

**People's Democratic Republic of Algeria**  
**Ministry of Higher Education and Scientific Research**  
**University M'Hamed BOUGARA – Boumerdes**



**Institute of Electrical and Electronic Engineering**  
**Department of Electronics**

Final Year Project Report Presented in Partial Fulfilment of  
the Requirements of the Degree of

**‘MASTER’**

**In Electrical and Electronic Engineering**  
**Option: Computer Engineering**

Title:

**FPGA-based Arabic LCD Display  
Controller IP Design**

Presented By:

- **RAHIL Lillia**
- **AGGOUNE Khedidja**

Supervisor:

**Dr. A.KHOUAS**

Registration Number:...../2016

## **Acknowledgement**

First and foremost, we would like to thank **Almighty ALLAH** for giving us the courage and the determination, as well as guidance in conducting this project study, despite the difficulties.

Our sincere thanks and gratitude goes to our supervisor “**Dr. KHOUAS**”, for proposing and directing this work.

Next, we owe a special thanks to the **members of the jury** who have agreed to evaluate our work.

Finally; we would like to thank all those who helped us accomplish this modest work, near or far.

**RAHIL & AGGOUNE**

## Dedication

*“To the light of my world, my precious dad,  
To the apple of my eye, my loving mom,  
To my brother and sisters,  
To my special friend who has never left my side,  
To all the ones who helped me accomplish this work,  
To all friends and computer classmates  
This work is dedicated.”*

*R. Lillia*

## Dedication

*“I am deeply grateful to my family, especially my parents, brothers and sister, you are the light that shows me the way.*

*To the memory of my grandmother I dedicate this work, and to all my friends over the world.”*

*Khedidja*

# Abstract

A 2x16 character Liquid Cristal Display (LCD) is one of the most common Input/Output (I/O) devices. It allows designers to communicate with the outside world. Alphanumeric LCD can display English and some other special characters. However, Arabic characters cannot be displayed since the Arabic fonts are not embedded in the LCD controller, thus there are no corresponding American Standard Code for Information Interchange (ASCII) codes for it. To allow Arabic display on an LCD module and reduce designs' time, we suggest to create an Intellectual Property (IP) core for Arabic character LCD display controller. The design was done by creating a block diagram of the LCD controller and interface it to a clock divider module and an IP core memory namely a Block Random Access Memory (BRAM). The whole design was initialized and tested using VHDL codes. The LCD initialization is done using some specific commands. These commands allow Arabic characters to be generated and read from the Character Generator RAM (CG RAM) which is one of the controller's memory regions. The complete LCD module was described and synthesized using Xilinx Integrated Synthesis Environment (ISE) design suite tools, and implemented and tested using Xilinx Virtex 5 Field Programmable Gate Array (FPGA) and Digilent Genesys board. After creating the needed design, a simulation of each component has been made using a test bench VHDL code. However, Displaying Arabic characters using FPGA couldn't be achieved due to lack of time and some problems, mainly changing the FPGA board, wasting time in working on some other Xilinx tools and lack of the needed documentations. Moreover, once we implemented the design LCD displayed just a cursor blinking to show clearly that the generated character couldn't be displayed .It has not been well introduced in the CGRAM of the controller. What we suggest is to try to reuse those commands and test it once again since the same method works once we tested it on a microcontroller.

# Table of contents

Acknowledgement .....	i
Dedication.....	ii
Dedication.....	iii
Abstract.....	iv
List of figures .....	vii
List of tables .....	viii
Acronyms .....	ix
CHAPTER I: Introduction to FPGA .....	3
I.1. Introduction .....	3
I.2. Field Programmable Gate Array .....	3
I.2.1. Overview of general FPGA device .....	3
I.2.2. FPGA's applications .....	4
I.2.3. Important Factors for comparing FPGAs.....	4
I.2.4. FPGA Xilinx Virtex 5 LXT family .....	5
I.3. Hardware development platform and tools .....	6
I.3.1. Hardware development platform.....	6
I.3.2. Hardware development tools.....	7
a. ISE design suite .....	7
b. Design flow .....	7
I.4. IP core .....	9
I.4.1. Advantages and Disadvantages of IP cores.....	9
I.4.3. IP cores type .....	9
I.4.4. Core Generator software tool .....	9
a. Overview.....	9
b. Block Memory Generator (BMG) .....	10
I.6. Conclusion.....	12
CHAPTER II: Character LCD Display Background.....	13
II.1. Introduction.....	13
II.2. Character LCD display .....	13
II.2.1. Overview of an LCD.....	13
II.2.2. LCD pins configuration .....	14
II.2.3. LCD controller .....	15
II.2.4. LCD control and display commands.....	19
II.2.5. LCD display modes.....	22
II.3. conclusion .....	22
CHAPTER III: LCD Controller Design .....	23
III.1. Introduction .....	23

III.2. LCD controller design .....	23
III.2.1. Design of LCD module.....	23
III.2.2. Block memory LCD interfacing .....	26
a. Single port block memory .....	26
b. SDP block memory .....	28
III.2.3. Arabic Character Display .....	31
a. Microcontroller design .....	31
b. FPGA design .....	34
III.3. Conclusion .....	35
CHAPTER IV: Implementation and Experimental Results .....	36
IV.1. Introduction .....	36
IV.2. Experimental results .....	36
IV.2.1. LCD interfacing.....	36
IV.2.2. Block memory LCD interfacing.....	40
IV.2.3. FPGA Arabic characters on LCD.....	41
IV.3. Conclusion.....	42
General Conclusion .....	43
References .....	xi

## List of figures

<b>Figure I. 1:</b> FPGA structure[1].....	3
<b>Figure I. 2:</b> Virtex 5 FPGA LXT family members[8].....	6
<b>Figure I. 3:</b> Genesys board's architecture [3] .....	7
<b>Figure I. 4:</b> Xilinx FPGA design flow [8]. .....	8
<b>Figure I. 5:</b> BMG signals pinout [8].....	12
<b>Figure II. 1:</b> 2x16 LCD module [14].....	13
<b>Figure II. 2:</b> Structure of an LCD module [11]. .....	14
<b>Figure II. 3:</b> Xilinx Virtex 5 FPGA interfaced with a character LCD [3].....	15
<b>Figure II. 4:</b> 2 lines by 16 characters display [5].....	17
<b>Figure II. 5:</b> The CG ROM patterns used by the ST7066U-0A controller [5].....	17
<b>Figure II. 6:</b> Block diagram of the Sitronix ST7066U LCD controller [5]. .....	18
<b>Figure III. 1:</b> Transition diagram of LCD module initialization.....	24
<b>Figure III. 2:</b> Simulation of the clock divider used.....	24
<b>Figure III. 3:</b> Block diagram for LCD module.....	25
<b>Figure III. 4:</b> Pinout of LCD display. ....	25
<b>Figure III. 5:</b> Simulation of LCD module. ....	25
<b>Figure III. 6:</b> Example of COE file to display “Master students”. ....	27
<b>Figure III. 7:</b> Single port RAM pinout.....	27
<b>Figure III. 8:</b> Block diagram of a 16x8 single port RAM interfaced with LCD.....	28
<b>Figure III. 9:</b> The simulation of single port RAM using COE file. ....	28
<b>Figure III. 10:</b> The COE file used for SDP RAM design. ....	29
<b>Figure III. 11:</b> Simple SDP RAM pinout.....	30
<b>Figure III. 12:</b> Block diagram of a 16x8 SDP RAM interfaced with LCD.....	30
<b>Figure III. 13:</b> Simulation results for LCD display using COE file.....	31
<b>Figure III. 14:</b> Design of a PIC 16F877 interfacing with 2x16 LCD module.....	32
<b>Figure III. 15:</b> Example of 5x8 dot matrix representation for an Arabic character (•). ....	32
<b>Figure III. 16:</b> Flowchart of LCD custom character display. ....	33
<b>Figure III. 17:</b> Custom characters LCD displays. ....	33
<b>Figure III. 18:</b> Transition diagram of LCD module initialization for Arabic display.....	34
<b>Figure IV. 1:</b> Counter Output after (a) one pushbutton's press and (b) three presses.....	37
<b>Figure IV. 2:</b> Digital circuit for debouncing a push button.....	38
<b>Figure IV. 3:</b> Counter output after (a) one push button press and (b) two pressed. ....	38
<b>Figure IV. 4:</b> LCD module display using a simple VHDL. ....	40



## List of tables

<b>Table I. 1:</b> Basic factors for FPGA's comparison. ....	4
<b>Table I. 2:</b> Memory types included in BMG. ....	10
<b>Table II. 1:</b> Corresponding LCD pins on Genesys board [3]. ....	15
<b>Table II. 2:</b> IR and DR Registers operation. ....	16
<b>Table II. 3:</b> Example of custom character at the first location in CG RAM. ....	19
<b>Table II. 4:</b> Instructions and codes of some LCD commands. ....	21
<b>Table III. 1:</b> The Content of the CG RAM for an Arabic character display. ....	35
<b>Table IV. 1:</b> Pins configuration of the LCD display. ....	36
<b>Table IV. 2:</b> Pins configuration for the 16 bit up counter. ....	37
<b>Table IV. 3:</b> ASCII code of the characters to be used. ....	39
<b>Table IV. 4:</b> BRAMs LCD interfacing implementation. ....	40

# Acronyms

<b>μC</b>	<b>Microcontroller</b>
<b>A/D</b>	<b>Analog to Digital converter</b>
<b>AC</b>	<b>Address Counter</b>
<b>ASCII</b>	<b>American Standard Code for Information Interchange</b>
<b>ASIC</b>	<b>Application Specific Integrated Circuit</b>
<b>B</b>	<b>Blink</b>
<b>BF</b>	<b>Busy Flag</b>
<b>BMG</b>	<b>Block Memory Generator</b>
<b>BRAM</b>	<b>Block Random Access Memory</b>
<b>C</b>	<b>Cursor</b>
<b>CAD</b>	<b>Computer Aided Design</b>
<b>CG RAM</b>	<b>Character Generator Random Access Memory</b>
<b>CG ROM</b>	<b>Character Generator Read Only Memory</b>
<b>CLB</b>	<b>Configurable Logic Block</b>
<b>CMT</b>	<b>Clock Management Tile</b>
<b>COE</b>	<b>Coefficient</b>
<b>CPLD</b>	<b>Complex Programmable Logic Device</b>
<b>D</b>	<b>Display</b>
<b>DB</b>	<b>Data Bus</b>
<b>DCM</b>	<b>Digital Clock Manager</b>
<b>DD RAM</b>	<b>Display Data Random Access Memory</b>
<b>DL</b>	<b>Data Length</b>
<b>DLL</b>	<b>Delay Locked Loop</b>
<b>DP RAM</b>	<b>Dual Port Random Access Memory</b>
<b>DR</b>	<b>Data Register</b>
<b>DSP</b>	<b>Digital Signal Processing</b>
<b>E</b>	<b>Enable</b>
<b>EDA</b>	<b>Electronic Design Automation</b>
<b>EDK</b>	<b>Embedded Development Kit</b>
<b>FF</b>	<b>Flip Flop</b>
<b>FIFO</b>	<b>First Input First Output</b>
<b>FPGA</b>	<b>Field Programmable Gate Array</b>
<b>FSM</b>	<b>Finite State Machine</b>

<b>GPIO</b>	<b>General Purpose Input Output</b>
<b>HDL</b>	<b>Hardware Description Language</b>
<b>I/O</b>	<b>Input/Output</b>
<b>IC</b>	<b>Integrated Circuit</b>
<b>IOB</b>	<b>Input Output Block</b>
<b>IOE</b>	<b>Input Output Element</b>
<b>IP</b>	<b>Intellectual Property</b>
<b>IR</b>	<b>Instruction Register</b>
<b>ISE</b>	<b>Integrated Synthesis Environment</b>
<b>ISim</b>	<b>Integrated Synthesis Environment Simulator</b>
<b>LAB</b>	<b>Integrated Synthesis Environment</b>
<b>LC</b>	<b>Liquid Crystal</b>
<b>LCD</b>	<b>Liquid Cristal Display</b>
<b>LED</b>	<b>Light-Emitting Diode</b>
<b>LUT</b>	<b>Look-Up Tables</b>
<b>MIF</b>	<b>Memory Initialization File</b>
<b>MSB</b>	<b>Most Significant Bit</b>
<b>PC</b>	<b>Personal Computer</b>
<b>PIC</b>	<b>Programmable Interface Controller</b>
<b>PLL</b>	<b>Phase Locked Loop</b>
<b>RAM</b>	<b>Random Access Memory</b>
<b>ROM</b>	<b>Read Only Memory</b>
<b>RS</b>	<b>Register Select</b>
<b>RTL</b>	<b>Register Transfer Level</b>
<b>RW</b>	<b>Read/Write</b>
<b>SDP RAM</b>	<b>Simple Dual Port Random Access Memory</b>
<b>SIMPRIM</b>	<b>Simulation Primitive</b>
<b>SoC</b>	<b>System on Chip</b>
<b>UCF</b>	<b>User Constraints File</b>
<b>VHDL</b>	<b>VHSIC Hardware Description Language</b>
<b>VHSIC</b>	<b>Very High-Speed Integrated Circuit</b>
<b>XST</b>	<b>Xilinx Synthesis Technology</b>

# **General Introduction**

## General Introduction

In the world of digital design, a major revolution has taken place over the past period. Field programmable gate arrays (FPGAs) can contain over a million equivalent logic gates and tens of thousands of flip flops (FFs). This means that it is not possible to use traditional methods of logic design involving the drawing of logic diagrams when the digital circuit may contain thousands of gates, traditional integrated circuits design has been replaced by the design flow based on hardware description languages (HDL) such as VHDL, Verilog, and tools for logic synthesis. FPGAs can be reprogrammed to desired applications or functionality requirements after manufacturing. It is integrated inside a specific board to achieve the required system's objectives. In fact, Input/output (I/O) devices are very useful in such system such as Liquid Crystal Device (LCD). The character LCD display is the most common device which is attached to Genesys Virtex 5 FPGA board, that LCD can display only English/Japan characters ASCII codes. Due to the systems development Arabic displaying is needed, however, ASCII codes do not include any Arabic characters.

HDL is used to describe complex logic function, these are included in design suites such as Xilinx's Integrated Synthesis Environment (ISE) and similar tools. HDL based design flow offers portability, reduction of the design cycle, independence of technology, and automatic synthesis and logic optimization. However, if design engineers had to code commonly used complex digital circuits in large projects, they would end up wasting more time and money. Because of this, a new generation of methodology, intellectual property (IP) core is provided. An IP core is a block of logic or data to perform a specific function that is used in making an FPGA or an application specific integrated circuit (ASIC). IP cores can be used in a complex design where a designer wants to save time. The CORE Generator System is suggested as an ISE design tool that delivers parameterized cores optimized for Xilinx FPGAs. These include Digital Signal Processing (DSP) function, memories, storage elements, math functions and other basic elements. However, the core library of ISE tool does not contain a core for character LCD controller.

In our project, mainly we have introduced some designs based on Genesys' 2x16 character LCD display by using a single port and simple dual port RAM including in the block memory generator (BMG) which is an existed IP and make comparison with them, in

order to allow a later IP design for that LCD. In addition, we displayed Arabic characters on LCD display using microcontroller and FPGA.

The main objectives of our project are to display Arabic characters on LCD display in the Genesys board and suggest a design of IP core for FPGA-based LCD controller to integrate it with ISE design suite tool.

Our report is divided into four chapters. Chapter one describes the FPGA's technology. Chapter two presents a 2x16 character LCD display. Chapter three explains the design of LCD controller integrating Arabic characters. Chapter four presents implementation and experimental results. Finally in the general conclusion, a summary of the work is presented, with suggestions for future work.

# **CHAPTER I**

## **Introduction to FPGA**

# CHAPTER I: Introduction to FPGA

## I.1. Introduction

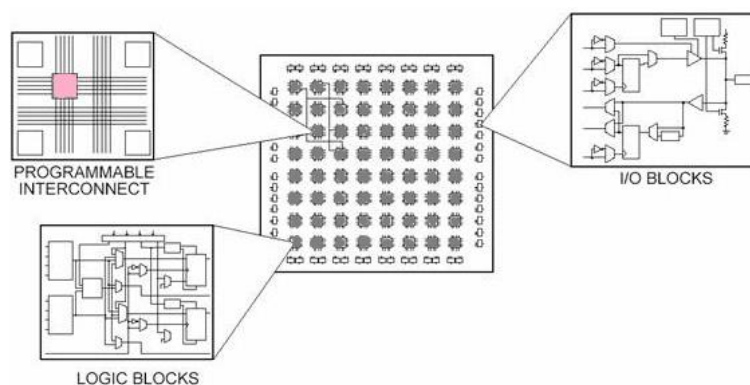
In today's digital world, FPGA has become one of the major technologies used to create sophisticated digital systems based on schematic designs or HDL languages; mainly VHDL or Verilog languages. Actually, custom hardware functionality can be implemented using FPGA without need to use a breadboard or soldering iron. Xilinx Virtex 5 FPGA has been used in our project. In this chapter, some FPGA features will be introduced.

## I.2. Field Programmable Gate Array

In the following sections, FPGA is going to be described with details.

### I.2.1. Overview of general FPGA device

FPGA is a two dimensional logic array that can be programmed and reconfigured by the user in the field after manufacturing, hence field programmable. It allows the implementation of complex digital circuits. This device has been around since 1980s. This semiconductor chip is made up of different logic blocks named Configurable Logic Blocks (CLBs) by Xilinx or Logic Array Blocks (LABs) by Altera. Each one of these logic blocks; seen in gray in **Figure I.1**; is surrounded by reconfigurable interconnections (the spacing between gray blocks) to allow the routing of all these blocks together. These blocks are surrounded by a configurable General Purpose Input/Output pins (GPIO). These pins are named as Input/Output Blocks (IOBs) by Xilinx whereas Altera suggests Input/Output Elements (IOEs) as a name.



**Figure I. 1:** FPGA structure[1].



### I.2.2. FPGA's applications

FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from ASICs, which are custom manufactured for specific design tasks. They can cover an extremely wide range of applications including DSP, speech recognition, defense system, face detection, ASIC prototyping, medical imaging, industrial applications...etc.

### I.2.3. Important Factors for comparing FPGAs

FPGAs are similar to Complex Programmable Logic Devices (CPLDs). The only difference is that CPLDs are much smaller in size and compatibility than FPGAs. They are manufactured by many different companies like Xilinx, Altera, Actel, Lattice, QuickLogic and Atmel. These companies use some specific factors to distinguish between FPGAs. These factors are described in **Table I.1**.

**Table I. 1:** Basic factors for FPGA's comparison.

Factor	Description
Fabrication process	It is clear that more advanced fabrication process brings higher integration, and thus higher density and/or reduced size of chips.
Logic density	The unit used to express the logic capability of an FPGA.
Clock management	Clock management comprises two basic functions: removing clock skew and propagation delay and generating new clock signals with different frequencies. Generally, this can be done using either Delay Locked Loops (DLLs), or Phase Locked Loops (PLLs).
On-chip memory	As FPGA applications grow in complexity so does their need for memory. Using Look-Up Tables (LUTs) as registers for storing data couldn't possibly provide enough space for serious applications. Especially if these applications require numerous arithmetical computations to be performed, and are time dependent. As this is often the case, the outside memory could not produce desired efficiency. This is why, with every new generation of FPGAs, more and more memory gets embedded into FPGA. [8]
DSP capabilities	The majority of FPGA applications require some sort of DSP in order to reduce the time and increase efficiency of computations.

I/O compatibility	As FPGAs continue to grow in size and capacity, the larger and more complex systems designed for them demand an increased variety of Input/Output standards. Furthermore, as system-clock speeds continue to increase, the need for high-performance I/O becomes more important.
Software support and other design services	Developing an FPGA-based hardware system is a complex process. Designers divide this process into different stages (Design flow), providing a complete software solution for each of them.

In fact, Xilinx and Altera are the current market leaders and long-time industry rivals. Both of these two companies provide a proprietary Windows and Linux design software mainly ISE design suite or Vivado as Xilinx tools and Quartus for Altera to allow the user to design, analyze, simulate and compile his design in an easy way.

Since FPGA Virtex 5 is the one used in our project, it will be discussed in the following section.

#### **I.2.4. FPGA Xilinx Virtex 5 LXT family**

This FPGA family offers an advanced platform that meets the growing need for programmable systems with higher performance, higher density, lower power consumption, and lower overall system cost. It is available in -3, -2, -1 speed grades, with -3 having the highest performance.

**Figure I.2** shows a summary of the available resources for different Virtex 5 LXT FPGAs family. It incorporates LUTs with six independent inputs, with a new diagonal interconnect structure. In addition, a PLL have been added to each clock management tile (CMT), which now contains two digital clock managers (DCMs) and one PLL. The CMT thus offers the best of both worlds: the robust versatility and precise incremental phase shift capability of a digital clock manager combined with the jitter reduction from the analog PLL. The largest device in the family has six CMTs capable of generating and manipulating 550-MHz clocks, supporting the performance of Virtex-5 logic and block functions. Moreover, synchronous dual-ported block Random Access Memory (RAM) is an important block memory. The size of each block RAM has been increased to 36 Kb, but you can also use it as two independent 18-Kb block RAMs. Furthermore, the DSP48Eslice contains a 25x18 multiplier, an adder, and an accumulator, thus it is dedicated to arithmetic operations[8] .

Device	Configurable Logic Blocks (CLBs)			DSP48E Slices	Block RAM Blocks			CMTs	PowerPC Processor Blocks	Endpoint Blocks for PCI Express	Ethernet MACs	Max RocketIO Transceivers		Total I/O Banks	Max User I/O
	Array (Row x Col)	Virtex-5 Slices	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)					GTP	GTX		
XC5VLX20T	60 x 26	3,120	210	24	52	26	936	1	N/A	1	2	4	N/A	7	172
XC5VLX30T	80 x 30	4,800	320	32	72	36	1,296	2	N/A	1	4	8	N/A	12	360
XC5VLX50T	120 x 30	7,200	480	48	120	60	2,160	6	N/A	1	4	12	N/A	15	480
XC5VLX85T	120 x 54	12,960	840	48	216	108	3,888	6	N/A	1	4	12	N/A	15	480
XC5VLX110T	160 x 54	17,280	1,120	64	296	148	5,328	6	N/A	1	4	16	N/A	20	680
XC5VLX155T	160 x 76	24,320	1,640	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX220T	160 x 108	34,560	2,280	128	424	212	7,632	6	N/A	1	4	16	N/A	20	680
XC5VLX330T	240 x 108	51,840	3,420	192	648	324	11,664	6	N/A	1	4	24	N/A	27	960

**Figure I. 2:** Virtex 5 FPGA LXT family members[8].

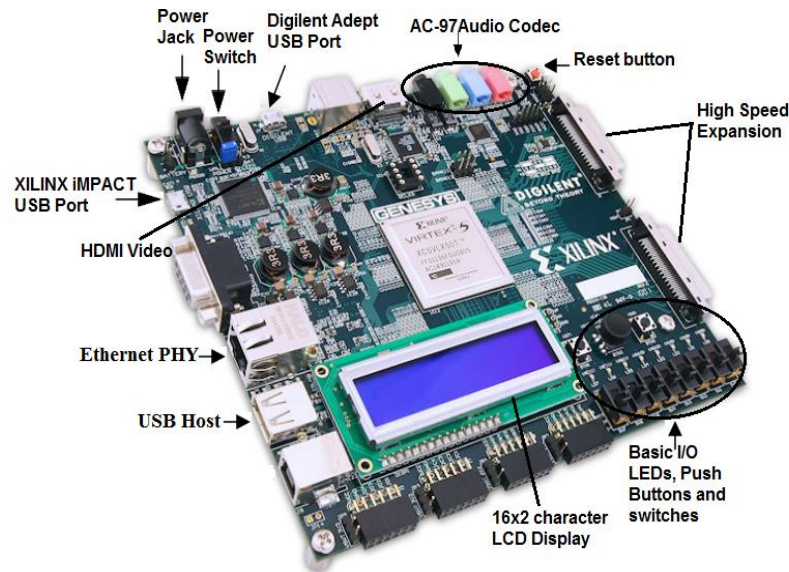
After dealing with all these information about FPGA technology, it is recommended to describe briefly the hardware development platform and all the needed tools used to for digital system design.

### I.3. Hardware development platform and tools

In our project Xilinx Virtex 5 FPGA integrated with the Genesys board have been used. This FPGA development board will be presented in the following sections.

#### I.3.1. Hardware development platform

The Digilent's Genesys board seen in **Figure I.3** brings the power of the Xilinx VIRTEX 5 FPGA to a surprisingly uncomplicated design platform. With gigabit ethernet, high speed memory, high resolution audio and video circuits, and a host of USB connectivity options, the Genesys board includes proven circuits used in the most demanding designs. From complex systems to dedicated high performance applications, the Genesys board brings workable solutions to complex problems. Genesys is fully compatible with all Xilinx Computer Aided Design (CAD) tools, including ChipScope, Embedded Development Kit (EDK), and the free WebPack, so designs can be completed with no extra costs. The Virtex5 LX50T is optimized for high performance logic. It includes Digilent's newest Adept USB2 system that offers device programming, real time power supply monitoring, automated board tests, virtual I/O, and simplified user data transfer facilities. A second USB programming port (iMPACT USB2), based on the Xilinx programming cable, is also built into the board [3].



**Figure I. 3:** Genesys board's architecture [3].

Genesys Virtex 5 includes different inputs and outputs that can be used to perform specific tasks. The major ones are pushbuttons, eight switches, and eight LEDs for basic digital input and output. It also contains a standard 2x16 character LCD which is of our interest to deal with it later on.

The upcoming sections will discuss the tool used to program the FPGA device.

### **I.3.2. Hardware development tools**

Xilinx offers different tools to interact with the Xilinx FPGA devices. These tools are described below.

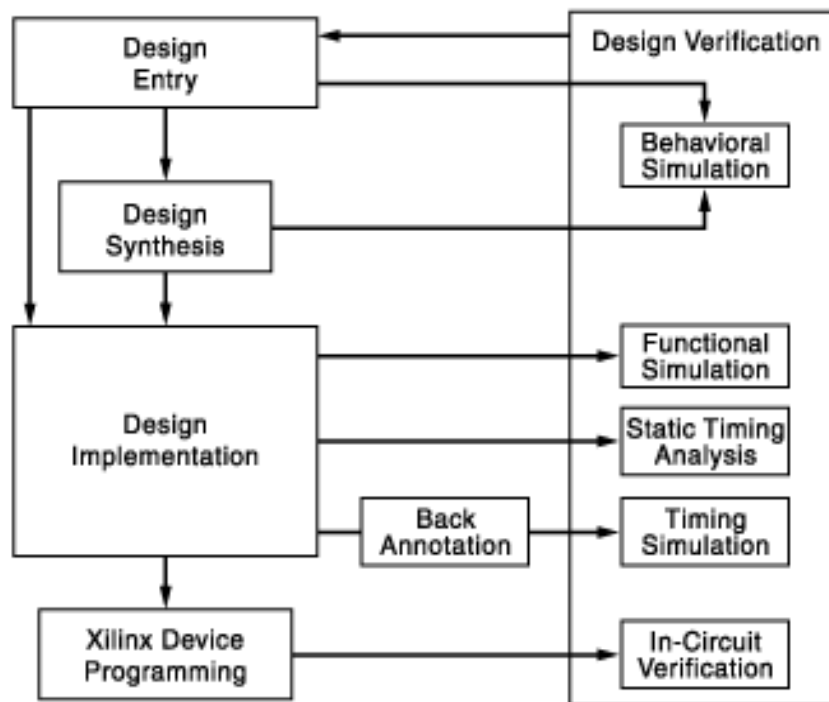
#### **a. ISE design suite**

Some tools are needed to design digital systems so that they perform the required tasks. In our case, it is necessary to use Xilinx ISE design suite as a tool to program the Virtex 5 integrated in the Genesys board, so that it converts the HDL or Schematic design into bits stream (machine language). The HDL used in this project is VHDL. Before creating any VHDL module it is really important to understand the design flow described in the next section.

#### **b. Design flow**

Designing a digital system in Xilinx ISE tools goes through different steps before the designer himself can test the functionality of that system and download it in the FPGA (Virtex 5 in our case). The ISE design flow seen in **Figure I.4** comprises the following steps: design entry, design synthesis, design implementation, and Xilinx device programming. Design verification, which includes both functional verification and timing verification, takes places at different points during the design flow[8]. The main steps are described below:

- **Design Entry:** is required to create design files using hardware description language (In our case VHDL) or schematic editor.
- **Functional verification:** is used to test functionality of the design at different points in the design flow as follows:
  - ✓ Before synthesis, run behavioral simulation (also known as Register Transfer Level (RTL) simulation).
  - ✓ After Translate, run functional simulation (also known as gate-level simulation), using the SIMPRIM library.
  - ✓ After device programming, run in-circuit verification[8].
- **Design Synthesis :** During synthesis, the synthesis engine compiles the design to transform HDL sources into an architecture specific design netlist. The ISE software supports the use of Xilinx Synthesis Technology (XST), which is delivered within it[8].
- **Design Implementation :** Partition, place, and route to create bit stream file, called bit file.
- **Timing verification:** is performed during after Map and place & route implementation.
- **Xilinx Device Programming :** After generating a programming file, it is possible to configure the used device by generating configuration files and downloading the programming files from a host computer to a Xilinx device using iMPACT Xilinx tool.



**Figure I. 4:** Xilinx FPGA design flow [8].

The following section explains the IP core in details. In fact, existing IP cores have been used in our designs.

## **I.4. IP core**

IP core is a ready-made function that can be inserted in any digital design as a block diagram with user's specifications. In other words, it is a block of logic or data that is used in making a FPGA or ASIC for a product. As essential elements of design reuse.

### **I.4.1. Advantages and Disadvantages of IP cores**

Like any new technologies and tools, IP cores have their advantages and disadvantages. Although they may simplify a given design, the engineer has to design the interfaces to send and receive data from this black box. Moreover, many cores are designed for particular parts but some come free but other cores may cost you thousands of dollars.

### **I.4.3. IP cores type**

IP cores split up into three different categories: soft cores, firm cores and hard ones. The difference between the three cores is that the soft core is the one which exists either as a netlist (a list of the logic gates and associated interconnections making up an integrated circuit (IC)) or HDL code, whereas the hard core is a physical manifestations of the IP design. Hard core is the best for plug and play applications, and is less portable and flexible than the other two types of cores. Like the hard core, firm or simply semi-hard core also carry placement data but is configurable to various applications. As a way to create some IP cores it is suggested to use the CORE Generator System as an ISE design tool that delivers parameterized cores optimized for Xilinx FPGAs.

### **I.4.4. Core Generator software tool**

Some existing IP cores can be generated using a special Xilinx tools described in details in the next coming sections.

#### **a. Overview**

The CORE Generator software is based on the use of the IP catalog. This IP catalog includes cores that are already defined to perform some specific operations in order to replace many coding lines. One of these IP cores is the Memory and storage elements used to create different block memories.

A memory is a storage element used to hold data and programs in a binary format. It can be found in every electronic device including FPGAs with two main different types:

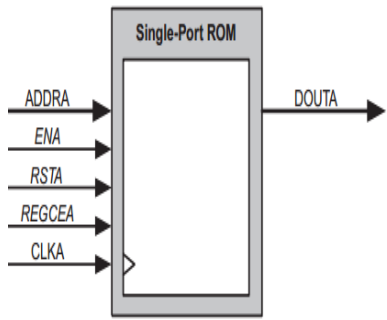
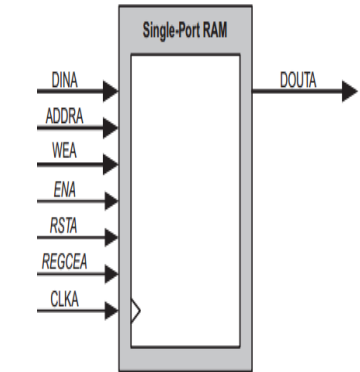
RAM and ROM. RAM is a volatile memory that can be access randomly at any time in any order. Its contents are lost when the power is turned off whereas ROM is a non-volatile memory used to read from it only.

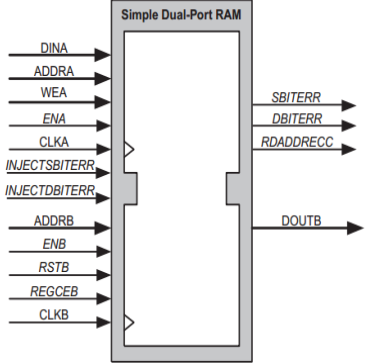
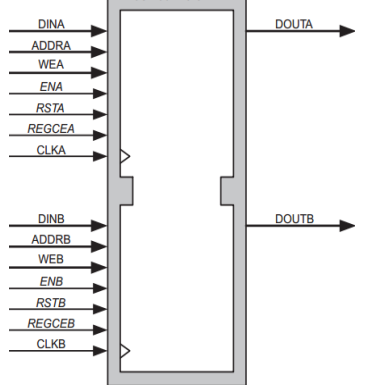
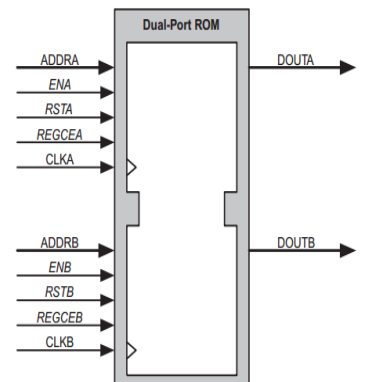
### b. Block Memory Generator (BMG)

Memories and storage Elements IP core; included in the Core Generator; contains three different cores FIFOs, Memory Interface Generators, RAMs and ROMs which contains two memory Generators, Block and Distributed. What is interesting for us is the BMG.

The Xilinx LogiCORE IP BMG core, found in the Core Generator is an advanced memory constructor that generates area and performance-optimized memories using embedded block RAM resources in Xilinx FPGAs [1]. Users can quickly create optimized memories to leverage the performance and features of block RAMs and ROMs in Xilinx FPGAs. Actually, BMG provides different types of IP core memories that are described briefly below. Mainly Single port RAM, SDP RAM, True Dual port RAM, Single port ROM, and Dual port ROM. s These memories can be explained in the **Table I.2** seen below.

**Table I. 2:** Memory types included in BMG.

Memory type	Block diagram	Description
Single port ROM		In single port ROM, only one address port is available for read operation.
Single port RAM		On a single port RAM, the read and write operations share the same address at port A, and the data is read from output port A.

SDP RAM		<p>In SDP RAM mode, a dedicated address port is available for each read and write operation (one read port and one write port). A write operation uses write address from port A while read operation uses read address and output from port B.</p>
True Dual port RAM		<p>In true dual port RAM mode, two address ports are available for read or write operation (two read/write ports). In this mode, we can write to or read from the address of port A or port B, and the data read is shown at the output port with respect to the read address port.</p>
Dual port ROM		<p>The dual port ROM has almost similar functional ports as single port ROM. The difference is dual port ROM has an additional address port for read operation.</p>

**Figure I.5** describes the BMG signals. The widths of the data ports (dina, douta, dinb, and doutb) can be selected in logiCORE tool. The address port (addra and addrb) widths are determined by the memory depth with respect to each port. The Write enable ports (wea and web) are buses of width 1 when byte-writes are disabled. When byte-writes are enabled, wea and web widths depend on the byte size and Write data widths selected.



Name	Direction	Description
clka	Input	<b>Port A Clock:</b> Port A operations are synchronous to this clock. For synchronous operation, this must be driven by the same signal as CLKB.
addra	Input	<b>Port A Address:</b> Addresses the memory space for port A Read and Write operations. Available in all configurations.
dina	Input	<b>Port A Data Input:</b> Data input to be written into the memory through port A. Available in all RAM configurations.
douta	Output	<b>Port A Data Output:</b> Data output from Read operations through port A. Available in all configurations except Simple Dual-port RAM.
ena	Input	<b>Port A Clock Enable:</b> Enables Read, Write, and reset operations through port A. Optional in all configurations.
wea	Input	<b>Port A Write Enable:</b> Enables Write operations through port A. Available in all RAM configurations.
rsta	Input	<b>Port A Set/Reset:</b> Resets the Port A memory output latch or output register. Optional in all configurations.
regcea	Input	<b>Port A Register Enable:</b> Enables the last output register of port A. Optional in all configurations with port A output registers.
clkb	Input	<b>Port B Clock:</b> Port B operations are synchronous to this clock. Available in dual-port configurations. For synchronous operation, this must be driven by the same signal as CLKA.
addrb	Input	<b>Port B address:</b> Addresses the memory space for port B Read and Write operations. Available in dual-port configurations.
dinb	Input	<b>Port B Data Input:</b> Data input to be written into the memory through port B. Available in True Dual-port RAM configurations.
doutb	Output	<b>Port B Data Output:</b> Data output from Read operations through Port B. Available in dual-port configurations.
enb	Input	<b>Port B Clock Enable:</b> Enables Read, Write, and reset operations through Port B. Optional in dual-port configurations.
web	Input	<b>Port B Write Enable:</b> Enables Write operations through Port B. Available in Dual-port RAM configurations.

Figure I. 5: BMG signals pinout [8].

## I.6. Conclusion

In this chapter we presented some backgrounds and features of FPGAs. We examine Genesys board and gave some needed information about needed tools to be used later on. Also, we presented the general architecture of the FPGA we've used. Moreover, we tried to give brief ideas about the IP cores and the core generator. This tool will be used in the next coming chapters in order to create some needed IP cores in digital design world.

## **CHAPTER II**

### **Character LCD Display Background**

## CHAPTER II: Character LCD Display Background

### II.1. Introduction

Most digital systems combine hardware design and software design to achieve the required system's objectives. Actually, each system contains different input and output ports to allow the designer to control and communicate with the outside world. One of these I/O devices is the character LCD display. This chapter introduces the needed information for this module which is attached to Genesys Virtex 5 FPGA board; that is the LCD display.

### II.2. Character LCD display

Genesys board has a 2x16 character LCD display connected to it to allow the designer to display some data and integrate that module in the design of the digital system as required. **Figure II.1** shows a 2x16 character LCD display with its corresponding pins. These pins will be explained later on.

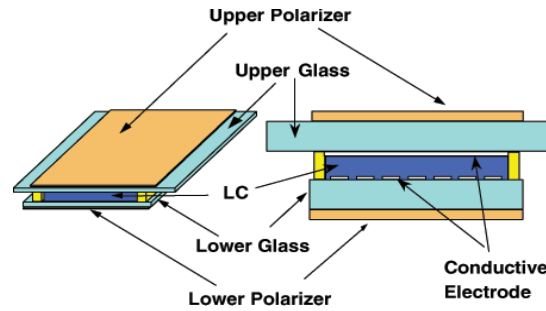


**Figure II. 1:** 2x16 LCD module [14].

For the coming papers, we will describe LCD's features.

#### II.2.1. Overview of an LCD

An LCD is a very basic electronic module; commonly used in various devices and circuits. This module is preferred over seven segments display and Light-Emitting Diodes (LEDs). The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special and even custom character (unlike seven segments), animations and so on[4]. This I/O device consists of a panel of liquid crystal molecules (LC) illustrated in **Figure II.2** that can be induced by electrical fields to take certain patterns which block light or allow it through, hence Liquid Cristal Display. Liquid crystals do not emit light directly.



**Figure II. 2:** Structure of an LCD module [11].

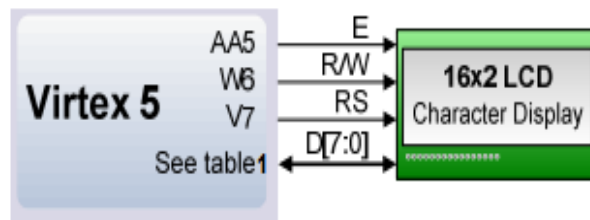
LCDs are used all over the world in calculators, digital clocks, CD players...etc. In addition, character LCDs are ideal for displaying text thus alphanumeric LCDs. They can also be configured to display small icons with no more than 5x7 pixels in size.

There exist different types of LCDs display mainly the character LCDs display and the graphical one. Obviously, it is enough to describe the character LCDs display since they are the needed one in this project. Sometimes it is called alphanumeric LCDs. They are inexpensive and easy to use. They range in different sizes including 1x8, 2x8, 1x16, 2x16 characters, 4x16 characters plus many more. What is interesting for us now is the 2x16 character LCD displays, that is, two lines of sixteen characters each.

In fact, it is necessary to understand how a character LCD is configured. The next section deals with pins configuration.

### II.2.2. LCD pins configuration

Genesys board contains a standard 2x16 character LCD, that is, two lines of sixteen characters each connected as seen in **Figure II.3**. Each character is represented in 5x8 pixel dotted matrix (5x7 pixels for character one extra for the cursor). This LCD has sixteen different pins that are connected directly to specific pins on the Xilinx Virtex 5 FPGA. Two of these pins are not used due to the fact that they are used for optional backlight. The rest of pins includes eight data signals D7...D0, three control signals (read and write signal (R/W), Enable signal (E) and Register Select signal (RS)), and three voltage supply signals. E line is used to allow the LCD to send or receive data. RS line is used for data and instructions, when it is low the data is to be treated as a command or special instruction (such as clear screen, position cursor...etc.), but when it is high, the data being sent is text data which should be displayed on the screen. R/W line is used to control operation, when it is low, the information on the data bus is being written to the LCD, and when it is high the program is effectively reading the LCD.



**Figure II. 3:** Xilinx Virtex 5 FPGA interfaced with a character LCD [3].

The Xilinx Virtex 5 FPGA pins assignment is illustrated in **Table II.1**.

**Table II. 1:** Corresponding LCD pins on Genesys board [3].

LCD pins	Signal	FPGA pins	Description
1	Vss		Ground
2	Vdd		5V Power Supply
3	Vo		Contrast Voltage (typically 100mV-200mV at 20C)
4	RS	V7	Register select: high for data, low for instructions
5	R/W	W6	Read/write signal: high for read, low for write
6	E	AA5	Read/write: high for OE; falling edge writes data
7	D0	Y8	Bidirectional data bus 0
8	D1	AB7	Bidirectional data bus 1
9	D2	AB5	Bidirectional data bus 2
10	D3	AC4	Bidirectional data bus 3
11	D4	AB6	Bidirectional data bus 4
12	D5	AC5	Bidirectional data bus 5
13	D6	AC7	Bidirectional data bus 6
14	D7	AD7	Bidirectional data bus 7

### II.2.3. LCD controller

The 2x16 character LCD of Genesys board uses a Sitronix ST7066U as a controller module. It is used to communicate with the FPGA and control the LCD display to perform the required task. Its block diagram seen in **Figure II.6** has two 8 bit registers Instruction Register (IR) and Data Register (DR) to store information sent from FPGA. The operation and selection of the two registers are defined by **Table II.2**. The IR register stores instruction codes, such as display clear, cursor shift, and address information for display data RAM (DD RAM) and character generator RAM (CG RAM). The IR can only be written from FPGA. The DR register temporarily stores data to be written into DD RAM or CG RAM and temporarily stores data

read from DD RAM or CG RAM. These memories are going to be explained in the next coming sections. The Busy Flag (BF) seen in that table gives an indication whether the LCD has finished the previous instruction and ready with the next.

**Table II. 2:** IR and DR Registers operation.

RS	R/W	Operation
0	0	Instruction Write operation (IR writes an internal operation like clear display).
0	1	Read Busy Flag (DB7) and address counter (DB0 ... DB6).
1	0	Data Write operation to DD RAM or CG RAM (DR to CG RAM or DD RAM).
1	1	Data Read operation from DD RAM or CG RAM to DR.

It also includes an address counter (AC) which receives an initial address through IR based on a command/instruction code, assigns and updates addresses to both DDRAM and CGRAM. The DDRAM is used to store the display data represented in 8 bit character codes and sent from FPGA. Each address of DDRAM corresponds to a position on the LCD [5].

As it has been mentioned earlier, the LCD Controller has three internal memory regions, which are described in this section. A character generator ROM (CG ROM) with 208 preset 5x8 character patterns, these preset patterns are identified by their American Standard Code for Information Interchange (ASCII) codes (up through 7F are standard ASCII which includes all normal alphanumeric characters ). In fact, the Sitronix ST7066U uses English / Japan ASCII table (ST7066U-0A). The leftmost column on **Figure II.5** represents CG RAM. CG RAM custom character pattern is programmed line by line.

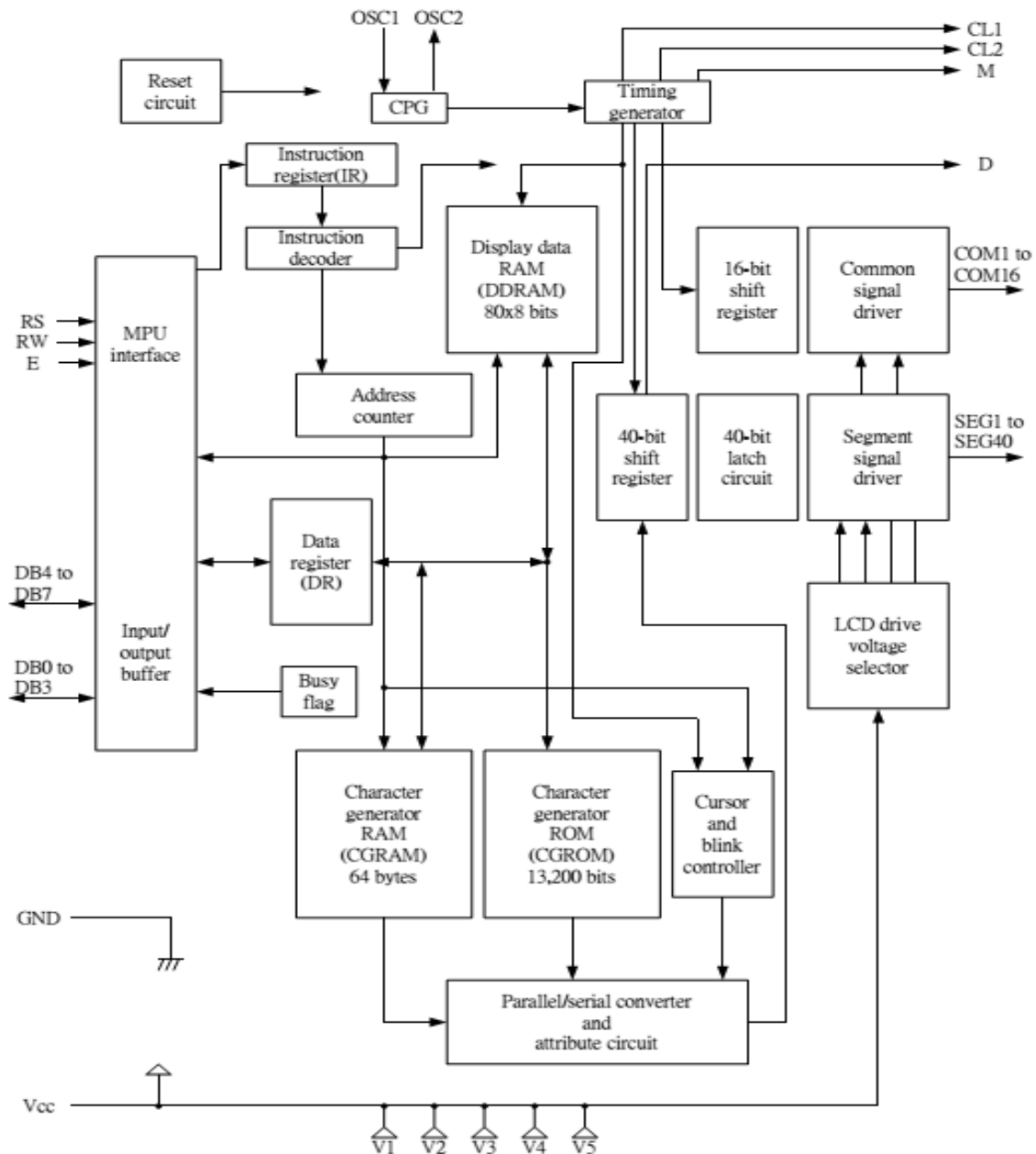
Also has two RAMs, naming DD RAM and CG RAM. CG RAM allows user to define their eight custom 5x8 characters. The DD RAM can hold up to 80 character codes at a time. Each byte of DD RAM represents each unique position on the LCD display. The LCD controller reads the information from the DD RAM and displays it on the LCD screen by mapping the locations 00H to 0FH to the first display row, and locations 40H to 4FH map to the second row seen in **Figure II.4**. Normally, DD RAM location 00H maps to the upper left display corner, and 40H to the lower left. Shifting the display left or right can change this mapping. The display uses a temporary data register (DR) to hold data during DD RAM /CG RAM reads or writes, and an internal address register to select the RAM location. Address register contents, set via the IR, are automatically incremented after each read or write operation. RAM read/write requests will be directed to DD RAM or CG RAM, depending on which address register was most recently accessed [3].

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

Figure II. 4: 2 lines by 16 characters display [5].

Upper 4 bit Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	(2)															
LLHL	(3)															
LLHH	(4)															
LHLL	(5)															
LHLH	(6)															
LHHL	(7)															
LHHH	(8)															
HLLL	(1)															
HLLH	(2)															
HLHL	(3)															
HLHH	(4)															
HHLL	(5)															
HHLH	(6)															
HHHL	(7)															
HHHH	(8)															

Figure II. 5: The CG ROM patterns used by the ST7066U-0A controller [5].



**Figure II. 6:** Block diagram of the Sitronix ST7066U LCD controller [5].

Moving beyond just displaying text, one awesome feature of this LCD display is the possibility to build our own characters in flexible way thus user's defined characters. However, the number of characters to be added are limited. This can be possible done by using the CG RAM. In fact, CG RAM provides space for eight custom characters only. Any character can be made to appear on a 5x8 pixel matrix element without knowledge of its ASCII value. Below in **Table II.3** it can be seen an example of an Arabic character (ﻻ) drawn inside a box of a 5x8 pixel grid. The 1's represent a green pixel and the 0's represent a white pixel. Since the data bus is 8 bits wide, the upper 3 bits are not used, and should always be set to zero (Don't care case x). This is used to encode the desired character to be defined in binary.



**Table II. 3:** Example of custom character at the first location in CG RAM.

						Upper Nibble				Lower Nibble			
						Write Data to CG RAM							
A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
Character Address			Row Address			Don't care			Character Bitmap				
0	0	1	0	0	0	x	x	x	0	0	0	0	0
0	0	1	0	0	1	x	x	x	0	0	0	0	0
0	0	1	0	1	0	x	x	x	1	0	0	0	1
0	0	1	0	1	1	x	x	x	1	0	0	0	1
0	0	1	1	0	0	x	x	x	1	1	1	1	1
0	0	1	1	0	1	x	x	x	0	0	0	0	0
0	0	1	1	1	0	x	x	x	0	0	1	0	0
0	0	1	1	1	1	x	x	x	0	0	0	0	0

Note: x stands for don't care case.

The steps to define new custom characters in CG RAM are listed below:

- Set CG RAM address: sets initial CG RAM Address. This command loads AC with the value specified in the address field and causes subsequent data to be stored in the character generator RAM.
- Write to CG RAM: write binary 8 bit data into the CG RAM memory.
- Read from CG RAM: this command is used to read the written data and to display it into the LCD display.

These three steps uses some LCD commands which are explained in the following section.

#### II.2.4. LCD control and display commands

The LCD controller uses some commands to drive the LCD. These commands are in an 8 bit format and are delivered on the same Data Bus (DB). **Table II.4** resumes the main features of LCD instructions and codes. These instructions are explained below:

- Clear Display:** This instruction is used to clear all display and returns the cursor to the home position. It writes space code (20h) into all DD RAM addresses and set the address Counter to DD RAM location address 0. In other words, the display disappears and the cursor goes to the left edge of the display (the first line if 2 lines are displayed).
- Cursor Home:** returns the cursor to the home position (Address 0).

3. **Display ON/OFF Control:** Controls display of characters and cursor. Whenever Display line (D) is ON the LCD display is turned ON and OFF when  $D = 0$ . The DD RAM contents remain unchanged. The Cursor line (C) is displayed when  $C = 1$  and is not displayed when  $C = 0$ . The cursor is displayed as 5 dots in the 8th line when the 5 x 7 dot character font is selected and as 5 dots in the 11th line when the 5 x 10 dot character font is selected. The character at the cursor position blinks when the Blink (B) is 1.
4. **Entry Mode:** This instruction sets the effect of subsequent DD RAM read or write operations. It specifies whether to increment ( $I/D = 1$ ) or decrement ( $I/D = 0$ ) the AC after subsequent DD RAM read or write operations. If  $S = 1$  the display will be shifted to the left (if  $I/D = 1$ ) or right (if  $I/D = 0$ ) on subsequent DD RAM write operations. This makes it look as if the cursor stands still and the display moves when each character is written to the DD RAM. If  $S = 0$  the display will not shift on subsequent DD RAM write operations.
5. **Cursor/Display Shift:** moves the cursor and shifts the display without changing DD RAM contents.
6. **Function set:** sets interface data length (DL), number of display lines (N) and character font (F). This command should be issued only after automatic power on initialization has occurred, or as part of the module initialization sequence. When the 4 bit length is selected, data must be sent or received in pairs of 4 bits each. The most-significant 4 bits are sent or received first.
7. **Set CG RAM Address:** sets the CG RAM address. Subsequent read or write operations refer to the CG RAM.
8. **Read Busy Flag And Address Counter:** reads the state of the busy flag (BF) and the contents of the address counter.  $BF = 1$  indicates that the module is busy processing the previous command. Whereas  $BF = 0$  indicates that the module is ready to perform another command. The value of the address counter is also returned. The same address counter is used for both CG and DD RAM transfers. This command can be issued at any time. It is the only command which the LCD module will accept while a previous command is still being processed.
9. **Set DD RAM Address:** sets the DD RAM address. Subsequent read or writes refer to the DD RAM.
10. **Write Data from RAM:** this instruction writes a byte to the CG or the DD RAM. The destination (CG RAM or DD RAM) is determined by the most recent set RAM Address command. The location to which the byte will be written is the current value of the address counter. After the byte is written the address counter is automatically incremented or

decremented according to the entry mode. The entry mode also determines whether or not the display will shift.

**11. Read Data from RAM:** reads a byte from the CG or DD RAM. The source (CG RAM or DD RAM) is determined by the most recent Set RAM Address command. The location from which the byte will be read is the current value of the address counter. After the byte is read the address counter is automatically incremented or decremented according to the entry mode.

**Table II. 4:** Instructions and codes of some LCD commands.

Instruction	Codes									
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear Display	0	0	0	0	0	0	0	0	0	1
Cursor home	0	0	0	0	0	0	0	0	1	x
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B
Entry Mode	0	0	0	0	0	0	0	1	I/D	S
Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	x	x
Function set	0	0	0	0	1	DL	N	F	x	x
Set CGRAM Address	0	0	0	1	CG RAM Address					
Read Busy Flag And Address Counter	0	1	BF	DD RAM Address						
Set DD RAM Address	0	0	1	DD RAM Address						
Write Data to RAM	1	0	Write Data to CG RAM or DD RAM							
Read Data from RAM	1	1	Read Data from CG RAM or DD RAM							

➤ x stands for don't care case

To configure an LCD display, four commands must be sent to LCD in either 4 bit mode, or in 8 bit mode. These commands are:

- Function set
- Display ON/OFF control
- Display Clear
- Entry Mode Set

After talking about the most important LCD commands, it is mainly needed to talk briefly about the two different display modes of LCD. These modes are given in the next section.

### **II.2.5. LCD display modes**

LCD includes two different displaying modes 4 bit or 8 bit mode. The difference between these two modes is the way data are sent to LCD. In the 8 bit mode, to write an 8 bit character to the LCD module, ASCII data is sent through the data lines D0-D7 and data strobe is given through the E line. 4 bit mode uses only 4 data lines. In other words, in this mode the 8 bit ASCII data is divided into two parts which are sent sequentially through data lines D4 – D7 with its own data strobe through the E line. The idea of 4-bit communication is to save as much pins that used to interface with LCD. The 4 bit communication is a bit slower when compared to 8 bit. The speed difference is only minimal, as LCDs are slow speed devices the tiny speed difference between these two modes is not significant. Thus the 4-bit mode data transmission is most commonly used [13].

## **II.3. Conclusion**

We have mentioned briefly some background parts for a 2x16 character LCD display. Mainly, we mentioned the LCD integrated with the Genesys board and we presented the needed configuration and initialization in order to display characters on the LCD display. In the next coming chapter, designs, simulations and experimental results will be presented.

# **CHAPTER III**

## **LCD Controller Design**

## CHAPTER III: LCD Controller Design

### III.1. Introduction

After understanding how a simple 2x16 character LCD display works, in this chapter we will design and simulate a digital system for later use to test LCD functionality and examine its performance.

### III.2. LCD controller design

Alphanumeric or more simply text LCD module is cheap and easy to interface using a microcontroller ( $\mu$ C) or FPGA. In the following sections, we designed LCD module in VHDL, interfaced it with a simple DUAL port memory. Moreover, we defined Arabic characters and displayed them on that LCD using two different techniques (Xilinx FPGA Virtex 5 and a Programmable Interface Controller (PIC) 16F877).

#### III.2.1. Design of LCD module

The Genesys FPGA development board includes a 2x16 character LCD module. This module can be used to display text and characters by sending appropriate commands from FPGA chip to that LCD module. To create an LCD module and interface it with Xilinx Virtex 5 FPGA we should create new ISE project and use new VHDL module. This VHDL code; based on sequential approaches; has two main parts: initialization and text display. The initialization part seen in **Figure III.1** consists of all steps listed in chapter II which are described below. After turning on this LCD, power on, at least 40ms must elapse before the function set instruction code can be written to set the bus width, number of lines, and character patterns (8 bit interface, 2 lines, and 5x8 dots are appropriate). After the function set instruction, at least 37 $\mu$ s must elapse before the display control instruction can be written (to turn the display on, turn the cursor off, and set the cursor to no blink). After another 1.52ms, the entry mode instruction sets address increment mode and display shift mode on. After this sequence, data can be written into the DDRAM to allow the display of characters.

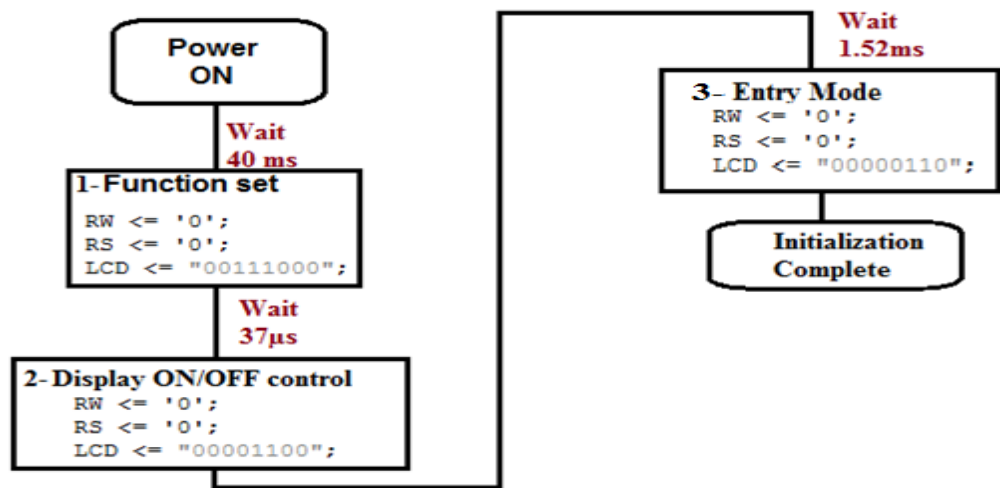


Figure III. 1: Transition diagram of LCD module initialization.

Before initializing the LCD module, it is recommend to connect it to a clock divider module to reduce the Virtex 5 internal clock of 100MHz because there is a maximum speed after which LCD display can no longer be able to receive and send data. The clock or frequency divider is a simple component implemented through the use of the scaling factor and a counter. The scaling factor is found by dividing the input frequency (FPGA internal frequency 100MHz) and the desired output frequency (400Hz). Therefore, the counter of the frequency divider generates an output signal of 400Hz each 250000 cycles. The clock divider module has been designed depending on those data. It generates the 400Hz signal by using a counter from 1 to 124999 because a clock signal is a square wave with a 50% of duty cycle (same time active and inactive); for this case, 125000 cycles active and 125000 cycles inactive. Since the counter begins at zero, the superior limit is 125000 - 1. This clock divider has been mentioned in every part on our designs. **Figure III.2** shows a simulation of that clock divider.

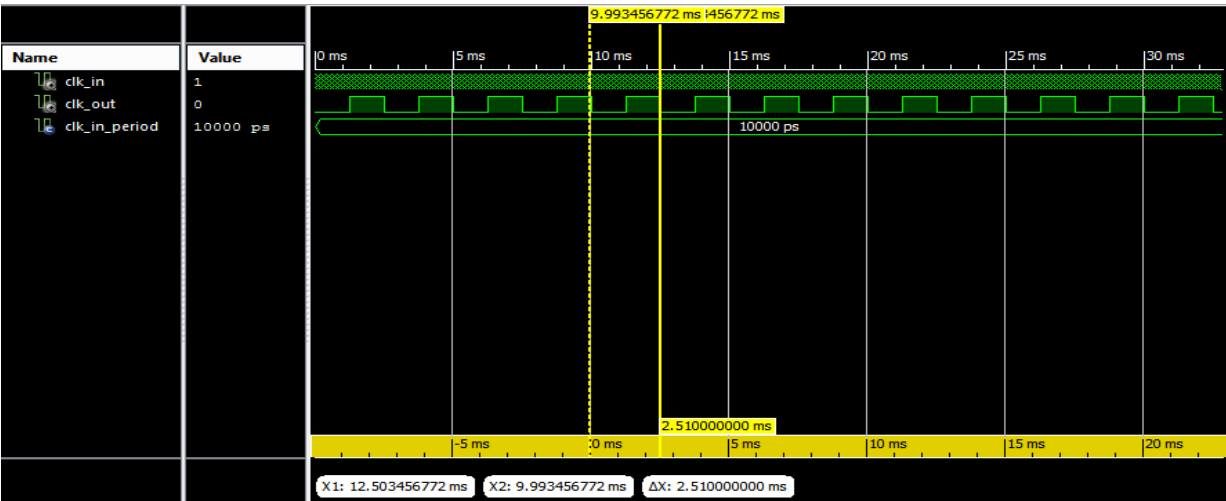


Figure III. 2: Simulation of the clock divider used.

Seven I/Os pins are used to design our LCD module. Five of them are outputs used to control the LCD module, say, Clear, RS, RW, En and LCD[7..0]. The rests are used as inputs one for the clock Clk and another one for the eight bit input data Data\_in[7..0]. The clock is the output of a clock divider module used to reduce the high frequency of the internal clock. This design allows user to enter data and display it directly in the LCD module. **Figure III.3** shows the block diagram of LCD module whereas **Figure III.4** shows the module pinout.

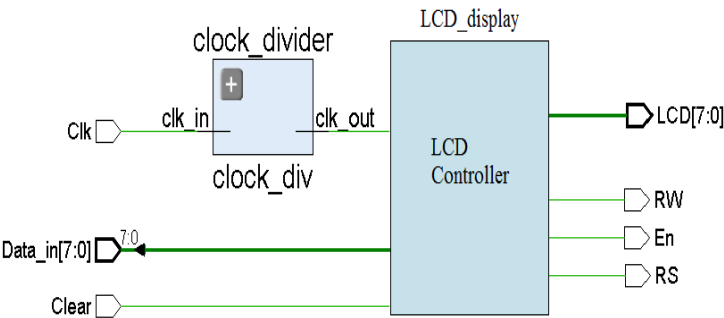


Figure III. 3: Block diagram for LCD module.

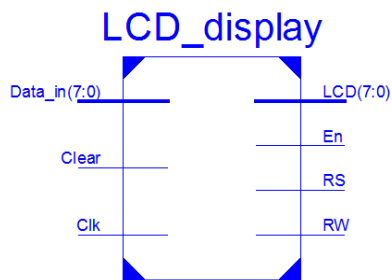


Figure III. 4: Pinout of LCD display.

To test the functionality of our design, we tried to display “Hello world” on LCD screen. The inputs are initially zero because we tried to send direct commands using VHDL codes. **Figure III.5** gives the simulation result.

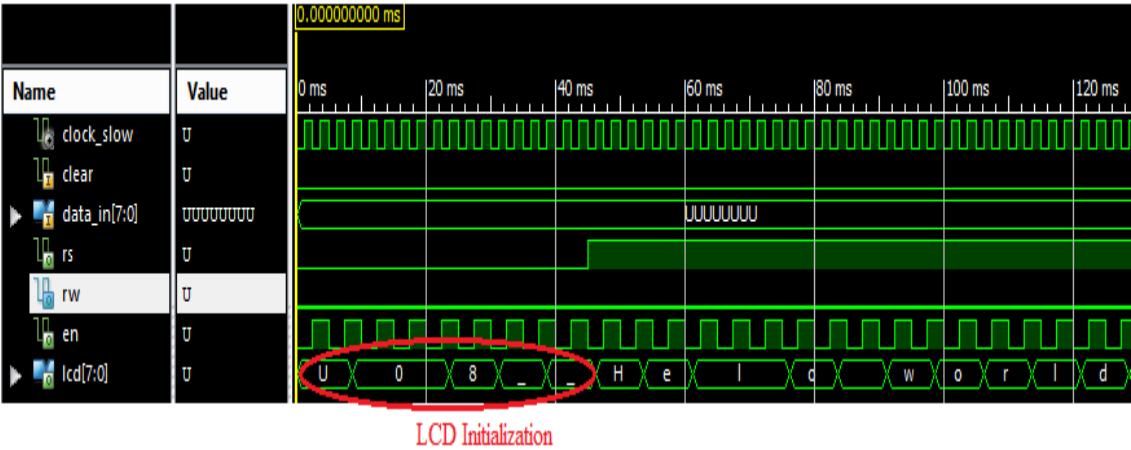


Figure III. 5: Simulation of LCD module.



After designing a system with an LCD module, we move to interface it with two different block RAMs. This is explained in the next sections.

### III.2.2. Block memory LCD interfacing

As it has been mentioned earlier in chapter I, the core generator is one of the Xilinx tools used to create an existing IP cores such as BRAMs. The BRAM Block is a configurable memory module created using the Core Generator tool (part of ISE design flow). In the following sections two BRAMs (single and dual BRAMs) have been used for LCD module design.

#### a. Single port block memory

The single port block memory module is generated based on the user specification width and depth. When this single BRAM is enabled, all memory operations occur on the active edge of the clock input (clk). The Block Memory can be configured to be active on the rising edge and the falling edge. When the block memory is disabled (enable inactive), the memory configuration and output value remain unaltered. During a write operation (wea asserted), the data presented at the port's data input is stored in memory at the location selected by the port's address input. The Virtex implementation supports a single write mode option, Read-After-Write. This write mode causes the data being written to the addressed memory location to be transferred to the data output port when a write operation occurs[8].

In this part, we have created a 16x 8 single port RAM with 16 bit as memory size and an 8 bit as a width size for the word we want to display later on our module.

Eight bit width due to the fact that the LCD displays characters 8 bit each. So we need it to hold the ASCII code of each character on one memory location. As another specification for that memory is the operating mode which was the Write First mode. A last specification is the memory initialization option which was used to initialize the memory content. This can be done by specifying the desired information to be initialized in a separate text file called a COE file. In our design, we used a simple COE file given in **Figure III.6** that contained the ASCII codes in binary format (taking in to consideration the ASCII code of space (00100000)) for the "Master students" sentence.

When specifying the initial contents for our memory in a COE file, the keywords MEMORY\_INITIALIZATION\_RADIX and MEMORY\_INITIALIZATION\_VECTOR were used. The MEMORY\_INITIALIZATION\_VECTOR takes the form of a sequence of comma separated values, one value per memory location, terminated by a semicolon. Any amount of white space, including new lines, can be included in the vector to enhance readability.

The format of an individual value in the vector will depend on the MEMORY\_INITIALIZATION\_RADIX value (in our case it is a binary format 2). The vector must be consistent with the MEMORY\_INITIALIZATION\_RADIX value and must fall within the range of 0 to 2DATA\_WIDTH -1, that is 0 to 15. Values must not be negative. Note that the first entry in the COE file corresponds to the lowest block memory address. To support HDL simulations, Memory Initialization Files (MIFs) containing the initialization values are generated. These files must be copied to the active simulation directory for a successful simulation of a single port block memory core [8].

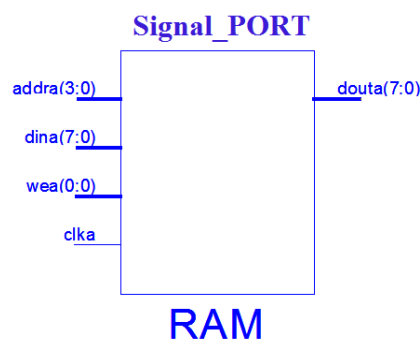
```

1 memory_initialization_radix=2;
2 memory_initialization_vector=
3 01001101,
4 01100001,
5 01110011,
6 01110100,
7 01100101,
8 01110010,
9 00100000,
10 01010011,
11 01110100,
12 01110101,
13 01100101,
14 01101110,
15 01110100,
16 01110011,
17 01110011,
18 00100000;

```

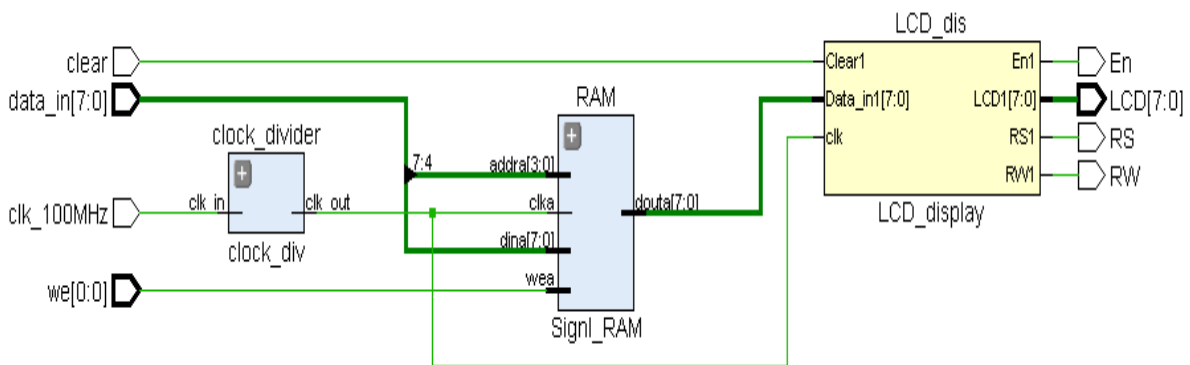
**Figure III. 6:** Example of COE file to display “Master students”.

All other non-mentioned specifications are used as default. The pinout of the desired single port RAM can be seen in **Figure III.7**. This memory has five I/O pins, one of them is eight bit vector used for the output data (douta [7..0]), the others are inputs: the addra signal is used as a four bit length to specify the address for read or a write operation vector . The wea signal is used to enable the read and the write operations. Also, the dina signal used as 8 bit input data to allow users to enter data (dina). Without forgetting the clock signal.



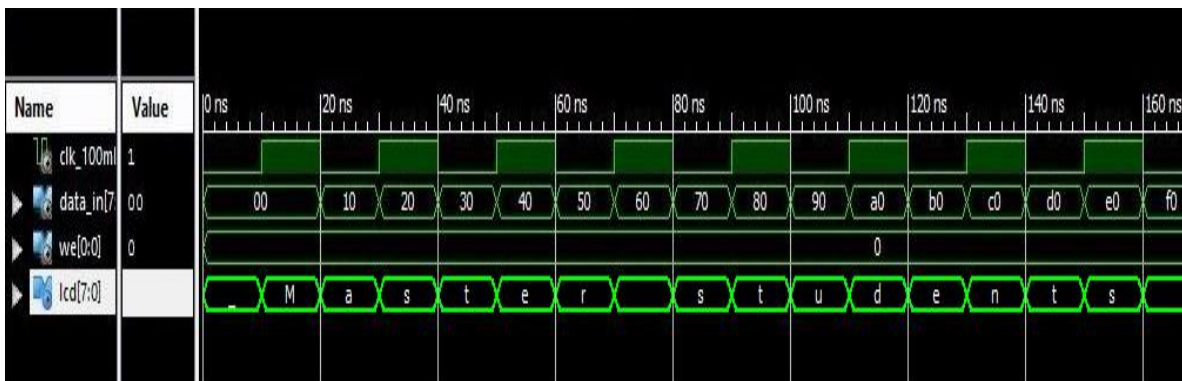
**Figure III. 7:** Single port RAM pinout.

After creating that block memory, we interfaced it with the 2x16 character LCD by connecting the BRAM output to the LCD controller that was generated in section II.2.1s and the clka signal to the clock divider module. Also, the dina single port BRAM's pin is connected to the Data\_in signal and the addra pin is connected to the fourth MSB of Data\_in (7 downto 4) as it can be seen in **Figure III.8**. This design was performed to display the content of the COE file in LCD module first. As a next step, memory content was changed by allowing the user to enter different 8 bit content (using Data\_in signal that is connected to the dina of single port RAM) respecting the fact that those code must be included within the ASCII code table.



**Figure III. 8:** Block diagram of a 16x8 single port RAM interfaced with LCD.

The simulation given in **Figure III.9** shows the initialized COE file displayed on LCD module with its corresponding address.



**Figure III. 9:** The simulation of single port RAM using COE file.

After designing that system, we went one step further to examine another BRAM type that is the simple dual port BRAM memory.

### b. SDP block memory

SDP RAM can read and write different memory cells simultaneously at different addresses. In other words, it has two different ports port A and port B. As a result independent Read and Write operations can occur simultaneously, where port A is the primary Write port and port B

is the primary Read port. When the Read and Write port access the same data location at the same time, it is treated as a collision. This is the main difference between dual port RAM (DP RAM) and single port RAM, as single port RAM can only be accessed at one address at a time. Therefore, single port RAM allows only one memory cell to be read/write during each clock cycle. In this part, we design a digital circuit such that the LCD module is interfaced with a SDP RAM. This BRAM has the following specifications:

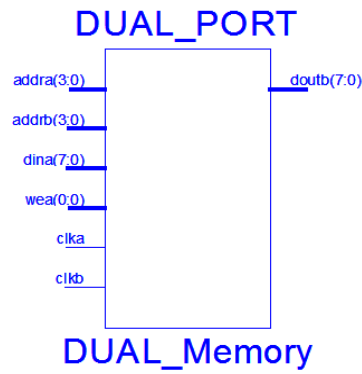
1. The read/write width is 8 bit.
2. The read/write depth is 16 bit.
3. The clock used for the read and the write ports are common.
4. Port A is with **Write First mode** (The data input is stored in memory and mirrored on the output) whereas port B is specified to be in **Read First Mode**. In this mode, data previously stored at the write address appears on the output latches.
5. The content of that memory was initialized using a COE file seen in **Figure III.10**. This COE file contains the ASCII codes in a binary format of the sentence we want to display “Welcome to IGEE“.

```

1 memory_initialization_radix=2;
2 memory_initialization_vector=
3 01010111,
4 01100101,
5 01101100,
6 01100011,
7 01101111,
8 01101101,
9 01100101,
10 00100000,
11 01110100,
12 01101111,
13 00100000,
14 01001001,
15 01000111,
16 01000101,
17 01000101,
18 00100000;
```

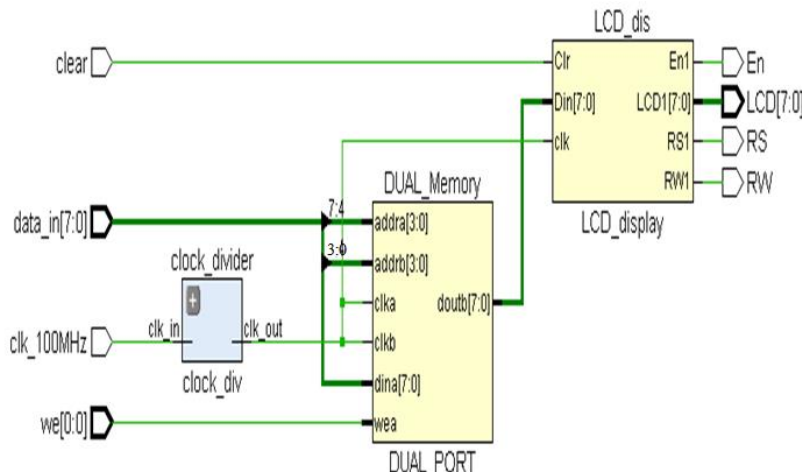
**Figure III. 10:** The COE file used for SDP RAM design.

The pinout of our 16x8 SDP RAM can be seen in **Figure III.11**. It has seven pins, one of them is an eight bit output vector. The inputs are two bus addresses for port A and port B, say addra and addrb with four bit length each to specify the address for read and write operation. It has also an eight bit vector for the input data (dina), one vector for write enable to force that memory to write the entered data. Without forgetting the clock of each ports to drive that memory clka and clkb. To resume, port A has its corresponding I/O pins which are addra for address , dina for input data , wea for write enable and clka for the clock. Port B has a clock clkb , addrb for address and doutb for the output of that memory.



**Figure III. 11:** Simple SDP RAM pinout.

After creating that block memory, we interfaced it into a 2x16 character LCD. The input data (dina) to the SDP RAM are connected to some inputs data say data\_in (8 bit) and the port A address (addra) is the fourth MSB of that data\_in (7 downto 4) whereas the addrb is connected to the fourth LSB of the same input data as it can be seen in **Figure III.12**. The clock of that memory is drives by a simple clock divider to minimize the speed of the internal clock with frequency of 100 MHz. This clock is connected to both ports' clock (clka and clkb). The output to that memory is connected to the LCD module generated previously. By using the COE file we can display directly the initial memory content to the LCD module without need to send input data to it first.



**Figure III. 12:** Block diagram of a 16x8 SDP RAM interfaced with LCD.

The simulation part should be mentioned to see clearly the LCD output after it is been initialized and received data. The waveform seen in **Figure III.13** is for the COE file only. As it can be seen LCD module takes few time till it is initialized and once initialization is completed the LCD displays the data given in its corresponding address.

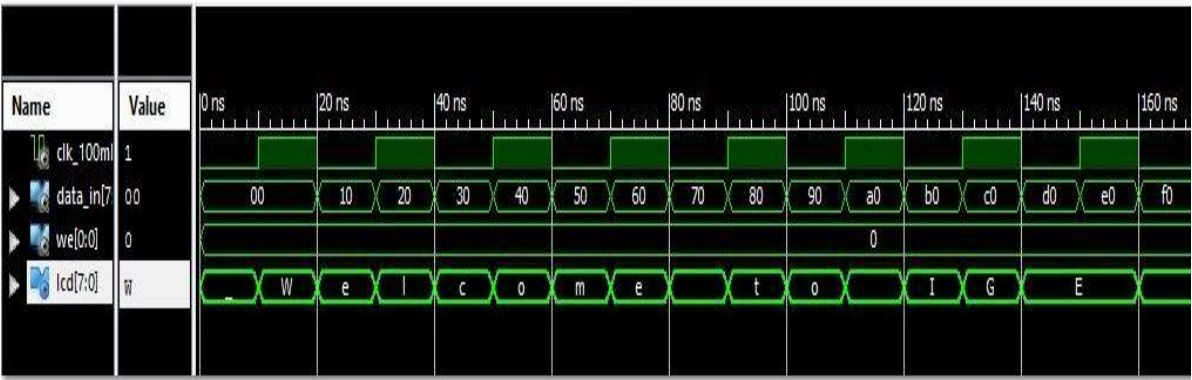


Figure III. 13: Simulation results for LCD display using COE file.

III.2.3. Arabic Character Display

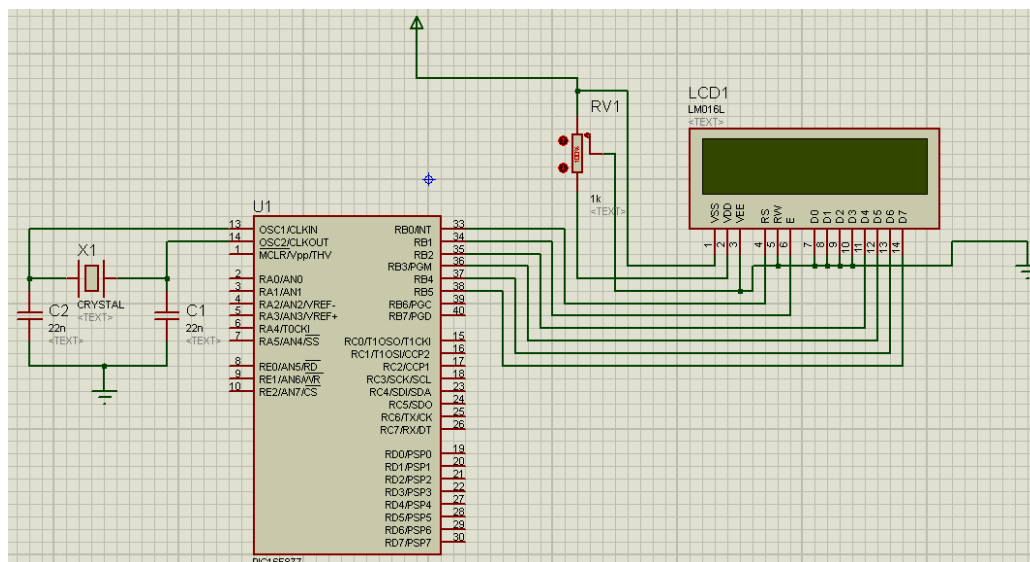
In this part, we have described the method used to create Arabic characters and displayed them on LCD display by using both a PIC 16F877 and FPGA. In fact, we design the  $\mu$ C system first due to the fact that this PIC is one of the most advanced microcontroller from Microchip. It is widely used for experimental and modern applications because of its wide range of applications, high quality, and ease of availability.

a. Microcontroller design

This design was done using two different software the mikroC PRO for PIC and ISIS Proteus. The former was used to write the C code to drive the LCD module and the later was used for hardware design to connect the PIC 16F877 to a 2x16 character LCD module.

As it has been mention earlier, LCD display can operate in two different modes. In fact, it is preferable to use 4 bit mode with the microcontroller PIC 16F877 instead of 8 bit mode. This is because the 4 bit data transfers use 4 I/O lines less than 8 bit data transfers.

As it can be seen in **Figure III. 14** the MSB bidirectional LCD data bus was used D4 through D7 the rest was connected to ground. PORTB is being used as data bus for the LCD. Also, RB0 pin is used as RS (Register Select for LCD) and RB1 pin is used as E (Enable pin for LCD).



**Figure III. 14:** Design of a PIC 16F877 interfacing with 2x16 LCD module.

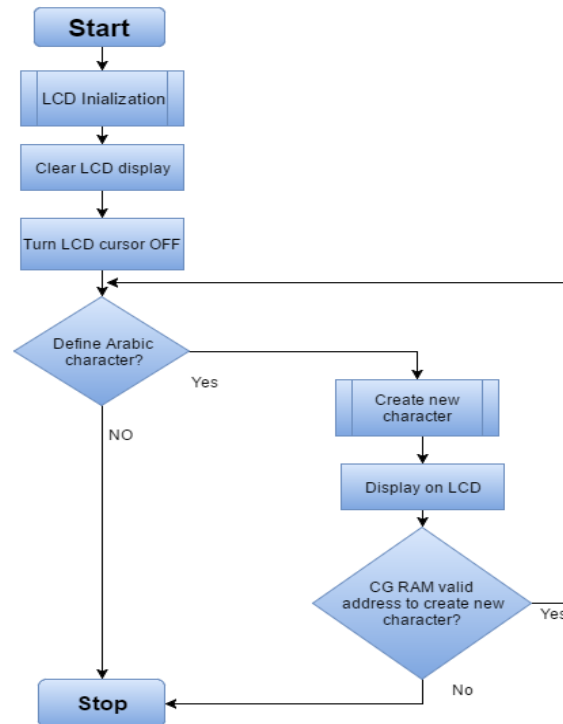
After connecting the needed digital circuits, it is recommended now to use a software design to test the functionality of that digital circuit using the same previous procedures to define an Arabic character. **Figure III.15** shows a simple Arabic character which has been tested by trying in our design. After determining the 5x8 dot matrix values (5x7 dot matrix+ cursor line), we have made an array of these values to transmit it to the CGRAM of LCD.

■ ■ ■ ■ ■	Line 1 0b00000
■ ■ ■ ■ ■	Line 2 0b00000
■ ■ ■ ■ ■	Line 3 0b01100
■ ■ ■ ■ ■	Line 4 0b10100
■ ■ ■ ■ ■	Line 5 0b11100
■ ■ ■ ■ ■	Line 6 0b00111
■ ■ ■ ■ ■	Line 7 0b00000
■ ■ ■ ■ ■	Line 8 0b00000

**Figure III. 15:** Example of 5x8 dot matrix representation for an Arabic character (ة).

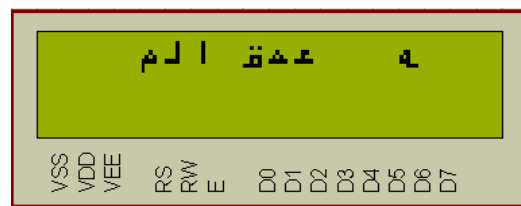
The same procedures must be repeated to in the case the user wants to display other characters. LCD works with the same principle within a system designed with PIC or FPGA. It needs some commands, initialization and some cursor's operation to obtain the required results. This can be done using simple functions that are already defined in mikroC libraries. The flowchart seen in **Figure III. 16** describes the algorithm used to test the functionality of that digital circuit. At the beginning an initialization function is needed in order to initialize the LCD Module connected to the PIC pins. This initialization includes two line display, 5x8 dots and a 4 bit mode. Next, a clear display and turn OFF cursor functions are also needed to provide a pure display with no overlap with cursor once it is blinking or with previous display. Then a function to create the desired character is defined. It uses the 5x8 dot matrix values in an array to translate

it to an Arabic character and display it in one memory location. Since CG RAM can hold only eight characters, program must check if a valid address.



**Figure III. 16:** Flowchart of LCD custom character display.

We try to print the custom characters created previously. In order to do that we programmed the hex file in our PIC16F877 and run it. The displays can be seen in **Figure III.17**. Once the dot matrices are loaded into the CGRAM, the data will stay there unless the LCD module is power cycled. So our custom character now resides at character location (CG RAM address). If we want to display the newly created custom character, it is as simple as is displaying any other character, just send the character location to the LCD and it shows up.



**Figure III. 17:** Custom characters LCD displays.

Only eight Arabic character has been introduced due to the fact that CG RAM can hold only eight bytes only.

Once we designed that digital system in a PIC and tested. We were sure that such system could be designed using FPGA tools too.

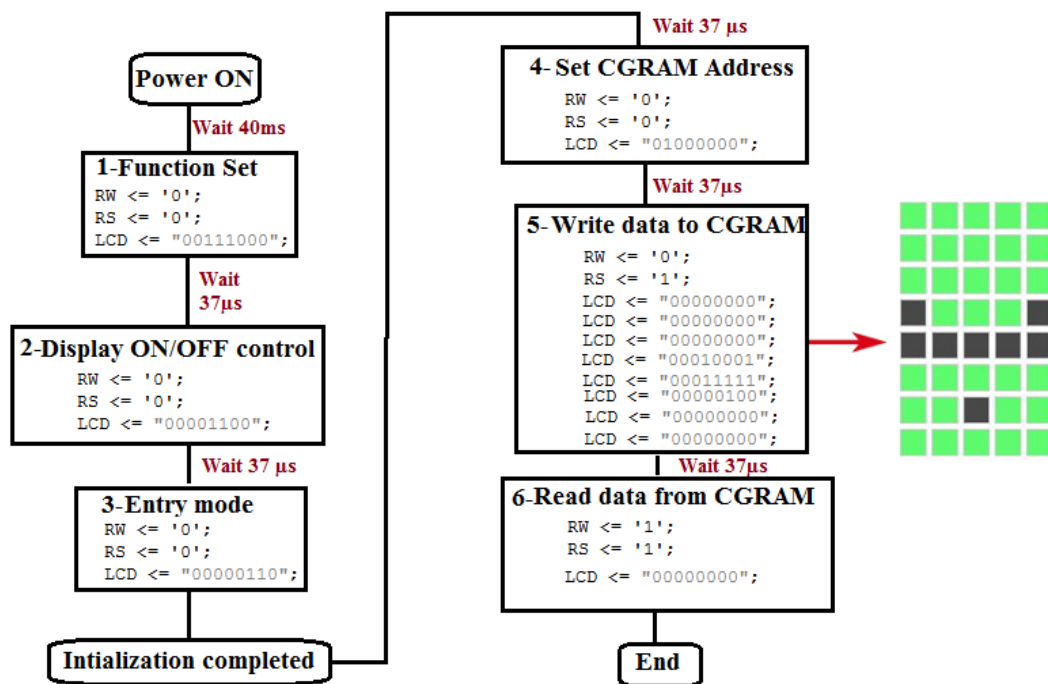


### b. FPGA design

By using the same digital design seen in section III.2.1 and the same steps we could design another new system to allow users to displays Arabic characters on the LCD module. However, the used commands differ in some bits with the previous system. There is a need to set CG RAM to the location we want to display our Arabic character say address 00h, write to it and read from it after initializing it. These commands can be seen in **Figure III.18**.

The CG RAM write command is used to identify the dot matrix used to define the character. It use eight different bits depending on the character we're about to display. A simple Arabic character was been tested by trying to generate it, first we made a box of 8 by 5 dots. Then filled the dots required to make the custom character we want to display.

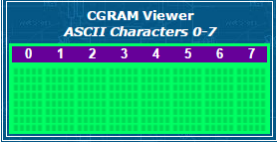
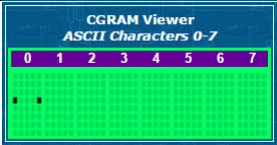
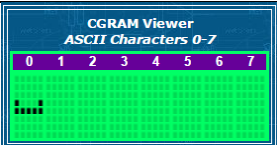
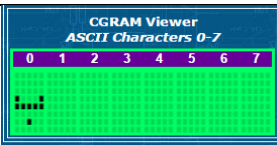
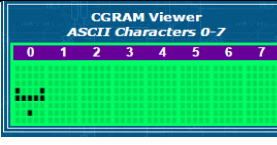
After filling the dots, we found out the value of each line. For example, the first line has a value of 0b000000. Similarly, the second, the third, the sixth and the eight line has the same value 0b000000. The fourth line has a value of 0b10001 and fifth line has a value of 0b11111. Finally, the eight line has one dots to be displayed, so it has the value of 0b00100. After finding out these values, we have used them in the required 8-bit commands to transmit them to the CGRAM of LCD.



**Figure III. 18.** Transition diagram of LCD module initialization for Arabic display.

The dot matrix is resumed in **Table III.1**. It shows the CG RAM content once LCD initialization is done and the address of the CG RAM has been set. The figures shown there were obtained using one of LCD simulator software.

**Table III. 1:** The Content of the CG RAM for an Arabic character display.

CG RAM content	Description
	The data “00000000” is sent to CG RAM to define the first row of the dot matrix going from top to bottom. AC will increment to the next CG RAM address. The figure shown to the right represents the content of the CG RAM. The data written to CG RAM has been entered three time defining the three rows content, hence AC=2. So CG RAM is at address 2.
	The next data to be written is “00010001” and AC=4. This data will occupy the address 4 in CG RAM memory.
	The data to be written is “00011111” on row 5 and AC=5.
	The next data to be written is “00000100” and AC=7. It can be seen in the next figure.
	The last entered data is “00000000” and AC=8. After defining the eight locations this how would an Arabic letter display looks like inside the CG RAM and on LCD display.

Once these commands are modified they can be added to the VHDL module to test the functionality of the design system.

### III.3. Conclusion

At the end of this chapter, LCD module has been examine by presenting the needed LCD configuration and initialization. We have interfaced two different BRAMs (the simple port RAM and SDP RAM) to LCD module in order to test them through the use of our designs. Moreover, Arabic character LCD display system was designed.

## **CHAPTER IV**

### **Implementation and Experimental Results**

# CHAPTER IV: Implementation and Experimental Results

## IV.1. Introduction

In this chapter we show the implementation of the LCD module and the experimental results found.

## IV.2. Experimental results

After dealing with the design part, we are going to specify the obtained results and discuss them.

### IV.2.1. LCD interfacing

The designed system discussed in chapter III section **III.2.1** allow connection between the LCD module and the Virtex 5 FPGA development board. In fact, this module is connected to it using eleven pins (for more details go back to section **II.2.2** and **Table II.1**). The whole design has twenty one pins. Eleven of them are for the LCD module. The rest pins are described in **Table IV.1**. This table shows the port name, a description of what it physically corresponds to clock, switch or push button, and the Genesys pin number that it is connected to. Those pins are assigned using a PlanAhead Xilinx tool and a User Constraint File (UCF).

**Table IV. 1:** Pins configuration of the LCD display.

PORT NAME		Genesys' pins	Descriptions
Clear	Push button	G6	BTN0 used for reset signal
Clk	Internal clock	AG18	100 MHz oscillator used for timing
Data_in	Slide sw0	J19	Data input 0
	Slide sw1	L18	Data input 1
	Slide sw2	K18	Data input 2
	Slide sw3	H18	Data input 3
	Slide sw4	H17	Data input 4
	Slide sw5	K17	Data input 5
	Slide sw6	G16	Data input 6
	Slide sw7	G15	Data input 7

The slide switches have been connected to the Data\_in 8 bit input vector. The Virtex 5 internal oscillator is connected to the input signal of the clock divider.

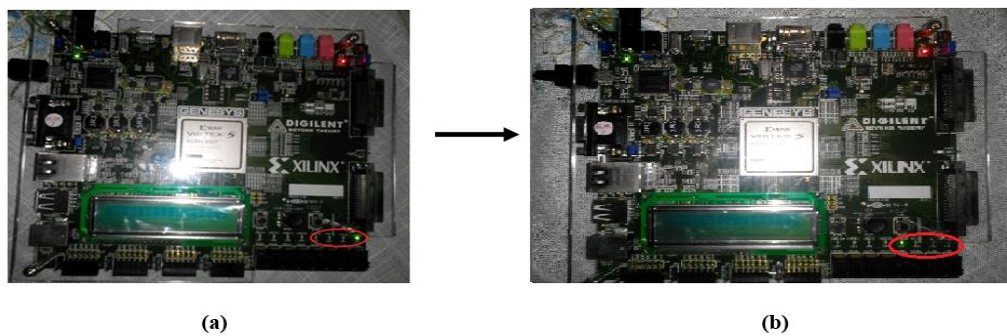
Pushbuttons often generate spurious open/close transitions when pressed, due to mechanical and physical issues. These transitions may be read as multiple presses in a very short time fooling the program. Because we have used a pushbutton we have to go through the process used to debounce it.

One solution would be to add an RC hardware filter. But a simpler solution is suggested. A digital hardware solution (digital circuit) can be used. To see the effect of the bouncing phenomena, a 16 bit up counter (from 0000 up to 1111) has been implemented and downloaded to the Genesys board. This design has six pins. **Table VI.2** shows pin definitions for that counter with a description.

**Table IV. 2:** Pins configuration for the 16 bit up counter.

PORT NAME		Genesys' pins	Descriptions
clr	Slide switch 0	J19	Used for reset signal
clock	Pushbutton	G7	Used as clock for the 16 bit counter
Q	LEDs0	AG8	Q(0) Data output 0
	LEDs 1	AH8	Q(1) Data output 1
	LEDs 2	AH9	Q(2) Data output 2
	LEDs 3	AG10	Q(3) Data output 3

We have used a push button pin for a clock, a slide switch for clr input and the fourth LSBs LEDs to display the content of the 16 bit counter. **Figure IV.1 (a)** shows the output displays on LEDs of a 16 bit counter after one clock cycle and **Figure IV.1 (b)** show the output after three clock cycles. In fact the output of the counter should be 0011 but not 1000. This was due to the fact that the push button used for the clock signals is bouncing.



**Figure IV. 1:** Counter Output after (a) one pushbutton's press and (b) three presses.

To debounce any pushbutton the following a digital circuitry must be used. **Figure IV.2** shows one of these circuits. The circuit continuously clocks the button's logic level into FF1 and subsequently into FF2. So, FF1 and FF2 always store the last two logic levels of the

button. When these two values remain identical for a specified time, then FF3 is enabled, and the stable value is clocked through to the result output. The XOR gate and N-bit counter accomplish the timing. If the button's level changes, the values of FF1 and FF2 differ for a clock cycle, clearing the N-bit counter via the XOR gate. If the button's level is unchanging (i.e. if FF1 and FF2 are the same logic level), then the XOR gate releases the counter's synchronous clear, and the counter begins to count. The counter continues to increment in this manner until it reaches the specified time and enables the output register or is interrupted and cleared by the XOR gate because the button's logic level is not yet stable. The counter's size determines the time required to validate the button's stability. When the counter increments to the point that its carry out bit is asserted, it disables itself from incrementing further and enables the output register FF3. The circuit remains in this state until a different button value is clocked into FF1, clearing the counter via the XOR gate.

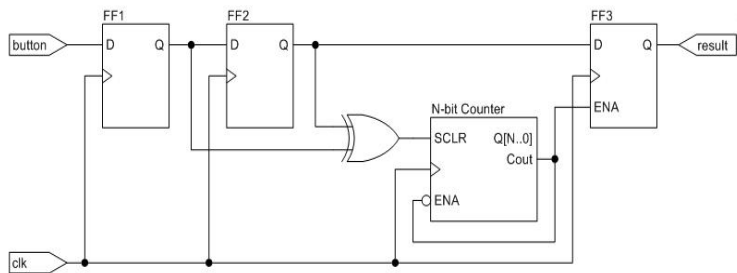


Figure IV. 2: Digital circuit for debouncing a push button.

To check the functionality of this problem, an implementation design has been made. After compiling and running the program, we could notice that by pressing the same pushbutton clock (say G7) the counter starts the count and display a 1 in LEDs. Once we press it for second time the counter will be at value 2 (0010) .The result was displayed on LEDs and can be seen in Figure IV.3.

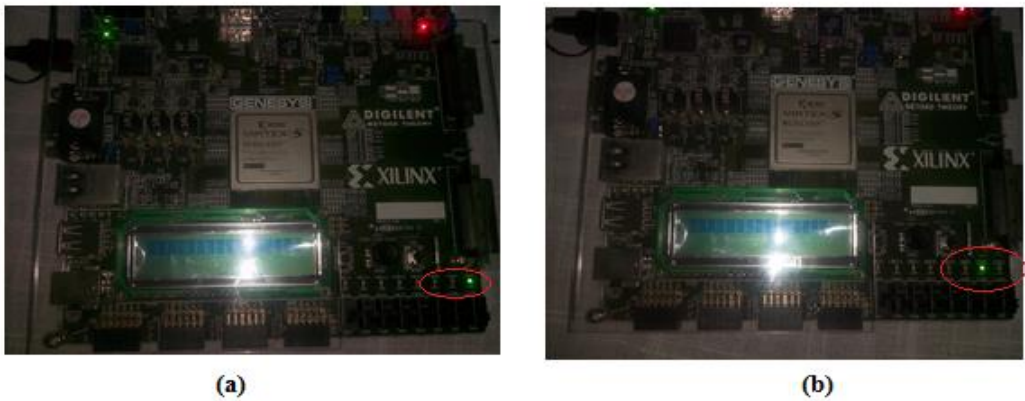


Figure IV. 3: Counter output after (a) one push button press and (b) two pressed.

However, Xilinx offers direct solution to eliminate it by enabling a pull-up or pull-down resistors via the UCF file.

After debouncing the push button used for the Clear signal in our design, connecting the Virtex 5 internal clock to the clock divider module, assigning pins and compiling the program, we tried to implement the design into the LCD module integrated with the Genesys board. The input data (connected to the Genesys eight slide switches) is going to be displayed into the LCD module are introduced by the user. As an example, “Hello World” was displayed in the first LCD module line. The corresponding ASCII code of each character is given in **Table IV.3**.

**Table IV. 3:** ASCII code of the characters to be used.

Character	ASCII code in binary
H	01001000
e	01100101
l	01101100
o	01101111
W	01010111
r	01110010
d	01100100
Space	00100000

The next stage involves going through the Synthesize, Translate, Map and Place and Route Steps. These steps are carried out by the Project Navigator software (ISE), and are briefly described as follows:

- **Synthesize:** generates netlists for each source file.
- **Translate:** merges multiple files into a single netlist.
- **Map:** the design is mapped to slices and I/O blocks.
- **Place and Route:** works out how the design is to be placed on the chip and components connected.

The last step was to generate the program file, and download it to the Genesys board. **Figure IV.4** shows the LCD displays after generating the program file, and downloading it to the Genesys board using iMPACT Xilinx tool.



**Figure IV. 4:** LCD module display using a simple VHDL.

The use of Genesys switches allows the user to display any character he would like to display within the existing ASCII code table seen earlier. It is necessary to know the different commands and the sequence of operation to be able to write letters on the LCD. These commands require some microseconds and sometimes more than 1 millisecond as a time delay to offer a clear displays. In the digital circuit, a clear signal (Clear) has been introduced since it is an essential part in any digital system and its function in this component is to restart the LCD controller; reinitialize it an retest the display once again.

One step further is to discuss and analyze the obtained results using an existing IP core interfaced with an LCD.

#### IV.2.2. Block memory LCD interfacing

In this part, we tried to implement the designed digital system of both sections **III.2.2 a** and **b**. That is interfacing the LCD module with both single port RAM and SDP RAM. The design of single port BRAM seen in **Figure III.6** and the one for the SDP RAM seen in **Figure III.9** are discussed in the table **Table IV.4**.

**Table IV. 4:** BRAMs LCD interfacing implementation.

Single port RAM	SDP BRAM
The system has 22 I/O pins to be connected to the Genesys board (8 pins for input data (data_in) connected to the 8 board switches, 1 pin for clear connected to one of the pushbutton, 1 pin for Virtex 5 internal clock (100MHz), 1 pin for the write enable signal and the rests are outputs we have the LCD pins seen previously.	Same thing about the pins number with the single port RAM.  The connection of those pins to the designed RAM and LCD module have discussed previously so we mentioned here how these pins are interfaced to the Genesys board only.



Repeating the same procedure described in section <b>IV.2.1</b> in order to test the functionality of our design. The content of the COE file was displayed first “Master Students “. After that we overwrite data in some memory locations by specifying the need address to display “LCD display”.	The LCD of the Genesys board after downloading the program into it. It displays the content of the COE file “Welcome to IGEE “. However, once we clear the content of the LCD and write on our Dual port memory one of the existing ASCII codes; the previous content at the desired address will be overwritten to display “Lily & DIDja”.
Activating the write enable pin enables writing to the memory locations. When active, the contents of the dina bus is written to memory at the address pointed to by the Addra bus. The output are loaded (Write First operating mode). When wea is inactive, a read operation occurs, and the contents of the memory addressed by the addra bus are driven on the doutb bus.	The SDP RAM can read and write different memory cells simultaneously at different addresses. This is why, we used different address for port A and B the four MSB switches for port A and the four LSB ones for port B. Once “we” is enabled by setting it to 1, a write operation (entering data using switches) can be performed in port A simultaneously with another read operation (display memory content to LCD module) with the different memory address.

A simple comparison between single and DP RAM can be made in this level. A single port memory has one data/address port to read or write at a time whereas the SDP RAM has two data/address ports. It can read and write at the same time using both ports.

If we wanted to send multiple commands at once, the En signal must not be one all the time. It is recommended to switch between one and zero state before sending data to LCD module. One step further is to comment the obtained results once creating custom Arabic characters.

### IV.2.3. FPGA Arabic characters on LCD

This part dealt with the implementation and experimental results once trying to create Arabic characters in LCD. The design seen in section **III.2.1** with the same pins. Once we set the CG RAM address and set RS signal to 1 data will be sent to CG RAM instead of the DD RAM. Eight characters are available, and they reside in the ASCII codes 0 through 7. The dot matrix values discussed previously are sent in the 8 bit bytes from the top row to the bottom row and is left justified, meaning that only the bottom 5 bits matter(it is a 5x7 dot matrix). After that a read command was needed to display the content in to the LCD module.

Since the CG RAM address has been set to 00000000 the AC will start at value 0 and keep incrementing till it reached the value 8. This means that the first character has been created and saved in the first location in CG RAM memory. We didn't succeed to display the Arabic character, it is just blinking the cursor. First of all, we didn't have enough time to do a full troubleshooting to the designs since we started to work with Virtex II pro FPGA. We didn't find the necessary resources to work on this board, thus we decided to use the Genesys Virtex 5 development board for which we took too much time to get familiar with it.

### IV.3. Conclusion

In this chapter we presented our implementation of LCD module interfaced with a Xilinx FPGA development board, then we described the method used to create an Arabic character. Also, we discussed the obtained results. IP cores were used and block diagrams of LCD component has been made in order to develop the idea of creating an LCD custom IP core.

# **General Conclusion**

## **General Conclusion**

In this project, we have examined a 2x16 character LCD module, interfaced it with two different block memories. Moreover, we defined Arabic characters and displayed them on that LCD using two different techniques (Xilinx FPGA Virtex 5 and Programmable Interface controller (PIC) 16F877). Our work is done by presenting features of FPGAs. We learned the general architecture of FPGAs then we examined in detail the specific architecture of Xilinx Virtex 5 LXT family. After that we presented our hardware platform which is a Genesys board and its associated development tools, we learned that any hardware design passes through different steps in the design flow before it can be loaded to the FPGA. Moreover, we tried to give brief ideas about the IP cores and the core generator in order to create some needed IP cores. We examined one of the most common device which is attached to Genesys Virtex 5 FPGA board; that is the LCD display by presenting the needed configuration and initialization in order to display characters on it. A very important LCD aspect has been introduced that is the Arabic character display. IP cores were used and block diagrams of LCD component has been made in order to develop the idea of creating an LCD custom IP core.

This work was done to show benefits and importance of such modules in the digital world. Actually, once designers build a real life/real world electronics based projects, they need a medium device to display output values and messages. The most available electronic display is the character LCD module with its different size and specifications. LCD module forms a very important part in many digital designs. So the knowledge on interfacing this module to FPGA is very essential in designing embedded systems and other digital systems.

Our objectives was about integrating Arabic characters in FPGA and make a design to reuse it in the creation of a custom IP core, we didn't reach the point of displaying Arabic character. Because we didn't have enough time to do a full designs since we started working with the Virtex II Pro Xilinx FPGA which does not include an interfaced LCD module. Thus we used Virtex 5 FPGA instead.

We can use the BMG design and the created block diagrams of a character LCD to create custom LCD IP core for future work. The development of this idea gives user chance to select different options for LCD displays including the LCD mode, size (2x16, 4x16, 4x20 ... etc.) and to generate a VHDL or Verilog code even more to create its schematic block diagram for LCD display depending on the users specifications.

# References

## References

- [1] S.Brown and J.Rose. “Architecture of FPGAs and CPLDs: A Tutorial”. Presented in Department of Electrical and Computer Engineering, University of Toronto, in 2000.
- [2] Core technologies. “FPGA Architectures Overview”. 02 May 2016. [Online]. Available: [https:// www.pdx.edu.nanogroup/files/FPGA-architecture.pdf](https://www.pdx.edu.nanogroup/files/FPGA-architecture.pdf)
- [3] “Genesys board reference manual”. 22 March 2016. [Online]. Available: [https://reference.digilentinc.com/\\_media/genesys:genesys\\_rm.pdf](https://reference.digilentinc.com/_media/genesys:genesys_rm.pdf)
- [4] “16x2-lcd-module-datasheet”. 15 April 2016. [Online]. Available: <http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet>,
- [5] “Dot LCD Controller/Driver”. 15 April 2016. [Online]. Available: [http://www.newhavendisplay.com/app\\_notes/ST7066U.pdf](http://www.newhavendisplay.com/app_notes/ST7066U.pdf)
- [6] “IP core”. 07 March 2016. [Online]. Available: <http://whatis.techtarget.com/definition/IP-core-intellectual-property-core>.
- [7] “CORE Generator Guide”. 22 April 2016. [Online]. Available: <http://www2.informatik.huberlin.de/~fwinkler/psvfpga/synthese/ISE-Dokumentaion/docs/cgn/cgn.pdf>,
- [8] <http://www.xilinx.com>. [Online].
- [9] “Character LCD Module Controller (VHDL)”. May 2016. [Online]. Available: <https://eewiki.net/pages/viewpage.action?pageId=4096079>
- [10] “FPGA Comparative Analysis”. 25 April 2016. [Online]. Available: <https://eewiki.net/pages/viewpage.action?pageId=4096079>
- [11] “LCD”. 15 April 2016. [Online]. Available: <http://www.birnboim.com/nyu/pcomp/techresearch/howlcdswork.html>
- [12] “Architecture-Specific Packing for Virtex-5 FPGAs”. 30 March 2016. [Online]. Available: <http://janders.eecg.toronto.edu/pdfs/fpga45-ahmed.pdf>
- [13] “LCD interfacing with PIC Microcontroller”. 12 April 2016. [Online]. Available: <https://electrosome.com/lcd-pic-interfacing/>
- [14] “HD44780 Character LCD Displays”. 23 May 2016. [Online]. Available: <http://www.protostack.com/blog/2010/03/character-lcd-displays-part-1/>