

*“Implementation of Mo soft core in FPGA.”*



Author

Muazzam Shahzad

Fall –MS2015 (CE) 00000117611

Supervisor

Dr. Farhan Hussain

COMPUTER ENGINEERING DEPARTMENT  
COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY  
ISLAMABAD  
SEPTEMEBR 2019

**“Implementation of Mo soft core in FPGA.”**

Author

Muazzam Shehzad

Fall 2015-MS (CE) 00000117611

In partial fulfillment of the requirements for the degree of  
MS Computer Engineering

Thesis Supervisor

Dr. Farhan Hussain

Thesis Supervisor's Signature: \_\_\_\_\_

DEPARTMENT OF COMPUTER ENGINEERING  
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING  
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,  
ISLAMABAD  
AUGUST, 2019

## **Declaration**

I solemnly certify that this research work titled “Implementation of Mo core in FPGA” is my own work. I haven’t presented that work anywhere for assessment; neither I have taken material from other resources.

The contents that I have referenced from other resources is properly acknowledged / referred.

Signature of Student

Muazzam shehzad

Fall 2016-MS (CE) 00000117611

## **Language Correctness Certificate**

This thesis has been thoroughly read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. The thesis is as per format provided by the university.

Student Signature

Muazzam Shehzad

Fall 2015-MS (CE) 00000117611

Signature of Supervisor

Dr. Farhan Hussain

# Copyright Statement

- The copyright on the text of this thesis belongs to the student author. Copies (by any means), in whole or in part, may only be made in accordance with the instructions given by the author and filed with the NUST College of E & ME Library. Details can be obtained by the librarian. This page must be part of all copies made. Other copies (by any means) cannot be made without the written permission of the author.
- The ownership of all intellectual property rights that may be described in this thesis belongs to NUST College of E & ME, subject to any prior agreement, and may not be made available to third parties without the written consent of the owner. College of E & ME, which will prescribe the terms and conditions of said agreement.
- Additional information on the conditions under which disclosures and exploitation may take place is available at the NUST College Library of E & ME, Rawalpindi.

## **Acknowledgements**

All praise and glory to the Almighty God (the most glorified, the highest) who gave me the courage, patience, knowledge and ability to carry out this work and persevere and carry it out satisfactorily.

Without a doubt, he has given me the way and without HIS blessings, I cannot achieve anything.

I wish to express my sincere thanks to my advisor, Dr. Farhan, for raising the spirit of my spirit and for his continued support, motivation, dedication and valuable advice in my quest for knowledge. I am lucky to have a cooperation advisor and a kind mentor for my research.

With my advisor, I would like to thank all my dissertation committee: Dr. Muhammad Usman Akram, Dr. Arslan Shaukat of CEME and Fahad Alghazali, Dr. Nasir Mohyuddin, Dr. Asad of NESCOM for their cooperation and cautious suggestions.

My gratitude would be incomplete without thanking the greatest source of my strength, my wife.

Finally, I would like to express my gratitude to my colleagues and to the people who have encouraged and supported me during this period.

*Dedicated to my caring and loving wife: **Quratulain Ghaffoor**, whose tremendous motivation, cooperation and support guided me to accomplishment it.*

## **Abstract**

M0 core developed by ARM is a widely used general purpose processor. M0 core is a RISC based processor utilizing only 12000 logic gates in its small configuration. This makes it very feasible for low powered devices as the power ratings are really low. We can utilize M0 core as a co-processor providing communication services, crypto graphical services, malware detection services, wireless sensor nodes etc. ARM Cortex M0 core is a propriety processor and has a few limitations of its own. As a propriety core the details of its architectures are not published. Also interfacing of the memory (RAM, ROM) and UART needs to be done to use it for a specific application. In our work we comprehensively study the architecture of M0 core. We thoroughly investigated the Bus (AHB Lite) of M0 core in order to interface it with the RAM and ROM. We were able to successfully interface the Bus with the memories. Four Blocks of 8x2048 Bits RAMs are designed. We also interfaced the Peripheral bus (APB) for UART. Our experiment demonstrated successful transmission of data. For our experiment we used the Xilinx Spartan 3-E kit to port Cortex M0 processor on FPGA. Xilinx FPGA XC3S1600E with 33,000 logic gates are enough to port Cortex M0 soft core. In order to visualize results Digital Clock Manager (DCM) is used to scale down 50 MHz clock frequency to 5MHz.



## Table of Content:

Declaration.....	3
Language Correctness Certificate .....	4
Acknowledgements .....	6
Abstract.....	8
Table of Content: .....	9
List of Tables .....	13
Chapter 1: INTRODUCTION.....	14
1.1 Advantages of FPGAs .....	14
1.1.4 ARM series of processor .....	15
1.2 Problem Statement .....	16
1.3 Aims and Objectives.....	16
1.4 Contributions.....	16
1.5 Structure of Thesis .....	17
Chapter 2: LITERATURE REVIEW .....	18
2.1 Embedded Systems and FPGAs.....	18
2.2 Architecture of Microcontroller:.....	18
2.3 FPGA and soft cores of Processors .....	19
2.4 ARM Understanding Different Types of Processors .....	21
2.5 ARM Cortex M0.....	25
2.6 M0 advantages: .....	27
2.7 Applications: .....	28
Chapter 3: Hardware and Software Details .....	30
3.1 FPGA Spartan-3E as proto Board .....	30
3.2 Hardware overview.....	30
3.2.1 Slide switches .....	31
3.3 Clock.....	33
3.4 FPGA Configuration Options .....	34
3.5 Serial ports .....	36
3.6 FPGA programming.....	36
Chapter 4: Proposed Methodology .....	40
4.1 M0 processor .....	40
4.2 Slave .....	41

4.3 Slave select.....	41
4.4 Address decoding.....	42
4.5 APB Bus .....	42
4.6 ROM .....	42
4.6 SRAM.....	43
4.7 ABP Slave select .....	43
4.8 ABP Timer.....	43
4.9 ABP UART .....	44
Chapter 5: EXPERIMENTS AND RESULTS.....	45
5.1 Simulation .....	45
5.2 Implementation .....	48
Chapter 5: EXPERIMENTS AND RESULTS .....	52
5.1 Results.....	52
Chapter 6: CONCLUSION & FUTURE WORKS .....	53
6.1 Conclusion .....	53
6.2 Future Work .....	53
References .....	54

## List of figures

Figure 1-1 ARM Microcontroller Cores.....	15
Figure 2. 1 Basic microcontrollers Architecture.....	17
Figure 2. 2 Trade-off in processor designs.....	20
Figure 2. 3 ARM processor families Overview.....	21
Figure 2.4 Instruction table of M family.....	23
Figure 2.5 Cortex M series processor compatibility.....	23
Figure 2.6 Cortex M0 simplified block diagram.....	24
Figure 2.7 Cortex M0 simple systems.....	25
Figure 3.1: Xilinx Spartan FPGA Board .....	29
Figure 3.2: Four Slide Switches.....	30
Figure 3.3: UCF for Slide Switches.....	30
Figure 3-4: Rotary push buttons and four push buttons.....	30
Figure 3.5: Push buttons must have internal pull down resistor in FPGA.....	31
Figure 3.6: UCF for Push-Button Switches.....	31
Figure 3.7: Pull up resistor for push button switch in FPGA.....	31
Figure 3.8: UCF file for Rotary Push-Button Switch.....	31
Figure 3.9: Eight surface mount LEDs.....	32
Figure 3.10: UCFs for Eight Discrete LEDs.....	32
Figure 3.11: Available Clock.....	33
Figure 3.12: UCF for Clock Sources.....	33
Figure 3.13: Configuration settings of Spartan-3E Starter Kit forFPGA.....	34
Figure 3.14: Detailed Configuration Options.....	34
Figure 3.15: RS-232 Serial Ports.....	35
Figure 3.16: UCF for DCE RS-232 Serial Port .....	35
Figure 3.17: iMPACT open up option.....	36
Figure 3.18: New configuration file for FPGA.....	36
Figure 3.19: iMPACT Programming Succeeded .....	36
Figure 3.20: Set Properties for Bit stream Generator.....	37
Figure 3.21: PROM, ACE, or JTAG File generation.....	37
Figure 3.22: Click PROM File Formatter.....	38
Figure 3.23: Selection of XCF04S Platform Flash PROM.....	38
Figure 3.24: PROM File Formatter Succeeded.....	38

Figure 4.1 Block diagram of M0.....	39
Figure 4.2: 'cmsdk_ahb_default_slave.v' model.....	40
Figure 4.3: 'cmsdk_ahb_slave_mux.v' model .....	40
Figure 4.4 cmsdk_ahb_rom model.....	41
Figure 4.5 cmsdk_ahb_to_sram.v model.....	42
Figure 4.6 cmsdk_apb_slave_mux.v module.....	42
Figure 4.7 block diagram of timer.....	43
Figure 4.8 block diagram of ABP UART.....	44
Figure 5.1 Keil Program.....	45
Figure 5.2 FPGA settings.....	46
Figure 5.3: Path for image.hex file in cmsdk_ahb_rom.v .....	46
Figure 5.4: Path for image.hex file in cmsdk_fpga_rom.v. ....	47
Figure 5.5: Path for BRAM files in cmsdk_fpga_rom.v.....	47
Figure 5.6: Simulation of M0 core processor.....	48
Figure 5.7: Path for image.hex file in cmsdk_ahb_rom.v.....	48
Figure 5.8: BRAM settings.....	49
Figure 5.9: 8x2048 size BRAM.....	49
Figure 5.10: BRAM file path.....	49
Figure 5.11: DCM clock frequency settings.....	50
Figure 5.12: UCF of M0 core .....	50
Figure 5.13: top level M0 core implementation.....	51
Figure 5.14: M0 core implementation on FPGA.....	51
Figure 5.1: Result data .....	52

## List of Tables

Table 2.1: Typical components in microcontrollers.....	19
Table 3.1 Clocks, Global buffers and DCM.....	33

## Chapter 1: INTRODUCTION

In the age of technology, to be competitive, the product must have dynamic and unexpected changes. In this work, field programmable gate matrices (FPGA), in the age of technology, are enough to show the dynamic and unexpected changes of the product adapted to meet these requirements. They are economical, adaptable, powerful and economical and can be configured in a hardware using HDL language.

Traditionally, the designer can perform has following three options to perform hardware control platforms:

- Digital signal processors (DSP)
- Application specific integrated circuits (ASIC)
- Field programmable Gate arrays (FPGA).

ASICs provide optimal performance because they are designed to meet the requirements of the application. However, the budgets for ASIC-based solutions are maximized, since the number of production machines is usually relatively small due to the small number of units compared to the products it produces. On the other hand, DSP-based solutions are economical, but their processing speed is not sufficient due to software execution.

### 1.1 Advantages of FPGAs

The development of FPGA technology has grown in recent years in terms of usable resources, processing speed, number of access points, retail prices and energy consumption. Advanced FPGAs are therefore a reliable alternative to common microcontrollers and ASICs for applications. Recent FPGA-based systems also include many advantages of DSP and ASIC. That is, high flexibility, reuse capacity, rapid development cycles, moderate costs and easy updating (due to the use of the abstract hardware description language (HDL)), and feature expansion (provided FPGA resources are not depleted) in addition, current FPGAs can be integrated as software processors. Therefore, an FPGA can have typical processor capabilities.

#### 1.1.1 FPGAs as ARM microprocessors:

Primary use of an FPGA is for computationally intensive, parallel processing tasks and high-speed, however the ARM microprocessor is used mainly due to its versatility many manufacturer used it as the core of their applications i.e ZYNQ family of devices. ARM because of their wide use as a processor variations of operating systems are available.

### 1.1.2 Embedded processors on FPGA soft-core vs hardcore

In integrated systems, field programmable gate array (FPGA) is a very effective solution due to their ability to reconfigure, scale and be low cost. Due to the configurable logical capability of the FPGA, designers have been forced to integrate software processors into the FPGA for use in various peripherals. To this end, many software and hardware cores have been developed for different software and hardware.

### 1.1.3 Linux soft cores

The cores of the arm processor are called cortex. The Cortex-M FPGA solutions provided by Altera (soft cores of M1 is provided) also Actel provide the same package, while the Cortex-M3 is in available as IC package. Xilinx has provides no paratical solution to date. ARM has released a low-cost and low-cost solution of the M0 processor that can be programmed in an FPGA or used as ASIC applications. XILINX does not provide a soft MB core.

### 1.1.4 ARM series of processor

The Cortex series of processors as shown in Figure 1, includes cores of economical microcontroller solutions and state-of-the-art processors, capable of handling huge tasks and advanced operating systems. Other processors of the same family are ARM7 series, ARM9 series and ARM11 series. The specialized SecurCore™ series is also primarily intended for security as well as cryptography applications.

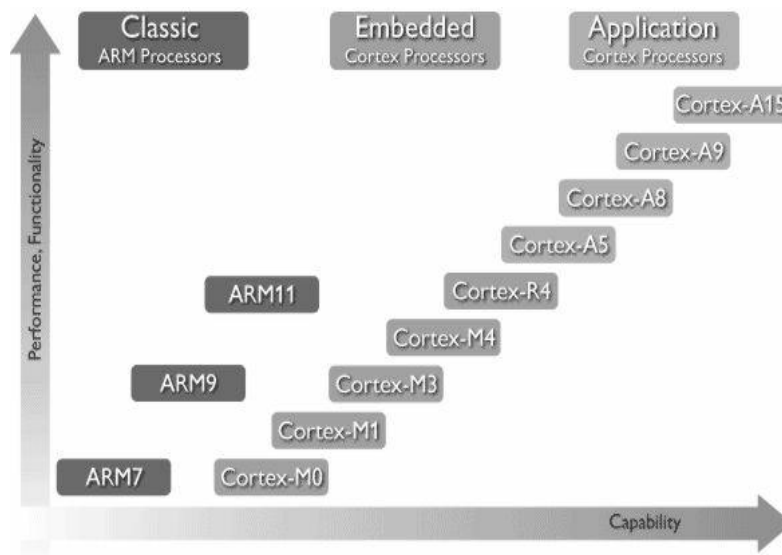


Figure 1-1 ARM Microcontroller Cores

### 1.1.5 Advantages of ARM processor family:

ARM cortex processors mainly used for high performance applications with real time and embedded operating systems. These processors are low cost, low power, fixed latency interrupt

handling processors. Cortex Mo is alternative for 8 bit microcontrollers as they have high processing capacity.

### **1.1.6 Motivation**

In the field of embedded systems, a hardware which can be configured and changed even after installation and deployment is much needed to cope with the ever increasing and changing demand of market. The motivation behind this thesis is to develop indigenous hardware platform for various industrial and military configurable systems and devices with low power, high interrupt latency, good efficiency and low cost.

This design or platform can be used as IOT, Encryption controller, intrusion detection controller and as communication control (for implementing various communication protocols).

The most attractive feature of this platform is its speed, low cost and run time configuration feature.

## **1.2 Problem Statement**

To develop an indigenous FPGA based hardware configurable solution for various industrial and military applications with all features of ARM processor in form of soft-core of ARM family.

## **1.3 Aims and Objectives**

The main goals of the investigation are the following:

- It can be used as indigenous IOTs (Internet of Things)<sup>1</sup>.
- It can also be used as parallel intrusion detection processor like PHANTOM<sup>2</sup>.
- Can be used as platform for computer vision applications<sup>3</sup>.
- FPGA based multiprocessor systems<sup>4</sup>

## **1.4 Contributions**

- Comparison of working of soft-core and hardcore ARM processors implementations.
- Selection of FPGA instead of ASIC and DSP processors.
- FPGA Spartan 3 board all functionalities and features.



## 1.5 Structure of Thesis

This work is structured are the following:

**Chapter 2:** States the literature review of related work.

**Chapter 3:** Give details about the Hardware used in this thesis.

**Chapter 4:** Consists of the proposed methodology in detail. It includes: coding, system configuration, simulations, results and analysis.

**Chapter 5:** Experiments and results are discussed in detail with all desired figures and tables.

**Chapter 6:** Closes the thesis and reveals the future prospect of this study.

## Chapter 2: LITERATURE REVIEW

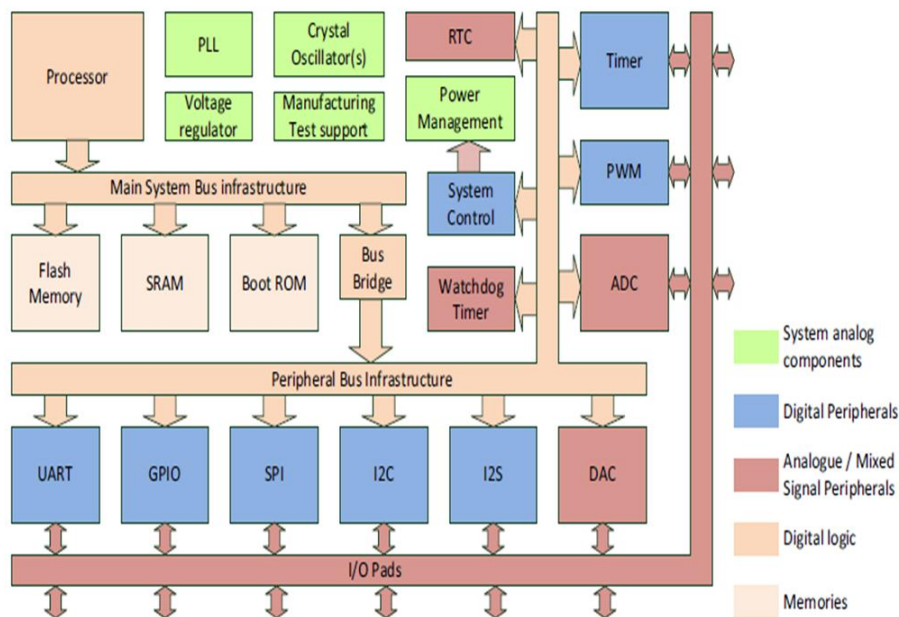
### 2.1 Embedded Systems and FPGAs

In embedded system industry FPGAs use is very crucial and vital. FPGA offers possibility of a wide range of electronic gates and IC to be incorporated.

Processors main use in electronic like cell phones, computers, television, machines used for washing purposes and in various cards. And even in simple devices contain processor in them like remote control of your radio, which is actually fabricated in an IC called microcontroller .In modern microcontrollers, memory system and other peripherals are also incorporated.

### 2.2 Architecture of Microcontroller:

Typical elements of a microcontroller are as under mentioned



**Figure 2. 1 Basic microcontrollers Architecture**

Items	Descriptions
Read Only memory (ROM)	It is a nonvolatile memory used for storage of program.

Flash	Flash is special type of ROM that can be reprogrammed several times, typically used for storing program codes.
Static Random Access memory (SRAM)	Used for data storage.it is a form of volatile memory.
Phase Lock Loop (PLL)	Its primary used for generation of variable clock frequency from a reference clock.
Real Time Clock (RTC)	It is simply a low power timer to calculate seconds and works with a low power oscillator.
General purpose Input/output (GPIO)	It is a sub module with parallel data interfacing and used for controlling external modules and to read back external signal status.
Universal Asynchronous Receiver and Transmitter (UART)	It's a sub module used to cater serial transfer of data to external interface.
Inter Integrated Circuit (I <sup>2</sup> C)	It is an interface used for serial data transfer; its data transfer rate is higher than serial.
Serial Peripheral Interface (SPI)	It's commonly used for serial communication with other off chip deices.
Inter IC sound (I <sup>2</sup> S)	It's commonly used for serial communication for audio signal.
Pulse Width Modulator (PWM)	It is a module to output configurable square signal of flexible time duration.
Analog to Digital Convertor (ADC)	Its primary purpose is to transform analog signal to digital format.
Digital to analog convertor (DAC)	Its primary function is to convert digital signal form to analog signals.
Watch Dog Timer	It's a programmable time device to ensure that processor is running as program. In enable state, the timer should be updated by the programmer after certain time interval. When the program is crashed that timer gets expired and on the basis of this critical system interrupt or system reset is triggered.

Table 2.1: Typical components in microcontrollers

### 2.3 FPGA and soft cores of Processors

FPGA facilitated the incorporation of soft processor cores. Processors as soft core can be considered equivalent to "computer on a chip" or microcontroller. Core conglomerates a

Central processing Unit, memory and peripherals on a single chip. Custom interfaces can also be accessed beyond the real standard integrated FPGA of integrated chip.

Many types of microcontrollers are available with many types of processors, peripherals and memory sizes in them which can be marketed in various packages in market. Processors are used in various applications so that using software system can be controlled and various features can be developed.

The ARM cortex M0 and cortex M0+ processors mainly used as SOCs, microcontrollers, ASIC and ASSPs. And sometimes used in many subsystems.

Available reduced instructions set computer (RISC) architectures in market are:

1. ALTERA launched CPUs like NIOS and NIOS II.
2. Gaisler Research launched CPU like LEON2 and LEON3.

Microcontroller systems are now becoming increasingly advanced and demanded and can achieve greater performance to fulfill the requirements of more suitable functions and operations. Sometimes, they are costly; therefore, microcontroller systems are determined as designs of a chip only detected in high-performance 32-bit processors, which dominate the current market. Variety of peripheral are evolved after requirement rose from software development and pressure of costly standardization.

Industry designed ARM M series of processors to meet above challenges, now over 200 companies are producing these chips from standard core to meet all applications like sensor nodes to radio communication for IOTs.

With the access of microcontrollers to more and more people, the software will be available for you to see. real-time operating systems (RTOS) has quickly become a recommended industry exercise and their use gaining importance for software engineering. The combination of these components in the industry presented a problem for those who were developers, who were in charge of the industry to reduce system costs and marketing success. Therefore, the architectures of Cortex-M processors coupled with CMSIS standard is basis of hardware and software standard.

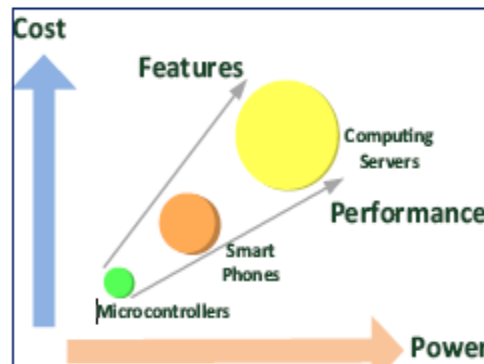
### **2.3.1 Applications of soft-core processors:**

Soft-core processors have many applications:

1. As Embedded machine learning processor for intrusion detection [5].
2. As a secure processor called PHANTOM in which practical oblivious computation is performed [6].
3. Design and development of a security coprocessor based on a chip system (SOC) and a program protection mechanism for wireless sensor nodes (WSN).[7]

## 2.4 ARM Understanding Different Types of Processors

Industry has designed many processors for various applications. ARM also designed different processor series for different applications. For server, a processor with high data rate is required. For battery applications, performance can be compromised in order to achieve low power. For high performance, processor needs to have more transistors as this is rule of physics. As frequency increases so as the power dissipation by the processor.



**Figure 2. 2 Trade-off in processor designs**

Increasing the size of silicon results in increases in production cost (Figure 2.2). Hence different processor have been designed for different applications. Also, chip designer's needs to select suitable processor for their application. Luckily, many vendors provide different processors for different applications. ARM also provides processors as per designer need.

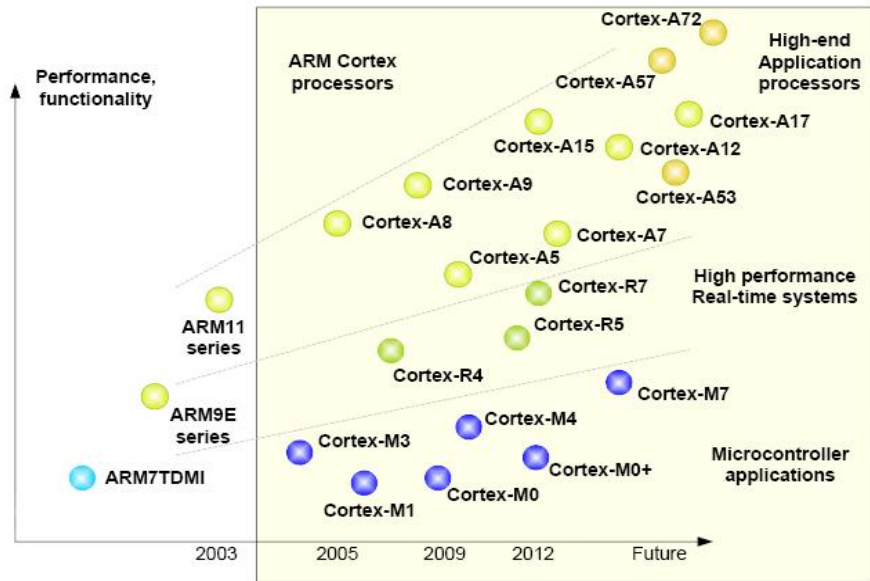
### 2.4.1 Overview of the ARM Processor Families

ARM has designed many processors for several applications as shown in the Figure 2.3. Lets see what ARM has to offer for designers.

ARM designed the processor since long. Over 20 years of their experience ARM has provided 32 Bits processors but recently they have designed processor for mixed architecture of 32 Bits and 64 Bits.

ARM7 processor is the first series of processors marketed for the designers. It is high efficiency and high code density allows the designers to use state of the art operating system. They are use

frequently in next generation mobile phones. ARM after great response from designers continue design new series like ARM9/9E and ARM11 family of processor.



**Figure 2. 3 ARM processor families Overview**

**2.4.1.1 Cortex-A series processors:**

Some application requires high performance so that they support advance operating system. Such processors have longer processing channels and can operate at high clock frequency.(1 GHz or greater). For functionality MMU (Memory management unit) supports virtual memory addressing. These are used in cell phones, mobile computing and energy efficient processor. For fast response ARM has designed R series of processor.

**2.4.1.2 Cortex-R series processors:**

For fast response Cortex-R series is launched. Their clock frequency is less than A series (between 500 MHz to 1GHz). They have tightly coupled memories to improve response time. Some are provided with additional features like ECC (Error correction code) for reliability.

Their application includes disk drive controllers, wireless controllers/ modems, automotive controllers and in industrial controllers. They consume a lot of power and have complex architecture. For integrated products ARM launched another M series of processor.

### 2.4.1.2 Cortex-M series Processors:

M series is used for application where less processing is required at the cost of low power. Their pipeline is short (2 stage for M0 and 3 stage for M3, M4), however, M7 has pipeline of 6 steps because of greater performance requirements; however, it is much smaller than pipeline of high end application processors. Due to tube optimization and power consumption in the application, the maximum clock rates of them are slower than those of the R and A series processors. However, for low processing application this is not a issue.

Cortex-M series processors can handle very fast interrupts response. To achieve this goal they are equipped with special module called nested vector interrupt controller (NVIC). This module has very handy features of interrupt handling. They are easy to use and can be programmed in C language. Because of features like low power, high performance and user friendly Cortex-M series processors are used in sensors, chips used for mixed-signal ASIC / ASSP, and are even used as a controller in some of the complex use processor / SoC product subsystems.

Instruction set is compared in figure 2.4, M0, M0+and M1 support small instruction set (56 instructions). Most are 16 Bits as they provide good code density. M0 and M0+ instructions are simple in nature and can perform complex task easily. For state of the art operating system M3 processor is used as it has 32 bit of instruction support and can support following:

- More addressing modes of memory
- In the 32-bit instructions Larger immediate data
- Longer branch and conditional branch ranges
- Additional branch instructions
- Hardware divide instructions
- Multiply accumulate (MAC) instructions
- Bit field processing instructions
- Saturation adjustment instructions

M3 can handle complex data quickly. With same code size as M0 and M0+ because it uses same instructions for same task. 32 bit instructions can do same task efficiency then 16 bit instructions. For DSP applications that involves filtering and signal transformation. M3 also equipped with SIMD (Single Instruction multiple data). Data path of processor is also reduce to accommodate that.

VSEL	VCVTA	VCVTN	VCVTF	VCVTM	VMAXNM	VMINNM	Cortex-M7 FPU (single and double precision floating point)				
VRINTR	VRINTA	VRINTN	VRINTP	VRINTM	VRINTX	VRINTY					
VABS	VADD	VCMP	VCMP	VCVT	VCVTR	VDIV	VLDM	VLDR	Cortex-M4 FPU (single precision floating point)		
VMMLA	VMML5	VMOV	VMRS	VMSR	VMUL	VNEG	VMMLA	VMML5	VFNMA		
VNMUL	VPOP	VPSH	VSQRT	VSTM	VSTR	VSUB	VFMA	VFMS	VFNMS		
QDADD	QADD	QADD16	QADD8	SADD16	SADD8	UADD16	UADD8	UHADD16	UHADD8		
QDSUB	QSUB	QSUB16	QSUB8	SSUB16	SSUB8	USUB16	USUB8	UHSUB16	UHSUB8		
ADC	ADD	ADR	AND	ASR	B	BIC	SHADD16	SHADD8	SHSUB16		
BFC	BFI	CLZ	CDP	CLREX	CMN	CMP	PKH	SEL	SHSUB8		
CBNZ	CBZ	DBG	EOR	LDR	LDRH	LDRB	LDRD	SMULTT	SMULTB		
LDMA	LDMDB	LDRT	LDRHT	LDRBT	LDRSH	LDRSB	SMULBT	SMULBB	UQSUB16		
LDRSBT	LDRSHT	LDREX	LDREXB	LDREXH	LSL	LSR	SMLATT	SMLATB	UQADD8		
LDC	MCR	MRC	MCR	MRR	PLD	PLI	SMLABT	SMLABB	UQSUB8		
MOV	MOVW	MOVVT	MUL	MVN	MLS	MLA	SMLALTT	SMLALTB	SMMUL		
NOP	PUSH	POP	ORR	ORN	PLDW	RBIT	SMLALBT	SMLALBB	SMLLWT		
ADC	ADD	ADR	BKPT	BLX	BIC	REV	REV16	REVSH	ROR		
AND	ASR	B	BX	CP5	CMN	RSB	RRX	SBC	SEV		
BL	MRS	MSR				SUB	STC	UBFX	SBFX		
DSB	DMB	ISB				STR	STRD	UDIV	SDIV		
CMP	EOR	LDR	LDRH	LDRB	LDM	STRB	STRH	UMULL	SMULL		
LDRSH	LDRSB	LSL	LSB	MOV	NOP	STMIA	STMDB	LIMLAL	SMLAL		
REV	REV16	REVSH	MUL	MVN	ORR	STREX	STREXB	UXTB	SXTB		
PUSH	POP	ROR	RSB	SEV	SVC	STRHX	STRT	USAT	SSAT		
SBC	STR	STRB	STRB	STM	SUB	STRHT	STRBT	UXTH	SXTH		
SXTB	UXTB	SXTH	UXTH	TST	YIELD	TBB	TBH	WFI	WFE		
WFE	WFI					TST	TEQ	YIELD	IT		
Cortex-M0/M0+/M1 (ARMv6-M)				Cortex-M3 (ARMv7-M)				Cortex-M4 (ARMv7E-M)			
16-bit instructions				32-bit instructions							

Figure 2.4 Instruction table of M family

### 2.4.1.3 Software Portability of Cortex®-M Processors:

M0, M0+ and M1 series processor are based on ARMv6. M3, M4 and M7 are based on ARMv7 architecture as shown in the figure 2.5 for instruction set support.

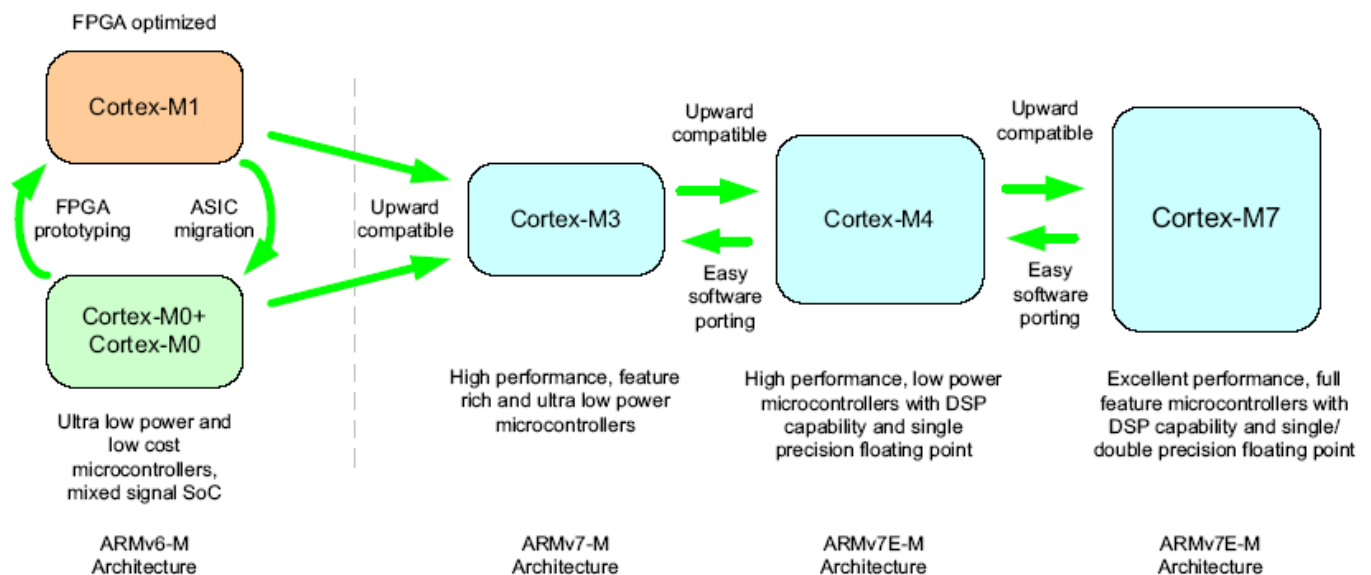


Figure 2.5 Cortex M series processor compatibility



## 2.5 ARM Cortex M0

It is 32 Bit processor; it means that internal registers banks, data path and bus interfaced are all 32 Bits. It has single bus interface means its architecture is Von Neumann type. It has three stage pipeline (Fetch, decode and execute). Most of the instructions are 16 bit, however only few are 32 bits. It can support optional 32x32 bit multiplier. Address supports 4GB of memory interfacing. M0 bus interface is based on AHB-Lite protocol that can support 8, 16 and 32 bit data transfer. Protocol is pipelined and can support high operating frequency and peripheral can be connecting AHB-Lite with APB Bus Bridge. M0 supports 32 interrupts trough Nested Vector interrupt controller (NVIC). It supports Two sleep modes (normal sleep and deep sleep) are used for power saving.

### 2.5.1 Block Diagram:

Block diagram of M0 is shown as figure 2.6

M0 contains register for data storage, ALU and control logic. Three stages of pipeline for fetch decode and execute stage. Banks are of 32 bits. Some are special usage like PC while others are used as general purpose.

NVIC can handle upto 32 interrupts with functionality to compare priority between interrupts request and current interrupt priority level. In case of a interrupt, NVIC communicate with the processor to execute interrupt correctly.

There are 32 bits AHB-Lite bus interface, processor core, internal bus system, and data path. AHB-Lite bus

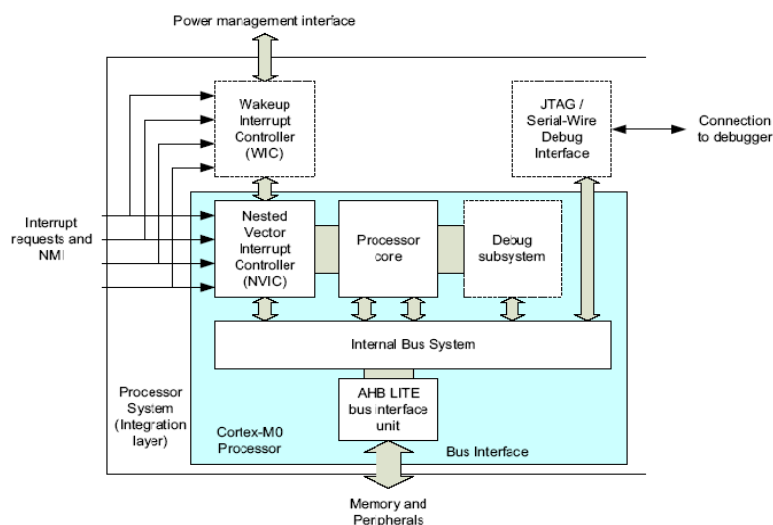


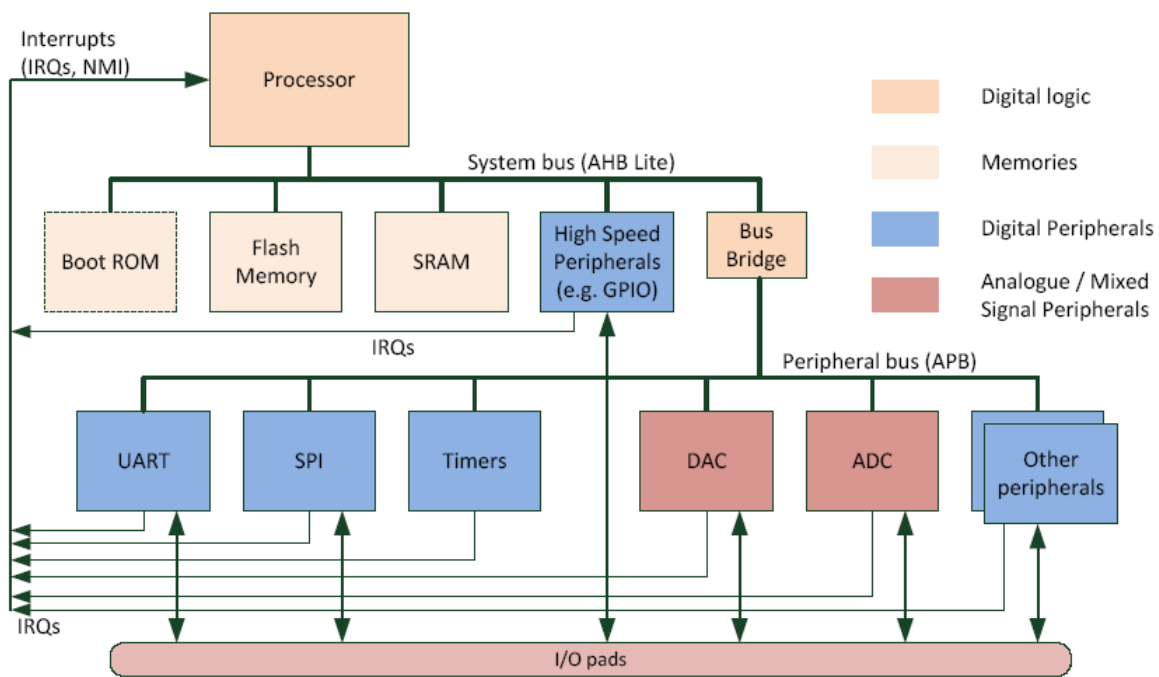
Figure 2.6 Cortex M0 simplified block diagram

### 2.5.2 System overview:

M0 processor inherently does not have any memory and peripherals. However, designers need them for their design, so M0 processors based IC have different addressing range, interrupts and memories. Normally M0 have following peripherals:

- Program code storage use ROM (Read only memory) . e.g flash memory
- SRAM (Static RAM) for data read write.
- Bus interfacing for various memories and processor joining.

M0 processor can be look like as shown in the Figure 2.7



**Figure 2.7 Cortex M0 simple systems**

M0 based design might have bus partition into two parts:

- RAM, ROM, Flash memory, SRAM, few other peripherals connected by the bus and a bus bridge.
- Peripherals attached by the help of peripheral bus, might have different operation frequency.

APB is connected with AHB-Lite bus through Bus Bridge. APB bus use is as follows:

- Low cost solution as APB bus is non-pipelined and is simpler to implement than AHB-Lite bus.

- Makes possible to interface others sub modules at different frequency of operations than main bus.
- Where large combinational blocks are required for logic implementation, they are attached with APB bus to avoid bottle neck for high operating frequency.

Also, interrupts, GPIO (General Purpose Input/Output) modules can be handled easily through APB bus.

### **2.5.3 ARMv6-M Architecture**

M0 cores are based on ARMv6-M Architecture. This refers to following two areas:

- ISA Model also called programmers model (software point of watching) and debug methodology (what debugger sees).
- Microarchitecture: An implementation detail like signal interfacing, execution sequence, pipelines etc. it is design specific. Like M0 has three stage of pipeline.

## **2.6 M0 advantages:**

### **2.6.1 Low Power and Energy Efficiency**

M0 is very energy efficient and consumes 12.5 uW/MHz with size of 90 nm. This is low power for 32 bit processor. ARM achieves it with less gate count, High efficiency and Low power and through logical cell enhancement.

#### **2.6.1.1 Low gate count**

Low gate count is achieved through careful design techniques at each stage. Every part was developed very carefully and helps in minimizing gate count to 12000 only. This is lower than even 16 bit processor keeping the performance almost double.

#### **2.6.1.2 High efficiency**

M0 has many low power features to use in battery powered applications. Two modes sleep and deep sleep are available. Sleep mode is invoked using instructions like WFE and WFI or sleep on exit. To save further power debug system can be turned off.

#### **2.6.1.3 Logic Cell Enhancements**

Ultra Low Leakage logical cell library developed by ARM support special state retention cells that holds information in case of system power failure.

#### **2.6.1.4 High density of code**

For 16 bit instructions M0 has very high density of code. Hence application can be accommodated into small memory.

#### **2.6.1.5 Low interrupt latency**

M0 can handle interrupt in 16 clock cycles. This involves stacking of registers in stack, so that ISR can work without any software overhead.

NVIC can handle interrupt prioritization and ISR starting address so that exact IRQ can be serviced. Interrupt response is much lower than 8 or 16 bit system when supported by good programming practices.

#### **2.6.1.6 User friendly**

M0 is very user friendly as most of the software is in C language. This helps in shorter code development and easy portability.

### **2.7 Applications:**

Mo offer high performance, low power and user friendly. Cortex-M series processors are carefully chosen from most of the microprocessors products. They are widely used in:

- Sensor nodes.
- Wireless communication chipset.
- Mixed signal application-specific standard product (ASSPs) and ASICs.
- In complex application systems as controller in subsystems.
- Security sensitive products as secure core processors e.g., SC000.which are used in SIM cards, electronic ID cards and banking/payment systems.

#### **2.7.1 Advantages**

Cortex Mo processors has number of key advantages

- Flexible interrupt management is provided using NVIC (Network Interrupt Controller).
- OS support features.
- Low power support like sleep modes.
- High code density.
- Integrated debugging.
- User friendly.

- High energy efficiency because of small size and better performance.
- For power management and boot sequence cortex ARM processor can be used as System Control Processor (SCP).

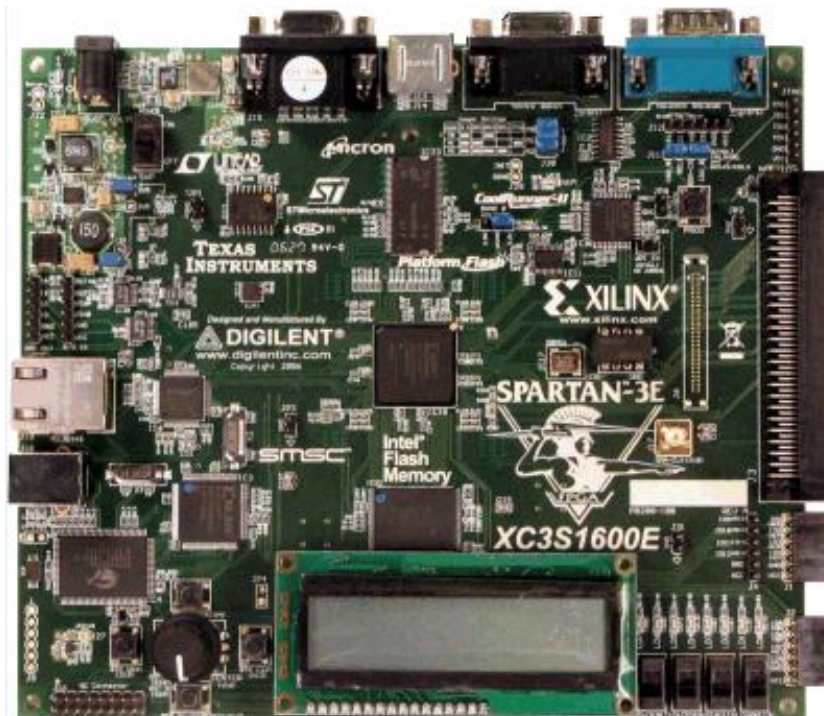
## Chapter 3: Hardware and Software Details

In order to implement our task we need FPGA with over 12k gates. After investigation and comparison with others options Spartan 3E board is selected.

### 3.1 FPGA Spartan-3E as proto Board

Spartan 3E starter FPGA development board is a digital platform for various embedded implementation. It support 16M bytes of SDRAM and 16M bytes of ROM. It has 50MHz crystal oscillator and base for secondary oscillator. USB2 power socket empowers all circuits, programming and data transfer modules. Also some other sub modules such as LCD screen, LEDs and switches, etc. LEDs can be used as event indicators. You can take advantage of Xilinx impact software such as Chipscope Pro, xmd, etc. In our case, this has helped to program and see the status of it easily.

The Spartan 3E starter kit is compatible with Xilinx FPGA S3E500-4. In total, 500 logic gates, 20 hardware multipliers, 10,500 logic cells, 73 Kbits of DRAM, 360Kbits of RAM, having 4 clock sources and clock frequency of 300MHz are maintained.



**Figure 3.1: Xilinx Spartan FPGA Board**

### 3.2 Hardware overview

In order to support design Xilinx board provides many interfaces. Some are as under:

### 3.2.1 Slide switches

There are four slide switches as shown in the figure 3.2. Located on lower right corner are designated as SW3, SW2, SW1 and SW0. When moved to up position a switch is connected to FPGA pin to high logic (3.3V). However, in down position will ground the switch. Switch does not have any active de bouncing circuit so it should be added by the programmer.

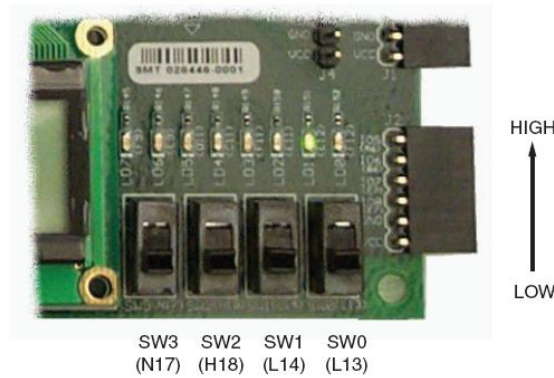


Figure 3.2: Four Slide Switches

UCF for slide switches is shown in the figure 3.3

```
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTTL | PULLUP ;
```

Figure 3.3: UCF for Slide Switches

### 3.2.2 Push Buttons

Four push buttons are shown in figure 3.4. Located in the lower left corner and labeled as Btn North, Btn East, Btn south and Btn west.

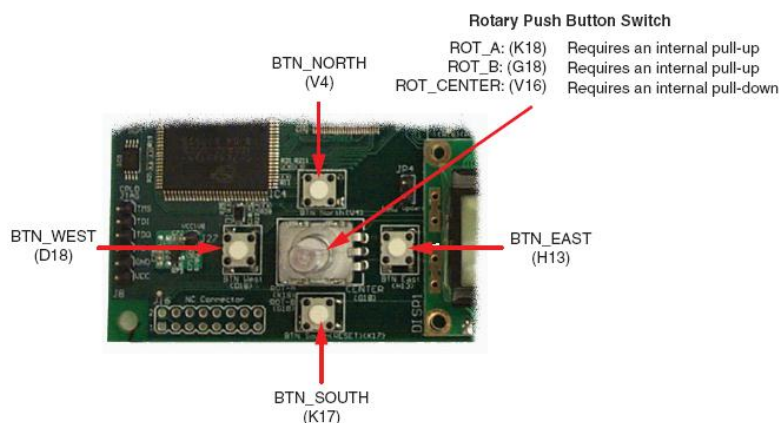
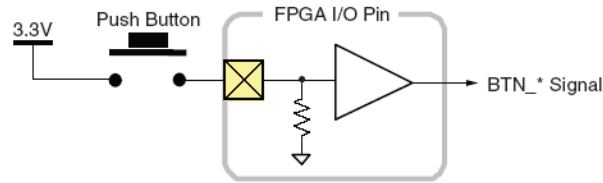


Figure 3-4: Rotary push buttons and four push buttons

In order to connected FPGA pin to high logic, push button needs to be pressed. The circuit with pull down resistor is shown in the figure 3.5.



**Figure 3.5: Push buttons must have internal pull down resistor in FPGA**

In UCF user must define the pull down settings along with I/O pin and I/O standards as shown in the figure 3.6

```

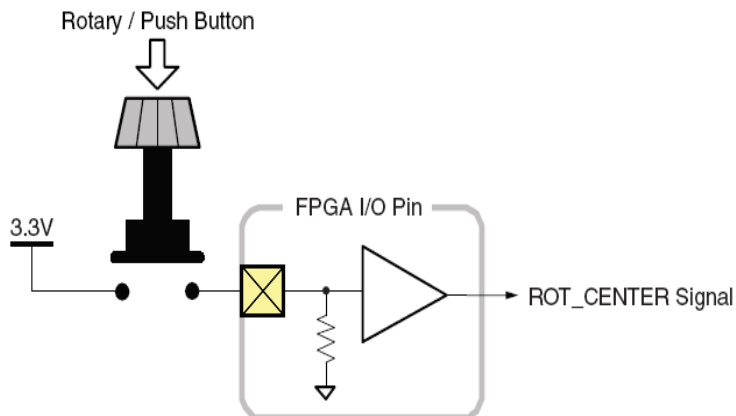
NET "BTN_EAST"   LOC = "H13" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "BTN_NORTH"  LOC = "V4"  | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "BTN_SOUTH"  LOC = "K17" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "BTN_WEST"   LOC = "D18" | IOSTANDARD = LVTTTL | PULLDOWN ;

```

**Figure 3.6: UCF for Push-Button Switches**

### 3.2.3 Rotary Push Button

It is located in the middle of push buttons switches with three outputs. These are Rot\_A, Rot\_B and Rot\_center. Rotary switches can act as dual functions, when shaft turns output values change and when pressed acts as a push button switch as shown in the figure 3.7.



**Figure 3.7: Pull up resistor for push button switch in FPGA**

It acts like two push buttons connected by central shaft. These can act as make before break. When shaft is stationary in these positions both switches are closed. UCF constraints for four push buttons is shown in figure 3.8

```

NET "ROT_A"      LOC = "K18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "ROT_B"      LOC = "G18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "ROT_CENTER" LOC = "V16" | IOSTANDARD = LVTTTL | PULLDOWN ;

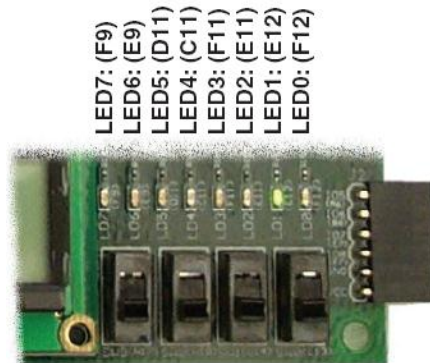
```

**Figure 3.8: UCF file for Rotary Push-Button Switch**



### 3.2.4 LEDs

Demo board is equipped with eight surface mount LEDs as shown in the figure 3.9. LEDs are labeled from LED0 to LED7.



**Figure 3.9: Eight surface mount LEDs**

LEDs are grounded from one side via current limiting resistors of 390Ω. To drive LED, FPGA pin must be high. UCF of LEDs is shown in the figure 3.10

```
NET "LED<7>" LOC = "F9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "E9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
```

**Figure 3.10: UCFs for Eight Discrete LEDs**

### 3.3 Clock

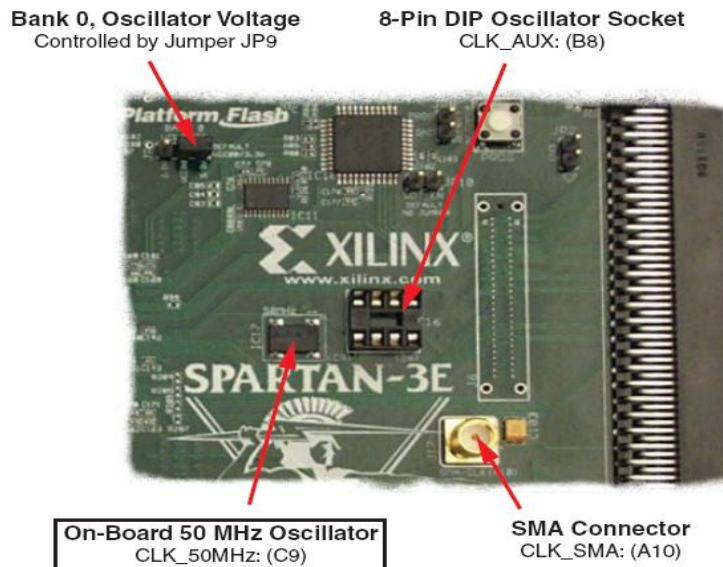
Board supports three clocks as a source to FPGA as shown in the figure 3.11.

- Clock oscillator of 50 MHz.
- SMA connector for external clock input. It is high speed connector.
- 8 pin socket for oscillator

Clocks are connected to Bank 0 of FPGA. Each pin is also connected to DCM. The settings are as shown in table 3.1

Clock Input	FPGA Pin	Global Buffer	Associated DCM
CLK_50MHZ	C9	GCLK10	DCM_X0Y1
CLK_AUX	B8	GCLK8	DCM_X0Y1
CLK_SMA	A10	GCLK7	DCM_X1Y1

**Table 3.1 Clocks, Global buffers and DCM**



**Figure 3.11: Available Clock**

### 3.3.1 50 MHz oscillator

Onboard oscillator frequency is about 50 MHz with 40% to 50% duty cycle. It is accurate to  $\pm 2500$  Hz.

### 3.3.2 Oscillator socket

There is an eight-pin socket for oscillator. It is normally used when operating frequency is greater than 50 MHz. Also we can use DCM for other frequency generation.

### 3.3.3 SMA clock

SMA connector is used for external clock provision. For external device it is single-ended clock.

UCF files for clock generation are shown in figure 3.12

```
NET "CLK_50MHZ" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
NET "CLK_SMA"   LOC = "A10" | IOSTANDARD = LVCMOS33 ;
NET "CLK_AUX"   LOC = "B8"  | IOSTANDARD = LVCMOS33 ;
```

**Figure 3.12: UCF for Clock Sources**

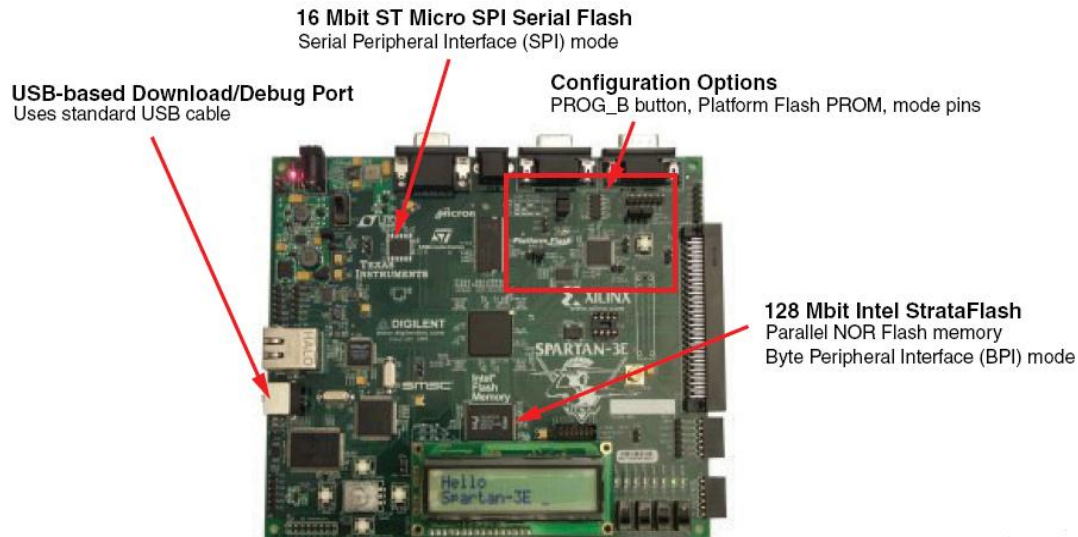
## 3.4 FPGA Configuration Options

Board supports many configuration options

- Code downloading via JTAG or on USB. This combination also programs board Flash PROM and XC2C64A CPLD.
- Code downloading to 4M-bit platform PROM, FPGA can be programmed from image stored in it using master serial

- Code downloading to 16M bit PROM; FPGA can be programmed from image stored in it using SPI.
- Code downloading to 128M bit flash, FPGA can be programmed from image stored in it using BPI UP or BPI down.

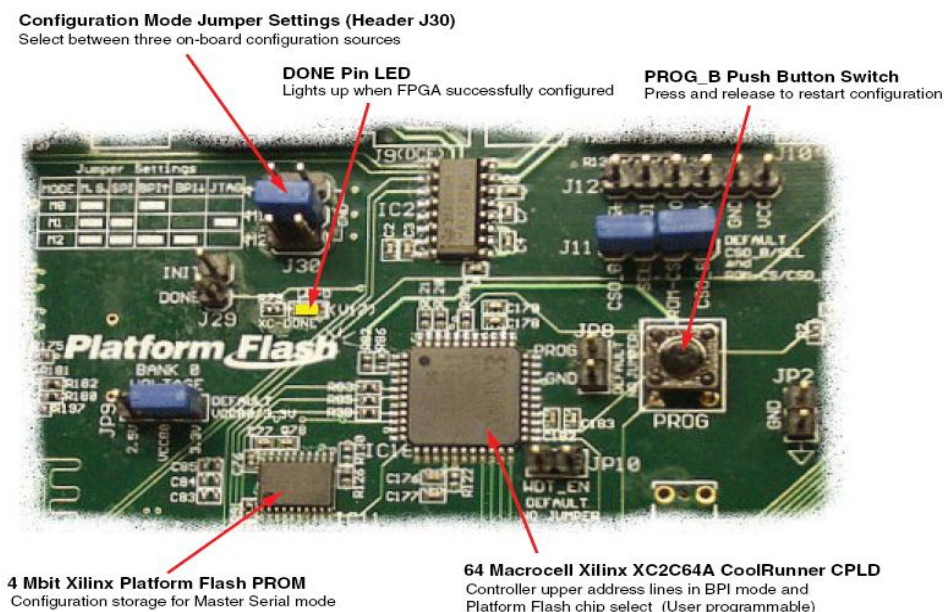
Figure 3.13 shows above described options in detail



**Figure 3.13: Configuration settings of Spartan-3E Starter Kit for FPGA**

### 3.4.1 Programming push button

This button is shown in figure 3.14 that force FPGA to reconfigure from selected configuration memory source. This can be done by using press and release of button.



**Figure 3.14: Detailed Configuration Options**

### 3.5 Serial ports

There are two serial ports, DB9 female connector and DTE male connector. DCE connects directly to PC via standard serial cable. DTE connector connects madams, printers etc. Both connectors are shown in figure 3.15

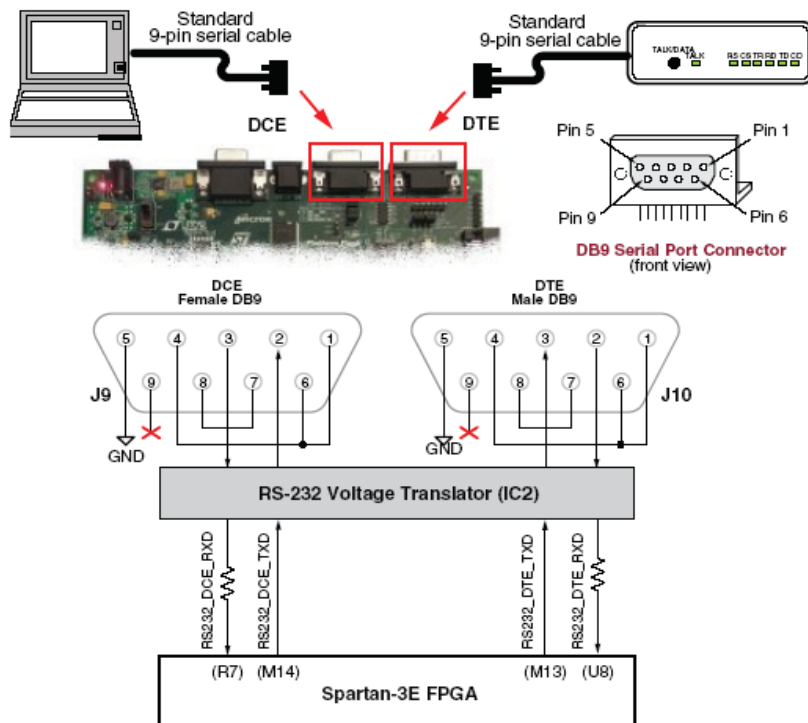


Figure 3.15: RS-232 Serial Ports

FPGA connects between two DB9 connectors. Its output data on LVTTTL or LVCMOS levels which converts the RS-232 voltage level. Figure 3.16 shows UCF for DTE and DCE ports.

```
NET "RS232_DTE_RXD" LOC = "U8" | IOSTANDARD = LVTTTL ;
NET "RS232_DTE_TXD" LOC = "M13" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = SLOW ;
```

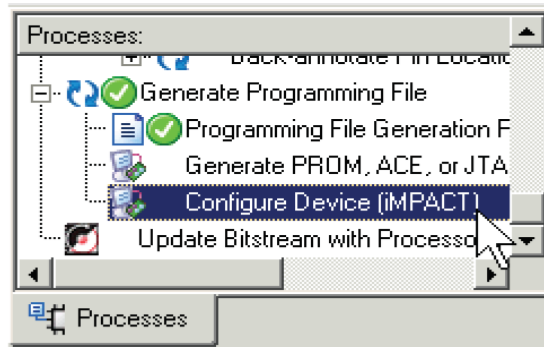
UCF Location Constraints for DTE RS-232 Serial Port

```
NET "RS232_DCE_RXD" LOC = "R7" | IOSTANDARD = LVTTTL ;
NET "RS232_DCE_TXD" LOC = "M14" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = SLOW ;
```

Figure 3.16: UCF for DCE RS-232 Serial Port

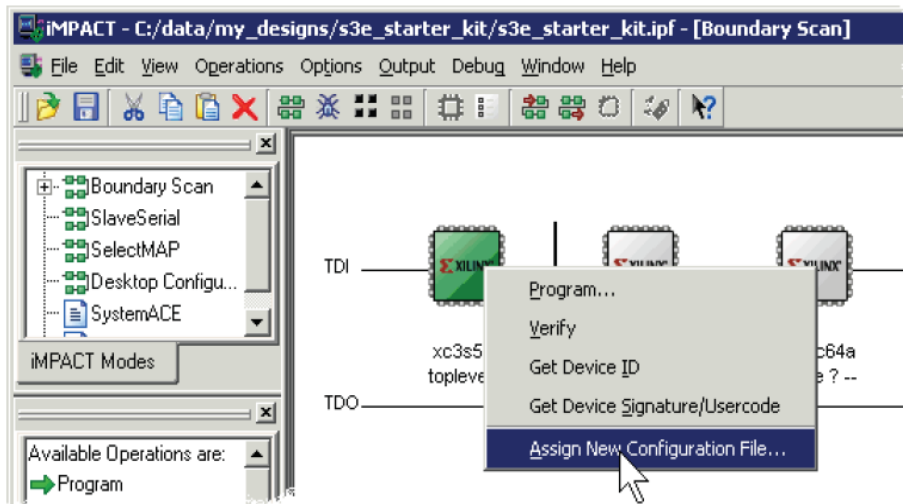
### 3.6 FPGA programming

FPGA can be programmed using USB provided with kit. Attach USB and iMPACT programming software can be used directly to program FPGA. Programming options for parallel or serial PROM is not supported. As USB is connected a green LED turns on shows good connection. iMPACT can be directly launched from ISE project navigator as sown in the figure 3.17



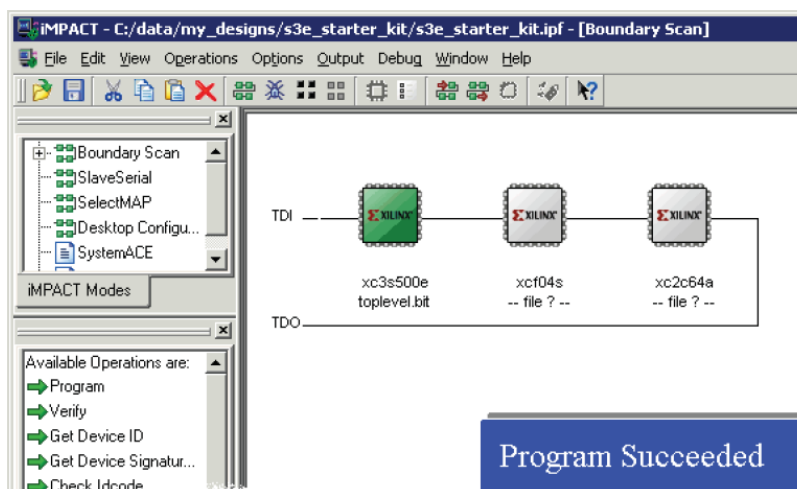
**Figure 3.17: iMPACT open up option**

As board is connected software recognizes three devices in the chain. To select FPGA, right click on it and assign new configuration file to it. Select file to program the device as shown in the figure 3.18



**Figure 3.18: New configuration file for FPGA**

When programming successful, FPGA application starts execution and DONE pin LED glows. The screen appears as shown in the figure 3.19

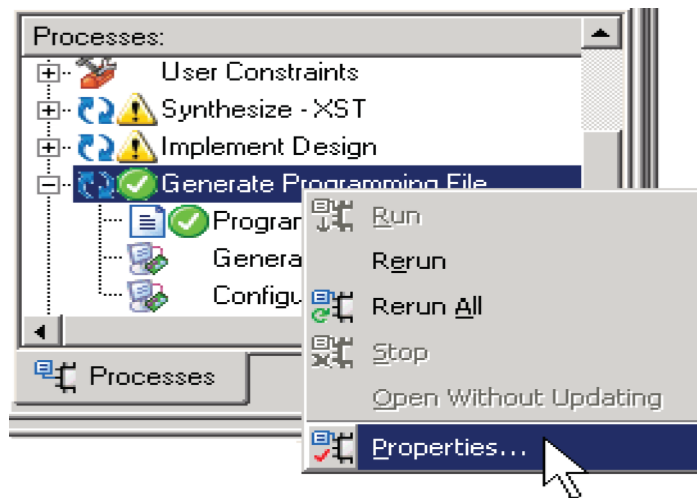


**Figure 3.19: iMPACT Programming Succeeded**

### 3.6.1 Generating Bit stream file

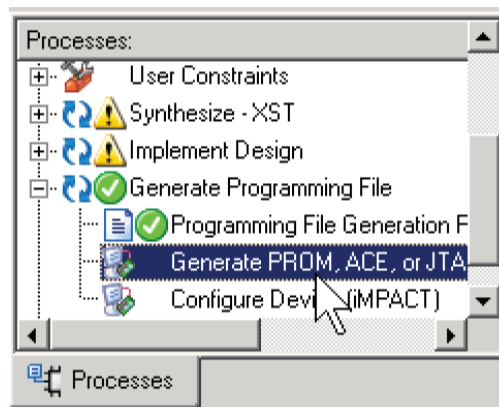
Bit stream is used for PROM programming. FPGA provides external clock to load external PROM. Oscillator starts from lower settings 1.5 MHz. PROM support higher frequency so this frequency can be increased. XCF04S flash support 25MHz.

Right click Generator programming file as shown in the figure 3.20



**Figure 3.20: Set Properties for Bit stream Generator**

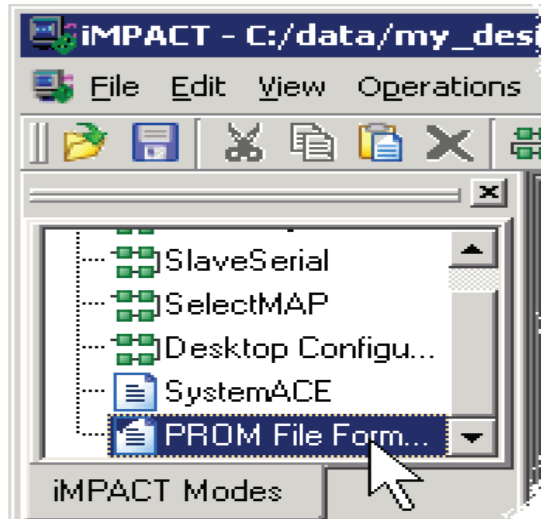
In properties option increase clock to 25MHz. click on generate programming file. When file is generated, use option as shown in the figure 3.21



**Figure 3.21: PROM, ACE, or JTAG File generation**

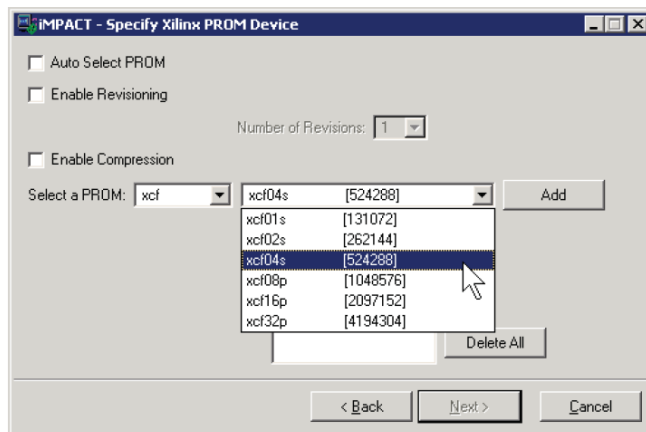
As iMPACT starts click on PROM file formatter as shown in the figure 3.22





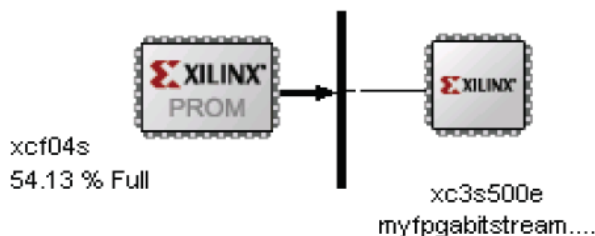
**Figure 3.22: Click PROM File Formatter**

Here target is Xilinx PROM. Select from any format. MCS (Intel Hex format) is popular. Enter path for file storage. Select XCF04S PROM from the options to program it  
 Kit has XCF04S Platform Flash PROM. Select XCF04s from list as shown in the figure 3.23



**Figure 3.23: Selection of XCF04S Platform Flash PROM**

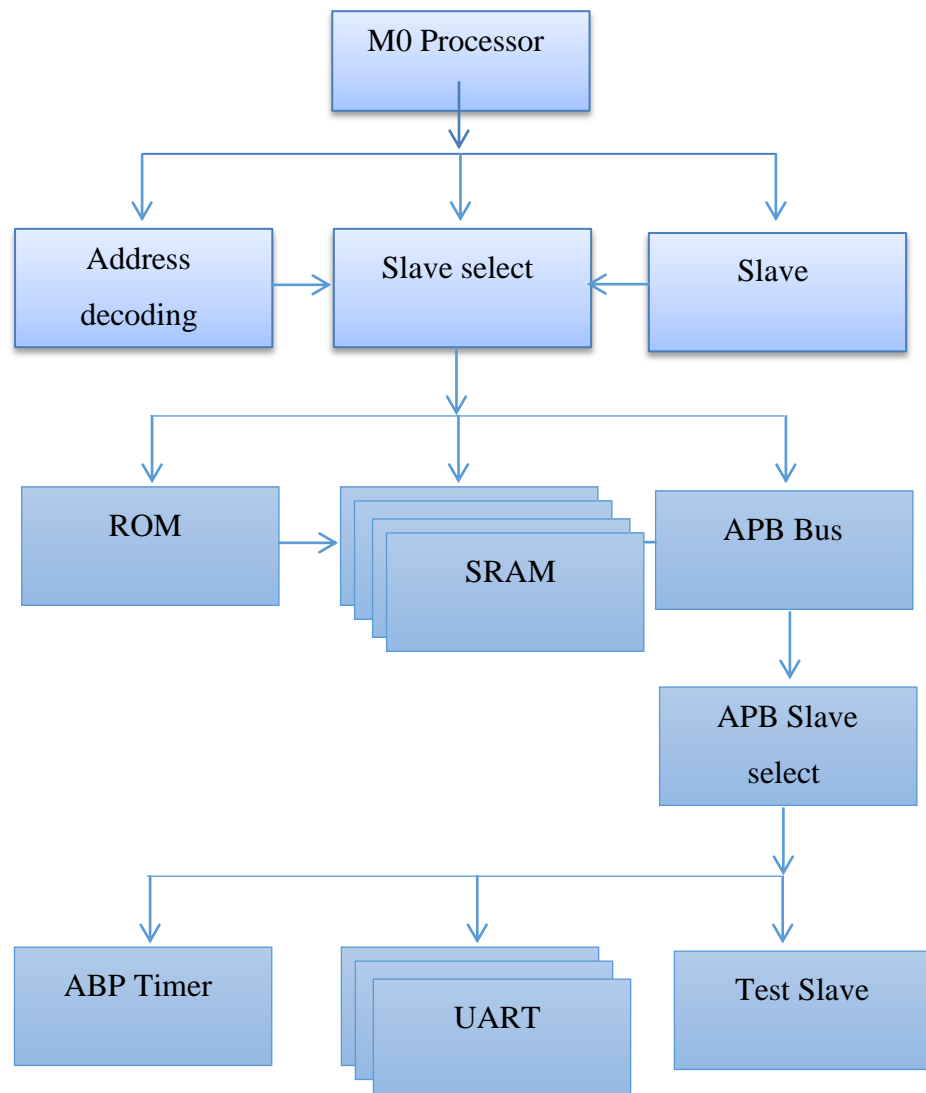
Next step is to format the PROM and after that select bit stream file. Presses continue. PROM file thus created is as shown in the figure 3.24



**Figure 3.24: PROM File Formatter Succeeded**

## Chapter 4: Proposed Methodology

As already discussed in Figure 2.7 that M0 processor is connected with others modules through AHB-Lite bus (Advanced high speed bus) and APB (Advanced Peripherals Bus). So, in order to make M0 core we need files in which settings and configuration of M0 are included. Also we need files containing information about AHB and APB bus, clock settings modules. These are shown in figure 4.1. We are going to discuss this one by one.



**Figure 4.1 Block diagram of M0**

### 4.1 M0 processor

In project we have use files like 'cmsdk\_ahb\_memory\_models\_defs.v'. This file defines memory models that we are using for our M0 core. In total M0 support five memory models.

- None.



- Behavioral ROM model, using behavioral SRAM with write disabled.
- SRAM model with an AHB SRAM interface module, suitable for FPGA flow, and permitting read and write operations
- Flash wrapper with simple 32-bit flash memory.
- Flash wrapper with simple 16-bit flash memory.

In project we have used 'cmsdk\_apb\_dualtimers\_defs.v'. In this files 32 bit down timers used in M0 core are initialized. In 'cmsdk\_apb\_watchdog\_defs.v' file watchdog for M0 is initialized. In 'cmsdk\_mcu\_defs.v' file M0 core main definitions are initialized.

## 4.2 Slave

Slave is defined in 'cmsdk\_ahb\_default\_slave.v' file. Slave responds to transfer in case master bus tries to access undefined address. When bus is idle a zero wait state or ok response is generated, however, slave generates error response when sequential or non-sequential transfer takes place. AHB default slave components are shown in figure 4.2

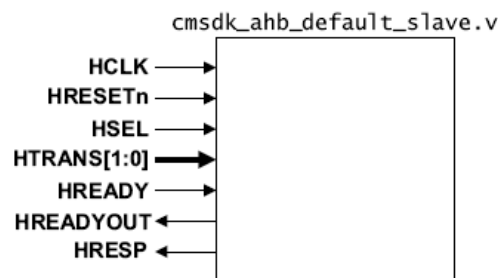


Figure 4.2: 'cmsdk\_ahb\_default\_slave.v' model

## 4.3 Slave select

Slave select module is named as 'cmsdk\_ahb\_slave\_mux.v' file. It supports upto 10 slaves that are connected with AHB bus. Parameters that define slave port usage are also defined in it, so that synthesis does not generate additional logic. Block is shown in the figure 4.3

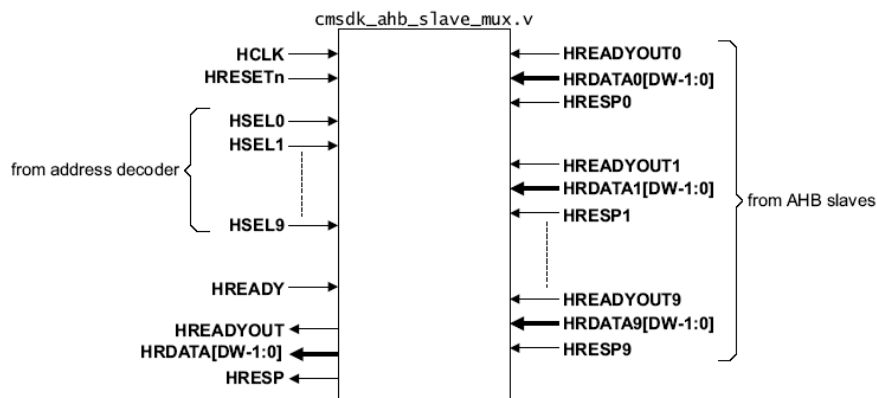


Figure 4.3: 'cmsdk\_ahb\_slave\_mux.v' model

Address decoder determine selected slave and generates correspondence HSEL signal to AHB slave and AHB slave multiplexer. Slave use register version of HSEL as data and signal are valid during data phase. For more than 10 slaves more AHB slave modules can be cascaded.

#### 4.4 Address decoding

In project 'cmsdk\_mcu\_addr\_decode.v' file is used in conjunction with others file. The main aim of this file is to check the M0 generated address and compare it with assigned addresses in to the modules of M0. If valid address is matched it grants the AHB bus access to that module for data transfer.

#### 4.5 APB Bus

APB bus platform is provided in 'cmsdk\_apb\_subsystem.v' file. It is used to interface module slike APB timers, APB UART, Dual input timers, AHB to ABP Bridge, Test slave and IRQ synchronizer. It includes signals to enable all its modules. When address generated by AHB bus matches the address allocated in APB Bus it generates corresponding enable signal. Here HCLK (Clock for AHB) is converted into PCLKG (Gated clock for APB bus). For our code we are using the same clock frequency for AHB to PCLKG. APB bus is used here as 16 bit Address Width and 32 bit data bus.

#### 4.6 ROM

In project 'cmsdk\_ahb\_rom.v' file is used with other files. ROM is of 16 Bit address bus and 32 bit data bus. The ROM model is shown as figure 4.4

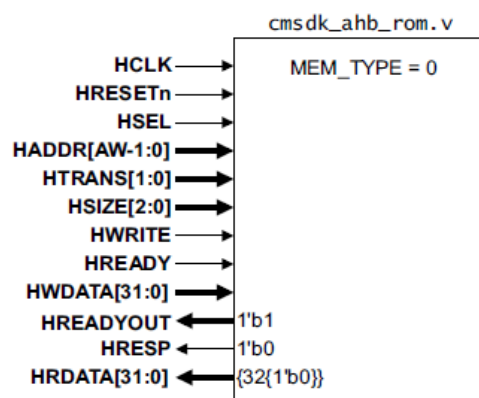


Figure 4.4 cmsdk\_ahb\_rom model

## 4.6 SRAM

It performs read write operations with zero wait state. The design only supports 32 bit memory interfacing. Here, we have made four SRAMs of 8x2048 Bits. The data transferred in it byte wise. The main model is shown in figure 4.5

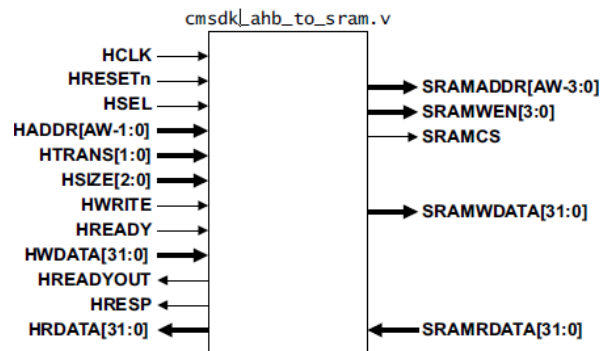


Figure 4.5 cmsdk\_ahb\_to\_sram.v model

## 4.7 ABP Slave select

The slave multiplexer is included in 'cmsdk\_apb\_slave\_mux.v'. It supports upto 16 slaves. In order to do this it uses four bits PADDR to generate corresponding PSEL signal. PADDR can be configured to decode slaves. Figure 4.6 shows APB slave mux module

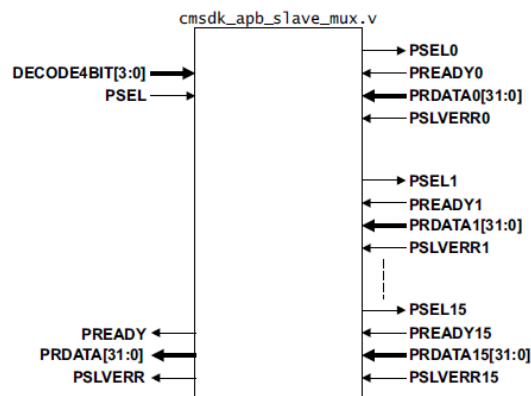


Figure 4.6 cmsdk\_apb\_slave\_mux.v module

## 4.8 ABP Timer

Timer is included in project using cmsdk\_apb\_timer.v file. It is a 32 bit timer with following features:

- Interrupt can be generated using TIMERINT signal as counter reaches zero.
- EXTIN signal can be used as external input signal to enable timer.

- APB timer reaches zero and at the same time software clears previous interrupt status, interrupt is status is set to one.

ABP timer block diagram is shown in figure 4.7

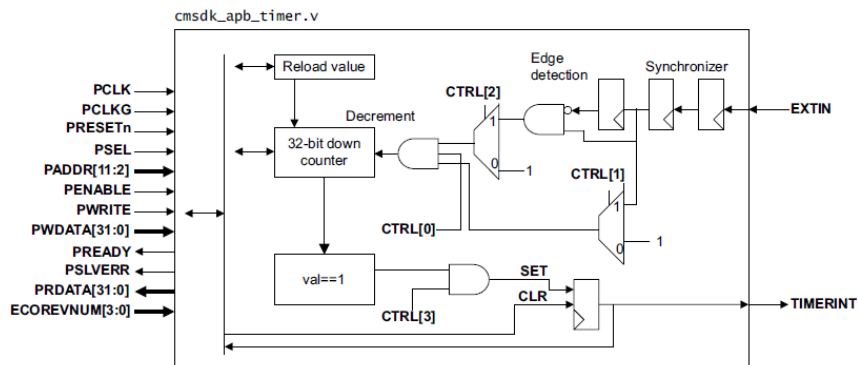


Figure 4.7 block diagram of timer

## 4.9 ABP UART

cmsdk\_apb\_uart.v is APB UART, simple in design that supports 8 bit communication. It has no parity and is zero as stop bit. Block diagram of UART is shown in figure 4.8

UART has two buffers one for data reception and another for data transmission. Interrupt handling execution time is short this leaves sufficient time for processor.

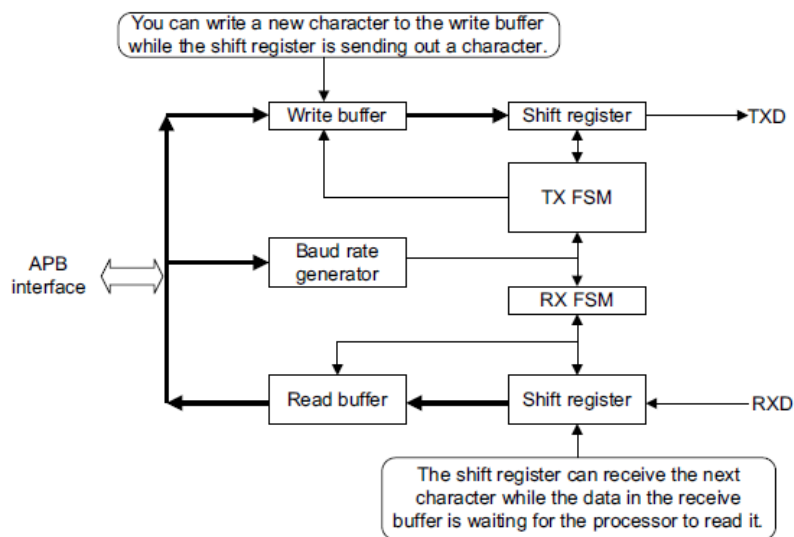
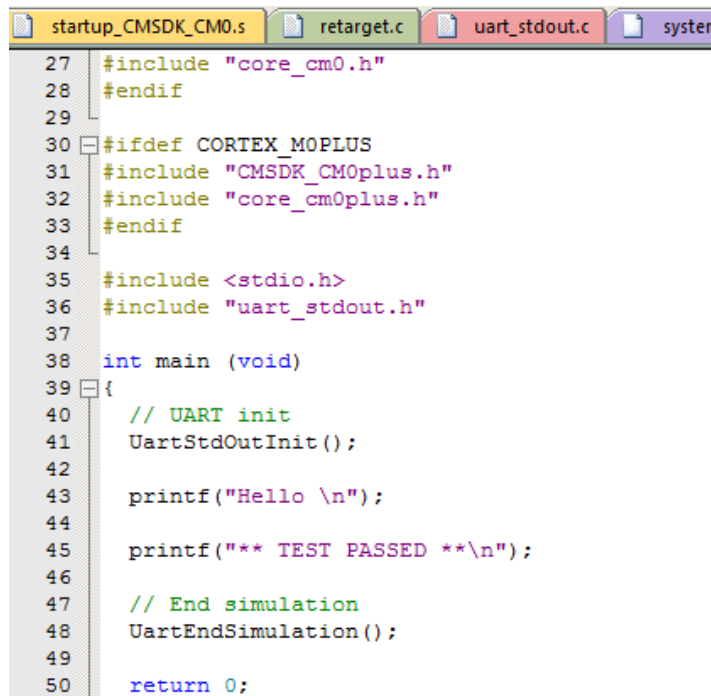


Figure 4.8 block diagram of ABP UART

## Chapter 5: EXPERIMENTS AND RESULTS

M0 core can be programmed in C language, so we need some tool to make hex file of the project. In order to carry out our task we use Keil integrated development environment for our project. In our program we are printing simple message as shown in the figure 5.1



```
27 #include "core_cm0.h"
28 #endif
29
30 #ifdef CORTEX_M0PLUS
31 #include "CMSDK_CM0plus.h"
32 #include "core_cm0plus.h"
33 #endif
34
35 #include <stdio.h>
36 #include "uart_stdout.h"
37
38 int main (void)
39 {
40     // UART init
41     UartStdOutInit();
42
43     printf("Hello \n");
44
45     printf("** TEST PASSED **\n");
46
47     // End simulation
48     UartEndSimulation();
49
50     return 0;
```

**Figure 5.1 Keil Program**

The message is simple Hello \*\* TEST PASSED \*\*. The final file is saved as 'image.hex'

### 5.1 Simulation

In order to simulate our C code, we now need ISE navigator. Open ISE and select the FPGA as shown in the figure 5.2

In file 'cmsdk\_ahb\_rom' file give the path of 'image.hex'. as shown in the figure 5.3

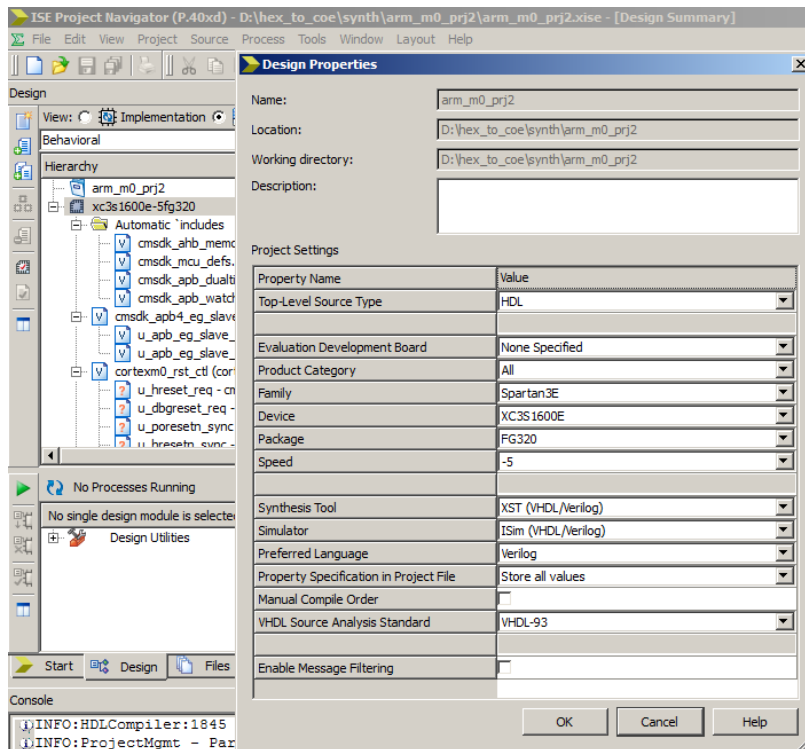


Figure 5.2 FPGA settings

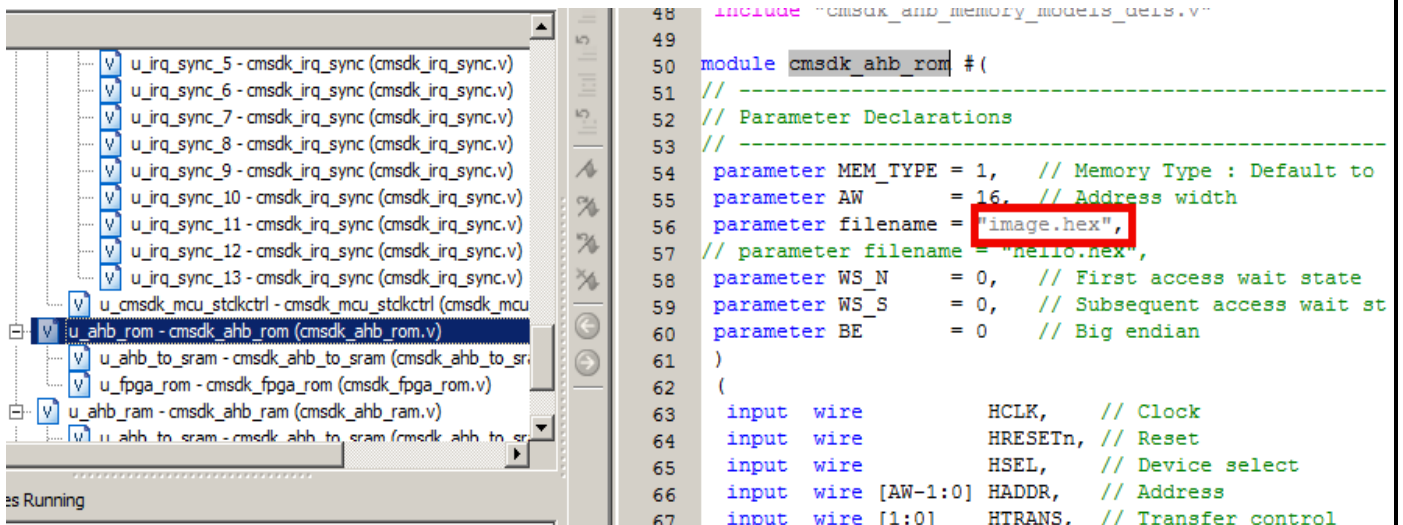
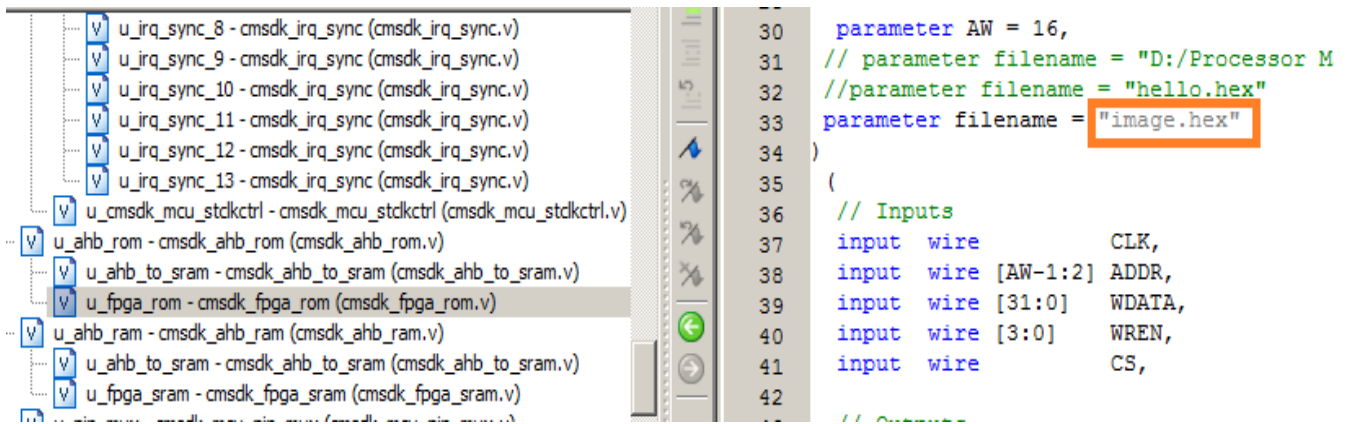


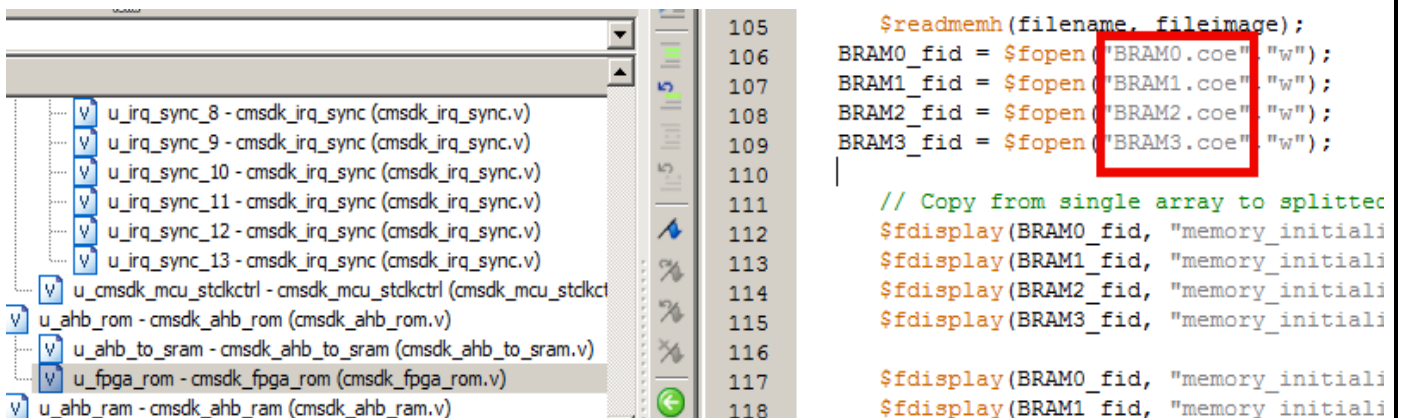
Figure 5.3: Path for image.hex file in cmsdk\_ahb\_rom.v

Also, add file 'image.hex' in 'cmsdk\_fpga\_rom' file as shown in the figure 5.4



**Figure 5.4: Path for image.hex file in cmsdk\_fpga\_rom.v.**

In order to make M0 core we need four BRAMs each of 8x2048 byte. Make four files in same project directory. Save as "BRAM1.coe", "BRAM2.coe", "BRAM3.coe" and "BRAM4.coe". Add files in the cmsdk\_fpga\_rom.v file as shown in the figure 5.5



**Figure 5.5: Path for BRAM files in cmsdk\_fpga\_rom.v.**

After that simulate the code and output is shown in the figure 5.6.

The simulation results are confirmed by matching first byte of 'image.hex' file with "BRAM1.coe" file. Both bytes should be same. We can check this by modifying first byte of 'image.hex' file and simulate the M0 core. After simulation both the bytes should be same.



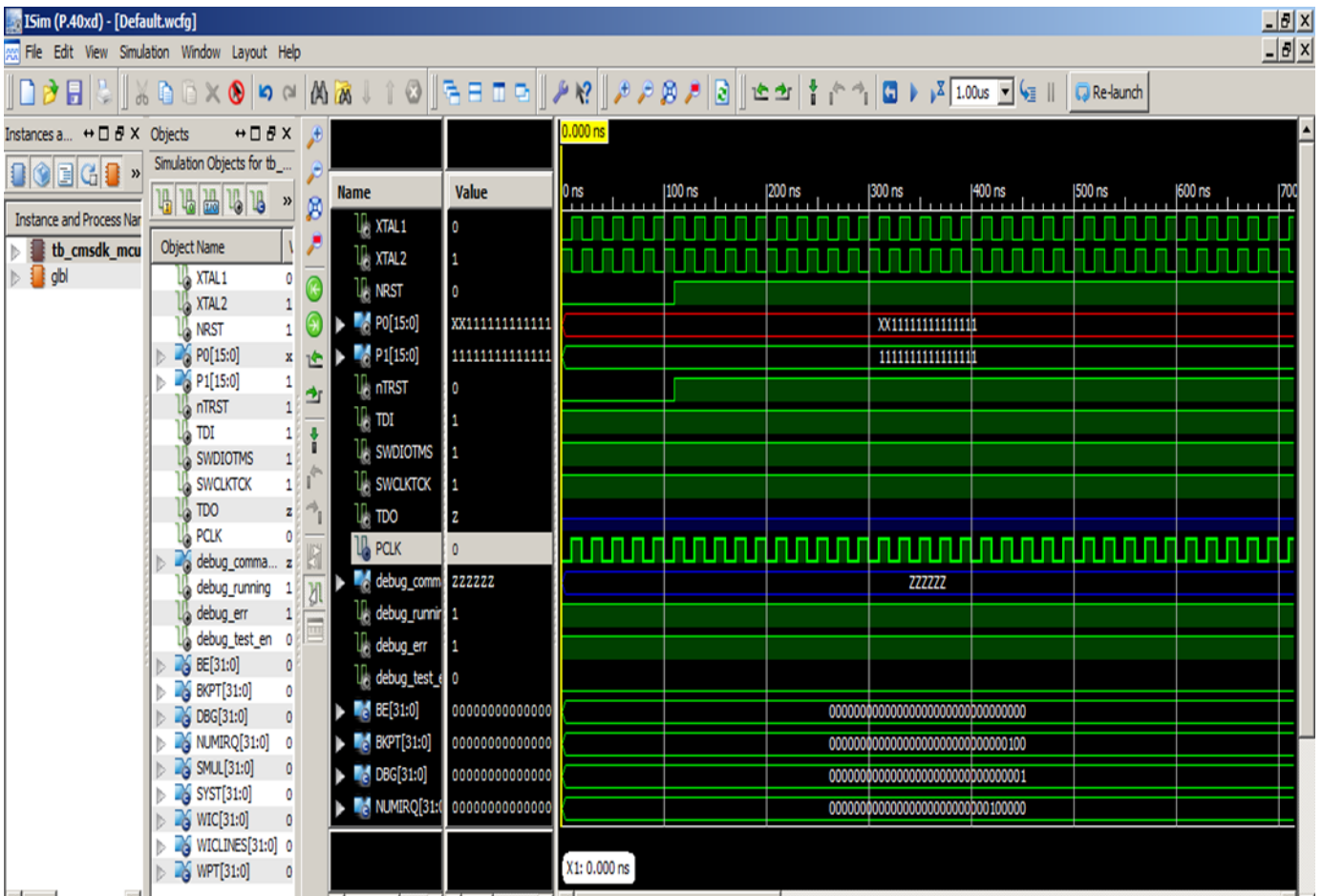


Figure 5.6: Simulation of M0 core processor

## 5.2 Implementation

In order to implement M0 core, so open project ‘ARM\_prj2\_m0’. Add ‘image.hex’ file in ‘cmsdk\_ahb\_rom’ as shown in the figure 5.7

```

50 module cmsdk_ahb_rom #(
51 //
52 // Parameter Declarations
53 //
54 parameter MEM_TYPE = 1, // Memory Type : Default
55 parameter AW = 16, // Address width
56 parameter filename = "image.hex",
57 parameter WS_N = 0, // First access wait state
58 parameter WS_S = 0, // Subsequent access wait state
59 parameter BE = 0 // Big endian
60 )
61 (
62 input wire HCLK, // Clock
63 input wire HRESETn, // Reset
64 input wire HSEL, // Device select
65 input wire [AW-1:0] HADDR // Address

```

Figure 5.7: Path for image.hex file in cmsdk\_ahb\_rom.v.



Make all four BRAMs using IP core generator with specifications as shown in the figure 5.8

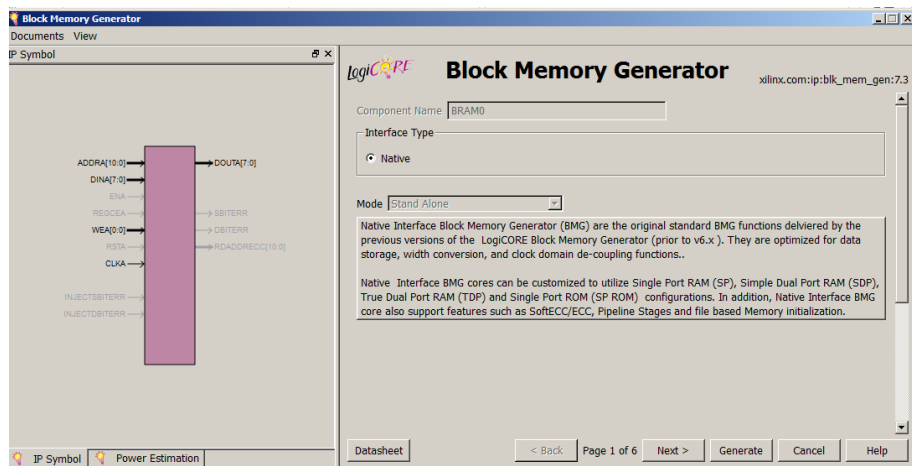


Figure 5.8: BRAM settings

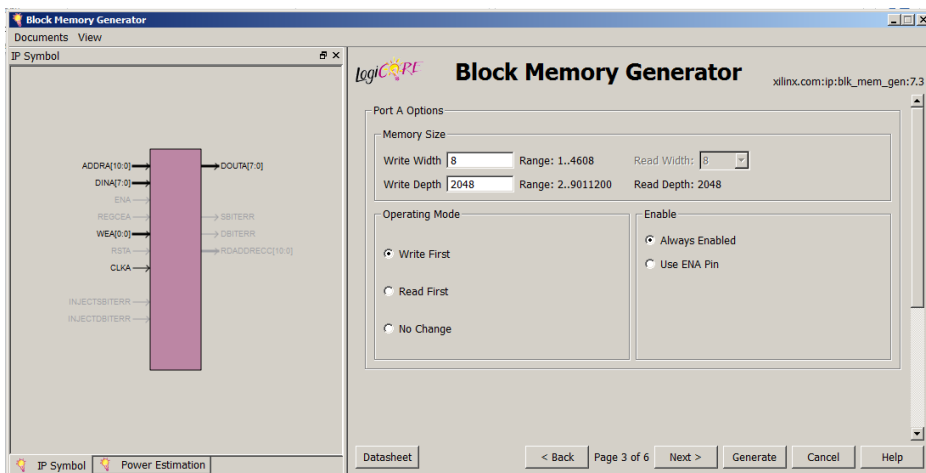


Figure 5.9: 8x2048 size BRAM

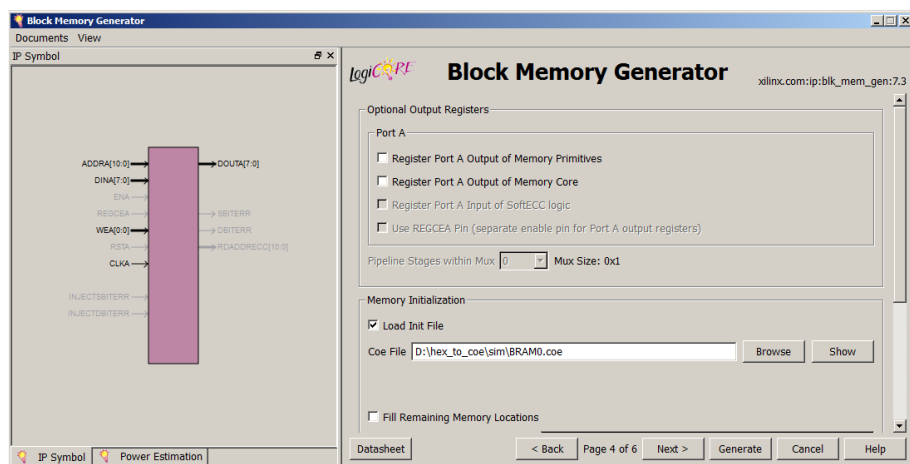


Figure 5.10: BRAM file path

Press generate button to make BRAM.

For clock settings we use DCM IP core. Set Input clock frequency 50 MHz and set output clock frequency. The settings should be as shown in the figure 5.11

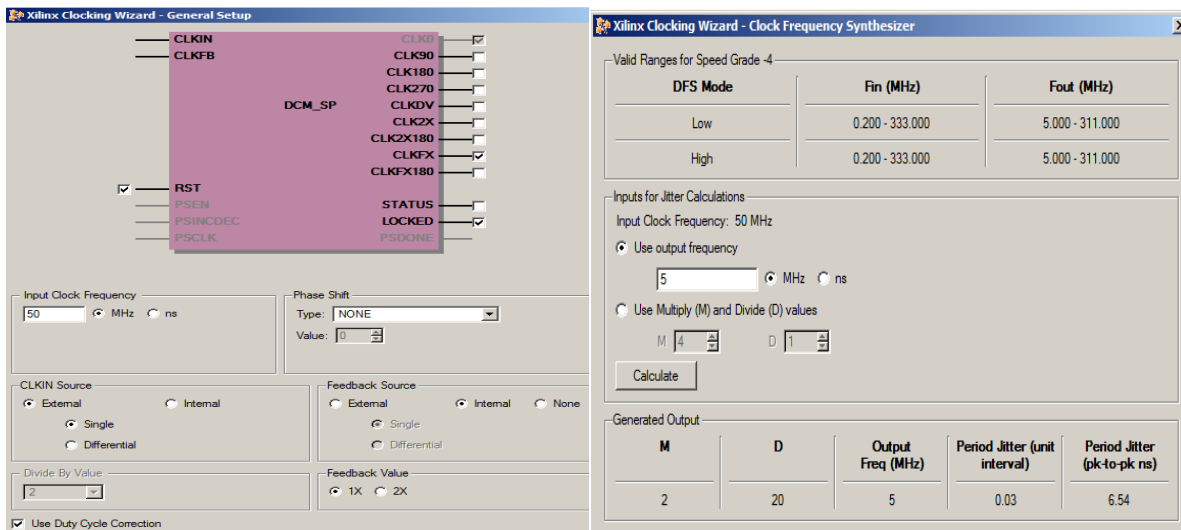


Figure 5.11: DCM clock frequency settings

#### 4.2.1 UART pulse counter

In order to show proper working of M0 core, we need to make a file of 'uart\_pulsecounter.v' file. As it is already discussed that M0 core sends out data on UART. We can observe that data by attaching serial port with our board and using any standard serial port GUI we can observe data transmitted by M0. Here we are using LEDs for same purpose. In 'uart\_pulsecounter.v' file we are counting 210 pulse transitions from 1 to 0. After that we are toggling a LED D13 on demo board. UCF for same function is as shown in the figure 5.12

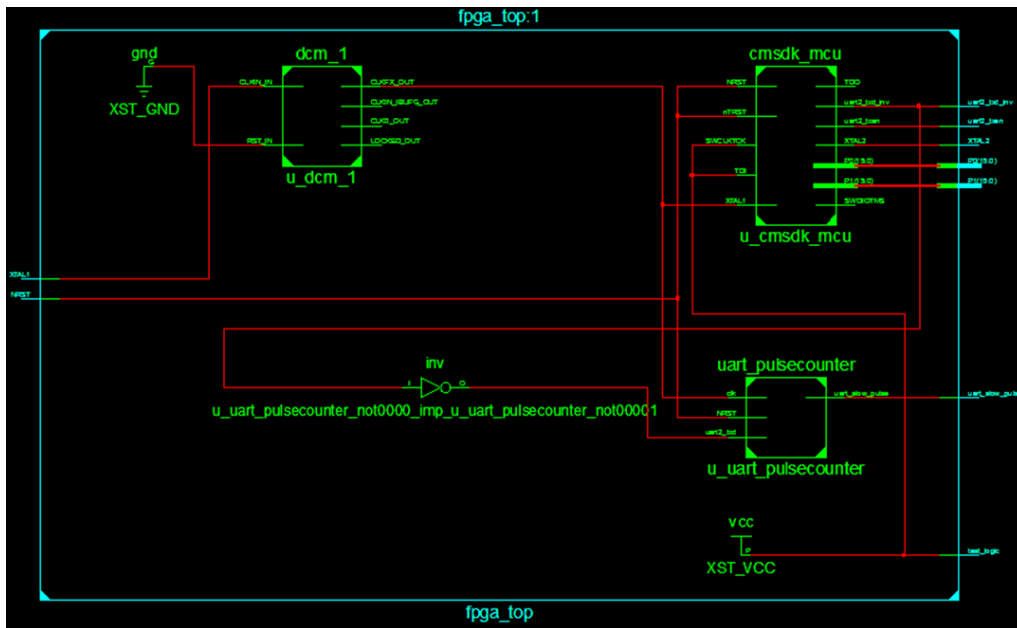
```

9  NET "XTAL1" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
10 NET "XTAL1" PERIOD = 20.0ns HIGH 40%;
11
12
13
14
15
16
17
18 # ==== Discrete LEDs (LED) ====
19 # These are shared connections with the FX2 connector
20 NET "P1<5>" LOC = "R14" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
21 NET "uart2_txd_inv" LOC = "C3" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
22 NET "uart2_txen" LOC = "E6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
23 NET "test_logic" LOC = "D6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
24 NET "uart_slow_pulse" LOC = "D13" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
25 #NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
26 #NET "LED<6>" LOC = "E9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
27 #NET "LED<7>" LOC = "F9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
28
29
30 # ==== Slide Switches (SW) ====
31 NET "NRST" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP ;
32 #NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP ;

```

Figure 5.12: UCF of M0 core

Top level M0 core implementation is shown in figure 5.13



**Figure 5.13: top level M0 core implementation**

After code compilation and generation of file, we can download same file in FPGA and results are as shown in figure 5.14



**Figure 5.14: M0 core implementation on FPGA**

## Chapter 5: EXPERIMENTS AND RESULTS

In this chapter, we evaluate the experiments done and their results on ISIM, Keil as well as on hardware. The results show that the internal memory of embedded M0 soft-core is accessed, data is processed and output is achieved according to requirement.

### 5.1 Results

#### 5.1.1 M0 core implementation on FPGA



Figure 5.1: Result data

## **Chapter 6: CONCLUSION & FUTURE WORKS**

### **6.1 Conclusion**

The remarkable conclusion of this work is that cortex Mo customized soft core can be implemented in a smart FPGA which can be used as customized smart ASIC (Application Specific Integrated Circuit) for various applications.

In that work, ARM Cortex Mo soft-core is synthesized with code memory of FPGA using Advance Microcontroller Bus Architecture (AMBA Lite).

### **6.2 Future Work**

In future works, the implemented cortex Mo core will be used for encryption and security purpose.

For that purpose, in order to increase the processors capacity, other peripherals will be connected to the AMBA bus. Linux operating system can be run over this processor that will make possible to get Linux implementation in a small footprint design.

## References

- [1] ARM Ltd, "ARM DDI 0419C ARMv6-M Architecture Reference Manual", September 2010.
- [2] ARM Ltd, "ARM IHI 0033A AMBA 3 AHB-Lite Protocol V.1 Specification", June 2006.
- [3] Calix A Recado , Sankaran Rajesh "On the feasibility of an embedded machine learning processor for intrusion detection," IEEE international conference on Big Data, 2016.
- [4] Maas Martin, Love Eric, Stefanov Emil, Tiwari Mohit, Song Dawn "PHANTOM: Practical Oblivious Computation in a Secure Processor", university of California Berkeley.
- [5] Wang yi, Shilong Lu "Design and implementation of a SOC based security coprocessor and program protection mechanism for WSN".
- [6] ARM Ltd, "AT510-DC-80001-r0p0-00-re10 ARM Cortex M0 DesignStart Release Note" August 2010.
- [7] ARM Ltd, "ARM DDI 0432C Cortex M0 Revision r0p0 Technical Reference Manual", November 2009.
- [8] ARM Ltd, "ARM DUI 0497A Cortex M0 Devices Generic User Guide", October 2009.
- [9] Xilinx, "DS312 Spartan-3E FPGA Family: Datasheet", August 2009.
- [10] Digilent, "Digilent Spartan 3E Starter Kit Reference Manual", June 2008.