JONATAS FARIA ROSSETTI

HARDWARE DESIGN AND PERFORMANCE ANALYSIS FOR CRYPTOGRAPHIC SPONGE BLAMKA

PROJETO DE HARDWARE E ANÁLISE DE DESEMPENHO PARA A ESPONJA CRIPTOGRÁFICA BLAMKA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2017

JONATAS FARIA ROSSETTI

HARDWARE DESIGN AND PERFORMANCE ANALYSIS FOR CRYPTOGRAPHIC SPONGE BLAMKA

PROJETO DE HARDWARE E ANÁLISE DE DESEMPENHO PARA A ESPONJA CRIPTOGRÁFICA BLAMKA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:

Engenharia de Computação

Orientador:

Prof. Dr. Wilson Vicente Ruggiero

São Paulo
2017

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 15 de julho de 2017.

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Rossetti, Jonatas Faria
Hardware Design and Performance Analysis for Cryptographic Sponge BlaMka

Projeto de Hardware e Análise de Desempenho para a Esponja Criptográfica BlaMka/ J. F. Rossetti. – ed. rev. – São Paulo, 2017.
112 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

1. Hardware. 2. Análise de Desempenho. 3. Criptografia. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais (PCS). II. t.

“All have their worth and each contributes to the worth of the others.”

J. R. R. Tolkien, The Silmarillion

AGRADECIMENTOS

Aos meus pais e minha família pelo incentivo à leitura e aos estudos.

Ao Professor Wilson Ruggiero pela orientação, motivação e incentivo ao longo de todo o trabalho.

Ao Professor Marcos Simplicio pelos comentários e sugestões valiosos e por seu incentivo durante todo o trabalho.

Ao Professor Alexandre Tenca pelos seus conselhos, comentários e esclarecimentos de questões importantes sobre projeto de hardware.

À Synopsys e à USP pela disponibilização das ferramentas necessárias para a obtenção dos resultados experimentais.

.....

To my parents and my family for the incentive to read and to study.

To Professor Wilson Ruggiero for the guidance, motivation, and encouragement throughout the work.

To Professor Marcos Simplicio for the valuable comments and suggestions provided, and for his encouragement throughout the work.

To Professor Alexandre Tenca for his valuable pieces of advice, comments, and clarification on important hardware design issues.

To Synopsys and USP for providing the necessary tools to obtain the experimental results.

RESUMO

Para avaliar o desempenho de um projeto de hardware, é necessário selecionar as métricas de interesse. Várias métricas podem ser escolhidas, mas em geral três delas são consideradas básicas: área, latência e potência. A partir delas, podem ser obtidas outras métricas de interesse prático, tais como vazão e consumo de energia. Essas métricas relacionam-se entre si, criando trade-offs que os projetistas precisam conhecer para executar as melhores decisões de projeto. Alguns trabalhos abordam o projeto de hardware otimizado para melhorar uma dessas métricas. Em outros trabalhos, as otimizações são feitas para duas delas, mas sem analisar como uma terceira métrica se relaciona com as demais. Outros analisam o trade-off entre duas dessas métricas. Entretanto, a literatura carece de trabalhos que analisem o comportamento de três métricas em conjunto. Neste trabalho, pretendemos contribuir para preencher essa lacuna, propondo um método que permita a análise de trade-offs entre área, potência e vazão. Para verificar o método proposto, foi escolhida a função de permutação da esponja criptográfica BlaMka como estudo de caso. Até o momento, nenhuma implementação em hardware foi encontrada para esse algoritmo. Dessa forma, uma contribuição adicional é apresentar seu primeiro projeto de hardware. Circuitos combinacionais e sequenciais foram projetados e sintetizados para ASIC e FPGA. Com os resultados de síntese, foi realizada uma análise de desempenho detalhada para cada plataforma, a partir de uma análise unidimensional, passando por uma análise bidimensional e culminando em uma análise tridimensional. Duas técnicas foram apresentadas para tal análise tridimensional, chamadas abordagem das projeções e abordagem dos planos. Embora passivel de melhorias, o método apresentado é um passo inicial mostrando que, de fato, um trade-off entre três métricas pode ser analisado, e que também é possível encontrar pontos de desempenho balanceado. A partir das duas abordagens, foi possível derivar um critério para selecionar otimizações quando há restrições, como um faixa de vazão desejada ou um tamanho físico máximo, e quando não há restrições, caso em que é possível escolher a otimização com o desempenho mais balanceado.

ABSTRACT

To evaluate the performance of a hardware design, it is necessary to select the metrics of interest. Several metrics can be chosen, but in general three of them are considered basic: area, latency, and power. From these, other metrics of practical interest such as throughput and energy consumption can be obtained. These metrics relate to one another by creating trade-offs that designers need to know to execute the best design decisions. Some works address optimized hardware design for improving one of these metrics. In other works, optimizations are made for two of them. Others analyze the trade-off between two of these metrics. However, the literature lacks of works that analyze the behavior of three metrics together. In this work, we intend to contribute to bridge this gap, proposing a method that allow analyzing trade-offs among area, power, and throughput. To verify the proposed method, the permutation function of cryptographic sponge BlaMka was chosen as a case study. No hardware implementation has been found for this algorithm yet. Therefore, an additional contribution is to provide its first hardware design. Combinational and sequential circuits were designed and synthesized for ASIC and FPGA. With the synthesis results, a detailed performance analysis was performed for each platform, starting from a one-dimensional analysis, going through a two-dimensional analysis, and culminating in a three-dimensional analysis. Two techniques were presented for such analysis, namely projections approach and planes approach. Although there is room for improvement, the proposed method is a initial step showing that, in fact, a trade-off between three metrics can be analyzed, and that it is also possible to find balanced performance points. From the two approaches presented, it was possible to derive a criterion to select optimizations when we have restrictions, such as a desired throughput range or a maximum physical size, and when we do not have restrictions, in which case we can choose the optimization with the most balanced performance.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AES	Advanced Encryption Standard
ASIC	Application-Specific Integrated Circuit
ASM	Algorithmic State Machine
CLB	Configurable Logic Block
CRHF	Collision-Resistant Hash Function
CS	Circuit-Switched
CSA	Carry-Save Adder
DSP	Digital Signal Processor
FPGA	Field-Programmable Gate Array
GE	Gate Equivalent
GPU	Graphics Processing Unit
HDL	Hardware Description Language
I/O	Input/Output
LUT	Look-Up Table
MD	Message-Digest
NOC	Network-On-Chip
NIST	National Institute of Standards and Technology
OWHF	One-Way Hash Function
PBKDF	Password-Based Key Derivation Function

PHC	Password Hashing Competition
PHS	Password Hashing Scheme
RAM	Random Access Memory
ROM	Read-Only Memory
RTL	Register-Transfer Level
SHA	Secure Hash Algorithm
SHAT	Sponge Hash Algorithm
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

LIST OF SYMBOLS

$\oplus$	Bitwise Exclusive-OR (XOR) logical operation
$+$	Wordwise addition operation
$\cdot$	Wordwise multiplication operation
lsw	Least significant word
$\ll$	Left shift operation
$\ggg$	Right rotation operation
t_{pd}	Propagation delay

1 INTRODUCTION

To evaluate the performance of a hardware design, it is necessary to select the metrics of interest. Several metrics can be chosen, but in general three of them are considered basic: area, latency, and power. From these, other important metrics of practical interest such as throughput, cost, and energy consumption can be obtained. Area, latency, and power relate to one another by creating trade-offs that the designers need to know to execute the best design decisions and accomplish their goals. For example, when design techniques are used for a small circuit area, there is usually an increase in latency. If the goal is a low latency circuit, there will usually be an increase in power consumption. In addition, increasing the area of the circuit leads to an increase in power. When one of these parameters is changed, the other two vary in response.

It can be established as a design hypothesis that one of the metrics is limited to a certain margin, while the other two can be varied in order to find the best trade-off between them. In another approach, one can accept variations in one of the parameters while a trade-off between the other two is analyzed. There are projects that aim to obtain the best possible performance specifically in relation to one of these metrics, accepting the consequences in the performance of the others, not because they are indifferent to such variations, but because the application of interest imposes its restrictions and costs on this metric. Additionally, other projects, besides a goal for maximum optimization for only one metric, still aim to obtain satisfactory results in relation to the remaining ones. Regardless of the type of trade-off analyzed, usually the analysis

is focused in one or two metrics, becoming, in these two cases, a one-dimensional and a two-dimensional analysis, respectively. One possibility is to extend this analysis to a three-dimensional case where a trade-off between three metrics is analyzed. Any metrics that have trade-offs between each other could be used in this three-dimensional analysis, such as throughput, number of clock cycles, maximum frequency, cost, or efficiency measures such as energy-per-bit and throughput-per-area, among others.

1.1 Motivation

The intensive use of mobile devices in a variety of applications, from entertainment to banking, has created a growing need of security solutions for such platforms (DAPP, 2012; NACHENBERG, 2011; LOOKOUT, 2014; FEDERAL RESERVE BOARD, 2016; SVAJCER, 2014; DIMENSIONAL RESEARCH, 2013). Despite the increasing processing power available in modern devices, many still have limited performance. Furthermore, they have restrictions on the physical size and mainly in energy consumption. In the information security field, various algorithms are applied to provide users with a variety of services such as confidentiality, integrity, authenticity, availability, and non-repudiation. Although many of the algorithms are implemented in software, there is a rapidly growing demand for hardware implementations, be it for the nature of the application, such as embedded technologies and sensor networks, be it for requirements such as performance, energy, and cost (TANOUGAST et al., 2012; INIEWSKI, 2012). An important class of algorithms is the cryptographic hash functions used for mapping data of arbitrary size to a bit string of a fixed size (MERKLE, 1979), that allows the construction of several cryptographic primitives such as stream ciphers, message authentication codes, pseudo-random functions, password hashing schemes, among others. The cryptographic sponge functions, a generalization of cryptographic hash functions, take an input bit stream of any length and produce an output bit stream of any desired length (BERTONI et al., 2011). Considering these observations, the development of ideas

and design techniques that allow analyzing a three-dimensional trade-off between area, throughput, and power, thus allowing a balance between them, is of great value within the field of hardware design, especially for implementations of cryptographic algorithms.

1.2 Goals

Our goal is to present a three-dimensional performance analysis for a hardware implementation of a cryptographic algorithm, observing the area utilization, circuit throughput (derived from latency), and the power consumption. The results presented herein can contribute to the problem of analyzing three-dimensional trade-offs between any metrics. Furthermore, some ideas that can be used in this type of analysis for hardware implementations of any other algorithm can be delineated. To allow this three-dimensional performance analysis of a hardware design, it is necessary to choose an algorithm to be implemented as a case study. The algorithm chosen was the permutation function of the cryptographic sponge BlaMka (SIMPLICIO et al., 2015).

The BlaMka algorithm (SIMPLICIO et al., 2015) is a cryptographic sponge function that has been used in Password Hashing Schemes (PHS) (NIST, 2010), which are algorithms capable of generating a sequence of pseudo-random bits from a certain user-defined password, usually employed in modern password-based authentication systems to protect against brute force attacks. Examples of password hashing schemes are Lyra2 (SIMPLICIO et al., 2015; ANDRADE et al., 2016) and Argon2 (BIRYUKOV; DINU; KHOVRATOVICH, 2016), both finalists of the Password Hashing Competition (PHC) (PHC, 2013). It is worth noting that BlaMka was proposed by the same authors of Lyra2 and presented as a modification of the Blake2b algorithm (AUMASSON et al., 2013; AUMASSON et al., 2014), integrating multiplications in some stages of its execution, specifically in its permutation function. The inclusion of multiplications aims to reduce the hardware performance and to increase the costs for brute force attacks that

usually use hardware platforms (SIMPLICIO et al., 2015). As far as we are concerned, there is no hardware implementation for BlaMka, specifically for its permutation function, which is the structure that incorporates the multiplications and what differentiates BlaMka from Blake2b. Therefore, an additional contribution is to provide its first hardware design, besides verifying the impact of the inclusion of multiplications in its performance compared with a hardware implementation for the permutation function of Blake2b.

1.3 Method

To successfully accomplish the goals of this work, we follow a well-defined working method. Initially, a literature research is carried out to survey the state of the art in hardware design, both for cryptographic and general applications, selecting the relevant works that present results and analyses of different trade-offs between latency, area, throughput, power, among other metrics. As the next step, the hardware design and implementation of the BlaMka permutation function algorithm is performed, using VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) to describe the architectures. The resulting circuits are synthesized into programmable logic devices, specifically in FPGAs (*Field-Programmable Gate Array*), and in application specific circuits, the ASICs (*Application-Specific Integrated Circuit*). Different architectures are implemented to address the trade-offs between circuit throughput, area utilization, and power consumption. These metrics are used as a measure of performance for the proposed architectures. Other metrics such as throughput-to-area and energy-per-bit are used to measure the efficiency of the implementations. The performance analysis is conducted using the idea of a three-dimensional analysis, observing together the behavior of the selected basic metrics. In addition, the results for BlaMka permutation function is compared with the results for a hardware implementation of the Blake2b permutation function, and the theoretical properties of the algorithm will

be analyzed experimentally. The thesis writing is developed in conjunction with the previous steps. Note that such steps are not strictly sequential, but are interleaved throughout the development of the work.

1.4 Document Organization

This document is organized as follows. Chapter 2 briefly describes some theoretical and design concepts. Chapter 3 presents the related works in the literature containing analyses of several trade-offs between area, power, throughput, latency, among other possible metrics in hardware design. Chapter 4 presents the hardware design and implementation of the BlaMka permutation function algorithm. Chapter 5 presents experimental results and performance analyses for the implemented architectures. Chapter 6 presents the final conclusions and also outlines some future work.

2 THEORETICAL AND DESIGN ASPECTS

In its first part, this chapter presents a brief review of the main concepts of hardware design. In the second part, some concepts related to cryptography are presented.

2.1 Hardware Design Aspects

In this section, we present the main concepts of combinational and sequential circuits, as well as two main approaches for hardware synthesis, FPGAs and ASICs, and their characteristics. We describe the main metrics involved in the performance analysis of a hardware design, the hardware description languages, and the tools used to accomplish the design and evaluation of the proposed circuits.

2.1.1 Combinational Circuits

Combinational devices are modules that have a set of inputs, a set of outputs, a functional specification that specifies the output values for each combination of input values, and a timing specification (usually a propagation delay t_{pd}) that indicates the time required for the device to generate the outputs from a set of valid inputs. Combinational circuits are built from one or more combinational devices. They are circuits whose outputs are a function of the current input values, that is, they have no memory elements. In other words, the sequence of input values do not change the behavior of the outputs. Furthermore, this type of circuit contains no cyclic path that goes from inputs to outputs. Figure 1 shows a general model of a combinational circuit.

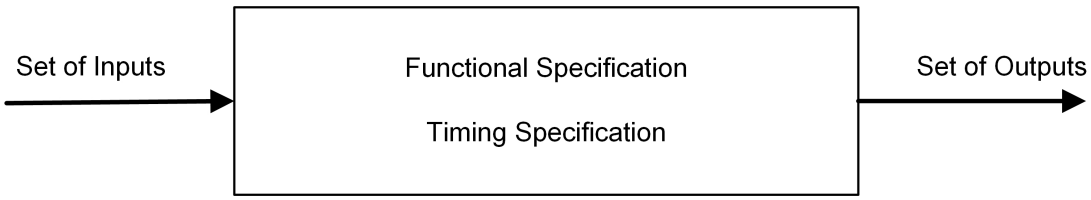

Figure 1: Combinational circuit model

2.1.2 Sequential Circuits

Sequential circuits are circuits whose outputs depend on the current and past inputs, that is, they have some type of memory. Basically, they can be seen as combinational circuits with a storage element to remember past inputs, namely an internal state, and a feedback path to update this state. They also have at least one clock signal to synchronize their operation. Figure 2 shows a general model of a sequential circuit called Moore machine, which outputs depend only on the current internal state. Figure 3 shows a general model of a sequential circuit called Mealy machine, which outputs depend on the current internal state and the current inputs.

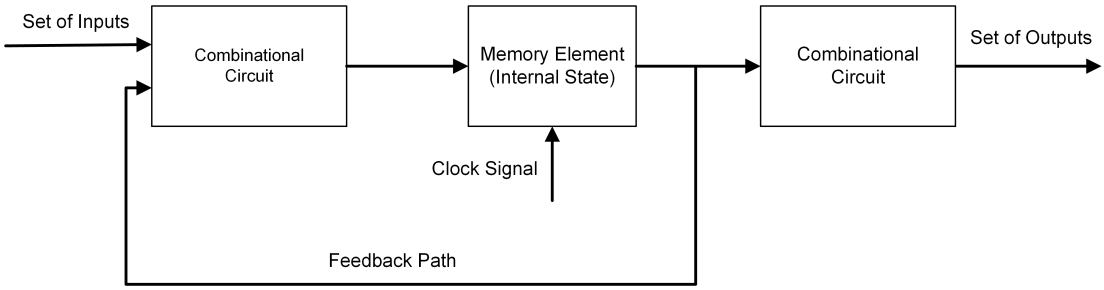

Figure 2: Sequential circuit model (Moore)

2.1.3 FPGA and ASIC

FPGAs belong to the category of programmable logic devices. These devices have several predefined structures in a flexible hardware architecture allowing, according to the designer's decisions, such structures to have their function configured and their

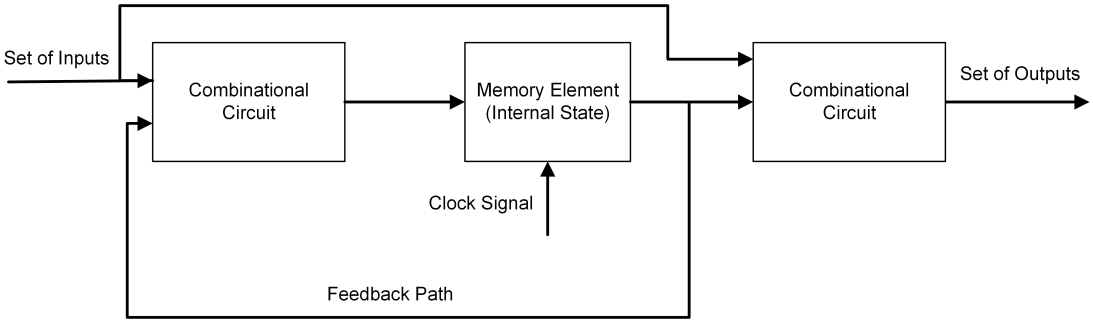

Figure 3: Sequential circuit model (Mealy)

connections defined. This can be done by a configuration file generated by implementing the circuit for a given technology using a synthesis tool. This flexibility allows testing several different architectures on the same FPGA. It also facilitates the comparison between these architectures, and the identification and correction of problems in the initial stages of the project, before a particular architecture be chosen for manufacturing the final circuit (HORTA, 2013). This makes the design cycle simpler and predictable, and it also decreases the time for the final product to reach the market. FPGAs are devices commonly used in the prototyping phase, because they allow the reusability of the same circuit in a fast and direct way, as well as in applications that do not require a very large number of units, since their production cost is high. Modern FPGAs also allow reconfiguring parts of the device, which makes it possible to reprogram them while the other parts continue to operate, a property called partial reconfiguration (XILINX INC., 2016a). On the other hand, they do not allow control over energy optimizations at the level of logic gates. For this reason, they usually consume a greater amount of energy when compared to ASICs. In addition, because a given FPGA family has limited and specific capabilities, this can be a limiting factor to the complexity of the design, and the total cost for large manufacturing volumes can become prohibitive.

ASICs are integrated circuits built for a specific purpose. Their structures and connections are usually defined in terms of logic gates, but they can also be defined

at the transistor level, allowing more granular optimizations than possible in FPGAs. In addition, modern ASICs may present some complete internal blocks such as ROM (*Read-Only Memory*) e RAM (*Random Access Memory*), and even some coprocessors. Once the circuit is defined and manufactured, it can not be modified. This requires the use of techniques and tools to design and to validate the architecture developed before defining the final version. The synthesis for ASIC is performed using a standard cell library from a particular manufacturing process. Depending on the complexity and on the tools used, the initial cost of ASIC design tends to be high compared to FPGA design, the so-called non-recurring costs (circuit design, prototyping, tests, etc.). However, this can be mitigated with future manufacturing costs, which are lower for a large number of units, facilitating applications that require high manufacturing volumes. ASICs also provide a greater degree of freedom to the designer for optimizations in the power consumption and in the area utilization. On the other hand, in addition to the longer time to reach the market, modifications at more advanced stages of the project become more complicated and costly.

The synthesis of a circuit in FPGA or ASIC has its advantages and disadvantages, scope and limitations. If these characteristics are known, the designer will be able to choose the best alternative to meet the objectives of the project. The design flow varies as we want the implementation to be carried out in one or other of these two technologies, although they have some common points. Modernly, designers can use the properties of both, for example using FPGAs for fast prototyping and initial testing. After achieving a certain degree of maturity in the design, they can migrate to ASIC for optimizations of power consumption and area utilization, and for the final fabrication of the circuit. Table 1 and Table 2 show some advantages and disadvantages of designing with FPGAs and ASICs, respectively (XILINX INC., 2015). More details can be found in (ALTERA CORPORATION, 2009; XILINX INC., 2015; KUON; ROSE, 2007).

Table 1: Advantages and disadvantages of designing with FPGA

Advantages	Disadvantages
Shorter time to reach the market	Higher power consumption compared to ASIC
Lower non-recurring cost (circuit design, prototyping, tests)	No control over power optimizations at logic gate level
Simpler and predictable design cycle	Resources limited to those provided by a given technology, thus restricting the complexity of the design
Partial configuration / Reusability	Higher cost per unit

Table 2: Advantages and disadvantages of designing with ASIC

Advantages	Disadvantages
Lower cost per unit for large quantities	Longer time to reach the market
Flexibility for area and power optimizations	Higher difficulty to perform small changes in advanced design steps
Smaller form factor	Higher non-recurring cost

2.1.4 Hardware Description Languages

One of the most important aspects in the design of a digital system is the Hardware Description Language (HDL) used. Two of the main description languages currently used are VHDL and Verilog, which allow describing circuits in terms of signals flow between registers and the operations performed with those signals (*Register-Transfer Level* - RTL). VHDL is a rich, powerful language with strong type checking, and its syntax and constructions are more directly associated with real hardware structures, thus making the language be considered self-documented. The language regards for descriptions with unambiguous semantics, in addition to allowing portability between various hardware design tools, be it for ASIC or FPGA. Verilog is also a rich and powerful language but with poor type checking. It is also more concise, using a simpler and more compact notation. All data types are pre-defined by the language, which has its syntax close to the programming language C. On one hand, based on its structure, VHDL allows identifying most of the errors in the initial stages of the project. On the

other hand, Verilog allows creating descriptions faster and more compactly.

In order to describe the architecture of a digital system, basically two different models are used: structural and behavioral. In the structural model, the architecture is described in terms of interconnected modules, each implementing a particular function. Each module can also be built by the interconnection of simpler modules, generating different levels of abstractions in a hierarchy of descriptions. This makes easier to maintain and reuse hardware modules. On the other hand, the behavioral description is the highest level of abstraction provided by an HDL, where all operations are at only one level of description, and describes the function performed by the module, responsible for mapping its inputs to its outputs. In general, a mixture of the two forms of modeling is used. For example, one can begin with the behavioral description of the most basic and simple modules and, as the level of abstraction is increased, use the structural description to connect such modules in a specific way. It is also possible to start with a behavioral description and, as the level of abstraction decreases, each level above can be seen as a structural description interconnecting simpler modules.

2.1.5 Design Tools

To perform all design steps using the hardware description languages and synthesize the circuits for both FPGAs and ASICs, some design tools are required. Here are some examples.

To perform the synthesis for FPGA, one can use the development tools provided by Xilinx:

* **ISE Design Suite:** it is used for FPGA design, allowing simulation and synthesis using hardware description languages and their implementation in a given technology. In addition, it allows generating synthesis reports, with details about the usage of various building blocks of the FPGA technology, latency, power

consumption, among other metrics. (XILINX INC., 2013).

* **Vivado Design Suite:** successor of the ISE Design Suite. One of its main characteristics is allowing the implementation of FPGA designs in 20nm and 16nm fabrication technology, as well as integrating the tools for working with partial reconfiguration. (XILINX INC., 2016b).

To perform the ASIC synthesis, one can use various design tools provided by Synopsys:

* **VCS:** allows analyzing and simulating circuits described by hardware description languages. It supports multiple levels of description, but it is optimized for transfer-level descriptions between registers (SYNOPSYS INC., 2015c).

* **Design Compiler:** allows synthesizing a description in a circuit at the level of logic gates. It also allows performing optimizations related to power, area, and latency. Furthermore, it provides comprehensive synthesis reports with detailed data on these three metrics and many other details of important interest to the designer (SYNOPSYS INC., 2015a).

* **Formality:** this tool assists in the formal verification between two designs, that is, it verifies the functional equivalence between two implementations that perform the same function, even though both have different levels of description (SYNOPSYS INC., 2015b).

2.1.6 Metrics

After implementing a circuit in ASIC or FPGA, we can compare the results using several metrics (area, latency, frequency of operation, energy, power, throughput, cost). To allow this comparison, it is necessary to define each metric and how it will be measured (KUON; ROSE, 2007; POSCHMANN, 2009; AUMASSON et al., 2014).

2.1.6.1 Cycle Time

This metric corresponds to the critical path delay, that is, the time of the longest path between two memory elements, an input and a memory element, or a memory element and an output. It is the minimum period that can be applied to the circuit. The units commonly used are microsecond [μs] and nanosecond [ns].

2.1.6.2 Latency

The latency is the time required for a particular operation to be performed and its results to be available in the circuit outputs. It can be obtained using the maximum clock frequency or the critical path delay. The tools used to synthesize the circuits have the capabilities to perform such analysis and obtain this information directly. To calculate the latency using the estimated maximum clock frequency, it is necessary to divide by it the average number of clock cycles required to perform a given operation. The units commonly used are millisecond [ms], microsecond [μs], and nanosecond [ns]. The latency can also be measure in clock cycles and the corresponding measure in units of time can be obtained by multiplying this number of clock cycles by the cycle time.

2.1.6.3 Area

This metric measures the total silicon area used by the circuit. For the same design, due to the intrinsic characteristics of both FPGAs and ASICs, the area is obtained and measured in different ways. For ASICs, it is simply the total area obtained after implementation. For FPGAs, it is the sum of the areas occupied by each resource, even if they are not entirely used. In this way, different area measurements can be obtained. For synthesis with ASICs, the area is generally measured in square micrometers [μm^2]. It can also be measured in *gate equivalent* [GE], whose unit represents the area required to implement a two-input NAND gate in the fabrication technology

used. The total area occupied by the circuit can be obtained by dividing the area measured in [μm^2] by the respective two-input NAND gate area, also measured in [μm^2] (POSCHMANN, 2009). For FPGAs, the area is generally measured using the number of [$Slices$], and in modern FPGA families it can also be measured using the number of *Configurable Logic Blocks* [CLB].

2.1.6.4 Power

Power is usually divided into dynamic power and static power. A test bench with test vectors to stimulate the implemented circuit and measure the dynamic power can be used. In the absence of appropriate test benches, it is possible to leave all circuit networks on the same frequency and assume that all are equally likely to be used. This is not a realistic scenario, but it provides a first estimate of the dynamic power consumption of the implemented circuits. Static power measurement is easy to obtain because it depends on the intrinsic construction characteristics of ASICs and FPGAs. A component of the static power is the leakage power, an important factor in circuit technologies with short gate length. The comparison between the power consumption for a circuit implemented in these two technologies should consider the low operating frequencies of the FPGAs. This means that more time or greater degree of parallelism is required to perform the same amount of work as an ASIC. The design tools can provide estimated power results. Power is usually measured in microwatts [mW], and its multiples and submultiples can also be used.

2.1.6.5 Energy

Energy is obtained directly from the power consumption in a given period of time. Many applications have energy restrictions, and knowing the energy curve for the implemented circuit is of extreme importance, since the power curve is not always sufficiently behaved (presence of peak points) to draw direct conclusions about the energy

consumption. Energy is usually measured in $[mJ]$, and its multiples and submultiples can also be used.

2.1.6.6 Maximum Frequency

Parameter applied only to sequential circuits. The maximum operating frequency can be obtained as the inverse of the critical path delay. The units commonly used are $[MHz]$ and $[GHz]$.

2.1.6.7 Throughput

It defines the rate at which new output bits are generated in relation to the unit of time. It is calculated by multiplying the number of output bits by the circuit operation frequency, and dividing the result by the number of clock cycles required to generate these output bits. For combinational circuits, the throughput is the inverse of the circuit latency multiplied by the number of output bits. For communication and cryptographic applications, the common units are $[Mbps]$ and $[Gbps]$, but many other applications use $[operations/s]$ or $[instructions/s]$.

2.1.6.8 Throughput-to-area

This ratio can be used as a measure of efficiency. For circuits where throughput is a crucial parameter, it is important to measure how much throughput such a circuit achieves in relation to its silicon area utilization, since many applications have limitations on physical size. In addition, the area utilization contributes to the total cost of the final project. It is obtained by dividing the throughput by the circuit area and is usually measured in $[kbps/GE]$ for ASIC and $[Mbps/CLB]$ for FPGA.

2.1.6.9 Energy-per-bit

This ratio can be used as a measure of efficiency, indicating how much energy is spent to generate 1 bit of output. Many applications have restrictions on energy consumption and this contributes to the total cost of the final project. Its calculation is done by dividing the total power consumption by the throughput obtained and is measured in $[mJ/Gb]$.

2.2 Cryptographic Aspects

This section presents some important concepts about cryptographic hash and sponge functions, as well as password hashing schemes.

2.2.1 Cryptographic Hash Functions

Hash functions are a class of functions that transform an arbitrary-length string of bits into a fixed-length string of bits. The functions of this class that also satisfy the one-way or non-invertible condition (*One-Way Hash Functions* - OWHF) and the resistance collision condition (*Collision-Resistant Hash Functions* - CRHF) can be used in cryptographic applications (SOBTI; GEETHA, 2012). The cryptographic hash functions are used to achieve several security goals, such as authentication of messages and entities, integrity, digital signatures, generation of pseudorandom numbers, among others. Formal definitions for cryptographic hash functions and their requirements can be found in (MERKLE, 1979; ROMPAY, 2004).

2.2.1.1 Blake2

The Blake2 family of functions is the successor of Blake family, designed and developed after the SHA-3 (*Secure Hash Algorithm 3*) contest organized by NIST (*National Institute of Standards and Technology*). It was designed to maintain the high

efficiency and safety of its predecessor and to optimize it for new applications, with the main goals being simplicity and usability (AUMASSON et al., 2014).

The Blake2 family consists of two main algorithms: Blake2b, which is optimized for 64-bit platforms and can generate digests ranging from 1 to 64 bytes, and Blake2s, which is optimized for platforms up to 32 bits and can generate digests ranging from 1 to 32 bytes. Cryptanalysis results can be found in (GUO et al., 2014). Further details on Blake and Blake2 are found in (AUMASSON et al., 2014).

2.2.2 Cryptographic Sponge Functions

Sponge functions provide a mechanism for generalizing traditional hash functions to more general functions with arbitrary-length output. Sponge functions can operate in two distinct modes: sponge or duplex. These two modes allow implementing a wide range of symmetric cryptographic functions.

2.2.2.1 Sponge Mode

Figure 4 shows the structure of the sponge mode. Input message M is first extended by a given padding rule and then divided into blocks of r bits. Permutation function f operates with fixed-length inputs and outputs of b bits. The state of the sponge construction also has the size of b bits, where c are the bits considered the inner part of the state and r represents the bits of the external part. Initially, all bits are initialized with value zero. Parameter c is called capacity and determines the security level of the construction (BERTONI et al., 2011). In the *absorbing* phase, all blocks of r bits obtained previously from M are XORed with the r bits of the external part of the state, alternating with the execution of transformation f. In the *squeezing* phase, the bits in the external part of the state are iteratively placed in output blocks. Each output block is generated by the execution of transformation f in the previous output block. The resulting output has l bits. Thus, the number of iterations of the *squeezing* phase must

be such that it allows the generation of a *l*-bit output.

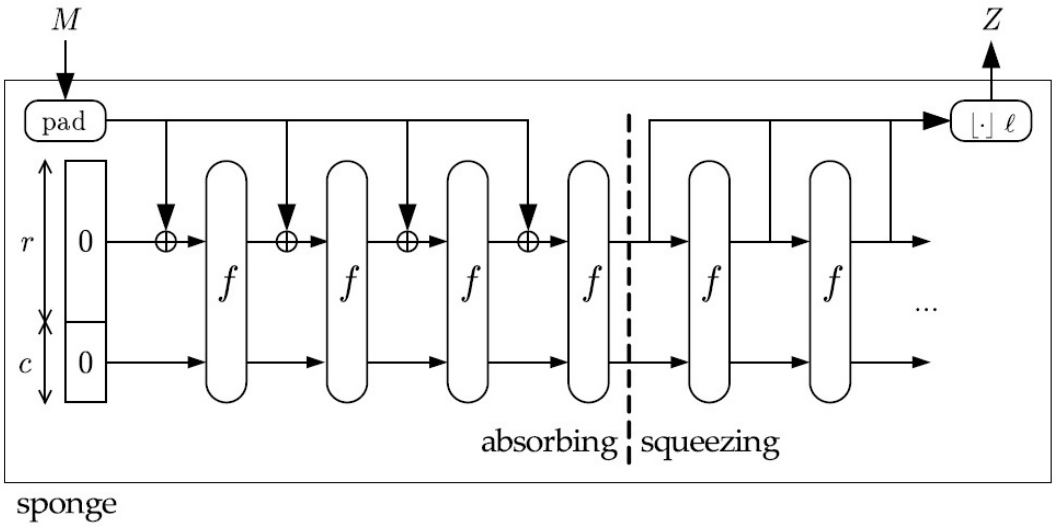

Figure 4: Sponge mode (BERTONI et al., 2011)

2.2.2.2 Duplex Mode

Figure 5 shows the duplex mode structure. In this mode, the input message σ_0 is extended into a single block of r bits. The block of r bits obtained previously from σ_0 are XORed with the bits from the external part of the state and then undergoes transformation f. In this step, unlike the sponge construction, the next call of the duplex construction will have the output of the previous call as the initial state. In this way, the output at any given time will depend on all the inputs previously used, generating a memory construction between the calls.

2.2.3 Password Hashing Schemes

A password hashing scheme is an algorithm that uses a pseudorandom function (NIST, 2011) having as its main input (although it usually has some others) a password and generating as output a pseudorandom sequence of bits (NIST, 2009). To accomplish this operation, this type of algorithm usually uses a hash function as one of its structures. This hash function must be such that it becomes computationally imprac-

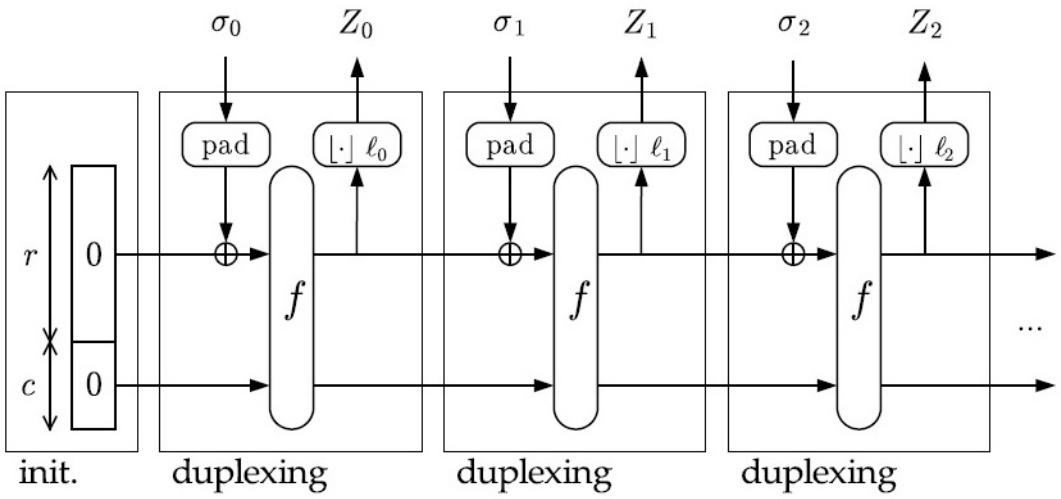

Figure 5: Duplex mode (BERTONI et al., 2011)

ticable to obtain from the sequence of output bits the password that generated it. The output of a password hashing scheme can be used in two main ways:

∗ **Cryptographic keys generation:** from the password, it obtains a key of appropriate size to be used as input in some algorithm of cyphering and/or authentication. This process is known as *key stretching* (NIST, 2009; SIMPLICIO et al., 2015), and its main purpose is to obtain a key of adequate size from the input password that makes unfeasible brute force attacks against the original password, which has less entropy than the resulting key (KELSEY et al., 1998; PERCIVAL, 2009).

∗ **Password storage:** as a bit string to be stored for later checks of user password, for example when accessing a computer network or performing a banking operation over the Internet. Legitimate users will need to perform the password hashing only once, while attackers on average will need to perform it many times during a brute force attack. Ideal password hashing schemes for this purpose allow configurable processing and memory costs to perform their operation. This can hinder attacks such as the brute force attack.

The passwords defined by users usually have an average entropy value below the value required in various protocols and cryptographic schemes used in modern systems (FLORENCIO; HERLEY, 2007). Thus, dictionary and exhaustive search attacks allow attackers to overcome the non-invertibility of hash functions, and such a task can be made easier if attackers have multiple processing cores at their disposal (for example, modern GPUs (*Graphics Processing Unit*) and customizable hardware synthesized in FPGA or ASIC), thus enabling many parallel tests (SIMPLICIO et al., 2015). Modern password hashing schemes often provide the ability to adjust some parameters to control processing and/or memory costs. In addition to the already mentioned Lyra2 and Argon2, other main password hashing algorithms present in the literature are PBKDK2 (*Password-Based Key Derivation Function 2*) (KALISKI, 2000), scrypt (PERCIVAL, 2009), and bcrypt (PROVOS; MAZIÈRES, 1999).

3 RELATED WORK

This chapter contains a survey of works from the literature that are of interest to this research. The works analyzed present several approaches to deal with area, power, latency, and throughput metrics, as well as providing proposals and techniques to analyze some trade-offs between them. The survey are divided in two parts: works that present hardware designs for cryptographic hash and sponge functions, and works that present hardware designs for other types of algorithms. At the end of the chapter, a brief discussion is presented.

3.1 Hardware Designs for Cryptographic Hash and Sponge Functions

The work by (SUNNY; SARANYA, 2014) presents some implementations of the Blake-256 algorithm, with a first architecture obtained directly from the specification of the algorithm, a second architecture oriented towards high speed, and a third and more compact architecture for a lesser area utilization. Synthesis results for ASIC are presented for the three architectures in terms of latency, area, and memory usage. The authors do not make comparisons with other works of the literature or evaluate trade-offs between the metrics involved. The work only aims to obtain the maximum possible optimization for one of them in each type of architecture proposed. The third and more compact architecture presents the best results in terms of area, memory usage, and latency. The first architecture obtained directly from the algorithm specification

presents the worst results for the three metrics. The second architecture for higher speed presents intermediate values for the three metrics. Despite the results, an analysis of the power or energy consumption is not found. In (AUMASSON et al., 2014), the authors of the Blake hash function present their reference hardware implementations, exploiting parallelism for lower latency over an increase in area utilization. They suggest some alternatives to minimize the area and to balance the trade-off with the latency of the circuit. Complete synthesis results for ASIC are presented for various degrees of parallelism and compared in terms of area utilization, maximum frequency of operation, latency, throughput, and efficiency (defined as the ratio of throughput to area). Other similar proposals synthesized in FPGA can be found in (LATIF et al., 2011; KUMAR; CHITRAVALAVAN, 2014).

The work by (RAO; NEWE; GROUT, 2014) presents an implementation of the SHA-3 algorithm in FPGA, which focuses mainly on the analysis of the results for power consumption. It also presents synthesis results for latency and throughput in various FPGA families. Despite the several synthesis results, no trade-off analysis between latency, power, and area is presented. Similar work can be found in (YALLA; HOMSIRIKAMOL; KAPS, 2014). The work by (PEREIRA et al., 2013) presents the use of pipeline and parallelism in the development of architectures for the Keccak sponge function. The proposed architectures are synthesized in FPGAs, and the results are presented and compared with other results in the literature. Compared with a reference implementation without pipeline, when using this technique, despite the greater area utilization (+15.3%), a higher operating frequency (+370%) and a higher throughput (+32%) are obtained. The same implementation using pipeline is compared with other implementations in GPU, comparing the resulting throughput and concluding that it is about 7 times superior to the best result of the literature presented in their work. Despite these analyses and comparisons, no ideias are presented to treat trade-offs. The work by (PROVELENGIOS et al., 2012) presents implementations of the same func-

tion, showing some techniques to increase the hardware speed using specific blocks present in FPGAs. They also present specific instructions and features offered by the manufacturer of the technology, obtaining better results in terms of area utilization when such techniques and features are exploited. The characteristic of modern FPGAs to allow manual work with LUT (*Look-Up Table*) primitives is exploited in (JUNGK; STOTTINGER; HARTER, 2014). The authors compare implementations with and without this feature, and explore trade-offs between throughput and area utilization. Although the throughput results are slightly lower than the related work presented by them, the reduction in area utilization is substantial. A similar work can be found in (SAN; AT, 2012). The creators of Keccak present their reference hardware implementation for the function in (BERTONI et al., 2012). An architecture for high performance is presented, as well as another for small area. A third architecture aiming to deal with the trade-off between throughput and area is proposed. When this architecture is compared to the high performance architecture, the authors obtain a decrease of about two times in the area with a reduction of four times in the throughput. Despite the interesting results, no power/energy analysis is performed.

The work by (ZHANG et al., 2014) presents a hash function based on the sponge construction called SHAT (*Sponge Hash Algorithm*). Initially, the authors present a direct implementation of the algorithm in hardware, comparing the throughput and area results with those for implementations of other hash functions present in the literature. Then, they apply various optimization techniques in order to obtain better performance for a specific metric or treat the trade-off between two of them. As the first optimization, they use the *unfolding* technique to improve the throughput of the circuit, obtaining a value 47.97 times greater for this metric as their best result. As a second optimization, they use the parallelism and pipeline techniques together to reduce the critical path delay by modifying the permutation function structure, thus obtaining a 6.31% reduction in circuit latency. As a third optimization, a technique is proposed to

calculate an operating frequency range that can be used to perform a trade-off between low dynamic power consumption and high throughput. For various frequency ranges, synthesis results for power consumption, area, and latency are presented. In spite of presenting the analysis of the measurements for these three metrics as a function of the variation of the operating frequency, an analysis of how it would be possible to obtain a balanced performance between these metrics is not performed. However, it constitutes an interesting example of some ideas to perform a three-dimensional analysis. As the last optimization, they use the *clock gating* technique to obtain a reduction in dynamic power consumption. With this approach, a reduction of 13.65% is obtained.

The work by (TEHRANIPOOR; WANG, 2011) presents some techniques to optimize the hardware execution speed of the MD5 (*Message-Digest algorithm 5*) hash function, such as pipeline and the use of *carry-save* and *carry-ripple* adders. When synthesis is performed in FPGA, blocks of RAM memory can be used to store constants used by the function, that is, more CLBs become available for the main flow of the algorithm, resulting in higher performance. The authors do not present any specific implementation of this algorithm, only a comparison of implementations present in the literature using pipeline with different stages. They also analyze parallel and iterative architectures, comparing the results in terms of latency and throughput. A similar approach is used for SHA-2 hash function, but the synthesis is also done in ASIC. No results are presented for area utilization or power consumption. Thus, no trade-off between them and latency or throughput is treated. In addition, the authors present a compilation of synthesis results for several SHA-3 candidates present in the literature, such as Blake, Keccak, and Skein. Latency, area, throughput, and maximum operating frequency of the implementations are compared for both FPGA and ASIC, but without analyzing trade-offs between these metrics. Similar works can be found in (LATIF et al., 2012; TILLICH et al., 2009). For comparisons involving other metrics, we have the work by (GUO et al., 2011), where synthesis results are presented and compared for the number

of cycles, area, maximum frequency, throughput, power, and energy consumption, thus providing basis on how to define and to measure each of these metrics. However, they do not analyze trade-offs among area, throughput, and power.

The work by (MIKAMI et al., 2010) presents a compact implementation for the Luffa hash function, which was one of the candidates in the SHA-3 competition. The authors present three distinct architectures for synthesis, exemplifying some optimization techniques and analyzing the trade-off between area and throughput. The latter can be obtained by the ratio of the operating frequency to the number of clock cycles required for each round of the function, and results for each of these metrics are obtained for each architecture. The synthesis is performed for ASIC and graphs representing the relation between the throughput of the circuit and its area are presented, showing how the throughput behaves when the area utilization increases. The conclusion is that, although the throughput increases with the increase in area, this growth is not uniform. Results and analyses for power consumption are not provided. A compact work that presents some strategies for developing architectures with trade-offs between area and throughput. For the Skein hash function, which was among the SHA-3 competition finalists, there is the work done by (WALKER et al., 2010). Other similar works are found for the hash functions Shabal (DETREY; GAUDRY; KHALFALLAH, 2011) and Whirlpool (MCLOONE; MCIVOR, 2007).

In the work by (TILLICH et al., 2010), the authors present high-speed architectures for 14 candidates of the second round of the SHA-3 competition. Given the number of different algorithms, the authors present their method to make implementations structurally similar to each other, except for the intrinsic differences between the algorithms. With this approach, it is possible to compare the synthesis results in a consistent and more realistic way. The design of all modules was planned to address area-latency trade-off to maximize circuit speed. All the results are compared to each other and to a reference hardware implementation for the SHA-2 algorithm as well. No power ana-

lysis is performed. The work by (GUO et al., 2010) is complementary, also presenting synthesis results for FPGA. The authors elaborate a method of comparison to decrease the impacts of the peculiarities of each platform. To evaluate the implementations, they propose the use of the maximum throughput and the throughput-to-area. They also use the area and power obtained for a specific value of throughput. The authors also analyze the trade-off between power and area, and the trade-off between area and maximum throughput. However, there is no analysis of any scenario that presents a three-dimensional trade-off among area, throughput, and power. The results for all the algorithms are presented and compared in terms of the proposed metrics.

3.2 Hardware Designs for Other Types of Algorithms

In (ALPHA TECHNOLOGY, 2013), an architecture is presented to execute the *scrypt* password hashing scheme. No special optimization in terms of area and energy is performed, but complete synthesis results for ASIC and FPGA are presented. The work by (WIEMER; ZIMMERMANN, 2014) presents a flexible and high-speed implementation for the *bcrypt* password hashing scheme, targeting password-search attacks. Due to the nature of the application, parallelism techniques are presented and applied, constituting a great example of application of this technique. Results for area, energy, latency, and cost are presented and compared with other results in the literature. In (MALVONI; KNEZOVIC, 2014), another parallel hardware architecture for the same PHS is presented. This architecture is optimized in terms of power consumption and production cost, complementing the work by (WIEMER; ZIMMERMANN, 2014).

The work by (BEDNARA et al., 2002) performs an area-latency trade-off analysis for hardware implementations of algorithms used in elliptic curve cryptography. Additionally, it provides an architecture for a cryptographic coprocessor that can be configured with variable parameters for area and latency, as well as the size of the underlying finite field. This design represents the idea for obtaining an architecture or group of

architectures that can be configurable in relation to some performance metrics. The authors also show how to apply high-level techniques for designing the control unit. The proposed FPGA architecture is detailed and can be adapted to achieve the mentioned trade-off by using some degrees of freedom such as the number of functional units in the datapath, degree of parallelism in the units of multiplication, and the implementation strategy for the control unit. Synthesis results comparing various coprocessor configurations are presented. The analysis performed is restricted to the trade-off between area and latency, and there is no synthesis results for power consumption or an analysis of any three-dimensional trade-off. Similar work is found in (WENGER; FELDHOFER; FELBER, 2011).

A guide for low power design of embedded systems is provided in (IVEY, 2011). Despite the specificity of the guide, the presentation of some ideas for this type of design and detailed definitions of static and dynamic power are of great value. Descriptions of techniques and approaches for low power design at the architecture and technology level are found in (HAVINGA; SMIT, 2000). Based on some of these ideas, (HÄMÄLÄINEN, 2006) presents strategies and techniques to obtain an implementation of the AES (*Advanced Encryption Standard*) algorithm with small area utilization, exploring the trade-off between this metric and the power consumption. The implementation is performed in ASIC and compared with other results in the literature. The authors indicate that the power consumption to process each input block of the algorithm is smaller, despite presenting a larger throughput and area similar to other works cited by them. Similar work can be found in (KHOSE; RAUT, 2014).

The work by (SUTTER; DESCHAMPS; BOEMO, 2004) presents implementations of three algorithms for modular multiplication (*Multiply and Reduce*, *Shift and Add*, *Montgomery Multiplication*). Each implementation is described in both combinational and sequential form and synthesized in FPGA. For the combinational form, the authors compare the synthesis results in terms of area utilization (number of CLBs) and

maximum latency. For the sequential form, the results are compared in terms of area (number of CLBs and Flip-Flops) and maximum operating frequency. In this step, no optimizations or trade-off analysis between the metrics is performed. Implementations are performed using the description obtained directly from the algorithm in question. Next, the authors measure the power consumption of the circuit, presenting an overview of several components of the total power: dynamic, static, *off-chip*, and synchronization. The comparison of the three algorithms in their combinational form is made in terms of area, energy, and latency. Similar measurements are performed for the algorithms in their sequential form. The work is direct in its goals and analyzes the area utilization, the power consumption, and the latency of three different algorithms for the same application. On the other hand, it does not present any optimization for some of these metrics or analyze any trade-off between them, but it constitutes an example of implementing an algorithm in the combinational and sequential approaches. The work by (ZHOU; HUANG; SMITH, 2011) presents efficient and high-performance area utilization proposals for Montogomery multipliers. Architectures derived directly from the algorithm together with an optimized architecture and other architectures from the literature are compared in terms of FPGA synthesis results, specifically area and latency. The optimized architecture provided by the authors shows an increase in the operating frequency and in the total throughput, in addition to a reduction of 60% in area resources when compared with other works in the literature.

The authors in (SARAVANAKUMAR; RANGARAJAN; RAJASEKAR, 2012) present an implementation of a CS (*Circuit-Switched*) router within the concept of network in a single encapsulation (*Network-On-Chip - NOC*). Two architectures are presented and compared, with and without the use of pipeline. Synthesis results are obtained for ASIC with the help of Synopsys tools, and the metrics of area, power, and latency are compared. Dynamic and static power, critical path delay, and combinational and non-combinational area are obtained directly from the synthesis tools. The

pipelined architecture presents a lower power consumption and a lower utilization of non-combinational area (about half of that used by the other architecture). The non-pipelined architecture presents a lower critical path delay and a lower combinational area utilization, but the difference is less than 4%. This work constitutes an interesting reference to the method to compare architectures with and without optimizations. A complementary work is (AHMED; ABDALLAH, 2013).

The work by (DENG et al., 2011) proposes models to estimate area, power, and latency for circuits synthesized in FPGA, specifically those of Xilinx. The purpose of these models is, employing a design method, to efficiently use available area resources. Other models proposed to estimate the power consumption in different parts of the circuit are derived from previous models. Furthermore, models for estimating latency are obtained in a similar way. Some circuits are proposed and used as an example to validate and to compare the models. Hence, the work described can provide ideas and tools for an optimized project for FPGAs, enabling them to use their natural characteristics more efficiently. It can also provide confidence when comparing the results with the synthesis results obtained for ASIC. Other considerations about trade-offs between performance metrics can be found in (JEAN-JACQUES et al., 2002; MARKOVIC; NIKOLIC; BRODERSEN, 2006).

3.3 Discussion

The three main metrics of interest for this research are the area utilization, the power consumption, and the throughput. Some works discussed earlier address optimized hardware design for improving one of these metrics. In other works, optimizations are made for two of them, but without analyzing how a third metric relates to the other ones. The values obtained for this third metric are assumed to be consequences of the previous optimizations in the other metrics. Other works analyze the trade-off between two of these metrics, such as area and throughput, or power and area, or among

other associated metrics such as latency and maximum operating frequency. Furthermore, the lack of results for power consumption can be verified in most of the works, despite being a crucial parameter in mobile and embedded devices. Only in one of the works were optimizations found for the three metrics (ZHANG et al., 2014). However, no three-dimensional trade-off analysis between them was found. Nevertheless, it constitutes a great example of some optimization ideias that can be used as a basis for this type of analysis. The work by (SUNNY; SARANYA, 2014), despite not presenting trade-offs analyses, presents a sequence of implementations for an algorithm, starting with an implementation obtained directly from the algorithm specification and then presenting optimizations for high speed and less area utilization, with one architecture for each optimized metric. It can be also noted that many works present synthesis results only for ASIC or only for FPGA. Thus, we intend to contribute to bridging these gaps, not just presenting complete synthesis results for both ASIC and FPGA, but also proposing ideas and architectures that allow analyzing a trade-off among three metrics (area, power, throughput). In addition, no hardware implementation was found for the BlaMka algorithm. Then, providing this implementation will be of great value not only to help us address the analysis of the trade-offs commented above, but also to present the first hardware design and performance analysis on this platform for this algorithm.

4 DESIGN AND IMPLEMENTATION

In Chapter 2, some basic topics about hardware design and cryptography were presented. In this chapter, the permutation function of the BlaMka cryptographic sponge and its hardware design and implementation are described.

4.1 Strategy

In order to carry out the hardware design and implementation of the chosen algorithm and to meet the goals proposed for this work, we will follow a well-defined design strategy, shown in Figure 6. Below, a brief description for each of the steps in this strategy is presented. Although the strategy is described as a function of the proposed algorithm, it could also be used for other types of algorithms.

1. After choosing the algorithm to be implemented as a case study, it is necessary to understand its operation. In this step, the algorithm is analyzed and its main features and characteristics are presented.

2. The next step consists of a basic design derived directly from the algorithm structure, simply mapping the operations and variables used by the algorithm into simple hardware modules such as registers, adders, multiplexers, among others.

3. Variations are proposed in the basic design to obtain optimizations, initially improving the throughput, which is one of the metrics of greater interest for the

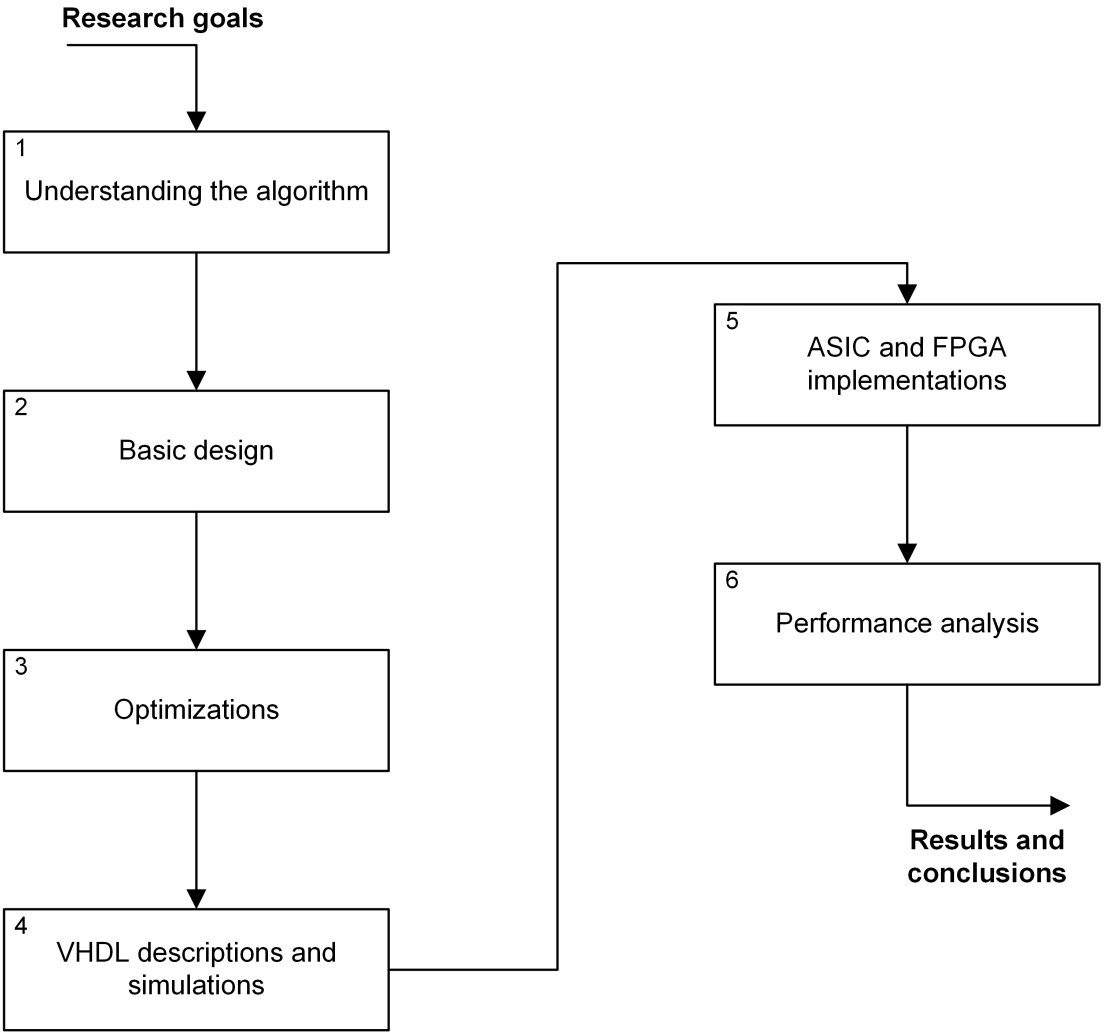

Figure 6: Design and implementation strategy

BlaMka algorithm, and later performing improvements in terms of area utilization and power consumption.

4. After the design of several architectures from the previous steps, they are described in VHDL using the design tools presented in Chapter 2. Functional simulations are performed to verify the correct operation of the architectures.

5. In this step, the various descriptions are implemented in both ASIC and FPGA, obtaining results for area, power, throughput, latency, among several other metrics.

6. Finally, from the experimental results obtained from the previous step, a detailed performance analysis is conducted, starting from a simple one-dimensional analysis until culminating in the three-dimensional analysis discussed in Chapter 1.

For designing and describing sequential circuits, a partitioning model will be adopted dividing the complete system into a control unit and a datapath unit, as shown in Figure 7. For combinational circuits, the complete system is reduced to the datapath unit; because they do not have memory elements, no control signals are necessary.

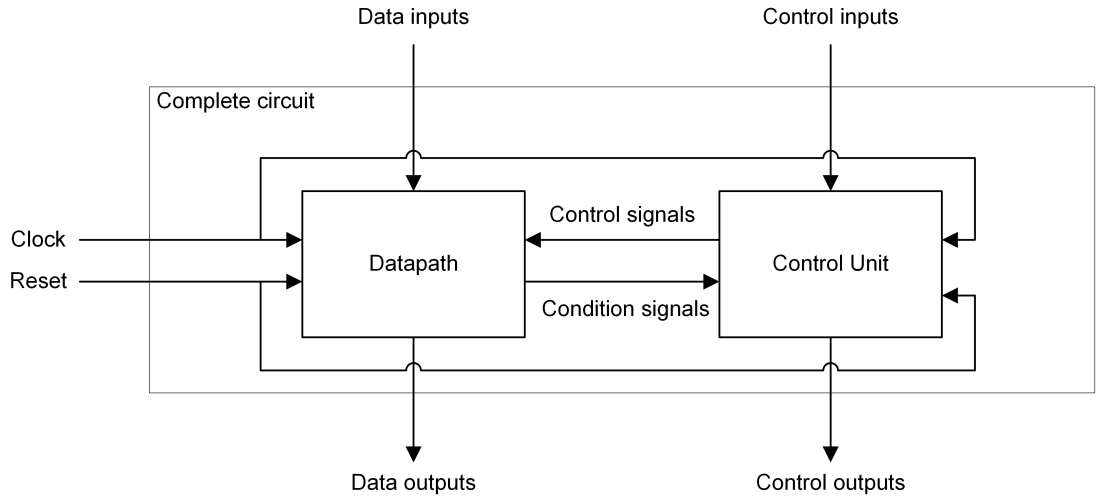

Figure 7: Partitioning model for sequential circuits

The datapath module is responsible for receiving the data inputs of the system, for storing and transforming these data, as well as for generating and presenting the output results after executing the algorithm it implements. The control unit is responsible for generating the signals that will control the execution of the datapath module, and for generating the external control outputs for the system. To perform its function, it is aided by condition signals received from the datapath module and by external control inputs. Here, the control unit will be designed only once and it will be reused with minor adjustments in all the proposed circuits. Optimizations will be suggested and

implemented only in the datapath module.

4.2 BlaMka Permutation Function

Figure 8a presents the permutation function of BlaMka, denoted by G. The G function has four 64-bit inputs, named a, b, c, d, which will be updated and presented as outputs after its execution. The basic structure of G consists of two updates for each input variable, and the values used for the calculations will be those that have already been updated in the previous steps of the algorithm. For example, the value of variable a used in step 2 corresponds to the result of step 1, and the value of variable d used in step 3 corresponds to the result of step 2. The same reasoning applies to the other steps, and the Figure 8b shows this strictly sequential nature of the G function. The $\oplus$ symbol denotes bitwise Exclusive-OR (XOR) logical operation, while + denotes wordwise addition, and $\cdot$ denotes wordwise multiplication. Assuming a little-endian convention, $\mathrm{lsw}(x)$ is the least significant word (32 bits) of x. The symbol $\ll$ denotes a left shift operation and the symbol $\ggg$ denotes a right rotation operation. The permutation function of Blake2b has the same structure, except for the absence of the multiplicative components $2 \cdot \mathrm{lsw}(a) \cdot \mathrm{lsw}(b)$ and $2 \cdot \mathrm{lsw}(c) \cdot \mathrm{lsw}(d)$, as can be seen in Figure 9.

1. $a \leftarrow a + b + 2 \cdot \mathrm{lsw}(a) \cdot \mathrm{lsw}(b)$
2. $d \leftarrow (d \oplus a) \ggg 32$
3. $c \leftarrow c + d + 2 \cdot \mathrm{lsw}(c) \cdot \mathrm{lsw}(d)$
4. $b \leftarrow (b \oplus c) \ggg 24$
5. $a \leftarrow a + b + 2 \cdot \mathrm{lsw}(a) \cdot \mathrm{lsw}(b)$
6. $d \leftarrow (d \oplus a) \ggg 16$
7. $c \leftarrow c + d + 2 \cdot \mathrm{lsw}(c) \cdot \mathrm{lsw}(d)$
8. $b \leftarrow (b \oplus c) \ggg 63$

(a) BlaMka G function

1. $\boldsymbol{a} \leftarrow a + b + 2 \cdot \mathrm{lsw}(a) \cdot \mathrm{lsw}(b)$
2. $\boldsymbol{d} \leftarrow (d \oplus \boldsymbol{a}) \ggg 32$
3. $\boldsymbol{c} \leftarrow c + \boldsymbol{d} + 2 \cdot \mathrm{lsw}(c) \cdot \mathrm{lsw}(\boldsymbol{d})$
4. $\boldsymbol{b} \leftarrow (b \oplus \boldsymbol{c}) \ggg 24$
5. $\boldsymbol{a} \leftarrow a + \boldsymbol{b} + 2 \cdot \mathrm{lsw}(a) \cdot \mathrm{lsw}(\boldsymbol{b})$
6. $\boldsymbol{d} \leftarrow (d \oplus \boldsymbol{a}) \ggg 16$
7. $\boldsymbol{c} \leftarrow c + \boldsymbol{d} + 2 \cdot \mathrm{lsw}(c) \cdot \mathrm{lsw}(\boldsymbol{d})$
8. $b \leftarrow (b \oplus \boldsymbol{c}) \ggg 63$

(b) BlaMka G function sequential nature

Figure 8: The BlaMka permutation function denoted as G. Adapted from (SIMPLICIO et al., 2015)

Each variable update operates with values of two other variables. The algorithm

1. $a \leftarrow a + b$
2. $d \leftarrow (d \oplus a) \ggg 32$
3. $c \leftarrow c + d$
4. $b \leftarrow (b \oplus c) \ggg 24$
5. $a \leftarrow a + b$
6. $d \leftarrow (d \oplus a) \ggg 16$
7. $c \leftarrow c + d$
8. $b \leftarrow (b \oplus c) \ggg 63$

(a) Blake2b *G* function

1. $a \leftarrow a + b\mathbf{+2 \cdot lsw(a) \cdot lsw(b)}$
2. $d \leftarrow (d \oplus a) \ggg 32$
3. $c \leftarrow c + d\mathbf{+2 \cdot lsw(c) \cdot lsw(d)}$
4. $b \leftarrow (b \oplus c) \ggg 24$
5. $a \leftarrow a + b\mathbf{+2 \cdot lsw(a) \cdot lsw(b)}$
6. $d \leftarrow (d \oplus a) \ggg 16$
7. $c \leftarrow c + d\mathbf{+2 \cdot lsw(c) \cdot lsw(d)}$
8. $b \leftarrow (b \oplus c) \ggg 63$

(b) BlaMka *G* function

Figure 9: Comparison between Blake2b and BlaMka G functions. Adapted from (SIMPLICIO et al., 2015)

has two different sequences of operations, which are repeated alternately during its execution. The first sequence (odd steps) consists of addition, multiplication, and shift operations. The second sequence (even steps) consists of XOR and rotation operations. The number of bits that must be rotated varies depending on which stage of the G function the execution is, but only four different values are used: 16, 24, 32 and 63 bits. Each pair of inputs of the G function goes through this basic structure twice. In the first sequence, we have a multiplicative component, and the detail for working with the least significant word was a choice in terms of performance by the creators of the algorithm (SIMPLICIO et al., 2015). Of course, multiplication between 64-bit variables would be more costly than using only 32-bit operands.

4.3 Design and Implementation

4.3.1 Basic Design

The four variables used by the G function have a size of 64 bits each, constituting 256-bit input data and 256-bit output data. The first two lines of the algorithm have the same structure as the other pairs of lines, except for the variables used in the calculations, and the number of bits to be rotated in the updates of variables b and d.

Sequential and combinational circuits can be constructed to implement the G function. For a sequential circuit, and as the basic design, we can assign a register to store each variable, and directly implement the structure of the first two lines of the algorithm, reusing their functional units to perform the other pairs of lines. Other approaches could be used, such as a circuit that implements four steps of the algorithm, in which case it would be necessary to duplicate the number of functional units as adders and multipliers. For simplicity and also for a lesser area utilization (mainly by reducing the number of multipliers required), the first approach was preferred for the sequential circuits. However, a circuit with four multipliers that can execute the whole algorithm at once is simply a combinational circuit, and will be described in the next section. For the sequencial circuit, in terms of storage, only four registers are used to store the variables, as well as other combinational components required to perform the desired operations, such as multipliers, adders, and modules for 64-bit XOR operation. Except for the two input signals of the multiplier, which have 32 bits each, all other signals used in the circuit are 64-bit long.

To update variable d, we need to wait for the result of variable a. Assuming we have separate units for addition and multiplication, one clock cycle will be necessary to perform the update of variable a, since $a + b$ and $2 \cdot \mathrm{lsw}(a) \cdot \mathrm{lsw}(b)$ can be calculated in parallel. In this same cycle, it is possible to update variable d, since shifts and rotations constitute only changes in wires connections, just adding the 64-bit XOR module latency to this cycle time. Similar comments are valid for variables c and b.

As previously mentioned, the Blake2b algorithm does not have multiplications, which is the main difference compared to BlaMka in terms of the types of components. However, all other considerations explained above are valid for both algorithms. Using multiplexers and mapping operations of the algorithm into functional units, the resulting basic sequential circuit is shown in Figure 10 for Blake2b and in Figure 11 for BlaMka. The shaded region in Figure 11 indicates the part of the circuit corresponding

to the differences between the permutation functions of both algorithms. The circuit presented for Blake2b will be used as a reference for the analysis of the experimental results for BlaMka.

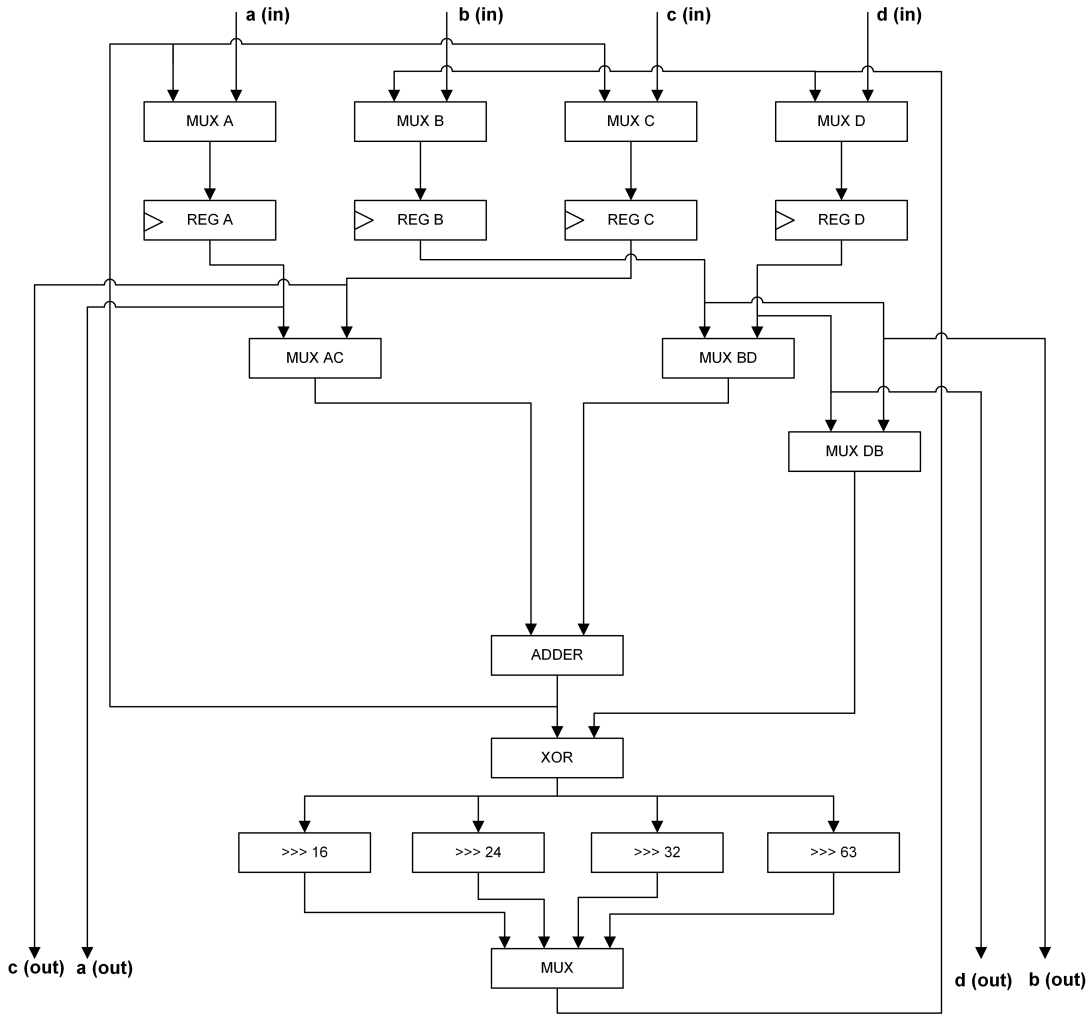

Figure 10: Sequential circuit for Blake2b *G* function

The multiplication by 2 indicated by << 1 can be accomplished by a simple left shit of 1 bit. This shift operation, as well as the rotation to the right of x bits, consists basically in changes of wires, that is, these operations do not require using specific combinational and sequential components, because the number of bits to be shifted and the number of bits to be rotated are known. Of course, if these pieces of information were not previously known, specific modules for such operations should be used.

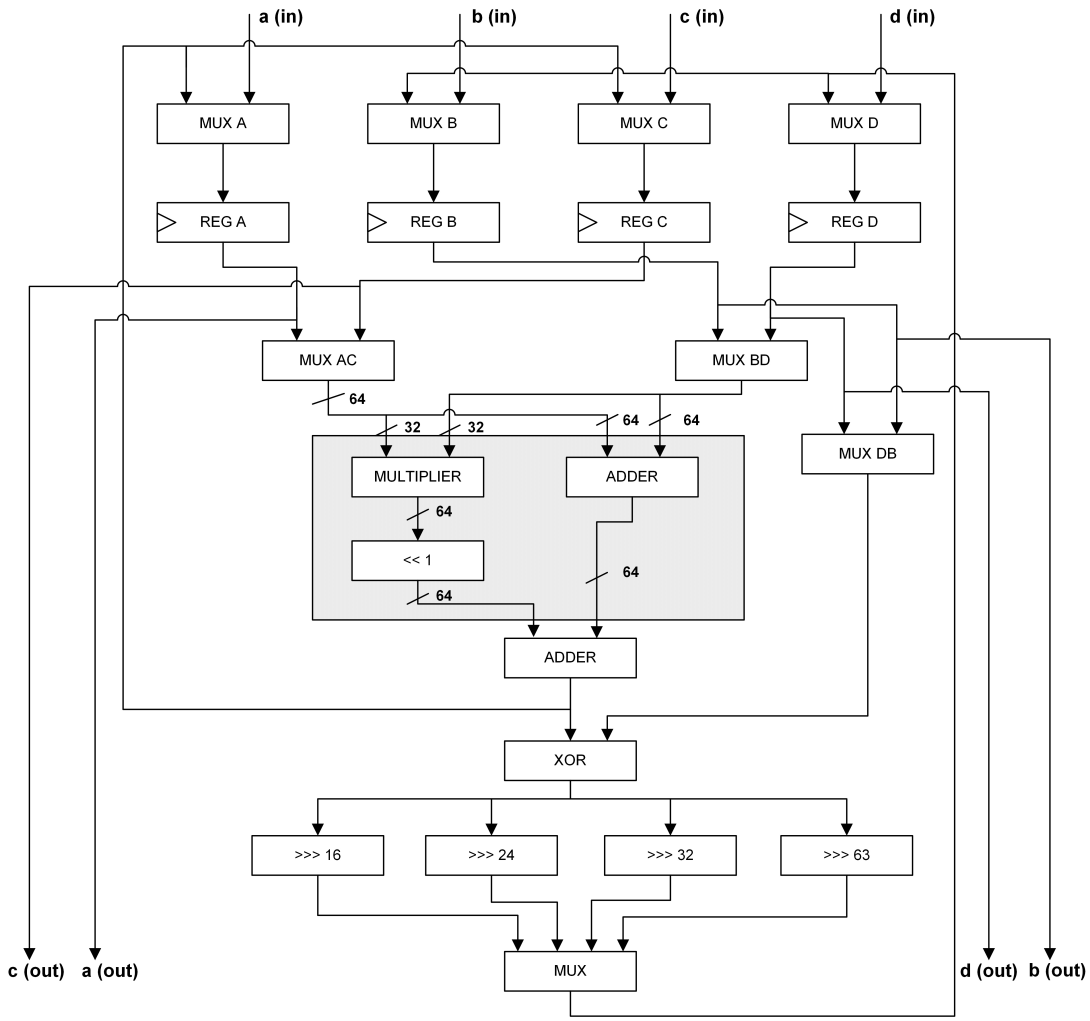

Figure 11: Sequential circuit for BlaMka *G* function

However, to facilitate understanding and mapping between the circuits and the algorithms of Figures 8a and 9a, these operations are represented as modules in Figures 10, 11, and in all other circuits presented herein. Furthermore, to simplify the figures and to facilitate their understanding, the control signals for the registers and the selection signals for the multiplexers were omitted. The sequential circuit shown is executed four times to obtain the final results generated by the algorithm at the outputs of the four registers. Such loop operation is controlled by the control unit, which is described below.

The ASM (*Algorithmic State Machine*) diagram for the control unit is shown in

Figure 12, and is the same for both datapaths of Figures 10 and 11 (but the diagram shows only the operations of BlaMka G function). This module is responsible for generating the enable signals for the registers and the selection signals for the multiplexers. In addition, it has an external control input named *start* which, by assuming the value 1, indicates that the circuit must start to operate. As a control output, it produces the *done* signal at the end of execution of the implemented algorithm, which presents the value 1 to indicate that the results are available and stable at the system outputs. The ASM diagram indicates the presence of seven states. There is an external control signal *reset* for the complete system, not shown in the figure, responsible for initializing the registers and putting the control unit in its initial state *IDLE*. There is also a *clock* signal responsible for the state transitions. The state *S_0* is responsible for receiving and storing the input variables. The following four states, *S_1* through *S_4*, update the variables stored by the registers by performing two steps of the algorithm in each state. The steps indicated next to each state correspond to those in Figure 8a. The last state, *S_5*, triggers the *done* signal indicating that the outputs are available. As can be seen in the figure, the complete system will need 6 clock cycles to perform its operation.

4.3.2 Optimizations

From the basic circuit obtained for BlaMka, one can obtain optimizations. The first optimization aims to increase the throughput, which is one of the most important characteristics for this type of algorithm. Then, how improvements could be done in terms of area utilization and power consumption are evaluated.

To obtain the optimizations, a possible strategy is to analyze the project from three different levels: algorithm, components, and microarchitecture.

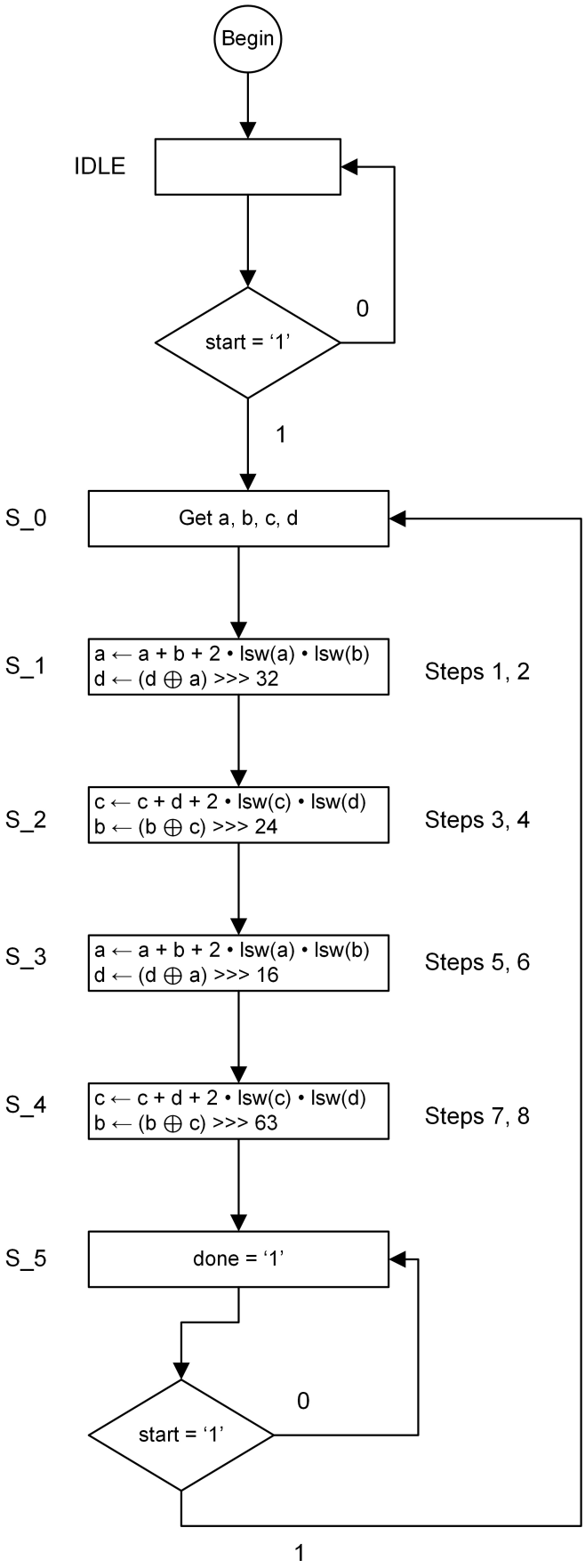

Figure 12: Control unit ASM diagram

4.3.2.1 Algorithm Level

At the algorithm level, its execution can be seen to be purely sequential, as shown previously. This does not allow the parallelism technique to be used at the algorithm level. It is worth noting that BlaMka is used as a subcomponent in more complex algorithms (for example, in password hashing schemes). At the level of these other algorithms, the parallelism technique may or may not be applicable during the use of multiple instances that implement the G function.

4.3.2.2 Components Level

In terms of components, two of them that perform arithmetic operations are present: adders and multipliers. For the permutation function, the metric of greatest interest is the throughput, which depends directly on the circuit latency. Thus, for implementing the adders and the multiplier, purely combinational circuits were chosen in order to obtain the best performance in terms of latency. This decision was based on the results presented in (DESCHAMPS; BIOUL; SUTTER, 2006; DESCHAMPS; SUTTER; CANTO, 2012). For simplicity, the implementations proposed herein use combinational adders defined by operation + of the VHDL language. For combinational multipliers, (DESCHAMPS; SUTTER; CANTO, 2012) presents various circuits and synthesis results for FPGA. For targeting more accurate results for this work, these circuits were synthesized again for both ASIC and FPGA. Combinational multipliers based on trees of counters for fast multiplication could also be used, such as the Dadda multiplier (DADDA, 1965; DADDA, 1976). Using the Synopsys Design Compiler (SYNOPSYS INC., 2015a) and a 90nm cell library for ASIC synthesis, the results for latency are presented in Table 3. Using the Xilinx Vivado Design Suite (XILINX INC., 2016b) and a 20nm Xilinx Virtex UltraSCALE for FPGA synthesis, the results for latency are presented in Table 4.

The `default_multiplier` corresponds to the synthesis of a multiplier from op-

Table 3: Comparison between combinational multipliers (ASIC)

Multipliers 32 x 32 bits	Latency (ns)
basic_base2_multiplier	42.99
basic_base16_multiplier	82.15
carry_save_multiplier	56.37
dadda_multiplier	20.00
default_multiplier	15.81
parallel_csa_multiplier	60.31
parallel_multiplier	119.16
ripple_carry_multiplier	112.75

Table 4: Comparison between combinational multipliers (FPGA)

Multipliers 32 x 32 bits	Latency (ns)
basic_base2_multiplier	39.855
basic_base16_multiplier	29.299
carry_save_multiplier	21.456
dadda_multiplier	12.515
default_multiplier	17.061
parallel_csa_multiplier	26.829
parallel_multiplier	40.766
ripple_carry_multiplier	34.453

erator $*$ of the VHDL, leaving to the synthesis tool the task of defining the specific algorithm and components to be used. For ASIC, this multiplier is observed to have the lowest latency among those presented. Hence, this multiplier was chosen for the circuits. Moreover, the `dadda_multiplier` has higher latency, but close to the `default_multiplier` and, as will be seen below, it has some features that allow optimization attempts at the microarchitecture level, such as the separation of the multiplier in two steps and the use of carry save adders. For FPGA, the `dadda_multiplier` has the lowest latency, hence it was chosen for the circuits. To maintain consistency between ASIC and FPGA, the `default_multiplier` will also be used for FPGA. In

general, these two multipliers had the best performance in terms of latency. But for this work, the multipliers chosen were those with minimum latency, because this metric directly influences the throughput. Other approaches are possible such as choosing the multiplier with the minimum power consumption or the minimum area utilization. More details about adders and multipliers, and the details about their implementation can be found in (DESCHAMPS; BIOUL; SUTTER, 2006; DESCHAMPS; SUTTER; CANTO, 2012). The architecture for the permutation function of Blake2b will be called *blake2b-ref*. The basic architecture for the permutation function of BlaMka, with the default multiplier, will be called *seq-blamka-default*. Replacing the default multiplier by the Dadda multiplier, we obtain the architecture *seq-blamka-dadda*.

4.3.2.3 Microarchitecture Level

The third approach for optimizations involves modifying the microarchitecture, which can be performed using the knowledge of the internal operation of the Dadda algorithm. The Dadda multiplier can be divided into two steps: the first one is responsible for addition of bits and carry generation, using the two 32-bit inputs and resulting in two 64-bit long partial values. The second step is where these two partial values are added, yielding the final result of the multiplication. In the case of adders, despite the use of the adder generated by the operator + of VHDL language, there are various other options. One of them is the *Carry-Save Adder (CSA)*, which the main characteristic is the application to the addition of three or more values. Noting that the two partial values generated in the first step of Dadda multiplier and the output of the first adder in Figure 11 have the same size of 64 bits, one can substitute the second adder and add these three values using a single CSA with three inputs to update variables a and c. This idea, which results in an architecture named *seq-blamka-csa3*, can be seen in Figure 13. For simplicity, only the modified parts of the sequential circuit in Figure 11 are shown. For the Dadda multiplier, its output $a * b$ can be represented as $x + y$, the sum of the partial values generated as outputs of its first step. Thus, the 1-bit left shift

operation should be applied to each partial value of the Dadda multiplier before using them in the carry-save adder, satisfying $2 * (a * b) = 2 * (x + y) = 2 * x + 2 * y$. The same reasoning applies to variables c and d.

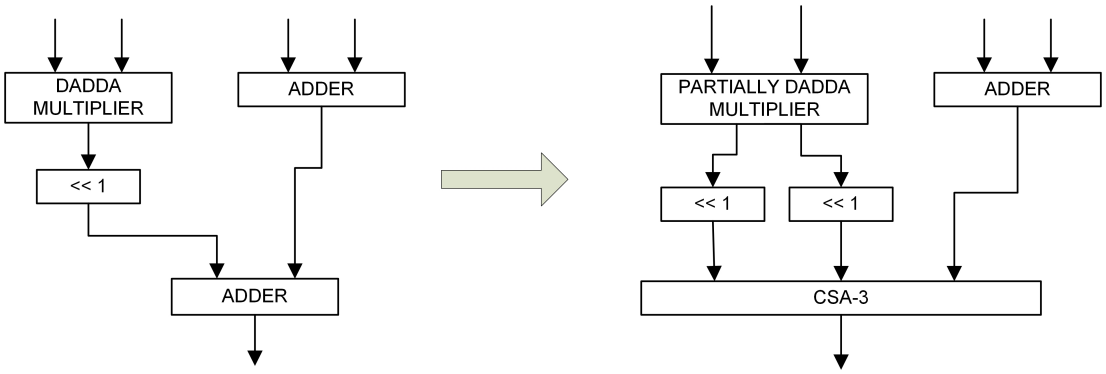

Figure 13: Dadda multiplier and CSA-3

Another possibility would be inserting registers in a possible critical path, namely between the first stage of Dadda multiplier and the carry-save adder, to increase the throughput (with impacts on area and power, to be analyzed in the next chapter), resulting in the circuit named *seq-blamka-reg*. Figure 14 shows this ideia. The control unit for this architecture will need four more states in its ASM diagram, because of the insertion of registers, totalizing 10 clock cycles.

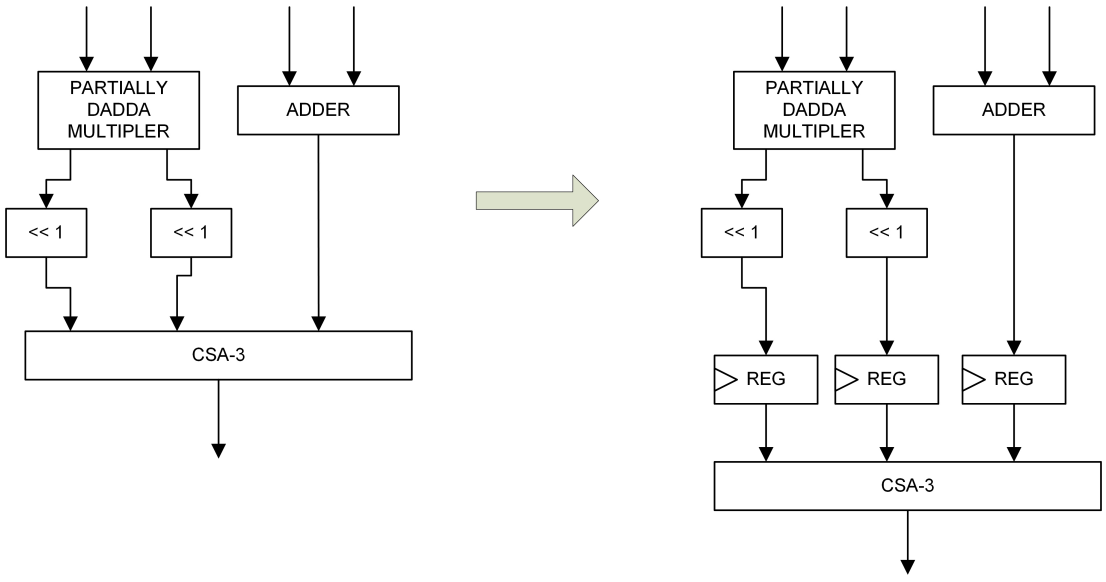

Figure 14: Dadda multiplier, CSA-3, and registers

An important and associated issue is the implementation of the proposed sequential circuits. The input variables have 64 bits each, which means that receiving them all in parallel would require a 256-bit input bus, with the same reasoning applied to the outputs. An alternative is to only allow the input/output of a smaller block of bits in parallel, for example, 64 bits (choice made for convenience because each variable has 64 bits, but several possibilities could be used). This strategy can decrease power consumption because it reduces the number of wires needed in the circuit and consequently fewer wires need to be powered in parallel. However, it is necessary to increase the number of clock cycles required to execute the algorithm. The control unit will need 16 clock cycles to perform the circuit operation. This architecture will be called *seq-blamka-serial*. In the case of FPGAs, reducing the number of input/output bits in parallel is interesting, since several families of FPGAs do not have many I/O (*Input/Output*) pins available, making it impossible for the main synthesis tools to perform the Place & Route step. It is worth noting that the FPGA tools used herein can perform a preliminary synthesis step providing results for the various metrics of interest, but these results can be very imprecise, since they do not yet take into account particularities of the chosen FPGA, such as the distribution of its basic components and the resources consumed by wires connections, contributions that will only be counted after the Place & Route step.

Combinational circuits that implement the function G can also be obtained. Figure 15 presents the basic cell of this architecture. Four cells are required, totalizing four multipliers. The combinational circuit does not have either registers or multiplexers and, as a consequence, it does not have a control unit. The complete combinational circuit can be seen in Figure 16, with the resulting architecture named *comb-blamka-default*. The same reasoning applied to the previous sequential circuits can be used to obtain the architectures *comb-blamka-dadda* and *comb-blamka-csa3*.

Finally, it is also possible to remove the first adder and use just one CSA with four

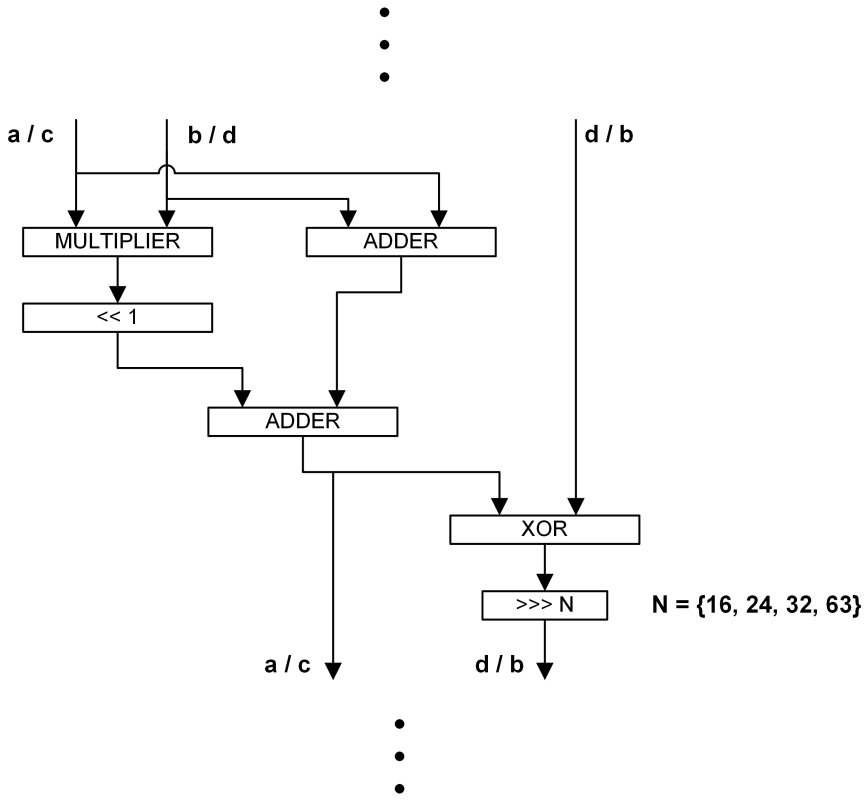

Figure 15: Combinational circuit basic cell

inputs. This attempt can reduce the area utilization and is shown in Figure 17. The resulting architecture will be called *comb-blamka-csa4*.

4.3.2.4 Remarks

As a complement, it should be noted that several other optimizations and variations could be obtained, especially if the chosen algorithm is changed, since the analyses performed in this work are focused on the chosen algorithm, respecting its characteristics. However, only a few variations were suggested to provide sufficient data to allow the proposed analysis to be performed. All optimizations were performed incrementally with the initial objective of increasing the throughput, and subsequently reducing area utilization and power consumption. Another approach would be to start with a basic

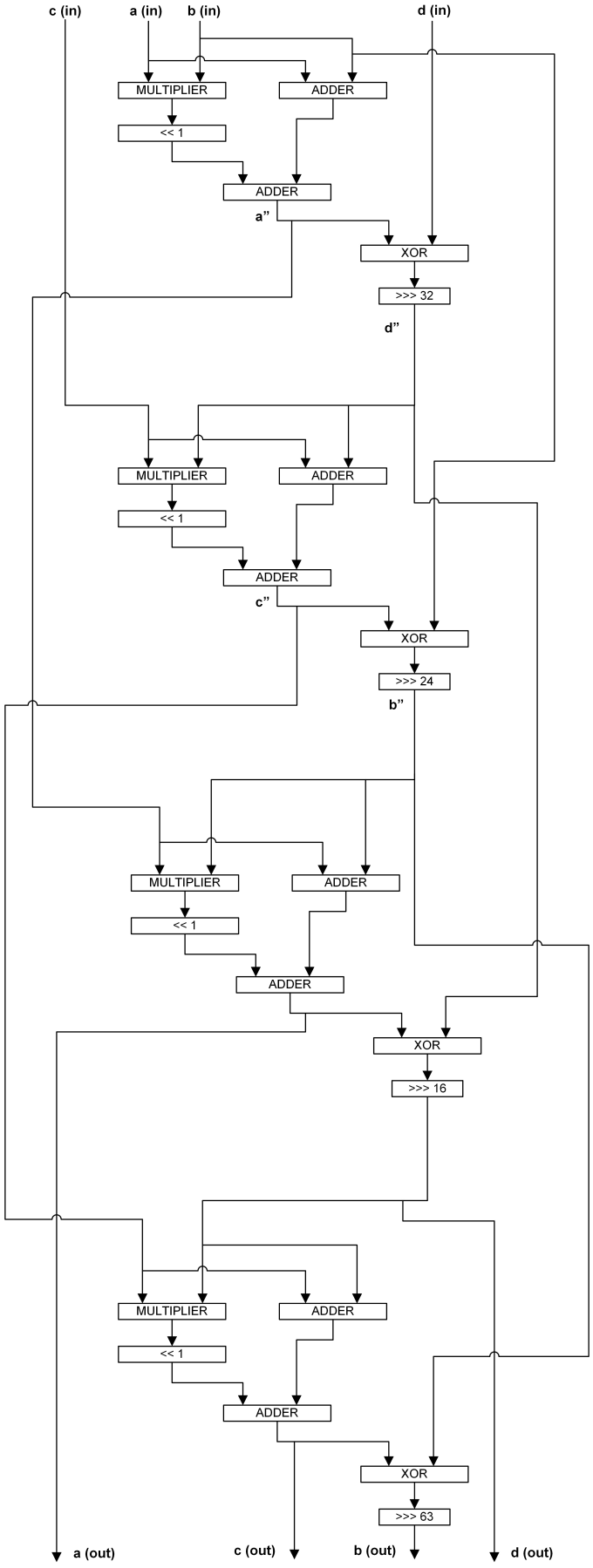

Figure 16: Complete combinational circuit

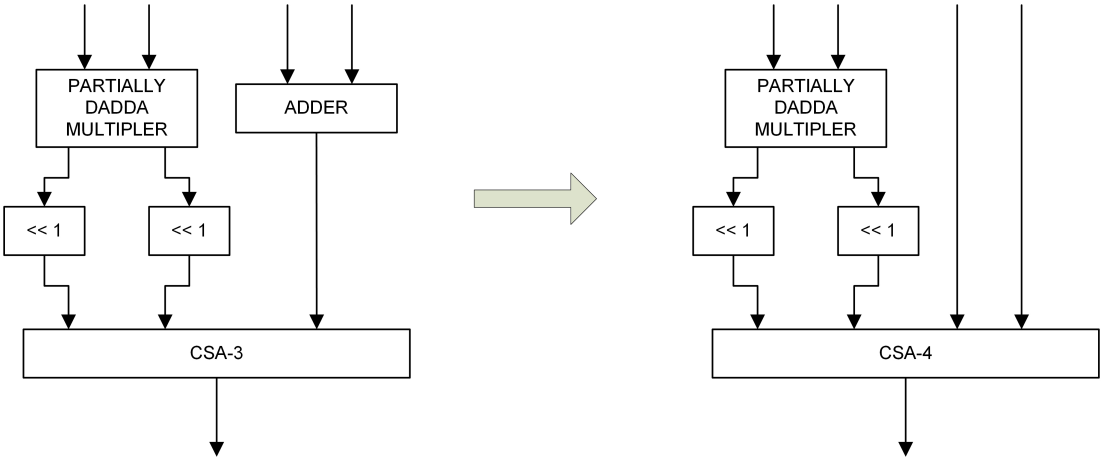

Figure 17: Dadda multiplier and CSA-4

circuit and to implement each optimization from that circuit or a combination of these two strategies.

5 EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

This chapter presents the experimental results, as well as their detailed analysis in order to evaluate the performance of the implementations. In addition, the traditional performance analysis will be expanded to explore a three-dimensional analysis involving the metrics of greater interest. The results and analyses for ASIC and FPGA are presented in different sections.

5.1 ASIC

All circuits were described using the VHDL language and synthesized for ASICs using the tools and the 90nm cell library provided by Synopsys. The latency is measured in ns and the throughput in $Gbps$. The area utilization is measured in kGE (Gate Equivalents), assuming NAND gates with an area of 5.777 μm^2. The power consumption is measured in mW, assuming an operating voltage of $0.75V$. All circuits have 256 bits of output, and the number of clock cycles required for their execution is also indicated. For combinational circuits, the number of clock cycles was assumed as 1. From these results, it is possible to derive some metrics of efficiency. For example, the throughput-area ratio, measured in $kbps/GE$, and the energy-per-bit, measured in mJ/Gb. The complete synthesis results are presented in Table 5. To assist the analysis, several graphs can be generated from the results. The architecture obtained for the permutation function of Blake2b will simply be called *reference architecture*. The set of results for a given architecture will be called a *performance point*.

Table 5: Implementation results (ASIC)

Architectures	Area (kGE)	Power (mW)	# Clock Cycles	Cycle Time (ns)	Latency (ns)	Max. Freq. (MHz)	# Output bits	Throughput (Gbps)	Throughput-to-area (kbps/GE)	Energy-per-bit (mJ/Gb)
blake2b-ref	5.23	0.405	6	4.43	26.58	225.73	256	9.631	1841.55	0.042
seq-blamka-default	14.13	0.384	6	27.36	164.16	36.55	256	1.559	110.36	0.246
seq-blamka-dadda	17.93	0.460	6	28.78	172.68	34.75	256	1.483	82.68	0.311
seq-blamka-csa3	17.54	0.489	6	29.05	174.30	34.42	256	1.469	83.74	0.333
seq-blamka-reg	19.92	0.582	10	16.63	166.30	60.13	256	1.539	77.28	0.378
seq-blamka-serial	19.76	0.423	16	17.65	282.40	56.66	256	0.907	45.88	0.466
comb-blamka-default	35.17	1.680	1	111.52	111.52	—	256	2.296	65.27	0.732
comb-blamka-dadda	52.07	1.876	1	136.14	136.14	—	256	1.880	36.11	0.998
comb-blamka-csa3	52.82	2.104	1	105.51	105.51	—	256	2.426	45.94	0.867
comb-blamka-csa4	51.78	2.072	1	107.78	107.78	—	256	2.375	45.87	0.873

5.1.1 One-Dimensional Analysis

To all one-dimensional graphs, the vertical bars from left to right correspond to the legend from top to down. The first graph in Figure 18 shows the absolute throughput values of the various architectures for the BlaMka permutation function compared to the value obtained for the reference architecture.

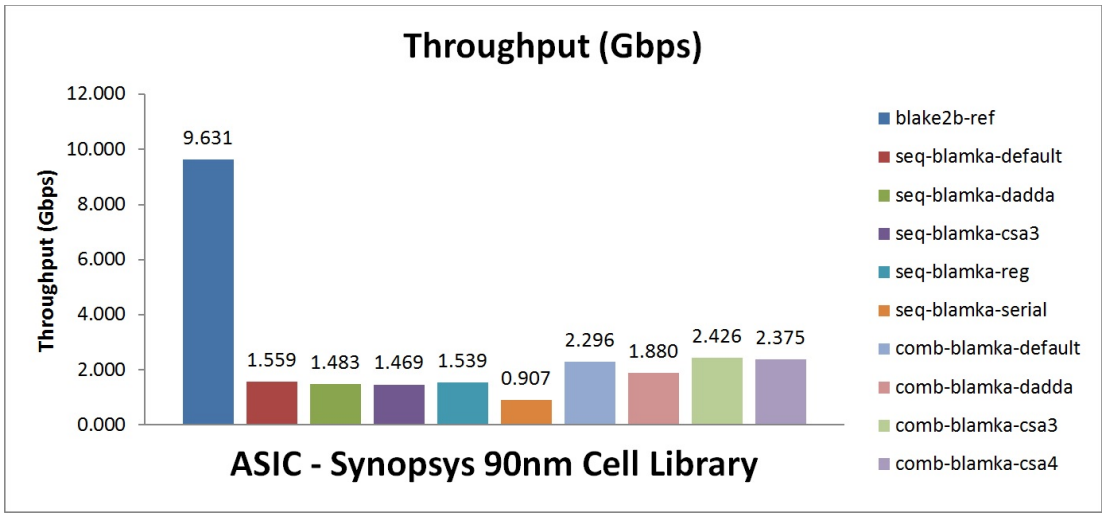

Figure 18: Throughput comparison (ASIC)

For BlaMka, the architecture *seq-blamka-default* has the highest throughput between the sequential circuitry and the architecture *comb-blamka-csa3* between the combinational versions. In general, the latter showed the highest throughput (2.426 *Gbps*), 56% higher than the best sequential version (1.559 *Gbps*). In addition, it can be noted that all combinational architectures presented throughput superior to the sequential architectures. The lowest overall performance occurred in the *seq-blamka-serial* sequential architecture, which receives inputs and outputs serially in 64-bit blocks. This was a theoretically expected result, since this architecture, which aims to reduce power consumption, on the other hand requires a greater number of clock cycles for its complete execution. It is also possible to verify that all architectures for BlaMka have lower throughput than the reference architecture. Quantitatively, these differ-

ences can be visualized in the graph of Figure 19, which indicates the throughput of each proposed architecture for BlaMka as a fraction of the throughput of the reference architecture. The architecture with the highest throughput, *comb-blamka-csa3*, has a throughput of about 25% of that obtained for the reference architecture. This fact is the first evidence that shows how the inclusion of multiplications affects the performance of the BlaMka algorithm in relation to this metric.

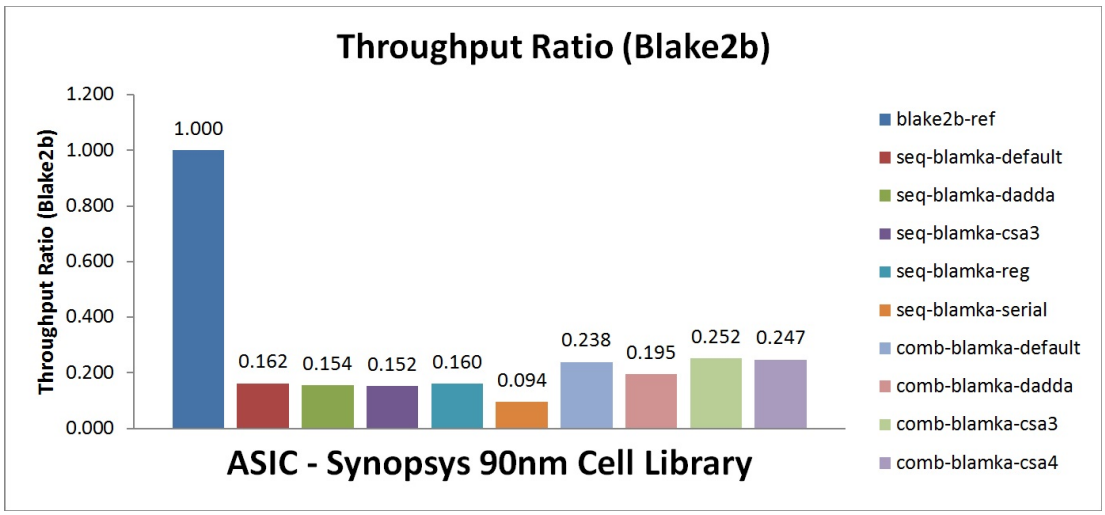

Figure 19: Throughput ratio comparison (ASIC)

Figures 20 and 21 show the performance in relation to the silicon area and power consumption, respectively. In terms of area, it is possible to see that the combinational architectures, as expected, presented the highest values. Among them, the inclusion of the Dadda multiplier caused a significant increase in area compared to the combinational circuit with the default multiplier, but this discrepancy is attributed to the presence of 4 Dadda multipliers in the combinational circuits. Approximately, the difference in terms of area is still similar to that of the sequential circuits, as it is possible to visualize when comparing the architectures *seq-blamka-default* and *seq-blamka-dadda*. It should also be noted that the BlaMka more compact architecture that was obtained, *seq-blamka-default*, still has an area about three times greater than the reference architecture, which illustrates the minimal increase in area cost caused by the

inclusion of combinational multipliers (for maximizing the throughput). However, it is not the overall minimum increase in area, because it is possible to use sequential multipliers, which are very compact in terms of area but need more clock cycles to execute and consequently reduce the throughput.

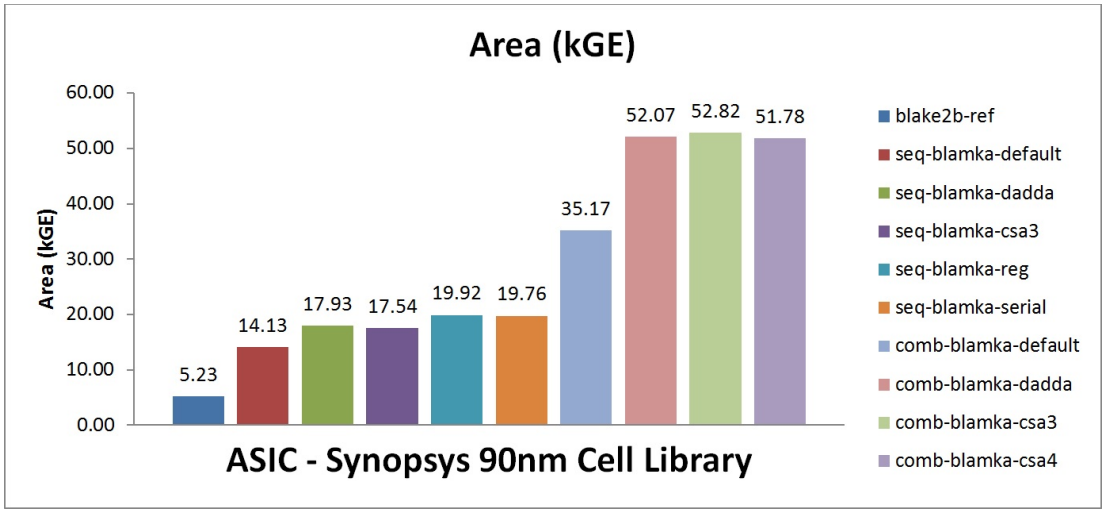

Figure 20: Area comparison (ASIC)

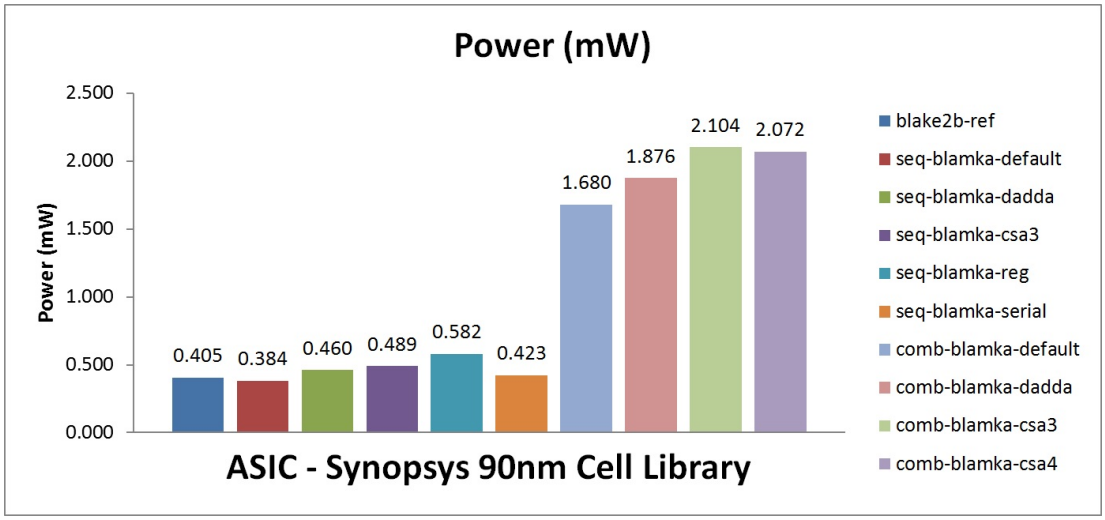

Figure 21: Power comparison (ASIC)

In the power graph, we can see the difference between the power consumption of the sequential and the combinational circuits, attributed not only to the greater number

of modules used in the combinational versions, especially the multipliers, but mainly to the increase in the number of necessary wire connections. However, there were no substantial differences in power between the sequential implementations for BlaMka and the reference architecture, due primarily to the maximum frequency that each circuit can operates. In addition, the synthesis tool has several methods to implement multipliers and the respective circuit in which they are included in a very optimized way. This can be visualized in the power consumption for the *seq-blamka-default* architecture, which is smaller than the consumption for the reference architecture. The power consumption is higher for the sequential versions with the Dadda multiplier, but it is still close to the consumption of the reference architecture. The power for the combinational circuits was estimated assuming that these circuits are the critical path of an external system. Thus, the frequency of operation is the inverse of the combinational circuit cycle time.

Finally, two graphs representing efficiency measures are shown in Figures 22 and 23. In terms of *throughput-to-area*, the architecture *seq-blamka-default* has the highest efficiency, although the other architectures do not present very high discrepancies. In terms of efficiency in energy, the combinational architectures have a consumption superior to that of the sequential architectures, when performing the same amount of processing. It is worth noting that, despite the proximity in power consumption between the sequential versions of BlaMka and the reference architecture, the latter is about 6 times more efficient. And, although the minimum difference in area utilization is about 3 times for the sequential circuits, the efficiency of the reference architecture in relation to the amount of throughput it can obtain on average by a certain amount of area is about 18 times greater than the best performance for BlaMka. These efficiency measures again evidence the impact caused by the inclusion of multiplications, mainly in the financial cost for manufacturing and operating the circuits, represented by the area utilization and the power consumption.

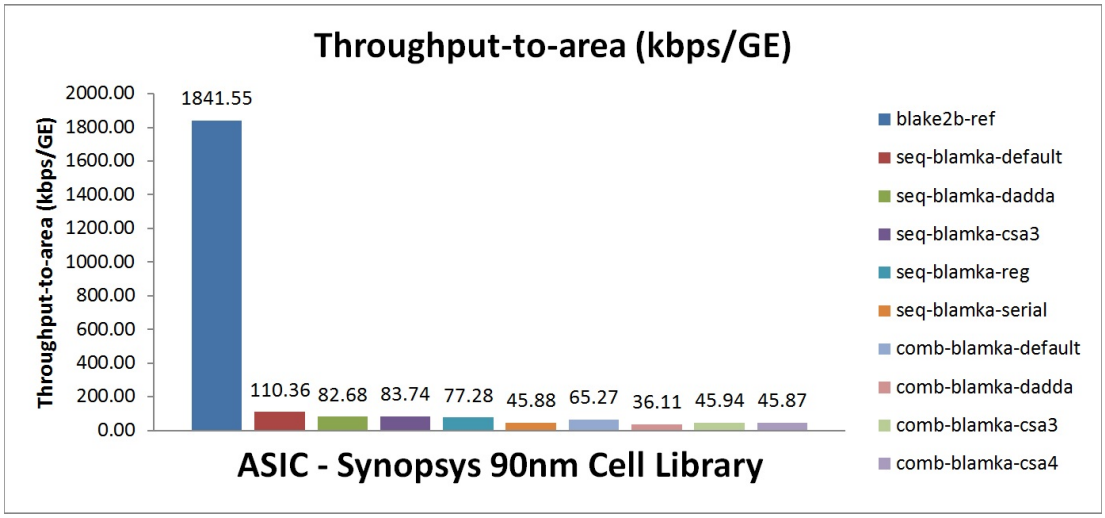

Figure 22: Throughput-to-area comparison (ASIC)

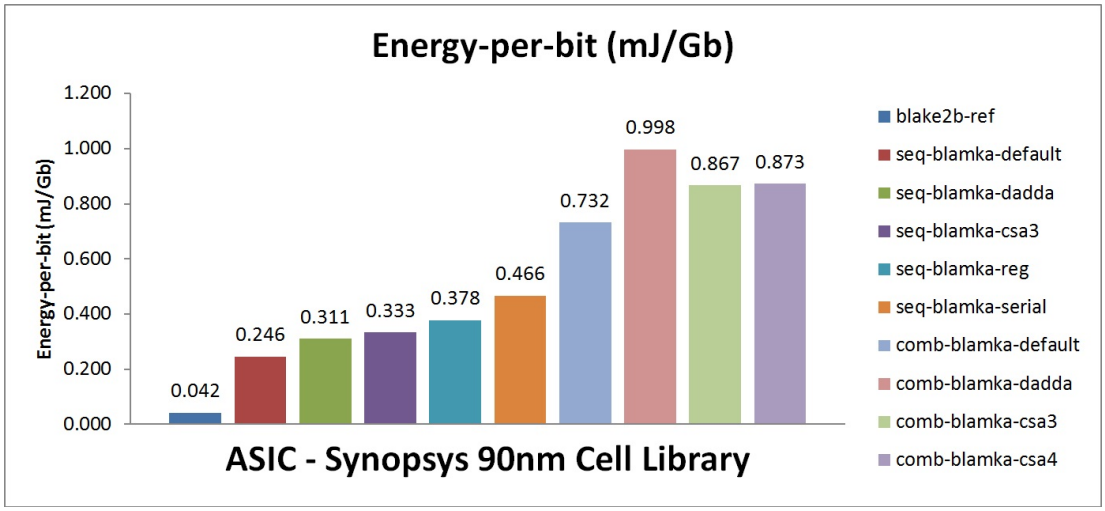

Figure 23: Energy-per-bit comparison (ASIC)

5.1.2 Two-Dimensional Analysis

After the first analysis, observing each metric of interest, a two-dimensional analysis of the trade-offs present in the implementations can be performed. In this work, three trade-offs of this type will be analyzed. The first one, between area and throughput, is shown in Figure 24. The trend line presented for this graph as the ones for the

other two-dimensional graphs are only approximate and they agree with the number of performance points available. The more points we have, the more accurate that curve will be. However, even roughly, it can help us to understand the behavior of the analyzed trade-off.

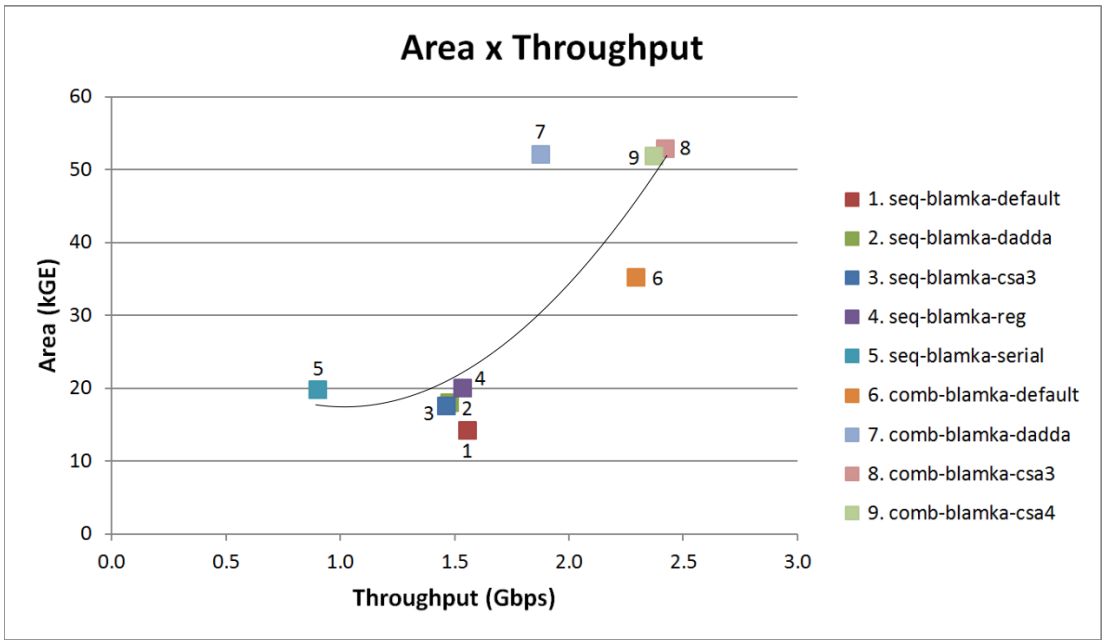

Figure 24: Trade-off between area and throughput (ASIC)

As can be seen in the graph, as we want to increase the throughput of the system, it will be necessary to increase the use of the silicon area. However, it is possible to observe that some circuits with a throughput approaching 1.5 Gbps (points 1 to 4) have an area very close to each other, which allows directly selecting the circuit with the greatest throughput, since the differences in area utilization are small. In addition, we have a point with a similar area, but with a significant decrease in throughput (point 5); if only this trade-off is observed, it could be concluded that it is not a relevant option. It can also be observed that if the objective is to obtain higher throughput, the best options are among the combinational versions; however, the area tends to grow significantly in response. Yet, as commented above, again we have some points with

throughput close to each other, but with relatively different areas (points 6, 8, and 9). Thus, assuming that we initially desire maximum throughput, it is possible to give up a small amount of that throughput by a significant decrease in area. Finally, regardless of the approach, the choice within this trade-off can be misleading, because when judging each performance point, we lack the information on the power consumption, another crucial factor in terms of cost. This absence is not only graphical, but also conceptual, as analyzed and explained in chapters 1 and 3.

In Figure 25, we have the trade-off between power and throughput. The considerable increase in power is directly verified when the maximum throughput is desired.

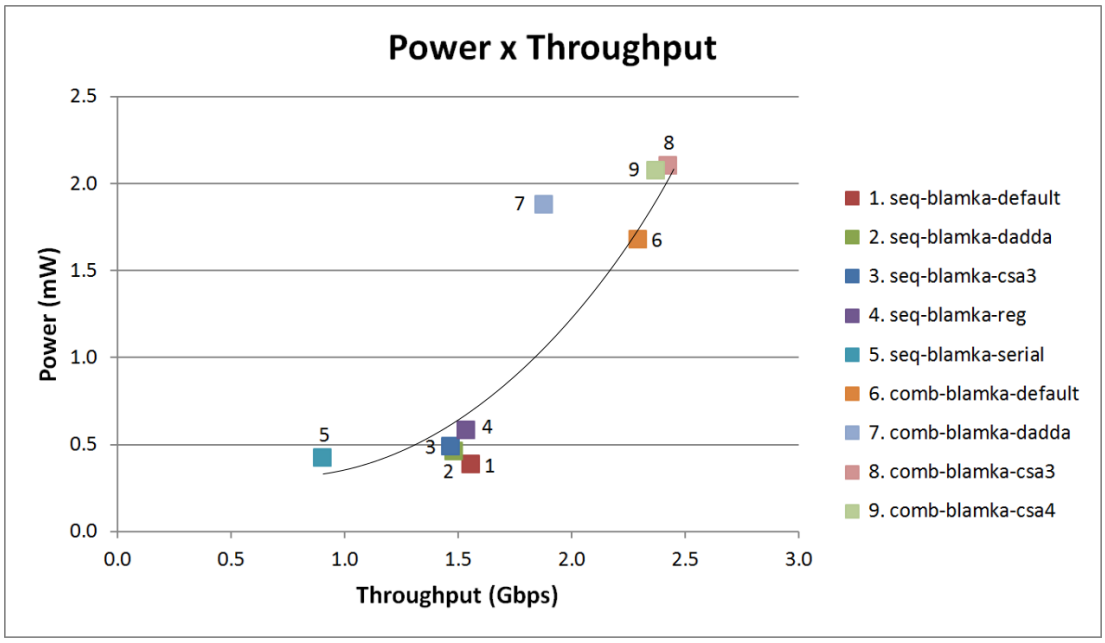

Figure 25: Trade-off between power and throughput (ASIC)

Some circuits have lower power consumption but are limited to a throughput range from 0.9 to 1.6 Gbps (points 1 to 5). If the focus is on mobile devices, it might be worth to giving up maximum throughput for a slightly lower throughput, but substantially reducing power consumption, a factor of crucial importance for this type of device that usually is battery-powered. Of course, it is possible to see how power responds to the trade-off between area and throughput, but this is possible here because of the small

number of performance points. If we have dozens of points, observing the response of a third metric would become complicated. Additionally, we can analyze the trade-off between area and power shown in Figure 26.

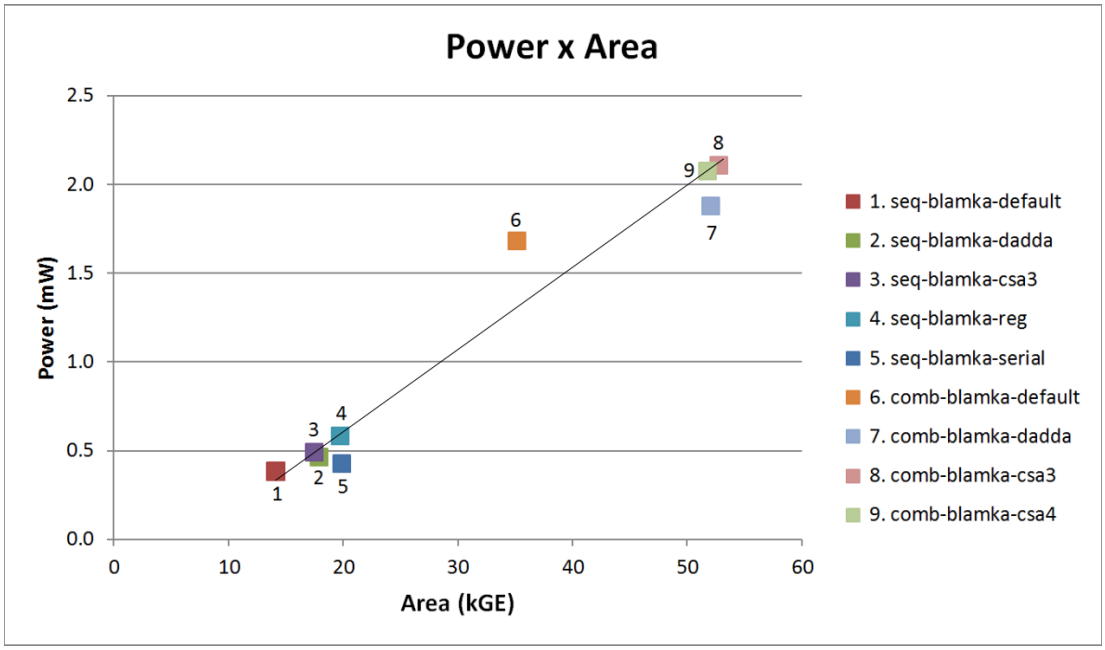

Figure 26: Trade-off between power and area (ASIC)

More compact versions are observed to have lower power consumption, while versions with greater area utilization present higher consumption of power, because power is directly dependent of the number of logic gates in the circuit. Area and power together form a crucial factor in the financial cost for the design. In ASICs, despite the low cost of area for high production quantities, the power consumption directly influences its application, mainly in mobile devices. In addition, the higher the power consumption, the greater the thermal dissipation, which can result not only in the rapid depletion of the battery but also in shortening the life of several components if an efficient cooling system is not present.

5.1.3 Three-Dimensional Analysis

The performance of a circuit in terms of speed has always been a requirement in most projects. Nowadays, with the massive use of devices with restrictions of energy resources and physical size, area and power have also become important factors. From the two-dimensional analyses it is possible to identify two main difficulties in analyzing the behavior of these metrics together. The first is to observe the response of the third metric when analyzing the trade-off between the other two. The second is the difficulty in obtaining balanced performance points. For example, instead of searching for the maximum optimization of a given metric, it is possible to decrease the performance of this metric to obtain better performances for the others. An analogy can be made by treating each metric as a currency, which allows exchanging some units of this currency from one metric to another. To address these difficulties, an analysis of a three-dimensional trade-off among area, power, and throughput is proposed. To perform this analysis, two approaches are presented and can be used together.

5.1.3.1 Projections Approach

Figure 27 presents the first approach, which is to treat each two-dimensional graph as a projection on a three-dimensional graph. As aforementioned, the curves are approximate and the three-dimensional analysis is as well. However, extending the reasoning used in the two-dimensional case, one can imagine each curve as a projection of a surface in space on a plane, roughly representing the relationship among area, power, and throughput. Designing such a surface is not possible in this case, owing to the finite and small number of available performance points and to the absence of mathematical models (that are not part of the scope of this work) to represent the relationships among the metrics. Moreover, even the estimation would be very arbitrary; we therefore chose only to use the curves as projections.

In this graph, each performance point is numbered and has its counterpart in each

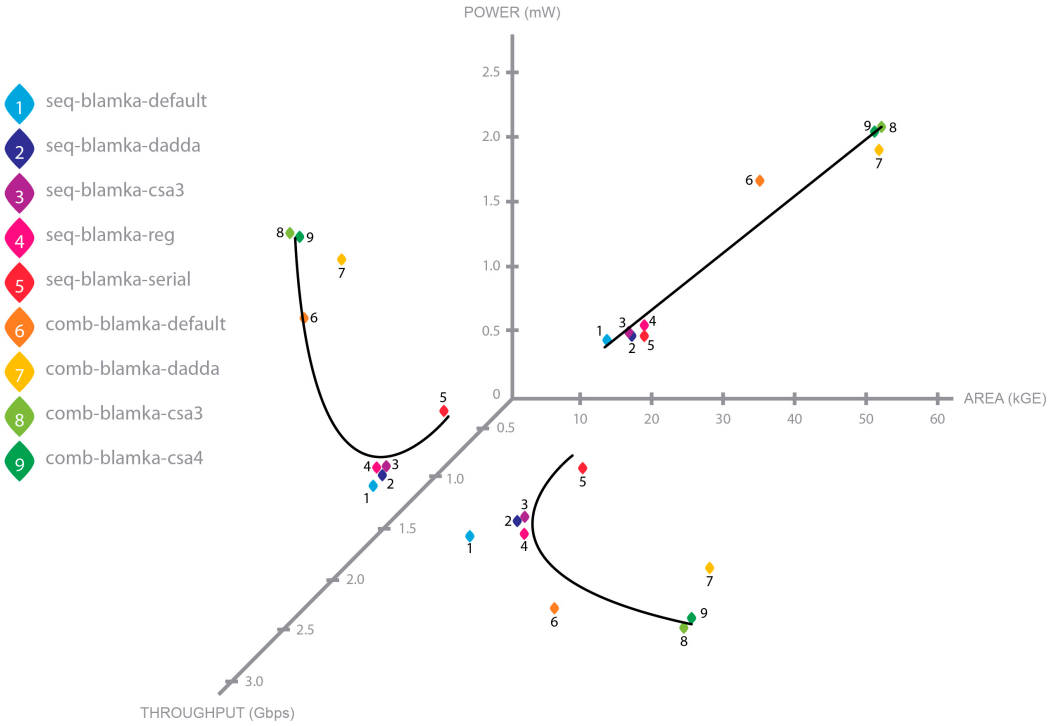

Figure 27: Projections approach (ASIC)

plan for easy visualization and analysis. Starting from the two-dimensional analyses carried out previously, one can expand them, but now observing the behavior of the three metrics together. For example, we can start at point 1 (Figure 28) and if the goal is to obtain maximum throughput, we can observe by the *power x throughput* plane that points 8 and 9 have greater throughput than point 1 and provide close solutions for both throughput and power, with point 8 having the best throughput performance (Figure 29). However, by slightly sacrificing the throughput, we can go to point 6 and observing this point in the plane *power x area*, there is a considerable reduction in the area utilization (Figure 30).

It is possible to verify quantatively these variations using the data of Table 5, representing each point as a triple (**T**hroughput in Gbps, **A**rea in kGE, **P**ower in mW). Moving from point 1 to point 8:

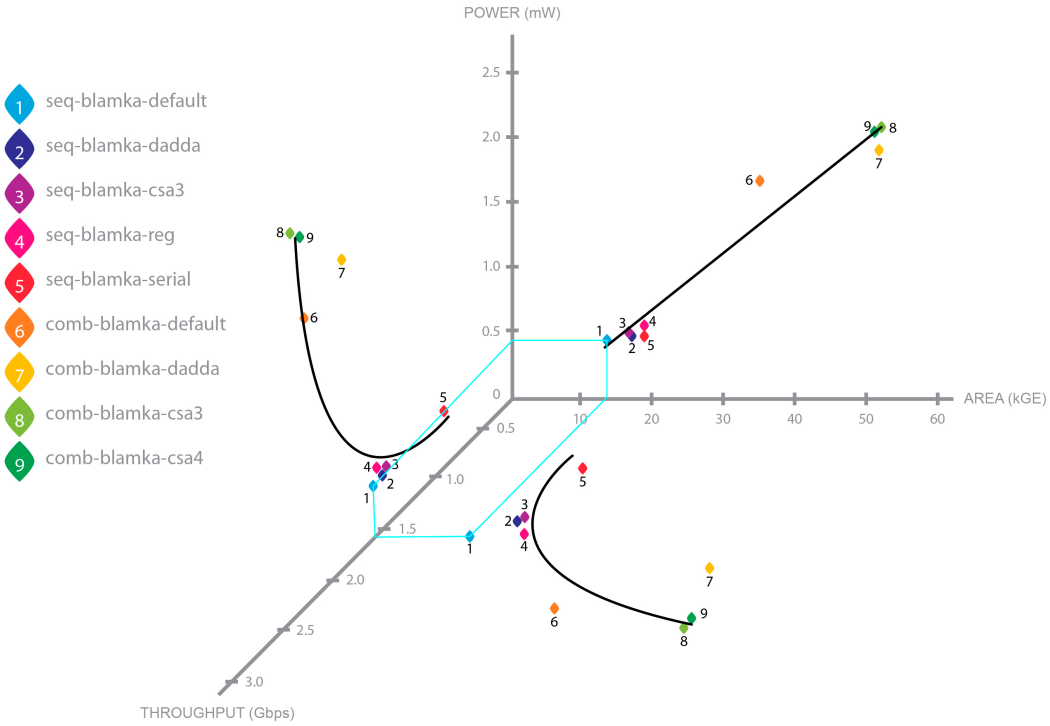

Figure 28: Projections approach - Point 1 (ASIC)

$$Point\ 1 : (1.559, 14.13, 0.384) \Rightarrow Point\ 8 : (2.426, 52.82, 2.104)$$

$$\Delta T = 2.426 - 1.559 = 0.867\ Gbps \quad [\Delta T\% = 0.867/1.559 = +55.6\%]$$

$$\Delta A = 52.82 - 14.13 = 38.69\ kGE \quad [\Delta A\% = 38.69/14.13 = +273.8\%]$$

$$\Delta P = 2.104 - 0.384 = 1.720\ mW \quad [\Delta P\% = 1.720/0.384 = +447.9\%]$$

Thus, to increase the throughput in 1 Gbps, it is possible to calculate the response in area and power:

$$1\ Gbps \rightarrow 44.62\ kGE$$

$$1\ Gbps \rightarrow 1.98\ mW$$

We can perform the same calculations when moving from point 1 to point 6:

$$Point\ 1 : (1.559, 14.13, 0.384) \Rightarrow Point\ 6 : (2.296, 35.17, 1.680)$$

$$\Delta T = 2.296 - 1.559 = 0.737\ Gbps \quad [\Delta T\% = 0.737/1.559 = +47.3\%]$$

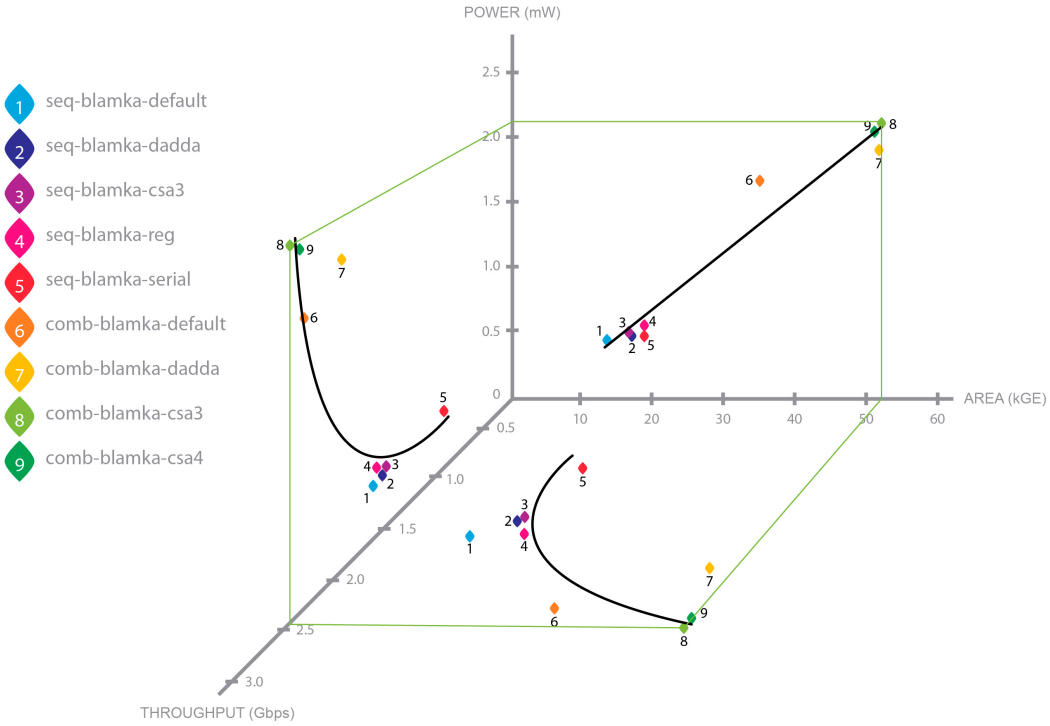

Figure 29: Projections approach - Point 8 (ASIC)

$$\Delta A = 35.17 - 14.13 = 21.04\ kGE \quad [\Delta A\% = 21.04/14.13 = +148.9\%]$$

$$\Delta P = 1.680 - 0.384 = 1.296\ mW \quad [\Delta P\% = 1.296/0.384 = +337.5\%]$$

To increase the throughput in 1 Gbps, the response in area and power is:

$$1\ Gbps \rightarrow 28.55\ kGE$$

$$1\ Gbps \rightarrow 1.76\ mW$$

These calculations show that in the case of point 8, it was possible to sacrifice slightly the throughput to reduce significantly area and power when moving to point 6.

On the other hand, if we have power limitation, by looking at the plane *power x area*, the points that have areas close to each other keep this proximity in terms of power. However, by looking at the corresponding points in the *area x throughput* plane, we see that point 5, which represents the architecture *seq-blamka-serial*, has lower throughput than the others. This makes it evident that the strategy used to reduce power

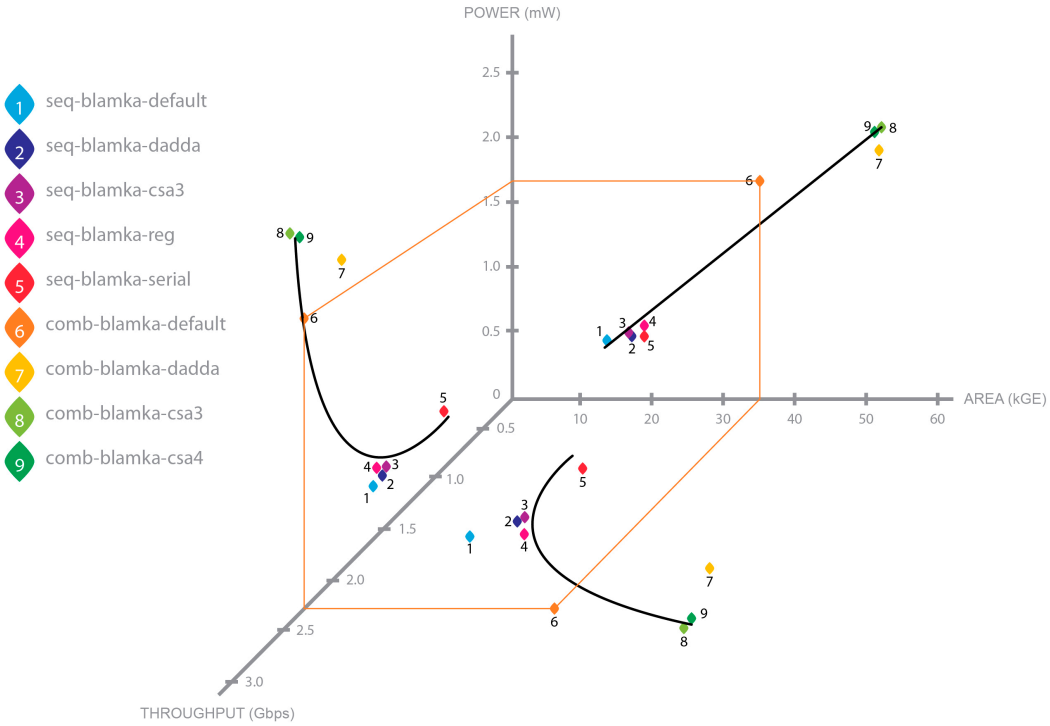

Figure 30: Projections approach - Point 6 (ASIC)

actually reduced it, but such a reduction was largely compensated by the decrease in throughput performance, which may not be an attractive alternative except in very specific applications where energy is an extremely important factor and speed is not.

We can also obtain a balanced performance point among the three metrics. For example, points 6 to 9 have higher power consumption as well as area utilization compared to points 1 to 5. Then, it is possible to choose between points 1 and 5 because they have lower and very similar power consumption, but differ in terms of area and throughput. It is possible to see in planes *power x throughput* and *power x area* that point 1 has the best throughput for this group of points, and it also has the lowest area in that group. It is evident from this representation that the conclusions obtained in the one-dimensional and the two-dimensional analyses can be obtained in this three-dimensional analysis, and it also facilitates the three-dimensional understanding of the behavior of these metrics, even if approximate. In addition, it is simpler to find bal-

anced performance points that do not sacrifice one metric in detriment of others, but rather adjust these metrics to each other until values are obtained that are not the maximum values, but which together provide a balanced performance among the points available.

Another variation is to change the throughput axis by its inverse. In this way, all the axes will have the same direction: greater values represent less desirable results (Figure 31). From that, it is possible to calculate a volume corresponding to each point and obtain a balanced performance point in a more direct way. Table 6 presents these volume calculations. It is possible to see that point 1 presents the lowest volume, corresponding to a better performance involving the three metrics together. In addition, if maximum throughput is desired, we can choose the point 8. However, the volume for this point is the highest. Therefore, it is possible to move to point 6, where we have a slightly decrease in throughput but a considerable decrease in volume, constituting a more balanced choice. This was the same result previously obtained but now in a more direct way (Figure 32). The volume establishes a criterion to select optimizations.

Table 6: Volume comparison (ASIC)

Architectures	Volume (kGE*mW/Gbps)
seq-blamka-default	3.476
seq-blamka-dadda	5.567
seq-blamka-csa3	5.844
seq-blamka-reg	7.528
seq-blamka-serial	9.217
comb-blamka-default	25.745
comb-blamka-dadda	51.951
comb-blamka-csa3	45.807
comb-blamka-csa4	45.179

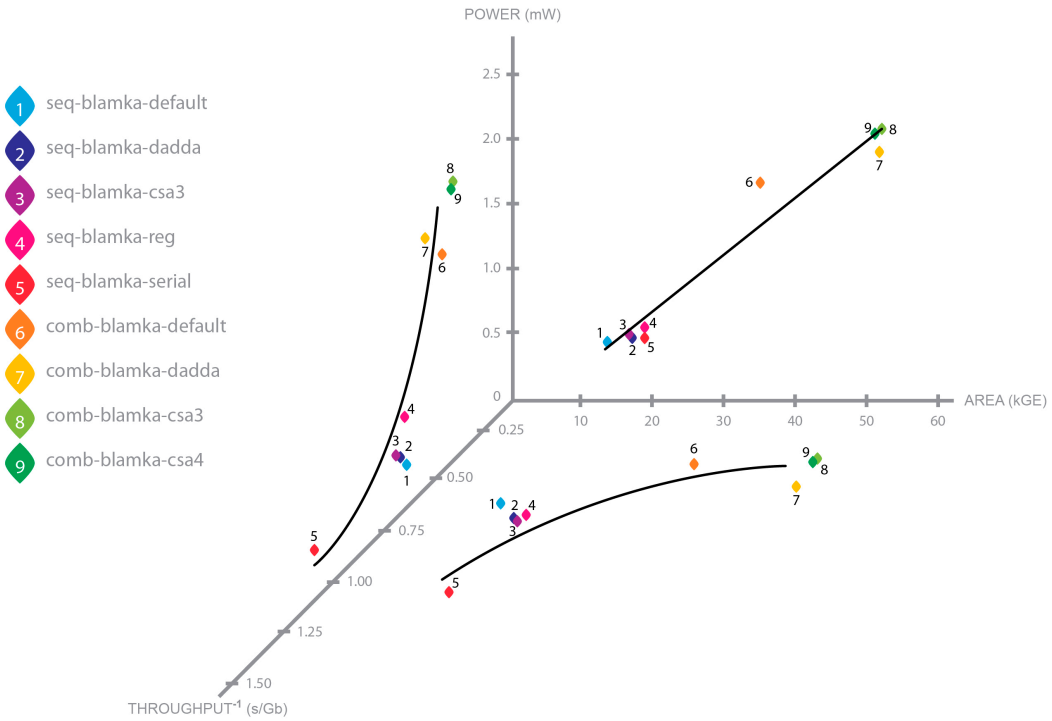

Figure 31: Projections approach - Inverse of throughput (ASIC)

5.1.3.2 Planes Approach

The second approach for a three-dimensional analysis is shown in Figure 33. In this graph, performance points are grouped into planes that go through certain values in one axis and allow us to analyze performance variations as we move from one group to another. This approach is based on theoretical models presented in (DEMICHELI, 1994). Using the throughput as a reference, three groups were formed: points with throughput near 1 Gbps, points with throughput near 1.5 Gbps, and points with throughput near 2.3 Gbps. These groups are approximate and each plane contains performance points where throughput is around and close to the throughput value through which the plane passes, but they are not exactly equal to this value. The green and orange planes show that point 7, although it is in the same plane as points 1 to 4, has a larger area and power consumption than these other points. From the green plane, one can have a 50%

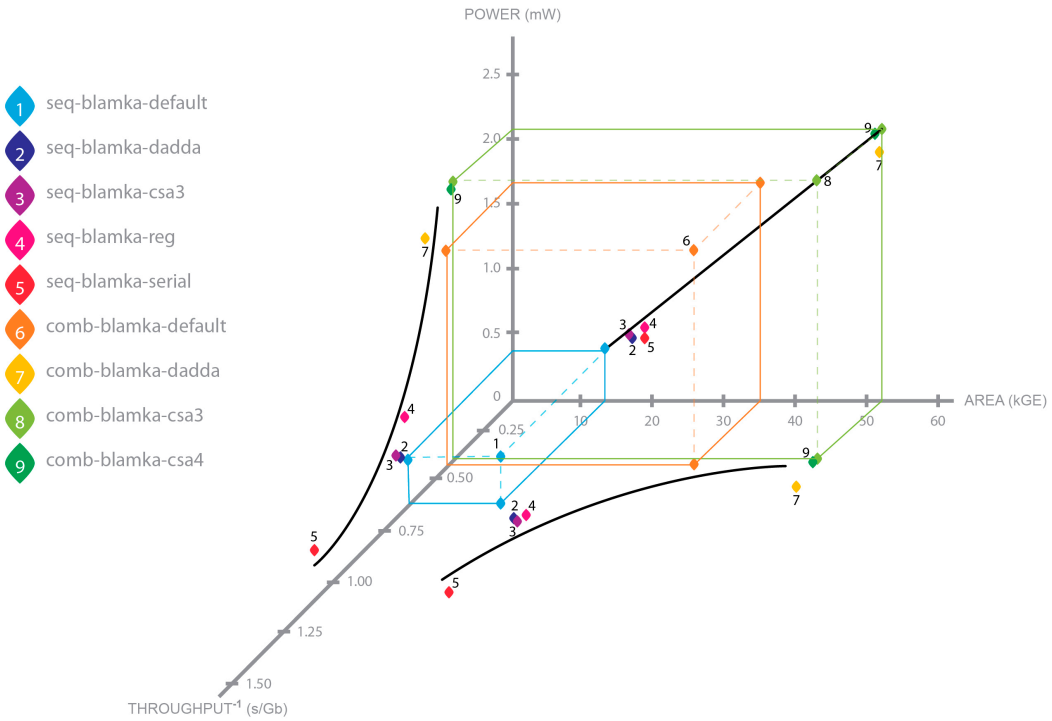

Figure 32: Projections approach - Volume (ASIC)

increase in throughput by going to the orange plane. This gain would be compensated by the increase in area and power of point 7. However, we have points 1 to 4, besides presenting the same increase in throughput, present performance similar to point 5 in terms of area and power. Point 1 performs better in all metrics for a point in this plane. From the orange plane, approximately another 50% increase in throughput can be obtained by going up to the purple plane. This division into planes also makes it possible to find points with a balanced performance between the metrics. As can be seen, points 1 to 4 do not have the highest overall values for throughput, but they do not have the minimum values, either. In addition, they have low power consumption compared to the values in the plane with the maximum throughput. Finally, they have low area utilization compared to the point with minimum throughput.

It is possible to verify quantatively these variations using the data of Table 5. The throughput interval in each plane can be obtained and an average throughput can be

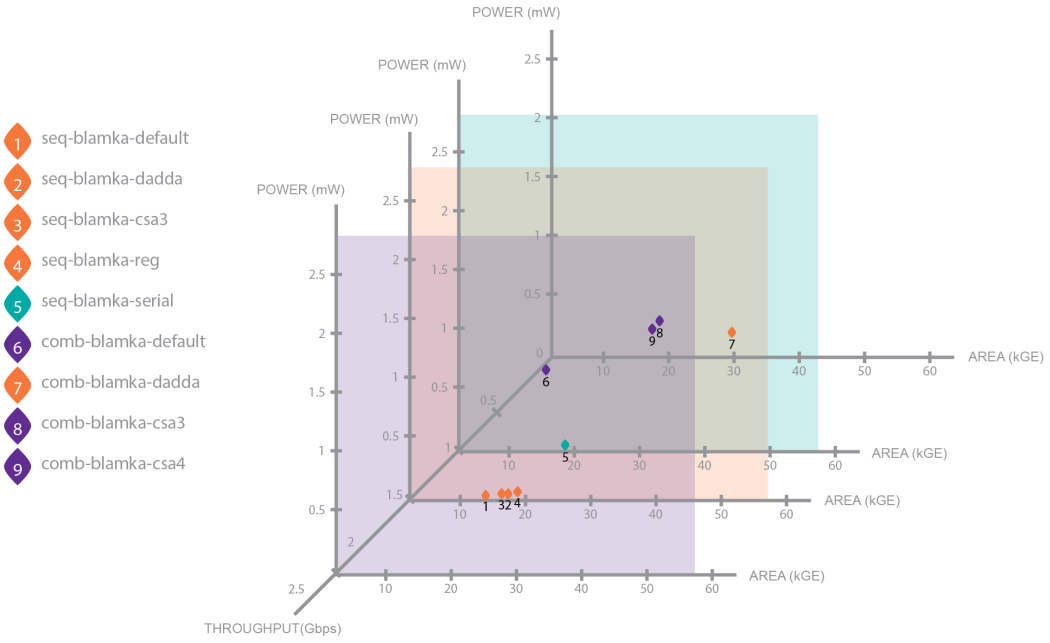

Figure 33: Planes approach (ASIC)

calculated:

$$\textit{Green Plane} : [0.907] \rightarrow T_{\text{average}} = 0.907\ \textit{Gbps}$$

$$\textit{Orange Plane} : [1.469, 1.880] \rightarrow T_{\text{average}} = 1.674\ \textit{Gbps}$$

$$\textit{Purple Plane} : [2.296, 2.426] \rightarrow T_{\text{average}} = 2.361\ \textit{Gbps}$$

With the throughput established in some interval, the remaining metrics can be analyzed. For example, starting with point 1 in the orange plane and going to the purple plane, it is possible to see that the best alternative is point 6, because on average it has the same increase in throughput than points 8 and 9 but with the minimum area and power for the purple plane. Representing each point as (**A**rea in kGE, **P**ower in mW), it is possible to calculate the response in terms of area and power:

$$Orange\ Plane \Rightarrow Purple\ Plane : \uparrow 50\%\ throughput\ on\ average$$

$$Point\ 1 : (14.13, 0.384) \Rightarrow Point\ 6 : (35.17, 1.680)$$

$$\Delta A = 35.17 - 14.13 = 21.04\ kGE \quad [\Delta A\% = 21.04/14.13 = +148.9\%]$$

$$\Delta P = 1.680 - 0.384 = 1.296\ mW \quad [\Delta P\% = 1.296/0.384 = +337.5\%]$$

5.2 FPGA

All circuits were described using the VHDL language and synthesized for FPGAs using the Xilinx tools and the board Virtex-UltraScale VCU 108 Evaluation Platform with a 20nm Virtex UltraSCALE model xcvu095-ffva2104-2-e. The latency is measured in ns and the throughput in $Gbps$. The area utilization is measured in CLB (Configurable Logic Blocks). The power consumption is measured in W. All circuits have 256 bits of output, and the number of clock cycles required for their execution is also indicated. For combinational circuits, the number of clock cycles was assumed to be 1. From these results, it is possible to derive some metrics of efficiency. For example, the throughput-area ratio, measured in $Mbps/CLB$, and the energy-per-bit, measured in mJ/Gb. The circuits were synthesized without using DSPs (*Digital Signal Processor*). The complete synthesis results are presented in Table 7. To assist the analysis, several graphs can be generated from the results. The architecture obtained for the permutation function of Blake2b will be simply called *reference architecture*. The set of results for a given architecture will be called a *performance point*.

Table 7: Implementation results (FPGA)

Architectures	Area (CLB)	Power (W)	# Clock Cycles	Cycle Time (ns)	Latency (ns)	Max. Freq. (MHz)	# Output bits	Throughput (Gbps)	Throughput-to-area (Mbps/CLB)	Energy-per-bit (mJ/Gb)
blake2b-ref	109	1.442	6	3.07	18.44	325.31	256	13.880	127.34	103.892
seq-blamka-default	452	1.175	6	9.80	58.78	102.08	256	4.356	9.64	269.773
seq-blamka-dadda	378	1.297	6	6.83	40.97	146.43	256	6.248	16.53	207.591
seq-blamka-csa3	347	1.350	6	6.37	38.23	156.96	256	6.697	19.30	201.582
seq-blamka-reg	432	1.601	10	3.50	34.97	285.96	256	7.321	16.95	218.699
seq-blamka-serial	353	1.130	16	3.58	57.20	279.72	256	4.476	12.68	252.484
comb-blamka-default	1052	1.073	1	49.02	49.02	—	256	5.222	4.96	205.480
comb-blamka-dadda	975	1.128	1	41.54	41.54	—	256	6.163	6.32	183.040
comb-blamka-csa3	1018	1.150	1	39.26	39.26	—	256	6.521	6.41	176.341
comb-blamka-csa4	1007	1.143	1	41.75	41.75	—	256	6.132	6.09	186.398

5.2.1 One-Dimensional Analysis

To all one-dimensional graphs, the vertical bars from left to right correspond to the legend from top to down. The first graph in Figure 34 shows the absolute throughput values of the various architectures for the BlaMka permutation function compared to the value obtained for the reference architecture.

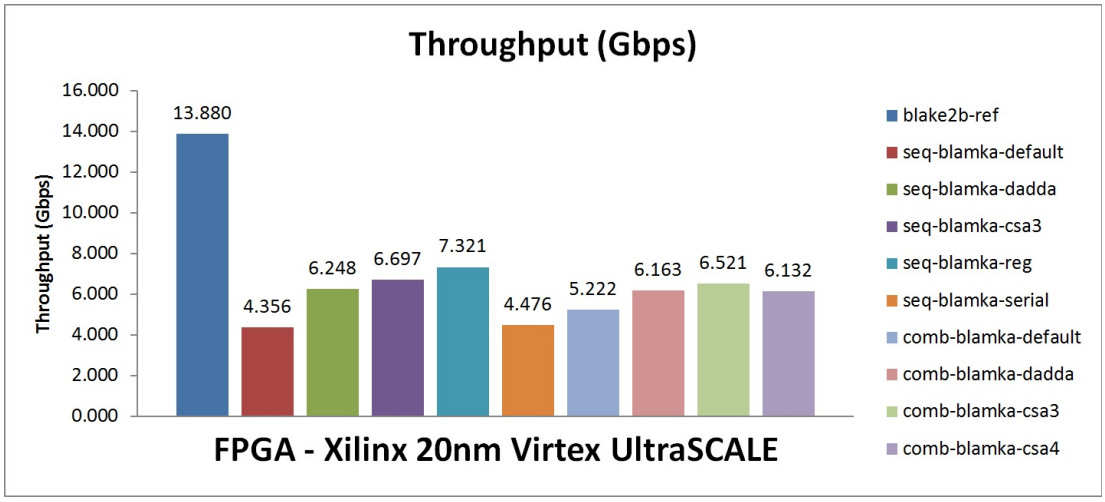

Figure 34: Throughput comparison (FPGA)

For BlaMka, the architecture *seq-blamka-reg* has the highest throughput between the sequential circuitry and the architecture *comb-blamka-csa3* between the combinational versions. In general, the *seq-blamka-reg* showed the highest throughput (7.321 *Gbps*), 12% higher than the best combinational version (6.521 *Gbps*). In addition, it can be noted that combinational and sequential architectures present similar throughput. The lowest overall performance occurred in the *seq-blamka-default* sequential architecture. This circuit is the only sequential circuit that does no use the Dadda multiplier, and this also indicates that synthesis tools for FPGAs were not capable of implementing a default multiplier efficiently in terms of speed. It is also possible to verify that all architectures for BlaMka have lower throughput than the reference architecture. Quantitatively, these differences can be visualized in the graph of Figure 35, which indicates the throughput of each proposed architecture as a fraction of the

throughput of the reference architecture. The architecture with the highest throughput for BlaMka, *seq-blamka-reg*, has a throughput of about 53% than that obtained for the reference architecture. This fact is the first evidence that shows how the inclusion of multiplication operations affected the performance of the BlaMka algorithm in relation to this metric.

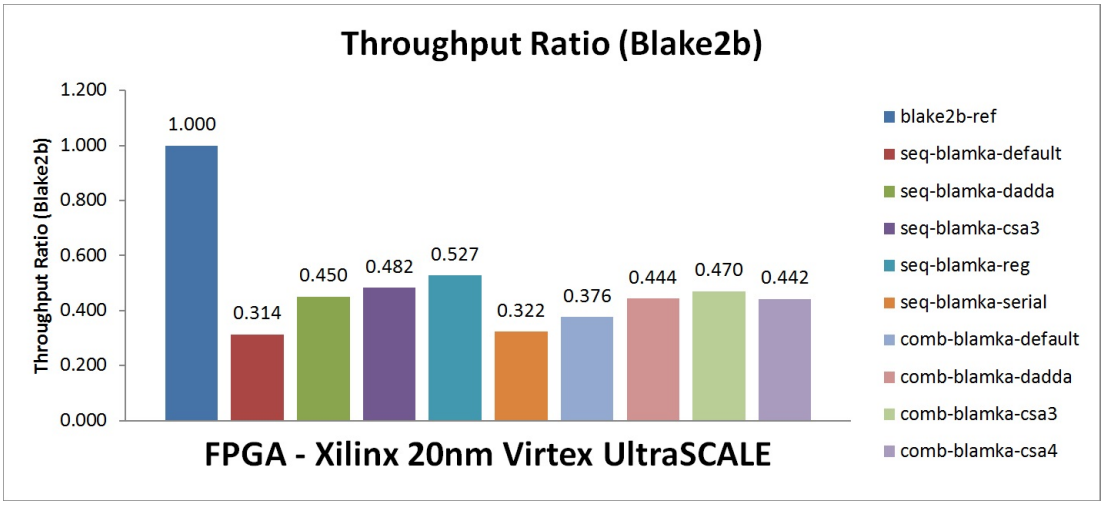

Figure 35: Throughput ratio comparison (FPGA)

Figures 36 and 37 show the performance in relation to the silicon area and power consumption, respectively. In terms of area, the combinational architectures, as expected, presented the highest values. It should also be noted that BlaMka more compact architecture, *seq-blamka-csa3*, still has an area about three times as great as that of the reference architecture, which illustrates the minimal increase in area cost caused by the inclusion of combinational multipliers (for maximizing the throughput). However, it is not the overall minimum increase in area, because it is possible to use sequential multipliers, which are very compact in terms of area but need more clock cycles to execute and consequently reduce the throughput.

In the power graph, we can see the difference between the power consumption of the sequential and combinational circuits. In spite of the greater number of modules used in the combinational versions, especially the multipliers, and also the increase

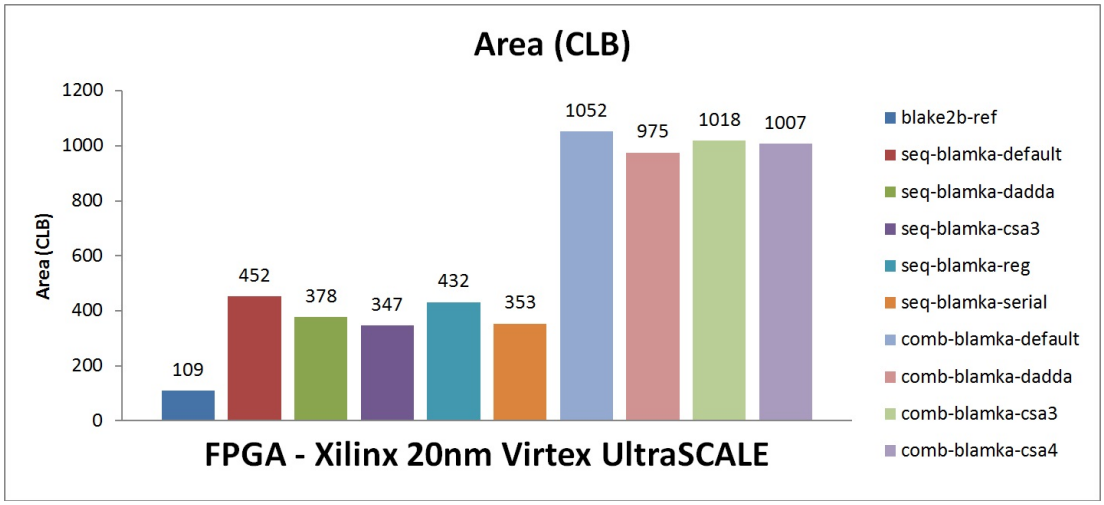

Figure 36: Area comparison (FPGA)

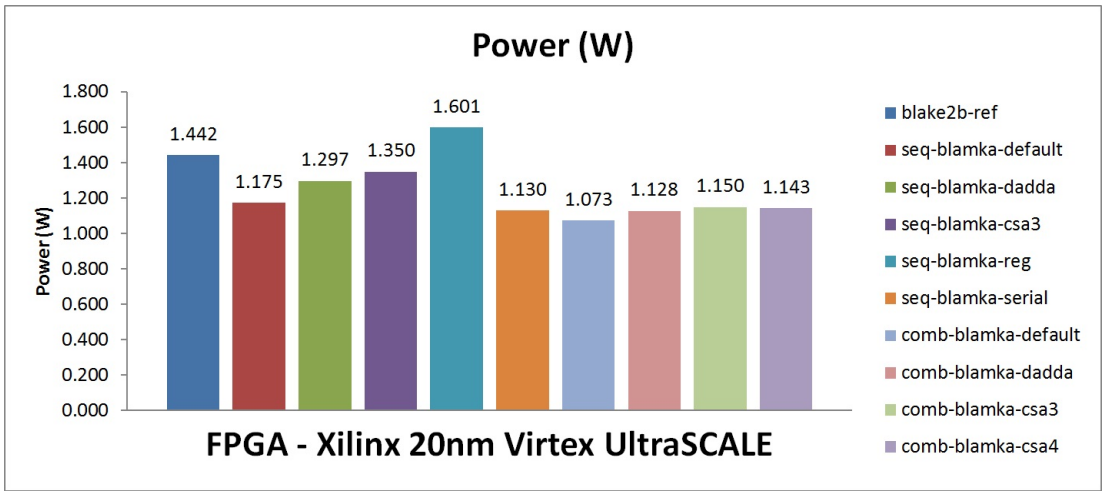

Figure 37: Power comparison (FPGA)

in the number of necessary wire connections, the power consumption for the combinational circuits were smaller if compared to the sequential circuits. This can be explained highlighting that power was estimated assuming that these circuits are the critical path of an external system. Thus, the frequency of operation is the inverse of the combinational circuit latency. These calculated frequencies are smaller than the frequencies obtained for the sequential circuits. For all circuits, frequency had a higher influence in power consumption than the increase in number of components and

wire connections. This can be visualized for all architectures that have smaller power consumption than that of the reference architecture, except for the *seq-blamka-reg*.

Finally, two graphs representing efficiency measures are shown in Figures 38 and 39. In terms of *throughput-to-area*, the architecture *seq-blamka-csa3* has the highest efficiency. However, the reference architecture is much more efficient than all the architectures for BlaMka, almost 7 times as high as the best performance achieved. In terms of energy efficiency, the sequential architectures have a consumption superior to that of the combinational architectures, when performing the same amount of processing. It is worth noting that, despite the proximity in power consumption between the BlaMka architectures and the reference architecture, the latter is about twice as efficient. These efficiency measures again evidence the impact caused by the inclusion of multiplications, mainly in the financial cost for manufacturing and operating the circuits, represented by the area utilization and the power consumption.

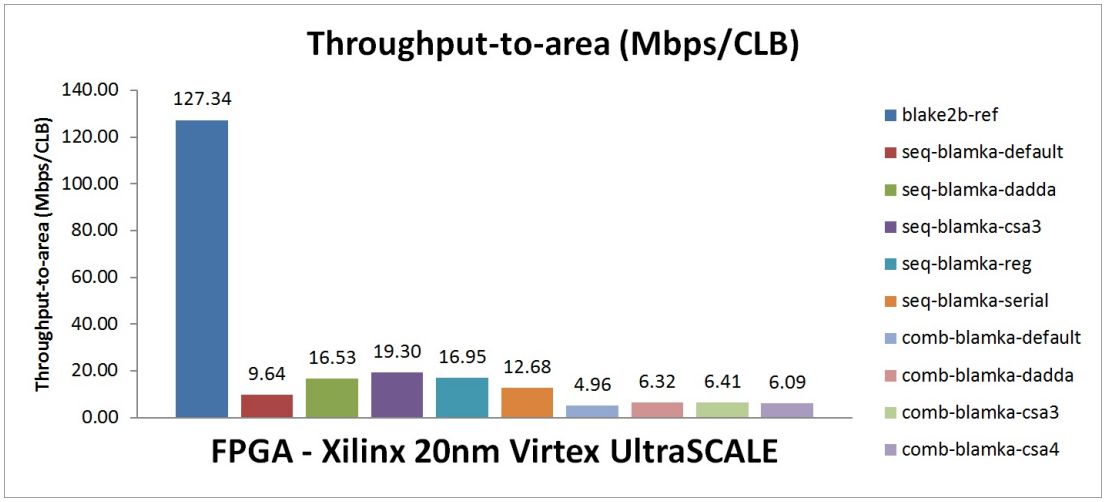

Figure 38: Throughput-to-area comparison (FPGA)

5.2.2 Two-Dimensional Analysis

After the first analysis, observing each metric of interest, a two-dimensional analysis for the trade-offs present in the implementations can be performed. In this work,

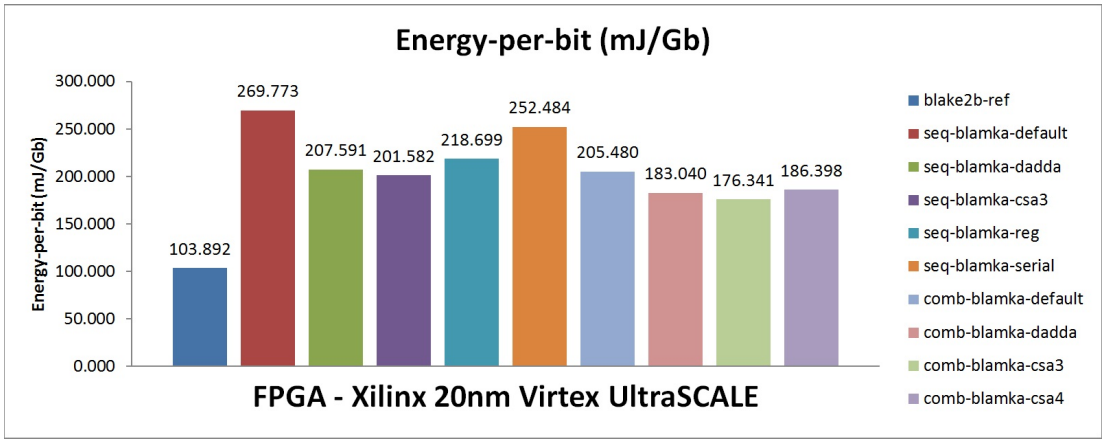

Figure 39: Energy-per-bit comparison (FPGA)

three trade-offs of this type will be analyzed. The first one, between area and throughput, is shown in Figure 40. The trend line presented for this graph as the ones for the other two-dimensional graphs are only approximate and they agree with the number of performance points available. The more points we have, the more accurate that curve will be. However, even roughly, it can help us to understand the behavior of the analyzed trade-off.

This graph shows that to increase the throughput of the system, it is not necessary to increase the area utilization. Sequential circuits can be observed to have similar area utilization, the same applying to the combinational circuits, which allows directly selecting the sequential circuit with the greatest throughput. In addition, we have performance points with similar throughput, but with a significant increase in area; if only this trade-off is observed, it could be concluded these are not relevant options. It can also be observed that if the objective is to obtain the highest possible throughput, the best options are between the sequential versions. Finally, regardless of the approach, the choice within this trade-off can be misleading, because when judging each performance point, we lack the information on the power consumption, another crucial factor in terms of cost. This absence is not only graphical, but also conceptual, as analyzed

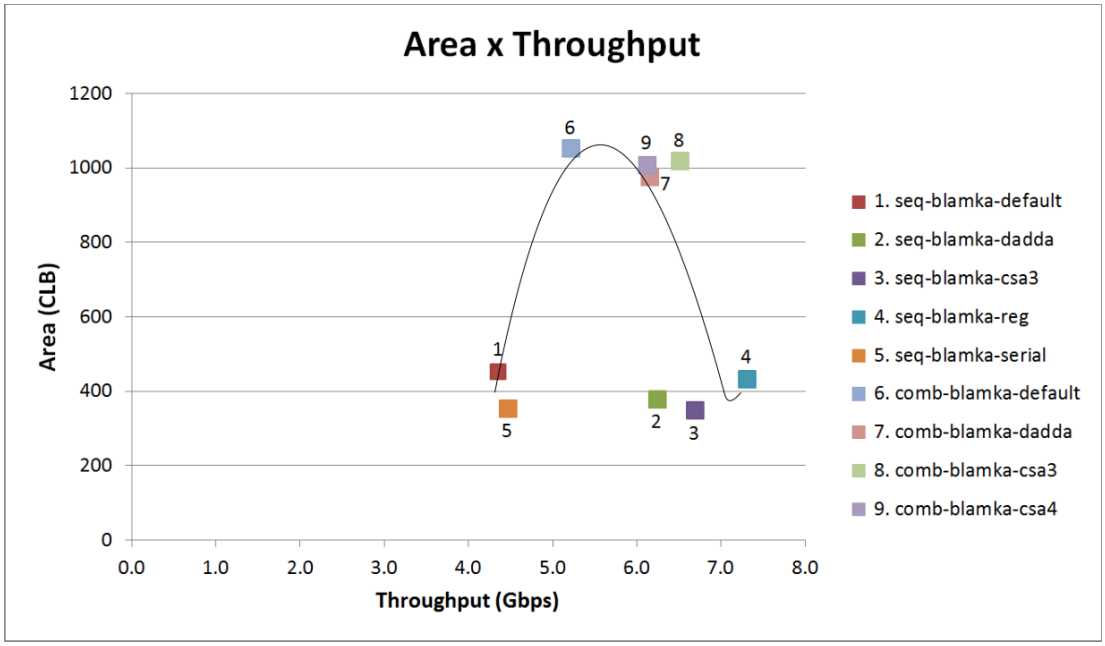

Figure 40: Trade-off between area and throughput (FPGA)

and explained in chapters 1 and 3.

In Figure 41, we have the trade-off between power and throughput. The considerable increase in power is directly verified when the maximum throughput is desired. Some circuits have a lower power consumption but are limited to a throughput range from 4.3 to 6.5 Gbps (point 1 and 5 to 9). If the focus is on mobile devices, it might be worth giving up maximum throughput for a slightly lower throughput, but substantially reducing power consumption, a factor of crucial importance for this type of device, which is usually battery-powered. Of course, it is possible to see how power responds to the trade-off between area and throughput, but this is possible here because of the small number of performance points. If we have dozens of points, this two-dimensional analysis, also observing the response of a third metric, would become complicated. Additionally, we can analyze the trade-off between area and power shown in Figure 42.

More compact versions can be observed to have higher power consumption, while

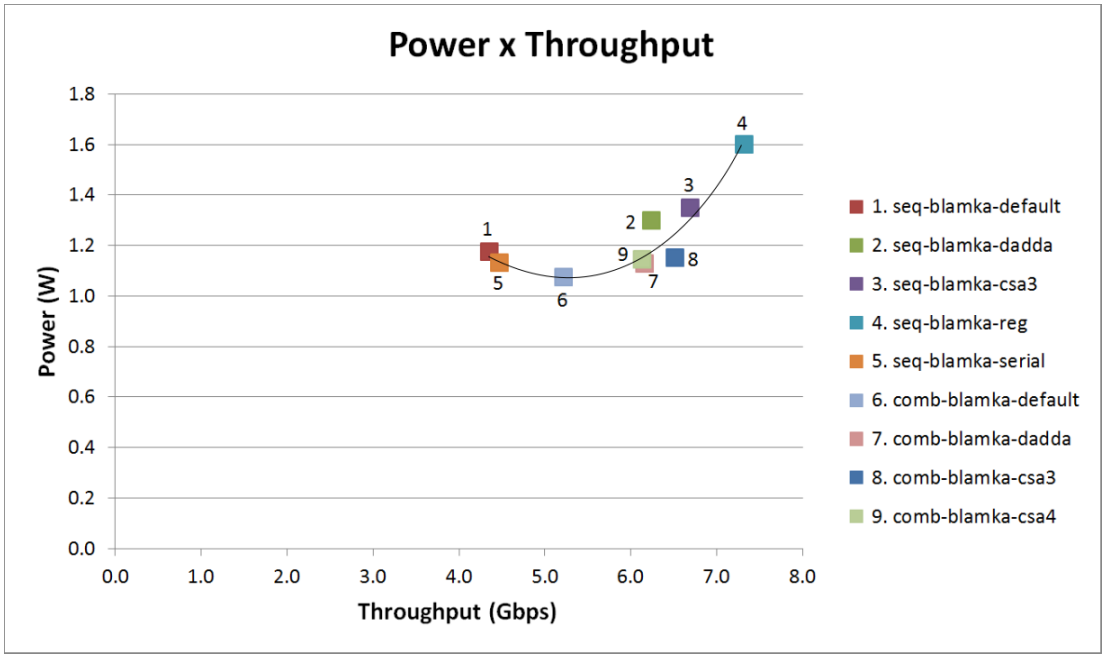

Figure 41: Trade-off between power and throughput (FPGA)

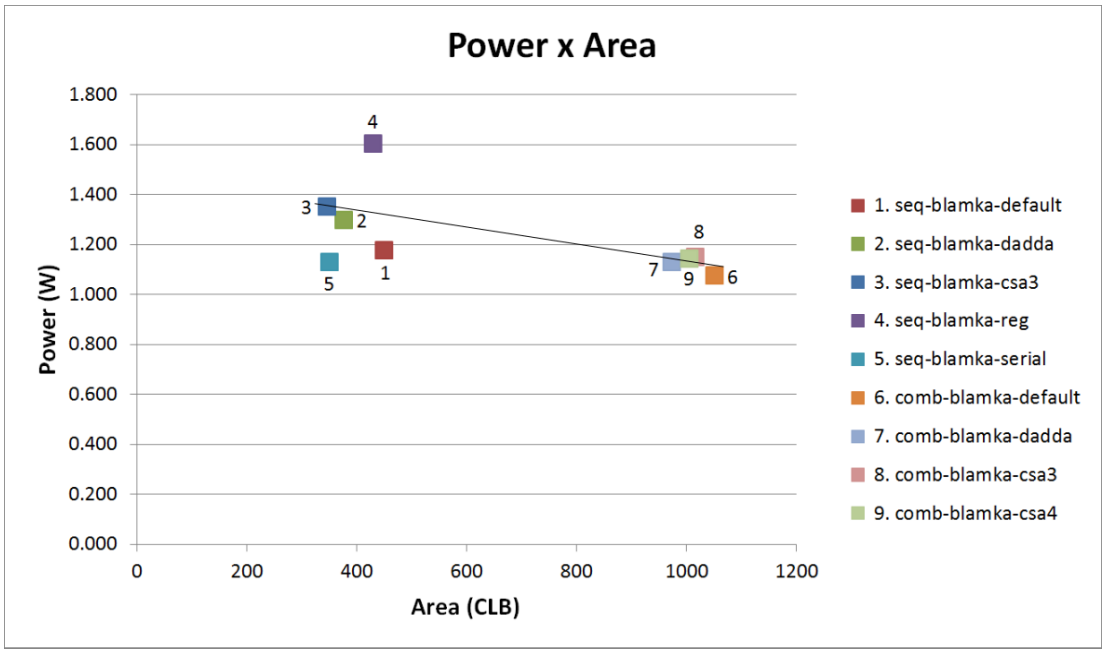

Figure 42: Trade-off between power and area (FPGA)

versions with greater area utilization present lower consumption of power. The reason for this behavior is that combinational circuits, despite having a larger number of

wire connections and functional modules, operate in a smaller frequency. Area and power together form a crucial factor in the financial cost for the design. In addition, the greater the power consumption, the greater the thermal dissipation, which can result in shortening the life of several components if an efficient cooling system is not employed.

5.2.3 Three-Dimensional Analysis

The performance of a circuit in terms of speed has always been a requirement in most projects. Nowadays, with the massive use of devices with restrictions of energy resources and physical size, area and power have also become important factors. From the two-dimensional analyses, two main difficulties in analyzing the behavior of these metrics together can be identified. The first is to observe the response of the third metric when analyzing the trade-off between the other two metrics. The second is the difficulty in obtaining balanced performance points. For example, instead of searching for the maximum optimization of a given metric, it is possible to decrease the performance of this metric in order to obtain better performances for the others. An analogy can be made by treating each metric as a currency, which allows exchanging some coins from one metric to another. To address these difficulties, an analysis of a three-dimensional trade-off among area, power, and throughput is proposed. To perform this analysis, two approaches are presented and can be used together.

5.2.3.1 Projections Approach

Figure 43 presents the first approach, which is to treat each two-dimensional graph as a projection on a three-dimensional graph. As aforementioned, the curves are approximate and the three-dimensional analysis is as well. However, extending the reasoning used in the two-dimensional case, one can imagine each curve as a projection of a surface in space on a plane, roughly representing the relationship among area, power, and

throughput. Designing such a surface is not possible in this case, owing to the finite and small number of available performance points and to the absence of mathematical models (that are not part of the scope of this work) to represent the relationships among the metrics. Moreover, even the estimation would be very arbitrary; we therefore chose only to use the curves as projections.

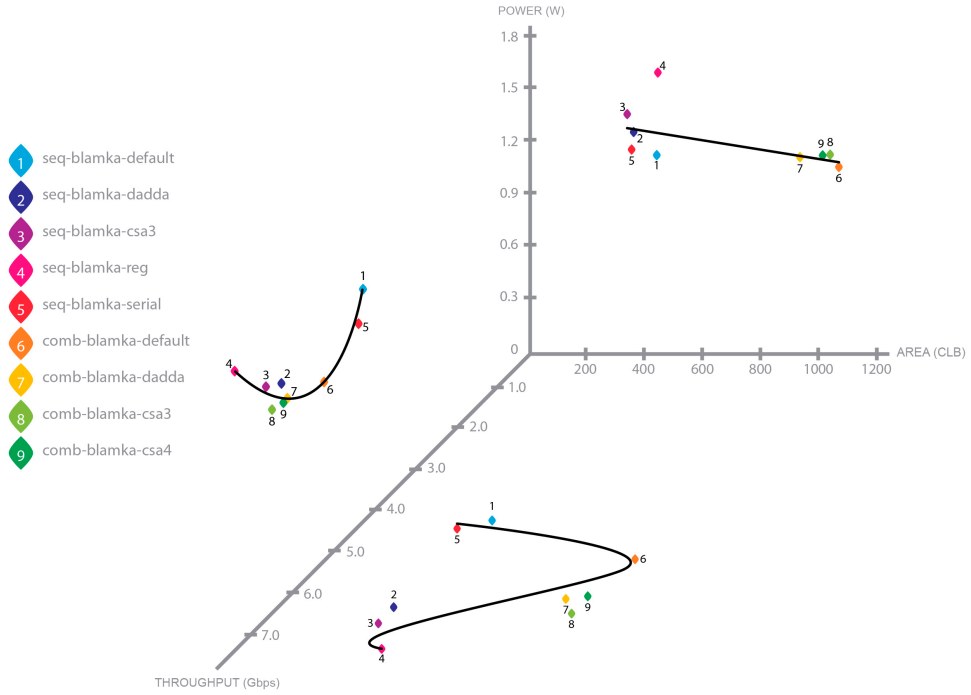

Figure 43: Projections approach (FPGA)

In this graph, each performance point is numbered and has its counterpart in each plan for easy visualization and analysis. Starting from the two-dimensional analyses carried out previously, one can expand them, but now observing the behavior of the three metrics together. For example, we can start at point 1 (Figure 44) and if the goal is to obtain maximum throughput with low power consumption, the *power x throughput* plane shows that point 4 has the best throughput performance (Figure 45). However, by slightly sacrificing the throughput, we can go to point 3 and observing this point in the planes *power x throughput* and *area x throughput*, there is a reduction in power

consumption and area utilization (Figure 46).

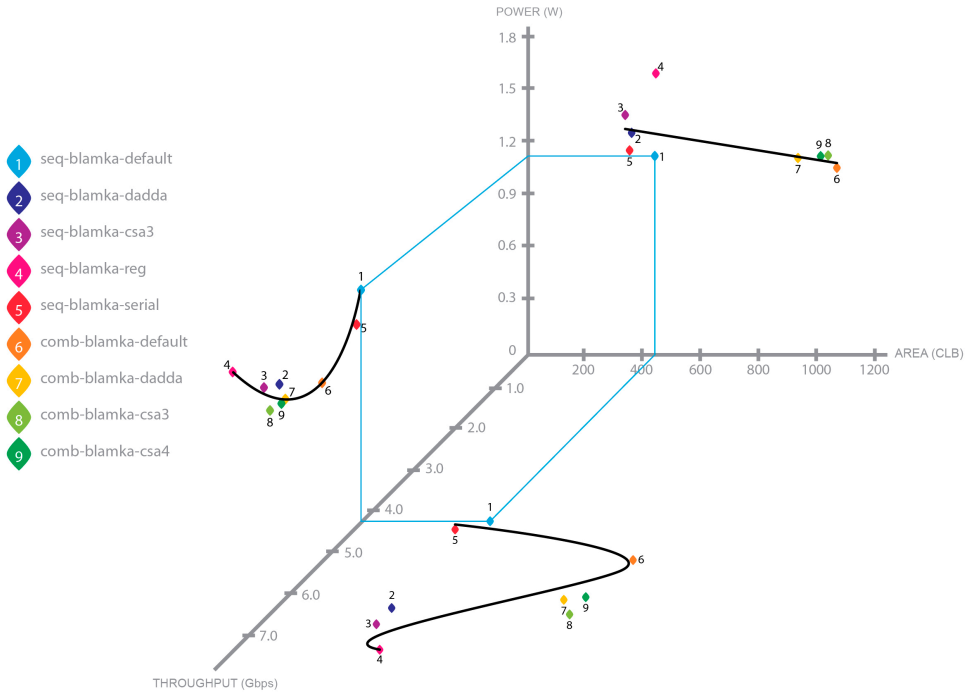

Figure 44: Projections approach - Point 1 (FPGA)

It is possible to verify quantatively these variations using the data of Table 7, representing each point as a triple (**T**hroughput in Gbps, **A**rea in CLB, **P**ower in W). Moving from point 1 to point 4:

$$Point\ 1 : (4.356, 452, 1.175) \Rightarrow Point\ 4 : (7.321, 432, 1.601)$$

$$\Delta T = 7.321 - 4.356 = 2.965\ Gbps \quad [\Delta T\% = 2.965/4.356 = +68.1\%]$$

$$\Delta A = 432 - 452 = -20\ CLB \quad [\Delta A\% = -20/452 = -4.4\%]$$

$$\Delta P = 1.601 - 1.175 = 0.426\ W \quad [\Delta P\% = 0.426/1.175 = +36.3\%]$$

Thus, to increase the throughput in 1 Gbps, it is possible to calculate the response in area and power:

$$1\ Gbps \rightarrow -6.7\ CLB$$

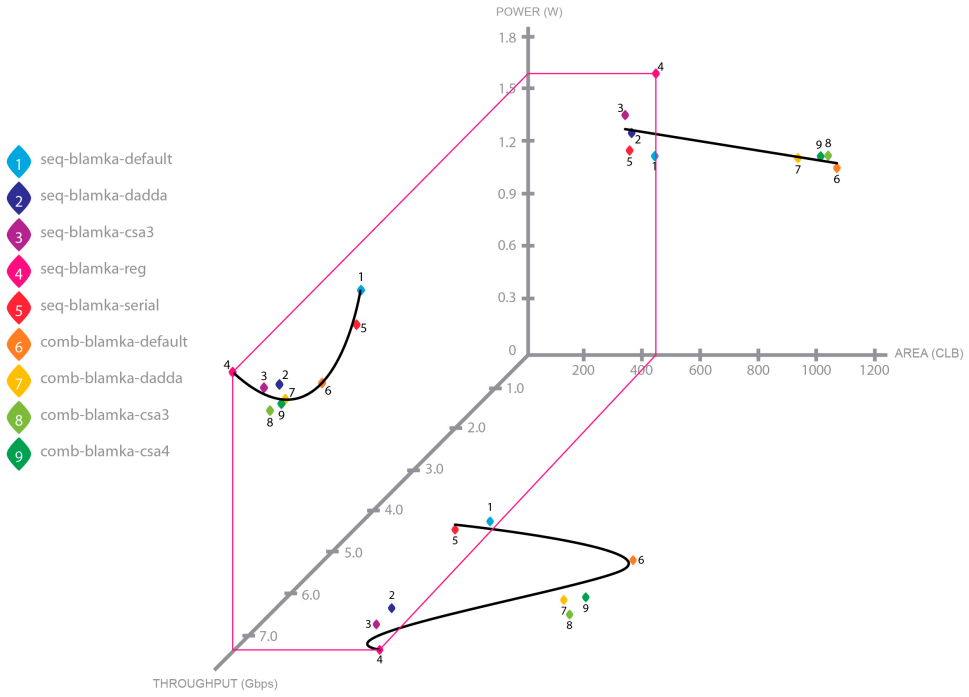

Figure 45: Projections approach - Point 4 (FPGA)

$$1\ Gbps \rightarrow 0.144\ W$$

We can perform the same calculations when moving from point 1 to point 3:

$$Point\ 1 : (4.356, 452, 1.175) \Rightarrow Point\ 3 : (6.697, 347, 1.350)$$

$$\Delta T = 6.697 - 4.356 = 2.341\ Gbps \quad [\Delta T\% = 2.341/4.356 = +53.7\%]$$

$$\Delta A = 347 - 452 = -105\ CLB \quad [\Delta A\% = -105/452 = -23.2\%]$$

$$\Delta P = 1.350 - 1.175 = 0.175\ W \quad [\Delta P\% = 0.175/1.175 = +14.9\%]$$

To increase the throughput in 1 Gbps, the response in area and power is:

$$1\ Gbps \rightarrow -44.8\ CLB$$

$$1\ Gbps \rightarrow 0.075\ W$$

These calculations show that in the case of point 4, it was possible to sacrifice slightly the throughput to reduce significantly the power and the area when moving to

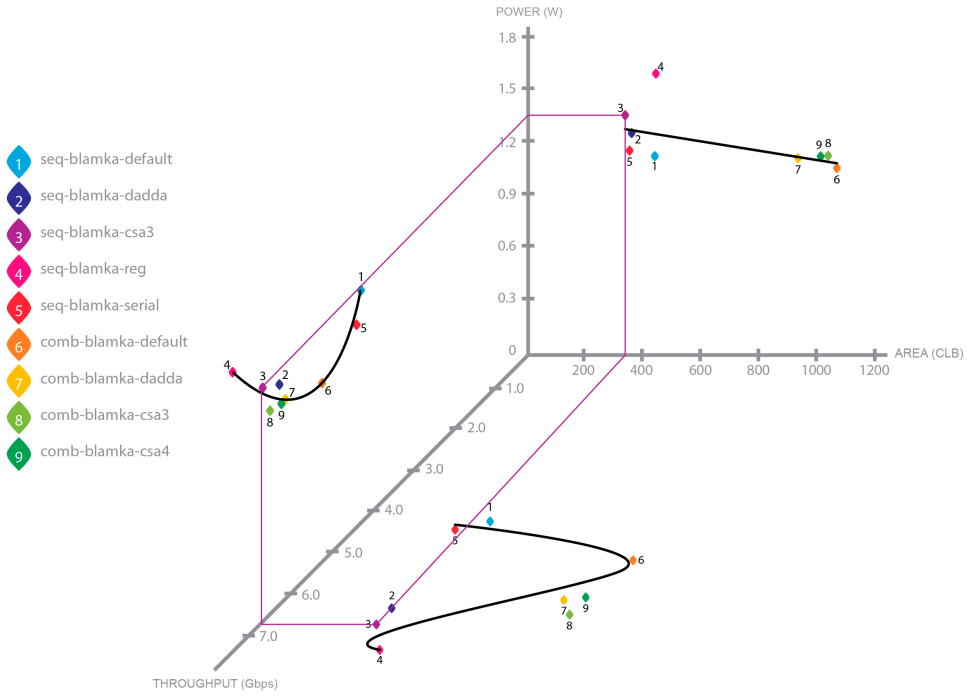

Figure 46: Projections approach - Point 3 (FPGA)

point 3.

On the other hand, if we have power limitation, by looking at the plane *power x area*, the points that have areas close to each other keep this proximity in terms of power. However, by looking at the corresponding points in the *area x throughput* plane, we see that point 5, which represents the architecture *seq-blamka-serial*, has the second lowest throughput. This makes it evident that the strategy used to reduce power actually reduced it in comparison to the other sequential circuits, but such a reduction was largely compensated by the decrease in throughput performance, which may not be an attractive alternative except in very specific applications where energy is an extremely important factor and speed is not.

We can also obtain a balanced performance point among the three metrics. For example, points 1 to 5 have higher power consumption but lower area utilization, and

points 6 to 9 have higher area utilization but lower power consumption. It is possible to see in planes *power x throughput* and *area x throughput* that point 4 has the best overall throughput, but it also has the highest power consumption. By slightly balancing the choice, we can have a small decrease in throughput in exchange for moving to point 3, which has a small area and a considerable reduction in power consumption. It is evident from this representation that the conclusions obtained in the one-dimensional and two-dimensional analyses can be obtained in this three-dimensional analysis, and it also facilitates the three-dimensional understanding of the behavior of these metrics, even if approximate. In addition, it is simpler to find balanced performance points that do not sacrifice one metric in detriment of others, but rather adjust these metrics to each other until values are obtained that are not the maximum values, but which together provide a balanced performance among the points available.

Another variation is to change the throughput axis by its inverse. In this way, all the axes will have the same direction: greater values represent less desirable results (Figure 47). From that, it is possible to calculate a volume corresponding to each point and obtain a balanced performance point in a more direct way. Table 8 presents these volume calculations. It is possible to see that point 3 presents the lowest volume, corresponding to a better performance involving the three metrics together. In addition, if maximum throughput is desired, we can choose the point 4. However, the volume for this point is about 35% higher. Therefore, it is possible to stay in point 3, where we have a slightly decrease in throughput but the lowest volume, constituting a more balanced choice. This was the same result previously obtained but now in a more direct way (Figure 48). The volume establishes a criterion to select optimizations.

5.2.3.2 Planes Approach

The second approach for a three-dimensional analysis is shown in Figure 49. In this graph, performance points are grouped into planes that go through certain values in

Table 8: Volume comparison (FPGA)

Architectures	Volume (CLB*W/Gbps)
seq-blamka-default	121.9
seq-blamka-dadda	78.5
seq-blamka-csa3	69.9
seq-blamka-reg	94.5
seq-blamka-serial	89.1
comb-blamka-default	216.2
comb-blamka-dadda	178.5
comb-blamka-csa3	179.6
comb-blamka-csa4	187.7

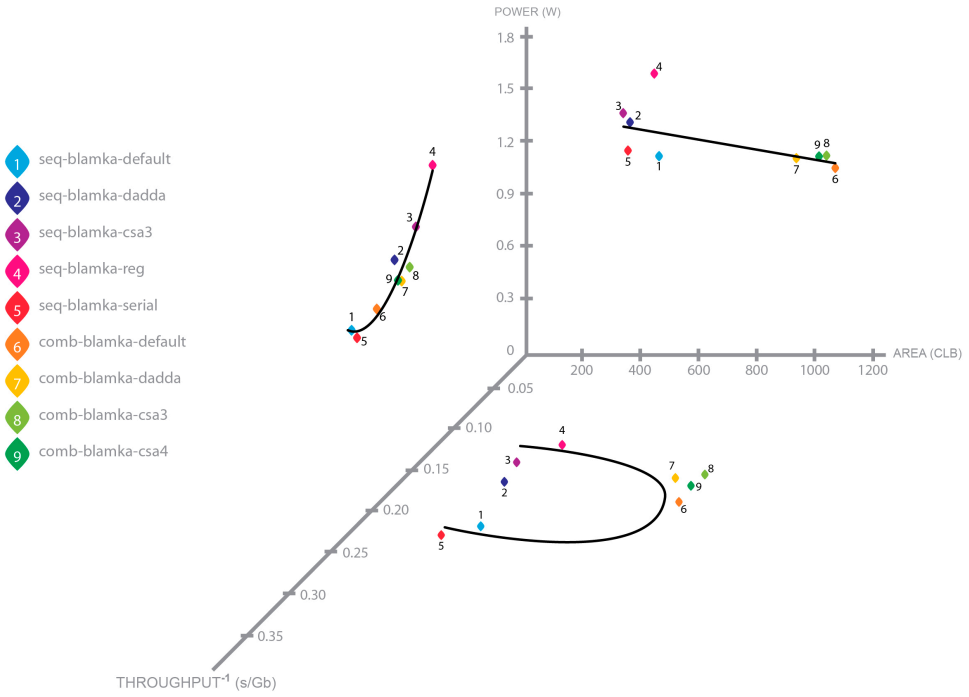

Figure 47: Projections approach - Inverse of throughput (FPGA)

one axis and allow us to analyze performance variations as we move from one group to another. This approach is based on theoretical models presented in (DEMICHELI, 1994). Using the throughput as a reference, four groups were formed: points with throughput near 4.3 Gbps, points with throughput near 5.3 Gbps, points with throughput near 6.3

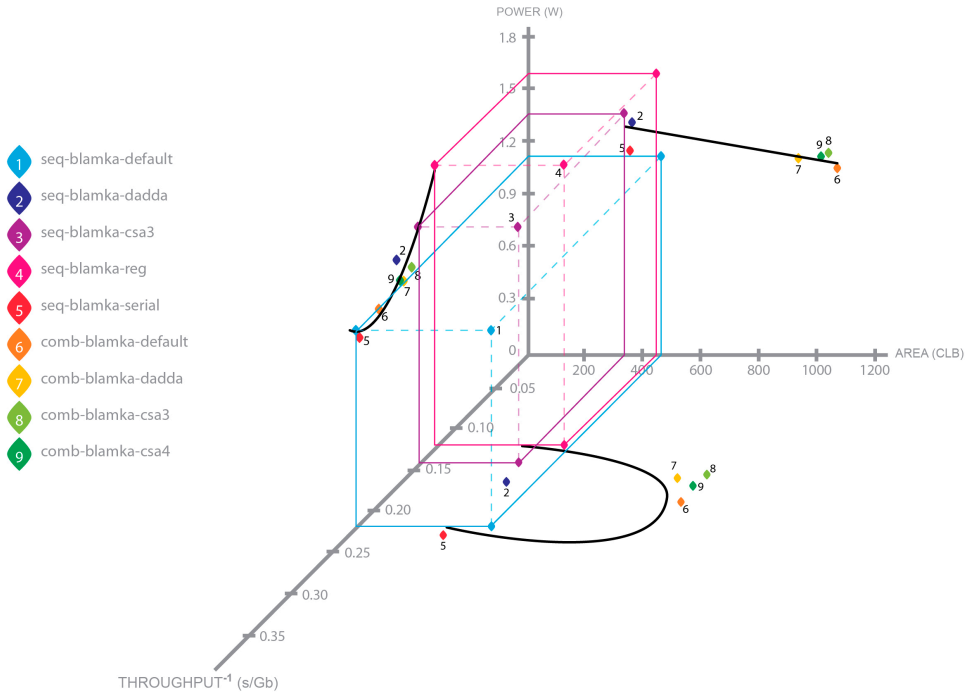

Figure 48: Projections approach - Volume (FPGA)

Gbps, and points with throughput near 7.3 Gbps. These groups are approximate and each plane contains performance points where throughput is around and close to the throughput value through which the plane passes, but are not exactly equal to this value. From the purple plane, one can have a 23% increase in throughput by going to the orange plane. This gain would be compensated by an increase in area and a decrease in power, represented by point 6. However, we have points 2, 3, and 7 to 9 in the green plane that have similar power consumption, but some of them even present a decrease in area, although all these points have a 19% increase in throughput. From the green plane, approximately another 16% increase in throughput can be obtained by going to the yellow plane. This division into planes also allows finding points with a balanced performance between the metrics. As can be seen, points 2 and 3 do not have the highest overall values for throughput, but they do not have the minimum values, either. In addition, they have lower power consumption compared to the point in the

plane with the maximum throughput.

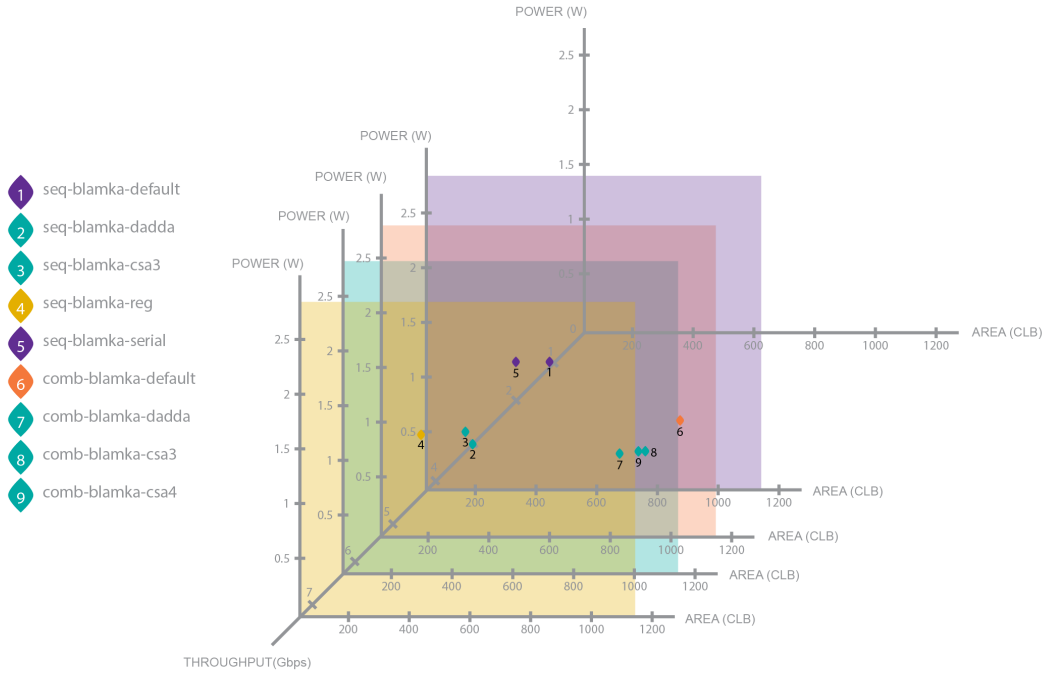

Figure 49: Planes approach (FPGA)

It is possible to verify quantatively these variations using the data of Table 7. The throughput interval in each plane can be obtained and an average throughput can be calculated:

$$Purple\ Plane\ : [4.356, 4.476] \rightarrow T_{\text{average}} = 4.416\ Gbps$$

$$Orange\ Plane\ : [5.222] \rightarrow T_{\text{average}} = 5.222\ Gbps$$

$$Green\ Plane\ : [6.132, 6.697] \rightarrow T_{\text{average}} = 6.415\ Gbps$$

$$Yellow\ Plane\ : [7.321] \rightarrow T_{\text{average}} = 7.321\ Gbps$$

With the throughput established in some interval, the remaining metrics can be analyzed. For example, starting with point 3 in the green plane and going to the yellow plane, we obtain the point 4 that presents the highest throughput. From the previous

analyses it was concluded that point 3 presents the most balanced performance, but it is possible to have an increase of about 16% if we move to the yellow plane. Representing each point as (**A**rea in CLB, **P**ower in W), it is possible to calculate the response in terms of area and power:

$$\textit{Green Plane} \Rightarrow \textit{Yellow Plane} : \uparrow 16\% \textit{ throughput on average}$$

$$\textit{Point } 3 : (347, 1.350) \Rightarrow \textit{Point } 4 : (432, 1.601)$$

$$\Delta A = 432 - 347 = 85\ CLB \quad [\Delta A\% = 85/347 = +24.5\%]$$

$$\Delta P = 1.601 - 1.350 = 0.251\ W \quad [\Delta P\% = 0.251/1.350 = +18.6\%]$$

6 CONCLUSIONS

We presented hardware implementations for the permutation function of the BlaMka algorithm. Combinational and sequential circuits were synthesized covering a range of possible choices taking into account throughput, area, and power. All circuits were synthesized for ASICs, with results for throughput ranging from 0.907 Gbps to 2.426 Gbps, area utilization ranging from 14.13 kGE to 52.82 kGE, and power consumption ranging from 0.384 mW to 2.104 mW. All circuits were also synthesized for FPGAs, with results for throughput ranging from 4.356 Gbps to 7.321 Gbps, area utilization ranging from 347 CLB to 1052 CLB, and power consumption ranging from 1.073 W to 1.601 W. The results were compared to an implementation of the permutation function of Blake2b, showing that the performance decrease is quite marked when multiplications are included. Furthermore, it was verified the difficulty in using parallelism at the algorithm level, confirming the strictly sequential nature of the algorithm. It was also verified that, in fact, the inclusion of multiplications significantly reduces the performance of the permutation function, as theoretically stated in (SIMPLICIO et al., 2015). This was the desired result, since for Lyra2 password hashing scheme it is interesting to have a cryptographic sponge that is slower in order to difficult brute-force attacks. With the synthesis results, a detailed performance analysis was performed for each platform, starting from a one-dimensional analysis, going through a two-dimensional analysis, and culminating in a three-dimensional analysis. Two techniques were presented for such analysis, namely projections approach and planes approach. Although there is room for improvement, the proposed method is a

initial step showing that, in fact, a trade-off between three metrics can be analyzed, and that it is also possible to find balanced performance points. From the two approaches presented, it was possible to derive a criterion to select optimizations when we have restrictions, such as a desired throughput range or a maximum physical size, and when we do not have restrictions, in which case we can choose the optimization with the most balanced performance.

Figure 50 shows a comparison with the most relevant works analyzed in Chapter 3. The comparison involves the presence of results for three different metrics that cannot be derived from each other, implementation results for FPGA and ASIC, the three types of analyses done in this work, and the criterion to select optimizations.

Literature / Results	[Sunny; Saranya, 2014]	[Aumasson et al., 2014]	[Rao; Newe; Grout, 2014]	[Bertoni et al., 2012]	[Zhang et al., 2014]	[Mikami et al., 2010]	[Guo et al., 2010]	This work (2017)
Latency/ Throughput	X	X	X	X	X	X	X	X
Area	X	X		X	X	X	X	X
Power/ Energy			X		X		X	X
FPGA			X	X			X	X
ASIC	X	X			X	X	X	X
1D Analysis	X	X	X	X	X	X	X	X
2D Analysis		X		X	X	X		X
3D Analysis								X
Criterion to select optimizations								X

Figure 50: Literature comparison

6.1 Research Contributions and Publications

The contributions can be divided into two groups:

(I) the hardware implementation for the permutation function of BlaMka cryptographic sponge, synthesized for both ASIC and FPGA. Based on the literature review, this is the first implementation of this algorithm in this platform. In addition, the analysis of the performance of this implementation compared to the implementation of Blake2b permutation function, from which the BlaMka algorithm was designed, as well as the experimental verification of some of its theoretical properties (parallelism difficulty at algorithm-level, performance reduction due to the inclusion of multiplications) were performed.

(II) the performance analysis of a hardware implementation by jointly analyzing the trade-offs associated with three different metrics, which was named a three-dimensional analysis, was performed. Two techniques were presented for such analysis, namely projections approach and planes approach. It was also presented a criterion to select optimizations when we have restrictions in some metrics and when we do not have restrictions.

As a partial result of all the work, a paper was developed and published in LASCAS 2017 (8th IEEE Latin American Symposium on Circuits and Systems) conference entitled "Hardware Implementation for Permutation Function of Multiplication-Hardened Sponge BlaMka", where the hardware design and implementation for the BlaMka permutation function was presented, with several architectures proposed and analyzed, as well as the comparison of its performance with the algorithm from which it originated. In addition, the experimental verifications of the theoretical statements as mentioned in the group of contributions (I) were carried out. A second paper will be developed furthering the analysis of performance to contemplate what was discussed in the group of contributions (II), aiming a publication at an international journal.

6.2 Future Work

As future work, we intend to design and to implement the complete BlaMka algorithm, as well as to analyze its performance when used in password hashing schemes, applying the three-dimensional analysis to these schemes. We also intend to expand the three-dimensional analysis to other types of algorithms, for example algorithms that allow parallelism. Finally, we intend to keep improving the ideas behind the three-dimensional analysis in order to obtain more accurate results, and perhaps more formal relations among the metrics involved.

REFERENCES

AHMED, A. B.; ABDALLAH, A. B. Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3D-Network-on-Chip (3D-NoC). *The Journal of Supercomputing*, Springer, v. 66, n. 03, p. 1507–1532, 2013.

ALPHA TECHNOLOGY. *Scrypt ASIC prototyping preliminary design document*. 2013. [Online]. Available: https://alpha-t.net/wp-content/uploads/2013/12/Scrypt_ASIC_Prototyping_Design_Document.pdf.

ALTERA CORPORATION. *AN 311: Standard cell ASIC to FPGA design: methodology and guidelines*. 2009. [Online]. Available: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/an/an311.pdf.

ANDRADE, E. R. et al. Lyra2: Efficient password hashing with high security against time-memory trade-offs. *IEEE Transactions on Computers*, v. 65, n. 10, p. 3096–3108, 2016.

AUMASSON, J.-P. et al. BLAKE2: simpler, smaller, fast as MD5. *Applied Cryptography and Network Security, 11th International Conference, ACNS 2013. Proceedings*, Springer Berlin Heidelberg, p. 119–135, 2013.

AUMASSON, J.-P. et al. *The Hash Function BLAKE*. [S.l.]: Springer Berlin Heidelberg, 2014.

BEDNARA, M. et al. *Reconfigurable implementation of elliptic curve crypto algorithms*. 2002. IEEE International Parallel & Distributed Processing Symposium. [Online]. Available: https://www12.informatik.uni-erlangen.de/publications/pub2002/raw2002.pdf.

BERTONI, G. et al. *Cryptographic sponge functions*. 2011. [Online]. Available: http://sponge.noekeon.org/CSF-0.1.pdf.

BERTONI, G. et al. *Keccak implementation overview*. 2012. [Online]. Available: http://keccak.noekeon.org/Keccak-implementation-3.2.pdf.

BIRYUKOV, A.; DINU, D.; KHOVRATOVICH, D. Argon2: new generation of memory-hard functions for password hashing and other applications. *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, p. 292–302, 2016.

DADDA, L. Some schemes for parallel multipliers. *Alta Frequenza*, v. 34, p. 349–356, 1965.

DADDA, L. On parallel digital multipliers. *Alta Frequenza*, v. 45, p. 574–580, 1976.

DAPP, T. F. *Growing need for security in online banking*. 2012. Deutsche Bank - DB Research. [Online]. Available: http://www.dbresearch.de/servlet/reweb2.ReWEB?rwsite=DBR_INTERNET_DE-PROD.

DEMICHELI, G. *Synthesis and optimization of digital circuits*. [S.l.]: McGraw-Hill, 1994.

DENG, L. et al. Accurate area, time and power models for FPGA-based implementations. *Journal of Signal Processing Systems*, Springer, v. 63, n. 01, p. 39–50, 2011.

DESCHAMPS, J.-P.; BIOUL, G. J. A.; SUTTER, G. D. *Synthesis of arithmetic circuits: FPGA, ASIC and embedded systems*. [S.l.]: John Wiley & Sons, 2006.

DESCHAMPS, J.-P.; SUTTER, G. D.; CANTO, E. *Guide to FPGA implementation of arithmetic functions*. [S.l.]: Springer, 2012.

DETREY, J.; GAUDRY, P.; KHALFALLAH, K. A low-area yet performant FPGA implementation of Shabal. *Selected Areas in Cryptography*, Springer, p. 99–113, 2011.

DIMENSIONAL RESEARCH. *The impact of mobile devices on information security: a survey of IT professionals*. 2013. Dimensional Research. [Online]. Available: https://www.checkpoint.com/downloads/products/check-point-mobile-security-survey-report.pdf.

FEDERAL RESERVE BOARD. *Consumers and mobile financial services 2016*. 2016. Board of Governors of the Federal Reserve System. [Online]. Available: http://www.federalreserve.gov/econresdata/consumers-and-mobile-financial-services-report-201603.pdf.

FLORENCIO, D.; HERLEY, C. A large scale study of web password habits. *Proc. of the 16th International Conference on World Wide Web*, Alberta, Canada, p. 657–666, 2007.

GUO, J. et al. *Analysis of BLAKE2*. 2014. CT-RSA.

GUO, X. et al. *On the impact of target technology in SHA-3 hardware benchmark rankings*. 2010. Cryptology ePrint Archive, Report 2010/536. [Online]. Available: http://eprint.iacr.org/2010/536.pdf.

GUO, X. et al. *Silicon implementation of SHA-3 finalists: BLAKE, Grøstl, JH, Keccak and Skein*. 2011. ECRYPT II Hash Workshop. [Online]. Available: http://rijndael.ece.vt.edu/schaum/papers/2011dsd.pdf.

HÄMÄLÄINEN, P. Design and implementation of low-area and low-power AES encryption hardware core. *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, IEEE, p. 577–583, 2006.

HAVINGA, P. J. M.; SMIT, G. J. M. Design techniques for low-power systems. *Journal of Systems Architecture*, Elsevier, v. 46, n. 01, p. 1–21, 2000.

HORTA, E. L. *Dispositivos lógicos programáveis: implementação de sistemas digitais em FPGAs*. São Paulo, Brasil: Editora Mackenzie, 2013.

INIEWSKI, K. *Embedded Systems - Hardware, Design and Implementation*. United States: Wiley, 2012.

IVEY, B. *Low-Power Design Guide*. 2011. Microchip Technology Inc. Application Note AN1416. [Online]. Available: http://www.microchip.com/LowPowerDesignGuide4212272.

JEAN-JACQUES, Q. et al. A cryptanalytic time-memory tradeoff: first FPGA implementation. *Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream*, Springer, p. 780–789, 2002.

JUNGK, B.; STOTTINGER, M.; HARTER, M. *Shrinking Keccak hardware implementations*. 2014. NIST. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/sha-3/Aug2014/documents/jungk_paper_sha3_2014_workshop.pdf.

KALISKI, B. *PKCS #5: password-based cryptography specification version 2.0*. United States: RFC Editor, 2000.

KELSEY, J. et al. Secure applications of low-entropy keys. *ISW '97 Proceedings of the First International Workshop on Information Security*, Springer-Verlag, London, UK, p. 121–134, 1998.

KHOSE, P. N.; RAUT, V. G. Hardware implementation of AES encryption and decryption for low area & power consumption. *International Journal of Research in Engineering and Technology*, eSAT Publishing House, Bangalore, India, v. 3, n. 05, p. 480–484, 2014.

KUMAR, M. J.; CHITRAVALAVAN. Implementation of Blake algorithm using pipelining in FPGA. *International Journal of Innovations in Scientific and Engineering Research*, IJISER, v. 1, n. 11, p. 79–82, 2014.

KUON, I.; ROSE, J. Measuring the gap between FPGAs and ASICs. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, IEEE, v. 26, n. 02, p. 203–215, 2007.

LATIF, K. et al. High throughput hardware implementation of secure hash algorithm (SHA-3) finalist: Blake. *International Journal of Academic Research*, v. 3, n. 06, p. 189–194, 2011.

LATIF, K. et al. *Efficient hardware implementations and hardware performance evaluation of SHA-3 finalists*. 2012. NIST. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/March2012/documents/papers/KASHIF_paper.pdf.

LOOKOUT. *2014 mobile threat report*. 2014. Lookout. [Online]. Available: https://www.lookout.com/img/images/Consumer_Threat_Report_Final_ENGLISH_1.14.pdf.

MALVONI, K.; KNEZOVIC, J. *Are your passwords safe: energy-efficient bcrypt cracking with low-cost parallel hardware*. 2014. USENIX Workshop on Offensive Technologies. [Online]. Available: https://www.usenix.org/system/files/conference/woot14/woot14-malvoni.pdf.

MARKOVIC, D.; NIKOLIC, B.; BRODERSEN, R. W. Power and area efficient VLSI architectures for communication signal processing. *Communications, 2006. ICC'06. IEEE International Conference on*, IEEE, v. 7, p. 3223–3228, 2006.

MCLOONE, M.; MCIVOR, C. High-speed & low area hardware architectures of the Whirlpool hash function. *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, Kluwer Academic Publishers-Plenum Publishers, v. 47, n. 01, p. 47–57, 2007.

MERKLE, R. C. *Secrecy, authentication and public key systems*. Tese (Ph.D. Thesis) — Department of Electrical Engineering, Stanford University, Stanford, USA, 1979.

MIKAMI, S. et al. *A compact hardware implementation of SHA-3 candidate Luffa*. 2010. Systems Development Laboratory, Hitachi, Ltd. [Online]. Available: http://www.hitachi.com/rd/yrl/crypto/luffa/.

NACHENBERG, C. *A window into mobile device security*. 2011. Symantec Corporation. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/white_papers/b-mobile-device-security_WP.en-us.pdf.

NIST. *Special publication 800-108 - recommendation for key derivation using pseudorandom functions*. 2009. National Institute of Standards and Technology, U.S. Department of Commerce. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf.

NIST. *Special publication 800-132 - recommendation for password-based key derivation*. 2010. National Institute of Standards and Technology, U.S. Department of Commerce. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf.

NIST. *Special publication 800-135-1 - recommendation for existing application-specific key derivation functions*. 2011. National Institute of Standards and Technology, U.S. Department of Commerce. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-135-rev1/sp800-135-rev1.pdf.

PERCIVAL, C. *Stronger key derivation via sequential memory-hard functions*. 2009. BSDCan 2009 - The Technical BSD Conference.

PEREIRA, F. D. et al. Exploiting parallelism on Keccak: FPGA and GPU comparison. *Parallel & Cloud Computing*, American V-King Scientific Publishing, v. 2, n. 01, p. 1–6, 2013.

PHC. *Password hashing competition*. 2013. [Online]. Available: https://password-hashing.net/.

POSCHMANN, A. Y. *Lightweight cryptography: cryptographic engineering for a pervasive world*. Tese (Ph.D. Thesis) — Faculty of Electrical Engineering and Information Technology, Ruhr-University, 2009.

PROVELENGIOS, G. et al. FPGA-based design approaches of Keccak hash function. *Digital System Design (DSD), 2012 15th Euromicro Conference on*, IEEE, p. 648–653, 2012.

PROVOS, N.; MAZIÈRES, D. *A future-adaptable password scheme*. 1999. Proceedings of 1999 USENIX Annual Technical Conference.

RAO, M.; NEWE, T.; GROUT, I. Secure hash algorithm-3 (SHA-3) implementation on Xilinx FPGAs, suitable for IoT applications. *Proceedings of the 8th International Conference on Sensing Technology*, ICST, Liverpool, UK, p. 352–357, 2014.

ROMPAY, B. V. *Analysis and design of cryptographic hash functions, MAC algorithms and block ciphers*. Tese (Ph.D. Thesis) — Electrical Engineering Department, Katholieke Universiteit Leuven, Belgium, 2004.

SAN, I.; AT, N. Compact Keccak hardware architecture for data integrity and authentication on FPGAs. *Information Security Journal: A Global Perspective*, Taylor & Francis, v. 21, n. 5, p. 231–242, 2012.

SARAVANAKUMAR, U.; RANGARAJAN, R.; RAJASEKAR, K. Hardware implementation of pipeline based router design for on-chip network. *Journal on Communication Technology*, ICTACT, v. 3, n. 04, p. 646–650, 2012.

SIMPLICIO, M. A. et al. *The Lyra2 reference guide version 3.0*. 2015. Password Hashing Competition. [Online]. Available: https://password-hashing.net/submissions/specs/Lyra2-v3.pdf.

SOBTI, R.; GEETHA, G. Cryptographic hash functions: a review. *IJCSI International Journal of Computer Science Issues*, IJCSI Publication, v. 9, n. 02, p. 461–479, 2012.

SUNNY, G.; SARANYA, C. Cryptography based on hash function Blake-32 in VLSI. *International Journal of Computer Science and Network Security (IJCSNS)*, IJCSNS, v. 14, n. 01, p. 117–123, 2014.

SUTTER, G.; DESCHAMPS, J.-P.; BOEMO, E. *Area-time-power of modular multipliers implemented in FPGA*. 2004. [Online]. Available: http://arantxa.ii.uam.es/~ivan/02004-jcra04-mod-multipliers.pdf.

SVAJCER, V. *Sophos mobile security threat report*. 2014. Sophos. [Online]. Available: https://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.pdf?la=en.

SYNOPSYS INC. *DC Ultra*. 2015. [Online]. Available: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCUltra/Pages/default.aspx.

SYNOPSYS INC. *Formality and Formality Ultra*. 2015. [Online]. Available: http://www.synopsys.com/Tools/Verification/FormalEquivalence/Pages/Formality.aspx.

SYNOPSYS INC. *VCS*. 2015. [Online]. Available: http://www.synopsys.com/Tools/Verification/FunctionalVerification/Pages/VCS.aspx.

TANOUGAST, C. et al. *Embedded systems - high performance systems, applications and projects*. Rijeka, Croatia: InTech Europe, 2012.

TEHRANIPOOR, M.; WANG, C. *Introduction to hardware security and trust*. [S.l.]: Springer Science, 2011.

TILLICH, S. et al. *Compact hardware implementations of the SHA-3 candidates ARIRANG, BLAKE, Grøstl, and Skein*. 2009. Cryptology ePrint Archive, Report 2009/349. [Online]. Available: https://eprint.iacr.org/2009/349.pdf.

TILLICH, S. et al. *Uniform evaluation of hardware implementations of the round-two SHA-3 candidates*. 2010. The Second SHA-3 Candidate Conference. [Online]. Available: http://www.cs.bris.ac.uk/publications/papers/2001240.pdf.

WALKER, J. et al. *A Skein-512 hardware implementation*. 2010. NIST. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/WALKER_skein-intel-hwd.pdf.

WENGER, E.; FELDHOFER, M.; FELBER, N. Low-resource hardware design of an elliptic curve processor for contactless devices. *Information Security Applications*, Springer, p. 92–106, 2011.

WIEMER, F.; ZIMMERMANN, R. High-speed implementation of bcrypt password search using special-purpose hardware. *ReConFigurable Computing and FPGAs (ReConFig), 2014 International Conference on*, IEEE, p. 1–6, 2014.

XILINX INC. *ISE Design Suite 14*. 2013. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/irn.pdf.

XILINX INC. *FPGA vs. ASIC*. 2015. [Online]. Available: http://www.xilinx.com/fpga/asic.htm.

XILINX INC. *Partial reconfiguration in the Vivado Design Suite*. 2016. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado/implementation/partial-reconfiguration.html.

XILINX INC. *Vivado Design Suite*. 2016. [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html.

YALLA, P.; HOMSIRIKAMOL, E.; KAPS, J.-P. *Multi-purpose Keccak for modern FPGAs*. 2014. Directions in Authenticated Ciphers DIAC 2014. [Online]. Available: http://2014.diac.cr.yp.to/slides/kaps-keccak.pdf.

ZHANG, Y. et al. *High performance and low power hardware implementation for cryptographic hash functions*. 2014. International Journal of Distributed Sensor Networks - Hindawi Publishing Corporation. [Online]. Available: http://www.hindawi.com/journals/ijdsn/2014/736312/cta/.

ZHOU, L.; HUANG, M.; SMITH, S. C. *High-performance and area-efficient hardware design for radix-2^k Montgomery multipliers*. 2011. Proc. of International Conference on Computer Design. [Online]. Available: http://www.csce.uark.edu/~mqhuang/papers/2011_MM_CDES.pdf.