Παναγιώτης Κρεμμύδας 1435

1/19/2016

# Lab exercize 4:

# LCD component driver

###### CE430: Digital Circuits Lab

###### Electrical& Computer Engineering

###### University of Thessaly

# Contents

0: Introduction

1: LCD Command Set modulation: Initialization sequence transmitted prior any instructions that establishes the four-bit interface to the LCD component and signal modulation for communicating through that interface

2: LCD Character Display Control: Sends commands to the LCD controller through the Command Set modulation module reading the appropriate blocks from the BRAM.

3: Challenges & Solutions in the designing, testing and implementing of the above

# Introduction:

Implementation of a Liquid Crystal Display driver. The goal is to experiment with the various functions of the Spartan3E's onboard LCD.

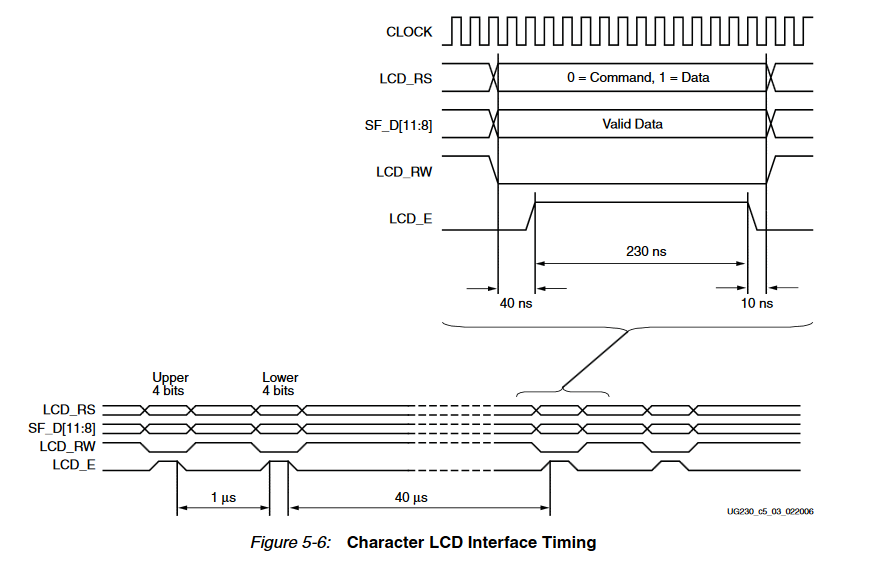
The suggested sample string for testing the aforementioned driver is the typical message ABCDEFGHIJKLMNOPabcdefghijklmno , the last(32nd) character on the LCD being a clockwise rotating single digit cursor. The message is registered in the BRAM memory of the FPGA and the LCD driver is utilized to continuously drive it in the LCD refreshed in 1 second intervals (including the display of the rotating cursor implementation).

Placement of the BRAM should be preconfigured to take place in the uppermost left part of the available memory on the Spartan3E board.

FPGAs like the Spartan3E board typically have a preconfigured synthesizable processor to drive the LCD and make the implementation less complicated . Because the use of the above processor can utilize a big portion of the LUTs and registers of the FPGA, it can prohibit functionallity or increase the hardware requirements of an FPGA implementation. This project involves direct implementation of some of the LCD driver's fanctions in the main system tailored for minimum footprint in the available resources.

# Part 1 - LCD Command Set modulation:

## Implementation



As shown in the above figure of the Xilinx Spartan3 manual write operations to the LCD are taking place by initializing the LCD\_RS, LCD\_RW, and SF\_D[11:8] signals at least 40 ns before the LCD\_E (enable signal) goes High. The LCD\_E signal must remain High for 230 ns or longer.

A power-on initialization sequence is transmitted prior any instructions. After the power-on initialization is completed,the four-bit interface is now established.

The name of the module implementing tha above is: Sync\_10\_bit\_interface:

3state FSM:

DATA\_INACTIVE: Communication interface remains inactive or performs the Power-On Initialization.

DATA\_INITIALIZATION: initializing the LCD\_RS, LCD\_RW, and SF\_D[11:8] signals according to the manufacturer’s specified modulation

ENABLE\_MODULATION\_START: initializing the LCD\_E signal according to the manufacturer’s specified modulation

Waitingtime:Configures the following delays: 1us (transmision of the next 4bits), 40us (transmision of the next command), 15000us (display activation time), 1.64ms or 1s depending on the LCD\_RS input (Clear Dsiplay or Display refresh period)

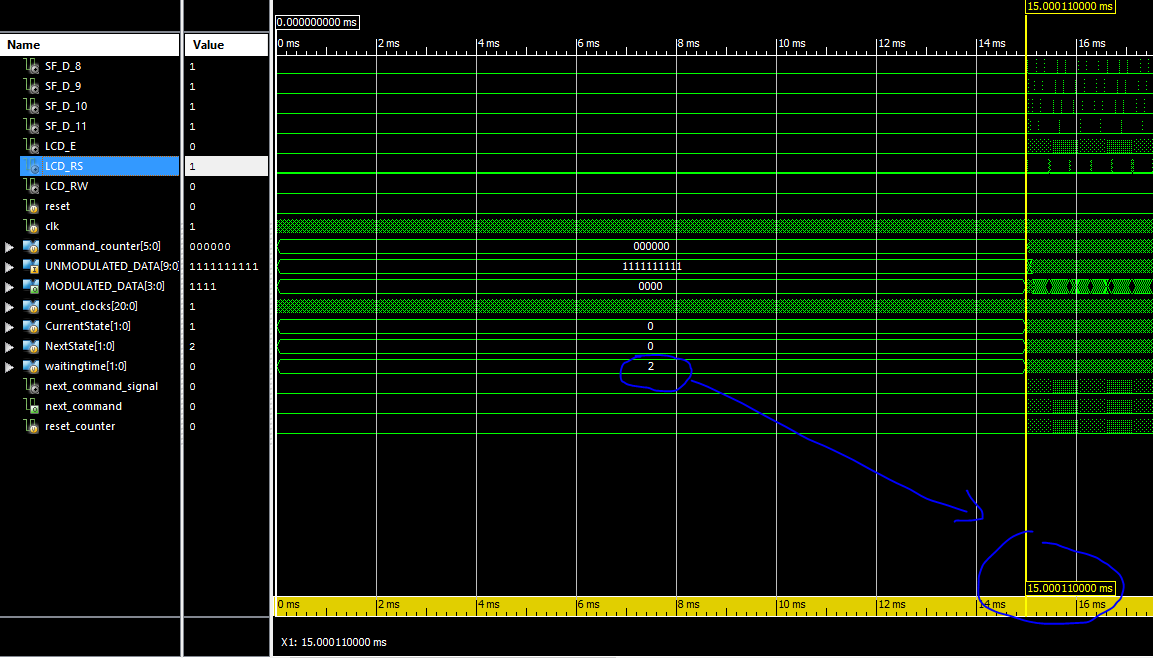
LCD\_RW = 0; Display permanently accepts data in write mode. The LCD\_RW signal can be tied Low permanently because the FPGA generally has no function in reading information in the current implementation.

next\_command*: Signals a posedge when next command transmition is available (implemented for partB)*

Module data input: [10:0] UNMODULATED\_DATA: { LCD\_E , LCD\_RS , LCD\_RW=1'b0 , 8bit command set} In the current implementation LCD\_E is 0 when running Clear Display or Display refresh functions. LCD\_RS is 0 when not writing characters to the LCD screen and LCD\_RW is always 0 because no characters are read using the LCD component.

<https://github.com/kmd178/Digital_Systems_lab4_LCD/tree/1c6729b569a659d839f0b7cd28ae04cb531f7ad7>

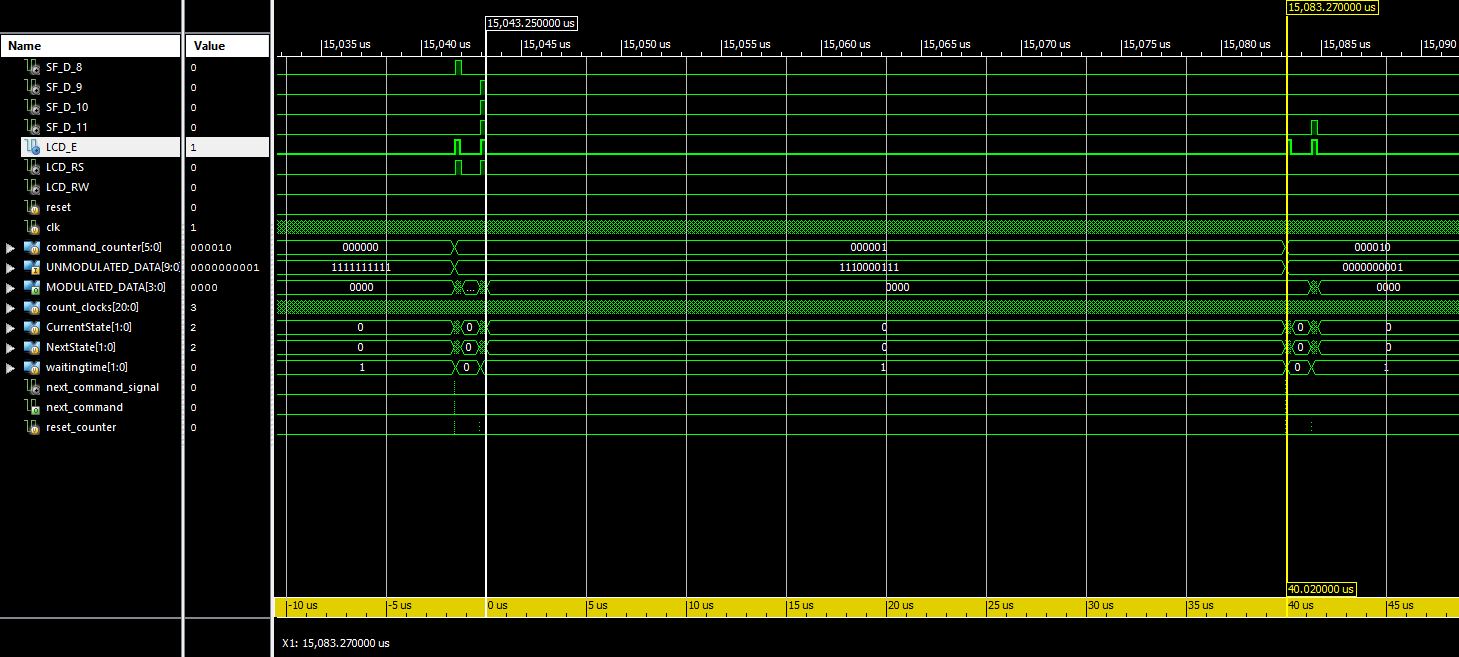
## Verification



15ms: Delay for display activation

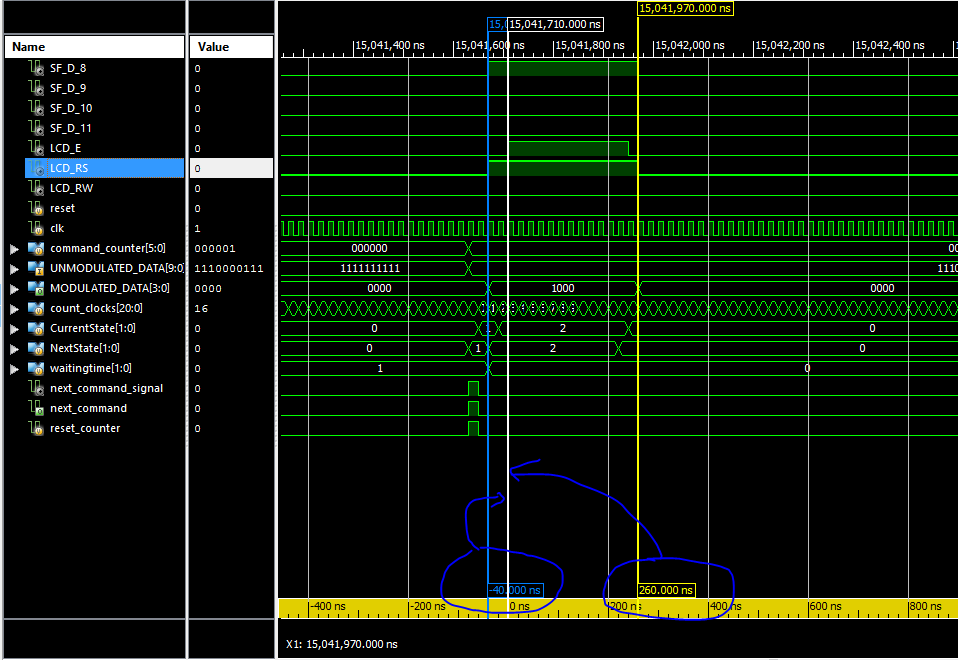


**1us:** Delay for the next 4bit transmision



**40us:** Delay for the next command transmision

40-240ns: Modulation of LCD\_RS, LCD\_RW, and SF\_D[11:8], LCD\_E signals



As indicated the corresponding output is assigned through the initialized memory. Output waveforms are performing as expected.

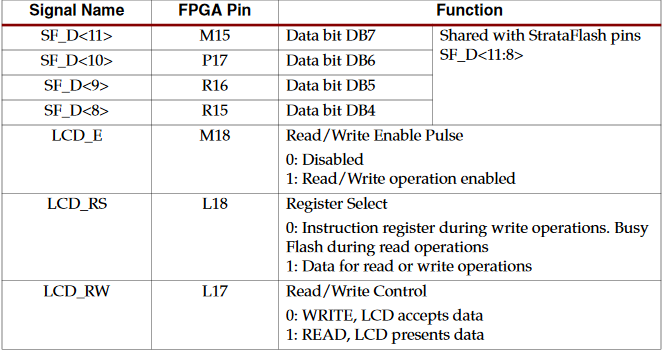
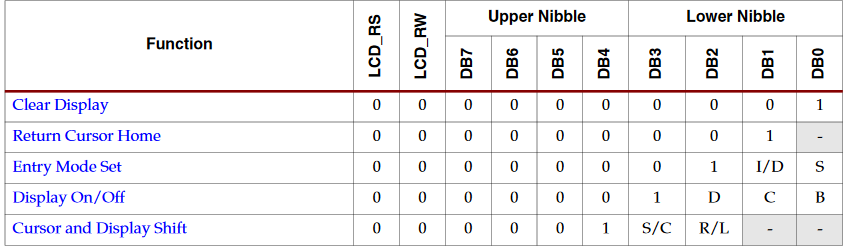
## Experiment/Resulting implementation

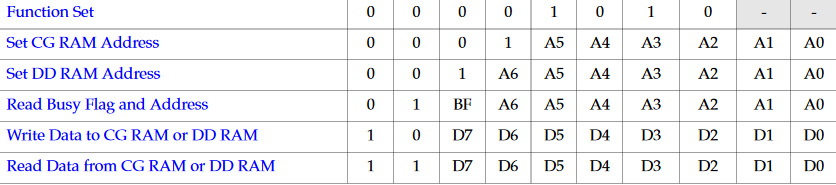
FPGA board testing was not necessary for this part of the assignment

# Part 2 - *LCD Character Display Control*:

## Implementation

As shown in the Xilinx Spartan3 manual the following tables describe the commands available:





Display Configuration is done with the following 4 commands that utilize the functions of the LCD through their specified parameters (Display characters stored in DD RAM,No cursor, Auto-increment address counter, Shifting disable, Clear Display).

Command counter: Increments by 1 every time a command finishes execution in the sync\_10bit\_interface module (Will increments 2 , 3 or 4 depending for displaying the 32nd character according to the currently active iteration). The corresponding LCD\_RS and LCD\_E signals to the active commands are assigned in the lcd\_controller module according to that counter. (a better implementation on different specifications would require 10 bit slots without character reading or 11 bit slots including the character reading function)

Function Set

Sets interface data length, number of display lines, and character font.

The Starter Kit board supports a single function set with value 0x28.

Execution Time: 40µs 00001010-- -> 0000101000 = 0x28

Entry Mode Set

Sets the cursor move direction and specifies whether or not to shift the display.

These operations are performed during data reads and writes.

Execution Time: 40µs //00000001-- -> 0000000110 = 0x06

Bit DB1: (I/D) Increment/Decrement

0 Auto-decrement address counter. Cursor/blink moves to left.

1 Auto-increment address counter. Cursor/blink moves to right.

Bit DB0: (S) Shift

0 Shifting disable

1During a DD RAM write operation, shift the entire display value in the direction

controlled by Bit DB1 (I/D). Appears as though the cursor position remains c onstant and the display moves.

Display On/Off

Display is turned on or off, controlling all characters, cursor and cursor position character

(underscore) blink.

Execution Time: 40µs 0000001--- -> 0000001100 = 0x0C

Bit DB2: (D) Display On/Off

0No characters displayed. However, data stored in DD RAM is retained

1Display characters stored in DD RAM

Bit DB1: (C) Cursor On/Off

0No cursor

1Display cursor

Bit DB0: (B) Cursor Blink On/Off

0No cursor blinking

1Cursor blinks on and off approximately every half second

Clear Display

Clear the display and return the cursor to the home position, the top-left corner.

This command writes a blank space (ASCII/ANSI character code 0x20) into all DD RAM

addresses. The address counter is reset to 0, location 0x00 in DD RAM. Clears all option

settings. The I/D control bit is set to 1 (increment address counter mode) Execution Time: 82µs - 1.64 ms = 82.000 cycles. 0000000001 = 0x01

## The following command set is stored into 8bit BRAM slots:

0:Function Set 0000101000 0x28

1:Entry Mode Set 0000000110 0x06

2:Display On/Off 0000001100 0x0C

3:Clear Display 0000000001 0x01

4: -Blank- 0000000000 0x00 Wait 1,64ms

5: CGRAM SET ADRESS 0001 000 001 0x41 {rs,rw,7,6,5,4,3,2,1,0}

6: INSERT " ^ " 1000011111 0x1F

7: CGRAM SET ADRESS 0001 001 001 0x49 //Implemented on the hypothesys that every WRITE CHAR will iterate CGRAM Memory address by 1 (meaning next row on the 5x8bitmap)

8: INSERT " |" 1000010000 0x10

9: INSERT " |" 1000010000 0x10

10: INSERT " |" 1000010000 0x10

11: INSERT " |" 1000010000 0x10

12: INSERT " |" 1000010000 0x10

13: CGRAM SET ADRESS 0001 010 110 0x56

14: INSERT " \_ " 1000011111 0x1F

15: CGRAM SET ADRESS 0001 011 001 0x59

16: INSERT "| " 1000000001 0x01

17: INSERT "| " 1000000001 0x01

18: INSERT "| " 1000000001 0x01

19: INSERT "| " 1000000001 0x01

20: INSERT "| " 1000000001 0x01

21: DDRAM SET ADRESS 001 0000000 {rs,rw,7,6,5,4,3,2,1,0} rs=0 0x80

...

22-37: WRITE CHAR ON THE SPECIFIED ADRESS (+ITERATION) rs=1 0x41 until 0x50

...

38: DDRAM SET ADRESS 001 011111 {rs,rw,7,6,5,4,3,2,1,0} rs=0 0xC0

...

39-53: WRITE CHAR ON THE SPECIFIED ADRESS rs=1 0x61 until 0x6F

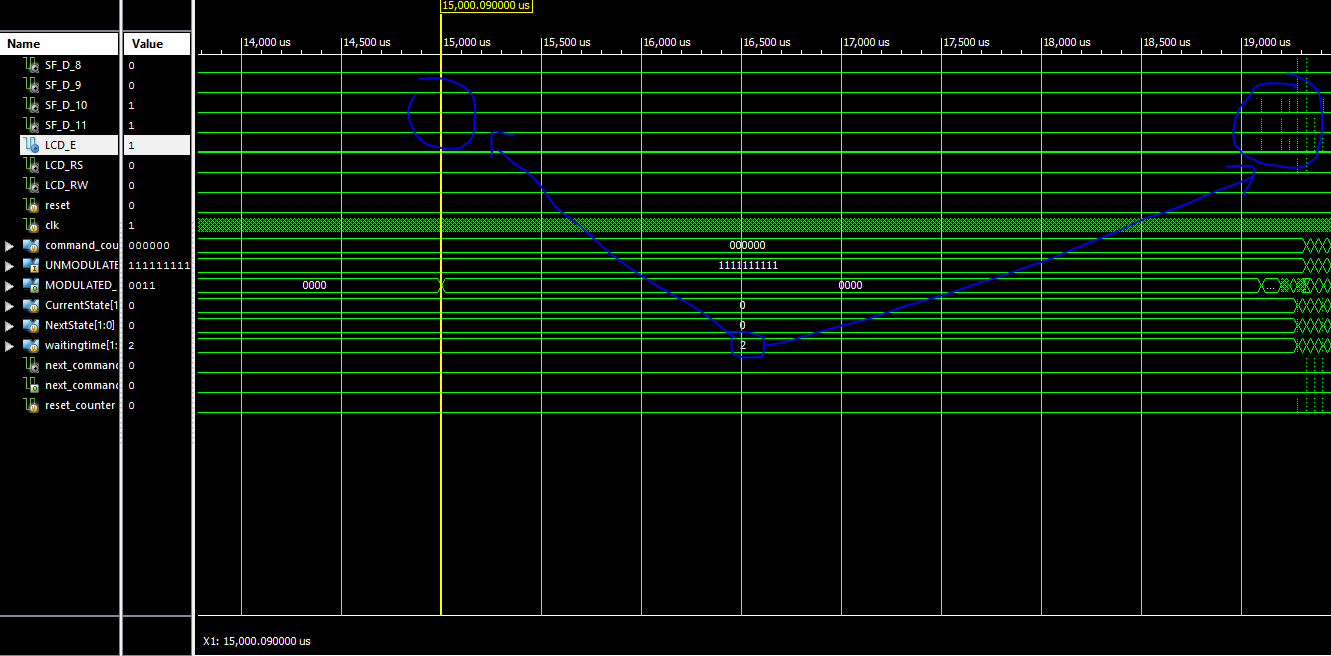
...

54-57: WRITE CHAR (rotationally every 1 second loop) rs=1 0x00 or 0x01 or 0x02 or 0x03

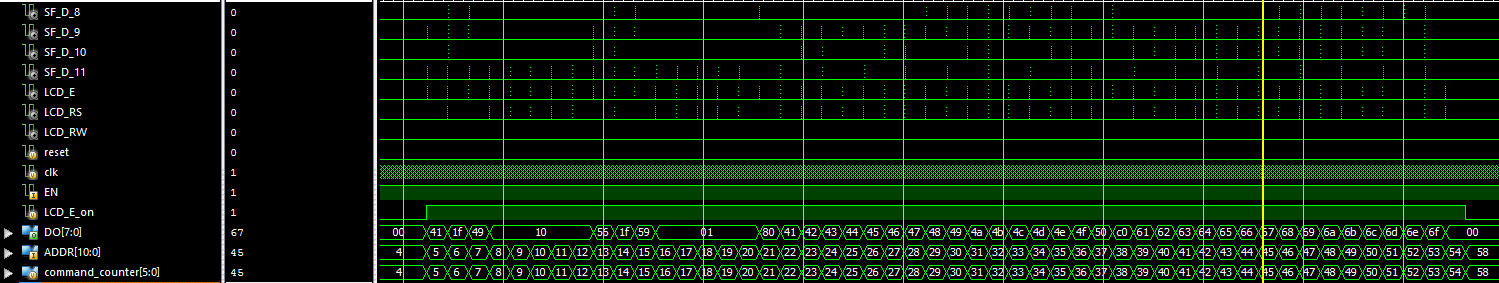
58: -Blank- 00000000 Wait 1 second and repeat from 21st command

## <https://github.com/kmd178/Digital_Systems_lab4_LCD/tree/96d0798722fa253aad8cf551160a57209614bffb>

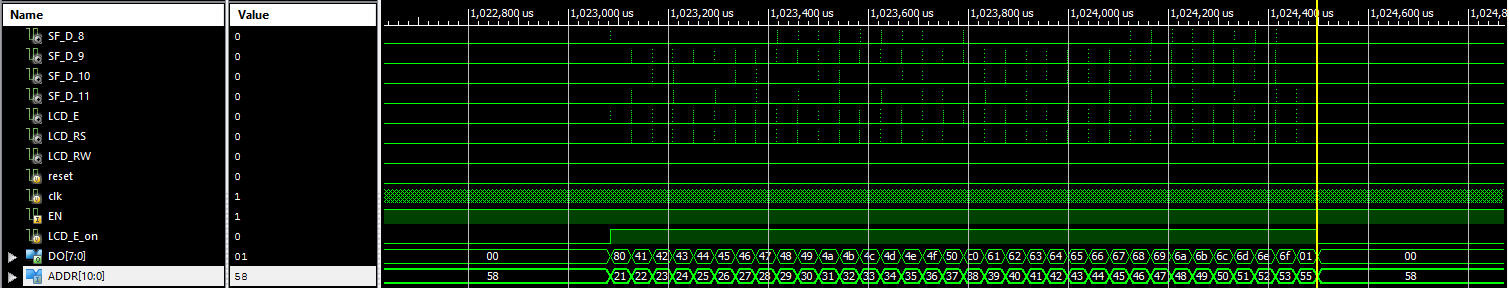
## Verification



Initialization modulation

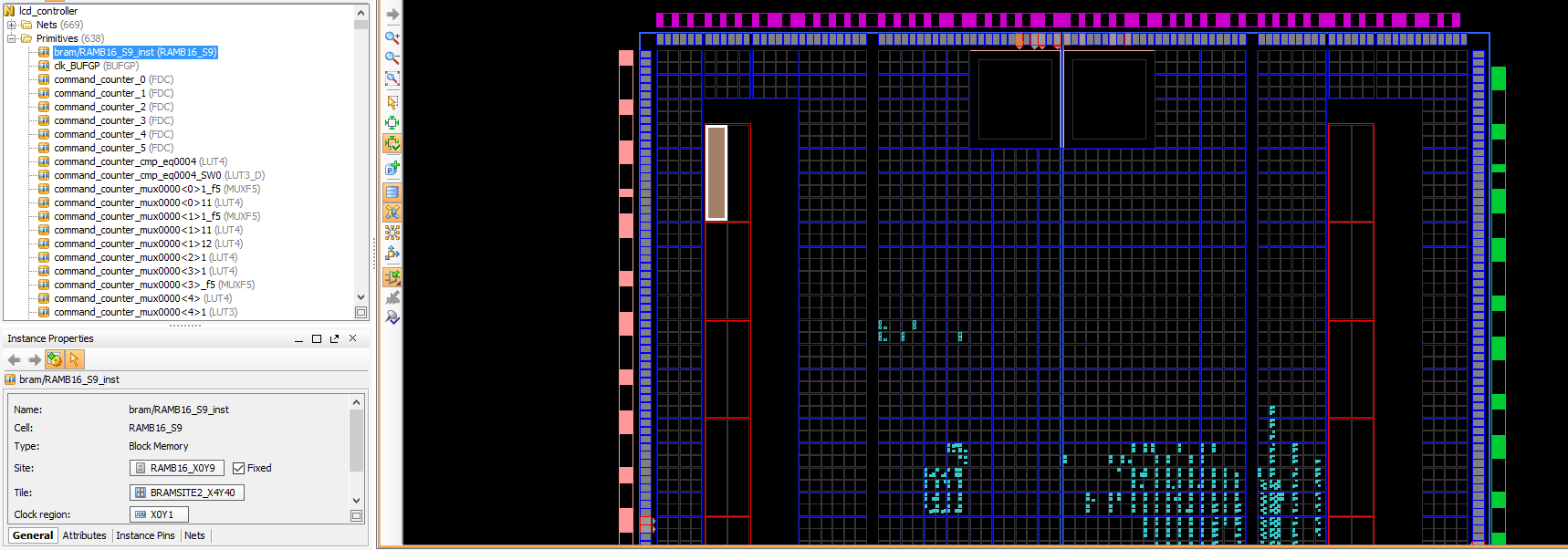


Instruction set



Instruction set for the 1second intervals

Output waveforms are performing as expected.



BRAM placement configured to take place in the uppermost left part of the available memory on the Spartan3E board as indicated.

## Experiment/Resulting implementation

First trial: LCD displaying unrecognizable characters

Second trial: Implentation without displaying characters or assigning bitmap data to the CGRAM resulted in a similar behavior

Third trial: Implementation without using SET ADDRESS commands 0x80 or 0xC0 gave expected results.

The configuration is set to display the message “ABCDE “ rotationally refreshing in 1 second intervals.

(The driver keeps writing data on the DDRAM until its address counter iterates to the beginning where it displays the data written again while going through the 0x00 to 0x0F and 0x40 to 0x4F memory addresses. )

# Challenges & Solutions:

DDRAM SET ADRESS 001 xxxxxxx {rs,rw,7,6,5,4,3,2,1,0} was not performing as expected in the supplied SPARTAN3E board . Probable causes can be hardware misimplementation or mulfunction or misinterpretation of the board’s documentation.

The problem above is clear using a different command set implementation where Command 0x81 does not set the DDRAM address to 1 as expected but causes erant behavior.

Command set written reverse: 00\_00\_34\_33\_32\_31\_30\_45\_44\_43\_42\_41\_81\_20\_00\_01\_0C\_06\_28

<https://github.com/kmd178/Digital_Systems_lab4_LCD/tree/fd10a6c63f158b86f32e481b33aea53895201bdf>

The third trial was succesful using the implementation below:

Command set written reverse: (continuing)\_43\_42\_41\_20\_45\_44\_43\_42\_41\_20\_20\_00\_01\_0C\_06\_28

<https://github.com/kmd178/Digital_Systems_lab4_LCD/tree/b895f2dc0feb562ca4ec983b583abb145a07629f>