Design of Hardware Accelerators with Configurable Pipeline

by

Gurveer Kaur

B.Tech, Lovely Institute of Technology, 2011

Project Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Engineering

in the
School of Engineering Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY

Summer 2016

Approval

Name:	**Gurveer Kaur**
Degree:	**Master of Engineering**
Title:	***Design of Hardware Accelerators with Configurable Pipeline***
Examining Committee:	**Chair:** Dr. Ivan V. Bajić Associate Professor
	Dr. Fabio Campi Senior Supervisor Lecturer
	Dr. Craig Scratchley Supervisor Senior Lecturer
Date Defended/Approved:	August 22, 2016

ABSTRACT

In today’s world, people are widely using technology to make their lives more comfortable and better. The development of semiconductors technology is making Integrated Circuits(IC) smaller and smaller in size, thus allowing IC designer to include more and more functionalities in their products. This development of technology has allowed a large diffusion of semiconductor devices in all aspects of human life, leading to the concept of “embedded” computation, described as the practice of including the small processor devices in all spaces of our world, from our houses, to our cars, to even “wearable electronics” that we carry around as we move.

In particular, floating point computation (FP) is a feature of computers that, at the price of significant additional hardware complexity and sometimes at the price of result accuracy, provides a much larger range of usable numbers, thus significantly enhancing the flexibility and usability of our computation. The additional hardware complexity imposed by FP units imposed a relevant price in Silicon Area (making the IC more expensive) and especially in terms of power consumption. In turn, energy consumption is a very severe issue in semiconductor technologies: first, it causes unreliability of the IC technology. Secondly, IC energy consumption leads to greenhouse gas emission: IT-related consumption is well above 20% of GHG highly in developed countries. Finally, many IC systems are battery operated and high consumption may jeopardize the system usability and/or user experience.

One very significant category of embedded processors is that of embedded sensors. Embedded sensors produce relevant quantities of raw data that needs to be adequately classified in order to provide significant information, and Machine Learning is often applied as a strategy for sensor data classification: most machine learning strategies requires floating point computation as a mean to handle the complex dynamics of the data to be classified

This MENG project aims at exploring design strategies for low-power FP computation. In the following, we will introduce the design of a hardware FPU unit whose sub-blocks can be programmed to change dynamically the computational speed with the change in the voltage. This enables the FPU to adapt their consumption to the requirement of the environment, offering high performance (and high consumption) whenever needed by the environment, but adapting to low power, low speed mode whenever intensive processing is not necessary.

ACKNOWLEDGMENT

I would like to express my gratitude towards my supervisor, Fabio Campi for the valuable guidance and motivation for this project. I am grateful to him for sharing his opinions and guiding me on various aspects of this project.

Finally, I would like to thank my parents for not just supporting me, but also for their guiding me throughout the duration of this project.

TABLE OF CONTENTS

Approval....II
Abstract....III
Acknowledgement....IV
Table of Contents....V
List of Tables VI
List of Figures.... VII
List of Acronyms ….VIII

Chapter 1. 1
1.1 Introduction ….1
1.2 IoT protocol Stack and Architecture….2
1.3 Power Consumption in IoT Devices….3
1.4 Low Power Design Opportunities for IoT processors….5
1.5 GALS Design style advantages of IoT processors….7

Chapter 2. Context of the Project….9
2.1 The Crescent Beach Floating Point processor….9
2.2 Register locking for GALS Design….10

Chapter 3. Description of the Work Performed….12
3.1 Target Technologies ….12
3.2 Configurable Pipeline Strategy ….13
3.3 Design and Verification of Hardware Accelerator….15
3.4 Physical Implementation of hardware Accelerator….19

Chapter 4. Power Measurements ….24
4.1 Power Measurements on Altera Cyclone IV FPGA ….25
4.2 Power measurements on VLSICMOS028 FDSOI….26

Chapter 5. Conclusions ….29

Chapter 6. References ….30

List of Tables

Table 1. Adder Simulation Results for a set of exemplar inputs ....………………15

Table 2. Multiplier Simulation Results for a set of exemplar inputs ………………………………17

Table 3: Square Root Simulation Results for a set of exemplar inputs …………………………...18

Table 4: Divider Simulation Results for a set of exemplar inputs …………………………………..19

Table 5: Synthesis performance of the FP Multiplier on Altera Cyclone FPGA technology …….20

Table 6: Synthesis performance of the FP Adder on Altera Cyclone FPGA Technology………..20

Table 7: Evaluation of the ideal area/speed tradeoff in the multiplier design …………………….21

Table 8: Synthesis performance of the FP Multiplier in cmos28 FDSOI Technology. Working conditions are slow, 0.85V, 125C ………………………………………………………………….22

Table 9: Synthesis performance of the FP Adder in cmos28FDSOI Technology. Working conditions are slow, 0.85V, 125C…………………………………………………………………..22

Table 10: Power Consumption of the floating-point multiplier accelerator on Altera Cyclone FPGA computing the algorithm described above. Working conditions are typical, 1V, 25C…………..25

Table 11: Power Consumption of the FP adder accelerator on Altera Cyclone FPGA computing the algorithm described above. Working conditions are typical, 1V, 25C……………………….25

List of Figures

Figure 1: A representation of the IoT protocol stack ……………………………………………….2

Figure 2: Representation of an IoT processor System-on-chip, composed by the microprocessor, the bus/memory architecture, external connectivity and embedded or tightly coupled Sense/Actuation logic ………………………………………………………………………………3

Figure 3: Power Supply waveforms in a GALS design. Red waveforms are currents due to a synchronous clock; Blue waveforms are cur-rents due to fully asynchronous clocks…………...8

Figure 4: Schematic representation of the open-source "Crescent Beach" architecture based on the RISC-V instruction set ………………………………………………………………………..10

Figure 5: Schematic description of the SOI technology process …………………………………..13

Figure 6: Description of the Bypass channel added during this MENG work to all pipeline FF in the FP Adder/Subtractor and the FP multiplier unit to enable power savings by means of a "Configurable Pipeline" ……………………………………………………………………………14

Figure 7: Adder simulation results in the case bypass='1' …………………………………………16

Figure 8: Adder simulation results in the case bypass='0'…………………………………………16

Figure 9: Multiplier simulation results in the case bypass='0' ………………………………………17

Figure 10: Multiplier simulation results in the case bypass='1'………………………………………17

Figure 11: Square root simulation results ……………………………………………………………18

Figure 12: Divider Simulation results …………………………………………………………………19

Figure 13: Area, Leakage and Dynamic power vs Timing period in ns for the FP multiplier……21

Figure 14: Power consumption related to the FP Multiplier accelerator unit as a function of the working frequency ……………………………………………………………………………………26

LIST OF ACRONYMS

IoT	Internet of Things
FP	Floating Point
VLSI	Very Large Scale Integration
IC	Integrated Circuits
CMOS	Complementary metal oxide semiconductor
SOI	Silicon On Insulator
SoC	System on Chip
LDA	Linear Discriminant Analysis
FPGA	Field Programmable gate array
DVFS	Dynamic Voltage Frequency Scale
FF	Flip Flops
SRAM	Static random-access memory
RTL	Register Transfer Level
HDL	High Descriptive Language
FPU	Floating Point Unit

1. INTRODUCTION

Intel developed the first microprocessor IC in 1970. From 1980 onwards, the microprocessor entered our houses in the form of phones, computers and then tablets. Historically, microprocessor have been designed to perform at the highest possible speed, while energy consumption was not considered a significant design constraint until the early 2000s, with the diffusion of embedded computing and the commercial issues of battery-operated devices. Since then, due to the diffusion of this energy-sensitive markets such as cell phones and tablets, microprocessor designers have been introducing low-power design strategies such as clock and power gating, near threshold computing, and low power circuits. The main feature of the aforementioned strategies is to involve a change in the way the circuit are designed. The designer needs to choose upfront, depending on the target application field, whether to implement a HIGH-SPEED or a LOW-POWER design.

In particular, to meet very aggressive implementation constraints, microprocessors have been steadily evolving from the late 1990s into complex Systems-on-Chip where a single IC integrates all components of a computer system including memory, peripherals, and bus architectures. In this context, a very common strategy to increase the computational capabilities of a microprocessor while mitigating the related power consumption is "Hardware/Software co-design" (6), intended as the practice of mapping a part of the computation normally deployed in software by a microprocessor into a specialized Hardware unit that is embedded in the microprocessor's own chip. Such computation units are usually defined "Hardware Accelerators". Classic example of kernels that are often mapped on hardware accelerators are graphic engines for cell phones and Floating Point Units in embedded microprocessors.

A second important change of perspective, is taking place in the last few years, when the processor market shifting focus towards the so called Internet of Things (IoT) (1)(2)(3). The Internet of things requires the distribution of sensor devices in almost any space related to our lives. Sensors monitor our environment, and generate very large quantities of data. While the ultimate goal of such data collection is to process information in a centralized server on the so-called "Cloud", the amount of data to be processed is so large that a first step of processing must necessarily take place in the sensor itself in some form of embedded processor. In the following of this report, we will define an IoT processor as a small Integrated Circuit (IC) containing a sensing/actuating logic and an embedded microprocessor core. Depending on the conditions in the environment where the sensor is located in the physical system it works with, it may need to work very fast for a short critical transitory, while and otherwise it would need to work as a low power circuit. For example, a capacitive proximity sensor that is mounted on robots to detect the presence of humans and avoid unwanted robot-human accidental collision such as the one described in (4) may spend long times processing data at low speed when no object is approaching the sensor. When the sensor detects an approaching object, the computation speed must be augmented dramatically while the processor tries to classify whether the object at risk of collision is a human body or a different category of objects that do not require the same safety precautions. Similarly, a security camera for an industrial site such as the one described in (5) may spend long times processing data at low speed when no moving object is detected in the camera's field. When the camera detects an approaching body, the computation speed must be augmented dramatically while the processor tries to classify whether the object at risk of collision is a human body or a car (possibly a thief) or a different category of objects (such as a cat or a leaf) that do not require the same level of alarm.

The focus of this projects is to design hardware accelerators for microprocessor units that are small enough to fit into a sensing/actuator logic, capable to work at low power low speed for most of the time, but yet capable to deliver enough computational power in case of peaks where the sensing activity and related data classification needs fast and accurate data acquisition.

1.2 IoT Protocol Stack and Architecture of IoT Processors

Following the definition introduced above, IoT processors are small integrated circuits massively distributed across our living space. They occupy the lowest hierarchical level in the "Internet-of-things (IoT)" communication chain and, because of this: we will address them in this document as "IoT LEAVES". As described in (1), (2), (3) the IoT is an embedded protocol stack comprising software processing, hardware sensors and network connectivity. The IoT protocol can be described as a 3-level hierarchy (Figure 1): the above comprise the lowest level described the IoT leaves. The second level is the Gateway node used to connect, merge and route the flow of information coming from the leaves. The third level of hierarchy is represented by the cloud data center.

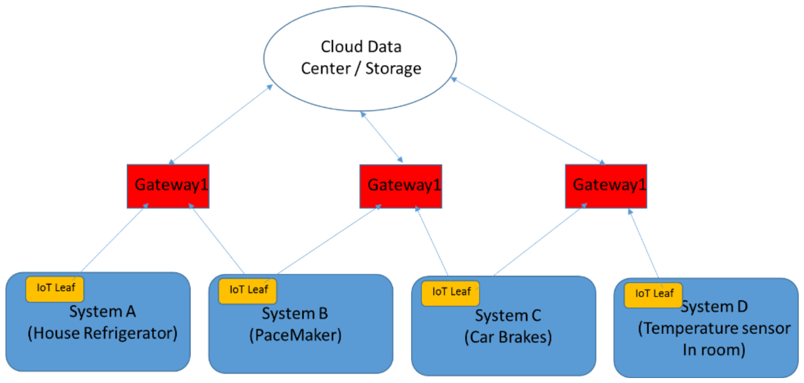


Figure 1: A representation of the IoT protocol stack

This work specifically targets the lowest level in the IoT hierarchy. In particular, the design of ***IoT processors***: *microprocessor devices embedded in the same IC with the IoT sensor, or anyway tightly coupled to it*. The features that are desirable in IoT processors are briefly described in the following:

1. IoT processors extract information from a physical system, classify that and route it to the cloud. In some cases, they can also actuate commands from the cloud on to the physical system. They are physically connected to the other leaves and to the HUB nodes in the network. Depending upon the application, the connectivity can be wired or wireless. The IoT processors need to be small devices so that they can be easily embedded in the physical system without affecting it.

2. If there is no power supply available in the system, for example, sensors in human body or public space, they must be designed in such a way that they consume little energy or are even able to scavenge it from the physical system.
3. IoT processors need to provide long-term reliability, since the physical system may be located in accessible locations where it is difficult and costly to reach and fix the IoT leaf.
4. The basic functionality of the IoT processors is to sense, gather, and store and process sensor information. In order to minimize data bandwidth to the cloud, IoT processors need to perform locally a first processing stage. Such processing may include classification. Pattern detection and encryption of collected data

Figure 2 shows a representation of the IoT Processor system-on-chip architecture outlined above. This project focuses on the uP core that is at the heart of the SoC. The following section will focus on the features of such uP core, and provide guidelines for its design, that will be the subject of the following chapters of this report.

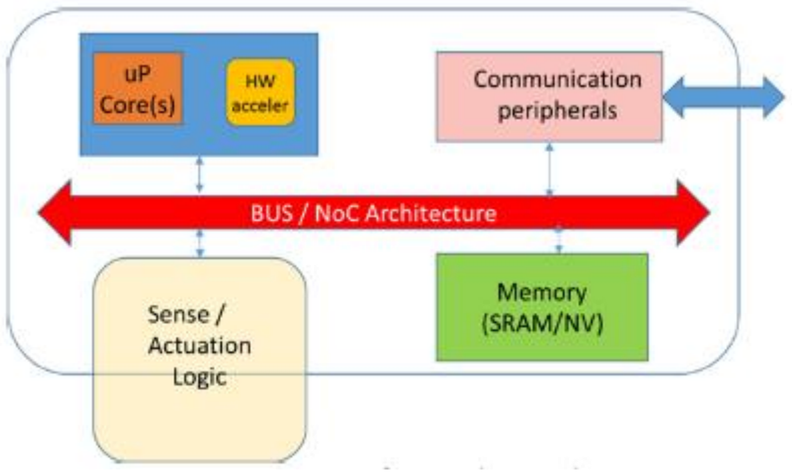


Figure 2 A Representation of an IoT processor System-on-chip, composed by the microprocessor, the bus/memory architecture, external connectivity and embedded or tightly coupled Sense/Actuation logic

1.3 Power Consumption in IoT Devices

Power consumption is an aspect of microprocessor design that historically had not been considered the foremost priority: in the design of traditional desktop or laptop, microprocessors from the 1970s to the 1990s computational performance and clock speed have always considered more important than consumption. Only later, with the advent of hand/held or portable electronic devices and embedded microprocessors, power consumption suddenly became a fundamental constraints, and low power design started attracting relevant interest both in the scientific community and in the industry. Moderate power consumption became a fundamental marketing vehicle to assert the success of a processor design. The large diffusion of the ARM processor architecture is a clear example of this trend.

Later still, due to advances in latest technology node development, power consumption has become, other than a marketing issue, a very critical concern for IC reliability. In CMOS-VLSI technology nodes with channel length <65 nm, the most of the causes of IC malfunctioning are related to the power consumption/energy dissipation in the chip area (7).

The IoT leave concept introduced in the previous pages represent a very specific case of embedded system. IoT leaves are invariably located in the human living space, and must be small and portable. Often, they may also be located in some inaccessible position, with no direct access to the power plug.

Referring to the points introduced in the previous sections:

1. IoT leaves must sustain severe peak computation requirements that require a computationally strong architecture: IoT leaves may need to be reprogrammed to support different sensing patterns or different classification algorithms. Such upgrades may be required after the embedded sensor has been located in the environment, so that it cannot be easily reached and substituted. For this reason, any ASIC hardware acceleration cannot be too specific and not flexible enough.
2. On the other hand, area, energy and design cost constraints pose many limitations on what designers can do to get better/high performance. In principle, in a processor architecture, performance can be obtained architecturally, i.e., Superscalar or multicore features to use heavy pipelining, at circuit/layout level(use of high-speed CMOS circuits) with the use of a more aggressive technology (i.e. FinFET or SOI technologies, deep sub-micron channel lengths), or using higher reference voltages (i.e. supply and substrate voltage scaling)

All methodologies introduced above impact severely the power consumption of the device. In budgeting the IoT leaves, the power consumption can be a severe issue. The effects of power consumption can be three-fold:

1. Localized- Since the size of the IoT leave chips should be small, power consumption may impose an undesired complexity to the chip. In many IC designs, especially in the case of small circuits such as IoT devices, the number of IO pads rather than the size of internal circuits define the floorplan area. High power consumption would require a higher number of pads to ensure an appropriate current feed. In turn, more pads will require more pins in the package and will increase the overall system complexity, making it costlier and less practical.

2. Globalized- The energy consumption related to Information technology (IT) in developed nations is steadily increasing across the years, increasing and is becoming a major cause of greenhouse gasses emissions. In 2013, the energy consumed by ICT was already more than 10% of the total consumption in many developed countries, and is on track to increase to 20% (8). Studies predict that by 2030, if current trends continue, electricity consumption caused by Internet-related devices will increase up to 30X, and energy prices will grow substantially (2). We are indeed approaching a paradox, where the IoT will be essential to enable smart energy utilization to minimize Green House Gas (GHG)

emissions (2),(3), but in turn will be in itself cause for severe increase in energy consumption and consequent GHS emissions. IoT leaves are numerically largely dominant with respect to other ICs that contribute to the IoT infrastructure. Hence, a careful attention to low-power design is essential. On the other hand, this cannot be obtained compromising performance, due to Real-Time constraints.

3. Reliability of Integrated Circuits- Reliability is defined as the capability of Integrated Circuits to work correctly for the time-span it is required to, and in the working conditions where it is expected to operate. Reliability is a major issue. It affects the design and manufacturing costs. Low reliability decreases the profit margins, and affects the trust of customers on the specific product, and towards the use of IoT services in general. The issues that can affect CMOS reliability can be categorized into spatial and temporal unreliability effects (7).
   - 3.1 Spatial Unreliability Effects- These effects are mainly dopant fluctuations, edge roughness and gradient effects, which are visible after the production. They affect yields by altering the geometry and structure of the circuit.
   - 3.2 Temporal Unreliability Effects- These effects can be seen when there is a change in the operating conditions: supply voltage, temperature, and computation workload. These are further classified into aging effects and transient effects:
     - 3.2.1 Aging Effects- Examples of the aging effects include Negative Bias Temperature instability, Hot-Carrier injection, Electro-Migration
     - 3.2.2 Transient Effects- Examples of transient effects include Static and Dynamic Voltage drops, Simultaneous Switching nodes, Thermal Runaway and temperature related timing degradations.

Spatial effects are visible immediately after production, so that they can be detected during post-fabrication tests: of course, they influence design costs, as they force to discard a portion of the manufactured silicon. However, once sorted, remaining ICs will be fully functional. Temporal effects are much more critical in terms of IC design for IoT because they are unpredictable, undetectable malfunctioning that can appear (or not) at any moment in time during the device lifetime, leading to transient malfunctioning or permanent ruptures. Such effects are difficult to fix, as they would require unplanned physical access to the device. In fact, *the majority of temporal effects are due power dissipation effects in the IC and time varying peaks in the current distribution of across the IC power grid*. The best way to ensure high reliability, especially against temporal effects, is to design in order to minimize power dissipation in general, and transient current peaks in particular.

1.4 Low Power Design Opportunities for IoT Processors

There are many ways to reduce the power consumption of a microprocessor circuits. Low power circuits and/or micro-architecture design are very popular and well established in the state of the art. Unfortunately, these approaches are not applicable to the IoT leaves, as they require modification of libraries and design flows, which in turn can increase the manufacturing cost.

Most importantly, low power circuits and micro-architectures achieve substantial power mitigation at the expense of lower peak performances. This is completely unacceptable in the IoT context where the sensing system, mostly working at very low rates, must be able to sustain at times sudden substantial performance peaks.

Other well-known practices for low power design are power gating and clock gating, and dynamic voltage and frequency scaling.

1.4.1 Power Gating

Power gating can be described as the practice to turn off the power supply voltage of a given digital block when the digital block is unused for a long time. The main drawback of this approach is that turning on the digital block from the off state requires very strong current transistors to charge the capacitance of the Vdd supply network. The On-chip current gradients are dangerous and to avoid the current gradients, specific current limiting regulators can be embedded at the expense of increasing design cost. Power gating can be applied in IoT processors, especially in the case where a specific accelerator in the processor architecture is not used for a long time. An example of such case could be the Square Root of the Divider accelerator in a floating-point processor as described in chapter 2 of this document. As square root and divider may not be used for long times in a given computation, they could be power gated to the advantage of the overall power consumption. Although this technique can be useful in the context of this work, it is well established in literature and will not be the focus of this activity

1.4.2 Clock Gating

Clock gating can be described as the practice to turn off the clock of a digital block when the digital block is not used. As opposed to power gating, clock gating is much easier to apply as it possible to activate or de-activate it in a single cycle. In addition, it requires minimal hardware overhead for its application.
Clock gating is fully embedded in modern HDL-based design flows and as such can be added on top of other strategies. In chapter 3, clock gating will be utilized to minimize the power consumption of HW accelerators. The main drawback in the application of clock gating is that its contribution, although useful, is typically moderate and does not offer substantial advantages.

1.4.3 Dynamic Voltage/Frequency Scaling (DVFS)

DVFS can be described as the practice of dynamically adapt the clock and voltage supply of a digital circuit to the computational load that the circuit is requiring, increasing the clock and voltage supply when the circuit needs to operate at high frequency and decreasing the same when a low rate of computation is sufficient to meet the real time constraints. In brief, in DVFS, the clock rate and consequently the power dissipation is dynamically driven by the real time constraints imposed on the circuit by the external world. This will help in saving power, as every part of circuit will always deliver the exact amount necessary to meet timing constraints. This strategy may be applied to the full circuit, or with a finer granularity providing independent clocking and voltage supply to different portions of the circuit. This is very appealing in the design of a microprocessor that features embedded hardware accelerators, such as the floating-point processor that will be described in Chapter 2 of this document. Every accelerator could be dynamically tuned to a high speed, high power state during computational peaks requiring that particular accelerator, or to a low speed, low power state when the accelerator is used only sporadically.

In this second case, opportunities for power saving are significantly higher, but they come at the price of additional design cost:

1. Architecture Design- Explicit synchronization circuitry is required in order to support the crossing of different clock domain boundaries.
2. Physical Design- During the floorplan phase, separate power supply grids need to the defined. In order to ensure the smooth electrical transition between different voltage levels, level shifter cells must be inserted in between the domains
3. Timing Characterization- The timing constraints are normally verified in all digital designs worst (setup checks) and best (hold checks) operating conditions. DVFS require the digital design to be operative in a significant range of Vdd conditions. Timing closure then must be repeated for each condition. Most importantly, timing libraries will have to be characterized in all desired voltage operating points.

Regardless the costs outlined above, DVFS is a very appealing technique for power reduction in IoT processors. Although DFVS is less effective, in terms of sheer power reduction, as compared to low power circuits or microarchitectures, the most considerable advantage of DVFS is that, thanks to its dynamic nature, it enables power reduction WITHOUT IMPACTING (or as little as possible) PEAK PERFORMANCE. This is visible in a real case, for example, in Figure 14, where the behavior of the same hardware unit with or without application of DVFS is shown. As it is evident from the figure, although the power consumption at different frequencies is different (and lower for the DVFS case), the maximum frequencies achieved by the two solutions is the same.

This feature is particularly suited to the operating environment of IoT processors that, as described in the previous pages, are intrinsically subject to intrinsically variable real time constraints.

1.5 GALS design style and advantages for IoT processors

GALS (Globally Asynchronous Locally Synchronous) is a design methodology largely applied in today's Systems-on-Chip designs. The term GALS refers to the practice of partitioning a digital design in different portions, each of whom runs with an independent asynchronous clocking scheme.
The extensive use of GALS is the natural enabling factor for the deployment of a fine-grained DVFS as described in the previous subsection. The advantages of this design option in the context of IoT processors are outlined in the following:

1. Since clock edges are not "centralized" across the IC, but different regions of the design have independent asynchronous clocks, the current peaks induced by the clock edge have a smoother behavior and they compensate with each other. A smoother current ensures a much higher level of reliability of the IC. Figure
2. Every region of the IC can run at the clock frequency imposed by the Real Time Constraints. This in turn avoids the need to over-design components that, due to their nature, may run at low speed but in a single-clock design need to be sped-up to match the frequency of other components. An example of such situation is the peripheral bus of

a high-performance DSP: while the DSP core needs to run at several hundred MHz to ensure high-end computation, peripherals such as UART or SPI may easily run at frequencies <100MHz. But, if the design has a single clock, they would need to be over constrained to meet chip-level specifications

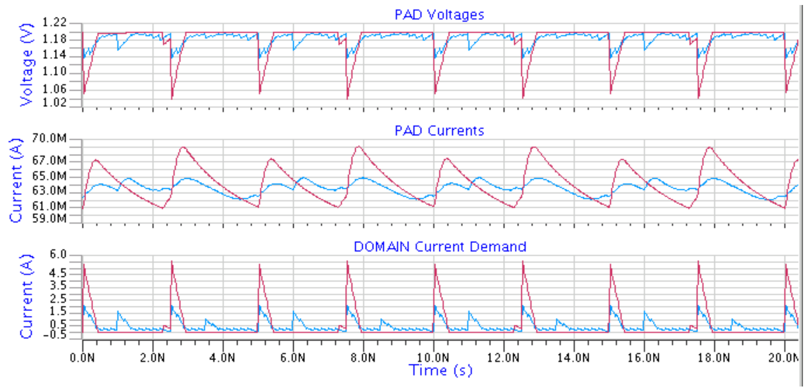


Figure 3: Power Supply waveforms in a GALS design. Red waveforms are currents due to a synchronous clock; Blue waveforms are cur-rents due to fully asynchronous clocks.

The application of the GALS design style and fine-grained DVFS offers the opportunity for significant power reduction. When writing software code for a processor applying GALS, the programmer (or the operating system) in the position to dynamically increase clock frequency and supply voltage of a hardware accelerator or communication peripheral to accommodate sudden demands in computation efficiency, only to tune it down when the transitory has exhausted or is not a priority any more. In processor systems featuring a solid RTOS, power as a design parameter would be simply one of the different factors involved in task scheduling.

In this section, power consumption was identified as a foremost priority in the design of embedded processors for IoT applications, in particular for the lowest level of the IoT protocol stack that is composed by IoT leaves, small systems-on-chip comprising sensing logic and a small-embedded microprocessor for first-order data classification. Power consumption mitigation, though, cannot be obtained at the expense of peak performance: while IoT leaves are expected to function for their largest part of their product lifetime in a quasi-idle state, they must be able to sustain brief but very intense transitory period of high computational demands.

In particular, clock gating and dynamic voltage and frequency scaling appear to be the more appealing strategies to obtain power mitigation without affecting peak performance.
The MENG project described in this document was part of a larger research effort targeted at the design of an open source processor architecture for IoT applications. In particular, this project focused on the application of specific modifications to the processor architecture in order to highlight and augment the benefits offered by the methodologies described above on the processor computation units. Although in this work two well-known floating point accelerators (a 32-bit IEEE single precision floating point adder and a 32-bit IEEE single precision floating-point multiplier) were used as test case, the proposed methodology can be applied to any kind of pipelined hardware accelerators in different design contexts.

In the following, chapter 2 will describe the floating-point processor design that is the context of this MENG project. Chapter 3 will describe the work performed and the proposed modifications to the floating-point accelerators pipeline. Chapter 4 will describe the numerical results derived from the proposed modifications, and Chapter 5 will draw the conclusions.

2. CONTEXT OF THIS MENG PROJECT

The aim of this work is to demonstrate in a “realistic” application case the benefit of applying clock gating and DFVS to a processor design targeted at IoT application, and to propose specific modifications to the processor’s hardware accelerator units in order to emphasize the benefit of such techniques. Choosing a specific microprocessor architecture for embedded systems, this work will evaluate what type of architectural modifications may maximize the energy gain made available by clock gating and DVFS and measure the available energy gain enabled by such modifications given a well-known standardized application commonly used in sensor data classification.

The application that will be used as a benchmark during this work is the Linear Discriminant Analysis (LDA) algorithm. LDA is a Machine Learning Algorithm normally used to organize sensor data in classes based on criteria extracted from a set of pre-classified “training data”. LDA is commonly considered the simplest and computationally lighter machine-learning algorithm. For this, it appears particularly suited to embedded systems and to sensor data classification in particular. Due to the intrinsic large range of the sensed data, Machine Learning algorithms are typically based on floating point numbers. Since we have to deal with sudden computation peaks with stringent real time constraints, we can consider that software emulation of floating point operation to be highly redundant and too power-hungry, so our reference system architecture will be a Processor unit with hardwired Floating Point acceleration.

2.1 The “Crescent Beach” Floating Point processor

As reference processor architecture, we will use the RISC-V instruction set. RISC-V is an open-source processor ISA for embedded systems made available by the RISC-V foundation at University of California, Berkley. RISC-V (pronounced RISC-five) represent a sort of de-facto, free standard for embedded processors, and supports a solid Floating Point extension. The implementation of the RISC-V instruction set that we will be using in this work is an open-source VHDL suite of the RISC-V processor developed as an educational and research tool in Simon Fraser University. The release of the processor including Floating Point support is described in Figure 4 and is code-named “Crescent Beach (CB)”. CB is organized as a standard RISC processor composed of Instruction Decode, Register File, ALU, Multiplier and PC-control logic. On top of this standard configuration, CB features an embedded hardware FP unit organized as a coprocessor, with an independent FP instruction decode, an independent FP Register file, and independent FP acceleration units (but without program counter control that is reserved to the main core). The supported FPU units are adder/subtractor, multiplier, and divider, and square root, integer to float and float to integer conversion

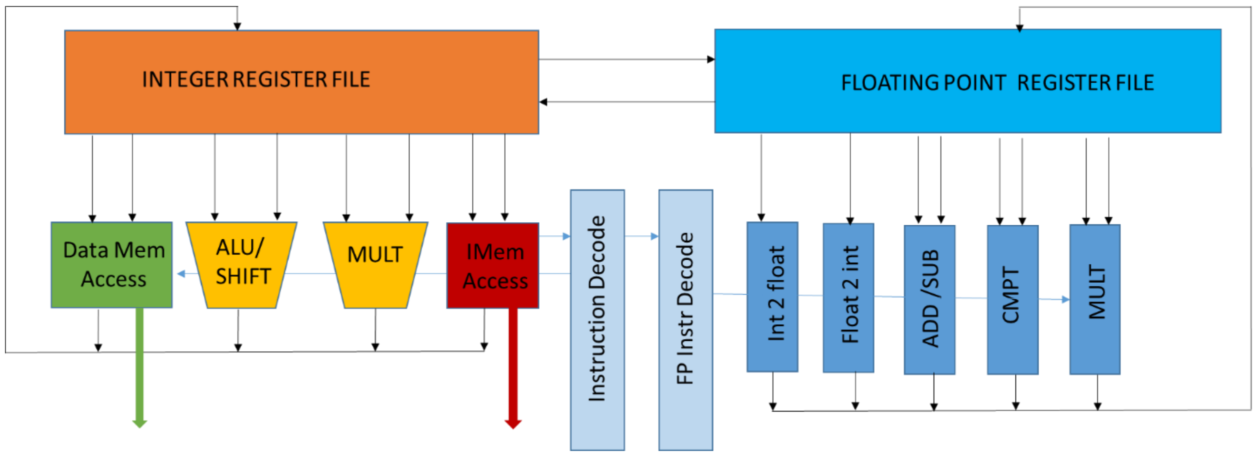


Figure 4: Schematic representation of the open-source "Crescent Beach" architecture based on the RISC-V instruction set

This MENG project has taken place in the context of a larger research effort that includes the full design and implementation of the Floating Point processor, its characterization in different working points to enable DVFS and voltage scaling, and the mapping of the LDA algorithm as a relevant test case. In particular, this work was oriented at the VHDL simulation and testing of the floating-point units, and the estimation of their performance. Another important contribution of this work has been the definition, implementation and measurement of “configurable pipeline”. In order to support high design speeds most of the Floating Point units in the processor feature a pipelined design. In particular, the adder/subtractor implements a 4-stages pipeline, and the Multiplier implements a 3-stages pipeline, the divider supports a 9-stages pipeline and the Square Root unit implements a 12-stages pipeline, while the int-to-float and the float-to-int units complete their computation in one cycle.

2.2 Register locking support for GALS design

In a standard RISC architecture (11), every computational instruction operates on internal registers, reading operands from the internal register file and writing back their results on the same register file. All instructions complete their execution in the same amount of pipeline cycles. In order to accommodate the presence of hardware accelerators with heterogeneous latency, the “Crescent Beach”(9)(10) processor architecture has been modified with respect to a standard RISC architecture, while respecting the RISC-V instruction set. A “*Register Locking*” strategy has been applied: every time an instruction starts computation on any unit, the destination register of such instruction is “locked” on a specific table. If (and only if) one of the following instructions needs to use the same register as source operand, the processor is stalled until the hardware unit that locked the register has completed its computation. Upon completion of the computation, the destination register is written with the operation result and the lock is removed. *This hardware mechanism ensures that the regular flow of the program is maintained regardless the latency of every unit.* For example, if the square root unit is activated by the assembly instruction “SQRT R1, R2”, no following assembly instruction can use the register R1 (the destination of the SQRT instruction), until the hardware SQRT unit has completed its computation. Given the numbers introduced above, this would happen 12 cycles after the locking

The relevant advantage of this solution in the context of the design of IoT processors is that the latency of the hardware accelerators embedded in the processor architecture DOES NOT NEED TO BE SET AT DESIGN TIME, as the locking ensures that no instruction can be started before all necessary operands are available. This enables a widespread application of the GALS concept described in the previous chapter: the speed (and Voltage) of each hardware accelerator can be increased or decreased at computation time depending on how often the specific accelerator is currently being used. Thanks to the register locking mechanism, the regular flow of the program will never be compromised. If, for example, the MUL operator has been set in very low power mode, so that its latency is actually 6 cycles instead of three, the register locking will ensure that its result is never read before being available. The worst thing that can happen is that the following instructions are stalled waiting for the MUL to complete, thus delaying the completion of the code running on the processor. If the MUL is part of a computational peak that must be rapidly resolved due to real time constraints, the voltage of the MUL unit can be increased so that it only takes two cycles to complete, instead of the nominal 3, and the following instruction will be unlocked earlier leading to faster computation.

3. Description of the work performed in the MENG project

The Crescent Beach processor was designed in Simon Fraser University as RTL VHDL code, and is distributed as an Open-Source hardware IP. Its architecture design was a prerequisite, rather than the focus of this specific MENG project. This MENG project focused on three specific aspects of the overall processor realization:

1) Verification of the Floating Point Accelerators, that were imported as IPs from a different IP project (12), and implementation on FPGA and CMOS-VLSI technology
2) Characterization of the benefits made available by clock gating and DFVS in the context of the Crescent Beach architecture, in particular for the implementation of the LDA algorithm
3) Modification of two of the Floating point accelerators in order to implement a "Configurable pipeline" that could emphasize the power mitigation benefits of clock gating and DVFS described above

3.1 Target Technologies

Since the floating-point processor that is the target of this MENG work is an open source computation suite written in VHDL, the two classic design environments for VHDL design were chosen as target technologies: FPGA technology and CMOS-VLSI technology.

Of course FPGAs do not allow (or only vey marginally allow) to reap the benefits introduced by the DFVS techniques introduced in the previous section. Voltage scaling on FPGAs is still at a very immature stage, and although possible (13), (14), it is not well supported by commercial EDA tools so that it is not within the focus of this MENG work. Similarly, FPGAs do not really meet the prerequisites outlined in the previous sections for IoT processors. Nevertheless, FPGAs represent an essential prototyping step for any design effort targeted at VLSI technology, including the design of IoT processors where this MENG work can be categorized: so it is useful to port the proposed processor on FPGA support, in order to enable efficient further work on algorithm implementation. In fact, future MENG theses in the same research context will be targeted at the mapping of algorithms on the FPU processor using FPGAs.

For the FPGA solution that is mostly used as a prototyping and reference environment an Altera board was selected (15). Logic Simulation and synthesis was deployed using the EDA tools distributed by Altera.

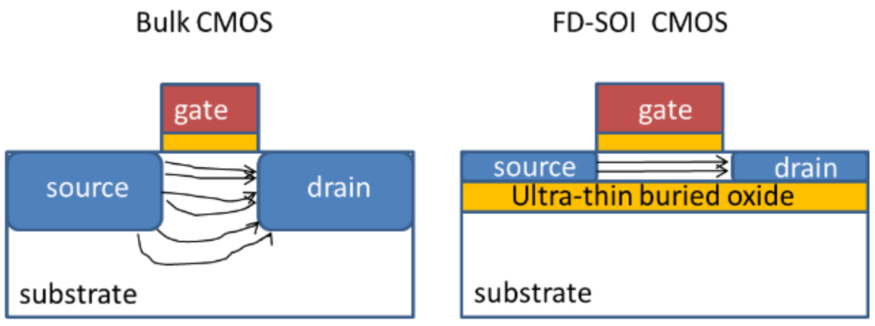


Figure 5: Schematic description of the SOI technology process

For the CMOS-VLSI, the selected technology is the one where today the benefit of DVFS are more evident, that is CMOS028-FDSOI (16). SOI (Silicon-Oxide-Insulator) is a flavor of the standard CMOS process where a thin oxide layer (Usually labeled buried oxide or BOX) is added underneath the conductive channel of the transistors as described in Figure 5. This allows for a much better control of the channel currents from the gate terminal, and limits the leakage component of the CMOS structures. The technology used in this work is a planar SOI CMOS, but the SOI option is often used even in more aggressive finFET technologies.

The result section of this work will include results for both FPGA and CMOS28FDSOI technologies described above.

3.2 The "Configurable Pipeline" strategy

We define "*pipelining*" as the practice to add flip-flops (FF) in a digital design, in order to beak a single critical path in multiple smaller ones. This, at the price of a higher data latency, allows the hardware units to work at a higher clock frequency.

This price is acceptable (and very willingly paid in essentially any digital design) to allow the computational units to increase their working frequency, and consequently the number of data processed per time unit (throughput). As introduced above, this happens at the expense of a longer latency between the start and the end of the computation of a given set of data. The presence of the pipeline in the Floating point accelerators that compose the "Crescent Beach" processor is indispensable for achieving the high work frequency that is required in case of sudden computational peaks. Unfortunately, it involves two unpleasant issues:

1) Unwanted power consumption: pipelining a design involves the addition of register in order to break critical paths thus creating different stages. This creates a relevant power overhead: FF are very power hungry components, and on top of their own specific consumption, they require the transmission of the clock signal. In turn, the clock signal is by definition the signal that creates the larger consumption, having a toggling rate of 2 changes per cycle (200%), as opposed to data signal that have typically toggling rates <10% .
2) Unwanted computation latency: as described above, the FFs implementing the pipeline stages induce a larger latency to the design. Thanks to the register locking mechanism implemented on the processor architecture, this does not cause any problem in the precedence of computed instructions. Hence, this is a price we can happily pay at high

speeds, in order to increase our computation throughput (increasing throughput at the expense of latency is the typical positive effect of pipelining). On the other hand, it may be cause of significant overhead at low speeds.

For the two reasons outlined above, ideally, it would be nice to have the FFs in the design when the units are required to work at high frequencies, and remove them when the unit is required to work at low frequency. This is of course not physically possible if the same circuit is required, such as in the case of IoT devices, to work at low speed when in quasi-idle mode and at high speed during computational peaks.

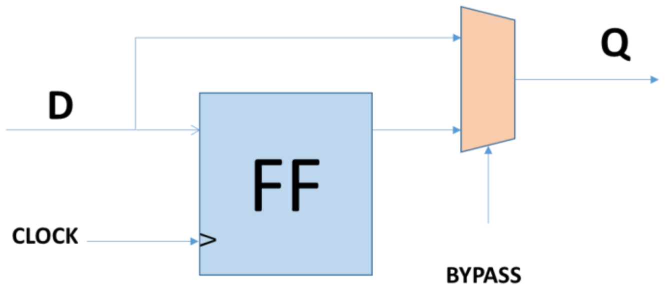


Figure 6: Description of the Bypass channel added during this MENG work to all pipeline FF in the FP Adder/Subtractor and the FP multiplier unit to enable power savings by means of a "Configurable Pipeline"

The FFs are edge triggered and the inputs are enabled when there is a transition in the clock signal. This will help to control the timing in the circuit as the edge of the clock triggers the FFs. The FFs will always be part of the circuit, and will always affect the design area. The next best thing to not having the FFs as part of the design is to avoid the relevant power consumption related to their presence in the circuit. This is possible, only paying a moderate overhead in the circuit design, by adding a BYPASS circuitry to each pipeline FF in the design as described in Figure 6: Activating the bypass signal, the FF is BYPASSED, and the circuit works as if the FF was not part of the design. This significantly increases the critical path of the design, while decreasing the related latency (because removing the pipeline only one cycle is required for the computation). Most importantly, if the FFs are BYPASSED, they do not need to receive the clock signal. As a consequence, the clock transmitted in the function unit when the bypass is active can be switched off (or, more appropriately, gated) leading to a significant power save (again at the expense of the related speed). It should be noted that clock gating normally has a modest impact because only the FF of a block that is NEVER used are gated, while in this context also FFs of units that used in low power mode are gated.

3.3 Design and Verification of Hardware accelerators

The Front End step of a Digital Design involves HDL Coding, Simulation and Synthesis. In this work, HDL codes for all floating-point accelerators were made available in form open source HDL blocks from the work described in (12)1). The focus of the project was to verify the correctness of the operators, modify the code in order to implement the "configurable pipeline" strategy outlined in section 3.2, and evaluate the available mitigation in power consumption.

Mentor ModelSIM was used as HDL simulation tool. Altera Quartus II and Synopsys dc_shell were used for synthesis. As described in the previous sections, the FPU is composed by six main sub-blocks- int2single, single2int, adder, multiplier, divider and square root.

The first step was to perform all the necessary modification to the open source HDL to enable the bypass architecture described in section 3.2. The Bypass signal was added to the main entity of the FPU blocks, and a combinational Mux was added in front of every pipeline register in the design (controlled by the bypass signal) as described in Figure 6. In addition, the clock signal was gated, again depending on the bypass signal, so that no clock would be distributed in the FP components in case bypass was activated.

RTL simulation was performed in three different configurations: (a) before the insertion of the bypass mux, (b) utilizing the pipeline registers through the bypass mux, and (c) bypassing the registers. More precisely, for the FP adder and multiplier, the simulation was performed in all three configurations. On the other hand, since the division and square root operator were not relevant for the LDA algorithm that is the focus of this MENG work, for the divider and square root, simulation was only performed in the case (a).

In the two sets of simulations with and without bypass='1' the simulated blocks yielded identical results, the only difference being in terms of Latency.

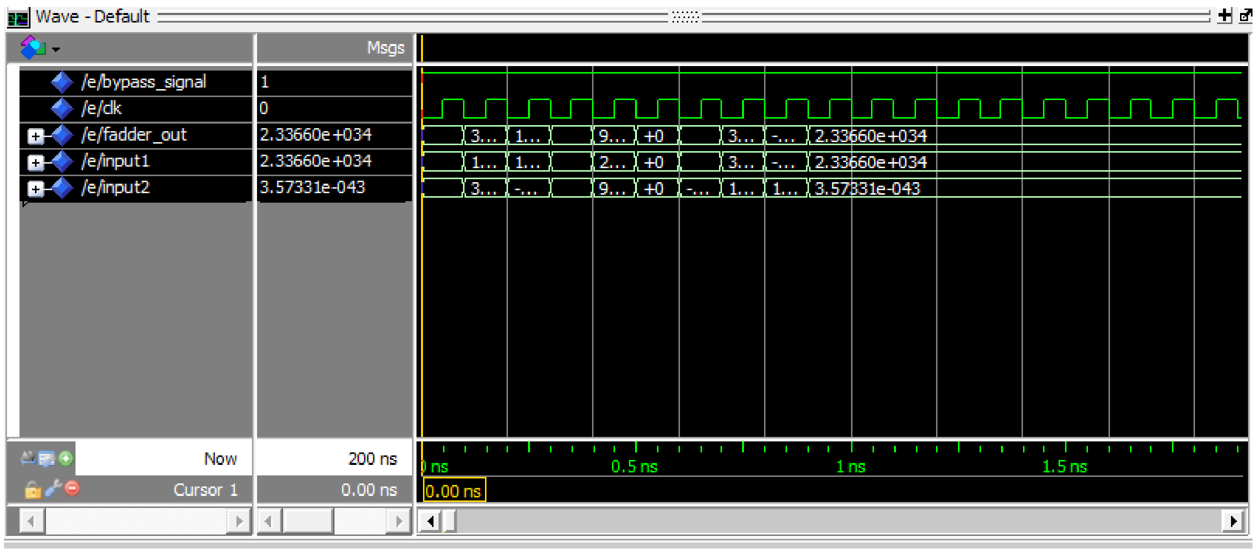
Table 1: Adder Simulation Results for a set of exemplar inputs

Adder Pipelining			
Sr no.	Input1	Input2	Output
1	1.40E-45	1.40E-45	2.80E-45
2	1.40E-44	3.35E-43	3.49E-43
3	1.58E-38	-1.02E-10	1.02E-10
4	2.34E+34	-3.73E+28	2.34E+34
5	2.34E+34	3.57E-43	2.34E+34

Adder Bypass=1			
Sr no.	Input1	Input2	Output
1	1.40E-45	1.40E-45	2.80E-45
2	1.40E-44	3.35E-43	3.49E-43
3	1.58E-38	-1.02E-10	1.02E-10
4	2.34E+34	-3.73E+28	2.34E+34
5	2.34E+34	3.57E-43	2.34E+34

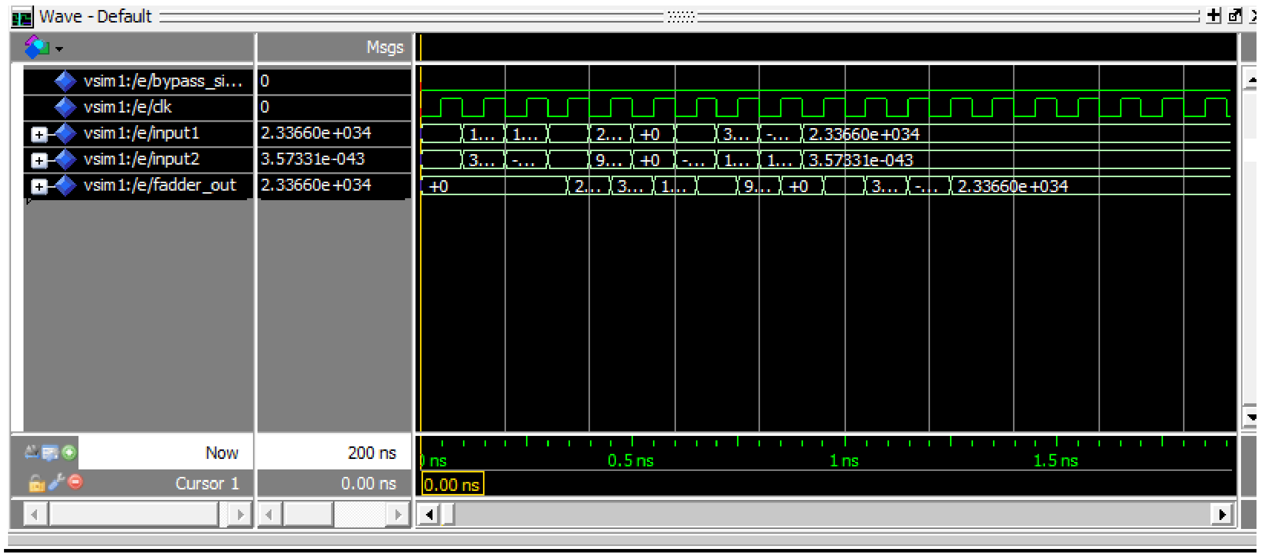
Adder Bypass=0			
Sr no.	Input1	Input2	Output
1	1.40E-45	1.40E-45	2.80E-45
2	1.40E-44	3.35E-43	3.49E-43
3	1.58E-38	-1.02E-10	1.02E-10
4	2.34E+34	-3.73E+28	2.34E+34
5	2.34E+34	3.57E-43	2.34E+34

As it can be observed from Figure 7, and Figure 8 the adder output is independent from the bypass input. However, the latency is different: Latency of the original VHDL code = 3 cycles, Latency for the modified VHDL with Bypass=0 = 3 cycles, Latency of the modified VHDL with bypass=1 = 0 cycles (as the system bypassing the pipeline FFs becomes combinational)



```
/e/bypass_signal   1
/e/clk             0
/e/fadder_out      2.33660e+034
/e/input1          2.33660e+034
/e/input2          3.57331e-043
Now                200 ns
Cursor 1           0.00 ns
```

Figure 7: Adder simulation results in the case bypass='1'



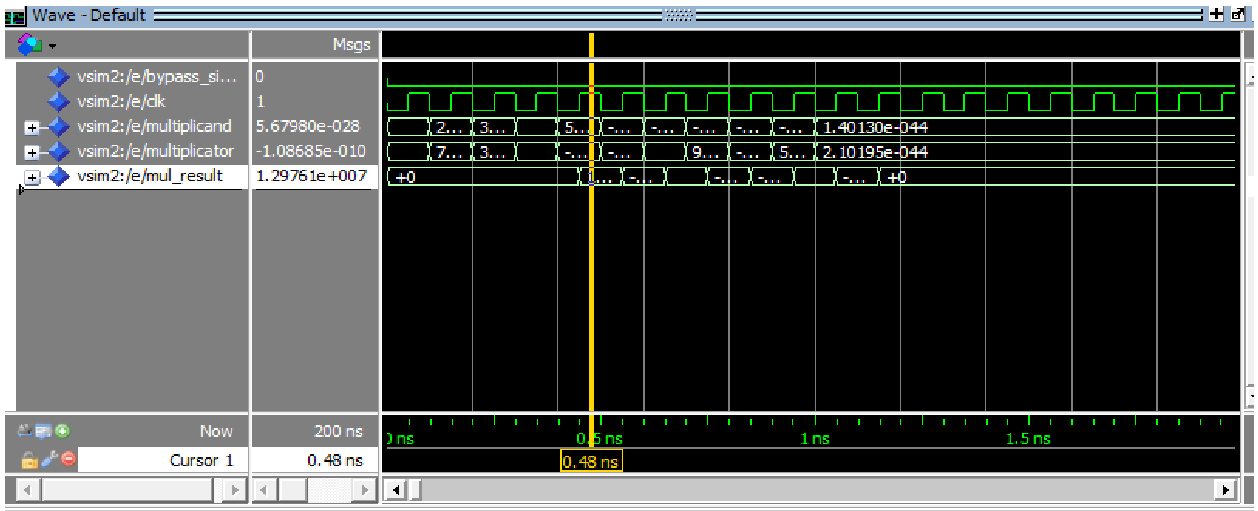
```
vsim1:/e/bypass_si...   0
vsim1:/e/clk            0
vsim1:/e/input1         2.33660e+034
vsim1:/e/input2         3.57331e-043
vsim1:/e/fadder_out     2.33660e+034
Now                     200 ns
Cursor 1                0.00 ns
```

Figure 8: Adder simulation results in the case bypass='0'

As it can be observed from Figure 9 and Figure 10, the multiplier output is independent from the bypass input. However, the latency is different: Latency of the original VHDL code= 4 cycles, Latency for the modified VHDL with Bypass=0 = 4 cycles, Latency of the modified VHDL with bypass=1 = 0 cycles (as the system bypassing the pipeline FFs becomes combinational)

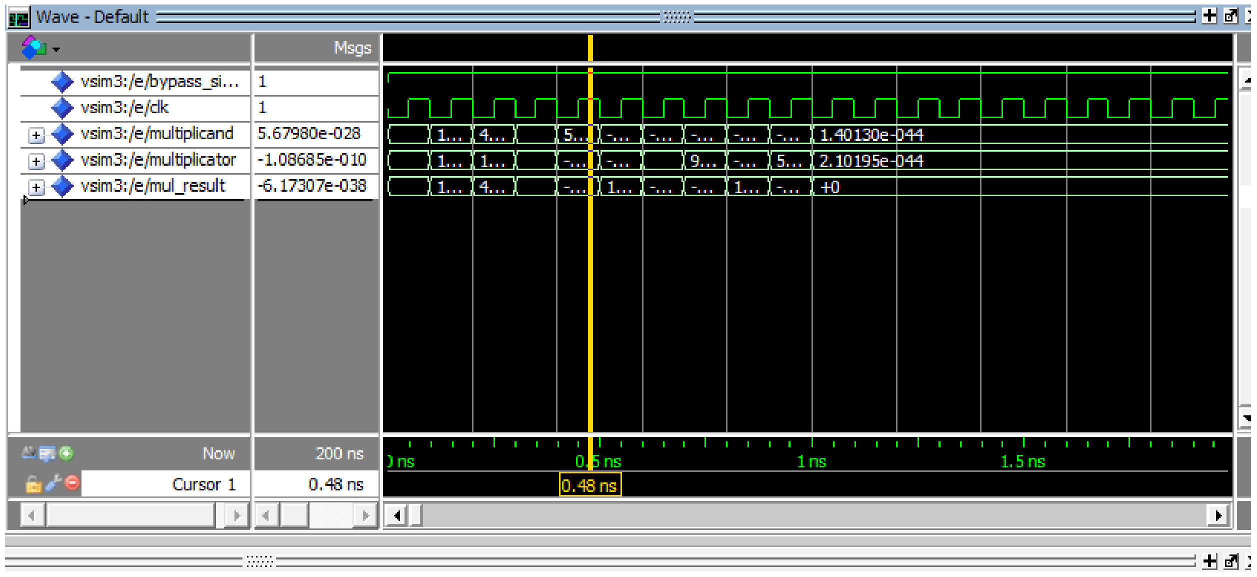
Table 2: Multiplier Simulation Results for a set of exemplar inputs

Multiplier Pipelining				Multiplier Bypass 0				Multiplier Bypass 1			
Sr no.	Multiplicand	Multiplicator	Output	Sr no.	Multiplicand	Multiplicator	Output	Sr no.	Multiplicand	Multiplicator	Output
1	5.55E-28	2.34E+34	1.30E+07	1	5.55E-28	2.34E+34	1.30E+07	1	5.55E-28	2.34E+34	1.30E+07
2	5.68E-28	-1.09E-10	-6.17E-38	2	5.68E-28	-1.09E-10	-6.17E-38	2	5.68E-28	-1.09E-10	-6.17E-38
3	-1.14E-13	-9.90E+27	1.13E+15	3	-1.14E-13	-9.90E+27	1.13E+15	3	-1.14E-13	-9.90E+27	1.13E+15
4	-1.12E-10	2.31E+18	-2.26E+08	4	-1.12E-10	2.31E+18	-2.26E+08	4	-1.12E-10	2.31E+18	-2.26E+08
5	-3.56E+35	9.31E-39	-3.75E-03	5	-3.56E+35	9.31E-39	-3.75E-03	5	-3.56E+35	9.31E-39	-3.75E-03
6	-1.14E-13	-9.90E+27	1.13E+15	6	-1.14E-13	-9.90E+27	1.13E+15	6	-1.14E-13	-9.90E+27	1.13E+15
7	-2.31E+18	5.59E-28	-1.29E-09	7	-2.31E+18	5.59E-28	-1.29E-09	7	-2.31E+18	5.59E-28	-1.29E-09



	Msgs
vsim2:/e/bypass_si...	0
vsim2:/e/clk	1
vsim2:/e/multiplicand	5.67980e-028
vsim2:/e/multiplicator	-1.08685e-010
vsim2:/e/mul_result	1.29761e+007
Now	200 ns
Cursor 1	0.48 ns

Figure 9: Multiplier simulation results in the case bypass='0'



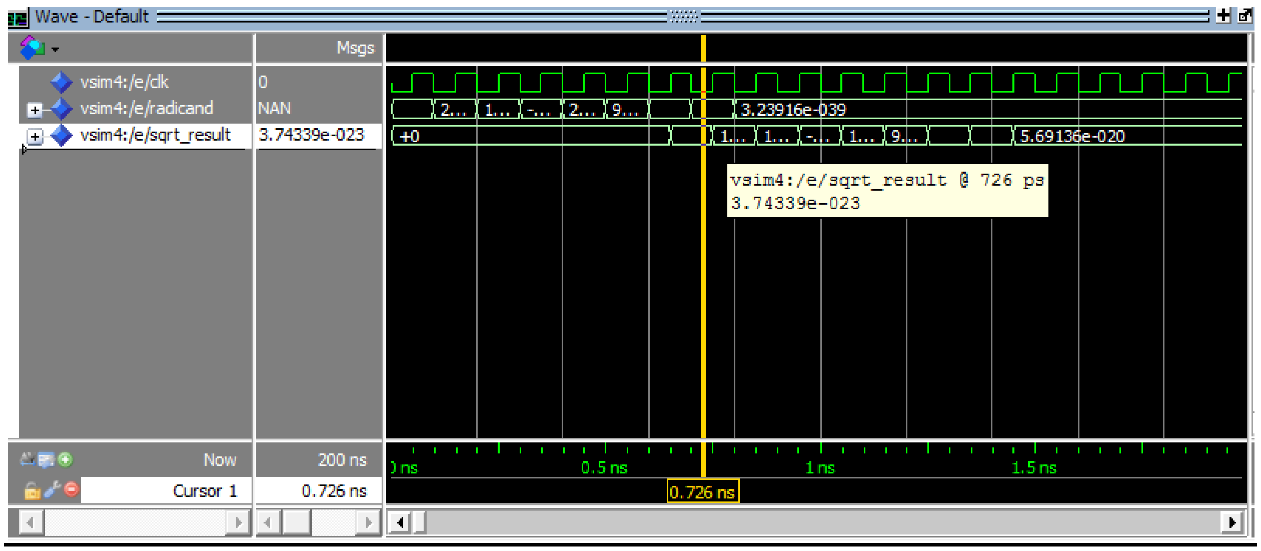
	Msgs
vsim3:/e/bypass_si...	1
vsim3:/e/clk	1
vsim3:/e/multiplicand	5.67980e-028
vsim3:/e/multiplicator	-1.08685e-010
vsim3:/e/mul_result	-6.17307e-038
Now	200 ns
Cursor 1	0.48 ns

Figure 10: Multiplier simulation results in the case bypass='1'

Table 3: Square Root Simulation Results for a set of exemplar inputs

Sr no.	Input1	Output
1	1.40E-45	3.74E-23
2	2.70E-42	1.64E-21
3	1.58E-38	1.26E-19
4	-1.47E-12	-1.47E-12
5	2.19E-38	1.48E-19
6	9.18E-41	9.58E-21
7	5.51E-28	2.26E-14
8	3.24E-39	5.69E-20

In the case of the Square Root and the Divider, the HDL code was verified, but it was not modified to support the "configurable pipeline". Table 3, Table 4, and describe the simulation activity on the two accelerators. The latency of the Square Root is 12 cycles that of the divider is 9 cycles.



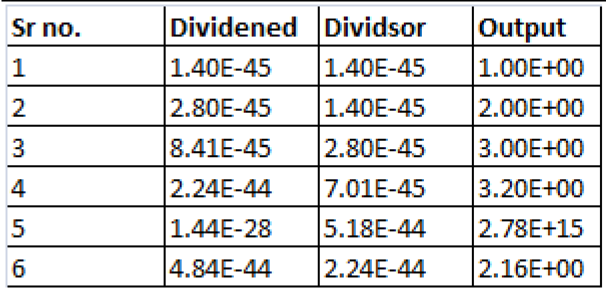
```
vsim4:/e/clk            0
vsim4:/e/radicand       NAN            2... 1... -... 2... 9...  3.23916e-039
vsim4:/e/sqrt_result    3.74339e-023   +0  1... 1... -... 1... 9...  5.69136e-020

Now       200 ns
Cursor 1  0.726 ns

vsim4:/e/sqrt_result @ 726 ps
3.74339e-023
```

Figure 11: Square root simulation results

Table 4: Divider Simulation Results for a set of exemplar inputs



Sr no.	Dividened	Dividsor	Output
1	1.40E-45	1.40E-45	1.00E+00
2	2.80E-45	1.40E-45	2.00E+00
3	8.41E-45	2.80E-45	3.00E+00
4	2.24E-44	7.01E-45	3.20E+00
5	1.44E-28	5.18E-44	2.78E+15
6	4.84E-44	2.24E-44	2.16E+00

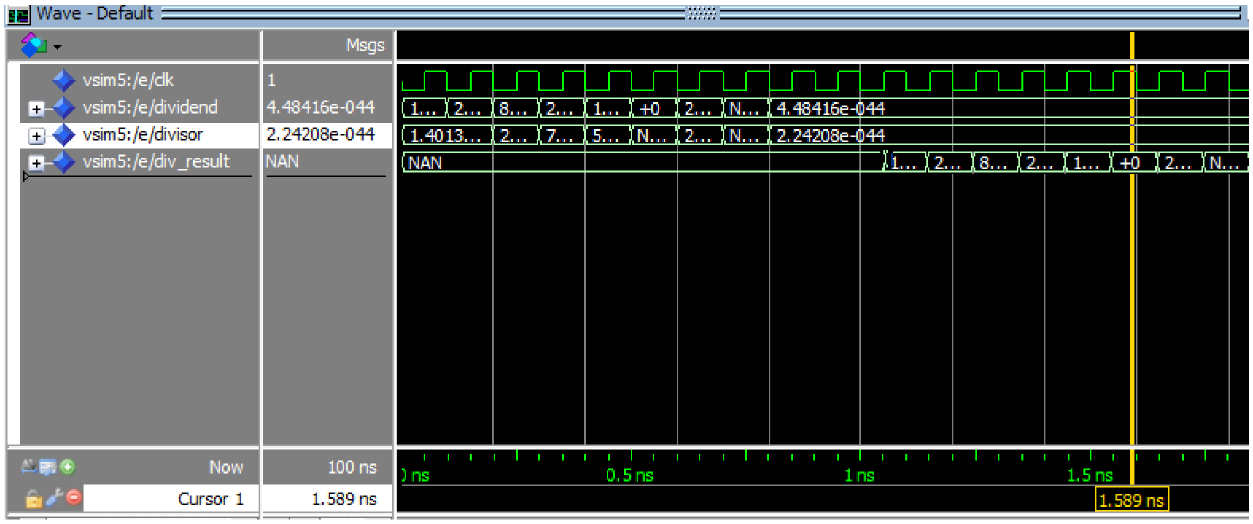


Figure 12: Divider Simulation results

3.4 Physical Implementation of the Hardware accelerators

The synthesis process transforms behavioral RTL code into a real circuit based on pre-designed cells that depend on the chosen target technology. In this work, implementation was provided for only the FP adder and the FP Multiplier. After the synthesis step, we can evaluate the performance of the circuit based on the following parameters:

1. **AREA:** The area reports provide a list of the cells used during synthesis, and the occupation of the same
2. **TIMING:** timing reports are used in order to understand how fast a given design can perform its task. A faster circuit will perform more computations that leads to a better performance in terms of FLOPS (Floating point Operations per Second).
3. **POWER:** Power reports depict the power consumption dissipated by the circuit. Power consumption is normally classified in terms of Dynamic power that depends on the clock frequency and the variation of the inputs, and Leakage power that is normally constant across all computation, and is represented by the consumption of the circuits when the clock and the inputs are constant

In this work, as reported in the previous sections, synthesis was targeted on two different technologies: Altera Cyclone FPGAs and CMOS 28nm FDSOI. Synopsys dc_shell was utilized for the CMOS-VLSI implementation of the FP operators and Altera Quartus II for the FPGA implementation. The circuit was synthesized on both technology targets both BEFORE and AFTER the insertion of the bypass functionality to enable a fair comparison.
This chapter will describe the timing / area result, while a comprehensive power analysis will be provided in chapter 4.

3.4.1: Synthesis on FPGA Technology

On FPGA support, the area occupied by the computational accelerator is provided in terms of the required utilization of FPGA elements (Logic Elements and embedded registers). In the case of the Adder, due to its intrinsic complexity, Alters Cyclone ICs utilized the synthesis tools for implementing pipeline registers; has embedded SRAM memory blocks

Table5: Synthesis performance of the FP Multiplier on Altera Cyclone FPGA technology

Accelerator Version	**Logic Elements**	**Registers**	**Memory Blocks**	**Maximum Frequency**
No Pipeline	**1065**	**0**	**0**	**40 MHz**
Pipelined	**1175**	**251**	**0**	**87 MHz**
With Bypass	**1328**	**251**		**85 MHz (Bypass=0) 30 MHz (Bypass=1)**

Table6: Synthesis performance of the FP Adder on Altera Cyclone FPGA Technology

Accelerator Version	**Logic Elements**	**Registers**	**Memory Blocks**	**Maximum Frequency**
No Pipeline	**5348**	**0**	**0**	**20 MHz**
Pipelined	**5046**	**370**	**198**	**37 MHz**
With Bypass	**5759**	**370**	**192**	**34 MHz (Bypass=0) 18 MHz (Bypass=1)**

From the results reported above, we can see that the adder is way more complex than the multiplier, which could be expected as floating point addition involves a normalization step that is absent in FP multiplication. The addition of the pipeline provides significant speedup (40 to 87MHz for the multiplier, 20 to 57 MHz for the adder) that could be expected, at a moderate hardware cost. This is due to a peculiar feature of the FPGAs: as every logic element in the FPGA feature an embedded register, the addition of pipeline registers is sometimes possible in FPGA using the LEs already utilized for the computation, so that pipelining is possible ad moderate cost. Most interestingly, the addition of the bypass functionality comes at a very moderate cost (~10%) in terms of both area and timing. If this modification allows power saving (this aspect will be analyzed in detail in chapter 4) then given its affordable hardware cost the proposed modification appears definitely appealing for the design of a dynamically power aware architecture.

3.4.1: Synthesis on VLSI-CMOS Technology

When synthesizing on FPGAs, the area occupation of the circuit is not very sensitive on the target frequency. On the other hand, the CMOS-VLSI solution can show very significant area variation depending on the target frequency. In order to evaluate fully the design tradeoff between area occupation and available clock speed, the CMOS synthesis process was repeated on a set of different clock periods. Note that for clocks below 2 ns (500MHz), the timing constraints are violated. As described in Table 7, Area, leakage power, and dynamic power predictably show similar trends vs. frequency, as the power of course strongly depends on the number and size of standard cells composing the design: all plots gradually increase when the frequency raises. Then they go through a stable region until the frequency goes up to 400MHz. After that figure, all of them increase sharply as frequency increases. To select the best tradeoffs among timing, area, and power consumption, the target frequency for this design should be around 400MHz, where we have a higher frequency as well as reasonable area and power consumption.
The better values of power and area suggest this option, event though technically the design could be pushed up to 500 MHz. Similar evaluations, not reported in this document for brevity, led to the estimation of 230MHz as the best speed vs area tradeoff for the FP adder design.

Table7: Evaluation of the ideal area/speed tradeoff in the multiplier design

Period	Frequency	Area (um2)	Leakage power	Dynamic Power	Slack	Status
10	100	7482.230386	64.0328	780.2682	6.9	Met
8	125	7483.372786	64.0567	974.0652	4.9	Met
6	166.6666667	7477.171186	63.9901	1298.7	2.9	Met
4	250	7512.585586	64.1843	1952.4	0.89	Met
3.5	285.7142857	7632.2112	65.3802	2228.9	0.4	Met
3.4	294.1176471	7685.4144	62.5269	2322	0.3	Met
3	333.3333333	7734.864005	62.833	2631.3	0.1	Met
2.5	**400**	**7886.640003**	**82.8534**	**3194.98**	**0**	**Met**
2	500	8656.438405	125.6647	4430.9	0.01	Met
0.5	2000	9299.13603	811.6769	55337.5	-0.42	Violated
0.2	5000	9804.566427	870.0426	155338	-0.74	Violated

Figure 13: Area, Leakage and Dynamic power vs Timing period in ns for the FP multiplier

Table 8: Synthesis performance of the FP Multiplier in cmos28 FDSOI Technology. Working conditions are slow, 0.85V, 125C

Accelerator Version	**Area (mm2)**	**Area (Kgates)**	**Maximum Frequency**	**Peak Speed degradation due to bypass**
No Pipeline	5940	28	440 MHz	
Pipelined	7260	36	880 MHz	
With Bypass	7886	39	820 MHz (Bypass=0) 400 MHz (Bypass=1)	6.9 %

Table9: Synthesis performance of the FP Adder in cmos28FDSOI Technology. Working conditions are slow, 0.85V, 125C

Accelerator Version	**Area (mm2)**	**Area (Kgates)**	**Maximum Frequency**	**Peak Speed degradation due to bypass**
No Pipeline	22101	110	310 MHz	
Pipelined	28240	141	640 MHz	
With Bypass	29876	149	480 MHz (Bypass=0) 290 MHz (Bypass=1)	7.5%

In order to describe the area occupation of a design in a Technology-independent way, very often in VLSI the metric “Equivalent Kilogates” is used. An equivalent gate is described as the area of a basic NAND gate. The area of the FP multiplier is in the rage of 35/40Kg (which is roughly 2 times the area of an integer 32x32 bit multiplier, or 8 times the area of a standard ALU, which is very reasonable.

Of course, the reported values are much higher of that in the FPGA case, but this could be expected. Other than being the FPGA a programmable device, hence very redundant with respect to the CMOS alternative, this CMOS implementation makes use of a very aggressive technology node, that provides low area and very low interconnect capacitance between cells, thus leading to very aggressive timing results. It should be noted that the results provided here are only after synthesis, so that a degradation of 20/30% can be expected after the place and route step.

In conclusion, from the reported values, we can again determine that the overhead in both timing and area related to the addition of the Bypass functionality is relatively low (<10%) so that the methodology appears affordable. The overhead introduced by the pipeline is very relevant, as in every CMOS-VLSI technology given the large area of FF cells, but as described in detail in the previous section the pipeline is indispensable to provide high-speed during the peaks of high computation demand. Our aim, in the context of this work, is to try to mitigate the power consumption imposed by the pipeline during low-speed computation, while affecting as little as possible the top speed. In this case, the addition of the bypass functionality only affected the top speed by roughly 7.5% for both accelerators, which can be considered acceptable in presence of significant power mitigation.

4 POWER MEASUREMENTS

The aim of this MENG project was to introduce to a processor architecture a minor modification to the pipeline of two hardware accelerators, in order to provide the higher possible power saving without significantly affecting the peak performance of the device.

Figures reported in chapter 3 demonstrated that the proposed modification has a very marginal impact on the area of the accelerators, and most importantly affects only up to ~7.5% the peak computation speed of the two accelerators. It is now interesting to investigate if the applied modification does provide the expected gain in power consumption.

Differently from area and timing, power measurement in digital circuits is a very tricky topic, as the consumption depend strongly on the inputs. In the case discussed in this document, this is even more important because the techniques that were applied in the design under evaluation, clock gating and Dynamic Voltage and Frequency Scaling strongly depend on the state of the inputs. For this reason, in order to provide a reliable power estimation, we need to refer to a specific application case and in this context; we will utilize the LDA machine-learning algorithm (17). LDA is a simple, yet largely utilized reference for the classification of sensor data in the realm of embedded systems. The critical kernel of LDA reside in the calculation of an inverse matrix of floating point numbers, and in particular in the calculation of the matrix determinant. As such, LDA requires the computation of a set of consecutive FP multiplications, followed by a set of additions. The number of consecutive multiplications and additions on the critical computation kernel depend on the software implementation of the determinant algorithm. In this analysis, we focused on a case that involves the determinant of a 3x3 matrix:

$$\det(A)=A_{11}(A_{22}A_{33}-A_{23}A_{32})-A_{12}(A_{21}A_{33}-A_{23}A_{31})+A_{13}(A_{21}A_{32}-A_{22}A_{31})$$

This calculation involves 6 consecutive Multiplications, 3 sums, 3 more multiplications and 3 more sums. This information is relevant, because it justifies at high speed the use of a pipeline: The 6 multiplications trigger once per cycle without need to wait for the relative result to appear. If there were a dependency between the result of a multiplication and the input of the next, the pipeline would not be applicable.

We assume that the processor will perform the same computation both during peak performance and in quasi-idle state, only at a different speed. The quasi-idle speed is conventionally set at 20MHz for both the FPGA and VLSI solution, while the peak speed is the top speed allowed by the technology.

It should be noted that the consumption of the memory accesses required to perform the computation would be relevant. Nevertheless, in a comparative analysis, it would be the same for all the solutions indicated below so it can be neglected from the reporting.

4.1: Power Measurements on Altera Cyclone IV FPGA

Running the calculation introduced above on the “Crescent Beach” floating point processor, we obtain the following consumption numbers for the accelerators (Table 10, Table 11):

Table10: Power Consumption of the floating-point multiplier accelerator on Altera Cyclone FPGA computing the algorithm described above. Working conditions are typical, 1V, 25C.

	Standard Pipeline	**Configurable Pipeline**	
Dynamic (Bypass=0)	0.23 mW/MHz	0.27 mW/MHz	
Dynamic (Bypass=1)		0.16 mW/MHz	

Table11: Power Consumption of the fp adder accelerator on Altera Cyclone FPGA computing the algorithm described above. Working conditions are typical, 1V, 25C.

	Standard Pipeline	**Configurable Pipeline**	
Dynamic (Bypass=0)	2.73 mW/MHz	3 mW/MHz	
Dynamic (Bypass=1)		1.6 mW/MHz	

When running at the maximum speed, the accelerators need to sustain the maximum speed: 85 MHz for the multiplier and 35 MHz for the adder. The total dynamic consumption at full speed is

Multiplier => 0.23 * 85 = 19.55 mW for the standard pipeline

=> 0.27 * 85 = 22.55 mW for the configurable pipeline

Adder => 2.73 * 35 = 95.5 mW for the standard pipeline

3 * 35 = 105 mW for the configurable pipeline

The slight increase at full speed is justified by the gain at low speed, where the pipeline can be bypassed thus leading to significant power gain. At the conventional rate of 20 MHz

Multiplier => 0.23 * 20 = 4.6 mW for the standard pipeline

0.16 * 20 = 3.2 mW for the configurable pipeline

Adder => 2.73 * 20 = 54.6 mW for the standard pipeline

1.6 * 20 = 32 mW for the standard pipeline

Since we can expect that the peak computation will be active on less than 1/100 of the device lifetime, the net power consumed by the proposed architectural modification is around 60% of the standard architecture in the case of the adder, and 70% in the case of the multiplier, which can be considered a very positive result.

4.2: Power Measurements on VLSI - CMOS 28 FDSOI

As described in the previous sections, CMOS 28 FDSOI is an advanced CMOS technology node that provides significant space for power optimization using DVFS. The node allows very high speeds using the nominal voltage: the single precision floating-point multiplier described in section 3 can reach up to 820 MHz of clock frequency when making use of the full pipeline. If the full speed of the device is not needed, the voltage can be scaled down to smaller voltages, allowing to obtain the same performance at lower power. This trend is described in Figure 14: the blue plot shows the power consumption of the Multiplier in uW when working at the nominal supply voltage of Vdd=1.0 V. We can describe the power consumption of a digital CMOS circuit as

*Total Power(uW) = Leakage(uW) + Dynamic(uW/MHz) * FREQ*

The leakage component is very low (~100uW) so that the overall consumption is essentially a linear function of the working frequency.

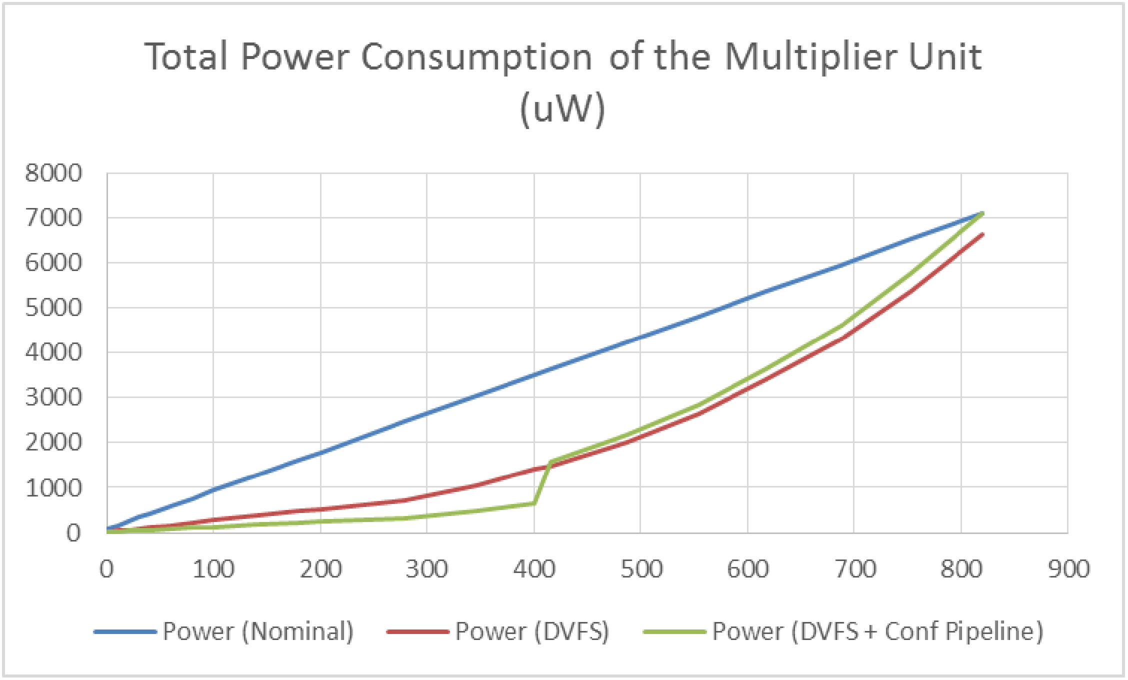


Figure 14: Power consumption related to the FP Multiplier accelerator unit as a function of the working frequency

The red plot shows the benefit of applying DVFS to the FP multiplier unit: when the working frequency is lower, it is possible to provide a lower Voltage supply to the unit, completing the computation at a significantly lower power consumption. This DVFS methodology is commonly used in SoC design today, and does not represent any element of novelty.

Figure 14 also highlights a well-known limitation of the DFVS approach for lower frequencies. In particular after 400 MHz, the pipeline is completely useless and represents a heavy burden: in fact, as specified in Table (section 3 of this document), at 400 MHz and below the multiplier completes its calculation without any need of using a pipeline. Applying the modifications described in section 3 of this document, that are the core of this MENG project, the pipeline registers of the multiplier unit are “frozen”, and the related clock tree is stuck at a ‘0’ logic value. As a result, as shown in the green plot in Figure 14, below 400MHz the power consumption of the multiplier unit featuring the “Configurable Pipeline” is lower than the one of the red plot, representing the multiplier with a standard pipeline.
Of course, the circuit overhead necessary to support the “Configurable Pipeline” induces slightly higher consumption than the standard case at frequencies > 400 MHz. This is described in Figure 14 by the fact that the green plot is slightly higher than the red plot for that frequency range.
But, according to the analysis provided in section 2 of this document, the design of the accelerators analyzed in this project is targeted at IoT applications. In this class of application, it can be assumed that the device will be in a high speed mode for a very small percentage of its lifetime (<1/100). In this conditions,

As an example, let us suppose that for 1/50th of the time the system works at the peak speed of 600 MHz (which is a very conservative case, as the ration could be much higher in real life applications):

Standard Pipeline:	3.4 mW
Configurable Pipeline:	3.6 mW

In the remaining 49/50th of the time, the system will work in a quasi-idel state at 40MHz performing the same computation

Standard Pipeline:	120 uW
Configurable Pipeline:	64 uW

Supposing 50s of computation, 1 of which is performed at high speed, the overall energy consumption would be

*Standard Pipeline: 120uW*49s + 3.4mW*1s = 9.280 mW*
*Configurable Pipeline: 64uW*49s + 3.6*1s   = 6.736 mW*

In conclusion, the proposed power mitigation strategy enables on the circuit used as reference a net energy decrease of ~30% at the price of minimal overhead in terms of timing and area overhead.

Similar results could be extracted for the floating point Adder/Subtractor accelerator. Unfortunately, the power analysis of the adder could not fit in the timeline of the present work and will be performed in the context of the following steps of the processor design.

5 CONCLUSIONS

This MENG project was realized in the context of the design of a Floating Point processor for Internet-of-Things application. The processor, code named “Crescent Beach”, is being developed as an open source hardware IP in Simon Fraser University, and is based on the well-known RISC-V instruction set developed at University of California-Berkeley.

The processor is targeted at the processing and classification of data derived from embedded sensors of various nature. The main specification of the processor is to be capable to deliver high peak computational power, while being able to function for long periods in a low-power mode.

The specific activity of this MENG project was to

1) Verify the functionality of all Floating Point accelerators embedded in the design
2) Implement the same accelerators in FPGA Altera Cyclone Technology and VLSI-CMOS 28nm FDSOI
3) Apply to two highly utilized hardware floating-point accelerators that are part of the processor design a peculiar technique, defined “configurable pipeline”. The technique consists in adding a programmable bypass channel to every pipeline register in the accelerator architecture in order to exclude during computation the pipeline from the device functioning when the desired data rate allow that, enabling significant power saving by means of power gating or dynamic voltage and frequency scaling.

While showing a moderate overhead on the accelerators area and timing performance (<10% both on Altera FPGA and CMOS 28nm SOI technology), the proposed strategy allowed to

- On FPGA, reduce power consumption on FPGAs to 60%/70% of the original consumption using the Linear Discriminant Analysis (LDA) algorithm as reference (LDA is widely used for data classification in embedded sensors systems).
- On CMOS VLSI technology, the application of the configurable pipeline mechanism on top of the well-known Dynamic Voltage and Frequency Scaling allowed to reach a power saving of ~30% on an exemplar test case again based on the LDA algorithm.

This strategy applied to processor architectures is to the best of the author’s knowledge not reported in any related literature, and represent an element of innovation that may lead to promising applications in the design of pipelined hardware accelerators. As such, it represents the most significant result of this MENG project.

REFERENCES

1) *"Internet of things for smart cities," A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, Internet of Things Journal, IEEE, vol. 1,no. 1, pp. 22–32, Feb 2014.*
2) *Energy efficiency in the future internet: a survey of existing approaches and trends in energy aware fixed network infrastructures R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti IEEE Communications Surveys & Tutorials, vol. 13, pp. 223-244, Jul. 2011.*
3) *Intelligent DC Homes in Future Sustainable Energy Systems: When efficiency and intelligence work together., Rodriguez-Diaz, E.; Vasquez, J.C.; Guerrero, J.M.Consumer Electronics Magazine, IEEE Year: 2016, Volume: 5, Issue: 1Pages: 74 - 80, DOI: 10.1109/MCE.2015.2484699*
4) *Multi-Purpose Capacitive Proximity Sensing System for Industrial Safety Applications, Fan Xia et al, IEEE Sensors 2016, October 2016, Orlando, FL*
5) *Intelligent cameras and embedded reconfigurable computing: a case-study on motion detection, Claudio Mucci et al, System-on-Chip, 2007 International Symposium on,Year: 2007, Pages: 1 - 4, DOI: 10.1109/ISSOC.2007.4427440*
6) *Hardware/Software Co-Design, edited by Giovanni DeMicheli, Mariagiovanna Sami, Springer Publishing co 1996 ISBN: 9400901879*
7) *Emerging Yield and Reliability Challenges in Nanometer CMOS Technologies, G. Gielen, P. De Wit, E. Maricau et al, Proceedings of the IEEE Symposium on Design and Test in Europe, 2008*
8) *Design methodology for low-power embedded microprocessors, Manuzzato, Andrea; Campi, Fabio; Liberali, Valentino; Pandini, Davide,Power and Timing Modeling, Optimization and Simulation (PATMOS), 2013 23rd International Workshop on Pages: 259 - 264 DOI: 10.1109/PATMOS.2013.6662187*
9) *The RISC-V instruction set, Andrew Waterman; Yunsup Lee; Rimas Avizienis; Henry Cook; David Patterson; Krste Asanovic 2013 IEEE Hot Chips 25 Symposium (HCS) Year: 2013 Pages: 1 - 1, DOI: 10.1109/HOTCHIPS.2013.7478332*
10) RISC-V: The Free and Open RISC Instruction Set Architecture. The RISC-V foundation, https://riscv.org/*. Accessed August 2016.*
11) Computer Architecture, a Quantitative Approach, J.*Hennessy, D.Pattersson, Morgan Kaufman, ISBN-13: 978-0123838728*
12) *A FPGA Implementation of An Open-Source Floating-Point Computation System, C. Brunelli; F. Garzia; J. Nurmi; C. Mucci; F. Campi; D. Rossi 2005 International Symposium on System-on-Chip Year: 2005 Pages: 29 - 32, DOI: 10.1109/ISSOC.2005.1595636*

13) *Dynamic voltage scaling for commercial FPGAs, C. T. Chow; L. S. M. Tsui; P. H. W. Leong; W. Luk; S. J. E. Wilton, Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, Year: 2005, Pages: 173 - 180, DOI: 10.1109/FPT.2005.1568543*

14) Energy Optimization in Commercial FPGAs with Voltage, Frequency and Logic Scaling, Jose Luis Nunez-Yanez; Mohammad Hosseinabady; Arash Beldachi, IEEE Transactions on Computers, Year: 2016, Volume: 65, Issue: 5, Pages: 1484 - 1493, DOI: 10.1109/TC.2015.2435771

15) de2-115.terasic.com. Accessed august 2016

16) 28 nm FD-SOI technology platform for high-speed low-voltage digital applications, N.Planes e. al, in Proc. Symp.VLSI Technology (VLSIT)," in Proc. Symp. VLSI Technology (VLSIT), 2012

17) D. Yang, "Implementation of a Wearable Feedback System Monitoring the Activities of Upper-extremities", MENG Project presentation, Simon Fraser University, April, 2015