downto 0)
            );
end LCD_Driver;

architecture Behavioral of LCD_Driver is
signal myCounter,con_C,dis_C:integer range 0 to 10000000:=0;
signal start_done,con_done,dis_char_done:std_logic :='0';
begin
process( clk )
begin
if (clk'event and clk='1')then
    myCounter <= myCounter + 1 ;
    if (start_done='0' )then--startup
            RS<='0';
            RW<='0';
            E<='0';
        if (myCounter=2000000)then
            myCounter <= 0;
            start_done<='1';
        end if;
```

Simulation :



☒ **Configuration :**

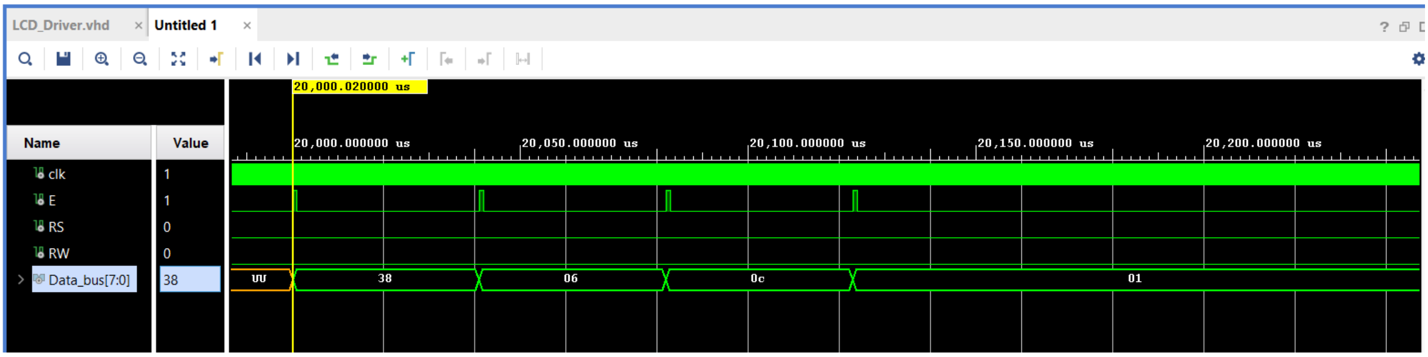
1. Issue a Function Set command, 0x38, to configure the display for operation on 8-bit data.
2. Wait for 40 µs (4000 clock cycles).
3. Issue an Entry Mode Set command, 0x06, to set the display to automatically increment the address pointer.
4. Wait for 40 µs (4000 clock cycles).
5. Issue a Display On/Off command, 0x0C, to turn the display on and disables the cursor and blinking.
6. Wait for 40 µs (4000 clock cycles)
7. Finally, issue a Clear Display command. Allow at least 1.64 ms (164,000) clock cycles).

VHDL code :

```vhdl
elsif (start_done='1' and con_done='0') then--configration
    if (myCounter = 1)then
        RS <= '0';
        RW <= '0';
        Data_bus <= x"38";          --Function Set command, 0x38
        E <= '1';
    elsif (myCounter = 100) then
        E <= '0';

    elsif (myCounter = 4100)then
        RS <= '0';
        RW <= '0';
        Data_bus <= x"06";          --Entry Mode Set command, 0x06
        E <= '1';
    elsif (myCounter = 4200) then
        E <= '0';

    elsif (myCounter = 8200 )then
        RS <= '0';
        RW <= '0';
        Data_bus <= x"0C";          --Display On/Off command, 0x0C
        E <= '1';
    elsif (myCounter = 8300) then
        E <= '0';

     elsif (myCounter = 12300)then
        RS <= '0';
        RW <= '0';
        Data_bus <= x"01";          --Clear Display command, 0x01
        E <= '1';
    elsif (myCounter = 12400) then
```

```
        elsif (myCounter = 12400) then
            E <= '0';

        elsif (myCounter = 176400)then-- wait for 1.64 ms

            myCounter <= 0;
            con_done <= '1';
        end if;
```

Simulation:



☒ Display :

In this stage I will send the commands that will print characters on LCD ,To show a character on the LCD, I have to:

1. Specify the start address with a Set DD-RAM Address command.
2. Display a character with a Write Data command.

VHDL code:

```
elsif (start_done='1' and con_done='1' and dis_char_done='0' )then --display
    if (myCounter = 1)then
        RS <= '0';
        RW <= '0';
        Data_bus <= x"8F";          --addres at 01 location command, 0x38
        E <= '1';
    elsif (myCounter = 100) then
        E <= '0';

    elsif (myCounter = 4100)then
        RS <= '1';
        RW <= '0';
        Data_bus <= x"41";          --          1 => X"41",--A
        E <= '1';
    elsif (myCounter = 4200) then
        E <= '0';

    elsif (myCounter = 8200)then
        RS <= '1';
        RW <= '0';
        Data_bus <= x"68";          --          2 => X"68",--h
        E <= '1';
    elsif (myCounter = 8300) then
        E <= '0';
```

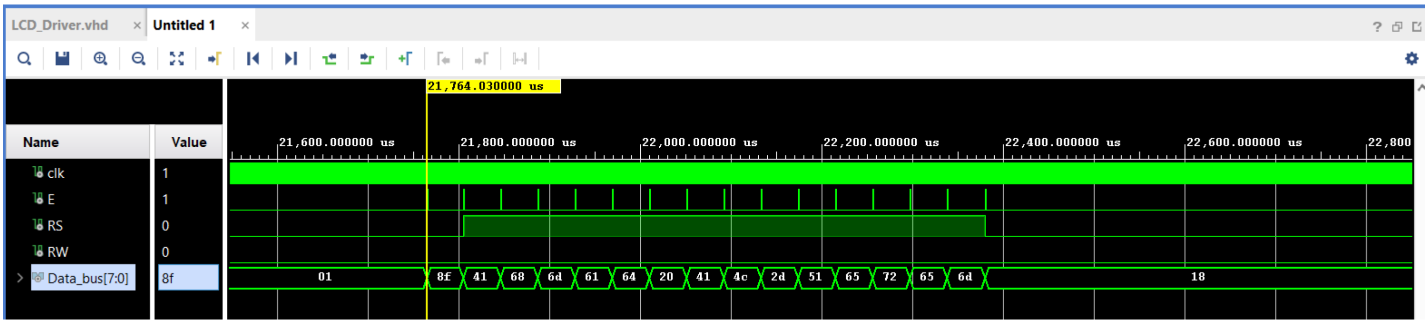
•

•

•

```
elsif (myCounter = 53300)then
    RS <= '1';
    RW <= '0';
    Data_bus <= x"65";          --        13 => X"65",--e
    E <= '1';
elsif (myCounter = 53400) then
    E <= '0';

elsif (myCounter = 57400)then
    RS <= '1';
    RW <= '0';
    Data_bus <= x"6D";          --        14 => X"6D",--m
    E <= '1';
elsif (myCounter = 57500) then
    E <= '0';

elsif (myCounter = 61500) then
    myCounter <= 0;
    dis_char_done<='1';
     end if;
```

Simulation:



☒ Shift Display (bonus part):

After write data in the memory in the LCD . I will shift display left every 1 s and show charters one by one using shift display left command (0x18)

```
elsif (dis_char_done='1')then
    if (myCounter = 1)then
    RS <= '0';
    RW <= '0';
    Data_bus <= x"18";          --        14 => X"6D",--shift display left command
    E <= '1';
elsif (myCounter = 100) then
    E <= '0';

elsif (myCounter=50000100)then
       myCounter<=0;
       dis_char_done<='0';
end if;
```

<u>Synthesis, Implementation, and Bitstream Generation :</u>

I had connected Pins in the board as shown in the Table

	Signal Name	Package Pin	I/O Std
DB0	JA1	Y11	LVCMOS33
DB1	JA2	AA11	LVCMOS33
DB2	JA3	Y10	LVCMOS33
DB3	JA4	AA9	LVCMOS33
DB4	JA7	AB11	LVCMOS33
DB5	JA8	AB10	LVCMOS33
DB6	JA9	AB9	LVCMOS33
DB7	JA10	AA8	LVCMOS33
RS	JB7	V12	LVCMOS33
RW	JB8	W10	LVCMOS33
E	JB9	V9	LVCMOS33

```
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 Y11 [get_ports {Data_bus[0]}]
set_property PACKAGE_PIN AA11 [get_ports {Data_bus[1]}]
set_property PACKAGE_PIN Y10 [get_ports {Data_bus[2]}]
set_property PACKAGE_PIN AA9 [get_ports {Data_bus[3]}]
set_property PACKAGE_PIN AB11 [get_ports {Data_bus[4]}]
set_property PACKAGE_PIN AB10 [get_ports {Data_bus[5]}]
set_property PACKAGE_PIN AB9 [get_ports {Data_bus[6]}]
set_property PACKAGE_PIN AA8 [get_ports {Data_bus[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports E]
set_property IOSTANDARD LVCMOS33 [get_ports RS]
set_property IOSTANDARD LVCMOS33 [get_ports RW]
set_property PACKAGE_PIN Y9 [get_ports clk]
set_property PACKAGE_PIN V9 [get_ports E]
set_property PACKAGE_PIN V12 [get_ports RS]
set_property PACKAGE_PIN W10 [get_ports RW]
```

And check the result :

Part 2: Decimal counter from (00 to 99):

Top level Entity LCD DRIVER :

contain two component

1- Clk Divider.
2- Two digit BCD Counter.

☒ Startup : the same as the previous part .

☒ Configuration : also the same as the previous part.

☒ Display :

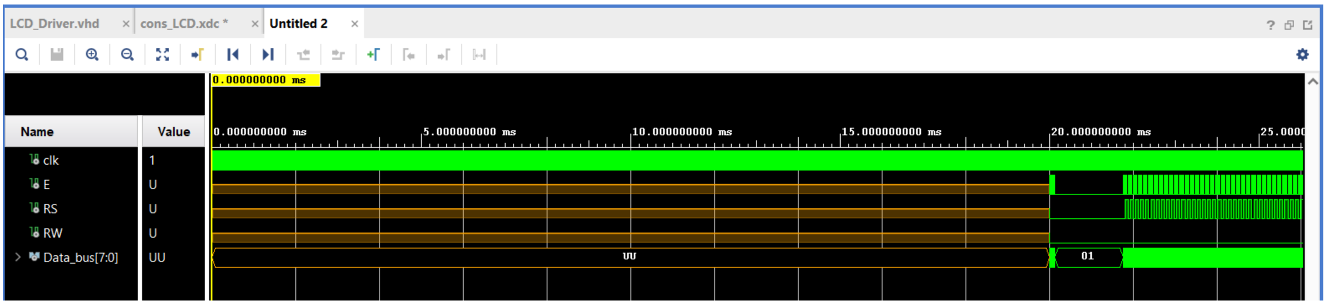
In this stage I selected the address to display the two digits and then I can easily create the number to display command by concatenated the output from the BCD counter with value " 3 " to become as : "3"+Digit 0 and "3"+Digit 1.

VHDL code :

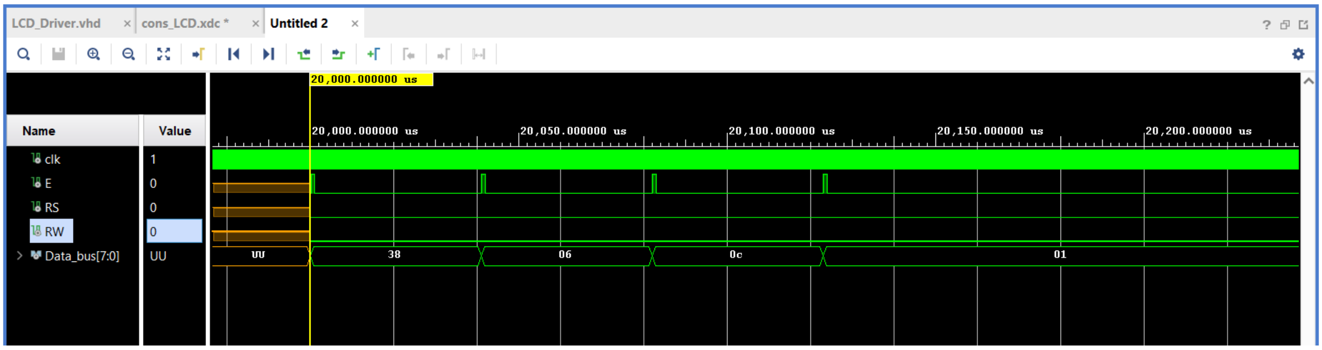
```
    elsif (start_done='1' and con_done='1')then --display
        if (dis_C = 0)then
            RS <= '0';
            RW <= '0';
            Data_bus <= x"86";          --addres at 06 location command, 0x38
            E <= '1';
        elsif (dis_C = 100) then
            E <= '0';

        elsif (dis_C = 4100)then
            RS <= '1';
            RW <= '0';
            Data_bus <= "0011"&Digit1;        --        1 => dig 1
            E <= '1';
        elsif (dis_C = 4200) then
            E <= '0';

        elsif (dis_C = 8200)then
            RS <= '1';
            RW <= '0';
            Data_bus <= "0011"&Digit0;        --        2 => dig 0
            E <= '1';
        elsif (dis_C = 8300) then
            E <= '0';
        end if;
        dis_C <= dis_C+1;
        if (dis_C=12300)then
            dis_C<=0;
```

Simulation :

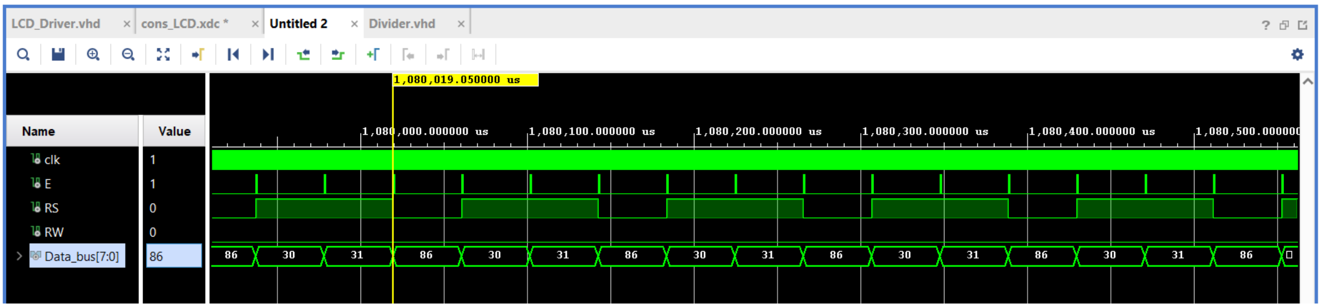
1- Wait for 20 ms:



2- Configuration commands :



3- Set address and display two digits :



<u>Synthesis, Implementation, and Bitstream Generation :</u>

I had connected Pins in the board as shown previous in the Table and tested the circuit .

<u>Conclusion:</u>

In this experiment, I learned how to deal with the LCD module attached with the Zedboard. I also learned how an LCD Driver works by giving extremely specifically timed commands that get the module working in the preferred mode.
I also learned how the RAMs and ROM present in the module operate to show data on the screen.
In conclusion, the LCD module is an important part of a system for showing textual information to the user, and one that can be easily integrated and programmed with special commands and instructions.

Source and References:

- ZedBoard User's Guide:
  http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf

  https://reference.digilentinc.com/_media/reference/pmod/pmodclp/ks0066.pdf

- Pmod CLP Reference Manual
  https://reference.digilentinc.com/reference/pmod/pmodclp/reference-manual

- Programmable Logic Master User Constraints
  https://reference.digilentinc.com/_media/reference/pmod/pmodclp/ks0066.pdf

  http://zedboard.org/sites/default/files/documentations/zedboard_master_XDC_RevC_D_v3.zip

- stack overflow
  https://stackoverflow.com/

- Youtube
  https://www.youtube.com/