Softcore HDL processor for implementation in FPGA and ASIC

Bartas Venckus

Thesis submitted for the degree of
Master in Informatics: Nanoelectronics and Robotics
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

August 2019

Softcore HDL processor for implementation in FPGA and ASIC

Bartas Venckus

Softcore HDL processor for implementation in FPGA and ASIC

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

The focus of this thesis is on finding a softcore processor that is capable of performing scientific calculations for applications such as the 4DSpace project. The required softcore processor should have an easy and well-documented development process, as well as usefull customization options. The portability of a softcore processor to application-specific integrated circuits (ASIC) is also an important requirement. An open-source LEON3 softcore processor was chosen for this purpose and tested for the multi-needle Langmuir probe (m-NLP) application. The test was conducted with the source code from an m-NLP project, which calculates plasma parameters. LEON3 was configured to meet the time requirements for the m-NLP project. Results for the performance, resource utilization and power consumption of each configurations are presented. The results conclude that a LEON3 configuration with a lite version of a floating-point unit (FPU) is required to meet the set requirements. Estimates for an ASIC implementation of LEON3 demonstrate that LEON3 is a viable option for a softcore processor for the intended use in the 4DSpace project.

Acknowledgments

I would like to thank my supervisor Joar Martin Østby for the motivation, patience and help throughout my whole master thesis. I also would like to thank my supervisor Philipp Dominik Häfliger for providing me with inspiration for this thesis, and for his guidance during the thesis. A big appreciation goes to some of my fellow students at UiO, for helping me out with particular programming problems faced during this thesis.

Nomenclature

ADC	Analog-to-Digital Converter
AHB	Advanced High-Performance Bus
AMBA	Advanced Microcontroller Bus Architecture
ARM	Advanced Reduced Instruction-Set Computer Machine
ASICs	Application-Specific Integrated Circuits
BCC	Bare-C Cross Compilation
CFI	Common Flash Interface
CISC	Complex Instruction-Set Computer
CLB	Configurable Logic Block
CLPD	Complex Programmable Logic Device
CPU	Central Process Unit
DDR	Double Data Rate
DMA	Direct Memory Access
DSP	Digital Signal Processing
DSU	Debug Support Unit
ESA	European Space Agency
FPGA	Field-Programmable Gate Array
FPU	Floating-Point Unit
FSL	Fast Simplex Link
FT	Fault Tolerant
GP	General Purpose
GPIO	General-Purpose Input/Output
GPL	General Public License
GRFPU	Gaisler Research Floating Point Unit
GRLIB	Gaisler Research IP Library

HP	High Performance
I2C	Inter-Integrated Circuit
IC	Integrated Chip
ICI	Investigation of Cusp Irregularities
ISA	Instruction-Set Architecture
LGPL	Lesser General Public License
LMB	Local Memory Bus
LRR	Least-Recently Replaced
LRU	Least-Recently Used
LUT	Look-Up Tables
MAC	Multiply and Accumulate
MC	Minimum Configuration
MCGL	Minimum Configuration with GRFPU-Lite
MIPS	Microprocessor without Interlocked Pipeline Stages
m-NLP	Multi-Needle Langmuir Probe
MPU	Memory Protection Unit
MWP	Multi-Project Wafer
P&P	Plug and Play
PCI	Peripheral Component Interconnect
PMU	Power Management Unit
PS	Processor System
RAM	Random Memory Access
RISC	Reduced Instruction-Set Computer
ROM	Read-Only Memory
SEU	Single-Event Upset
SIMDs	Single Instructions, Multiple Data
SMP	Symmetric Multi-Processor

SoC	System-on-Chip
SPARC	Scalable Processor Architecture
TSMC	Taiwan Semiconductor Manufacturing Company
UART	Universal Asynchronous Receiver-Transmitter
UMC	United Microelectronics Corporation

Table of Contents

List of figures

List of Tables

1 Introduction

The Nanoelectronics Group at the Institute of Informatics (IFI) develops application-specific integrated circuits (ASICs) for application areas such as medical, space, biology, health, and high-quality measurements. The group's intent is to have its own small general processor that could be synthesized on a field programmable gate array (FPGA) and on an ASIC. The softcore processor should be described in hardware description language or very-high-speed integrated circuit hardware language (VHDL).

The departments of Physics, Informatics, and Mathematics at the University of Oslo (UiO) are currently working with the 4DSpace strategic research initiative. The main goal of this initiative is to identify an integrated approach for understanding ionospheric plasma instabilities and turbulence and their role in space weather.

The multi-needle Langmuir probe (m-NLP) project is one part of that integrated approach. An instrument with four needle probes is placed in an Investigation of Cusp Irregularities (ICI) sounding rocket[1] or satellite, where it measures and collects the *in-situ* current from the individual needle probes [2]. On a satellite, for example NORSAT-1, the collected data is converted to voltage, then filtered and digitized. The plasma parameters are then calculated and sent to the central telemetry system for downlink to the ground stations. For a sounding rocket system, the data is collected, processed, and sent to the ground stations without calculating the plasma parameters. This is because of mission time limitations and onboard processing times. In the thesis written by Erik Nobuki Kosaka the possibility of doing plasma parameter calculation on the sub-payloads of the ICI sounding rocket's 4DSpace module was proven to be realistic. Onboard processing is considered because of the bandwidth limitations which does not allow for all of the raw data to be sent to ground.

In this thesis, softcore processor options will be explored and tested for possible use in the m-NLP project. There are many open-source softcore processors that are at different developmental and confirmation stages, as well as many commercial softcores for sale in a broad price range.

1.1 Motivation

One of the main reasons for using a softcore processor instead of a hardcore processor is that the development, testing, and redesigning times are exceedingly faster because of a simpler design cycle, a more predictable project cycle, field reprogrammability, rapid prototyping, and a feasibility study.

The advantage of a softcore processor is its simple customizability, processor parts can be added or removed in a relatively short amount of time. A softcore is also advantageous because of the relatively easy portability to ASICs. The test of a softcore processor on an FPGA takes less time because of multiple test patterns can be tested faster on an FPGA board than the protracted simulations for an ASIC design.

When the softcore processor is tailored, tested, and meets the application requirements, it can be ported to ASIC technologies, where it can support higher speeds, lower power consumption, lower or higher temperatures, and more radiation. The latter is possible to counter with mitigation propagation processes of single event upsets (SEUs) and radiation-hardened technology processes. The SEUs are caused by ionized particles that strike a micro-electronic device and can change the state of a logic element. Once ported to an ASIC, a softcore processor will have a lower unit price and a smaller size, and it may include other parts such as random memory access (RAM), microelectromechanical systems (MEMS), an image sensor, and power regulators.

The plasma parameter calculations for the m-NLP project are currently tested with a microcontroller unit (MCU) (TMS570ls1224), which possesses an advanced reduced instruction-set computer machine (ARM) Cortex-R4F central processing unit (CPU). These test results are used in the present thesis as benchmarks to determine wether the chosen softcore processor is capable of performing those calculations within the given time specifications.

The potential future use of the chosen softcore processor in this project is only one of many possible utilizations. Since softcore processors can be tailored to practically any extent, many possibilities exist for their utilization in other projects, both at the physics department and in the Nanoelectronics Group.

1.2 Goals

The objective of this thesis can be divided into three different main goals. The first goal is to understand the principles of softcore processors, identify the available softcore processors, analyze and compare them, and then draw a conclusion about which is the most suitable for use in this study. Since a significant amount of in-depth research has already been conducted by other researchers, the analysis and comparison herein are done based on the results reported in those studies.

The second goal of this thesis is to further analyze the chosen processor; describe its features and development tools; implement it on an FPGA; and test it against the Cortex-R4F processor, used in a 4D module, with the science routine for plasma parameter calculations. The third goal of this thesis is to determine the resource utilization, power consumption, and predictions for future implementation of the chosen softcore processor on an ASIC.

1.3 Thesis Outline

The remainder of this thesis is outlined as follows: Chapter 2 presents the data acquisition techniques and explains the main differences between hardcore processors, digital signal processors, and softcore processors. Then, Chapter 3 describes what a softcore processor is and discusses the processor architectures. It also presents an overview, analysis, and comparison of different softcore processors. In Chapter 4 in-depth information is provided about the chosen softcore processor and its development tools and Chapter 5 addresses the implementation processes of the chosen softcore processor on an FPGA. Thereafter, Chapter 6 discusses the testing, resource, and power consumption characterization of the softcore processor implementation, and in Chapter 7 the challenges for ASIC implementation of a softcore processor are presented followed by discussion about the rough estimates for its power, area, and price. Finally Chapter 8 concludes this thesis.

2 Data Acquisition and Signal Processing for Lower Data Rates

Most of the sensor data acquisition and digital signal processing (DSP) can be done using simple CPUs with specific techniques. Those techniques and possible CPU solutions are discussed in this chapter.

2.1 Fixed-point vs. Floating-point

Digital signal processing can be categorized as: fixed-point DSP or floating-point DSP. This essentially means that the processing is conducted with positive and negative whole numbers for fixed-point DSP via minimum bits, yielding 2^{16} (65,536) bit patterns. In contrast, floating-point DSP uses rational numbers via a minimum of 32 bits, which leads to 2^{32} bit patterns, where both large and small numbers can be represented. The latter is mostly used for a dynamic range, in which large data sets need to be processed and where they can also be unpredictable.

Floating-point DSP units cost more area and power than fixed-point DSP units. It is easier to develop a floating-point algorithm than a fixed-point algorithm, since the latter requires greater manipulation to quantize noise. Greater performance efficiency can be achieved with a fixed-point processor, since it uses less area and consumes less power than a floating-point processor for the task that must be accomplished [3].

For the IFI's Nanoelectronics group applications that require the use of a processor, the fixed-point calculations are the primary choice. However, the possible applications of floating-point calculations are still considered as options for some projects and is covered in this thesis.

2.2 Field-Programmable Gate Array vs Digital Signal Processors

With the recent advances in FPGA technology, VHDL development, and testing tools, a designer can implement complex integrated functionality on a single die. This can be further extended to a complete microcomputer system. An FPGA consists of three main components: logic blocks, I/O blocks, and interconnection wires. Logic blocks are composed of several inputs and one output, look-up-tables (LUT), small RAM, flip-flops (FFs), and special arithmetic logic support. These programmable blocks are connected with wires and programmable switches used to set up desired connections between the logic blocks [4, p. 100].

The development life-cycle of an FPGA technology processor-based system can be significantly reduced by providing the ability to incorporate several digital cores as reconfigurable, embedded processors in a single die. Using an FPGA approach to implement a DSP processor can drastically reduce the time to market compared to ICs or ASICs. An FPGA's flexibility reduces the long design time associated with ASICs, so that the delay in prototyping can be eliminated with FPGAs. By an using FPGA as one of the platforms for ASIC prototyping and testing, the development and verification of an ASIC processor can be achieved rapidly.

Field programming allows one to bridge flexibility and performance gaps between general-purpose processors and ASICs. Hardcore DSP processors can be programmed with software; however, their architecture is not flexible, since they are constrained by factory settings such as bus width, certain numbers of multiply and accumulate (MAC) blocks, and limited data widths. Field-programmable gate arrays can provide complete hardware customization for any DSP application requirement; they have been used in many DSP embedded systems, and some comparisons of different DSP implementation technologies are represented in Table 1 [5, p. 5].

Feature	ASIC	Structured ASIC	FPGA	Reconfigurable Hybrid	DSP processor	General-purpose µP
Operating freq. (MHz)	> 1,000	> 1,000	100–400	50–300	100–60	100–1,000
Power consumption	Moderate	Moderate	High	Moderate	Very High	Very High
Parallel execution	Maximum	Maximum	Maximum (flexible)	Moderate	Serial (ILP)	Serial (no spec. FUs)
Complexity of design	Very high (50–100-M gates)	Very high	Very high	Moderate	Very complex programs	Very complex programs
Size/are	Large	Large	Very Large	Moderate to high	Moderate to high	Moderate to high
Migration/ evolution	None	Low	Very high	Moderate to high	High (performance limited)	High (performance limited)
Customization	Difficult	Moderate	Easy	Easy to moderate	Easy	Easy
Design verification	Very difficult	Moderate	Moderate	Moderate to difficult	Moderate	Moderate
Design tools	Good, very expensive	Good, moderately expensive	Very good, less expensive	Very poor	Good, inexpensive	Very good, inexpensive

Table 1: Properties of different DSP implementation technologies

An FPGA permits the simultaneous execution of the algorithm's subfunctions. The FPGA can outperform a DSP processor by as much as 1,000:1, although it depends on, for example, clock rates and the degree of parallelism. The typical gains lie between 10:1 and 1,000:1 [6].

In conclusion, an FPGA provides faster development cycle than that of an ASIC or DSP processor, reconfigurability, embedded resources integration, and the development tool efficiency as well as device cost. All these advantages come at the cost of a higher power consumption and higher per-unit cost when compared to other ASIC processors, which can be addressed by porting a VHDL softcore processor design into an ASIC.

2.3 Softcore Processor Requirements for 4D Module Application

Current 4Dspace modules use a Cortex-R4F processor. Considering it as the baseline for softcore processors, some basic requirements can be outlined.

The first requirement is an integer and floating-point performance. A softcore processor should be able to perform, to a certain degree, as well as the hardcore processor. The leeway here is because a softcore implementation on an FPGA will always perform slower than an equal implementation on an ASIC or than a dedicated hardcore processor.

The second requirement is a broad spectrum of alternatives for communication interfaces. These alternatives should support basic interfaces such as Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), Universal Asynchronous Receiver-Transmitter (UART) among others. These interfaces are necessary for an easy integration of the chosen softcore processor in the existing 4D module.

The third criteria for the softcore processor are power consumption and area. These should be within reasonable numbers, which were abstracted after consultation with the Nanoelectronics group and physics department.

The fourth requirement is a possibility to configure the softcore processor against SEUs, which are caused by hard radiation. The SEU mitigation is mostly considered for the use of a softcore processor in 4DSpace modules placed on satellites, where it is exposed to hard radiation over a prolonged mission time.

3 Softcore Processor

In this chapter, necessary information is presented to provide an understanding and definition of a softcore processor. An overview of different processor architectures and their roles is also presented. Three commercial and three open-source softcore processor are presented and explained, and five of them are compared against each other based on multiple study results. Finally, the most suited softcore processor is chosen as the main contender based on multiple aspects and criteria set earlier in this thesis.

3.1 System-on-chip

A system-on-chip (SoC) is a system containing multiple computer components or non-computational components in a single, integrated chip (IC). In contrast to a circuit board, where components are assembled on-to it, an SoC fabricates the components into one unit. Most SoCs include components such as CPU and system memory like RAM or read-only-memory (ROM) [7, p. 1]. An SoC may also include, but may not be limited to the following:

- Real-time clocks, counters and timers.
- Digital Signal Processor
- External interfaces such as USB, Ethernet, or SPI
- Radio frequency (RF) interfaces
- Analog frontends towards sensors

3.2 Microprocessor

A microprocessor (µP) is the CPU of a single IC or at most a few integrated circuits. A µP works mostly in the same way as a CPU does, where it is driven by clocks, which are based on registers where it accepts binary data as an input and processes it according to instructions stored in its memory.

The main difference is that integrating a whole CPU onto single or a few ICs greatly reduces the cost of the processing power, and the price of a µP is lower than that of a CPU, since the former is produced in high numbers by highly automated processes.

Microprocessors are also used in many applications that are not computation related: mostly control systems. A µP in an embedded system controls and processes sensor inputs and outputs as well as other related processes.

3.3 Softcore Processor Definition

A soft µP or a softcore processor is a µP core that is described in an HDL and is implemented in an FPGA, an ASIC, or a complex programmable logic device (CPLD). Since most of softcores on FPGA have low gate utilization and rely on memory technology and LUTs, FPGAs tend to have higher power usage. This can be fixed by porting them to ASIC.

The FPGA implementation also causes lower speeds than compared to hardcore processors, the typical softcore processor has speeds from 200MHz up to 500MHz, whereas a hardcore processor will have speeds anywhere from 100MHz up to 4GHz. This is often addressed by the use of multi-softcore processors and parallel programming. Table 2 [8, p. 4] presents an overview of flexibility and speed tradeoffs for processors in different technologies; the flexibility decreases from the ASIC down to generic technology, while the speed increases from generic up to ASIC technology.

Speed increase	**Technology**	**Performance/ Cost**	**Time until running**	**Time to high performance**	**Time to change code functionality**	**Flexibility decrease**
	ASIC	Very High	Very Long	Very Long	Impossible	
	Custom Processor or DSP Processor	Medium	Long	Long	Long	
	FPGA	Low to Medium	Short	Short	Short	
	Generic	Low to Medium	Short	Not Attainable	Short	

Table 2: Speed and flexibility tradeoffs

3.4 Microprocessor Architecture and Instruction Set Architecture

This section covers the necessary background for a µP architecture and instruction-set architecture. A number of different µP architecture types are available; the most used types for both softcore and hardcore µP are described in this section.

3.4.1 Harvard Architecture

The Harvard architecture is a computer architecture that stores machine instructions and data in separate memory units that are connected by different buses. This architecture allows a processor to run a program and access data independently and therefore simultaneously.

3.4.2 Complex Instruction Set Computer

Complex instruction set computer (CISC) [9, p. 39] is a computer architecture family that incorporates multi-clock complex instructions where one instruction can execute several low-level operations. These operations include loading and storing from or to memory and arithmetic operations. Most of the later instruction sets evolved from CISCs.

3.4.3 Reduced Instruction Set Computer

A RISC [10, p. 11] is a type of µP architecture that was optimized from CISCs, where it shifted the analytical process of a computational task from the execution (run-time) to the preparation (compile time). The optimization allows a RISC to have much lower cycles per instruction than a CISC. A RISC generally utilizes a small, highly-optimized set of instructions, rather than a more specialized set of instructions often found in other types of architectures. This does not mean that it has a small amount of instructions; later versions of RISCs have a larger instruction set than most CISC CPUs. Furthermore, a RISC is a forerunner for the major architectures today, such as ARC, ARM, Atmel AVR, MIPS, RISC-V, SuperH and SPARC, some of which are discussed below.

3.4.4 Advanced RISC Machines

An advanced RISC Machine (ARM) is a RISC type of µP, that has been enhanced with an optimized instruction set and pathways, thereby requiring fewer transistors; this enables a smaller die size for IC and lower power consumption then the other type of RISC µP. The ARM processor's smaller size, reduced complexity and lower power consumption make it suitable for increasingly miniaturized devices. The processors of ARMs possess the same features as a RISC, (load/store architecture, single-cycle execution), which includes but is not limited to an orthogonal instruction set and an enhanced power-saving design [11, p. 2].

3.4.5 Microprocessor without Interlocked Pipeline Stages

A Microprocessor without Interlocked Pipeline Stages (MIPS) is a RISC-type µP architecture based on a 34/64-bit instruction set, and it uses a load/store data model, which is also known as register-register architecture. The architecture is streamlined to support the optimized execution of high-level languages. To further increase efficiency of instructions processing, MIPSs use a technique called pipelining, and since all instructions are 32 bits long, these µP simplify the accessing and decoding instructions [12].

The MIPS is now well developed, tested, and used in many devices around the world. It is a prime architecture for becoming acquainted with how CPUs work, and the guides and support for this architecture are numerous.

3.4.6 Scalable Processor Architecture

Scalable processor architecture (SPARC) [13] is heavily based on early RISC processors (I and II), with minimal operation codes and instruction execution rate at almost one instruction per clock cycle. This makes it a similar architecture to that of MIPS, although it lacks instructions like multiply and divide in the early versions. One primary feature of SPARC is that it is a scalable processor that is implemented with the use of 3 to 32 register windows. By implementing a chosen number of windows, there is a possibility to have maximum call-stack efficiency or reduced cost and complexity if needed.

3.5 Commercial Softcore Processors

Various commercial softcore processors exist. However, in this section only, the most popular ones are discussed.

3.5.1 NIOS II

The NIOS II processor [14] is Altera Corporation's flagship softcore general purpose RISC processor; it features a Harvard memory architecture and is one of the most widely used softcore processors in the FPGA industry. The processor features a 32-bit ISA, 32 general-purpose registers, single-instruction 32x32 multiply and divide operations, and dedicated instructions for 64-bit and 128-bit products of multiplication. NIOS II comes in three versions: economy, standard, and fast. Each core varies in size, register, and pipeline number, and it is possible to scale them for the desired performance, logic number, and power usage. Figure 1 [15, p. 10] illustrates a the standard NIOS II softcore processor block diagram.

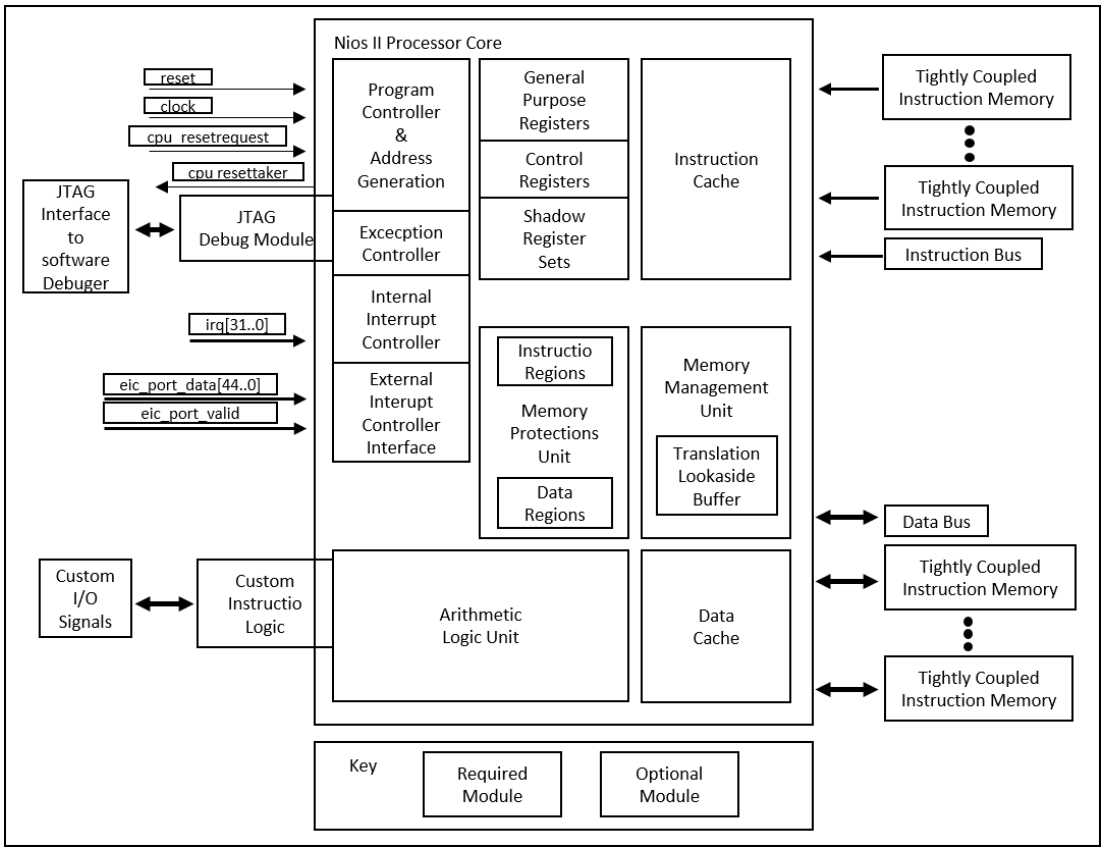


Figure 1: NIOS II Processor Core

3.5.2 MicroBlaze

MicroBlaze [16] is a 32-bit Harvard architecture softcore processor developed by Xilinx. It possesses 32 fixed 32-bit general-purpose registers, a 32-bit instruction word with three operands and two addressing modes, and a 32-bit bus. It also has optional 3 to 5 pipeline depths, a hardware divider, a barrel shifter, and debug logic, as well as a floating-point unit (FPU) and the local memory bus (LMB). The fast simplex link (FSL) interface allows it to include up to eight dedicated, 32-bit input and output ports. MicroBlaze targets Virtex and Spartan families of FPGAs. Figure 2 [17, p. 7] depicts a MicroBlaze core block diagram with fixed and optional features.

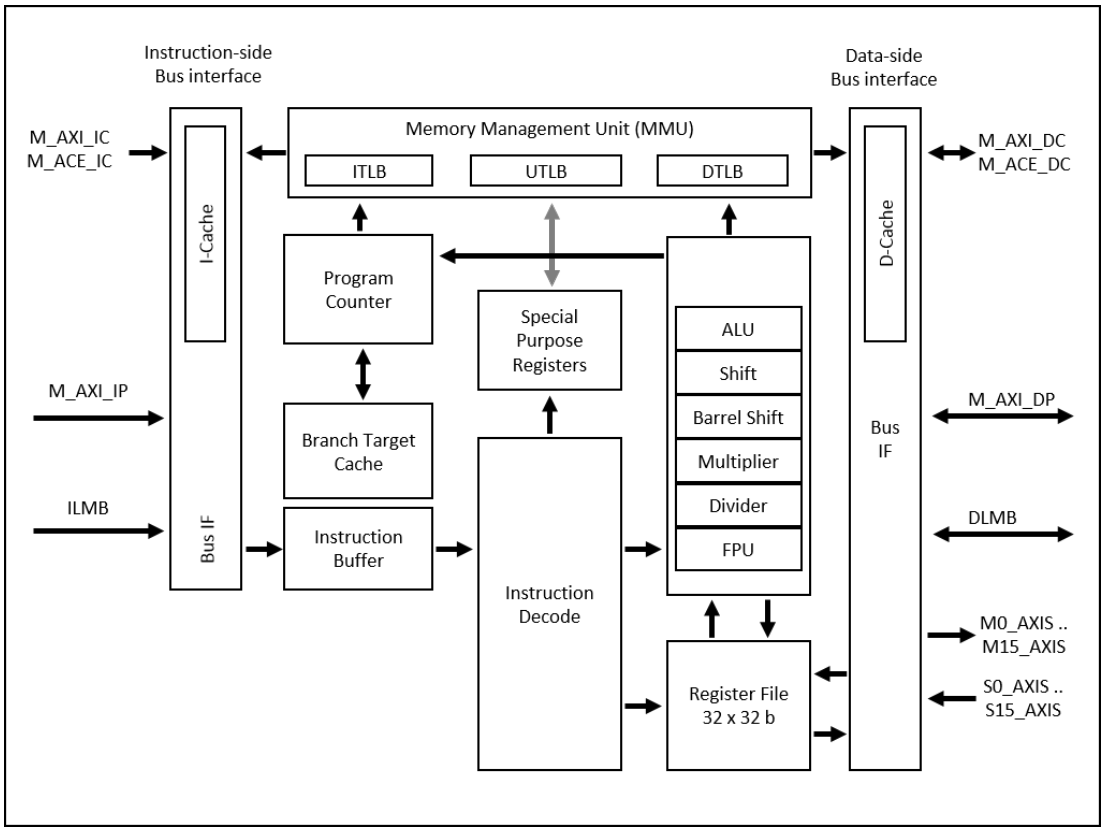


Figure 2: Block diagram of MicroBlaze core

3.5.3 The MIPS Warrior M-Class M5100

The M5100 [18] is a relatively new 32-bit MIPS softcore processor developed by Imagination Technologies; it features state-of-the art power reduction and management, fast interrupt handling, an advanced debug or profile, and total hardware virtualization. The processor implements over 150 instructions, including 70 single instructions and multiple data (SIMD) and 38 multiply or MAC instructions, as well as a seven-stage pipeline, an FPU, and

a memory controller. Figure 3 [18] illustrates an M5100 softcore processor block diagram with included and optional features.

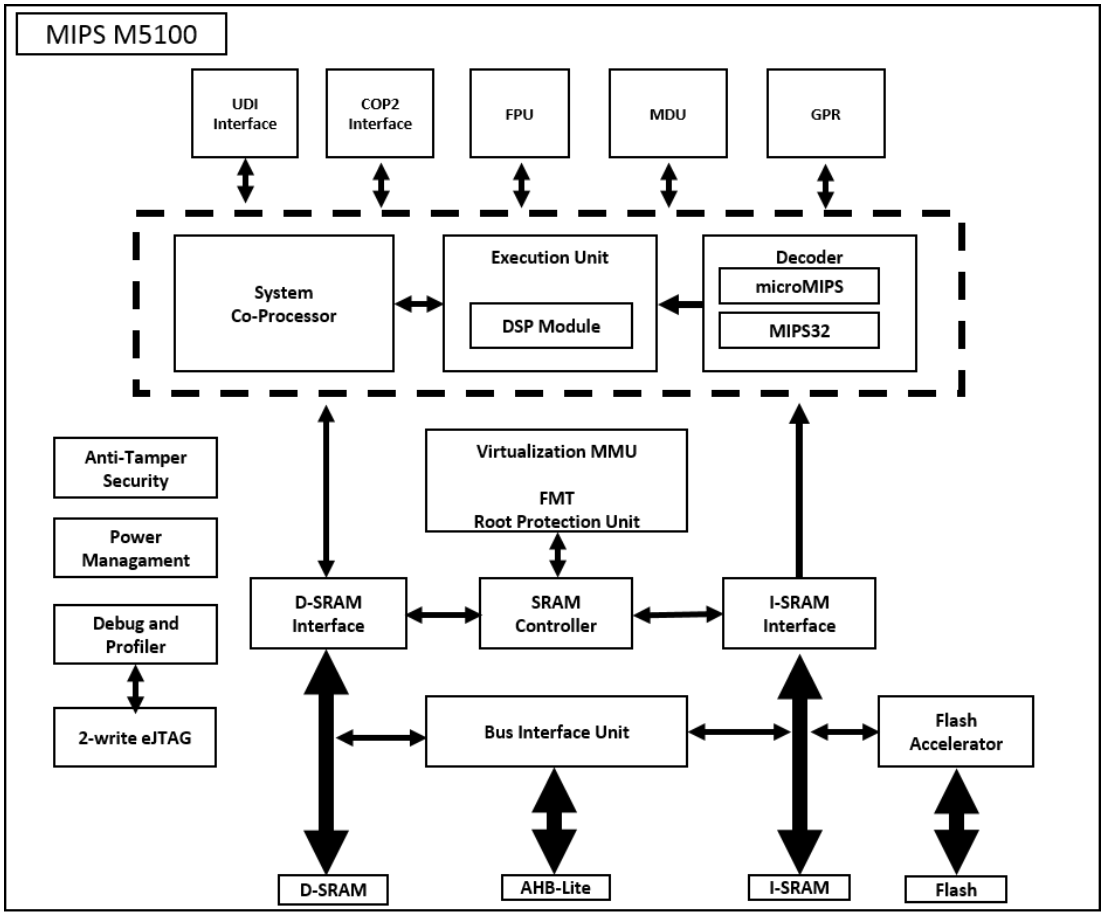


Figure 3: Block diagram of M5100

3.6 Open-Source Softcore Processors

A surprising number of open-source softcore processors exist. Since 1999 the open-source project called OpenCores [19] has published, collaborated, and shared almost 200 different softcore processors under the following licenses: The GNU General Public License (GPL), the Berkeley Software Distribution (BSD) license, and the GNU Lesser General Public License (LGPL) licenses. Although the number of softcore processors available at OpenCores is impressive, only a few are complete designs that have been tested on different FPGAs. Furthermore, not all projects provide the source code files, which made it difficult to study and understand them. Couple of the major ones, which have been utilized and developed by small groups as well as large organizations such as the European Space Agency (ESA) are discussed in the next subsection.

3.6.1 OpenRISC1200

OpenRISC1200 (OR1200) [20] comprises one of the most well-known and popular 32-bit and 64-bit RISC softcore processors found at the Open Core page. It features a five-stage pipeline, a virtual memory support memory management uinit (MMU), a power management unit (PMU), and an interrupt handler. Supplemental facilities include a debug unit for real-time debugging, a high-resolution tick timer, a hardware multiplier, and a divider available with the right configuration. Flextronics have also turned the processor design into an ASIC by Flextronics. The OR1200 is intended for embedded, portable, and networking applications. Figure 4 [21, p. 3] presents an OR1200 core block diagram.

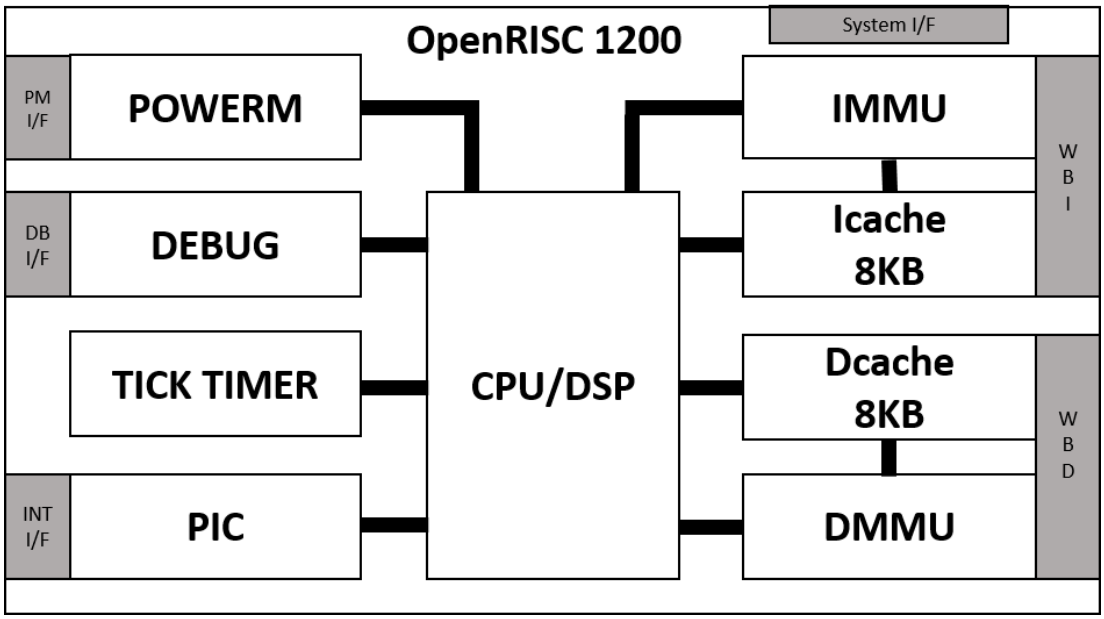


Figure 4: Block Diagram of OpenRISC1200

3.6.2 The Lattice Semiconductor LM32

The LM32 [22], similarly to most other processors described in this essay, is a 32-bit RISC Harvard architecture softcore processor with 32 32-bit general-purpose registers. It features a six-stage pipeline, with register-register arithmetic operations, although it does not possess an FPU. Three configurations are available to optimize area and performance, and several peripheral components might be integrated, for example memory controllers: parallel flash, DDR, DDR2, I/O DMA controller, and UART, among many others. Figure 5 [22] illustrates an LM32 block diagram.

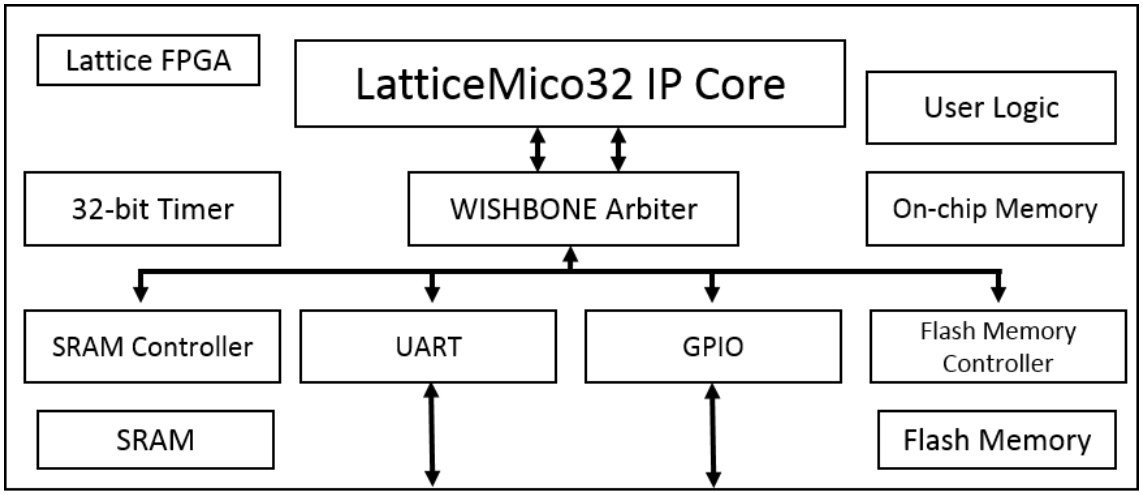


Figure 5: Block diagram for LatticeMicro 32

3.6.3 **LEON3**

LEON3 [23] is a 32-bit SPARC-V8 open-source softcore processor developed by Cobham Gaisler. An early version of the LEON processor was developed by the ESA and was targeted to eliminate SEUs caused by hard radiation. The LEON2 fault tolerant (FT) version was created and used in many satellites, including the Intermediate eXperimental Vehicle (IXV) [24]. Later versions were contracted to Cobham Gaisler, where LEON3, LEON3-FT, LEON4 and LEON4-FT have been developed. The latter is one most powerful 64-bit softcore processors in the world. LEON3 features a seven-stage pipeline, a hardware multiplier, divider and MAC units, as well as an FPU, an MMU, symmetric multi-processor support (SMP), and power management capabilities. Both LEON4 and LEON4-FT are commercial softcore processors.

3.7 Comparison

The research on the processors portrayed in this chapter are found online. All of the benchmarks rely on research and articles done by several different entities. This might cause inconsistencies in test conditions, so the portrayed data in the benchmarks can only be used to form a basic understanding of the relations between the processors in question. Any further assumptions are confirmed with coinciding data from multiple researches. The final choice of softcore processor will mostly depend on specifications, appliances, community, support, and

the documentation around it. Nevertheless, some benchmarks are included to understand softcore processor performances. Table 3 lists the features of both the commercial and open-source softcore processors, only the most promising and best-documented processors are listed.

Feature	NIOS II/f	MicroBlaze	OpenRISCSC 1200	LEON3	M5100
License	Altera IP core	Ships with Xilinx EDK	GNU LGPL	GNU LGPL	
Platform	Altera FPGA, ASIC	Xilinx FPGA	FPGA, ASIC	FPGA, ASIC	FPGA
Architecture	32-bit RISC	32-bit RISC	32-bit RISC	32-bit RISC	32-bit MIPS
ISA	NIOS II-ISA	MicroBlaze ISA	ORBIS32	SPARC V-8	MIPS32 Enhanced
Custom instructions	Yes	Yes	Yes	Yes	Yes
Pipeline stages	6	3-5	5	7	5
Register file	Flat	Flat	Flat	Window	Flat
Nr. of global registers	32	32	32	32	32
FPU support	Yes	Yes	Yes	Yes	Yes
MMU	Yes	Yes	Yes	Yes	Yes
Mac	Yes	N/A	Yes	Yes	Yes
Cache hierarchy	Harvard	Harvard	Harvard	Harvard	SRAM interface
Instruction cache size	512B to 64KB	64B to 64KB	512B to 8 KB	1B to 2MB	N/A
System interface	Ethernet, JTAG, RS232, SPI, PCI	LMB, IBM OPB, FSL, PLB, ICL, XCL	Wishbone SoC rev. B 32-bit	AMBA AHB, RS232, JTAG, PHY, LVDS, CAN, UART	AMBA 3 AHB, AMBA lite, JTAG, UDI
Power management	N/A	Sleep mode	Slow and idle mode, sleep mode, doze mode	Power down and idle mode.	Power down.
Memory interfaces	SRAM, SDRAM, Flash, Memory mapped I/O	DDR SDRAM, SDRAM, SRAM External Flash	SDRAM, SRAM, SSRAM, FLASH	SDRAM, SRAM, PROM Memory mapped I/O	SRAM, ISRAM, DSRAM, FLASH

Table 3: Feature overview of softcore processors

3.7.1 Comparison of MicroBlaze and LEON3(FT)

In “The Evaluation of Soft-Core Processors on a Xilinx Virtex-5 Field Programmable Gate Array” [25], Mark W. Learn compared the MicroBlaze, LEON3, and LEON3FT softcore processors. All figures and diagrams shown in this section were taken from Learn’s evaluation, and two performance benchmark applications were used to evaluate the different softcore processors. The first one was the Dhrystone benchmark, and the second was the Whetstone benchmark.

The Dhrystone [26, p. 4] is a synthetic general-performance benchmark developed by Reinhold P. Weicker in 1984. This benchmark contains no floating-point operations and is

intended to be representative of integer programming; furthermore, it is heavily influenced by hardware and software design. The output from the Dhrystone benchmark is the number of iterations of the main code loop per second, most commonly referred to as Dhrystone million instructions per second (DMIPS).

The Whetstone [26, p. 6] is a synthetic benchmark designed to measure the behavior of scientific programs. This benchmark test attempts to measure the speed and efficiency at which a computer performs floating-point operations. The output is usually given in units called kilo Whetstone instructions per second (KWIPS); however, in this thesis, it will be referred to as million Whetstone instructions per second or (MWIPS).

The results clearly demonstrated that all three processors were greatly improvement with an FPU enabled, although the utilized resources were higher than the without an FPU but still tolerable. With a cache and FPU enabled, the MicroBlaze performance increased by a factor of 40, while LEON3 increased its performance by a staggering factor of between 400 and 1,000. Without the FPU and cache, both processors utilized similar amounts of resources; Figure 6 [25, p. 24] and Figure 7 [25, p. 25] depict these results. LEON3 saw a larger increase in resource utilization in comparison to MicroBlaze once FPU was enabled as seen in Figure 8 [25, p. 26]. This larger increase accounts for the large performance increase when running the Whetstone benchmark test. It should be noted that the MicroBlaze softcore FPU unit can only perform single-precision, floating-point operations, where-as the LEON3 FPU provides the ability to use both single and double-precision floating-point operations.

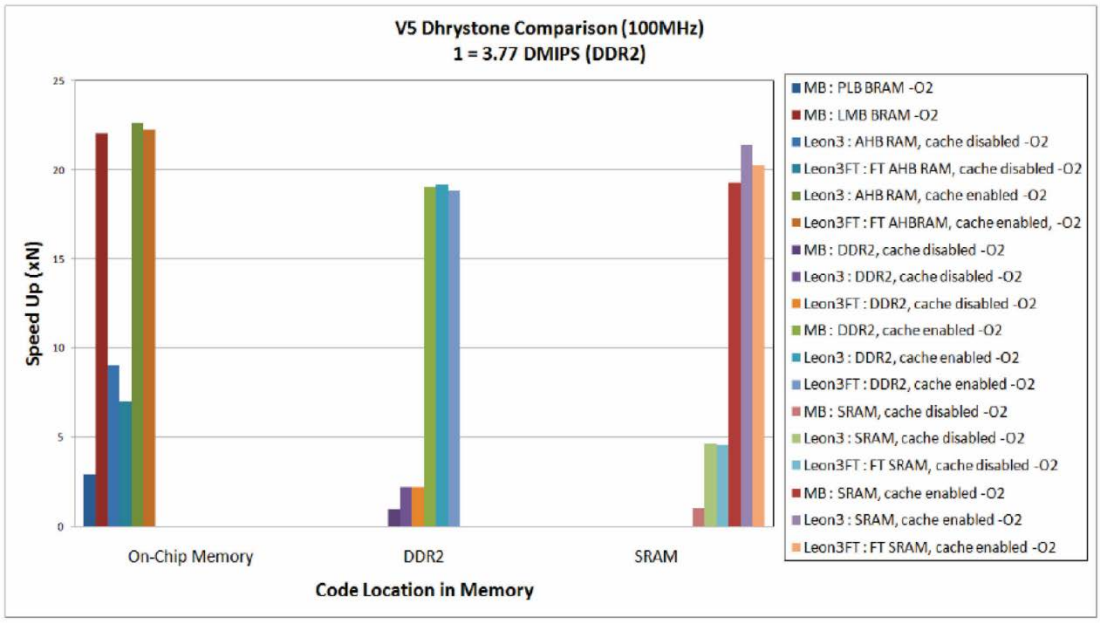


Figure 6: Virtex-5 Dhrystone comparison

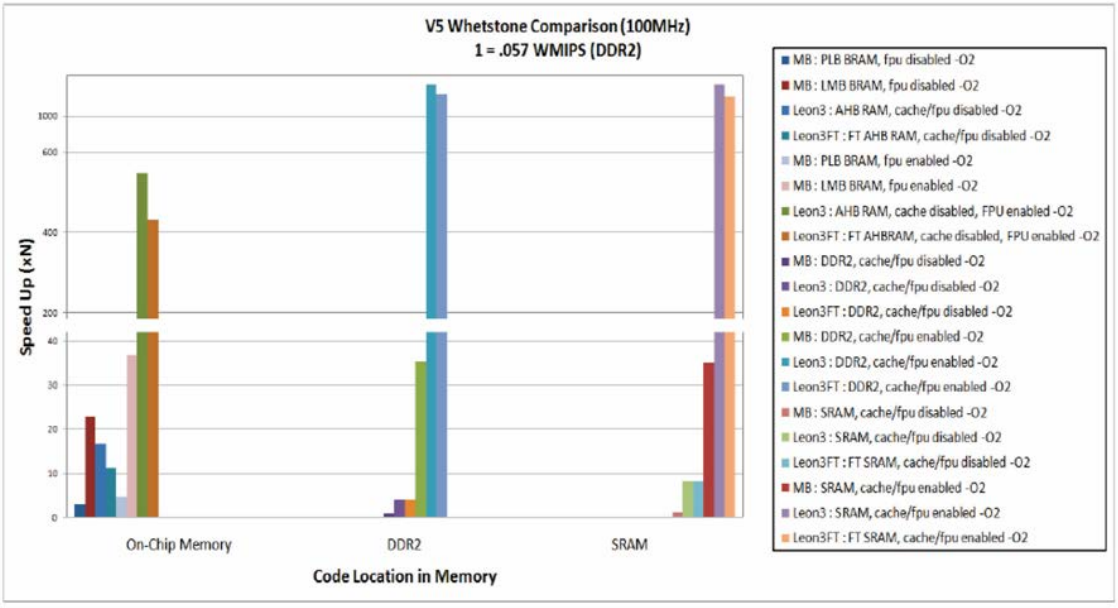


Figure 7: Virtex-5 Whetstone comparison.

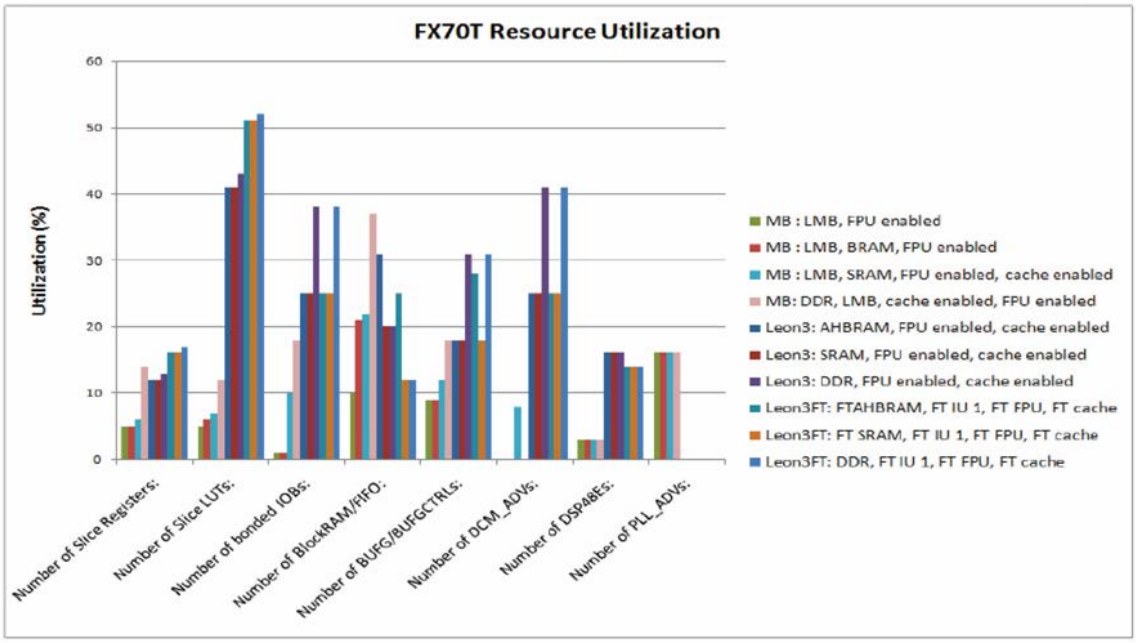


Figure 8: Processor resource utilization

3.7.2 Comparison of LEON3, MicroBlaze, OpenRisc1200 and Cortex-MO

In “An Evaluation of Soft Processors as a Reliable Computing Platform” [27] by Michael Robert Gardiner, LEON3, MicroBlaze, OpeRisc1200, and Cortex-MO were tested and benchmarked against themselves and the radiation hardened processor RAD750. The tests in this evaluation were performed on two FPGA boards, the first one being LX110T [28], which is a generic Xilinx FPGA board, and the second being Virtex-5QV [29], which is a space grade FPGA board. The results demonstrated in the current thesis concentrate on the LX110T board.

The Gardiner’s study used multiple benchmarks to determine the performance of the softcores. The benchmarks used in that study are the standard Whetstone and Dhrystone benchmarks; the CoreMark [30]; and ones such as basicmath, bitcount, dijsktra, fft and stringsearch from the MiBench suit [31]. For the purpose of this research, the results taken from Gardiner’s study focus on the Whetstone and Dhrystone benchmarks; however, data from the rest of benchmarks are also displayed.

Although the study came to many interesting conclusions, only the benchmarks for softcores with enabled and disabled FPUs are discussed next. As concluded earlier, on the one hand, MicroBlaze has better integer performance than LEON3. This is also true when compared to the other two softcores, as indicated in Table 4 [27, p. 57]. On the other hand, LEON3 mostly outperforms other processors on the floating-point performance. As mentioned in the previous subsection, the LEON3 FPU has the capacity to perform double-precision, floating-point operations; this is reflected well in Table 5 [27, p. 62]. The MicroBlaze softcore processor performed over 10 times worse than LEON3 in the double-precision, Whetstone benchmark, but it performed better at single-precision Whetstone benchmark by a factor of less than two. In Table 6 [27, p. 64] the LEON3 demonstrates the best performance-scaling with frequency, and MicroBlaze coming in second.

Benchmark	Units	MicroBlaze		LEON3		Cortex-M0 DesignStart	OpenRISC 1200	
		Design	Score	Design	Score	Score (CB0)	Design	Score
Dhrystone	DMIPS	MB0	192.22	LB16	76.23	73.94	OB16	44.56
CoreMark	CM	MB0	225.79	LB16	122.43	92.81	OB16	92.65
bitcount	BPS	MB0	3.20	LB64	1.14	N/A	OB16	0.77
dijkstra	BPS	MB0	1.07	LB64	0.53	0.46	OB16	0.30
qsort	BPS	MB0	642.17	LB64	295.73	148.94	OB16	140.30
stringsearch	BPS	MB0	52.96	LB64	30.66	24.06	OB16	14.94

Table 4: Highest integer benchmark scores and corresponding designs for each processor implemented in the LX110T

Benchmark	Units	MicroBlaze		LEON3		Cortex-M0 DesignStart	OpenRISC 1200	
		Design	Score	Design	Score	Score (CB0)	Design	Score
WhetstoneDP	WMIPS	MB0	8.37	LB16	93.40	1.89	OB16	3.12
WhetstoneSP	WMIPS	MB0	175.15	LB16	97.30	9.01	OB16	6.75
basicmath	BPS	MB0	0.14	LB16	3.70	0.08	OB16	0.04
fft	BPS	MB0	4.40	LB64	104.33	3.04	OB16	2.57

Table 5: Highest floating-point benchmark scores (with FPUs enabled) and corresponding designs for each processor implemented in the LX110T

Benchmark	Units	MicroBlaze		LEON3		Cortex-M0 DesignStart	OpenRISC 1200	
		Design	Score	Design	Score	Score (CB0)	Design	Score
WhetstoneDP	WMIPS/MHz	MB0	0.065	LB16	1.215	0.025	OB16	0.049
WhetstoneSP	WMIPS/MHz	MB0	1.365	LB16	1.266	0.117	OB16	0.106
basicmath	BPS/MHz	MB0	1.076E-03	LB16	0.048	1.063E-03	OB16	7.045E-04
fft	BPS/MHz	MB0	0.034	LB64	1.357	0.040	OB16	0.040

Table 6: Floating-point performance per MHz for each processor (with FPUs enabled) implemented in the LX110T

3.8 Conclusion to Chapter 3

After analysis and comparison of the available commercial and open-source softcore processors, it seemed safe to conclude that performance-wise, some more-than-capable open-source softcores exist. The major concern was the support and community around the open-source processors, compared to commercial ones, where one receives guaranteed support with an IP license. However, further investigation refuted that concern. OpenRISC1200 and LEON3 have been tested, developed and conformed for quite a while, although LEON3 is a clear winner in this case. The amount of research and written publications on LEON3 is extensive [32, 33, 34, 35, 36]. Moreover, the Yahoo group [37] on LEON3 is highly active today. The activities in the group have been monitored since October of 2017; it is suffice to say that most questions asked by users are addressed within one to two days.

The level of customization and scalability is the second aspect that makes LEON3 most the attractive of all the previously mentioned softcores. This is mainly because it has SPARC architecture. The third reason is that development tools are extensive, compared to other open-source softcores, and the possibility exists to further develop a free licensed LEON3 to FT version, which has already been done by Gaisler and some other researchers [38, 39, 40]. The fourth reason is the extent of documentation and guides on LEON3, which is second to none of the open-source softcore processors [41, 42, 43, 44, 45, 46, 47, 48].

4 LEON3

This chapter covers relevant parts of The Gaisler Research IP Library, its structure and IP cores. The IP cores are important parts of LEON3's customizability, and they are collectively an adaptability for future implementations. Necessary LEON3 tools for development and testing are discussed in section 4.2.

4.1 The Gaisler Research IP Library

The Gaisler Research IP Library (GRLIB) [49] is an integrated set of reusable IP cores, designed for SoC development and provided under the GNU GPL license. This library includes a LEON3 processor; an advanced microcontroller bus architecture (AMBA) AHP/APB control; an FPU; an SPI; a UART with first in first out (FIFO); a modular timer unit; an interrupt controller, a 32-bit GPIO port; and memory and pad generators for Virage, Xilinx, UMC, Atmel, Altera, Actel, eASIC and Lattice among others. Some of the said IP cores are discussed further in this section. The library can be obtained from the Gaisler download page [50].

4.1.1 LEON3/FT - High-performance SPARC V8 32-bit Processor

LEON3 is a 32-bit processor core conforming to the IEEE-1754 (SPARC V8) architecture. It is designed for embedded applications, combining high performance (HP) with low complexity and low power consumption. The LEON3 core has the following main features: a seven-stage pipeline with the Harvard architecture, separate instruction and data caches, an MMU, a hardware multiplier and divider, on-chip debug support, and multi-processor extensions. The LEON3 core block diagram is depicted in Figure 9 [42, p. 1177].

4.1.2 Integer Unit

The integer unit includes a signed or an unsigned 32x32 multiplier module (MUL32), and a signed or an unsigned 64/32 divider module (DIV32), and it supports the collection of HP multipliers from the Arithmetic Module Generator at the Norwegian University of Science and Technology (MULTLIB).

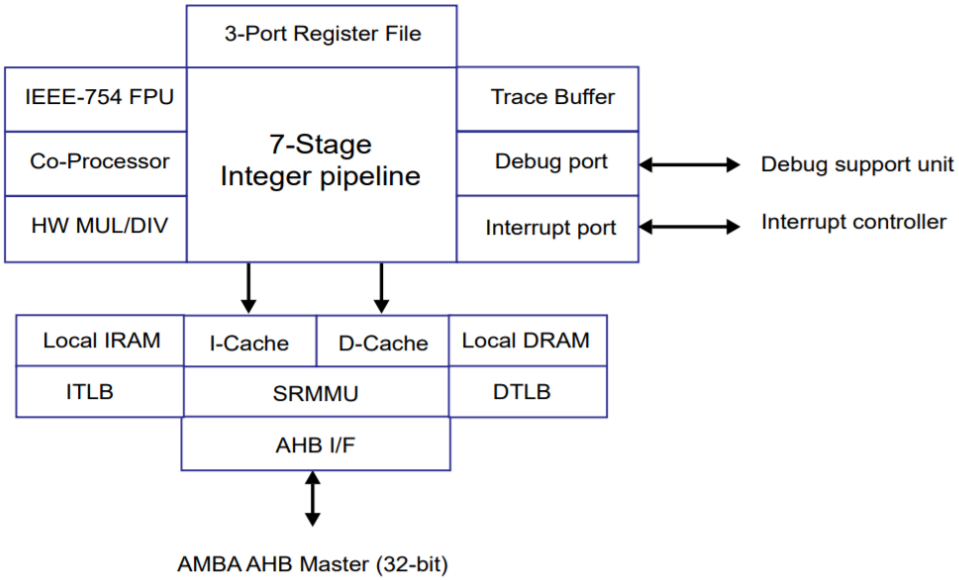


Figure 9: Block diagram of LEON3

4.1.3 The MUL32

The multiplier module takes two 32-bit signed or unsigned numbers and produces a 64-bit result. The MUL32's performance and latency is dependent on its configuration, which has many varieties. The module can be easily configured to perform DSP functions, with MAC operations.

4.1.4 The DIV32

The divider module utilizes the radix-2, non-restoring, iterative division algorithm to perform 64-bit by 32-bit division. The division leaves no remainder and takes 36 clock cycles.

4.1.5 High-Performance IEEE-754 Floating-point Unit

The GRLIB includes HP floating unit (GRFPU), which implements floating-point operations as defined in the IEEE Standard for Binary Floating-Point Arithmetic (IEEE-754) and the SPARC V8 standard (IEEE-1754). The GRFPU supports single- and double-precision floating-point formats, and it can be configured to utilize a non-blocking unit for the execution of divide and square-root operations.

4.1.6 The IEE-754 Floating-Point Lite Unit

The GRFPU-Lite FPU implements the same operation standards as the GRFPU. The key differences between the units are that GRFPU-Lite is not pipelined, and executes one floating-point operation at a time. This results in the GRFPU-Lite utilizing fewer resources at the cost of performance.

4.1.7 Cache Sub-system

LEON3 has a configurable cache system consisting of separate instruction and data caches. Both caches can be configured with one to four sets, 1-256 KiB/way, and 16 or 32 bytes per line. The cache system can be also configured to utilize least-recently-used (LRU), least-recently-replaced (LRR), or pseudo random replacement policies.

4.1.8 Memory Management Unit

A SPARC V8 reference MMU (SRMMU) provides mapping between multiple 32-bit virtual address spaces and physical memory. The MMU can be configured to up to 64 fully associative translation lookaside buffer (TLB) entries per implemented TLB.

4.1.9 Interrupt Interface

LEON3 supports the SPARC V8 interrupt model with a total of 15 asynchronous interrupts. The interrupt interface provides the functionality to both generate and acknowledge interrupts.

4.1.10 Advanced Microcontroller Bus Architecture

An AMBA is a bus architecture standard devised by ARMs. Three key points of this standard are technology independence, electrical characteristics, and timing specifications. The GRLIB includes two cores with AMBA standards, an advanced HP bus (AHB) and AHP.

4.1.11 Advanced High-Performance Bus

An AHB is an HP SoC bus that can connect a maximum of 16 masters and 16 slaves. It has a plug-and-play (P&P) functionality and is provided with an interrupt controller. The LEON3 processor uses one AHB master interface for all data, instruction and MMU table-walk accesses.

4.1.12 Advanced Peripheral Bus

An advanced peripheral bus (APB) is a peripheral bus designed for low bandwidth control accesses, such as register interfaces and on system peripherals. An AHP is the main peripheral bus system of the GRLIB, and it is connected through the AHB.

4.1.13 The AHB and APB Interfaces, Bridges and Controllers

The GRLIB includes many IP cores with AHB and APB interfaces as well as bridges, which increase the accessibility to other types of components. Some of the more notable bridges are AHB to AXI, I2C to AHB, SPI to AHB, PCI to AHB, Uni, and bi-directional AHB/AHB bridges.

Some of the AHB and APB interfaces are the JTAG debug link with an AHB master interface, the single-port RAM/ROM with an AHB interface, the AMBA AHB serial debug interface, the APB UART serial interface, the on-chip SRAM with an EDAC and AHB interface.

The main AHB controller is an AHB controller with P&P support.

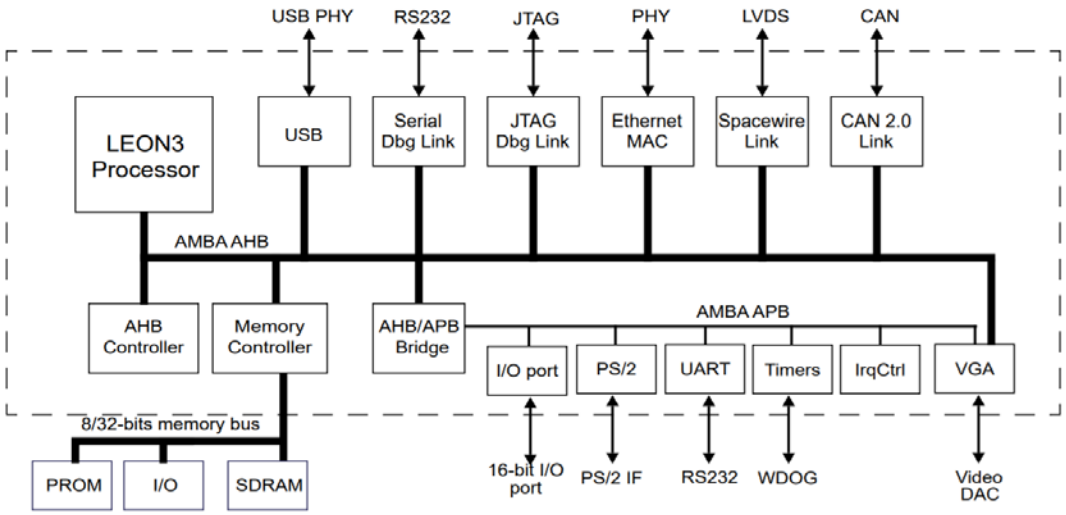


Figure 10: A typical LEON/GRLIB design centered around one AMBA AHB bus and a AMBA APB bus that connects some peripherals cores via an AHB/APB bridge

4.1.14 General-purpose I/O Port

The GP I/O port (GRGPIO) is an I/O port that is scalable from a 2- to a 32-bit width with an optional interrupt support. Each bit in the GRGPIO can be individually set to input or output and can optionally generate an interrupt.

4.1.15 The GRLIB Directory Structure

The main GRLIB directory includes five subdirectories:

- The Bin directory contains all the files that are required for handling of the GRLIB. Graphical user interface for configuration and synthesis (xconfig) files are found in this directory.
- The Boards directory contains folders with timing constraints, pin definitions and placing for most of the common FPGA boards in the market.
- The Design directory contains folders with template designs of LEON3 for different FPGA boards. All the template designs include finished scripts for the synthesis, simulations and FPGA board programming. Scripts are available as single files or compiled through a Make function. The tailored LEON3 is configured according to FPGA constraints, and it is explained in a detailed README.txt for all the FPGA boards.
- The Doc directory contains documentation such as the GRLIB IP Library User's Manual, the GRLIB IP Core User's Manual, the Configuration and Development Guide, and the SPARC V8 manual.

- The Lib directory is the core of the GRLIB. It contains behavioral VHDL descriptions for all the IP cores. Subdirectories are mainly divided into vendors and standards. The tech and techmap subdirectories contain all the necessary packages and hardware descriptions to ensure technology independence.

- The Software directory contains a few sample programs in C for LEON3 testing, GPIO interaction, and APB UART control.

- The additional Netlist directory needs to be downloaded and installed for LEON3 implementations on different FPGA boards.

4.2 LEON3 Tools

Cobham Gaisler provides several tools and utilities for LEON3 development, debugging, testing and simulation. Tools are also available from other vendors; they are important for monitoring, and simulation of LEON3.

4.2.1 Cobham Gaisler Monitor

Cobham Gaisler provides two debug monitors (GRMON2 and GRMON3) for LEON3. GRMON provides a debug environment for real target hardware, and it is used with a LEON3 debug support unit (DSU). This DSU supports connection and communication through numerous interfaces by control through any AMBA AHB master.

4.2.2 GRMON2

GRMON2 is a console-based debug monitor and provides (but is not limited to) the following basic features:

- Read/write access to all LEON registers and memory
- Download and execution of LEON applications
- Breakpoint and watchpoint management
- Remote connection to the GNU debugger (GDB)
- GRLIB P&P support
- Supported debug interfaces: PCI, USB, Ethernet, JTAG, UART, and SpaceWire
- Tcl interactive interpreter support

- Common Flash interface (CFI) compatible Flash PROM programming
- Auto-probing and initialization of LEON peripherals and memory settings

4.2.3 GRMON 3

GRMON3 is a newer version of GRMON with new graphical user interface. Some of the newer features are as follow:

- Execution control with support for multiple CPUs and OS threads
- Context-based virtual memory handling
- Basic execution control such as single-stepping, continuing, and breaking
- An GRLIB SoC system hardware overview
- An optimized SPARC/LEON IU register view
- Memory, CPU register, and I/O register inspection and edit views
- Tcl terminal views with history and tab-completion, among the other things
- Application terminals via UART forwarding

4.2.4 LEON C/C++ IDE for Eclipse

Aeroflex Gaisler provides a plugin for the Eclipse framework, which allows the Eclipse C/C++ development tooling (CDT) to be used for the development of LEON applications. Debugging through the GDB is available; however, it has limited performance. The simulation tool TSIM is available with this plugin.

4.2.5 The TSIM

The TSIM is Cobham Gaislers HP behavioral LEON3 simulator. It emulates an instruction-based, single-processor computer system based on LEON3. Additional custom I/O functions can be added through loadable modules.

4.2.6 LEON Bare-C Cross Compilation System

The Bare-C Cross Compilation system (BCC) allows for cross compilation of C and C++ applications for LEON3, and it is provided with the following components:

- The GNU GCC 7.2.0 C11, C++11 cross-compiler for LEON
- The LLVM (Clang) 4.0.0 C11, C++11 cross-compiler for LEON/LEON-REX
- GNU Binutils (assembler, linker ...)
- The Newlib embedded C library
- The Bare-C run-time library for LEON applications
- The GRLIB peripheral driver library

4.2.7 Other Types of Compilers and Toolchains

Cobham Gaisler provides a multitude of other compilers and toolchain support for the following:

- The RTEMS Cross-compiler system (RCC)
- The VxWorks 7 SPARC architectural port and BSP
- The VxWorks 6.9 SPARC architectural port and BSP
- Linux for LEON
- The ThreadX SPARC port

4.2.8 Xconfig

The Xconfig GUI supports configuration editing, synthesis, implementation and programming of LEON3. Furthermore, it provides options for minimal, GP and HP LEON3 configurations. Custom configurations are also available for almost all of the LEON3 parts.

4.2.9 Online Support

Only the purchased commercial license provides customer support. However, Cobham Gaisler has a LEON3 Yahoo group [37] with active users, where most questions are answered by LEON3 developers. This group has 2,546 members and well over 25,000 messages, the first of which dates back to 1999 with LEON1 as a topic. While the Yahoo group may not be the optimal communication service, all archived topics can offer answers to the most common problems with LEON3. The standard time for receiving a reply is one to three work days.

5 Implementation

This chapter covers the hardware and software used in this study to implement a LEON3 softcore processor on an FPGA. The LEON3 implementation section explains the processes and tools utilized in the implementation of LEON3 on a Zedboard.

5.1 Hardware

A personal laptop and an FPGA board were used for all the implementations and testing of LEON3.

5.1.1 ZedBoard

A Digilent ZedBoard Zynq-7000 ARM/FPGA SoC[51] development board was provided by the robotics department as the main FPGA for LEON3 implementation and testing. The main ZedBoard features are as follows:

- Xilinx Zynq-7000 AP SoC XC7Z020-CLG484
- Dual-core ARM Cortex™-A9
- 512 MB DDR3
- 256 MB Quad-SPI Flash
- On-board USB-JTAG Programming
- 10/100/1000 Ethernet
- USB OTG 2.0 and USB-UART

The XC7Z020 SoC contains Atrix-7 PL, which has the following:

- 85,000 logic cells
- 53,000 (LUTs)
- 106,000 flip-flops

5.1.2 Computer

A personal laptop was used for the implementation and testing of LEON3. The hardware specifications are as follows:

- Intel Core i7-4710HQ 2.5GHz
- 12-GB DDR3L RAM
- 500-GB SSD

It should be noted that the setup was tested with 8-GB of RAM and it had trouble compiling multiple configurations at the same time. The right amount of RAM for extensive research and testing proved to be 12-GB. Furthermore, a USB 2.0 cable was used to connect the FPGA test board with the laptop.

5.2 Software

The configuration, implementation and testing of LEON3 required a number of programs and software. The largest challenge was to make the whole setup work on the Windows 8 operating system.

5.2.1 Vivado Design Suite 2013.4

The Vivado 2013.4 WebPACK (VDS13) [52] version was used because of the LEON3 design for ZedBoard provided by Cobham Gaisler. As mentioned previously, the GRLIB contains finished designs for different FPGA boards. LEON3 for ZedBoard was designed with VDS13; therefore, the easiest way to synthesize, implement, and program LEON3 for ZedBoard was by utilizing VDS13. However, VDS13 does not support GUI in Windows 8. A way around this was to run Vivado in Windows the 7 mode by adding the following lines in the Xilinx\Vivado\2013.4\bin\vivado.bat file:

```
41:     set RDI_PATASK=yes
42:     set __COMPAT_LAYER=WIN7RTM
43:     call "%RDI_BINROOT%/loader.bat" -exec %RDI_PROG% %*
```

An additional shortcut needed to be created for vidado.exe with the following code:

Target: = “C:/Xilinx/Vivado/2013.4/bin/unwrapped/win32.0/vggl.exe C:/Xilinx/Vivado/2013.4/./bin/vivado.bat” and

Start in = “%APPDATA%/Xilinx/Vivado” options.

5.2.2 Xilinx Microprocessor Debugger 2013.4

The Xilinx µP debugger is required to program a LEON3 bitstream to the ZedBoard.

5.2.3 Cygwin

Cygwin [53] is a POSIX-compatible environment that runs natively on Windows. Such an environment was needed to compile and run scripts for the implementation and programming of LEON3. Additional packages for Cygwin are required to be able to run some of the scripts, most notably the Tcl and Tk packages.

5.2.4 Cygwin-X

To run the xconfig GUI, an XWIN server is required in addition to Cygwin. Exporting of the display is done by adding “EXPORT DISPLAY=:0.0” to the Cygwin .bashrc file.

5.2.5 GR Tools

GR tools compromises a full set of cross-compilers, simulators, IDE, CTD, and environments for LEON3. In this setup only, a few selected tools were installed. These include Eclipse Kepler, CTD, LEON3 IDE, TSIM2, GRMON2, Sparc BareC, and BCC.

5.3 LEON3 Implementation on ZedBoard

To implement LEON3 on a ZedBoard, a design found in the GRLIB was used. The given design of LEON3 utilizes DDR3 memory attached to the Cortex-A9 processor system (PS) as the LEON3 memory and is accessed through a custom AHB-AXI bridge (ahb2xi.vhd). The top 256 Mbytes of the DDR3 are mapped to an AHB address space at 0x40000000 - 0x50000000 using an AHB/AXI bridge and the S_AXI_GP0 interface on the

PS. The LEON3 system is clocked at 83.33 MHz, using FCLK_CLK0 from the PS; other frequencies can be used by reconfiguring the PS in Vivado or using GRMON.

5.3.1 Synthesis

To begin the synthesis with VDS13, one would need to start Vivado and source the leon3mp_vivaldo.tcl file. This is done by running the "vivado-launch" target of the main projects Makefile:

```
c/grlib-gpl-2018.3-b4226/designs/leon3-digilent-xc7z020

$ make vivado-launch
```

If the VDS13 GUI is not required, then “make vivado” will start VDS13 without a GUI. To ensure that the synthesis will work some files needs to be edited. On the first time launch VDS13 will yield an error about specified parts that could not be found; this is fixed by editing the following lines in C:\grlib-gpl-2018.3-b4226\designs\leon3-digilent-xc7z020\vivado\leon3mp_vivado.tcl. Red lines represent deleted lines, and green lines represent added lines on the corresponding line number on the right.

```
3   - create_project  ./vivado/ -part  -force
4   - create_fileset -simset
5   - set_property top  [get_filesets ]
  3 + create_project leon3-zedboard-xc7z020 ./vivado/leon3-zedboard-xc7z020 -part XC7Z020CLG484-1 -force
  4 + set_property top testbench [get_filesets sim_1]
6 5   set_property target_language verilog [current_project]
```

```
381     - add_files -fileset  prom.srec ram.srec
    371 + read_vhdl -library work config.vhd
    372 + read_vhdl -library work ahbrom.vhd
    373 + read_vhdl -library work leon3mp.vhd
    734 + read_vhdl -library work leon3_zedboard_stub_sim.vhd
    375 + set_property used_in_synthesis false [get_files leon3_zedboard_stub_sim.vhd]
    376 + read_vhdl -library work testbench.vhd
    377 + set_property used_in_synthesis false [get_files testbench.vhd]
    378 + add_files -fileset sim_1 prom.srec ram.srec
382 379   # Read board specific constraints
```

```
+ read_xdc leon3mp.xdc
+ set_property used_in_synthesis true [get_files leon3mp.xdc]
+ set_property used_in_implementation true [get_files leon3mp.xdc]
+ source stub.tcl
  # Board, part and design properties
- set_property target_simulator  [current_project]
+ set_property target_simulator ModelSim [current_project]
  set_property top_lib work [current_fileset]
  set_property top_arch rtl [current_fileset]
  set_property top leon3mp [current_fileset]
+ set_property board em.avnet.com:zynq:zed:c [current_project]
+ import_files ../../netlists/xilinx/Zynq
```

The C:\grlib-gpl-2018.3-b4226\designs\leon3-digilent-xc7z020\vivado\leon3mp_vivado_run.tcl file needs to be edited in the following way:

```
  synth_design -directive runtimeoptimized -resource_sharing off -keep_equivalent_registers -no_lc -rtl -name rtl_1
- set_property flow {} [get_runs synth_1]
- set_property strategy {} [get_runs synth_1]
+ set_property flow {Vivado Synthesis 2012} [get_runs synth_1]
+ set_property strategy {Vivado Synthesis Defaults} [get_runs synth_1]
  launch_runs synth_1
  wait_on_run -timeout 360 synth_1
  get_ips
  # Launch place and route
- set_property strategy {} [get_runs impl_1]
+ set_property strategy {Vivado Implementation Defaults} [get_runs impl_1]
  set_property steps.write_bitstream.args.mask_file true [get_runs impl_1]
  set_msg_config -suppress -id {Drc 23-20}
  launch_runs impl_1 -to_step write_bitstream
  wait_on_run -timeout 360 impl_1
```

The rest of synthesis, implementation and bitstream generating should run normally after this.

5.3.2 Zedboard Programming

During the programming of a ZedBoard it is recommended to perform *make target 'program-zedboard'* so that XMD is used to program the Bitstream.

5.3.3 Debugging with GRMON

Connecting to the implemented LEON3 on a ZedBoard is done by using the "grmon -digilent -u" command. If a different system frequency is desired for debugging, then it can be selected with the "-freq" flag in MHz. Furthermore, system information and implemented IP cores can be shown with the "sys info" command, and a program compiled with BCC can be uploaded to LEON3 memory with the "load" command. Finally, the execution of the uploaded program is performed with the "run" command.

6 LEON3 Testing

This chapter covers necessary the background for the chosen use case test for LEON3. In this case, the chosen test was to determine whether LEON3 can run a science routine from the 4DSpace project. The test's setup and processes are explained in depth. Moreover, different LEON3 configurations are implemented, and their performance is tested with the chosen use case tests. The resource utilization for each configuration is then displayed and explained. Finally, power consumption and future predictions for each configuration are described.

6.1 The 4D Module

Students at the UiO are participating in the G-Chaser project as part of the 4DSpace project, where a 4D module is installed on an ICI rocket called G-Chaser. The 4D module can be divided into the main-module and the sub-payloads. The main module contains sub-payloads and electronics used to communicate with those sub-payloads. The sub-payloads are divided into radio electronics and measurement electronics. The sub-payloads are divided into the radio electronics and the measurement electronics. The m-NLP hardware and software design are designed by ”Elektronikklabben” (ELAB) at the Physics department, University of Oslo, while the radio electronics is designed by Andøya Space Center. The measurement electronics and software are the focus of this chapter, as they are most relevant to the testing of LEON3.

6.2 The m-NLP System

The *in-situ* measurements in the sub-payloads are done by the m-NLP system. This system can be divided into two main parts: the processing unit and the four antennae or Langmuir probes. The processing unit is described in more detail below, as the aim of this chapter is to analyze and compare the performance between the present utilized hard processor and LEON3.

6.2.1 The Microcontroller Unit

The MCU used for the m-NLP system is a Texas Instruments microcontroller. It uses an ARM Cortex-R4F CPU. The MCU also utilizes multiple communication protocols (SPI, SCI/UART, and I2C) and has DMA modules.

6.2.2 The ARM Cortex-R4F

The CPU[54] used in the MCU implements an ARMv7-R architecture profile, which includes SIMD architecture for integer and floating-point vector operations, as well as Vector Floating-Point version3 (VFPv3) for floating-point computation that is fully compliant with the IEEE 754 standard. This Cortex-R4F processor complies with the AMBA3 protocol and has an AXI interface. In addition the CPU uses a Harvard L1 memory system with memory protection unit (MPU) and error checking and correction (EEC) on all RAM blocks. It can run at 160 MHz.

6.2.3 The Science Routine

The science routine code was produced by the physics department to calculate electron density, platform potential and the correlation coefficient. The science code takes five parameters: The first one is the masking integer, where a three-bit binary number defines which plasma parameters to calculate and the last four parameters are the 16-bit raw values collected from probes by analog-to-digital converter (ADC).

The science code calculates electron density by using collected current and potential from two probes, and it is expressed by:

$$n_e = \sqrt{K \frac{\Delta I_c^2}{\Delta V}}$$

where *K* is the constant:

$$K = \frac{m_e}{2q(q2rl)^2}$$

where m_e is the mass of the electron, q is elementary charge, r is radius of the probe, and l is length of the probe. When $\frac{\Delta I_c^2}{\Delta V}$ is known, the second function in the science code can calculate plasma potential can as follows:

$$V_p = \frac{a}{b} + V_e$$

where a and b are the coefficients for $\frac{\Delta I_c^2}{\Delta V}$ in the form of $ax + b$, and V_e is a correction factor that is dependent on the electron density [55].

The relation $\frac{\Delta I_c^2}{\Delta V}$ unfortunately does not provide an error margin; for that, three or more probes are used to provide a goodness of fit of the fitted line. The last function in the science code uses the linear regression least square method to find the coefficient of determination. A coefficient of determination is a percentage for goodness of fit, where it describes how well the fitted line describes the observed results [56, 57].

In his master thesis, Eirik Nobuki Kosaka [55], calculated the average time of the science routine on the 4D module, and his results are listed in

Table 7. The measurements were based on the average time of 100,000 electron density calculations, and the input values were fixed to keep the code as isolated as possible, to be able to measure the routine itself. These measured times are used as a basis for LEON3 testing in the remaining sections.

It should be noted that Kosaka found that no FPU at a 100 MHz result time was more than 20 times over the minimum required time for calculations in the m-NLP system to achieve in-flight calculations. He also mentioned that measurements taken by Bjørn Lybekk, demonstrated that it was not possible to execute the science routine in the required time without an FPU and that numbers shown in his thesis cannot be blindly used for any system. Considering these two factors, the implementation of LEON3 is tailored to meet the time results presented in

Table 7, even if there is sufficient leeway for slower plasma parameter calculation times.

100 MHz		160 MHz	
FPU	**No FPU**	**FPU**	**No FPU**
0.5690 s	4.0300 s	0.4225 s	3.1369 s

Table 7: Average measured time of the science routine on the 4D module.

6.3 Test Setup

Since LEON3 is a rather capable processor, there was no doubt it would perform as well as or better than Cortex RF4 at the cost of resource utilization. Therefore, the main goal of this test was to find a minimum LEON3 configuration to match the science routine time measured on Cortex RF4. This was done in stages, starting with the bear minimum LEON3 configuration.

6.3.1 Science Code

The science routine code provided by the physics department required some changes for it to be able to run on LEON3. These changes primarily consisted of necessary library adaptation for the BCC compiler. All the code compilations are done with the highest optimization flag “-O3”, to keep it in line with real-world applications. The code utilizes double-precision floating-point variables for most of the calculations. This affects the performance of the code if the CPU has no FPU enabled, since the compiler emulates the floating-point by translating it to integer arithmetic.

To perform an average time of electron density calculation on LEON3, a for-loop was used to call the science routine function with a 0x1 masking integer 100,000 times to calculate electron density. The clock(); function from the “time.h” library was used before and after the loop to obtain the system time from LEON3 timers and to calculate the time used by the for-loop. This setup was tested both with the TSIM and on a ZedBoard, and the results were consistent. All of the tests explained below were uploaded to LEON3 on the ZedBoard through GRMON, and they were run 10 times on a 100 MHz and 160 MHz system frequency to obtain the average time.

6.3.2 Xconfig Configurations

As concluded at the beginning of section 6.3, the science code was tested on different LEON3 configurations starting with the bare minimum until it meets the time of Cortex RF4. All the configurations were edited with the Xconfig tool, which tools has finished three templates for LEON3, minimum, GP and HP. These templates can be used as a starting point for LEON3 customization, where it is possible to add or remove different IPs from the existing templates. To make a custom template, one should select one of the three preexisting templates, save and exit, open the Xconfig tool again and select “custom template”, save it and exit again, reopen Xconfig tool, and then configure the desired LEON3 configuration. Otherwise Xconfig might not allow the selection of new values.

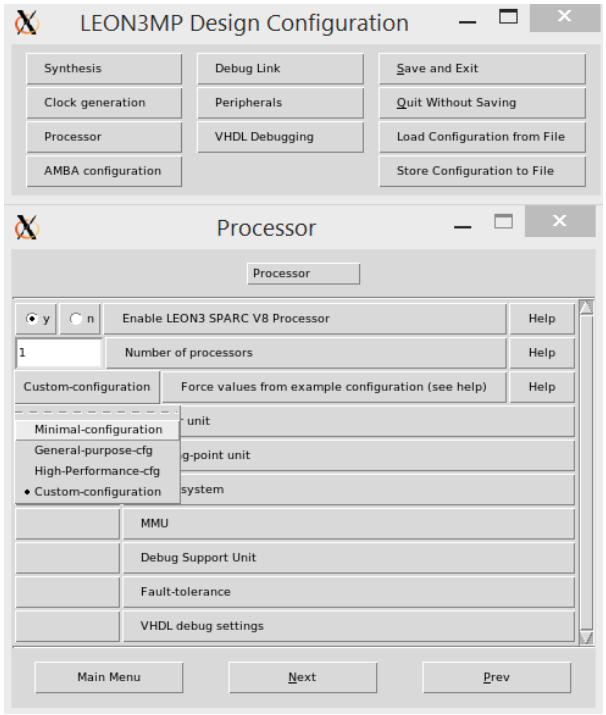


Figure 11: LEON3 custom configuration setup with Xconfig

6.4 LEON3 Configurations

In this section, five different LEON3 configurations that were tested with the science code on the ZedBoard are presented and explained. All the configurations utilize the same setup for DDR memory, attached to a Cortex A9 processor as explained in section 5.3. The first two test times are compared to the Cortex-R4 processor without the FPU times, while the

last three tests are compared to the Cortex R4F processor with an FPU. LEON3 system configurations will contain some of the IP cores displayed in Table 8 [42].

Core	Description
CLKGEN	Clock generator
RSTGEN	Reset generator generating a glitch-free on-chip system reset signal.
AHBCTRL	AHB arbiter/controller
APBCTRL	AHB/APB bridge/controller, which must be included in order to interface peripheral cores such as an interrupt controller and a timer unit.
LEON3/4	LEON3/4 processor
IRQMP	Interrupt controller
GPTIMER	General-purpose timer unit
MEMCTRL	Memory controller providing access to (P)ROM and RAM the GRLIB IP Library contains several memory controllers; it is also possible to include on-chip ROM and RAM by using the AHBROM and AHBRAM IP cores.
DSU3/4	LEON debug support Unit
AHBJTAG/ AHBUART / USBDCL/ GRETH	Debug communication link. AHBJTAG provides an external JTAG link; other examples include AHBUART (serial UART), USBDCL (USB), and GRETH (the Ethernet debug communication link is available as part of the Ethernet MAC core).
APBUART	8-bit UART
GRGPIO	General-purpose I/O port
GRFPU	High-performance IEEE-754 floating-point unit
GRFPU Lite	IEEE-754 floating-point unit

Table 8: LEON3 system IP cores

6.4.1 Minimum Configuration (MC)

The minimum configuration (MC) provided by Gaisler includes the following: eight SPARC register windows in the integer unit; one-cycle load delay; instruction cache 1 * 2 kB,

16 B/line; and data cache 1 * 2 kB, 16 B/line. All the IP cores from Table 8, except for GRFPU and GRFPU-Lite, are included in this configuration. Since no FPU core is included, the science code was compiled with soft-float emulation where the soft-float libraries break up floating-operations into operations that can be used in the integer unit.

Clock Frequency	LEON3	Cortex R4F	Ratio
100 MHz	33.1314 s	4.0300 s	0.12
160 MHz	21.5645 s	3.1369 s	0.14

Table 9: LEON3 minimal configuration times

6.4.2 Minimum Configuration with V8 SPARC Instructions (MCV8)

Since the MC was demonstrated to be too slow, the next step was to attempt to enhance the integer unit with SPARC V8 multiply and divide instructions. These instructions, in a code containing frequent integer multiplications and divisions, can increase performance significantly. An emulated floating-point would also benefit from these instructions. The other way in which to enhance an integer unit is to increase or decrease the pipelined multiplier. Three possible configurations for that are listed in Table 10:

Type	Implementation	issue-rate/latency
2-clocks	32x32 pipelined multiplier	1/2
4-clocks	16x16 standard multiplier	4/4
5-clocks	16x16 pipelined multiplier	4/5

Table 10: Configurations for multiplier

These options include one 32x32 pipelined multiplier, a standard 16x16 multiplier, and a 16x16 pipelined multiplier. Latency in Table 10 represents the time required to finish the instruction, while issue-rate represents the maximum number of instructions that can be issued in each cycle. The 2-clocks and 5-clocks multipliers were tested. The code was compiled with a -V8 flag to enable SPARC V8 multiply and divide instructions. The overall performance increase was approximately 21.15%; however it was not enough to meet the Cortex-R4 performance.

Clock Frequency	LEON3	Cortex-R4F	Ratio
100 MHz	26.1225 s	4.0300 s	0.15
160 MHz	16.3265 s	3.1369 s	0.19

Table 11: LEON3 minimum configuration with V8 instructions 2-clock multiplier times

Clock Frequency	LEON3	Cortex-R4F	Ratio
100 MHz	26.0252 s	4.0300 s	0.15
160 MHz	16.3937 s	3.1369 s	0.19

Table 12: LEON3 minimum configuration with V8 instructions 5-clock multiplier times

6.4.3 Minimum Configuration with GRFPU-Lite (MCGL)

Since the enhanced integer unit was not sufficient, the next step was to try minimal configuration with the lite version of the GLRIB FPU. For this configuration, the integer unit settings were reset to the normal MC settings. For the GRFPU-Lite the non-blocking controller was chosen to increase its performance by 20% by allowing controllers' floating-point load and store instructions to be executed in parallel with floating-point instructions. While, this naturally increases the resource required, it is said to be the best option for ASIC implementations. The times presented below are compared to Cortex-R4F with an FPU enabled.

Clock Frequency	LEON3	Cortex-R4F	Ratio
100 MHz	0.8738 s	0.5690 s	0.65
160 MHz	0.4106 s	0.4225 s	1.02

Table 13: Minimum configuration with GRFPU-Lite times

The increase in performance was 97% at 100 MHz, while at 160 MHz, the increase in performance was even higher at 98%. This increase in performance almost meets the Cortex-R4F's times at 100 MHz and surpasses it at a 160 MHz frequency.

6.4.4 Minimum Configuration with GRFPU-Lite and SPARC V8 (MCV8GL)

The next step was to try to increase the GRPFPU-Lite performance with the enhanced integer unit, as explained in subsection 6.4.2. The 5-clock multiplier was utilized in this configuration.

Clock Frequency	LEON3	Cortex-R4F	Ratio
100 MHz	0.7622 s	0.5690 s	0.76
160 MHz	0.4083 s	0.4225 s	1.03

Table 14: Minimum configuration with GRFPU-Lite and SPARC V8 times

The enhanced integer unit did not seem to increase performance much, only by a fraction. Furthermore, the 100MHz configuration still did not meet the Cortex R4F time but was within acceptable range.

6.4.5 Minimum Configuration with GRFPU (MCG)

As the GFRPU-Lite with V8 instructions did not meet the 100MHz requirements, an HP FPU, namely GRFPU, was added to the minimal configuration.

Clock Frequency	LEON3	Cortex-R4F	Ratio
100 MHz	0.4233 s	0.5690 s	1.34
160 MHz	0.2645 s	0.4225 s	1.59

Table 15: Minimum configuration with GRFPU times

The full GRFPU increased the performance by 99% for the 100-MHz frequency and by 98% for the 160-MHz frequency. This configuration surpassed the time of Cortex R4F, demonstrating that the LEON3 GRFPU is more than capable of executing floating-point tasks.

6.4.6 LEON3 Performance Summary

This user case test for LEON3 presented interesting challenges and produced alluring results. As mentioned in subsection 3.7.2, LEON3 does not have the best integer performance compared to the other softcore processors; however, it outperforms them in floating-point performance. This is also seen in the result produced by this test. The MC LEON3 was more than eight-times slower than the Cortex-R4, but it outperformed the Cortex R4F 1.4-times with the GRFPU enabled. Since Cortex-R4(F) is a hard processor, the fact that LEON3 on an FPGA managed to meet half of the time requirements is quite impressive.

6.5 Resource Utilization

In this section, all the configurations described in section 6.4 will be compared and explained in terms of resource utilization on the ZedBoard. To conduct a more realistic resource analysis, the configurations were reconfigured to exclude APBUART and DSU3/4, as the main purpose of those two IP cores is debugging support. Two main graphs for each configuration are displayed. The first one is for the total resource utilization of slice LUTs,

register LUTs, and memory and clock components in the ZedBoard. This graph is mainly for displaying different size of LEON3 for the different configurations.

The second graph is of the FPGA primitives in LEON3 implementations. This information can provide an insight into size of a potential ASIC implementation of LEON3.

The 7-series Xilinx FPGAs use a configurable logic block (CLB) as the main logic resource that consists of two slices. Each slice is composed of four six-input function generators or look-up tables (LUT6) and eight storage elements. The function generators are implemented as LUTs with six independent inputs and two independent outputs. Each LUT6 can optionally be divided into two five-input LUT5s. Any function that uses less than five input is implemented with the LUT5s and unused inputs. This is necessary to understand when studying the primitives' graph. The primitives, such as LUT4, LUT3, LUT2, and LUT1, are implemented with LUT6 and LUT5 primitives in the ZedBoard. On other technologies, such as ASICs, these LUTs can be achieved with fewer resources and a smaller die size. Some of the other types of primitives included in the graph are explained in the paragraph below.

First, the FDRE primitive is a D flip-flop with clock enable and a synchronous reset. Second, the BIBUF primitive is a simple bi-directional buffer with a high impedance capability. Third, the MUXF7 primitive is a two-to-one LUT multiplexer with general output; MUXF7 are mostly used in combination with double LUT6s to make a seven-input function generator.

6.5.1 Minimum Configuration Resource Utilization

The MC of LEON3 is the same as that explained in subsection 6.4.1 without the UART and DSU3/4 IP cores. The resource utilization of this implementation seems to be in accordance with the expected result from previous research of LEON3 implementations on different FPGA boards. The LEON3 processor IP core utilizes 7% of the total SLICE LUTs on the Zedboard; the rest of the 2% slice LUTs are all of the APB/AHB bridges and peripherals, timers, interrupt controller and GPIOs.

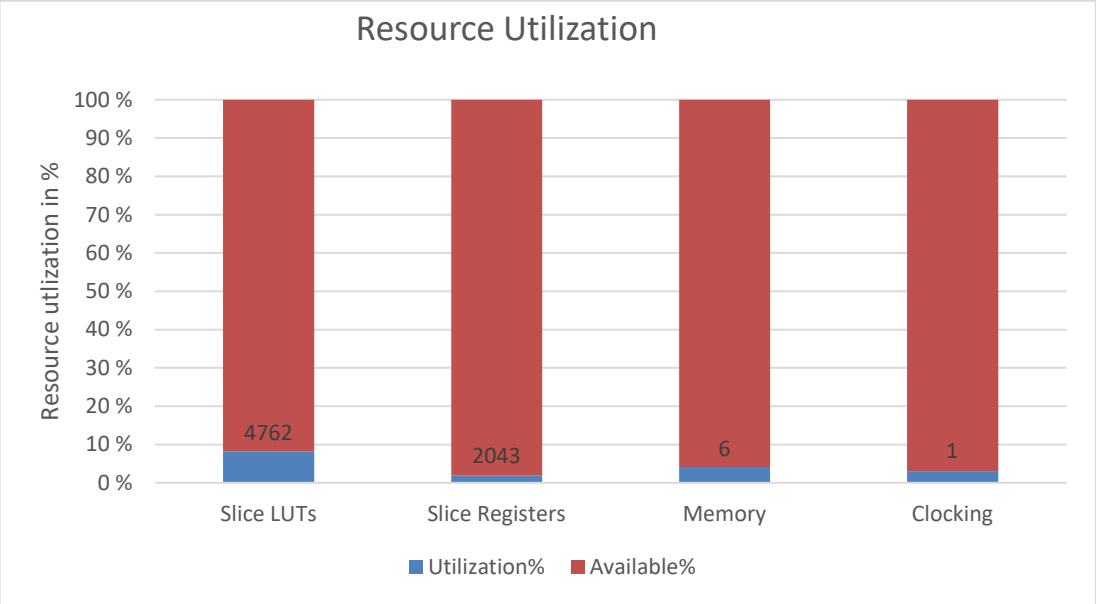


Figure 12: MC resource utilization

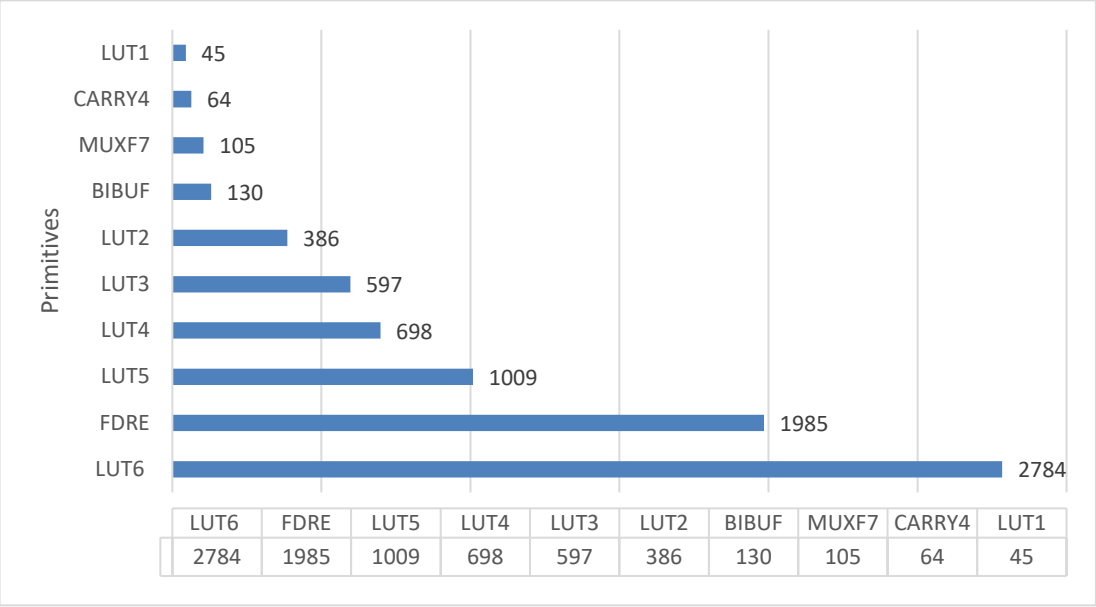


Figure 13: Primitive resource utilization in MC

6.5.2 Resource Utilization for MCV8

This configuration corresponds to the configuration tested in subsection 6.4.3. A slight increase in the resource utilization is noticeable. The LEON3 core utilized 10% of the total slice LUTs, 8% of which was utilized by the LEON3 IP core. The utilization increased by 1% from the previous configuration, where the MUL/DIV IP cores added approximately 0.5% of the slice LUTs.

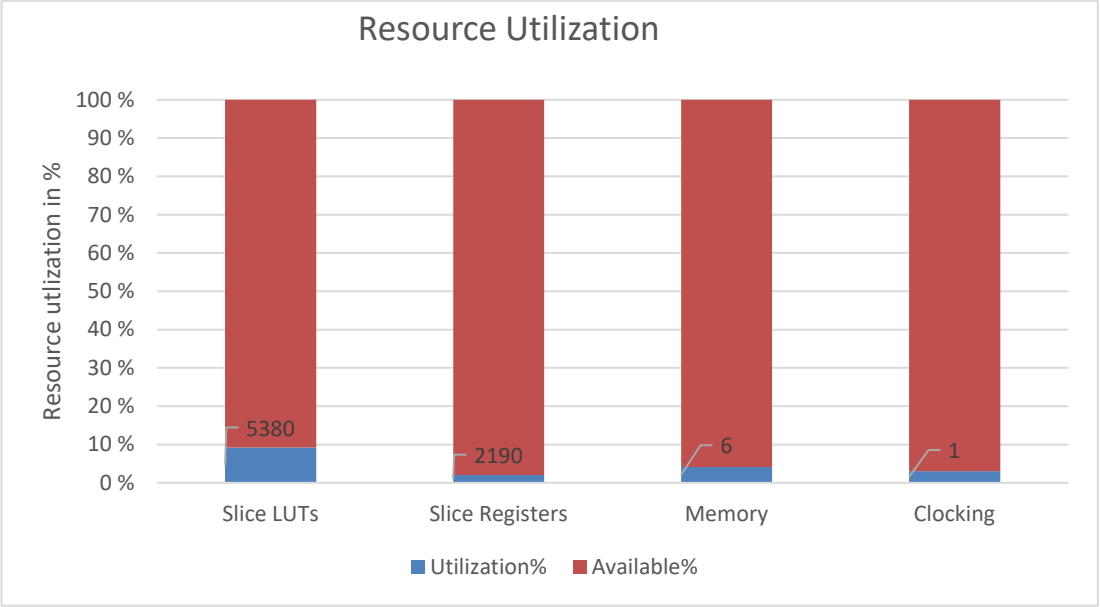


Figure 14: Resource utilization for MCV8

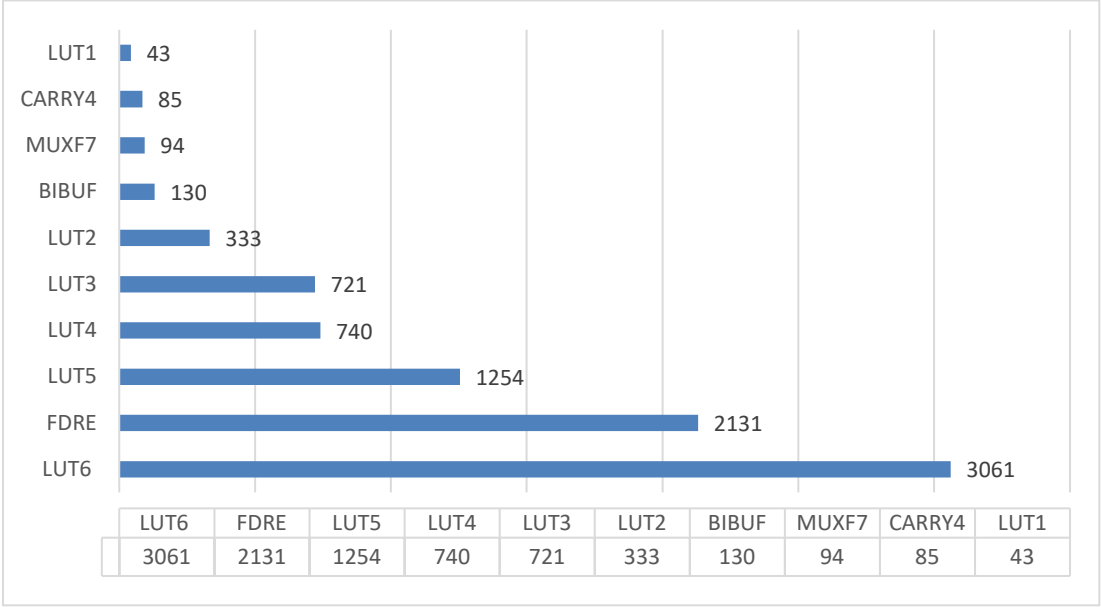


Figure 15: Primitives' resource utilization for MCV8

6.5.3 **Resource Utilization for MCGL**

The GRPFU-Lite configuration has the same setup as the one mentioned in subsection 6.4.3. The increase in resource utilization in this configuration is more noticeable than in the previous one. The total increase in slice LUTs was about 6%, where the LEON3 core contributed 4% of that increase. This 4% stemmed predominantly from the GRFPU-Lite IP core. A significant increase was also observed in other types of primitives such as RAMD32, where distributed memory was utilized in the GRFPU-Lite.

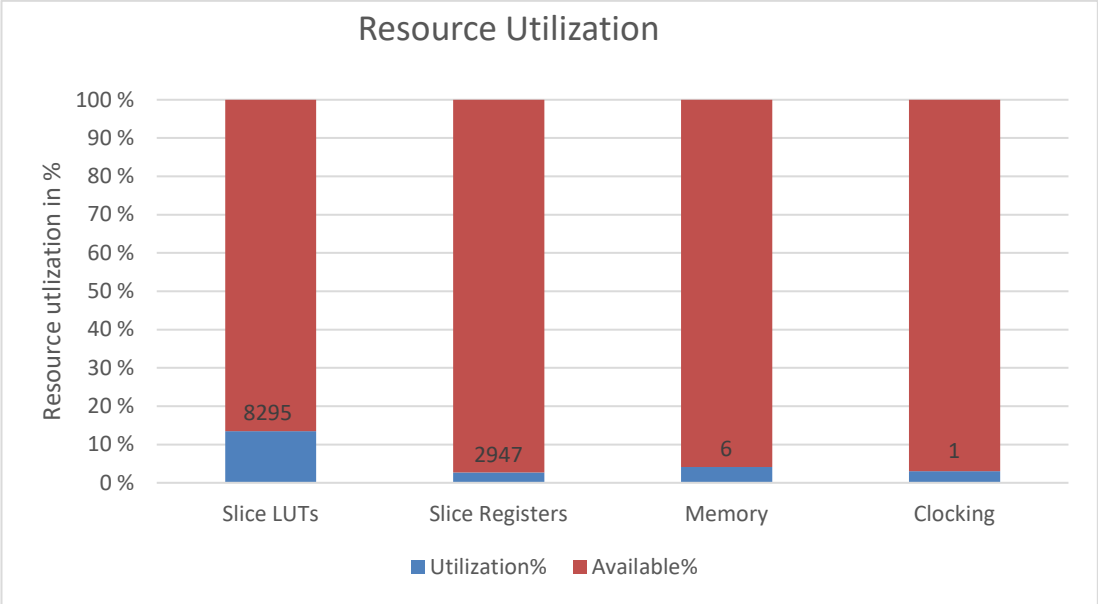


Figure 16: Resource utilization for MCGL

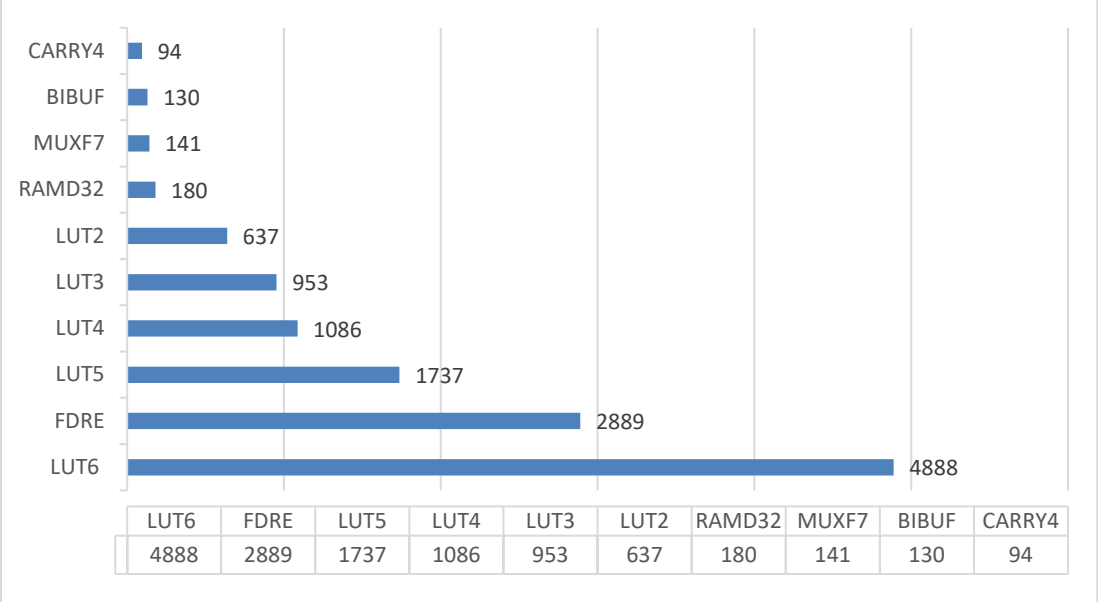


Figure 17: Primitives' utilization for MCGL

6.5.4 **Resource Utilization for MCGLV8**

This configuration corresponds to the configuration in subsection 6.4.4. In subsection 6.5.2, the integer unit was enhanced with SPARC V8 instructions, and the resource utilization increased by 1%. In this configuration, it was reasonable to expect the 1% increase in resource utilization, and as seen in Figure 18 below, this was precisely what occurred. This illustrates that IP cores have a consistent resource utilization with different configurations.

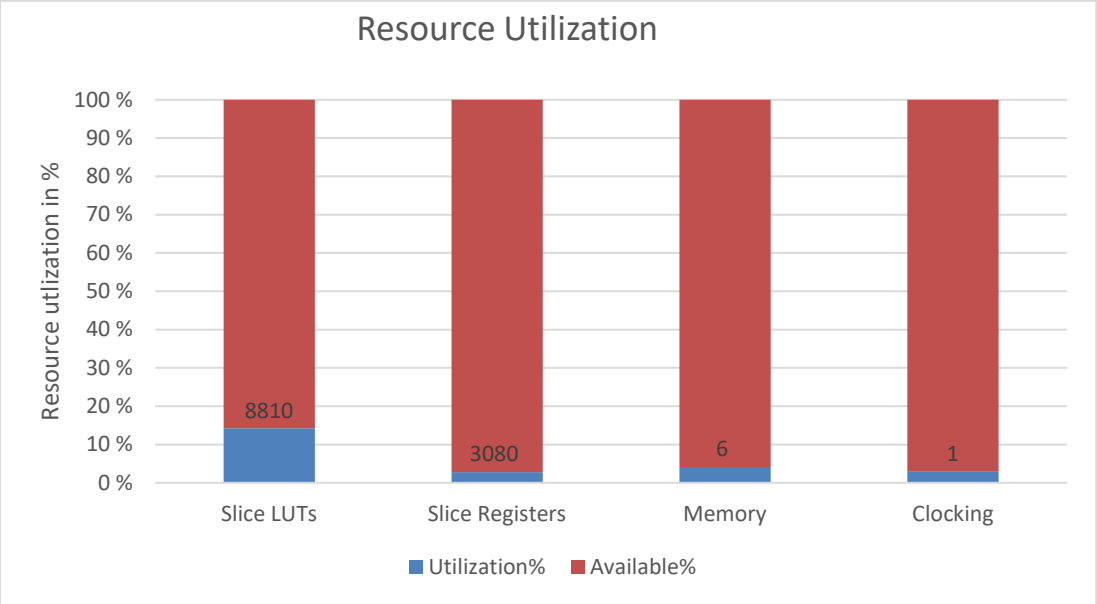


Figure 18: resource utilization for MCGLV8

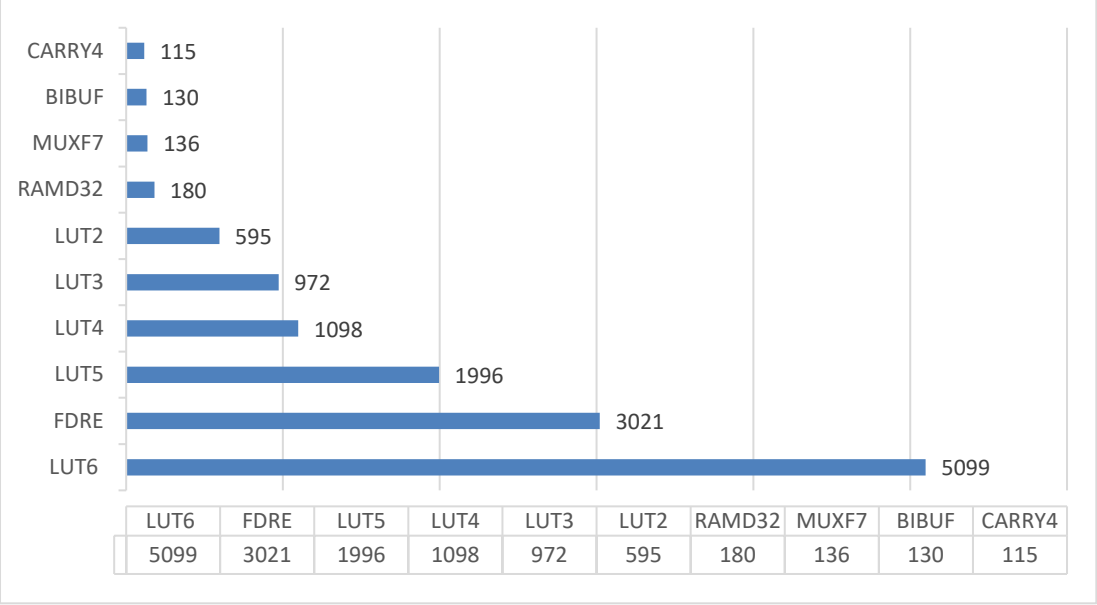


Figure 19: Primitives' utilization for MCGLV8

6.5.5 Resource Utilization for MCG

This configuration corresponds to the one described in subsection 6.4.5. The resource utilization doubled compared to the GRFPU-Lite configuration. The whole implementation used approximately 33% of the total LUTS, 31% of which whole is accounted by the LEON3 IP core. The actual GRPFU IP core used 24% of the total LUTs resources; this is almost six times more than the GRFPU-Lite IP core. There was a drastic increase in the lower input

LUT's primitives and CARRY4 primitives. This would be ideal for ASIC implementations, where lower input LUTs can be achieved with smaller die size.

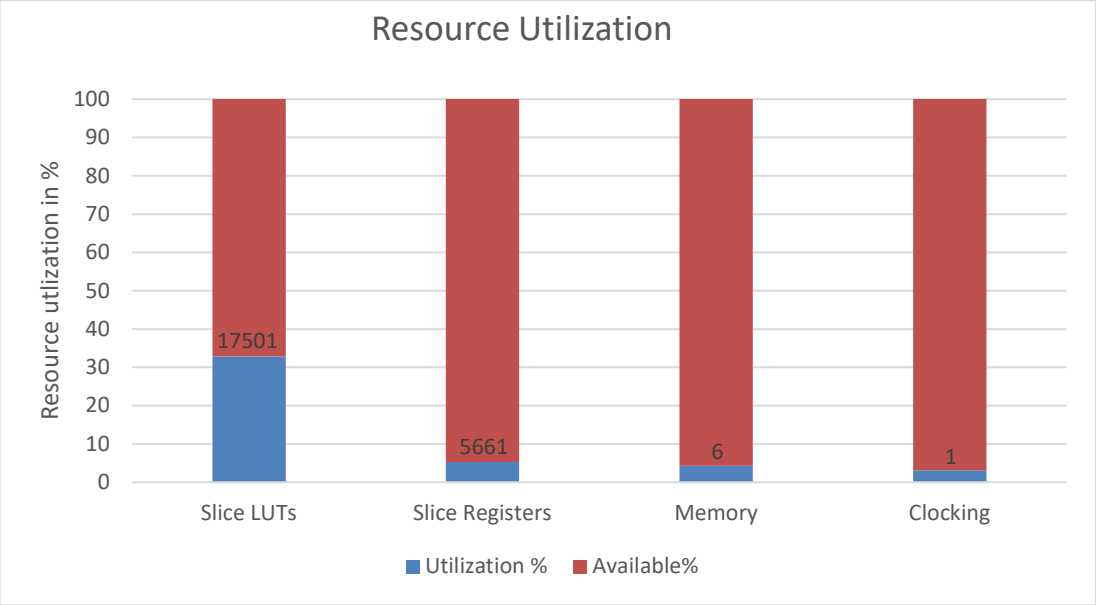


Figure 20: Resource utilization for MCG

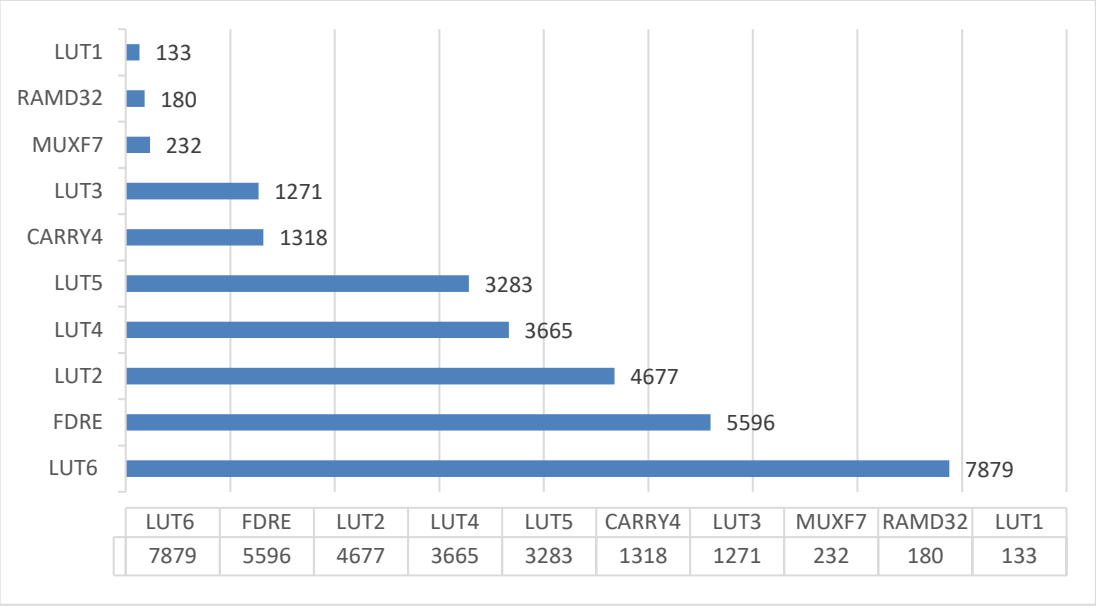


Figure 21: Primitives' utilization for MCG

6.5.6 Resource Utilization Summary

The different resource utilization for configurations seem to be consistent with research conducted by others. Figure 8 illustrated that the number of slice LUTs for LEON3 with an enabled FPU and cache on the FX70T is about 41% or 18,000 LUTs. Furthermore, LEON3 with a GRPFU enabled consumes 33% or 17,500 slice LUTs on the ZedBoard;this seems to correspond well to the FX70T utilization. The 500 LUT' difference stems mainly from the different CLB technologies, where the ZedBoard is a newer technology that allows for less LUT' utilization. The fact that the resource utilization seems to be consistent on different FGPA technologies demonstrates that a LEON3 implementation will be roughly the same when deployed on different FPGAs.

The MC utilized 3.6 times less LUTs than the configuration with GRFPU, while the configuration with the GRFPU-Lite consumed only 1.7 times more LUTs than the MC. Figure 22 displays a resource utilization summary, and Figure 23 illustrates the primitives' utilization summary for LEON3 configurations.

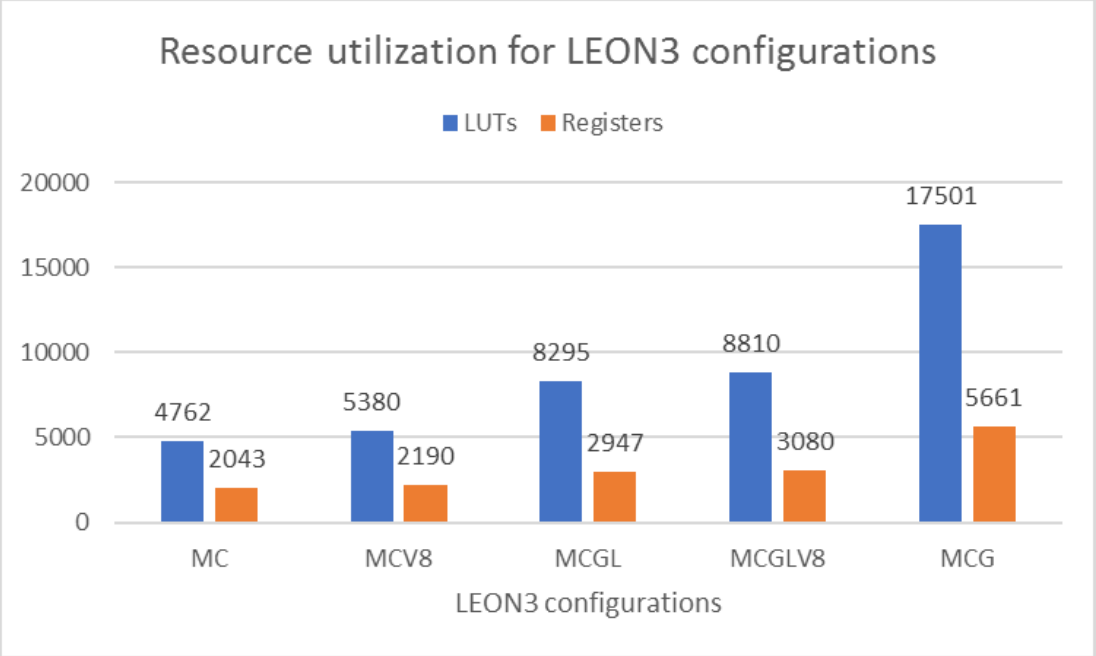
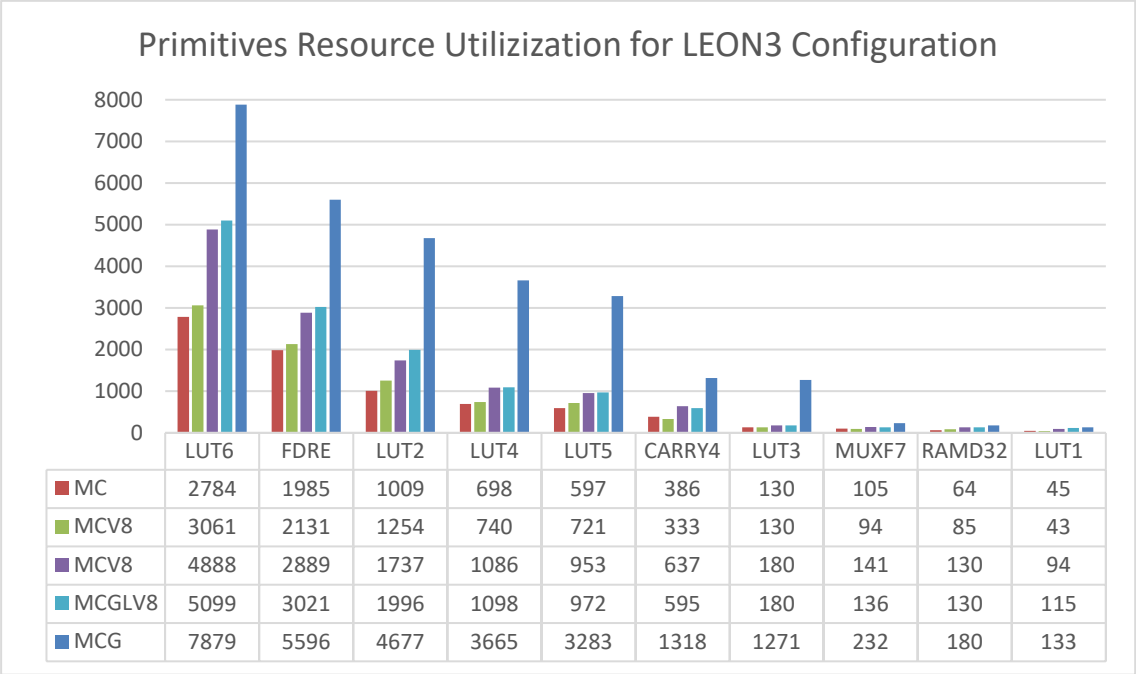


Figure 22: Resource utilization for LEON3 configurations



	LUT6	FDRE	LUT2	LUT4	LUT5	CARRY4	LUT3	MUXF7	RAMD32	LUT1
MC	2784	1985	1009	698	597	386	130	105	64	45
MCV8	3061	2131	1254	740	721	333	130	94	85	43
MCV8	4888	2889	1737	1086	953	637	180	141	130	94
MCGLV8	5099	3021	1996	1098	972	595	180	136	130	115
MCG	7879	5596	4677	3665	3283	1318	1271	232	180	133

Figure 23: Primitives' resource utilization for LEON3 configurations

6.6 Power Consumption Analysis

In this section, the power consumption of LEON3 implementations with different configurations is analyzed. The purpose of this analysis is primarily to draw an abstract perspective of relative changes in LEON3 power consumption with corresponding configurations. Since the VDS13 analyzes simulated power consumption with ideal parameters on the actual ZedBoard, only the relative changes between the different configurations can be considered for future LEON3 ASIC implementations.

Hussain , Hoffmann , Ahonen, and Nurmi [58, p. 23] compared the same design on a 28-nm FPGA and on a 90-nm ASIC, and the dynamic power consumption for the ASIC implementation was about one-fourteenth that of an FPGA. In their research, Kuon and Rose [59, p. 9] found similar results during a comparison of a 90-nm FPGA and a 90-nm ASIC implementation. The dynamic power consumption was 13 to 15 times less on the latter implementation than on the former. These results can offer an approximate idea of the amount of power LEON3 would consume on a future ASIC implementation.

Most of the simulated power reported by the VDS13 comes from the Zynq processing system (PS7) in the ZedBoard. The logic, signal, and clock power are consumed mostly by

the LEON3 core. In this power analysis, commercial grade temperature and typical process settings were used. Two cases were measured for each LEON3 configuration: the first one at a 100-MHz system frequency and the second at a 160-MHz system frequency. The system frequency was changed by adjusting the FCLK_CLK0 in the “grlib\designs\leon3-digilent-xc7z020\leon3_zedboard_stub.tcl” file and re-implementing LEON3 for each frequency. Two graphs for each LEON3 configuration are presented, one representing power consumption for the 100-MHz LEON3 and the second for the 160-MHz LEON3.

6.6.1 Minimum Configuration Power Consumption

This configuration corresponds to the same one used in subsection 6.5.1. The power on the chip was mainly dynamic, and most of the dynamic power was consumed by the PS7. The small remaining percentage was consumed by the logic, clocks, and signals. The total LEON3 IP core dynamic power consumption was approximately 0.033 W, and the other interfaces, timer, and I/O used less than 0.01 W. With an increase in the frequency, a higher power consumption in signals and clocks was expected. The power consumption for LEON3 at 160 MHz increased by approximately 0.029 W, and most of the increase was in the clocks and signal, as expected.

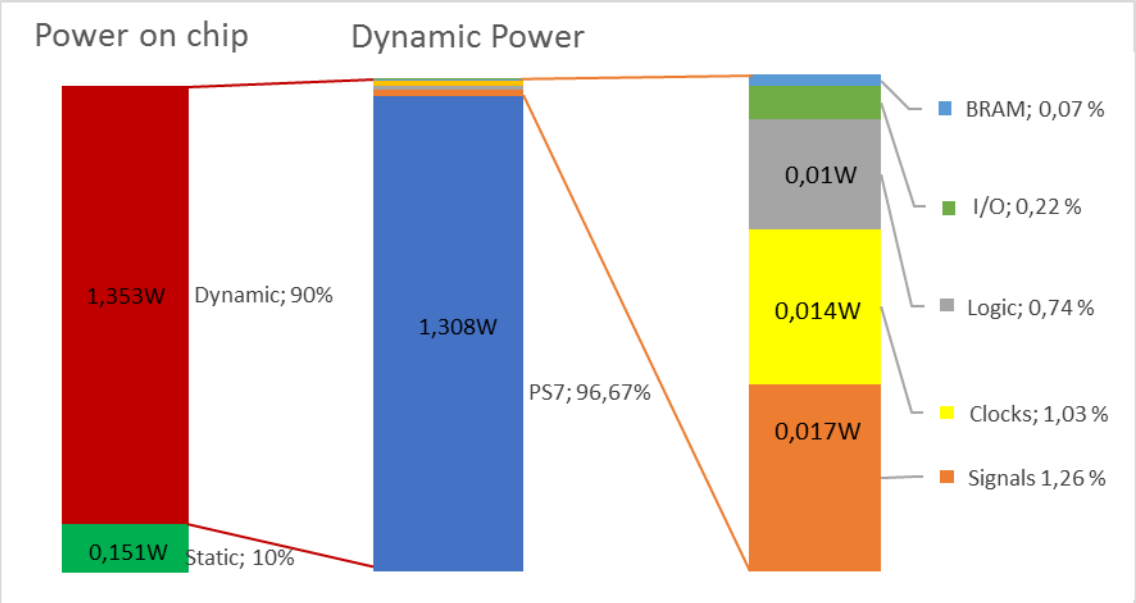


Figure 24: Minimum configuration power consumption at 100 MHz

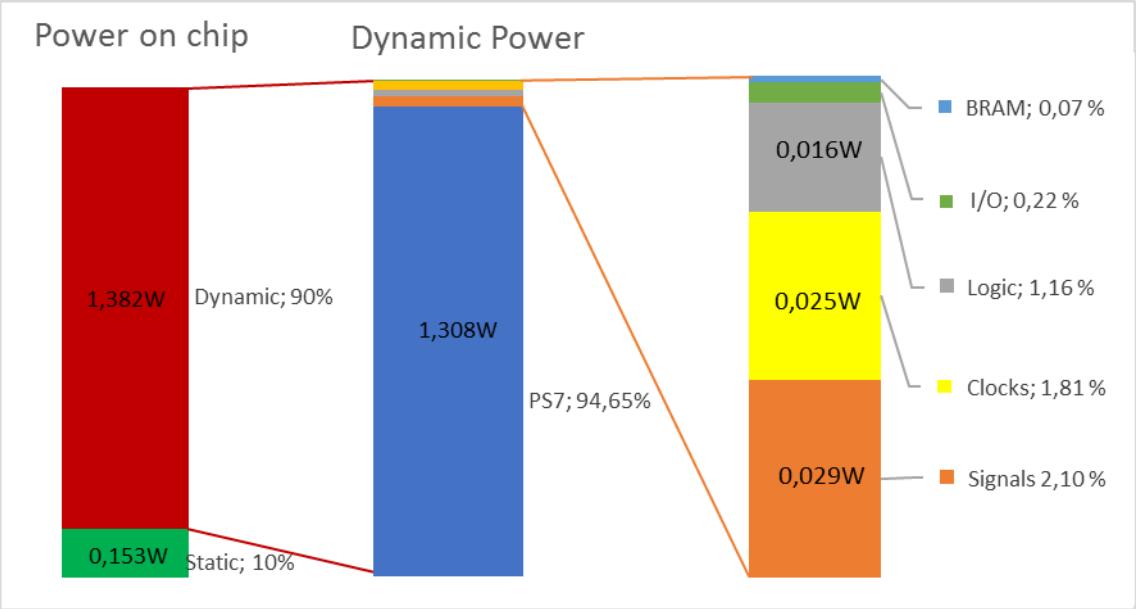


Figure 25: Minimum configuration power consumption at 160 MHz

6.6.2 **Power Consumption for MCV8**

The configuration used for this analysis corresponds to the one used in subsection 6.5.2. The enhanced integer unit increased power consumption slightly, by 0.01 W, where 30% of the power was consumed by DSP units on the ZedBoard. Furthermore, the LEON3 IP core consumed 0.038 W, and the rest of the interfaces consumed the same amount as in the previous configuration. At 160 MHz, the LEON3 implementation increased dynamic power consumption by 0.035 W; most of the increase was within the signal, clocks, and DSP units.

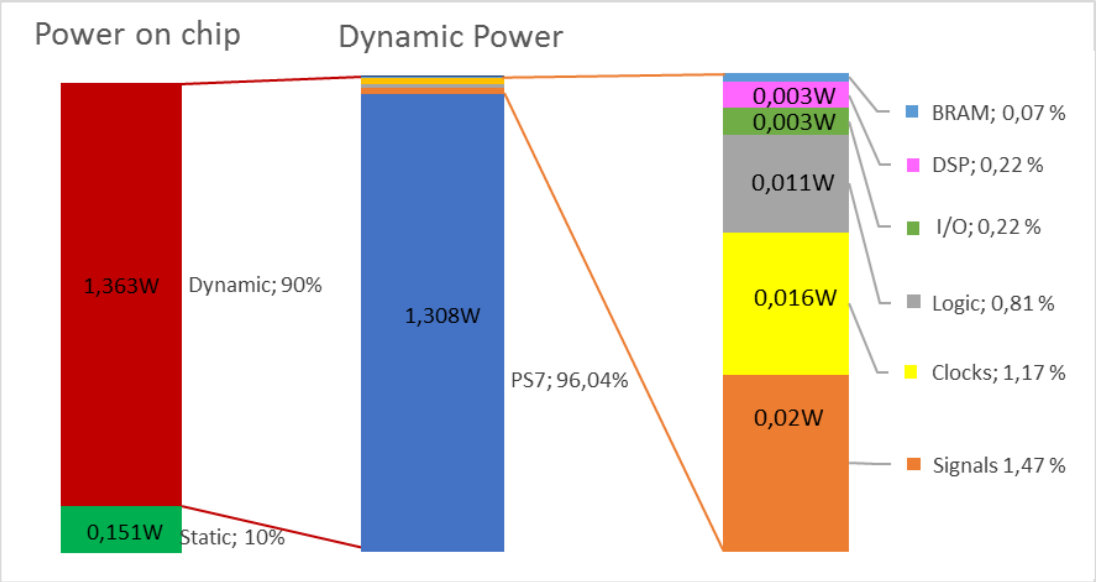


Figure 26: Power consumption for MCV8 at 100 MHz

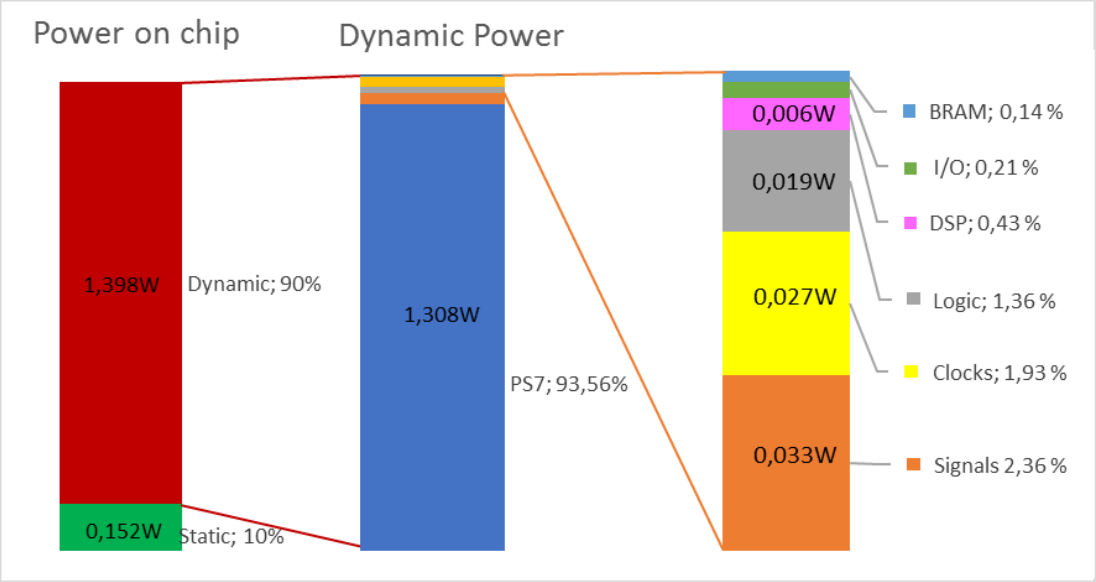


Figure 27: Power consumption for MCV8 at 160 MHz

6.6.3 **Power Consumption for MCGL**

This configuration corresponds to the one used in subsection 6.5.3. The added GRFPU-Lite core increased dynamic power consumption by 0.04 W. The LEON3 IP core used a total of 0.085 W, where the GRFPU-Lite consumed 50% of its power. As expected, the largest increase was in the signals and logic. At 160 MHz, this dynamic power further increased by 0.07 W, and most of the increase came from signals, logic, and clocks.

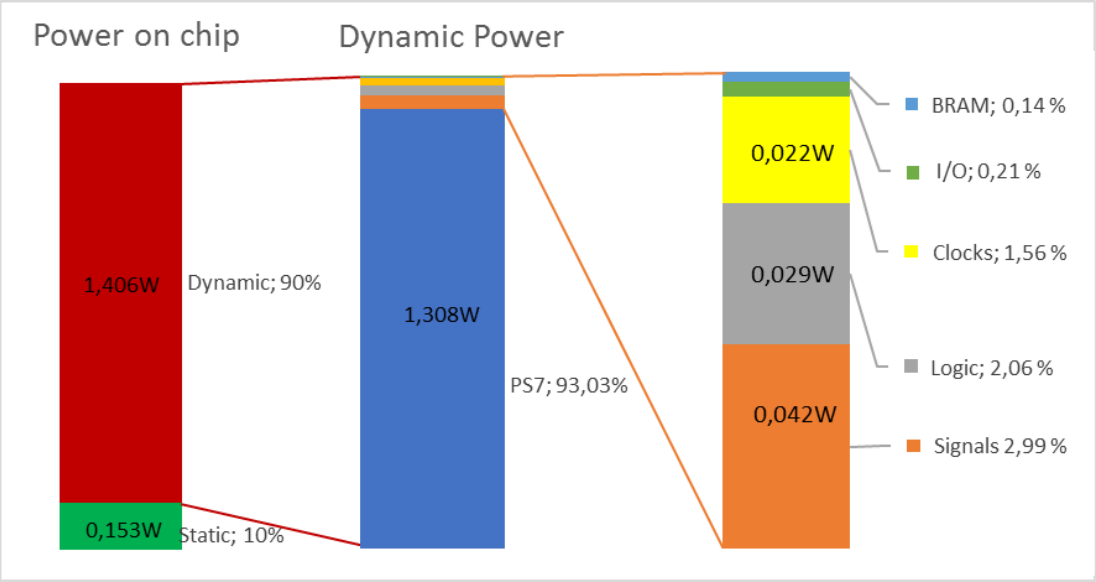


Figure 28: Power consumption for MCGL at 100 MHz

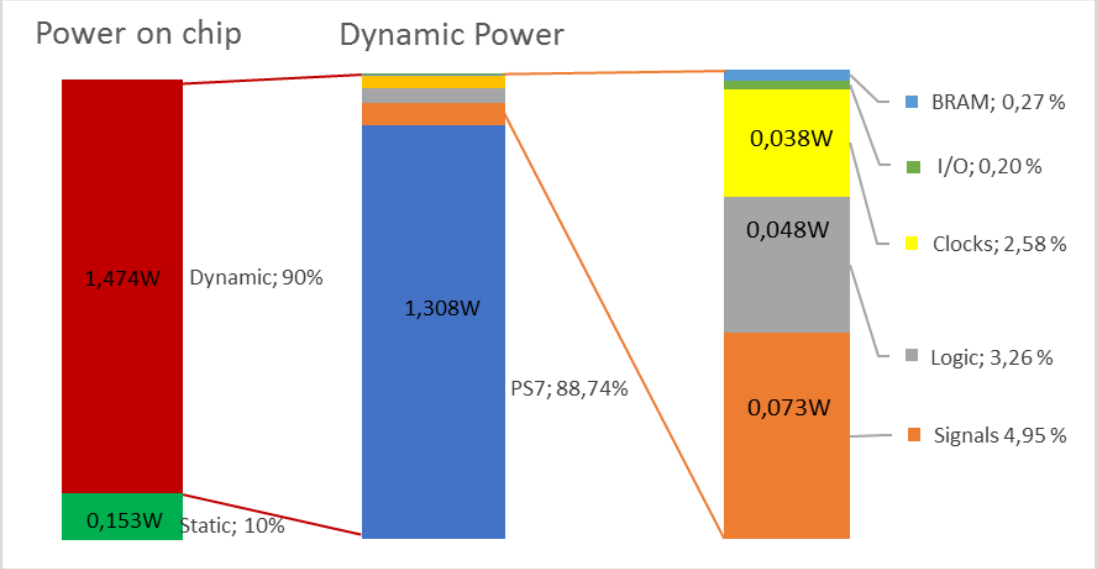


Figure 29: Power consumption for MCGL at 160 MHz

6.6.4 **Power Consumption MCGLV8**

The configuration used in this analysis corresponds to the one used in subsection 6.5.4. The enhanced integer unit added a mere 0.007 W increase in dynamic power and a small increase in power consumption by DSP units. This result is similar to the one analyzed in subsection 6.6.2. The LEON3 implementation at 160 MHz consumed an added 0.063 W in dynamic power.

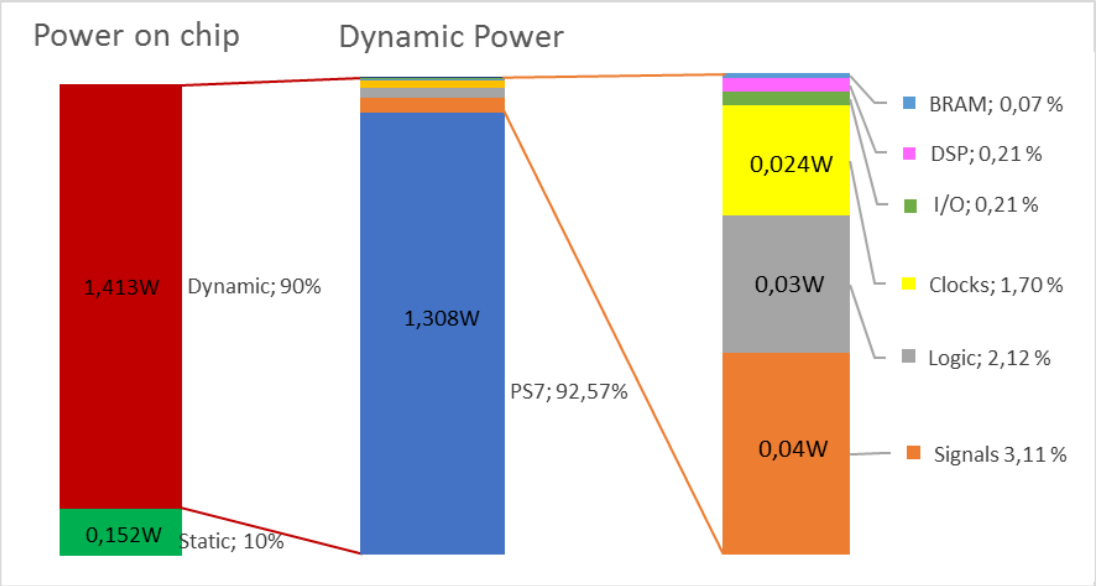


Figure 30: Power consumption for MCGLV8 at 100 MHz

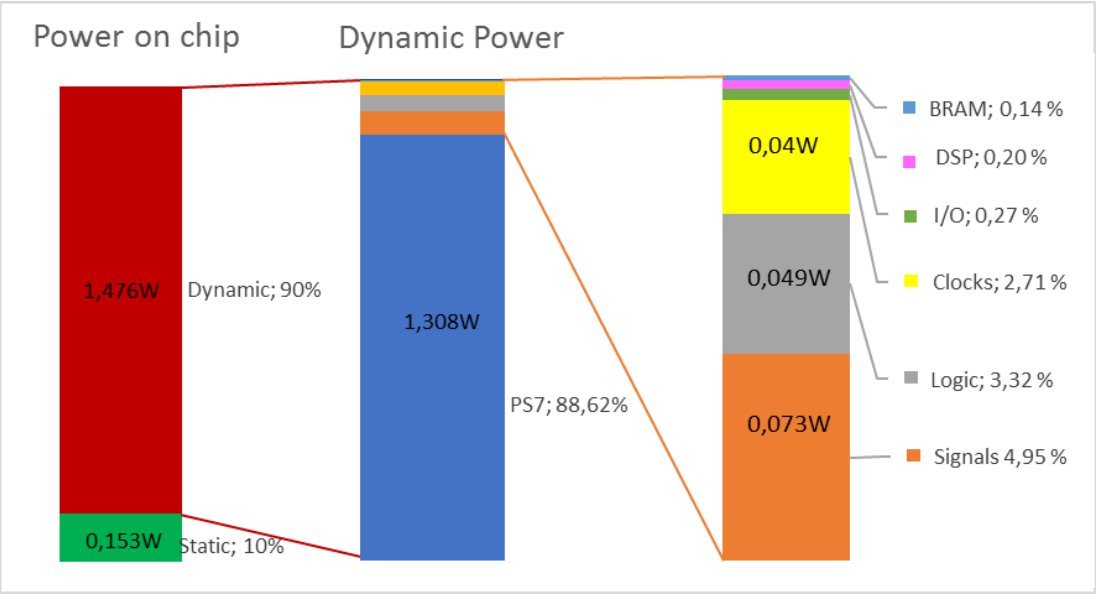


Figure 31: Power consumption for MCGLV8 at 160 MHz

6.6.5 **Power Consumption for MCG**

The final configuration that was analyzed corresponds to the one used in subsection 6.5.5. The GRFPU IP core increased the dynamic power consumption by 0.081 W, and the clock, logic, and signal power consumptions almost doubled compared to the MCGLV8 configuration. With the FPU, LEON3 consumed 0.173 W, of which 0.136 W was used for the GRFPU. At 160 MHz, the LEON3 IP core consumed 0.291 W of dynamic power.

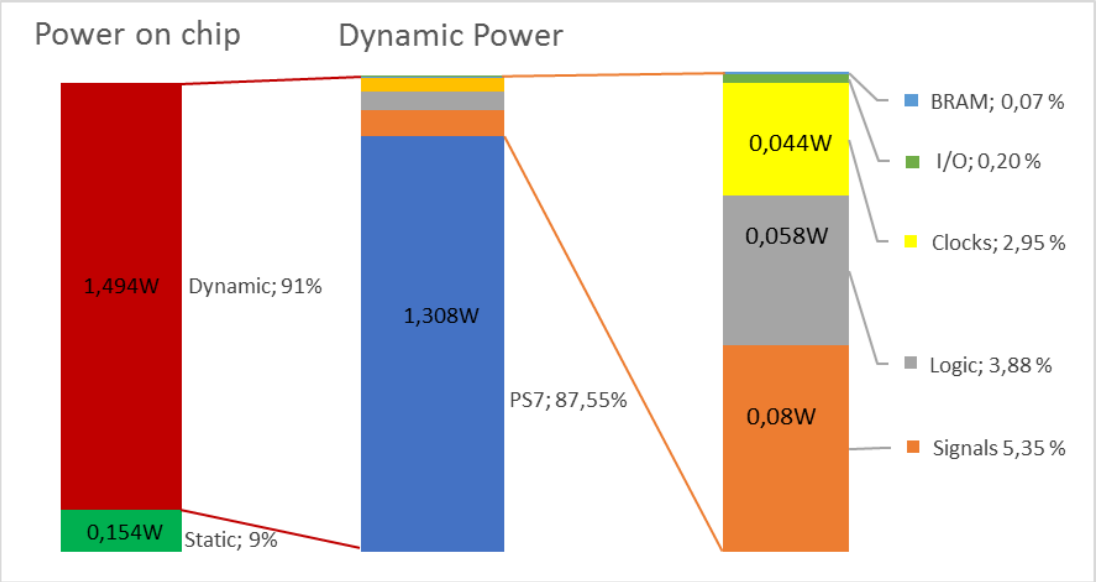


Figure 32: Power consumption for MCG at 100 MHz

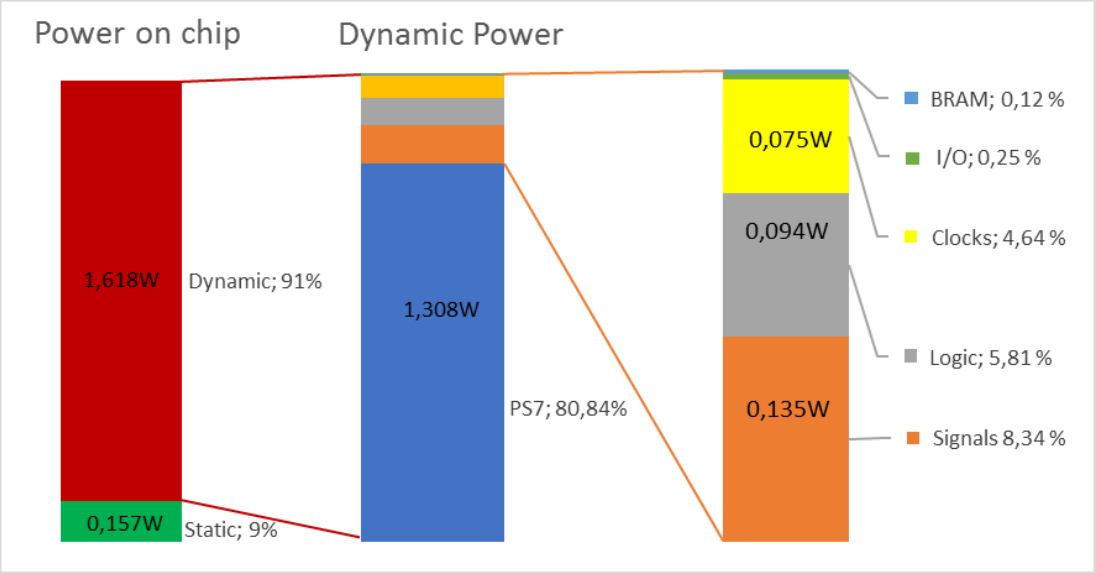


Figure 33: Power consumption for MCG at 160 MHz

6.6.6 **Power Consumption Summary**

This power consumption analysis for LEON3 should be considered only as an insight into LEON3s power consumption and not as an exact value representation. More extensive testing and simulating must be done to obtain exact values of LEON3 power consumption. Nevertheless, rough estimates can be made from the results in this analysis. Figure 34 displays a summary of the power consumption of LEON3 configurations.

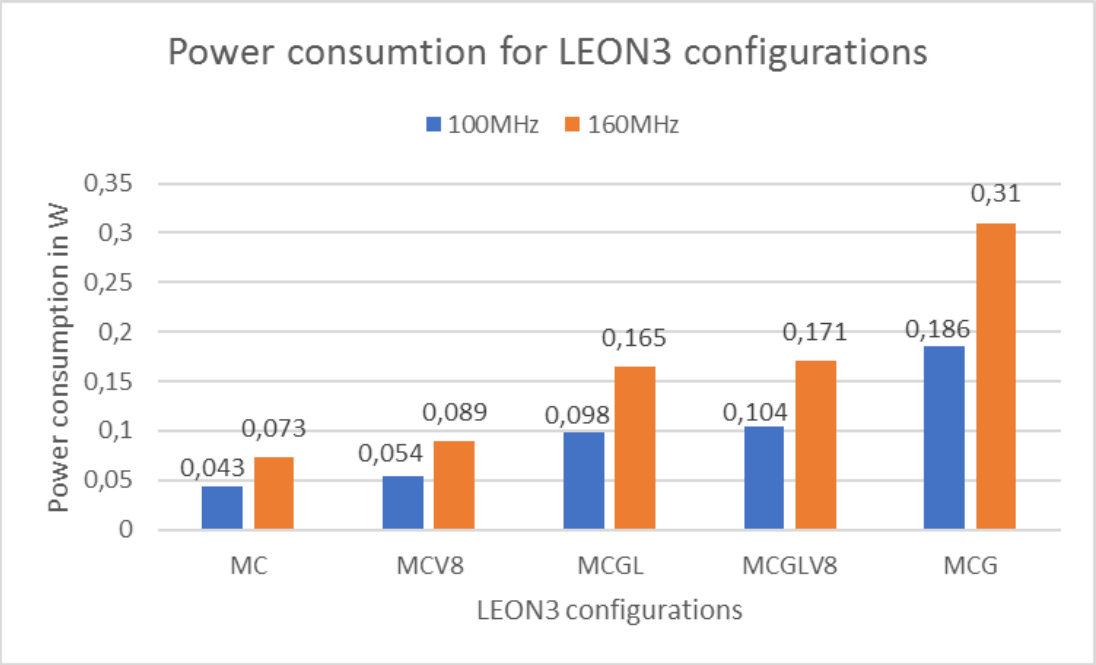


Figure 34: Power consumption for LEON3 configurations

In the MC, the LEON3 IP core together with the rest of its interfaces and bridges consumed 0.042 W at 100 MHz and 0.074 W at 160 MHz. The MC with GRFPU-Lite increased power consumption by 2.3 times for both frequencies, while the final configuration consumed 4.4 times more power than the MC for both frequencies. The increase rate in power consumption from 100 MHz to 160 MHz for all of the configurations are approximately 1.7. Finally, the linear relationship between frequency and power consumption demonstratets that dynamic power dominates the total effect.

6.7 LEON3 Test Summary

In this user case test of LEON3, most the of results coincided with those from research the mentioned in previous chapters. The integer performance of LEON3 is far worse than the Cortex-R4 processor; however, it outperforms the latter processor in floating-point performance. The resource utilization of LEON3 seems to be consistent with the numbers from other research.

For this user case specifically, the LEON3 configuration with a GRFPU is a clear winner. The MCGLV8 should also be considered as an option. The latter configuration has the best trade-off in performance increase versus resource utilization and power consumption. Although the MCGLV8 at 100 MHz did not quite meet the performance requirements with a

small modification in the science code and during further testing of configuration optimization, the MCGLV8 should be more than capable of meeting the performance requirements. Moreover, if the science code were to be rewritten to use an integer instead of a floating-point, then the MCV8 could be a promising option for LEON3 configuration.

In Table 16, a LEON3 test summary of this chapter is displayed. In this table, the performance ratio is a ratio of the science routine execution time on Cortex-R4F and to that on LEON3. The power consumption column displays the dynamic power consumption of the LEON3 core with all of the interfaces and bridges, excluding the PS7 power. Finally, resource utilization column summarizes the LUTs used in each of the configurations in this chapter.

Configuration	Performance ratio		Power consumption in W		Resource Utilization in LUTs
	100 MHz	160 MHz	100 MHz	160 MHz	
MC	0.12	0.14	0.043	0.073	4,762
MCV8	0.15	0.19	0.054	0.089	5,380
MCGL	0.63	1.02	0.098	0.165	8,295
MCGLV8	0.76	1.03	0.104	0.171	8,810
MCG	1.34	1.59	0.186	0.31	17,500

Table 16: LEON3 test result summary

7 LEON3 ASIC Implementation

In this chapter, challenges for LEON3 ASIC implementation are presented and discussed. Predictions and estimates for LEON3 ASIC implementations are also debated. In section 8.2, different ASIC technologies and their values are presented, and in the section thereafter, area and power estimates for possible LEON3 ASIC implementation will be devised. Finally, the probable cost of ASIC LEON3 is discussed.

7.1 ASIC Implementation

The implementation of LEON3 on an ASIC will require some work; However, it has been done before[60, 61, 62, 63], and it should not be an impossible task. That said, new custom technology mapping might need to be written if another ASIC technology such as XFAB is chosen to be synthesized and implemented on. The only current technology mapped for ASICs in the GRLIB is the Synopsys 32/28nm Generic Library for Teaching (SAED32). Some of the standard cell libraries Faraday 180 µm (UMC180) component descriptions can be found in the grlib\lib\tech\umc18 folder. In addition, the design template for SAED32 ASIC implementation can be found in the GRLIB design subfolder.

Gaisler has stated [64] that the bare minimum LEON3 processor in a 130-nm ASIC implementation will amount to 25,000 Gates, which is equal to 3,000 of Xilinx 7-series LUTs [65]. The minimum LEON3 configuration resource utilization in subsection 6.4.1 was 7,000 slices; this is twice the bare minimum amount, so it is possible to approximate that MC, MCGL and MCG configurations will be over 50,000 gates, 70,000 gates and MCG 145,000 gates respectively in a 130-nm ASIC process.

7.2 LEON3 ASIC Implementation Estimates

The estimates of the gate count for the LEON3 configurations were found in the previous section. They are used as an indication for all of the ASIC estimates in the sections below.

7.2.1 Taiwan Semiconductor Manufacturing Company

The Taiwan Semiconductor Manufacturing Company (TSMC) is the world's largest dedicated semiconductor foundry [66]. It provides technology processes for everything from 3 µm to 5 nm. For the use of the LEON3 implementation, 180 nm is considered; however, other processes from the TSMC are also discussed. Only a small amount of data on gate density and effect for the TSMC's technology is available online [67]. This data was used as the basis for power and area estimates in different TSMC processes. The effect was derived from static average leakage and internal dynamic power both per gate and per MHz. The area was calculated with the maximum possible gate density for the different processes, so it should be considered as a best-case scenario. Neither memory nor I/O was considered in either power or area estimates. Unfortunately, no gate density or effect figures for the 180 nm process were found; therefor, an educated guess was made that 109 Kgates/mm^2 is a probable gate density for the 180-nm process. This guess was made based on a thread on Edaboard [68] and is purely a speculative number.

In Figure 35, LEON3 area estimations for all the configurations are displayed in 90-nm, 130-nm, 150-nm and 180-nm processes from the TSMC.

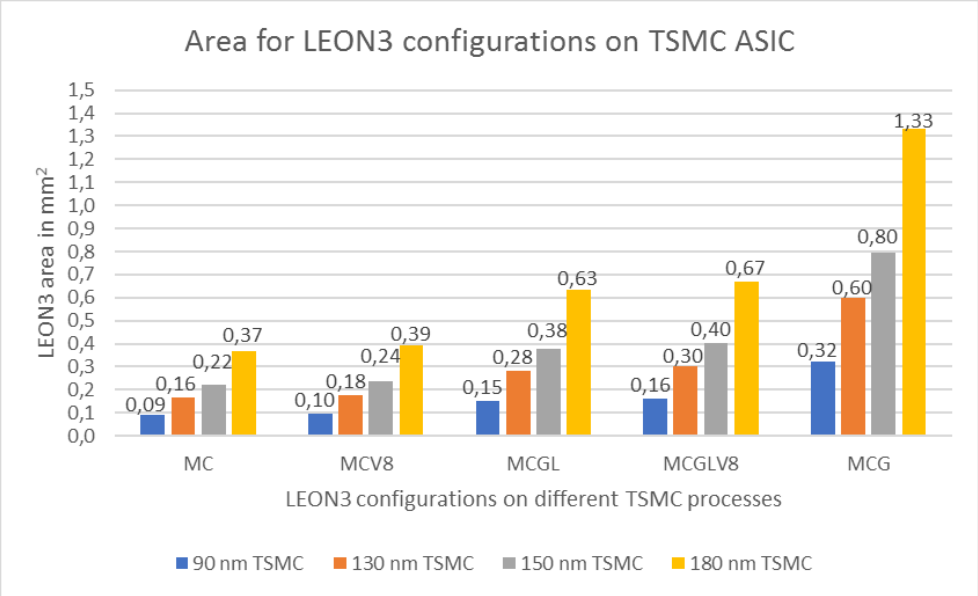


Figure 35: LEON3 configuration area on TSMC ASIC

The Table 17 displays the data used in Figure 36 to present total effect estimates for all of the LEON3 configurations in 90-nm, 130-nm, and 150-nm processes from the TSMC. The 90-nm process had the highest margin in static leakage power and dynamic power consumption and was therefore averaged to 30 nW and 9 nW respectively for the sake of presentation. The same type of averaging was done for dynamic power and static power leakage for the rest of the processes presented in Table 17. The averaging of these values causes an unexpected power effect for the 90-nm process; however, it should be noted that the best-case power consumption would be lower than that of the other three processes. As mentioned previously, the probable effect of the 180-nm process was not found; therefore, only the total effect for the three mentioned processes is illustrated.

Process	Gate Density (Kgates/mm^2)	Leakage (nW)	Internal Power (nW/MHz)
90 nm	448	0.02 - 61.8	1.9 - 16.8
130 nm	243	0.008 - 12.5	3.3 - 8.5
150 nm	182	0.058 - 0.51	6.1 - 9.8
180 nm	109	N/A	N/A

Table 17: Gate density and effect for TSMC technology

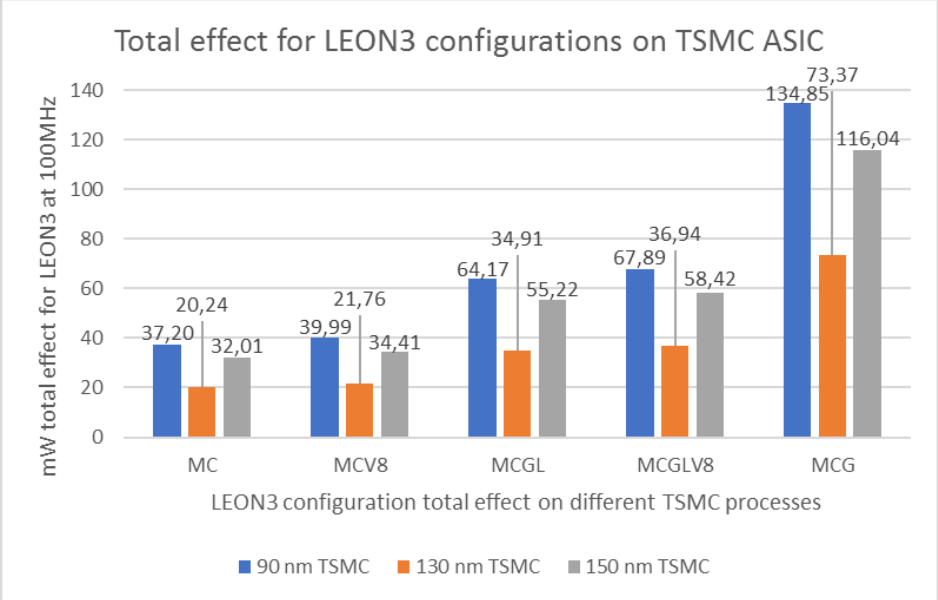


Figure 36: Total Effect for LEON3 configurations on TSMC ASIC

7.2.2 United Microelectronics Corporation

The United Microelectronics Corporation (UMC) [69] is a leading global semiconductor foundry that provides advanced IC production for applications spanning every major sector of the electronics industry. The UMC has a broad technology process, including everything from 450-nm down to 14-nm processes. The gate densities and power effect for the UMC's technology were found online from the 2002 IP Gold catalog [70]. Although the catalog is 17 years old, the gate density and power consumption for processes such as 130 nm and 180 nm should have remained relatively the same; nevertheless, it should be kept in mind that the figures are outdated. Two different Faraday libraries for the 130-nm process were compared for effect and area: The first one was High Speed Low-K and the other one was Low Leakage (FSG). For the area comparison, the Faraday standard 180-nm process library was added.

Error! Reference source not found. displays the area estimates for the LEON3 ASIC implementation in two 130-nm processes and one 180-nm process from the UMC. The UMC High Speed Low-K technology has a lower gate density than the FSG and causes a larger area. In the case of the 4DSpace module application, High Speed Low-K processes will not be necessary; FSG 130-nm and 180-nm processes should thus be considered for the UMC foundry.

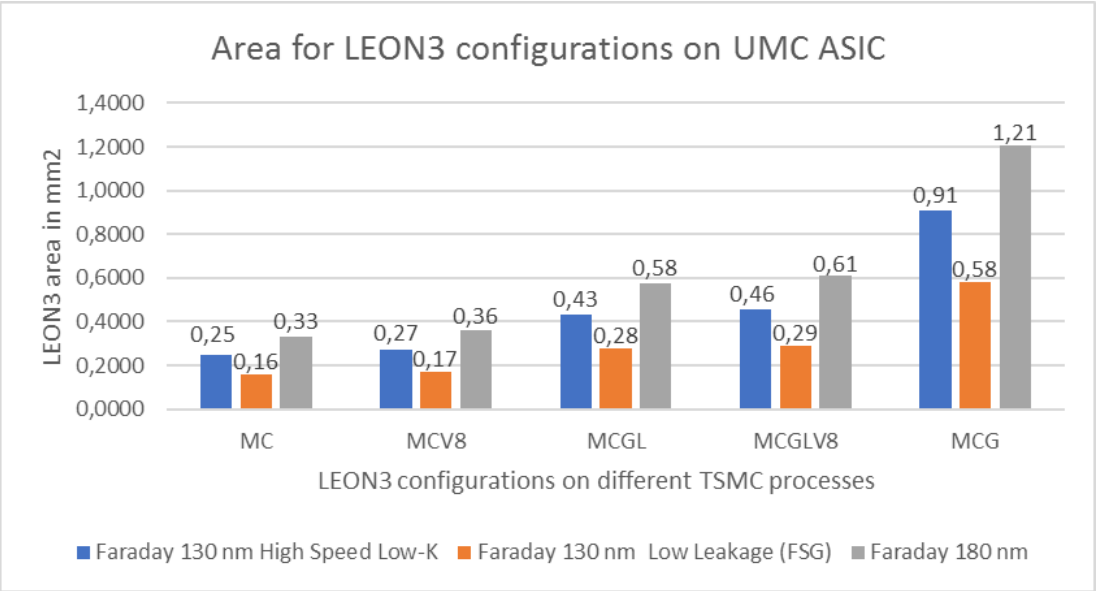


Figure 37: LEON3 configuration area on a UMC ASIC

Figure 38 illustrates the power consumption estimates for the 130-nm UMC processes are shown. The High Speed process has double the effect of the Low-Leakage process, as expected.

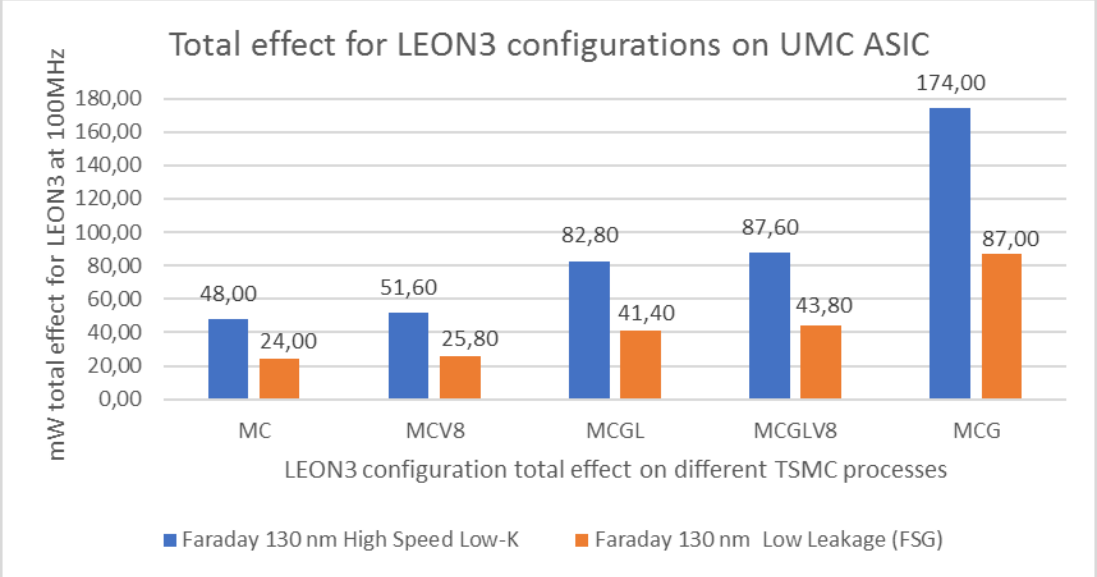


Figure 38: Total Effect for LEON3 configurations on UMC ASIC

7.2.3 **X-FAB**

The X-FAB [71] Silicon Foundries comprise a group of semiconductor foundries that specialize in the fabrication of analog and mixed-signal integrated circuits for fabless semiconductor companies. They provide CMOS processes from 180 nm up to 1 µm. The Nanoelectronics Group at IFI has used the X-FAB's foundries in recent years for multiple projects and is well-accustomed to the technology and processes. The most-used technologies are the 180-nm process (XH018) [72] with 125 Kgates/mm^2 and the 180-nm process (XC018) [73] with 115 kGates/mm^2. A technology called DARE180X [74] also exists; it is based on XH018 for space applications with 59 kGates/mm^2. DARE180X could be an option for LEON3 ASIC implementation for long duration space missions, where a space-hardened design is desired. There are, unfortunately, no power effect numbers found online, so it is left to speculation. Figure 39 illustrates area estimates for LEON3 configuration on XFAB ASIC.

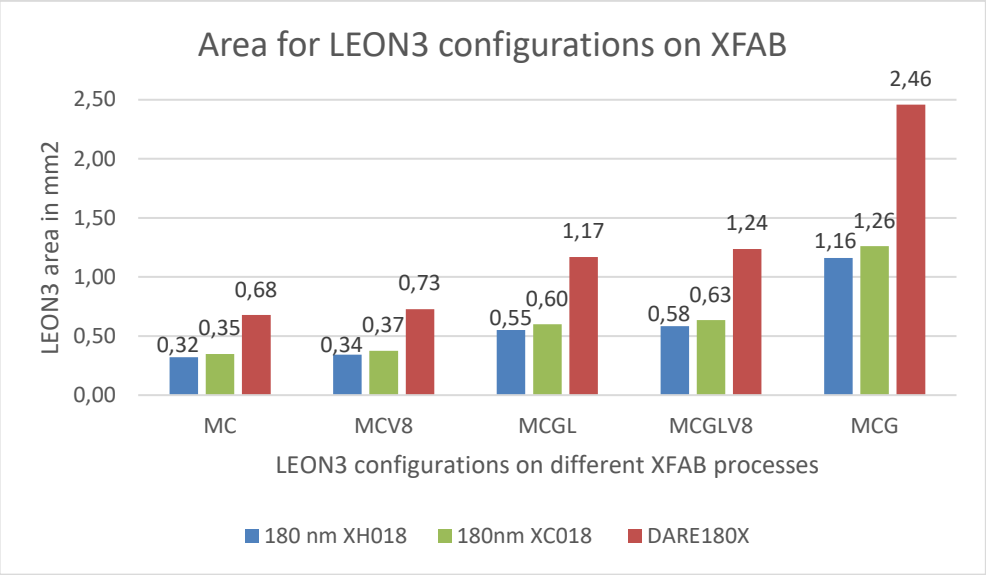


Figure 39:LEON3 configuration area on a XFAB ASIC

7.2.4 Price Estimates for LEON3 ASIC Implementation

The Nanoelectronics Group at IFI uses services that EUROPRACTICE provides. EUROPRACTICE offers affordable access to Multi Project Wafer (MPW) ASIC solutions and is a consortium of five renowned European research organizations that support academic institutions and medium-sized companies with IC prototyping services and system integration solutions [75].

The price estimates are based on area estimates from previous sections and technology processes available through EUROPRACTICE. These technology processes are mainly 180nm from the UMC and the TSMC, 130 nm from the UMC, and 180 nm from X-FAB. The prices [76] are based on block size designs, since even the largest LEON3 ASIC implementation found in the previous section is less than half of the minimum block size. All the prices are displayed for one block design for each technology process.

In Figure 40, the prices are displayed for the different technology processes offered by EUROPRACTICE. For the 180-nm process, prices vary by approximately 1,000 euros between the TSMC, UMC, and X-FAB technologies. The most expensive technology process is the 130-nm UMC, at 4,460 euros.

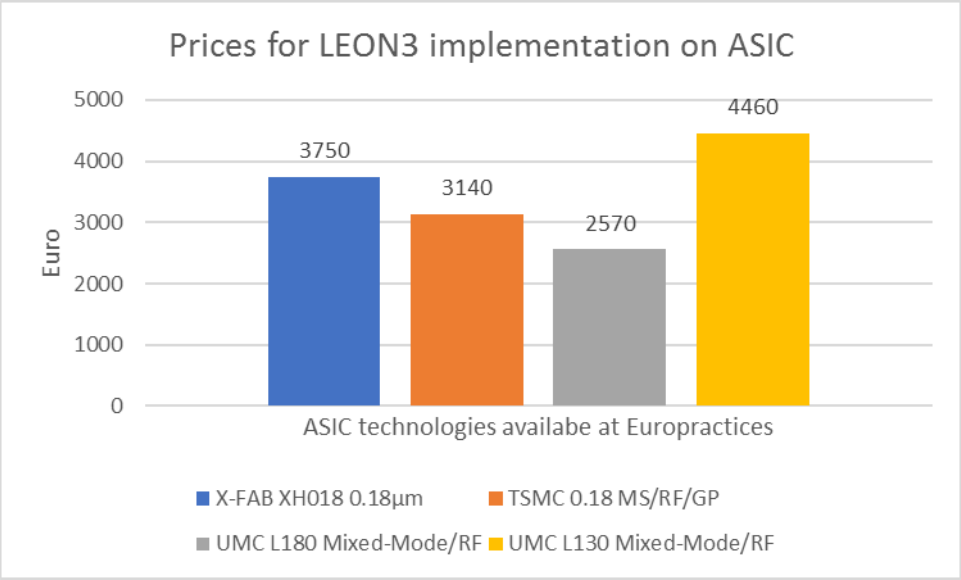
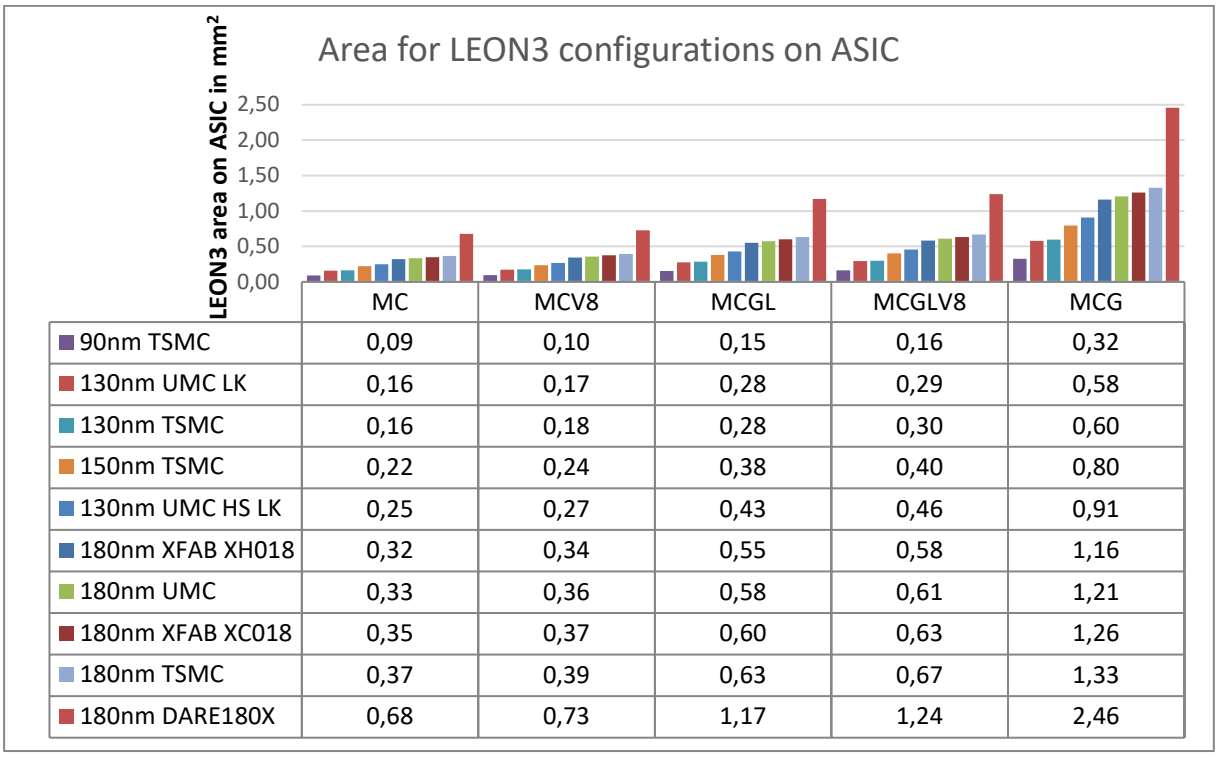


Figure 40: Prices for LEON3 implementation on different technology processes

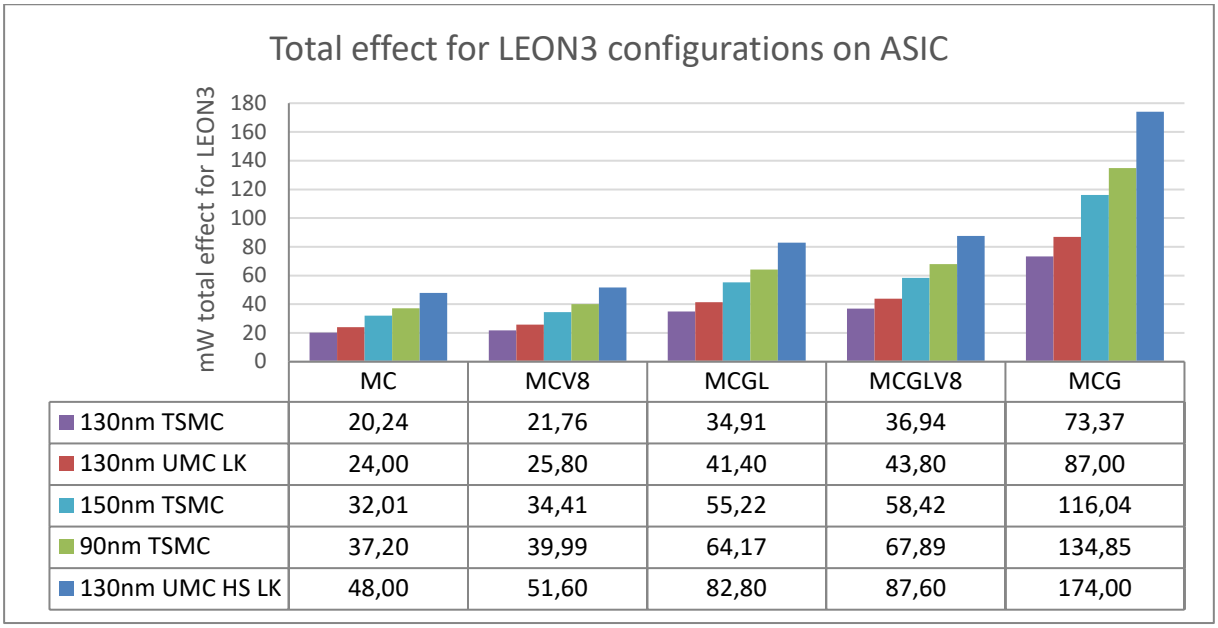
7.2.5 **Summary**

In this chapter rough, estimates were made regarding the area, power consumption, and prices of LEON3 ASIC implementations. The summary of these estimates are presented in Figure 41 and Figure 42. These numbers should only be used as a probable approximation estimate. The price estimates can considered to be more solid figures, because even if a LEON3 ASIC implementation consumes twice the number of gates, it will still be well within the block design size set by EUROPRACTICE. The power estimates conflict somewhat with predictions the made in subsection 6.5.6; however, this is to be expected, since only rough predictions were made.



	MC	MCV8	MCGL	MCGLV8	MCG
90nm TSMC	0,09	0,10	0,15	0,16	0,32
130nm UMC LK	0,16	0,17	0,28	0,29	0,58
130nm TSMC	0,16	0,18	0,28	0,30	0,60
150nm TSMC	0,22	0,24	0,38	0,40	0,80
130nm UMC HS LK	0,25	0,27	0,43	0,46	0,91
180nm XFAB XH018	0,32	0,34	0,55	0,58	1,16
180nm UMC	0,33	0,36	0,58	0,61	1,21
180nm XFAB XC018	0,35	0,37	0,60	0,63	1,26
180nm TSMC	0,37	0,39	0,63	0,67	1,33
180nm DARE180X	0,68	0,73	1,17	1,24	2,46

Figure 41: Summary of area for LEON3 configurations on ASICs



	MC	MCV8	MCGL	MCGLV8	MCG
130nm TSMC	20,24	21,76	34,91	36,94	73,37
130nm UMC LK	24,00	25,80	41,40	43,80	87,00
150nm TSMC	32,01	34,41	55,22	58,42	116,04
90nm TSMC	37,20	39,99	64,17	67,89	134,85
130nm UMC HS LK	48,00	51,60	82,80	87,60	174,00

Figure 42: Summary of total effect of LEON3 configurations on ASICs at 100 MHz

8 Conclusion

In this thesis softcore processors were presented with their advantages, disadvantages, strengths, and weaknesses. Several softcore processors were explored, both from the open-source and the commercial sectors. After reasonable consideration, LEON3 was chosen as the preferred softcore processor. Specifications, development, and testing tools were described in detail. The process of LEON3 configuration and implementation was presented. A relevant user case test was carried out to determine whether LEON3 has the capacity to match a hardcore processor, and the result reveal that it did. Resource utilization and power consumption were presented for all of the LEON3 configurations. Finally, estimates of power, area, and price for LEON3 ASIC implementations were made.

While LEON3 is an ideal softcore processor, it has its' limits in integer processing. A few other softcore processors exist that are more capable of integer calculation; however, they are all commercial ones. Considering the possibility of science code modifications to perform integer calculations and MCV8 LEON3 configuration, LEON3 should be more than capable of performing scientific calculations even with integers. This statement is also true for our use case test for the m-NLP scientific routine.

The MCGLV8 configuration has the best performance and area tradeoffs for the current m-NLP science application. This configuration was determined to be the best option for the science routine as it is; however, the MCGL is a close contender in terms of performance, and it should be investigated.

The overall simplicity, customization, development, debugging, and testing possibilities of LEON3 were strongly confirmed during the testing phase. As a result the choice of LEON3 as the main softcore processor candidate seems to be validated through chapters 5 and 6 of this thesis. During the latter chapter, LEON3 resource utilization coincided with numbers from other research mentioned in previous chapters on such utilization. Furthermore, the power consumption of LEON3 seems to be in the ballpark of numbers found in other research. The LEON3 ASIC implementation estimates demonstrate that LEON3 would be a prime contender for future processor use in the Nanoelectronics Group.

8.1 Future Work

The future testing of LEON3 for m-NLP application should include optimization of the science code and integration of interfaces required for connection with the 4D module. Testing LEON3 with the said interface can be done physically on an FPGA board, or it can be simulated with software. The integration of these interfaces should not require extensive work, as LEON3 AMBA and APB IP blocks are well documented with examples and explanations.

For the current state of the scientific code, where it utilizes floating-point numbers for plasma parameter calculations, MCGLV8 is the best-fitted LEON3 configuration. However, the GRFPU-Lite is an expensive unit in terms of both power and resources, and options for a co-processor for integer calculation, especially for the square root operation, should be investigated. A small co-processor dedicated to more optimized integer calculations should be smaller in size and consume less power than the GRFPU-Lite.

For future testing of LEON3 on an FPGA, an option for a newer FPGA board should be investigated. The author of this thesis suggests a Xilinx Kintex-7 FPGA KC705 [77] board. The templates for this board are newer than that of the ZedBoard and uses a newer Vivado version. Support for KC705 is also more updated and better documented than that of the ZedBoard.

Bibliography

[1] UiO. The ICI series of sounding rockets 2011 [Available from: https://www.mn.uio.no/fysikk/english/research/projects/ici/index.html] Access Date: 18.07.2019.

[2] Bekkeng TA. Prototype development of a multi-needle Langmuir probe system. Oslo: T.A. Bekkeng; 2009. https://www.duo.uio.no/handle/10852/11230.

[3] Fixed-Point vs. Floating-Point Digital Signal Processing | Education | Analog Devices 2019 [Available from: https://www.analog.com/en/education/education-library/articles/fixed-point-vs-floating-point-dsp.html] Access Date: 17.07.2019.

[4] Maxfield C. The Design Warrior's Guide to FPGAs: Devices, Tools and Flows: Elsevier Science; 2004.

[5] Salcic Z, editor Prototyping embedded dsp systems - from specification to implementation. 2004 12th European Signal Processing Conference; 2004 6-10 Sept. 2004.

[6] Group AC. The future of Digital Signal Processing is here now 2016 [Available from: http://www.andraka.com/dsp.php] Access Date: 17.07.2019.

[7] Flynn MJ, Luk W. Computer System Design: System-on-Chip: Wiley; 2011.

[8] Anemaet P, Anemaet PAM, vanAs T. Microprocessor Soft-Cores : An Evaluation of Design Methods and Concepts on FPGAs. part of the Computer Architecture (Special Topics) course ET4078, Department of Computer Engineering. 2003;2.

[9] Balagurusamy E. Fundamental of computers. New Delhi: Tata McGraw Hill; 2009.

[10] Patterson D. Reduced Instruction Set Computers Then and Now. Computer. 2017;50(12):10-2.

[11] Zlatanov N. ARM Architecture and RISC Applications 2016 [Available from: https://www.researchgate.net/publication/295921119_ARM_Architecture_and_RISC_Applications] Access Date: 18.07.2019.

[12] MIPS Architectures 2019 [Available from: https://www.mips.com/products/architectures/] Access Date: 17.07.2019.

[13] SPARC International, Inc 2019 [Available from: https://sparc.org/] Access Date: 17.07.2019.

[14] Nios® II Processors for FPGAs - Intel® FPGA 2019 [Available from: https://www.intel.com/content/www/us/en/products/programmable/processor/nios-ii.html] Access Date: 17.07.2019.

[15] Intel. Nios II Processor Reference Guide 2019 [Available from: https://www.intel.com/content/www/us/en/programmable/documentation/iga1420498949526.html] Access Date: 17.07.2019.

[16] Xilinx. MicroBlaze Soft Processor Core 2019 [Available from: https://www.xilinx.com/products/design-tools/microblaze.html#overview] Access Date: 17.07.2019.

[17] Xilinx. MicroBlaze Processor Reference Guide 2019 [Available from: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug984-vivado-microblaze-ref.pdf] Access Date: 17.07.2019.

[18] Mips. M-Class M51xx Core Family 2019 [Available from: https://www.mips.com/products/warrior/m-class-m51xx-core-family/] Access Date: 17.07.2019.

[19] OpenCores 2019 [Available from: https://opencores.org/] Access Date: 07.17.2019.

[20] Community O. OpenRISC 2019 [Available from: https://openrisc.io/] Access Date: 17.07.2019.

[21] OpenCore. OpenRISC 1200 IP Core Specification 2012 [Available from: https://github.com/openrisc/or1200/blob/master/doc/openrisc1200_spec.pdf] Access Date: 17.07.2019.

[22] Semiconductor L. LatticeMico32 Open, Free 32-Bit Soft Processor 2019 [Available from: http://www.latticesemi.com/en/Products/DesignSoftwareAndIP/IntellectualProperty/IPCore/IPCores02/LatticeMico32.aspx] Access Date: 17.07.2019.

[23] Gaisler C. LEON3 Processor 2019 [Available from: https://www.gaisler.com/index.php/products/processors/leon3] Access Date: 2019.17.07.

[24] Esa. LEON: THE CHIP THAT EUROPE BUILT 2013 [Available from: http://www.esa.int/Our_Activities/Space_Engineering_Technology/LEON_the_space_chip_that_Europe_built] Access Date: 17.07.2019.

[25] Walter M. Evaluation of soft-core processors on a Xilinx Virtex-5 field programmable gate array. SANDIA REPORT 2011;SAND2011-2733.

[26] Longbottom R. Classic Computer Benchmarks 2017 [Available from: https://www.researchgate.net/publication/318987839_Classic_Computer_Benchmarks] Access Date: 17.07.2019.

[27] Gardiner MR. An Evaluation of Soft Processors as a Reliable Computing Platform. All Thesis and Dissertations 2015(5509).

[28] Xilinx. Digilent XUPV5 Board 2019 [Available from: https://www.xilinx.com/support/university/boards-portfolio/xup-boards/DigilentXUPV5Board.html] Access Date: 17.07.2019.

[29] Xilinx. Space-grade Virtex-5QV FPGA 2019 [Available from: https://www.xilinx.com/products/silicon-devices/fpga/virtex-5qv.html] Access Date: 17.07.2019.2019.

[30] EEMBC. CoreMark an EEMBC Benchmark 2019 [Available from: https://www.eembc.org/coremark/] Access Date: 17.07.2019.

[31] umich. MiBench 2019 [Available from: http://vhosts.eecs.umich.edu/mibench/] Access Date: 17.07.2019.

[32] Damman C, Edison G, Guet F, Noulard E, Santinelli L, Hugues J, editors. Architectural performance analysis of FPGA synthesized LEON processors. 2016 International Symposium on Rapid System Prototyping (RSP); 2016 6-7 Oct. 2016.

[33] Nabil L, Jaafar B, Ben Saoud S. Embedded microprocessor performance evaluation case study of the LEON3 processor. Journal of Engineering Science and Technology. 2012;7:574-88.

[34] Li X, Hou L, Geng S, Wang J, Zhang H, editors. The FPGA Prototyping Implementation of LEON3 SoC. 2012 International Conference on Industrial Control and Electronics Engineering; 2012 23-25 Aug. 2012.

[35] Najam Z, Dar MN, Qadri MY, Najam S, Ahmed J, editors. Architectural Enhancement of LEON3 Processor for Real Time and Feedback Applications. 2016 International Conference on Frontiers of Information Technology (FIT); 2016 19-21 Dec. 2016.

[36] Kchaou A, Youssef WEH, Tourki R, editors. Performance evaluation of multicore LEON3 processor. 2015 World Symposium on Computer Networks and Information Security (WSCNIS); 2015 19-21 Sept. 2015.

[37] Groups Y. LEON_SPARC 2019 [Available from: https://groups.yahoo.com/neo/groups/LEON_SPARC/info?] Access Date: 17.07.2019.

[38] Keller AM, Wirthlin MJ. Benefits of Complementary SEU Mitigation for the LEON3 Soft Processor on SRAM-Based FPGAs. IEEE Transactions on Nuclear Science. 2017;64(1):519-28.

[39] Wirthlin MJ, Keller AM, McCloskey C, Ridd P, Lee D, Draper J. SEU Mitigation and Validation of the LEON3 Soft Processor Using Triple Modular Redundancy for Space Processing. FPGA '16 Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2016:205-14.

[40] Silva A, Sánchez Prieto S. LEON3 ViP: A Virtual Platform with Fault Injection Capabilities. Proceedings - 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, DSD 2010. 2010:813-6.

[41] COBHAM. GRLIB IP Library User's Manual 2018 [Available from: https://www.gaisler.com/products/grlib/grlib.pdf] Access Date: 18.07.2019.

[42] COBHAM. GRLIB IP Core User's Manual 2018 [Available from: https://www.gaisler.com/products/grlib/grip.pdf] Access Date: 18.07.2019.

[43] COBHAM. GRMON3 User's Manual 2019 [Available from: https://www.gaisler.com/doc/grmon-eval/grmon3.pdf] Access Date: 18.07.2019.

[44] International S. The SPARC Architecture Manual Version 8 1992 [Available from: https://www.gaisler.com/doc/sparcv8.pdf] Access Date: 18.07.2019.

[45] UNIX. SYSTEM V APPLICATION BINARY INTERFACE SPARC Processor Supplement Third Edition 1994 [Available from: https://www.gaisler.com/doc/sparc-abi.pdf] Access Date: 18.07.2019.

[46] COBHAM. BCC User's Manual 2019 [Available from: https://www.gaisler.com/doc/bcc2.pdf] Access Date: 18.07.2019.

[47] COBHAM. TSIM2 Simulator User's Manual 2019 [Available from: https://www.gaisler.com/doc/tsim-2.0.63.pdf] Access Date: 18.07.2019.

[48] COBHAM. Configuration and Development Guide 2018 [Available from: https://www.gaisler.com/products/grlib/guide.pdf] Access Date: 18.07.2019.

[49] Cobham. GRLIB IP Library 2019 [Available from: https://www.gaisler.com/index.php/products/ipcores/soclibrary] Access Date: 18.07.2019.

[50] Cobham. Download Leon/GRLIB 2019 [Available from: https://www.gaisler.com/index.php/downloads/leongrlib] Access Date: 18.07.2019.

[51] Digilent. ZedBoard Zynq-7000 ARM/FPGA SoC Development Board 2019 [Available from: https://store.digilentinc.com/zedboard-zynq-7000-arm-fpga-soc-development-board/] Access Date: 18.07.2019.

[52] Xilinx. Download Archive 2019 [Available from: https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html] Access Date: 18.07.2019.

[53] Cygwin. Cygwin: in Git; 2019 [Available from: https://www.cygwin.com/] Access Date: 18.07.2019.

[54] Arm. Cortex-R4 – Arm Developer: ARM Developer; 2019 [Available from: https://developer.arm.com/ip-products/processors/cortex-r/cortex-r4] Access Date: 18.07.2019.

[55] Kosaka EN. Development and characterization of the data flow for a spacecraft payload instrument: University of Oslo; 2018. https://www.duo.uio.no/handle/10852/66080.

[56] SatTrek. Coefficient of Determination: Definition 2019 [Available from: https://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination] Access Date: 21.07.2019.

[57] Weisstein EW. Least Squares Fitting: MathWorld--A Wolfram Web Resource; 2019 [Available from: http://mathworld.wolfram.com/LeastSquaresFitting.html] Access Date: 21.07.2019.

[58] Hussain W, Nurmi J, Isoaho J, Garzia F. Computing platforms for software-defined radio: Springer; 2017.

[59] Kuon I, Rose J. Measuring the Gap Between FPGAs and ASICs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2007;26(2):203-15.

[60] Glocker E, Qingqing C, Zaidi AM, Schlichtmann U, Schmitt-Landsiedel D, editors. Emulation of an ASIC power and temperature monitor system for FPGA prototyping. 2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC); 2015 29 June-1 July 2015.

[61] Wong T. LEON3 System-on-Chip Port for BEE2 and ASIC Implementation. Technical Report. 2015.

[62] Staunton D. Successful Use of an Open Source Processor in a Commercial ASIC 2005 [Available from: https://www.design-reuse.com/articles/12145/successful-use-of-an-open-source-processor-in-a-commercial-asic.html] Access Date: 18.07.2019.

[63] Cellier ANM. An Embedded Data Acquisition System with Leon Processor and Nand Flash for Wireless Sensor Networks. Turin, Italia: University of Turin; 2018.

[64] Habinc S, Gaisler G. The Gaisler Research Roadmap for FPGA IP-Cores 2007 [Available from: https://nepp.nasa.gov/mafa/talks/MAFA07_13_Habinc.pdf] Access Date: 18.07.2019.

[65] COBHAM. GRLIB AREA 2018 [Available from: https://www.gaisler.com/products/grlib/grlib_area.xls] Access Date: 18.07.2019.

[66] Tsmc. About TSMC 2019 [Available from: https://www.tsmc.com/english/aboutTSMC/index.htm] Access Date: 18.07.2019.

[67] Cadence. tsmc standard cell libraries: @studylib.net; 2003 [Available from: https://studylib.net/doc/18351185/tsmc-standard-cell-libraries] Access Date: 18.07.2019.

[68] Edaboard. Die Size 2019 [Available from: https://www.edaboard.com/showthread.php?54353-Help-me-to-estimate-die-size-for-chip-in-0-13-process] Access Date: 19.07.2019.

[69] Umc. UMC Overview 2019 [Available from: http://www.umc.com/English/about/index.asp] Access Date: 18.07.2019.

[70] UMC. Gold IP 2002 Catalog 2002 [Available from: http://www.umc.com/english/pdf/ip-08012-2002.pdf] Access Date: 18.07.2018.

[71] X-FAB. Analog/Mixed-Signal Semiconductor Foundry: Privacy Statement; 2019 [Available from: https://www.xfab.com/home/] Access Date: 18.07.2019.

[72] X-FAB. 0.18 Micron Modular Analog Mixed HV Technology 2019 [Available from: https://www.xfab.com/fileadmin/X-FAB/Download_Center/Technology/Datasheet/XH018_Datasheet.pdf] Access Date: 18.07.2019.

[73] X-FAB. 0.18 Micron Modular RF enabled CMOS Technology 2017 [Available from: https://www.xfab.com/fileadmin/X-FAB/Download_Center/Technology/Datasheet/XC018_Datasheet.pdf] Access Date: 18.07.2019.

[74] Imec. DARE180X 2019 [Available from: https://dare.imec-int.com/en/technologies/dare180x] Access Date: 18.07.2019.

[75] EUROPRACTICE. EUROPRACTICE IC SERVICE 2019 [Available from: http://europractice-ic.com/] Access Date: 18.07.2019.

[76] EUROPRACTICE. 2019 mini@sic Europractice MPW runs Schedule and Prices 2019 [Available from: http://europractice-ic.com/wp-content/uploads/2019/06/190603_MPW2019-miniasic-v5.0.pdf] Access Date: 18.07.2019.

[77] Xilinx. Xilinx Kintex-7 FPGA KC705 Evaluation Kit 2019 [Available from: https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html#hardware] Access Date: 23.07.2019.