Fair Comparison of ASIC Performance for SHA-3 Finalists

Yongbo Zuo

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Leyla Nazhandali, Chair
Patrick Robert Schaumont
Sandeep K Shukla

May 31, 2012
Blacksburg, Virginia

Keywords: SHA, NIST, ASIC, Finalist, Hardware, Encryption, Hash, Cipher, Key

Fair Comparison of ASIC Performance for SHA-3 Finalists

Yongbo Zuo

(ABSTRACT)

In the last few decades, secure algorithms have played an irreplaceable role in the protection of private information, such as applications of AES on modems, as well as online bank transactions. The increasing application of secure algorithms on hardware has made implementations on ASIC benchmarks extremely important. Although all kinds of secure algorithms have been implemented into various devices, the effects from different constraints on ASIC implementation performance have never been explored before.

In order to analyze the effects from different constraints for secure algorithms, *SHA-3 finalists*, which includes Blake, Groestl, Keccak, JH, and Skein, have been chosen as the ones to be implemented for experiments in this thesis.

This thesis has first explored the effects of different synthesis constraints on ASIC performance, such as the analysis of performance when it is constrained for frequency, or maximum area, etc. After that, the effects of choosing various standard libraries were tested, for instance, the performance of UMC 130nm and IBM 130nm standard libraries have been compared. Additionally, the effects of different technologies have been analyzed, such as 65nm, 90nm, 130nm and 180nm of UMC libraries. Finally, in order to further understand the effects, experiments for post-layout analysis has been explored. While some algorithms remain unaffected by floor plan shapes, others have shown preference for a specific shape, such as JH, which shows a 12% increase in throughput/area with a 1:2 rectangle compared to a square.

Throughout this thesis, the effects of different ASIC implementation factors have been comprehensively explored, as well as the details of the methodology, metrics, and the framework of the experiments. Finally, detailed experiment results and analysis will be discussed in the following chapters.

Acknowledgements

I would like to express my deep and sincere gratitude to Dr. Leyla Nazhandali for giving me the opportunity to work with her and her helpful guidance during my research process. I am quite privileged to be a student under the instruction of Dr. Leyla Nazhandali. Her unusual foresight and creative thinking has been a source of my inspiration and aspiration throughout the whole research period. Thanks to that, I have learned a lot from this lab.

I would like to express my ageless and immense appreciation to Dr. Patrick Schaumont for his kindness of being my co-advisor with the helpful instructions and inspiring advices. His extensive knowledge and unwearied attitude always encourages me during my research process. Due to the period I study with him, I would equip myself with the endless motivation and tireless perspiration, which I will benefit from for every single decision and success.

I would like to thank Dr. Sandeep Shukla for his participation in the my thesis committee.

I am much appreciatory to my friend Sinan Huang for his altruistic support and selfless friendship. Without him, I would have been stalemated when facing various obstructions. Without him, I can never come to this far. His participation in my life would never fade away.

I am much obliged to my lab mate and friend Meeta Srivastav for her patient instruction and tireless guidance. Her instruction and encouragement would account for a great part of my two year's life in VT.

I am grateful to my lab mate and friend Dinesh Ganta, Lalleh Rafeei for their kind help in the past.

My deepest gratitude should come to my parents, their never-stopping effort to make my life being better, their eternal love, tireless care and support make the meaning of my life meaningful, without them, I can never survive let alone come to this far and further, thrive.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Due to the improvement in feature size and clock speed, ASIC implementation of secure algorithms has gradually evolved into a critical topic in high performance computer architecture, especially hashing. Hashing is being used extensively in the hardware applications, such as page tables and address translations. Therefore, ASIC performance analysis of different algorithms becomes essential.

In November 2007, NIST organized a hash function competition, which has taken the security, cost, algorithm and implementation characteristics of a candidate into consideration. The quality of hardware implementation of an algorithm plays a significant role in the evaluation of finalists (five candidates in the round three). To address these issues, the effects from different ASIC implementation factors are sought for the finalists. In order to analyze the effects of the performance, the post-synthesis and post-layout analysis are considered. In the post-synthesis stage, the impact from different constraints are analyzed while executing the synthesis. Additionally, more details about how different libraries influence the performance of each candidates are evaluated. After that, it further considers the performance change across different technology nodes based on the UMC libraries. In the post-layout stage, the performance of each candidates with maximum frequency as well as different floor plan shapes have been considered.

1.2 Organization

The rest of the thesis is organized as follows:

Chapter 2 first introduces the basic background needed in this project, including the basic knowledge of hash function, and SHA3 competition. Then it continues to introduce the algorithm architecture of the finalists in the last round of SHA3 hash function competition.

Chapter 3 introduces the ASIC implementation flow as well as the library and implementation characteristics in order to lay the foundation for the future discussion.

Chapter 4 introduces the basic methodology for the comparison and the way the experiments have been conducted.

Chapter 5 introduces the organization of the analysis for the results and continues to analyze each part of the results.

Chapter 6 is the conclusion for the thesis work.

Chapter 2

Background

2.1 Introduction to Cryptographic Hash Function

Over the past few decades, the development of cryptography has been paralleled by the development of cryptanalysis—the "breaking" of coders and ciphers. Until the 1970s, secure cryptography was primarily the preserve of government. After that, the creation of a public encryption standard and the invention of public-key cryptography have brought it to the public. The secure hash algorithm has been extensively studied both in the public and academic domain. Starting in 1993, the first SHA standard was published, and now, NIST continues to organize the hash function competition, which will be given the name SHA-3 as the new generation of the standard.

2.1.1 General Description

In cryptography, encryption is the process of transforming information (referred as *plaintext*) using an algorithm (called a *cipher*) to make it unreadable to anyone except those who get the access to a special acknowledgement, usually referred to as a *key*. Therefore, due to the property of security, cryptographic hash functions are used intensively in various electronic applications.

A hash function H is a computationally efficient function that maps fixed binary chains of arbitrary length $0, 1$ to a bit sequences $H(M)$ of fixed length. In another word, let M be a message of an arbitrary length. A hash function operates on M and returns a fixed-length value, h, as shown in Figure 2.1. The value h is commonly called hash code, message digest or hash value.

Hash functions do not use a particular key, but rather a highly nonlinear function of all message bits. The hash value changes with the change of any bit or bits in the input

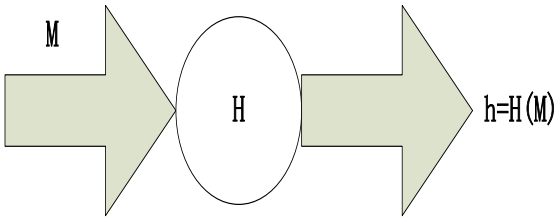


Figure 2.1: Hash Function

message and thus it provides error detection capabilities. If any two message can produce the same digest, then a collision for a specific hash function can be found. Collision is not desired when developing hash functions, but it is unavoidable. A summary for the properties of hash functions are as follows [1]:

- Hash function H applies to any block of data.
- Hash function H returns a fixed-length output.
- For any given value x, $H(x)$ is relatively easy to compute. That feature makes the hash function implementations more practical and easier to be accepted.
- Given x, it is easy to compute $H(x)$. Given h, it is computationally infeasible to find x such that $H(x) = h$. That is sometimes referred to as *one way* property of hash functions.
- For any given block x, it it computationally infeasible to find $y(y \neq x)$, with $H(y) = H(x)$. This is sometimes referred as weak collision resistance.
- To find a pair (x, y) such that $H(x) = H(y)$, is computationally infeasible. This is sometimes referred to as strong collision resistance.

2.1.2 Some Known Hash Functions

As mentioned before, for the typical hash functions, it has a general iterative structure, which is shown in Figure 2.2.

For this reason, although different algorithms may have their focused computation property, operation property, or strength concentration, they also share some typical implementation modules; for example, permutation operation, substitution and modulo computation methods are all shared.

In the development of encryption algorithms, confusion and diffusion operation are the two basic primitives, which would provide the obscurity and the quality of concealment. While in

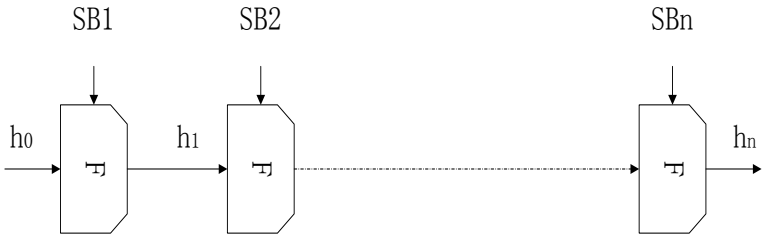


Figure 2.2: General Hash Structure

the shared computation methods, permutation is the way to create obscurity for algorithms. In mathematics, permutation is used to set an arrangement of a set of objects in a particular order, which can be done by using the factorial number system representation of integers up to $n!$ to generate permutation of n, and then convert the codes into the corresponding permutations needed. In secure hash algorithms, this is done by swapping a number of bits during the computation.

For the substitution component, from the literal, we can see the general meaning, which is to execute replacement. It is basically done by applying S-box, which is a basic component of symmetric key algorithms. The main functionality of the S-box is also to obscure the relationship between the key and the digest in order to increase the security of the algorithm. In general, an S-Box takes some number of input bits, denoted as m, and then transforms these bits into some number of output bits, denoted as n; in this case, an $m \times n$ S-Box can be implemented as a lookup table with $2m$ words of n bits each.

For modulo operation, it is a remainder computation method. In finding the remainder of division of one number by another, it scatters the influence of each bit in the computation to hide the information for each bit. In this way, it efficiently increases the ability of diffusion.

Aside from these shared computation methods mentioned above, hash functions also share another typical operation, which is named Padding. In the hash computation, it will hash the plain text into sub blocks, which have some fixed length m bits and will operate sequentially. While, during the computation, those message which are shorter than m in length will be padded with a number of zeroes.

Since the first publication, hash functions have been through a golden era of thriving; until now, there are various hash functions under use. Table 2.1 [1] shows a summary of some know hash functions and their basic properties, including the block size, and digest size since 1991.

Table 2.1: Some Known Hash Functions

Name	Author	Year	Block Size	Digest Size
AR	ISO	1992		
Boognish	Daemen	1992	32	up to 160
Cellhash	Daemen, Govaerts, Vandewalle	1991	128	128
FFT-Hash I	Schnorr	1991	128	128
GOST R 34.11-94	Government Committee of Russia for Standards	1990	256	256
HAVAL	Zheng, Pieprzyk, Seberry	1994	1024	128, 160, 192, 224, 256
MAA	ISO	1988	32	32
MD2	Rivest	1989	512	128
MD4	Rivest	1990	512	128
MD5	Rivest	1992	512	128
N-Hash	Miyaguchi, Ohta, Iwata	1990	128	128
PANAMA	Daemen, Clapp	1998	256	unlimited
Parellel FFT-Hash	Schnorr, Vaudenay	1993	128	128
RIPEMD	The RIPE Consortium	1990	512	128
RIPEMD-128	Dobbertin, Bosselaers, Preneel	1996	512	128
RIPEMD-160	Dobbertin, Bosselaers, Preneel	1996	512	160
SHA-0	NIST/NSA	1991	512	160
SHA-1	NIST/NSA	1993	512	160
SHA-224	NIST/NSA	2004	512	224
SHA-256	NIST/NSA	2000	512	256
SHA-384	NIST/NSA	2000	1024	384
hline SHA-512	NIST/NSA	2000	1024	512
SMASH	Knudsen	2005	256	256
Snefru	Merkle	1990	512-m	m = 128, 256
StepRightUp	Daemen	1995	256	256
Subhash Daemen	1992	32	up to 256	
Tiger	Anderson, Bihan	1996	512	192
Whirlpool	Barreto, Rijmen	2000	512	512

2.1.3 Typical Applications

During the past decades' development, hash functions have had various applications in the industry, especially in hardware security application and high performance computer architectures, such as the password protection, data integrity verification, and the technique of organizing tables for address translation, including bit extraction, and exclusive ORing hash methods. Among these applications, historically, digital signatures were the first application of cryptographically secure hash functions used most commonly.

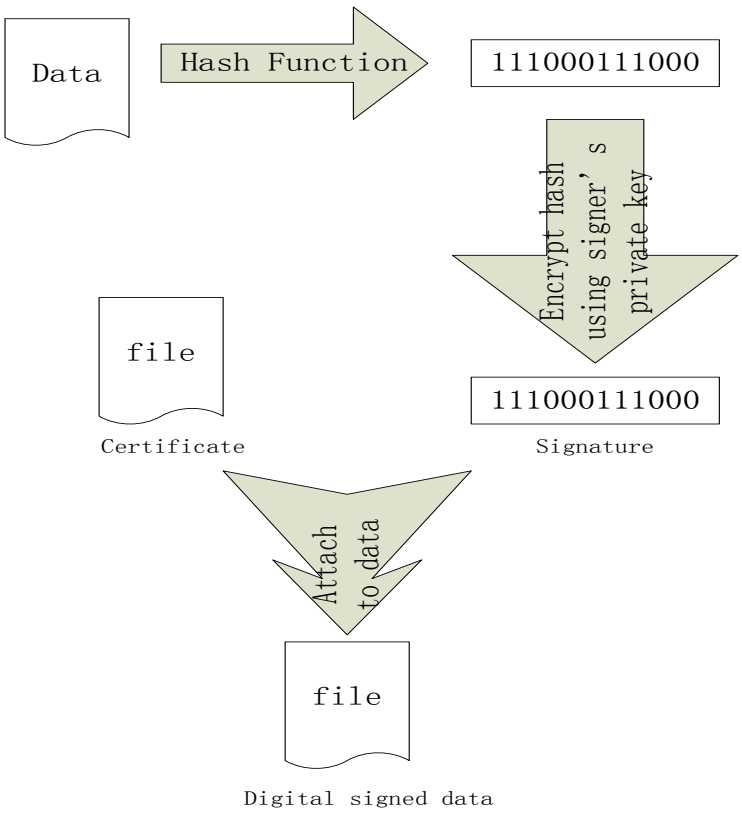


Figure 2.3: Basic structure for signing process

Digital signatures are prevalently used in financial transactions, purchase of merchant software, and application related to security or profits. A digital signature scheme is used to demonstrate the authenticity of a digital message or document, which needs a completely secure mathematical verification scheme. A valid digital signature should be secure enough to make the recipient believes that there is no modification during the transmission process.

For a general structure of digital signature systems, there are two major processes which are signing and verification. Figure 2.3 shows the signing process.

The main purpose of the signing process is to produce the digital signed data or document

for a transmission. Therefore, in the process, it first uses hash functions to generate the digest, then encrypts the hash value with the signer's private key, which is only accessed by the signer. After that, with the certificate provided by the third party, the digital signature produced by the hash computation is attached to the file for transit. In this process, the property that a signature is generated from a message of fixed length and the digital signature should be verified by using the corresponded public key by the receiver.

After the generation for the digital signature, it comes to the verification process which will verify the digital signature from the signing process. The structure of the process is shown in Figure 2.4.

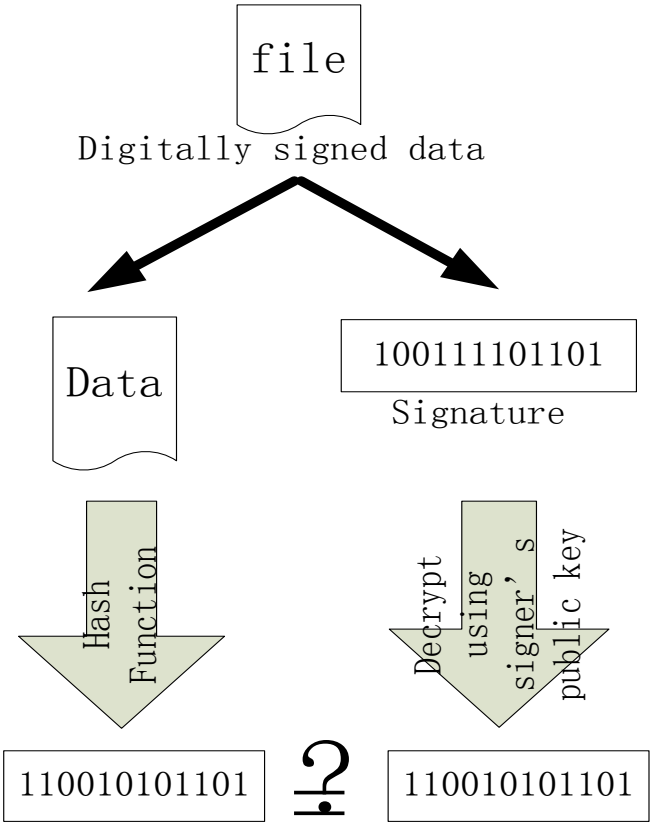


Figure 2.4: Basic structure for verification process

The main purpose of verification process is to verify the digital signature received by using the corresponding public key. As it can be seen in the process structure, it verifies the signature through the comparison of two computation results, which are the results of applying hashes to the data from the signed file and the results from decrypting the signature data by using the related public key. Through these steps, the property that the signature can be only verified by the corresponding public key and that it should be computationally infeasible to produce a valid signature for a third party who does not have the access to the private key

can be fulfilled.

Through the signing and verification process by using private key, public key and hash function computation, the digital signature application can provide a secure, trustable communication mechanism between the signer and the receiver.

2.2 Introduction to Secure Hash Algorithm

As mentioned before, hash functions are being used extensively in the current electronic applications, especially *Secure Hash Algorithms*, which is published by the *National Institute of Standard and Technology(NIST)* as a U.S. *Federal Information Processing Standard(FIPS).*

2.2.1 What is SHA?

In the cryptography domain, SHA stands for Secure Hash Algorithm, which is released by NIST/NSA. Since the first secure hash algorithm was published in 1993, SHA has been through four stages of developments[2]:

SHA-0: In 1993, the original version of the 160-bit hash function was published under the name of "*SHA*", which was known as SHA-0. It faded away and then was replaced by the slightly revised version SHA-1 shortly after its publication due to an undisclosed significant flaw.

SHA-1: SHA-1 is also a 160-bit hash function which resembles the earlier MD5 algorithm. This was designed by the National Security Agency to be part of the Digital Signature Algorithm.

SHA-2: After the SHA-1, a family, which is noted as SHA-2, of two similar hash functions with different block sizes was developed, known as SHA-256 and SHA-512. They are only different in word size; SHA-256 uses 32-byte words where SHA-512 uses 64-byte word. There are truncated version of each standardized, known as SHA-224 and SHA-384. These were also designed by NSA.

SHA-3: In 2012, an ongoing *NIST* hash function competition is scheduled to the end with selection of a winning algorithm, which will be given the name SHA-3.

2.2.2 Why SHA-3 Competition?

In recent years, hash functions serve as a deterministic role in the currently prevalent security applications, such as digital signatures, message authentication codes (MACs), and the implementation of hash function into the computer architecture in order to gain relatively high performance, secure hash algorithm also becomes increasingly popular.

Since the SHA-1 was attacked by finding a hash collision or some other subsequent methods by other publications, more advanced algorithm is eagerly needed in order to meet the desire for security. In response to the requirement for advancement in the cryptanalysis of hash algorithms, *NIST* organized an open competition in a similar way as what have done in selecting Advanced Encryption Standard(AES), in which the winner has been named SHA-2.

Although it has been proven that the previous attacks can not be applied to the SHA-2, *NIST* comes to feel it is prudent to seek for a more advanced algorithm through another open competition. The competition accepts all algorithms from different individuals, group and organizations. Through different evaluation metrics in various aspects, such as security, cost and implementation characteristics on different platforms, one algorithm will be selected out from the competition, which will be given the name SHA-3.

2.2.3 How is SHA-3 Competition Under Going?

There are three phases in the NIST SHA-3 competition in total.

In phase I, there are 64 submitted algorithms. The *First SHA-3 Candidate Conference* was scheduled for Febrary 22-25, 2009 at K.U. Leuven, Belgium, which allows all the submitters to present their algorithms, and for NIST to discuss the criteria to forward the competition. FRN-Nov07 identified three broad categories of evaluation criteria that will be used to compare the candidate algorithms throughout the SHA-3 competition:

(i) Security, which is the most important factor to be taken into account when evaluating the secure hash algorithms.

(ii) Cost and performance, which mainly includes the computational efficiency and memory cost.

(ii) Algorithm and implementation characteristics, which includes a variety of platforms and different architectures.

In phase II, 14 candidates were selected out to continue for the second round competition. On August 23-24, 2010, NIST hosted a *Second SHA-3 Candidate Conference* at the University of California, Santa Barbara to discuss the security and performance of the second-round candidates. In this phase, NIST evaluated the second-round candidates more specific.
For security:

(i) Applications of the hash functions.

(ii) Specific requirements when hash functions are used to support HMAC, Pseudo Random Functions(PRF), or Randomized Hashing.

(iii) Additional security requirements for hash functions.

(iv) Evaluations relating to attach resistance.

(v) Other consideration factors.

For cost and performance:

(i) Computational efficiency, which refers to the speed of the algorithm.

(ii) Memory requirement, which includes the code size and random-access memory(RAM) requirement for software implementation as well as gate accounts for hardware implementations.

For implementation characteristics:

(i) Flexibility and simplicity of the design. The FRN stated that candidates with more flexibility are preferred. Flexibility includes algorithms capable of running efficiently on a wide variety of platforms as well as algorithm that use parallelism or instruction-set extensions to achieve better performance. Additionally, FRN has stated that the algorithm should be judged according to the simplicity of the algorithm design so that it would be easier to be understood and analyzed by in the security of the design.

2.3 SHA-3 Finalists Archtecture Analysis

In the third round, which is the last round of the competition, NIST has selected out five final candidates as the finalists, which are Blake, Groestl, JH, Keccak, Skein. Before going to the ASIC performance evaluation, in this section, a detailed discussion of the architecture for each candidate will be performed.

2.3.1 BLAKE

BLAKE, which follows HAIFA iteration mode, operates on an inner state that can be represented as four by four matrix of words. The inner state is initialized using an Initial Value(IV), a salt and a counter. The state is updated using the G function, which is based on the *CHACHA* stream cipher[3].

Before the the message goes through the computation iteration, the input data, first, enters the combinational logic stage to create the message for the iteration, which can be seen in Figure 2.5 [4]. After going through the permutation stage, parallel G functions are implemented for four G functions each round with the modulo addition, XOR and rotational shifts operations. The hash value combined with IV will go into the inner state, then the finalization step will be taken in order to enter the iteration operation.

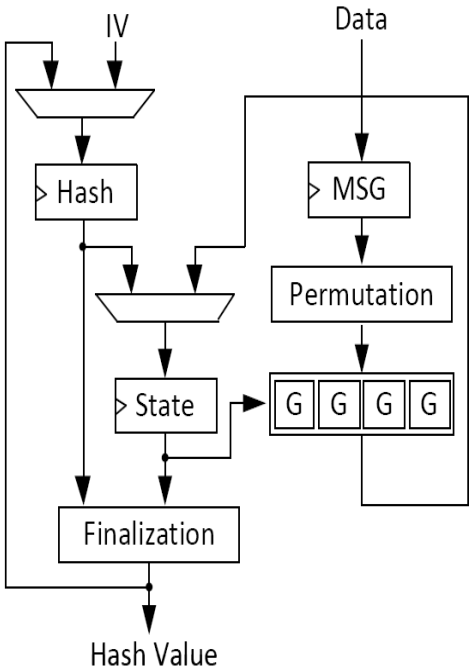


Figure 2.5: Basic structure of BLAKE-256.

2.3.2 Grϕestl

Grϕestl is known as a wide-pipe Merkel-Damgard hash algorithm, which has a novel compression function. The compression function uses $2n$ bit in the permutation stage with logic operation and truncation to achieve the collision and preimage resistance of an n-bit wide function in the ideal situation[5].

The fixed permutation and AES share a similar structure except the difference in the size expansion. For the structure of function P and Q, they can be implemented either parallelly or independently. In this case, shown in Figure 2.6, the computation is implemented parallelly in order to gain higher performance. For the final output result, through processing the chaining state, it discards half of the bits, generating n-bit digest [6].

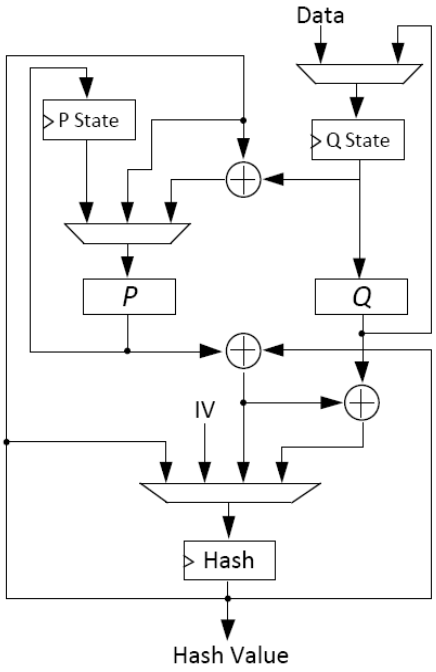


Figure 2.6: Basic structure of Grϕestl-256.

2.3.3 JH

JH hash algorithms have a single compression function F_8, which can be implemented with bitslice mode. Among the different algorithms in JH family, they can be distinguished from each other by the different initial value provided. For the domain extension, it also has a wide-pipe Merkle-Damgard structure.

As shown in Figure 2.7, the R_8 module has a layer of permutation, a layer of linear transformation, and a layer of S-boxes. There are three components of the underlying permutation

E_8, which are four-bit S-boxes, an eight-bit L-permutation and a P-permutation. The implementation of S-box can be a series of logic functions instead of lookup tables. For the final hash value, it can be abstained by truncation of the final output[7].

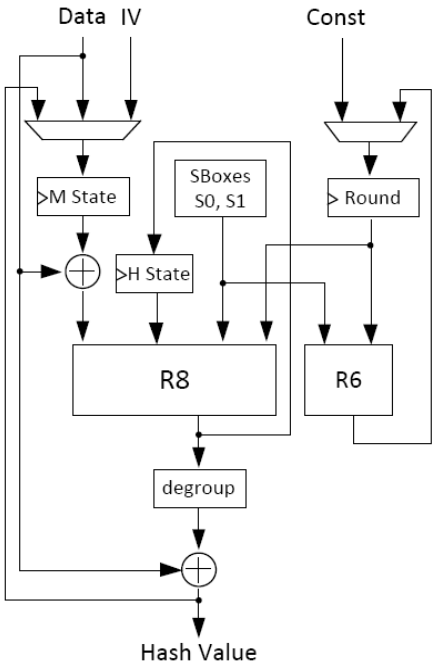


Figure 2.7: Basic structure of JH-256.

2.3.4 Keccak

Keccak is based on the sponge construction model, as shown in Figure 2.8. It has two internal storage registers. During the computation process, it first XORs with the input data stored in the first portion, then in the round computation, the result will be updated with the data stored in the second portion.

The permutation is a combination of linear and non-linear mixing operation, which can be treated as a substitution-permutation network with 5-bit wide S-boxes. For the final output, the value of the first storage portion will be a part of the hash value, it can accommodate its output to any size by updating its internal registers[8].

2.3.5 Skein

Figure 2.9 shows the basic structure for Skein. Skein is an iterative hash algorithm, which is based on a modified Meteas-Meyer-Oseas constructure. It uses tweakable block cipher,

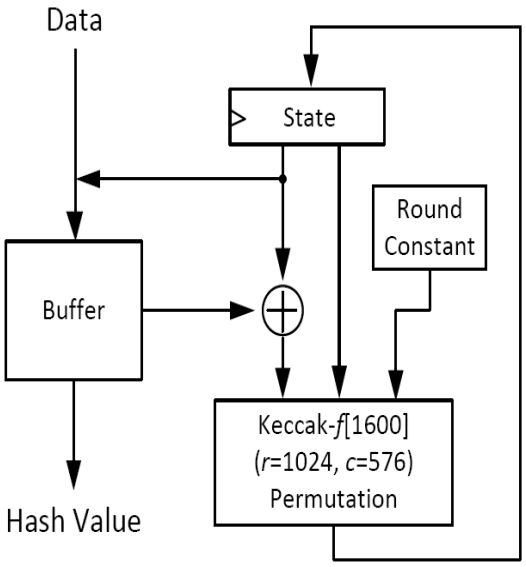


Figure 2.8: Basic structure of Keccak-256.

named Threefish, to build compression functions. Threefish is 72-round substitution-permutation network using 128-bit mixing function. Each round includes a permutation and four mixing parallel operation of addition, rotation, and XOR operations [9].

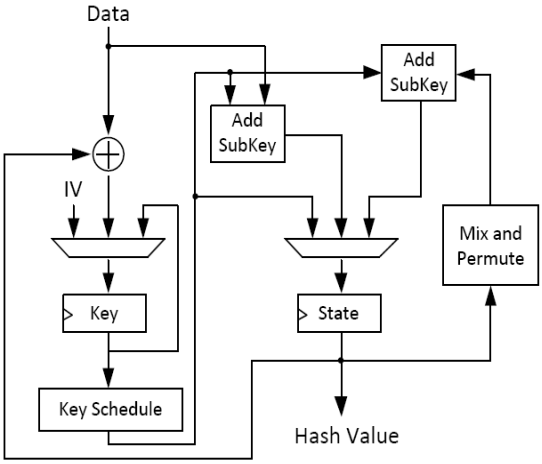


Figure 2.9: Basic structure of Skein-256.

Chapter 3

ASIC Implementation Analysis

The main objective of this chapter is to analyze various parameters of ASIC implementation flow. First, the definition of ASIC implementation will be discussed. Then it will introduce the general flow for ASIC design. After that, it will come to the discussion of effects and evaluation methods for different parameters during ASIC design, including timing, and power analysis. Finally, aiming at the future discussion in the later chapters, standard library characteristics will be discussed.

3.1 What is ASIC?

An ASIC is an Application Specific Integrated Circuit, which means that an IC would be called ASIC if we design it for the specific application. Example of ASIC design includes, chip desgined for satellites, chip designed for a modem, chip designed for SHA-3 algorithms. As compared with, the example of ICs which are not called ASIC include memories, microprocessors and the like.

The classification of ASIC design is described below:

- Full-Custom ASIC: For this type of design, there are no predefined logic gates or predefined cells out of logic gates. The designers have to develop all the logic cells and the layout for this specific implementation.
- Standard Cell ASIC: For this type of design, the designers can use various predefined logic gate including AND, OR, and NOT, and predefined cell modules including MUX, XOR, and ADD. All these gates and cell modules are called standard cells. The advantage of using standard cells is that it saves time, reduces the cost and risk during the design process. Although the standard cells are developed through the same methodol-

ogy as custom cells', each standard cell has been pre-tested and optimized individually; and also, each cell for different logic function has a variety of emphasis such as different driving strength, cells for optimum area, and cells for optimum power.

- Gate Array ASIC: For this type of design, the position of the transistors are predefined on the silicon wafer. This predefined pattern for transistors on a gate array is called Gate Array ASIC(base array), and the smallest unit in the base array is called base cell, each base cell has the same logic elements; only the the interconnection between base cells and inside the base cells can be customized. For the classification, Gate Array ASIC has three types, which are Channeled Gate Array, Channelless Gate Array, and Structured Gate Array.

3.2 ASIC Design Flow

In Figure 3.1 [4], an ASIC design flow starting from *RTL* design to *GDSII* file can be found.

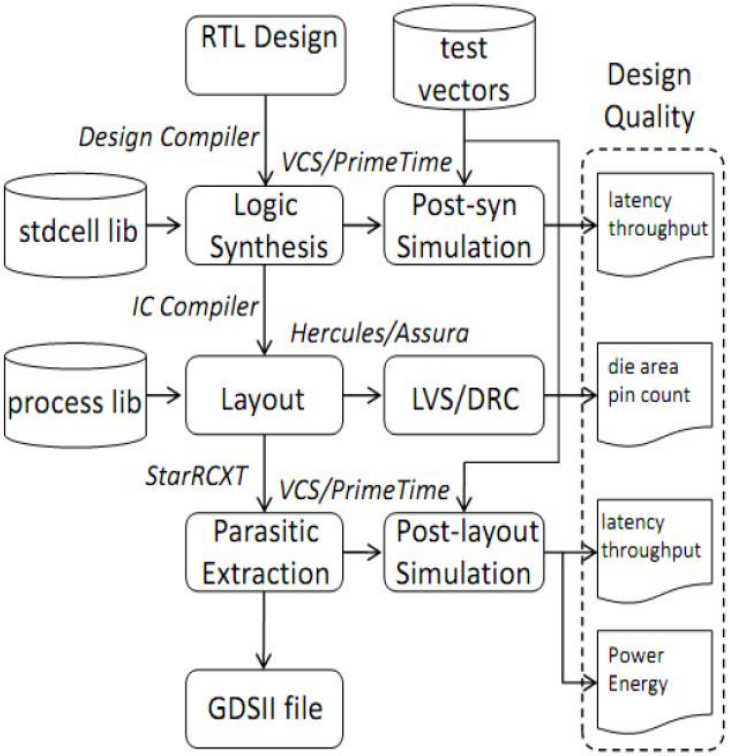


Figure 3.1: Complete ASIC design flow.

There are three basic stages: *RTL* design, *Logic Synthesis*, and *Layout*. RTL design is Register Transfer Level architecture design, which is the Structural and Functional Description of the design. The next stage is to convert RTL into the optimized Gate Level Netlist, which is denoted as RTL/Logic Synthesis. Typically, an estimation of performance can be done in this stage. After that, it comes to the stage of Physical Implementation of gate-level netlists, which is to map the produced gate-level to geometric representations. Under this stage, more accurate measurements of design performance can be achieved.

3.3 Performance to be Evaluated

Despite of the process, the performance of a design would be always the most pivotal point in this consideration. In this section, I will mainly focus on the performance of timing, throughput and the ratio of throughput to area.

3.3.1 Timing

With all the designs considered, the timing constraint is the highest priority. Timing constraint is to set a minimum time interval that the signal on the critical path can hold constantly long enough to ensure the correct functionality of any given cells. Timing calculation mainly includes setup time and hold time, which are shown in Figure 3.2.

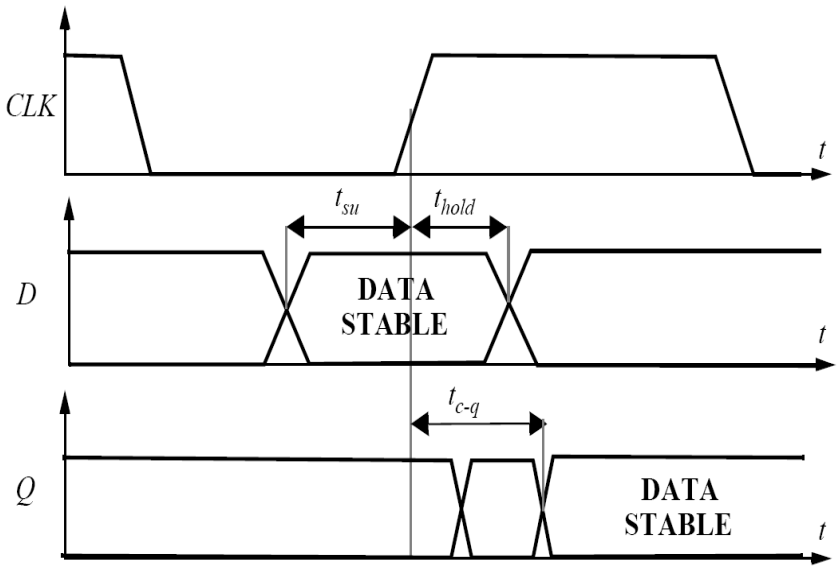


Figure 3.2: Definition of setup time and hold time.

Two basic timing constraints have been considered. One constraint is setup time, the other one is hold time. The setup time is defined as the minimum amount of time the signal should remain constant before the active clock edge to ensure the correct functionality of given cells. Hold time is defined as the minimum length of time that the input-signal must be stable after the active clock edge to ensure the correct evaluation of the input values. (3.1) and (3.2) describe the timing constraint for a proper operation of a circuit [10].

$$T_c \geq t_{pcq} + t_{pd} + t_{setup} \tag{3.1}$$

$$t_{cd} + t_{ccq} \geq t_{hold} \tag{3.2}$$

The general equation used to find the timing of a critical path, shown in (3.3), can be generalized to the EKV model by defining the drain source current as (3.4). $I_{S,CP}$ is the average specific current of the critical path, K is a fitting parameter, and IC is the inversion coefficient, which is defined in (3.5). The inversion coefficient represents the inversion of a transistor in both subthreshold ($IC < 1$) and superthreshold regions ($IC > 1$) [11].

$$t_d = \frac{C_{CP} V_{DD}}{I_{DS}} \tag{3.3}$$

$$I_{DS} = \frac{I_{S,CP} IC}{K} \tag{3.4}$$

$$IC(V_{DD}) = \ln\left(e^{\frac{V_{DD}(\lambda_D+1)-V_{TH}}{nV_T}} + 1\right)^2 \tag{3.5}$$

$$t_d = \frac{K C_{CP} V_{DD}}{I_{S,CP} IC(V_{DD})} \tag{3.6}$$

(3.6) shows the EKV equation for delay. Here, C_{CP} represents the total capacitance of the critical path and $I_{S,CP}$ represents the specific current of the critical path, both of which are assumed to be circuit dependent parameters.

3.3.2 Throughput

In the ASIC implementation for an algorithm, the evaluation of the throughput is the best method to judge the performance. Through the evaluation of throughput, we can get a sense of the computation ability of an algorithm. Throughput can be calculated as (3.7).

$$Throughput = \frac{BlockSize * MaxFreq}{Latency} \tag{3.7}$$

In the equation above, the variable "BlockSize", which is fixed once the algorithm has been developed, is the length of data that the algorithm can process each time. The core latency is dependent on the specific implementation architecture.

3.3.3 Throughput/Area

Throughput can be expressed as the speed to process data in a specific amount of time, which is referred as the performance of an algorithm. And, in hardware design, area stands for cost. Then the ratio of throughput to area can be expressed as the performance efficiency,

which provide a balance between the performance and the cost. Simply, the calculation for throughput to area is shown in (3.8).

$$Throughput = \frac{BlockSize * MaxFreq}{Latency * Area} \tag{3.8}$$

3.4 Power Calculation

Whenever the power consumption is mentioned, it means two aspects: dynamic power and leakage power, which are discussed in the following sections.

3.4.1 Dynamic Power

Dynamic power is often referred as active power, (3.9) gives the basic equation for the active power dissipation of a circuit.

$$P_{swiching} = \alpha * C_L * V_{DD}^2 * f \tag{3.9}$$

The active factor, α, is the probability that the circuit node transitions from 0 to 1, because that is the only time the circuit consumes power. C_L is the load capacitance, which shows the driving strength of a circuit. The supply voltage V_{DD} and clock frequency f are readily known by the designer.

3.4.2 Leakage Power

Leakage power is consumed even when a chip is not switching, which arises from subthreshold conduction, gate capacitance, junction leakage current, and contention current.

The EKV equations for digital CMOS are based on a circuit dependent parameter known as specific current, I_S, given in (3.10), which is the current when $V_{DS} = V_{GS} = V_{TH}$. The parameters of I_S are subthreshold slope, n, mobility, μ, oxide capacitance, C_{ox}, and thermal voltage, $\phi_t = kT/q$ [11]. (3.11) shows the EKV equation for leakage power where λ_D is the Drain-Induced Barrier Lowering (DIBL) Coefficient and $I_{S,T}$ is the total specific current of the entire circuit.

$$I_S = 2nC_{ox}\mu\frac{W}{L}\phi_t^2 \tag{3.10}$$

$$P_L = V_{DD}I_{S,T}e^{\frac{\lambda_D V_{DD} - V_{TH}}{nV_T}} \tag{3.11}$$

Based on the analysis, subthreshold leakage current flows when a transistor is supposed to be OFF. Gate leakage occurs when carriers tunnel through a thin gate dielectric when a voltage is applied across the gate. Junction leakage occurs when a source or drain diffusion region is at a different potential from the substrate.

3.5 Characteristic Analysis

In this thesis, many parameters are evaluated, such as library, timing, area, power, energy, throughput and the ratio of throughput to area. In the following sub-section, detailed discussion will be conducted.

3.5.1 Library Characteristics

As mentioned in the previous chapters, the same logic functionality can be implemented with different standard cell blocks. They would probably have different performance in area, power, and speed. Therefore, a detailed discussion about the characteristics of different library cells for the same logic functionality becomes quite necessary.

Using a multiplexer as an example. There are basically two methodologies to construct a multiplexer: one is to implement a MUX as a whole cell block, and the other one is to implement it by the combination of logic cells, like NAND, NOR and Inverter. Figure 3.3 shows three popular construction of MUX based the two methodologies mentioned above.

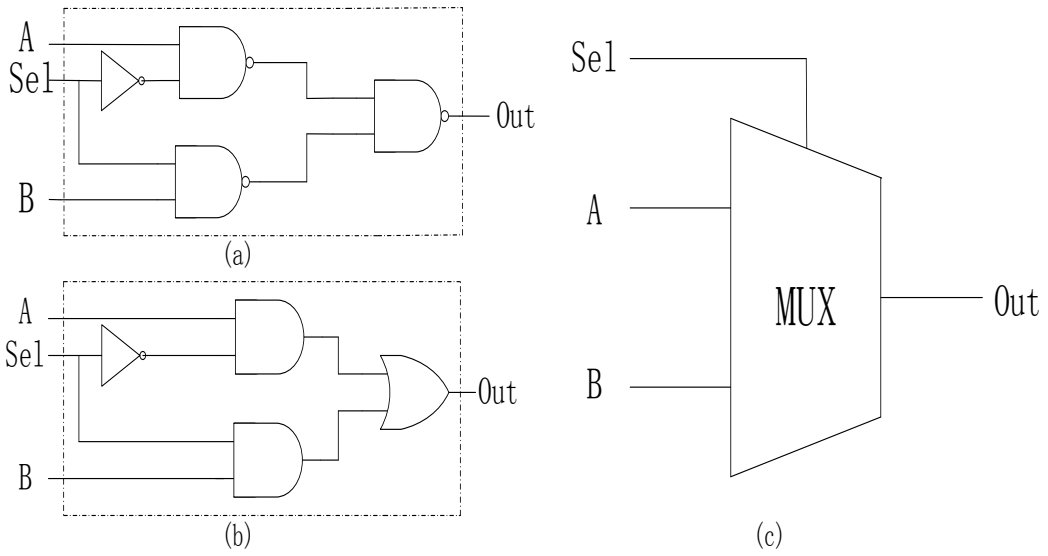


Figure 3.3: Several Structure for Multiplexer Implementation.

Because the standard cells AND2, NAND2, OR2, INV and MUX2 have different driving strength, power consumption, width, etc., the performance of the three constructions in the figure above will be different. Table 3.1 shows the range of the performance in area for each standard cell. Although the area of the MUX standard cell is slightly larger than each individual block of AND2, NAND2, OR2 and Inverter, standard cell block as a whole to form a MUX is much smaller.

Table 3.1: Cell Size Comparison

Cell	Height(um)	Width Range(um)	Area Range $(um)^2$
AND2	3.60	2.00∼4.40	7.20∼15.84
NAND2	3.60	1.60∼5.20	5.76∼18.72
OR2	3.60	2.00∼4.40	7.20∼15.84
INV	3.60	1.20∼5.60	4.32∼20.16
MX2	3.60	3.60∼6.40	12.96∼23.04

Therefore, in Figure 3.3 (a), the multiplexer is consisted of 4 NAND gates and one NOT gate, which resulting the minimum area for this implementation of 27.36 μm^2.

In Figure 3.3 (b), the multiplexer can be made up of two AND gates, one OR gate and one NOT gate, which results in the minimum area for this structure of 25.92 μm^2.

In Figure 3.3 (c), the multiplexer is directly provided as a standard cell block available in the library, which has a minimum area of 12.96 μm^2.

3.5.2 Implementation Characteristics

In the previous section, the difference in primitive gate in the performance of area have been observed. While, how is this difference reflected in the implementation for a particular design?

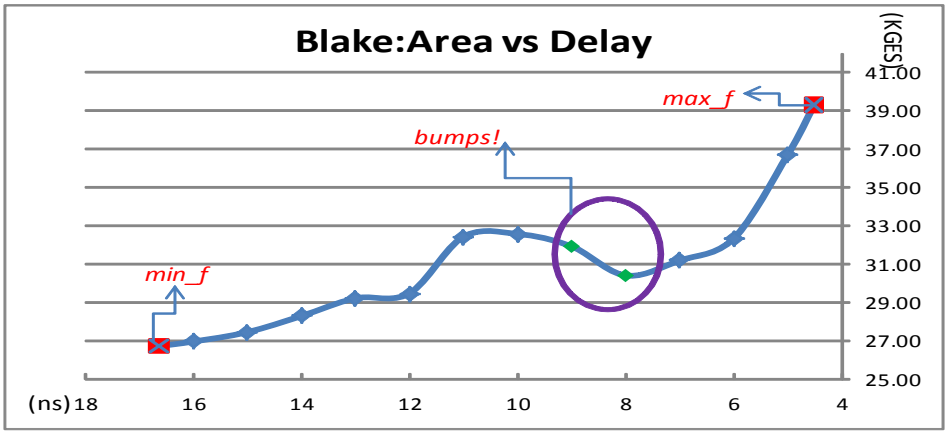


Figure 3.4: Bumps analysis in the synthesis implementation.

In Figure 3.4, the implementation for VT's Blake can be found to be an example. In the figure, two red points can be found, which are marked as min_f and max_f respectively.

min_f shows where constraints can be easily met on the critical path. Meanwhile, max_f point means the best implementation has been reached after searching all the standard cells available in the library exhaustively to meet the constraint.

Additionally, a bump is marked by the purple circle. Normally, when a tighter timing constraint is applied, a larger area should be achieved to meet the timing requirement. While, by analyzing Figure 3.4, an opposite conclusion can be found.

The two points, which form the bump, have different gate-level netlists. After analysis, the two netlists which have the same hierarchical module can be found. Among which a module named "add91" resulted in the largest difference of area. The one at "8ns", which has 198 cells, only used basic primitive logic gates, such as OR2, NAND2, NAND3, etc. While the one at"9ns", which has only 35 cells, used no basic gates at all, instead, it used a cell named "ADDFHX4TF" which is a full adder with driving strength foure. In order to find out the corresponding implementation of the cells, the implementation of bit 9 has been randomly chosen to do the comparison. The results are summarized in Table 3.2.

Table 3.2: Implementation Cells for Module "add91"

Cell Name	Height (um)	Width (um)	Area $(um)^2$	Cell Name	Height (um)	Width (um)	Area $(um)^2$
OR2X2TF	3.60	2.00	7.20	ADDFHX4TF	3.60	23.20	83.52
NAND2XLTF	3.60	1.60	5.76				
XNOR2XLTF	3.60	3.20	11.52				
Total			24.48				83.52

From the results in the table, it is known that, in order to meet the timing constraint, the tool chose to implement the current design with simpler, faster cells. And, the substitute from "ADDFHX4TF" has not only met the timing constraint but also reduced the area.

Chapter 4

Methodology

This chapter will present the methodology for the experiment. First it will introduce the methodology to conduct the comparison. Then it will come to the organization of experiments. Finally

4.1 Comparison Methodology

For the purpose of conducting a full comparison for different parameters, a two-dimension way has been chosen to conduct the comparison: *horizontal comparison*, which is termed as ***Intra Comparison***, and *vertical comparison*, which is termed as ***Inter Comparison***.

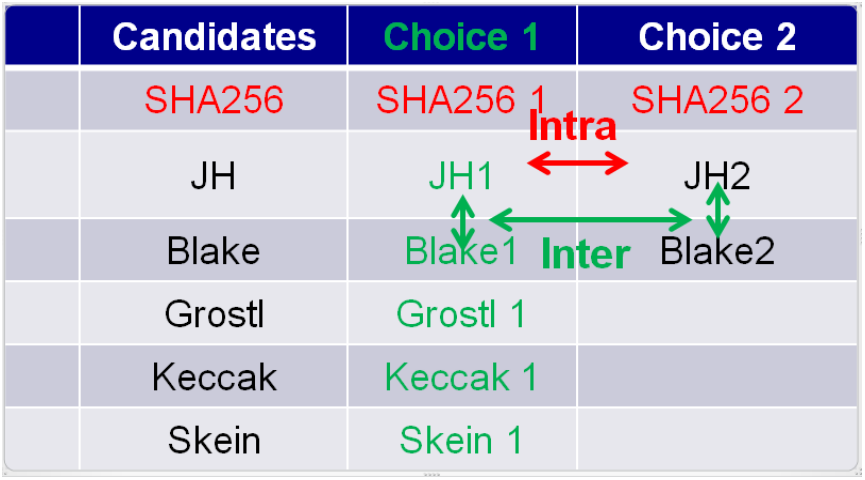


Figure 4.1: Comparison Methodology.

In Figure 4.1, a big picture of this methodology can be found. The ***Intra Comparison*** is the comparison within each candidate(algorithm) to evaluate the effects from different

choices, such as libraries or constraints. ***Inter Comparison*** is the comparison across different candidates with one specific choice to evaluate the relative quality of each candidate for a given choice.

Both the intra comparison and the inter comparison have a comparison base. All the absolute values will be normalized into the base for convenience. That is the base value will always be "1", and all the other values will be converted into relative values to the base. As stated in the figure, "choice 1", or the column with green color, serves as the base for intra comparison, while the candidate named "SHA256", or the row with red color, functions as the base for inter comparison.

4.2 Methodology for Experiment

4.2.1 Organization for Experiments

In Figure 4.2, a hierarchical view can be found, which summarize how these experiments have been done in the synthesis stage:

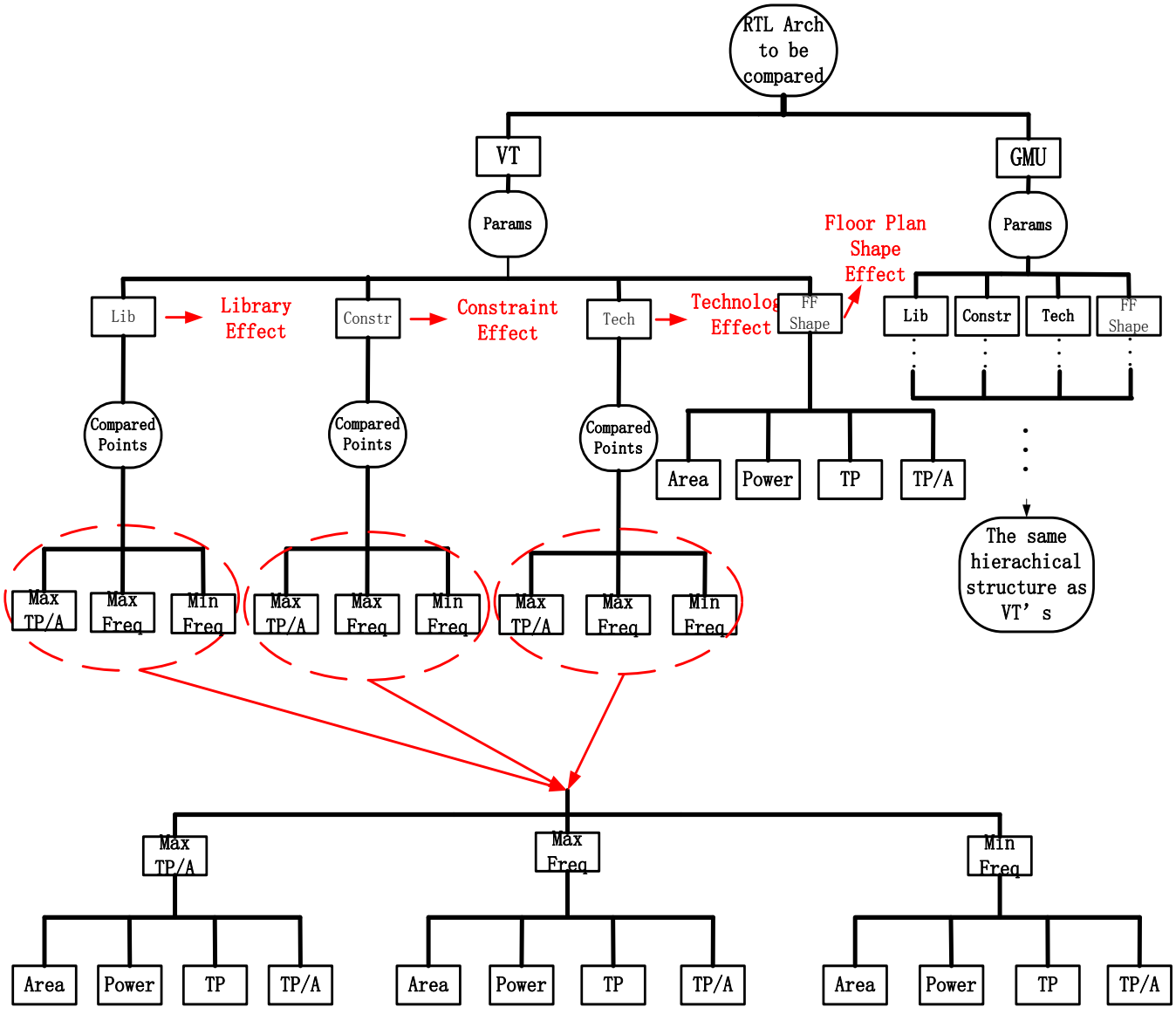


Figure 4.2: Hierarchical view of the experiments in synthesis stage.

Instead of comparing all the possible combinations, the points of interest hasve been chosen to conduct the comparison. In other words, the comparison is conducted using maxi-

mum frequency(maximum throughput) point, minimum frequency point, and the maximum throughput/area point based on different libraries and technologies available at hand.

4.2.2 Trend of Change Analysis

When different constraints are applied during the synthesis and layout stage, the current design will perform differently, especially in the synthesis stage. Therefore, before any further comparison, it is necessary to take a look at the varying trends for each performance parameter. In this section, the experiments showing the change in trends will be continued with Virginia Tech's Blake algorithm using different libraries and technology nodes.

In this section, the experiment for synthesis has been conducted with $1ns$ step when applying a series of timing constraints. Two sorts of experiments are performed:

- Blake with IBM130nm and UMC130nm, in order to check the difference using different libraries while maintaining the same technology.
- Blake with UMC65nm, UMC90nm, UMC130nm and UMC180, in order to check the difference between different technologies in UMC library.

4.2.3 Constraint Effect

In Figure 4.3, the UMC130nm library from Faraday has been chosen to explore the impact from different constraint choices.

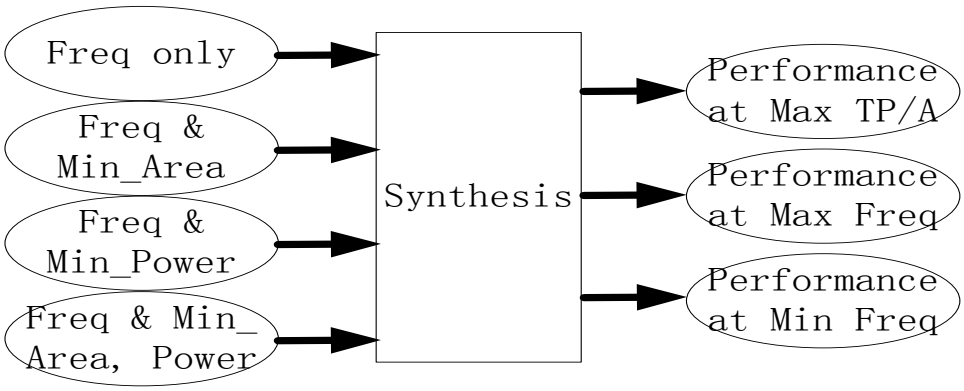


Figure 4.3: Experiment organization for constraint effects.

4.2.4 Library Effect

In this section, the IBM130nm library from ARM and the UMC130nm library from Faraday are chosen to conduct the experiments, which can be seen in Figure 4.4. The effect from different libraries is explored during this stage.

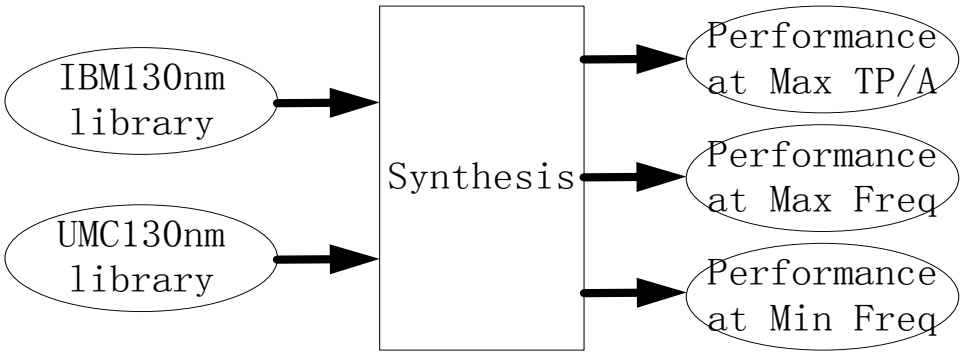


Figure 4.4: Experiment organization for library effects.

4.2.5 Technology Effect

Figure 4.5 shows the experiments of technology effects. In this section, library of UMC65nm, UMC90nm, UMC130nm and UMC180nm from Faraday have been chosen to explore the impact from the different technology choices.

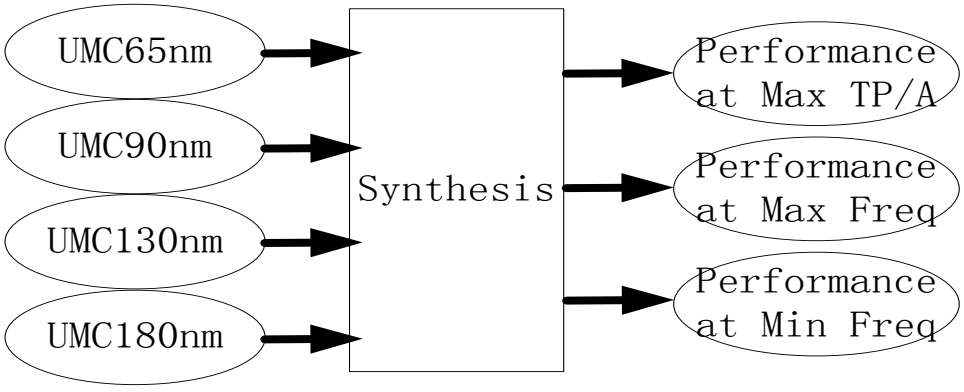


Figure 4.5: Experiment organization for technology effects.

4.2.6 Floorplan Effect

In this section, the IMB130nm library from ARM has been chosen to conduct experiments using various floor shapes (square(1:1), flat rectangle(1:2), flat rectangle(1:3), standing rectangle(2:1), and standing rectangle(3:1)), which can be seen in Figure 4.6. The netlist is created at the maximum frequency constraint from the synthesis.

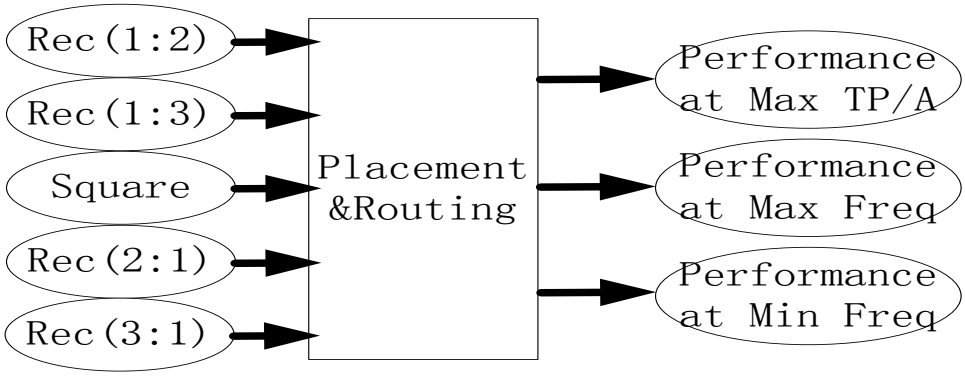


Figure 4.6: Floor Plan Shape.

Chapter 5

Results and Analysis

This section is the core part of this thesis. It basically has five sub-sections, which will show all sets of results, comparison, and analysis based on different performance parameters. In order to clearly present the data, in this first sub-section, a very detailed hierarchical structure about how this chapter is organized has been provided. And then, in the following sub-section, it is the analysis for the trend of changing based on the frequency constraint applied continuously. Then it will come to the comparison and analysis from four different aspects, which can be expressed as constraint effect analysis, library effect analysis, technology node effect analysis, and floorplan shape analysis on the selected implementation points. These implementation includes the one on maximum Throughput/Area, maximum frequency, and minimum frequency based on two sets of RTL architectures provided by Virginia Tech and George Mason University.

5.1 Hierarchical Structure for Result Analysis

Since there would be many sets of results presented in this chapter, before going the details of the result analysis, an overview of the structure has been used to show the results.

A hierarchical view of the organization to present the results in the analysis part can be found in Figure 5.1.

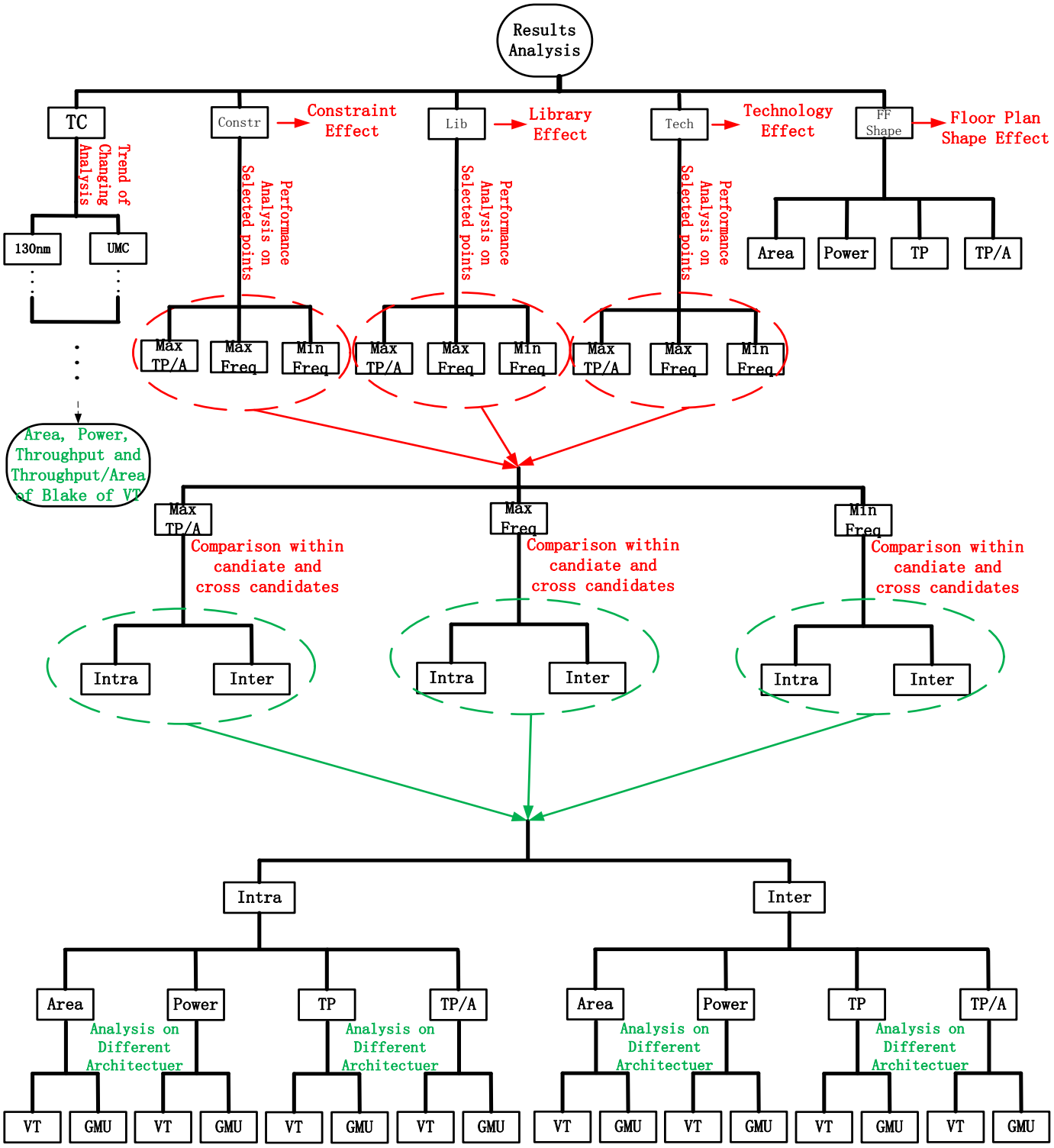
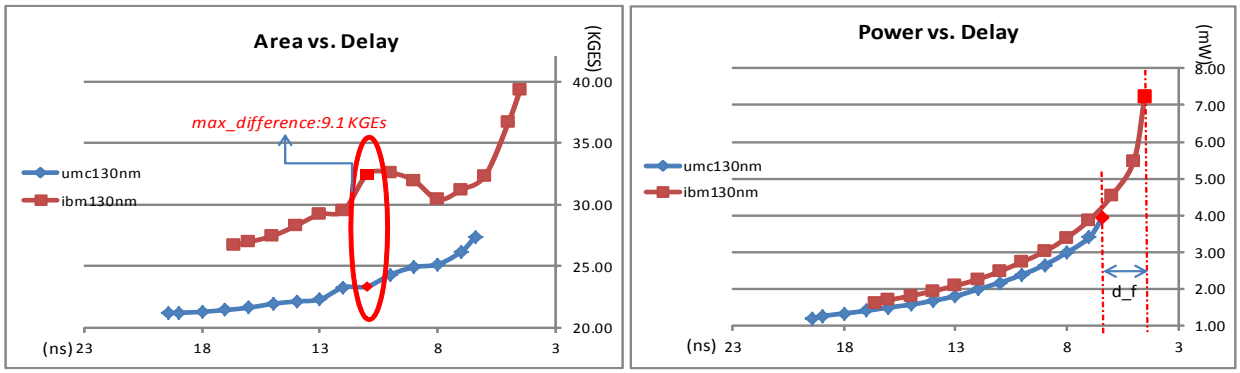


Figure 5.1: Hierarchical architecture of the result analysis

5.2 Trend of Performance Changing

5.2.1 Results Based on Different Library

This section presents the results from experiments based on different libraries with the same technology, which are IBM and UMC130nm. The candidate Blake has been randomly chosen to show the trend of change on performance according to different timing constraint for critical path.



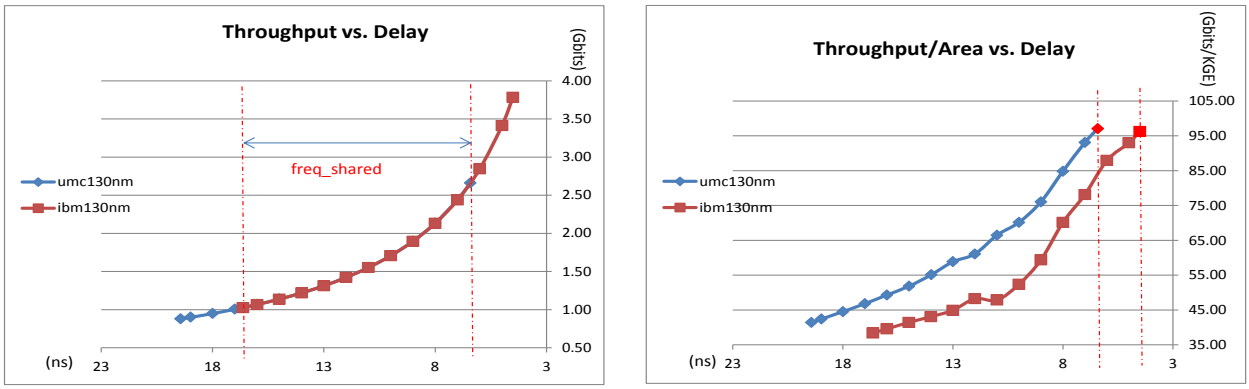
(a) (b)

Figure 5.2: Area and power versus delay plot for Blake on 130 technology

The performance in area is shown in the Figure 5.2 (a) above. Although the overall trend of the area keeps changing consistently as the tighter timing constraints are being applied, the performance of the two sets of implementations are different. First, the area increases as the tighter timing constraint is applied for both implementations. Second, the area implemented with IBM130nm is always larger than the one with UMC130nm. At the point of 11ns, it shows the maximum difference in area, which is 9.1 KGEs.

Except the area, the performance in power is shown in Figure 5.2 (b). UMC130nm always performs better than IBM130nm on power, while worse on the timing. This resulting performance is due to the different emphasis effort on the power when developing the standard cells. The d_f represents the difference between the maximum frequency that can be achieved.

Figure 5.3 (a) shows the performance of throughput and Figure 5.3 (b) shows the ratio between throughput to area (figure b). In 5.3 (a), it is not difficult to see that the implementation with IBM130nm has potential to implement a faster design. However, in the frequency range, it is common between this two libraries. The throughput is basically the same because of the same algorithm structure. In 5.3 (b), as compared with Figure 5.2 (a), we can say that although the implementation with IBM130nm is faster, when taking the area into consideration to evaluate the performance, the implementation with UMC130nm is better overall.

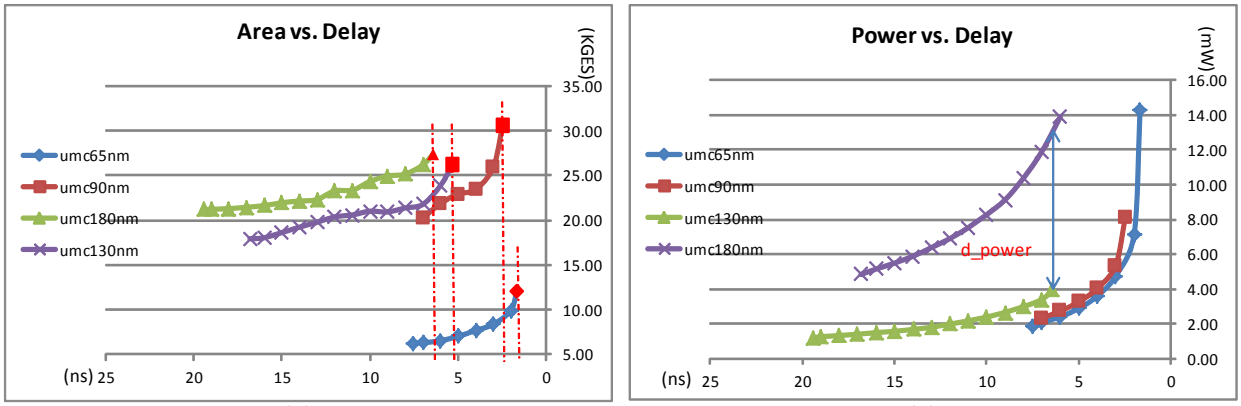


(a) (b)

Figure 5.3: Throughput and throughput/area plot for Blake on 130 technology

5.2.2 Results Based on Different Technology

After the performance in different libraries has been explored, it is comprehensive to take the performance of different technologies into the analysis.



(a) (b)

Figure 5.4: Area and power versus delay plot for Blake on UMC library

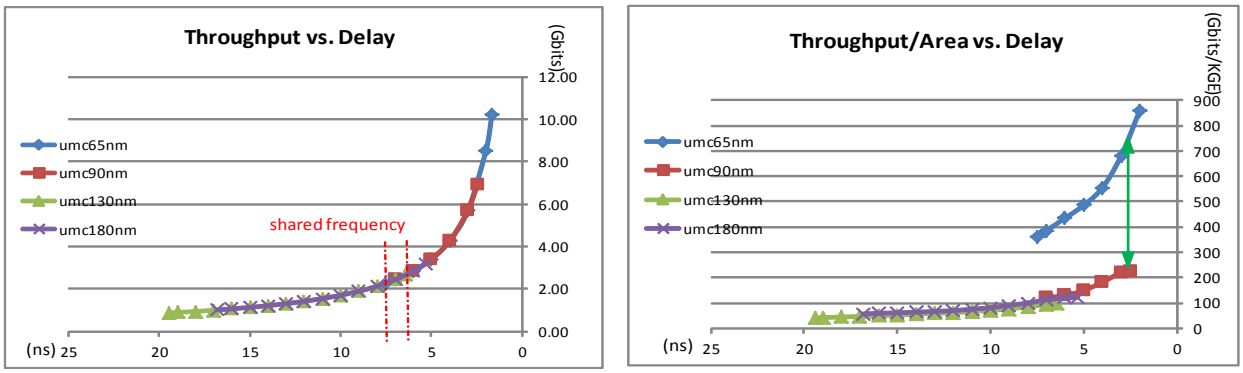
Figure 5.4 shows the area versus delay and the power versus delay based on the UMC library across different technologies.

UMC65nm has the smallest feature size, which is able to achieve the largest maximum frequency compared with other technologies. In Figure 5.4 (a), except for the advantage of the maximum frequency, the UMC65nm has far less area consumption than other libraries. And then combined with 5.4 (b), we can see, that UMC65nm also performs much better in power consumption.

With these results, we can conclude that the library with largest feature size will consume

more area and power than others and the ranking of performance both on area and power remains the same.

Figure 5.5 shows the performance in throughput and throughput to area:



(a) (b)

Figure 5.5: Throughput and throughput/area plot for Blake on UMC library

The shared frequency range for the four UMC libraries is extremely small in this case. Except the UMC65nm, all the other libraries have the same level of performance, especially on the ratio of throughput to area, as shown in the figure above. This matches our basic analysis according to the feature size of each library.

5.3 Constraint Effect Analysis

5.3.1 Implementation at Maximum Throughput/Area

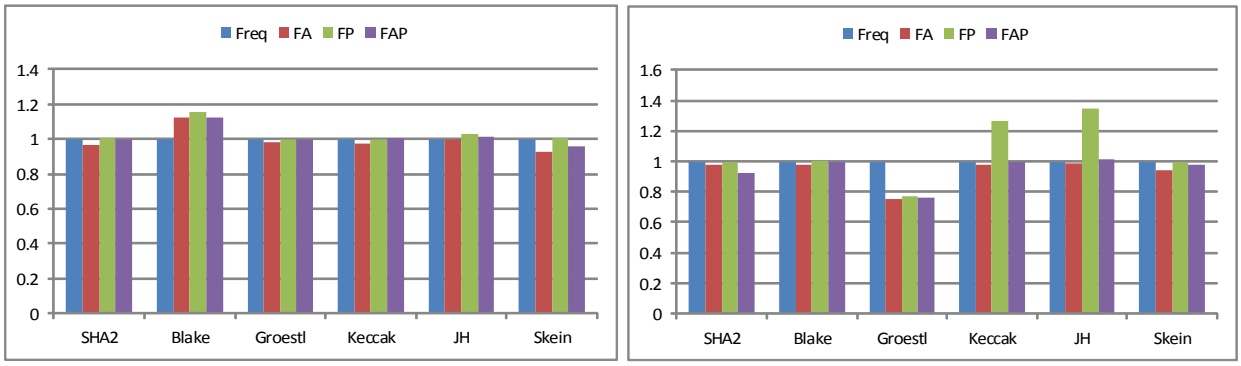
This section will discuss how the different constraints would influence on the area, power, throughput, and throughput/area with the implementation at maximum throughput/area. All comparisons are conducted through two basic methods which have been introduced in Chapter 4, *Intra Comparison and Inter Comparison*.

Intra Comparison

Intra comparison is to analyze the effects of different constraints. Among those constraints that can be applied, the frequency, area, power interest the designers most. So, the comparison of this section will mainly focus on four combination of these constraints, which are frequency (denoted as Freq), frequency and area (denoted as FA), frequency and power (denoted as FP), and frequency, area and power (denoted as FAP). For the comparison convenience, all the values of area, power, throughput, and throughput/area have been normalized to the value of of frequency. That means the value for the constraint of frequency should always be "1". Table A.1 has come up with a summary of the normalization for intra comparison.

- **Performance in Area**

Figure 5.6 shows the area variation between different constraints:



(a) VT (b) GMU

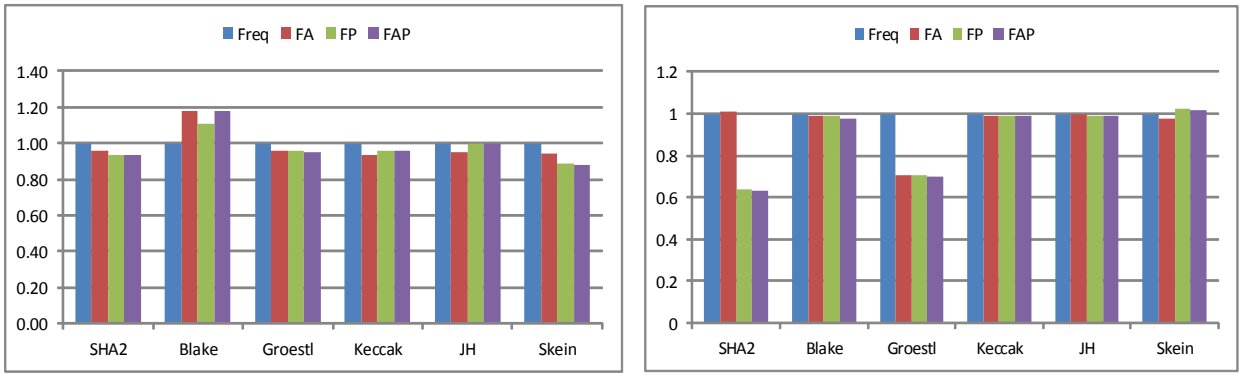
Figure 5.6: Area performance of different constraints

Although variation amongst different constraints in the aspect of area is extremely small, it still can be seen that the variation has a uniform trend. That is the performance in area

with area (referred to "FA" or FAP) constraint would be smaller than the one without area constraint (referred to "Freq" "FP"). This is uniformly applied to the architectures of both VT and GMU.

- **Performance in Power**

Figure 5.7 shows the performance of power between different constraints:



(a) VT (b) GMU

Figure 5.7: Power performance of different constraints

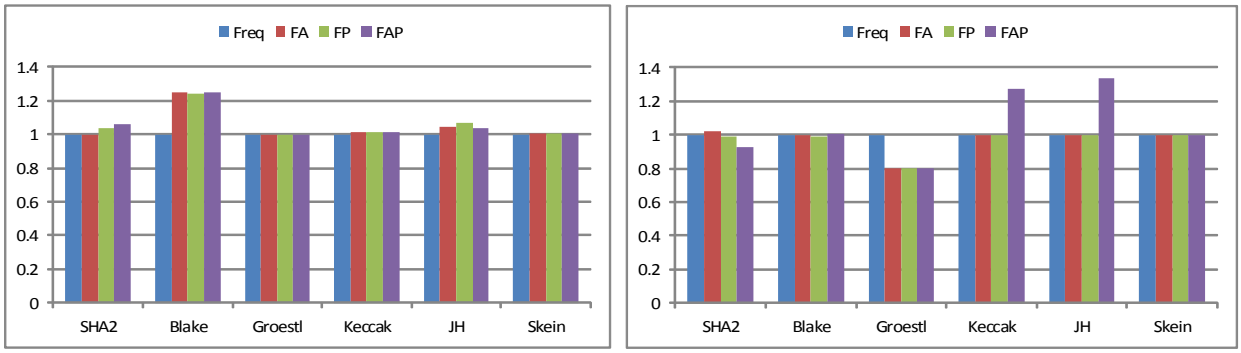
When we compare the results for constraint of "Freq" and "FP", the conclusion that they have the uniform trend of variation can be drawn. That is the power value for constraint of "FP" is always smaller than the one with a "Freq" constraint. This is in contrast to the "FAP" constraint, which does not have this kind of uniformity. Although the tool (referred to Synopsys Design Compiler) also pays efforts on the power optimization, it applies three constraints including frequency, area, and power in that order of priority.

- **Performance in Throughput**

Figure 5.15 shows the throughput variation between different constraints.

The more constraints applied to the design the better performance in the throughput can be concluded by comparing the relative values for RTL architecture of VT. That means with more constraints applied to the design, more efforts will be paid to get the best netlist as the output.

While comparing the relative values for the RTL architecture of GMU, the conclusion that there is no uniformity at all at this stage can be found. For example, for Keccak and JH, the throughput for "Freq", "FA" and "FP" almost remains the same, while the one for "FAP" has a increase around 30%. That means the library has better standard cells for all combinations of the constraints specially suitable for Keccak and JH architecture.

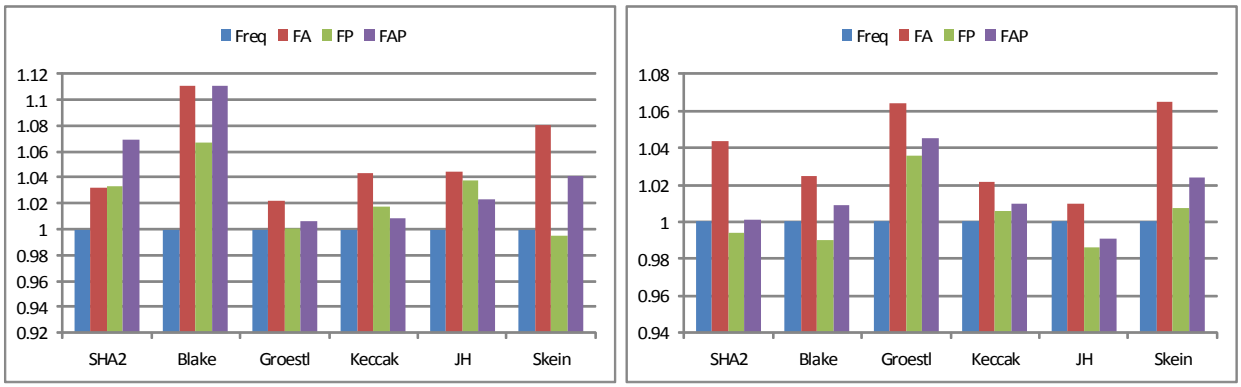


(a) VT (a) GMU

Figure 5.8: Throughput performance of different constraints

- **Performance in Throughput/Area**

Figure 5.9 shows the performance of throughput/Area amongst different constraints:



(a) VT (b) GMU

Figure 5.9: Throughput/Area performance of different candidates

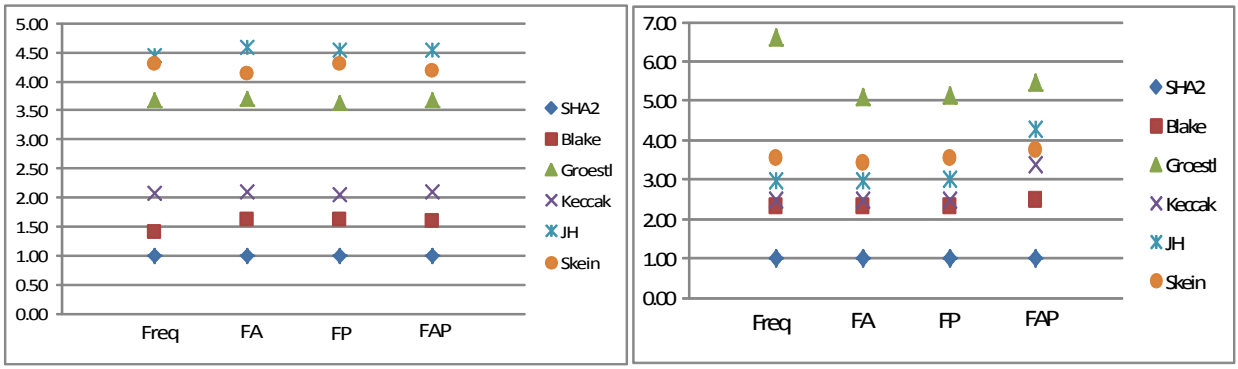
When comparing the relative value of "Freq" and "FA" constraints, the conclusion that they have a uniform trend of increase for both RTL of VT and GMU can be drawn. While comparing the relative value of "FP" and "FAP" to "Freq", we can come to the conclusion that they do not have an overall uniformity because the Design Compiler has to spend some extra effort on the optimization for power. To some extent, this would sacrifice the optimization performance for area.

Inter Comparison

Inter comparison compares the variation amongst different candidates with a specific constraint. Four constraints have been applied on the current design. For the convenience of the comparison, it is necessary to conduct the normalization to the comparison base, which is the winner of SHA2 hash function competition. Table A.2 has summarized the normalization to the comparison base (SHA2) for inter comparison.

- **Performance in Area**

Figure 5.10 shows the area ranking among candidates for each constraint. The relative relationship in quantity between candidates can be easily found by observation:



(a) VT (b) GMU

Figure 5.10: Area comparison between candidates

For the architecture of VT, the area of all SHA3 finalists are larger than that of SHA2. While for the architecture of GMU, only Keccak has a larger area than SHA2, which is almost 3 times than the other candidates. That means much more efforts on area has been spent on architecture of GMU.

In order to show the relative variation between each pair of candidates, the times of variation between candidates has been defined, which can be seen in Figure 5.11. The variation at the "Freq" constraint is set as the example to the definition of the relative variation times.

Figure 5.13 shows the relative variation in times between each pair of candidates in order to see the relative variation between finalists in area.

VT's architecture has an average variation of around 1.85 times. The largest variation between candidates is between JH and Blake, which is around 3.15 times. While the smallest variation between candidates is between JH and Skein, which is around 1.03 times.

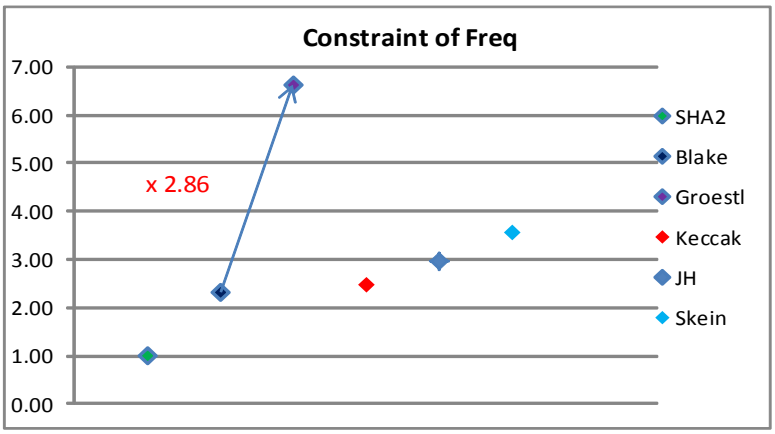
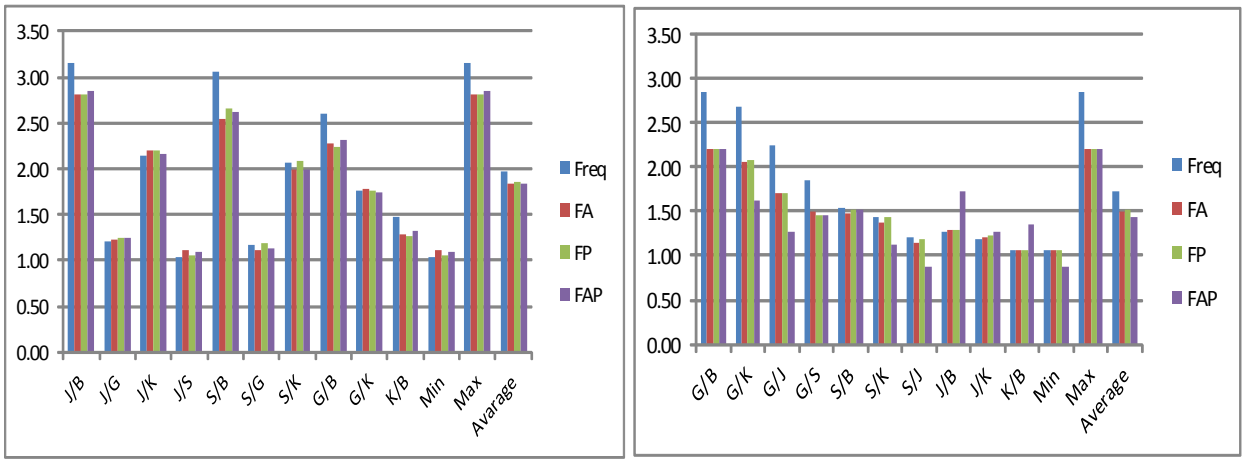


Figure 5.11: Definition of variation between candidates

GMU's architecture has an average variation of around 2.8 times. The largest variation between candidates is between Keccak and Skein, which is 6.99. While the smallest variation between candidates is between Groestl and JH, which is 1.00 times.



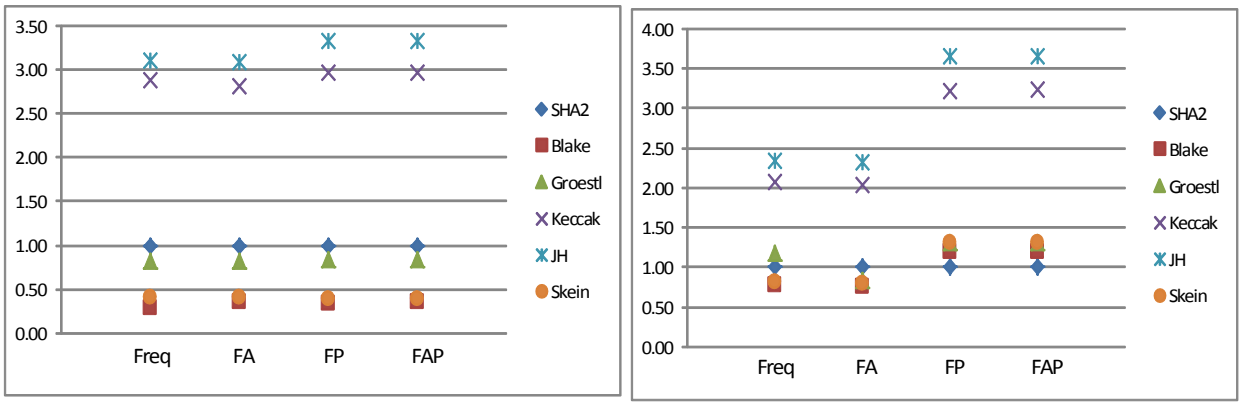
(a) VT (a) GMU

Figure 5.12: Relative area variation

- **Performance in Power**

Figure 5.13 shows the ranking of the power consumption among the finalists.

For VT's architecture, the power consumption for the candidates of Keccak and JH is far greater than the other candidates. Blake has the least power consumption among the finalist.

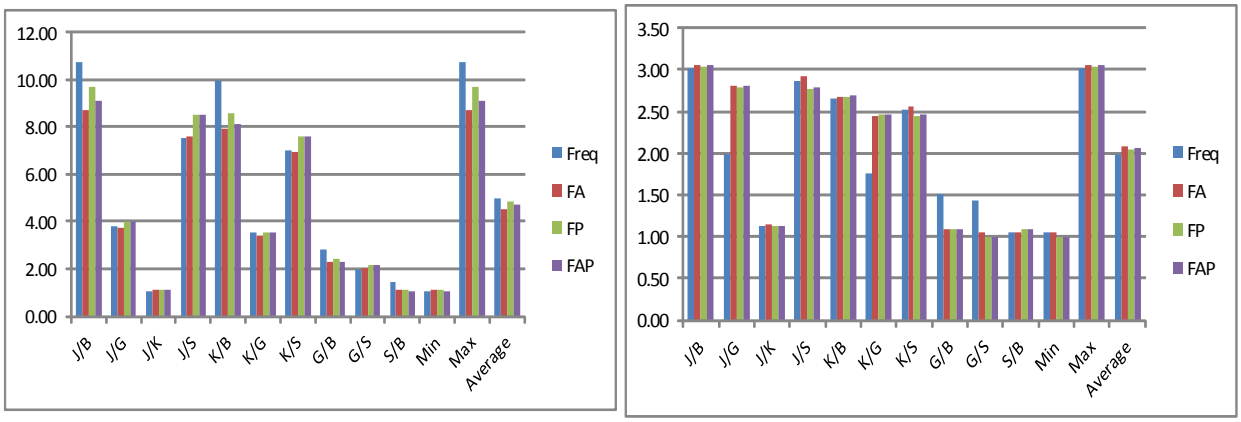


(a) VT (b) GMU

Figure 5.13: Power comparison between candidates

For GMU's architecture, Keccak and JH also have the power consumption a lot more the other candidates. For the other candidates, they competing with each other at the same level of power consumption, which is around the comparison base.

Figure 5.15 shows the times of relative variation in power between candidates.



(a) VT (b) GMU

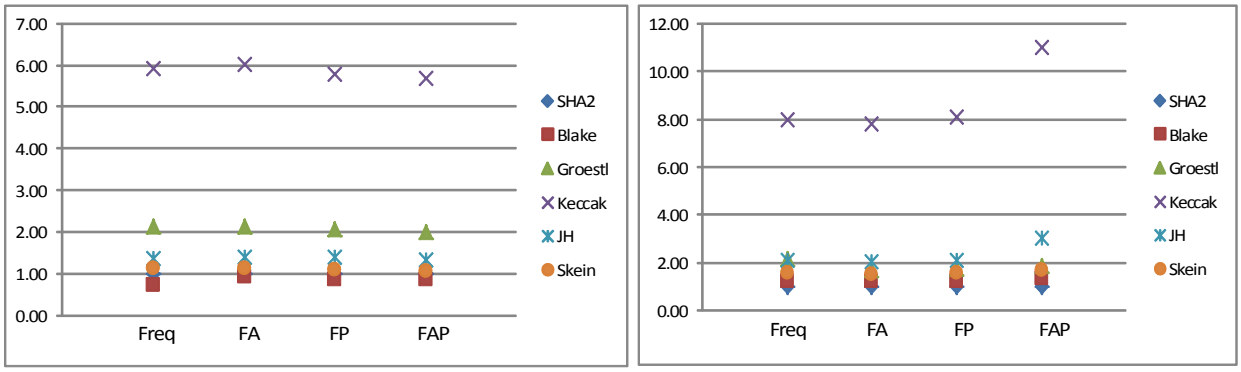
Figure 5.14: Relative Power variation

For the design of VT, in Figure 5.14, the maximum variation for all constraint is around 9, while the minimum variation is close to the value of comparison base. The variation between JH and Blake, Keccak and Blake is relatively larger than other pairs.

For the design of GMU, the maximum variation for all constraint are almost the same, which is around three times. The rule also applies to the minimum and average variation.

- **Performance in Throughput**

Figure 5.15 shows the ranking for the performance of throughput among the candidates. An overall concept of ability of processing data input on the finalists can be seen:

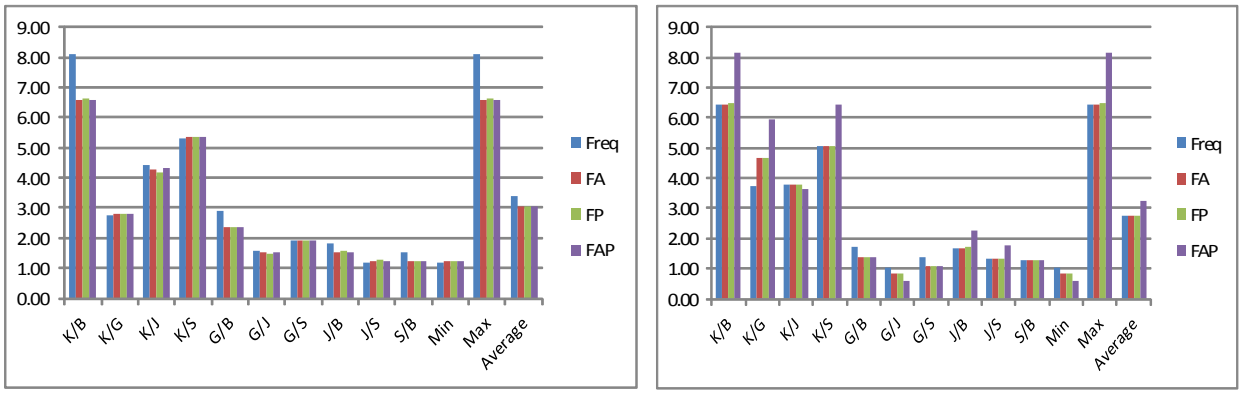


(a) VT (a) VT

Figure 5.15: Throughput comparison between candidates

In Figure 5.15, it can been seen that there is a huge gap between Keccak and the other candidates. This situation is applied to both VT and GMU's architecture. That means, in spite of the architecture, the Keccak algorithm performs much faster than others in processing the data input.

Figure 5.16 shows the variation times for each candidate pair:

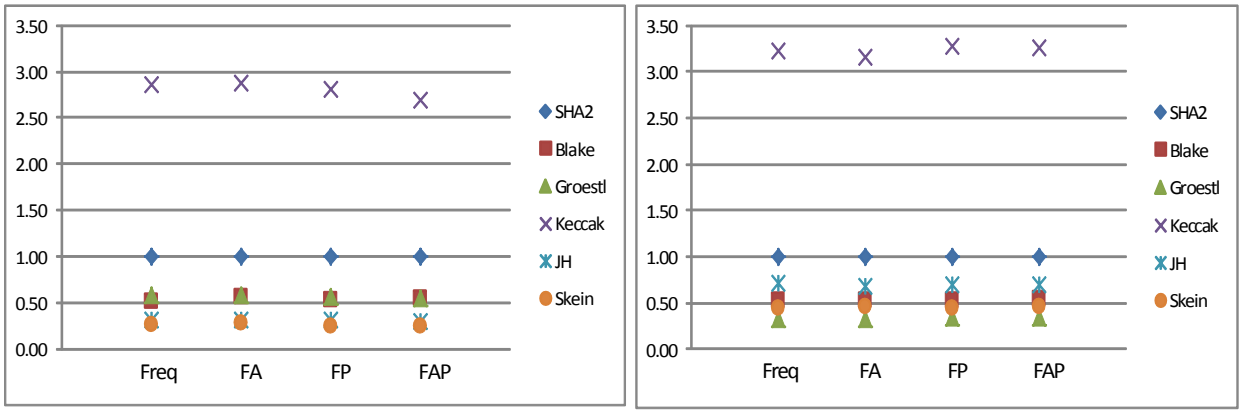


(a) VT (b) GMU

Figure 5.16: Relative throughput variation

- **Performance in Throughput/Area**

Figure 5.17 shows the ranking of performance in Throughput/Area:

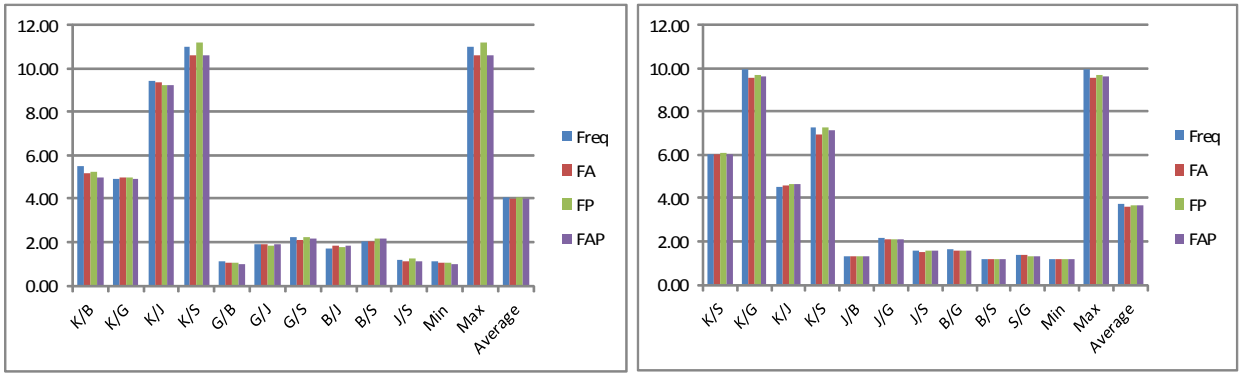


(a) VT (b) GMU

Figure 5.17: Throughput/Area comparison between candidates

The parameter, throughput/area, provides an evaluation of the ratio between the performance and the cost. In the aspect of throughput/area, the Keccak is far better than the others no matter what constraints are applied.

Figure 5.18 shows the times distribution of the variation for each candidate pair:



(a) VT (b) GMU

Figure 5.18: Relative throughput/Area variation

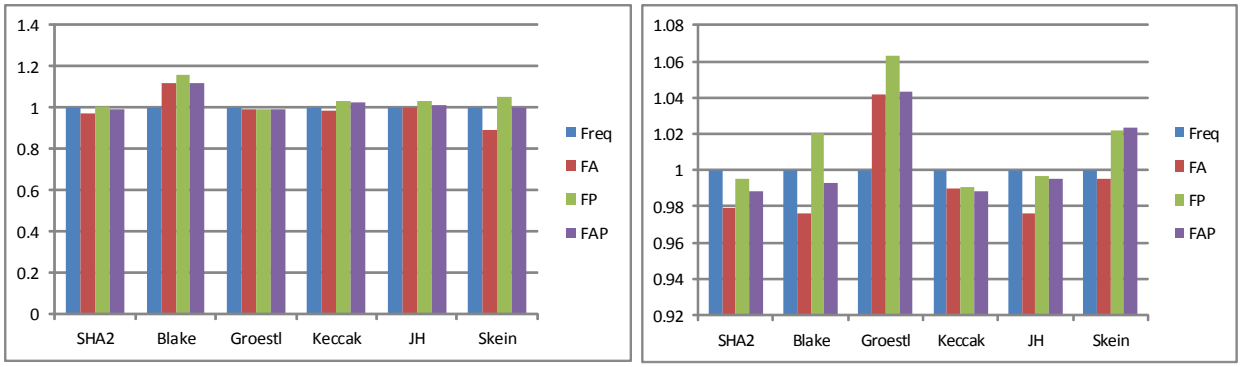
5.3.2 Implementation at Maximum Frequency

Intra Comparison

A summary of the normalization to the comparison base for intra comparison can be found in Table A.3.

- **Performance in Area**

Figure 5.19 shows the performance of area for different constraints, which can be seen directly by observation after the normalization.



(a) VT (b) GMU

Figure 5.19: Area performance of different constraints

The performance of area for design of GMU is more sensitive to constraints than that of VT when implementing at the maximum frequency.

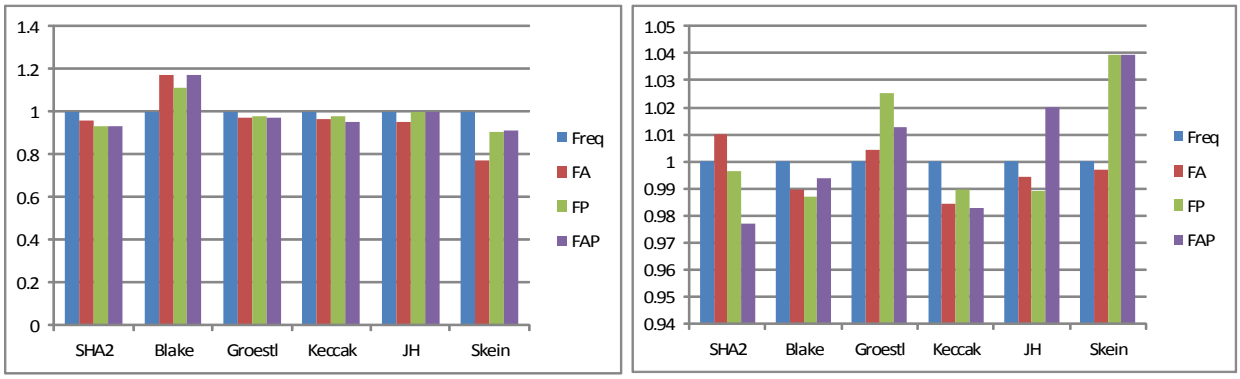
- **Performance in Power**

Figure 5.20 shows the performance of power between different constraints for each candidate:

The conclusion that the RTL design of GMU is more sensitive to constraints than that of VT when implementing at the maximum frequency.

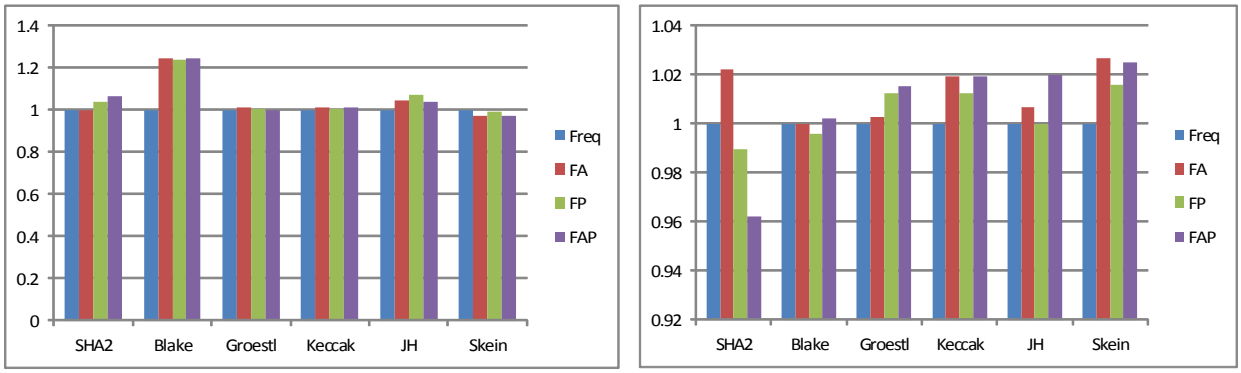
- **Performance in Throughput**

Figure 5.21 shows the performance of throughput between different constraints for each candidate.



(a) VT (b) GMU

Figure 5.20: Power performance of different constraints



(a) VT (b) GMU

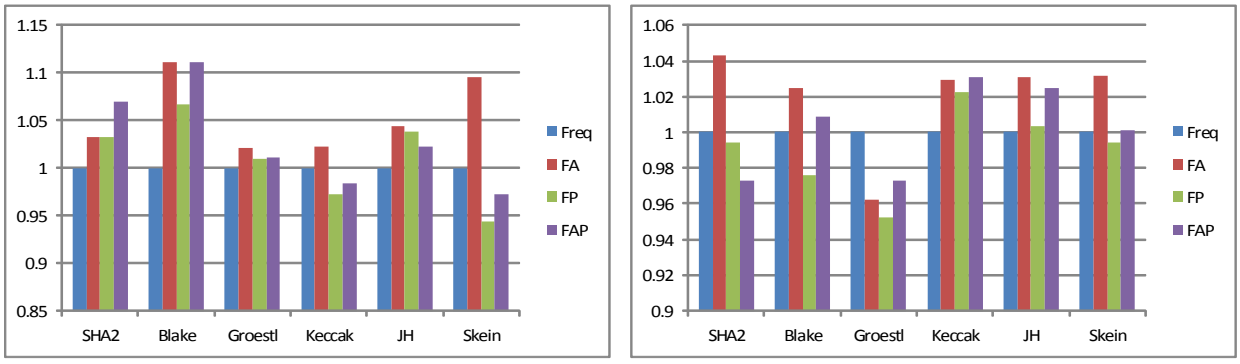
Figure 5.21: Throughput performance of different constraints

From what is presented in Figure 5.21, it can be seen that the RTL design of GMU is also more sensitive for the performance of throughput. Except for SHA2, all the finalists prefer the constraints of "FAP" in terms of throughput.

- **Performance in Throughput/Area**

Figure 5.22 shows the performance of area among different constraints:

Comparing to the performance in the area, power and throughput, the througput/area seems to be much more sensitive to constraints.



(a) VT (a) GMU

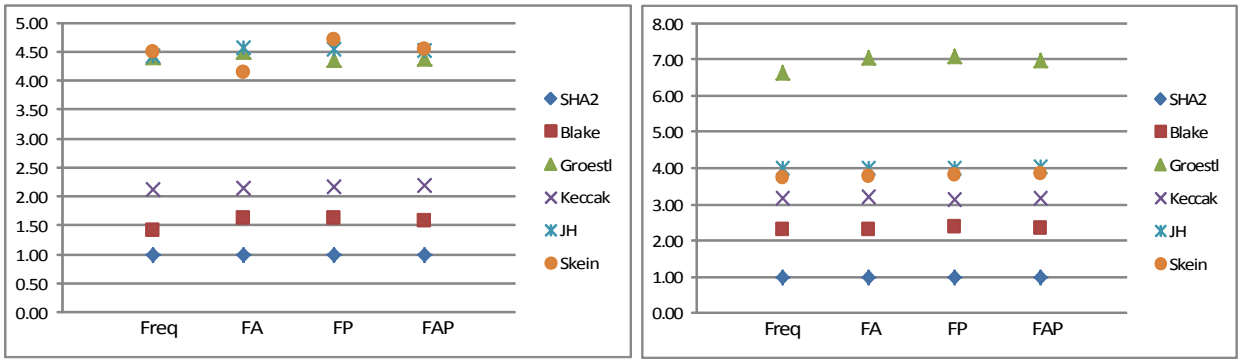
Figure 5.22: Throughput/Area performance of different constraints

Inter Comparison

Table A.4 shows the normalization for all the candidates from VT and GMU.

- **Performance in Area**

Figure 5.23 shows the area variation between candidates for each constraint:



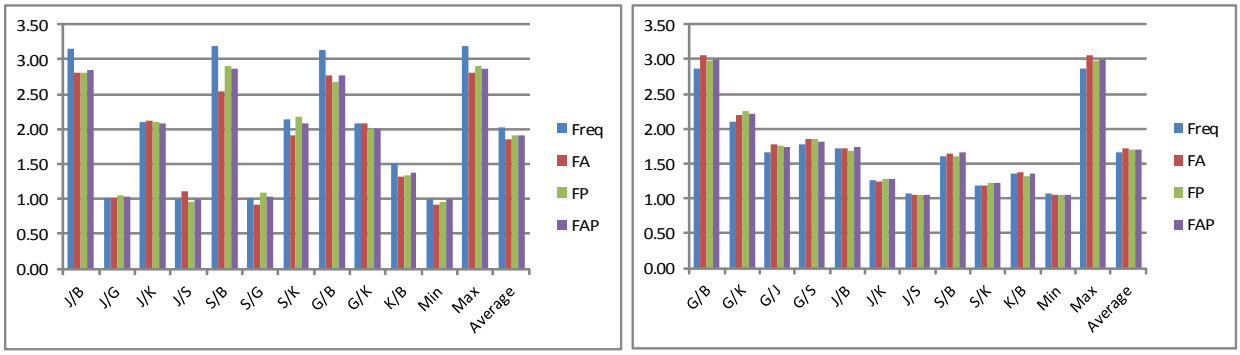
(a) VT (b) GMU

Figure 5.23: Area comparison between candidates

All the finalists have a large implementation area than SHA2. The largest one for VT is Skein, while the largest one for GMU is Groestl, which is around seven.

Figure 5.24 shows the area variation distribution between candidate pair for each constraint:

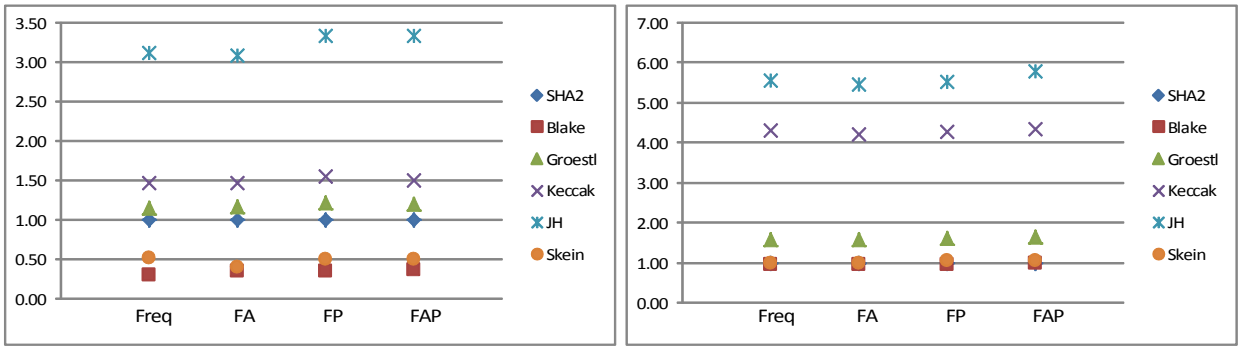
- **Performance in Power**



(a) VT (b) GMU

Figure 5.24: Relative Area variation

Figure 5.25 shows the power difference between candidates for each constraint:



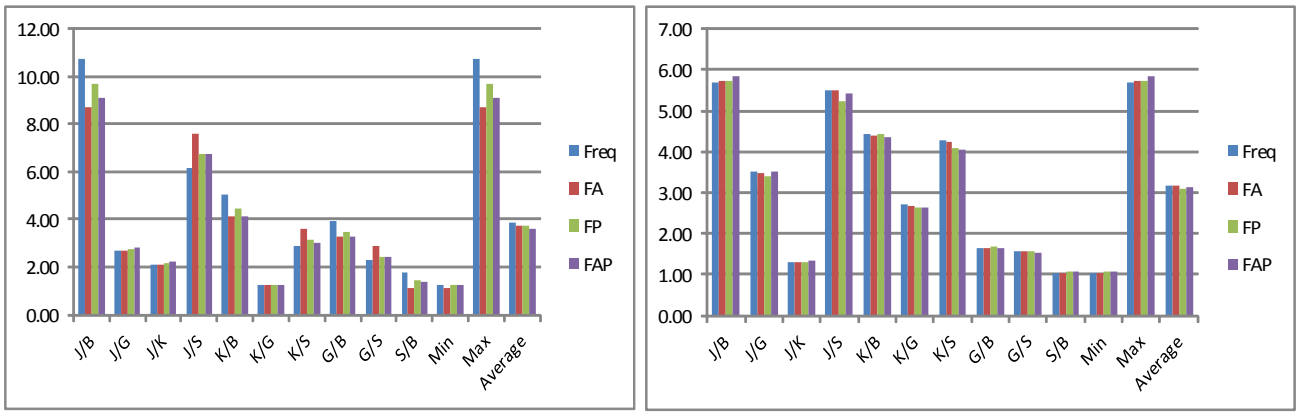
(a) VT (b) GMU

Figure 5.25: Power comparison between candidates

In Figure 5.25 (a), the power consumption of JH is far larger than others, which is up to three times as compared to SHA2. And Skein and Blake have even less power consumption than the comparison base.

In Figure 5.25 (b), no candidates has less power consumption than SHA2. And, at the same time, JH has the most power consumption, which is up six times of SHA2.

Figure 5.26 shows the distribution of power variation between candidate pairs for different constraints. Comparing the design of GMU, the variation between candidates of VT is slightly more sensitive to the constraints.

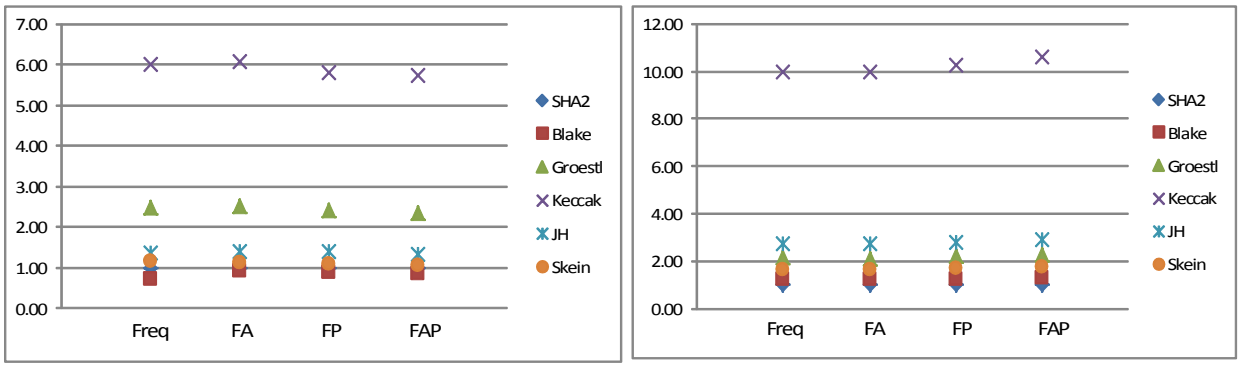


(a) VT (b) GMU

Figure 5.26: Relative power variation

- **Performance in Throughput**

Figure 5.27 shows the throughput difference amongst different candidates for each constraint:

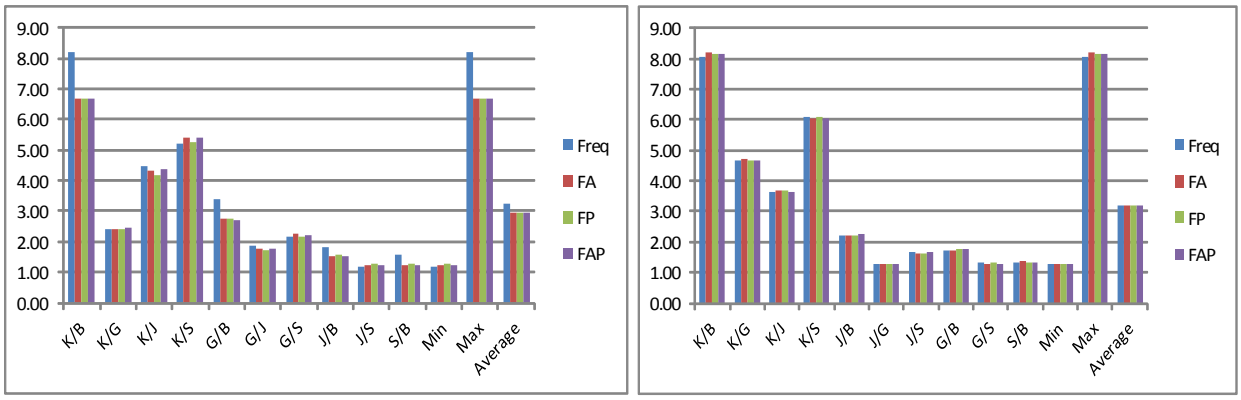


(a) VT (b) GMU

Figure 5.27: Throughput comparison between candidates

All the candidates except Keccak have a relatively close throughput compared to SHA2 for both VT and GMU. Keccak has a throughput almost up to six times to other candidates of VT and ten times to other candidates of GMU.

Figure 5.28 shows the distribution of the throughput variation between candidate pairs. The distribution in the above figure shows a relatively stable variation between candidates. That means all the candidates is not sensitive to the change of constraints for throughput.

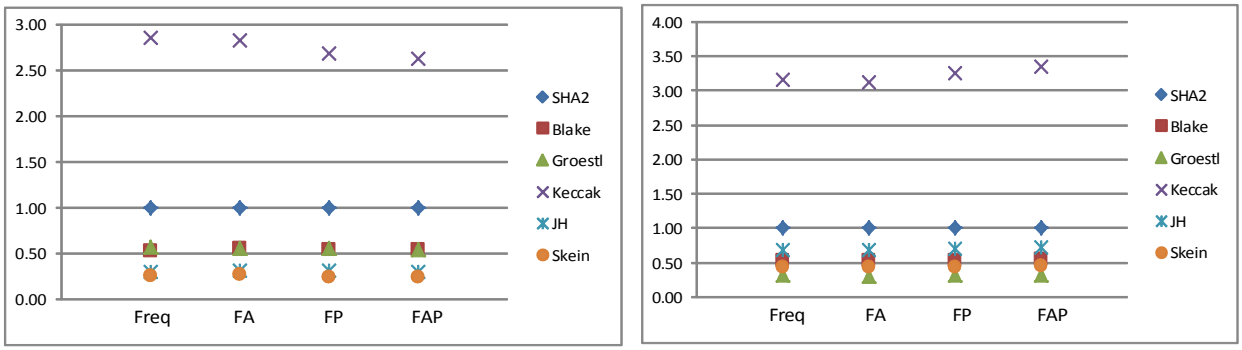


(a) VT (b) GMU

Figure 5.28: Relative throughput variation

- **Performance in Throughput/Area**

Figure 5.29 shows the throughput/area difference between candidates:

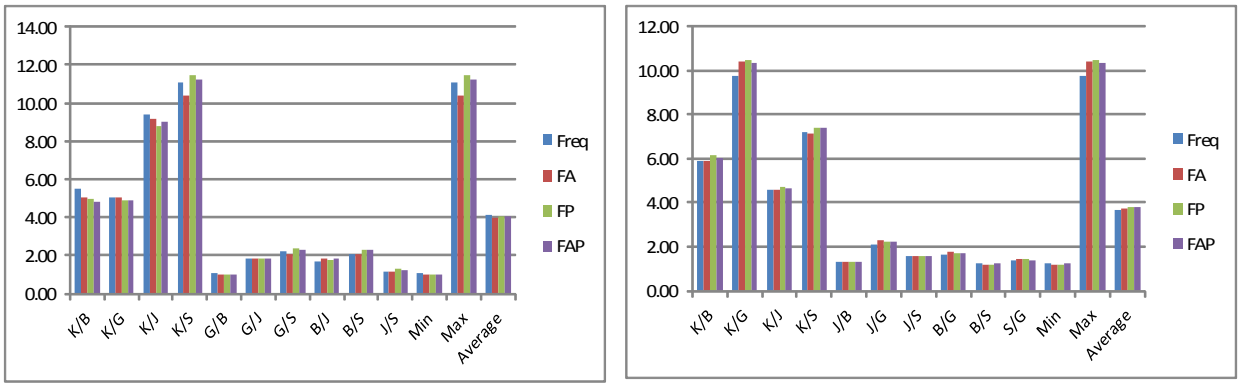


(a) VT (b) GMU

Figure 5.29: Throughput/Area comparison between candidates

The trend of variation almost remain the same as throughput except some candidates have a slight decrease in throughput/area compared to SHA2 due to an area increase.

The distribution of variation between candidates for throughput/area is shown in Figure 5.30.



(a) VT (b) GMU

Figure 5.30: Relative throughput/Area variation

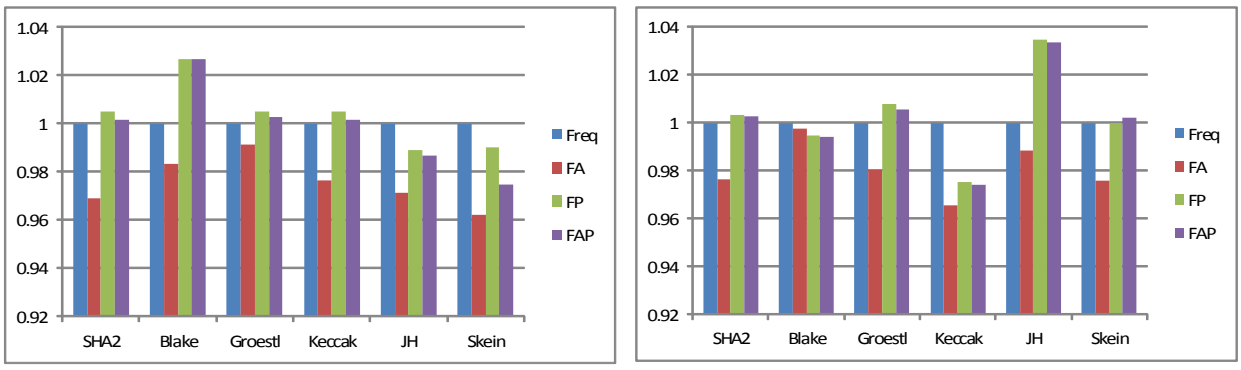
5.3.3 Implementation at Minimum Frequency

The implementation at the minimum frequency is the worst case the design can get. In this sub-section, the implementation at the minimum frequency will be explored.

Intra Comparison

Table A.5 has summarized the normalization to the constraint of "Freq" in order to see the effects of different constraints at the implementation of minimum frequency.

- **Performance in Area**



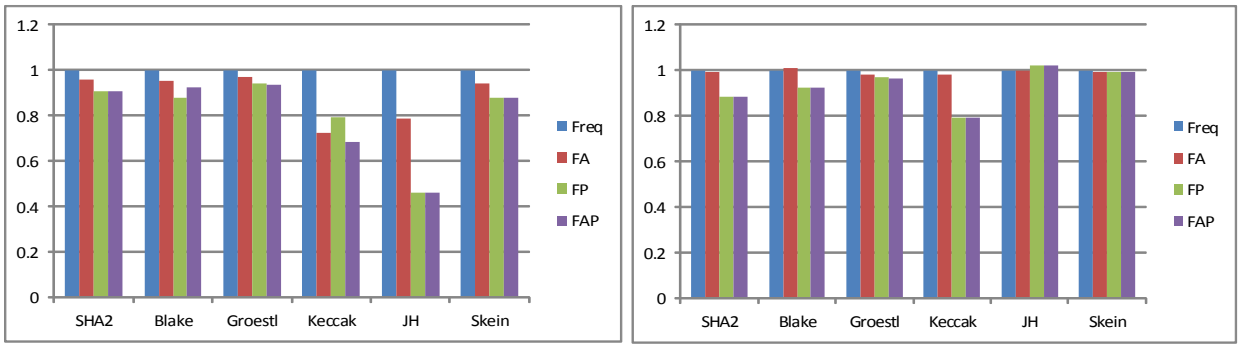
(a) VT (b) GMU

Figure 5.31: Area performance of different constraints

Figure 5.31 shows the performance of area amongst different constraints. With the area constraint applied, the area performance is also much better even at the implementation of the minimum frequency.

- **Performance in Power**

Figure 5.32 shows the performance of power between different constraints.

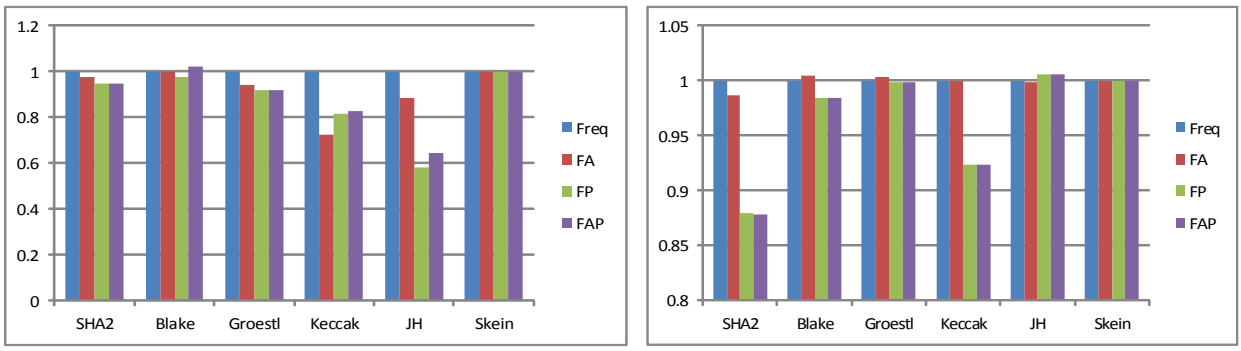


(a) VT (b) GMU

Figure 5.32: Power performance of different constraints

- **Performance in Throughput**

Figure 5.33 shows the performance of throughput between different constraints:



(a) VT (b) GMU

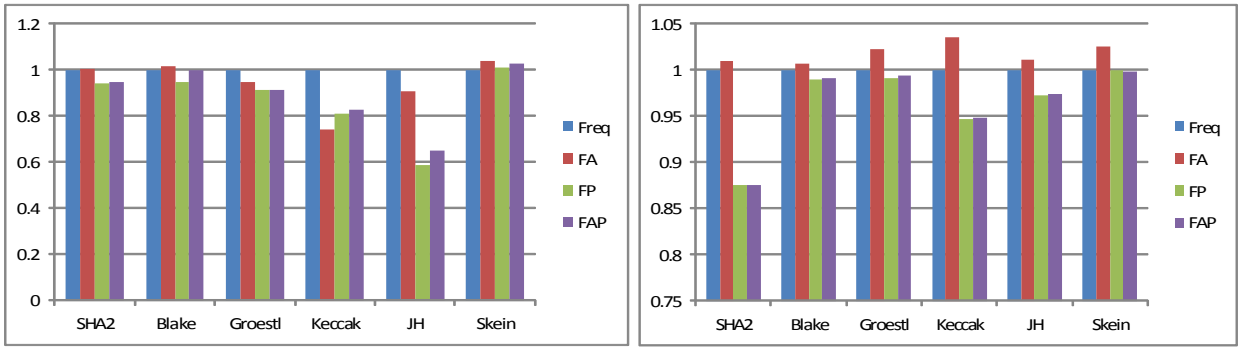
Figure 5.33: Throughput performance of different constraints

The performance of throughput is relatively stable except the candidate of JH and Keccak of VT. Keccak has a huge difference in throughput when "Freq" is applied. And the performance of JH shows that it will cause a decrease in throughput when any power effort is applied.

For GMU's Keccak, it also decreases its throughput when any power constraint is applied.

- **Performance in Throughput/Area**

Figure 5.34 above shows the performance of throughput/area between different constraints.



(a) VT (b) GMU

Figure 5.34: Throughput/Area performance of different constraints

The performance of throughput/area almost remains relatively the same as through.

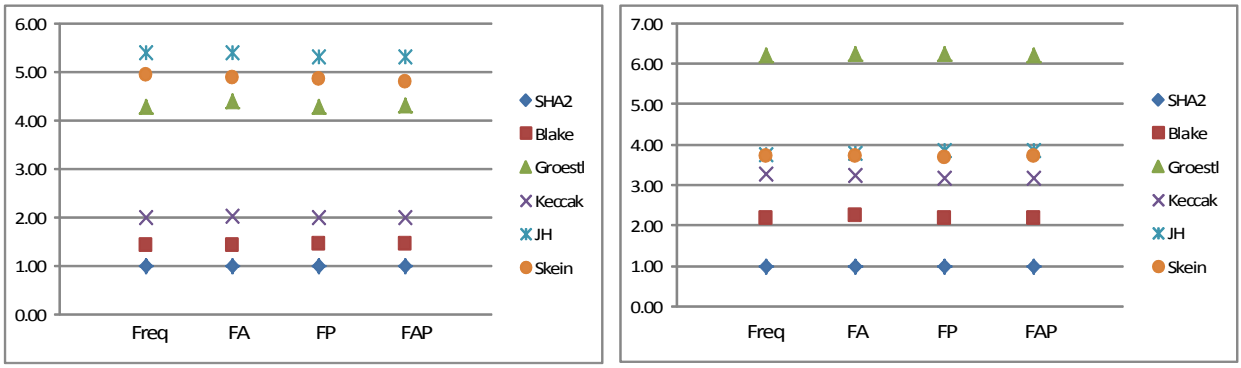
Inter Comparison

For the inter comparison of implementations at the minimum frequency, it is designed to see how the performance changes compared to other candidates at the minimum frequency. For convenience, the normalization to SHA2 is conducted as before, which is shown in the Table A.6.

- **Performance in Area**

Figure 5.35 shows the area variation between different candidates:

In Figure 5.35 (a), it shows all the finalists have a larger area than SHA2. JH has the largest area for all different constraints, which is around 5.2 times larger than SHA2. After that Skein has a very similar area as JH, and Blake has a quite good performance in area which is close to SHA2.

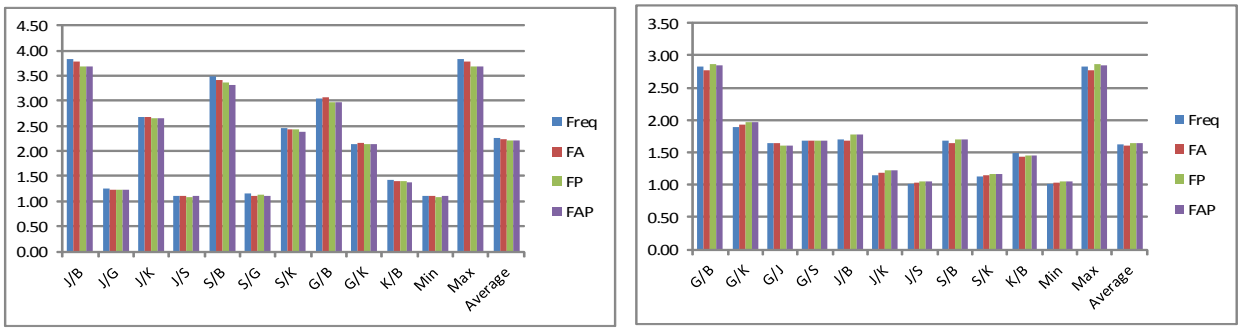


(a) VT (b) GMU

Figure 5.35: Area variation on different constraints

While in Figure 5.35 (b), we can also see that all the finalists have a larger area than SHA-2. Groestl has the largest area for all different constraints, which is about six times larger. After that, Skein and JH almost share the same characteristic curves, which is about 30% smaller than Groestl.

The distribution of the between candidate pairs for area is shown in figure 5.36.



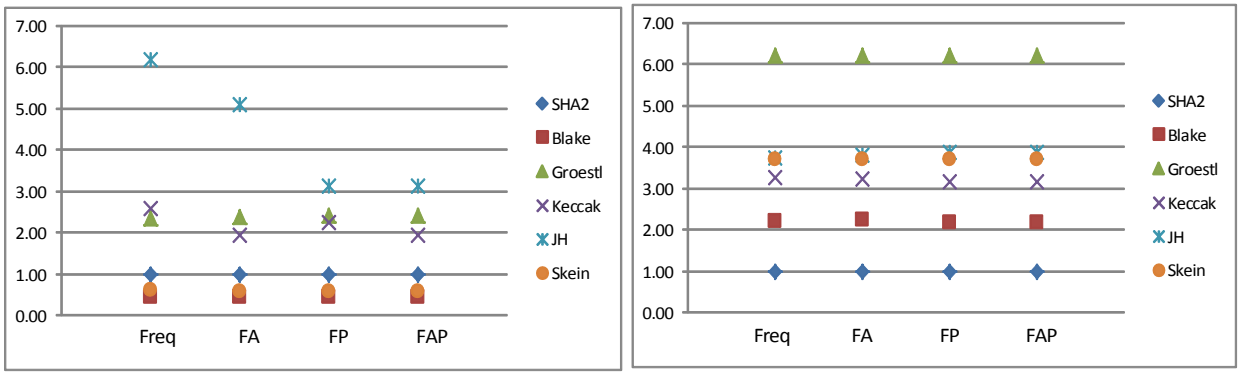
(a) VT (b) GMU

Figure 5.36: Relative area variation on different constraints

- **Performance in Power**

Figure 5.37 shows the power variation between different candidates.

In Figure 5.37 (a), all the candidates except JH keep relatively stable power for different constraints. JH is quite sensitive to the power constraints.

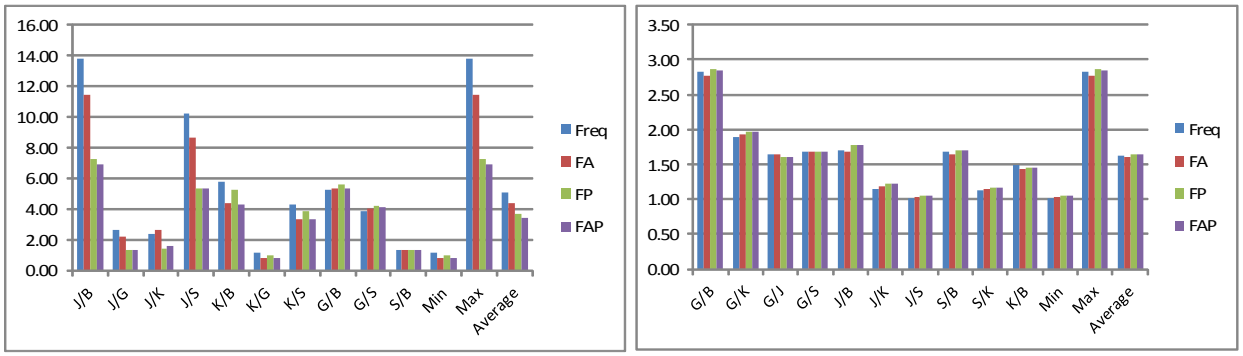


(a) VT (b) GMU

Figure 5.37: Power variation on different constraints

In Figure 5.37 (b), Greostl always has the largest area for different constraint sets. And, all the algorithms from GMU remain insensitive to all the different constraints.

The distribution of the between candidate pairs for area is shown in Figure 5.38.



(a) VT (b) GMU

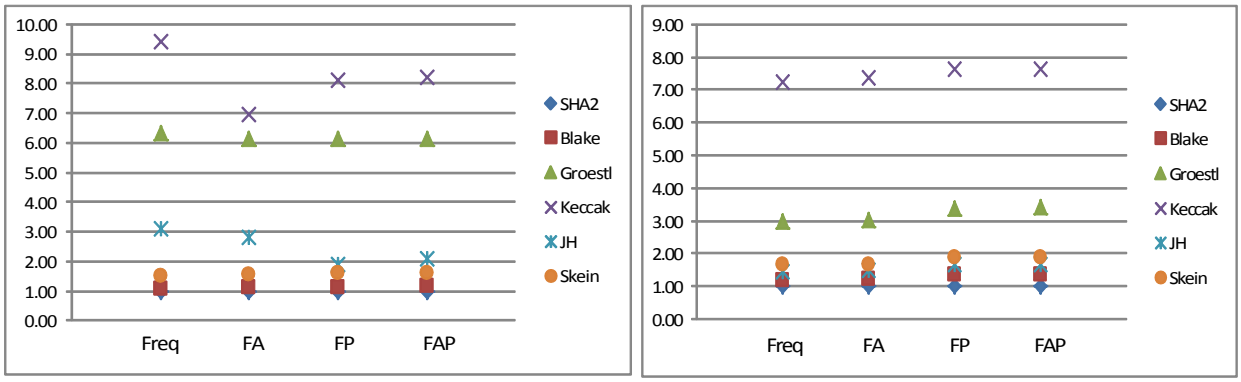
Figure 5.38: Relative power variation on different constraints

- **Performance in Throughput**

Figure 5.39 shows the power variation between different candidates for different constraints.

In Figure 5.39 (a), JH, Skein, and Blake have a relative close throughput performance, Keccak has the largest throughput.

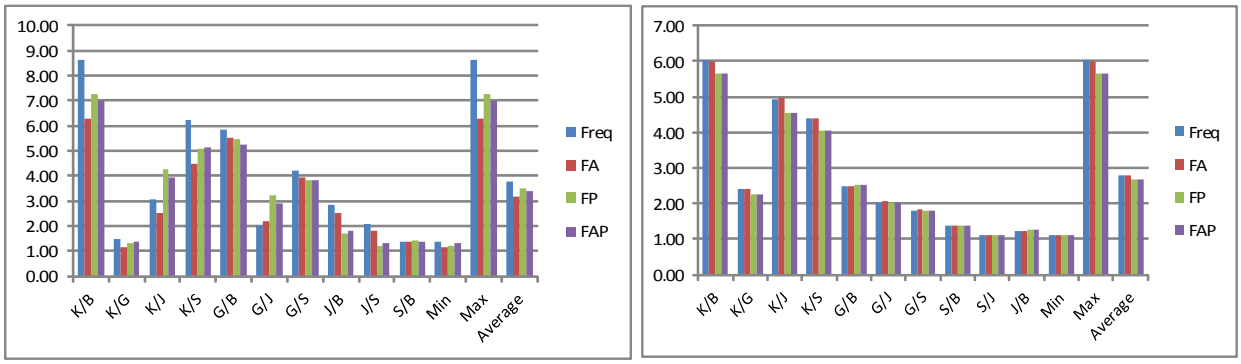
In Figure 5.39 (b), JH, Skein, Greostl, and Blake also have a relative close throughput performance, while Keccak has increased a lot for each constraints applied.



(a) VT (b) GMU

Figure 5.39: Throughput variation on different constraints

The distribution of the variation between different candidate pair for throughput is shown in Figure 5.40:



(a) VT (b) GMU

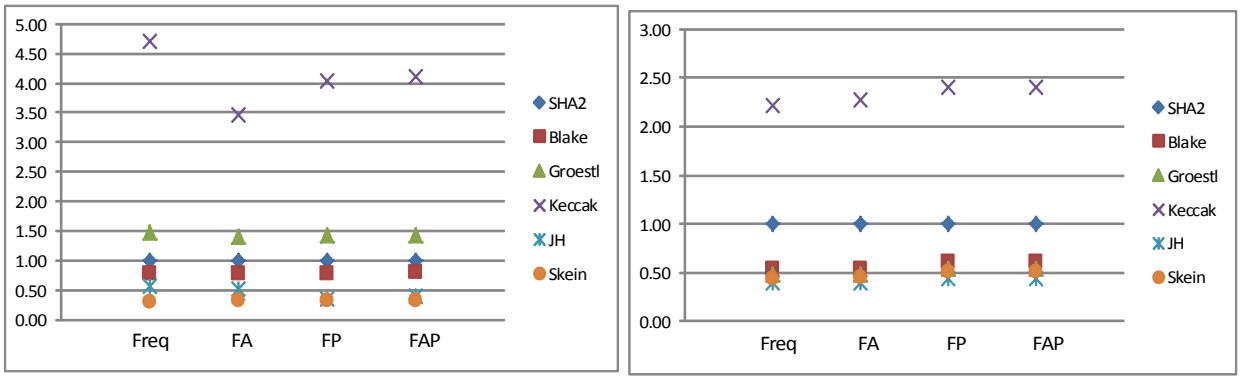
Figure 5.40: Relative throughput variation on different constraints

In the figure of variation distribution, the design from VT also shows a sensitivity to the constraints as compared to the insensitivity of the design from GMU.

- **Performance in Throughput/Area**

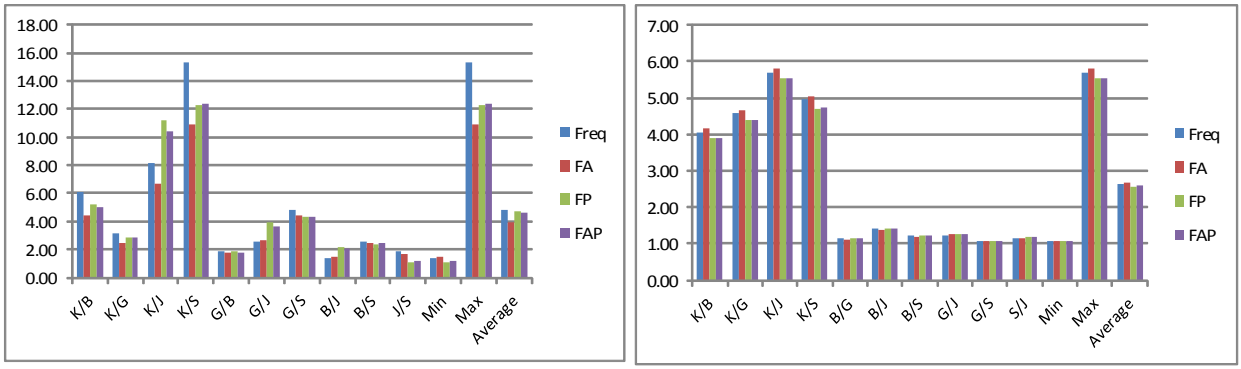
Figure 5.41 shows the power variation between different candidates for different constraints.

The quantity ranking of the candidates for each constraint remains the same, and the relative sensitivity also remains the same as throughput. This, in some extent, makes sense due to the throughput/area is related to the frequency (which can be expressed as throughput) and the area.



(a) VT (b) GMU

Figure 5.41: Throughput/Area variation on different constraints



(a) VT (b) GMU

Figure 5.42: Relative Throughput/Area variation on different constraints

The distribution of variation in throughput/area is shown in Figure 5.42, which can be predicted to be close to that of the throughput.

5.4 Library Effect Analysis

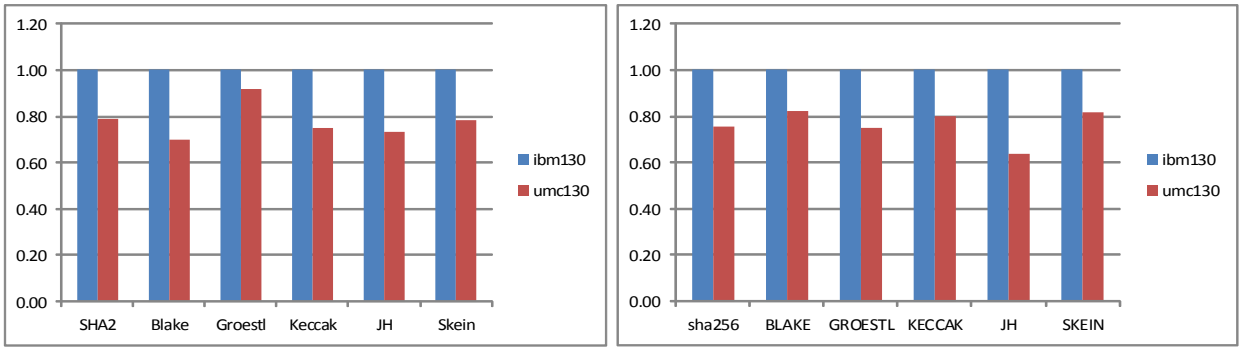
5.4.1 Implementation at Maximum Throughput/Area

Intra Comparison

In Table B.1, there is a summary for all the normalization to the IBM 130nm.

- **Performance in Area**

Figure 5.43 shows the difference of the performance in area:



(a) VT (b) GMU

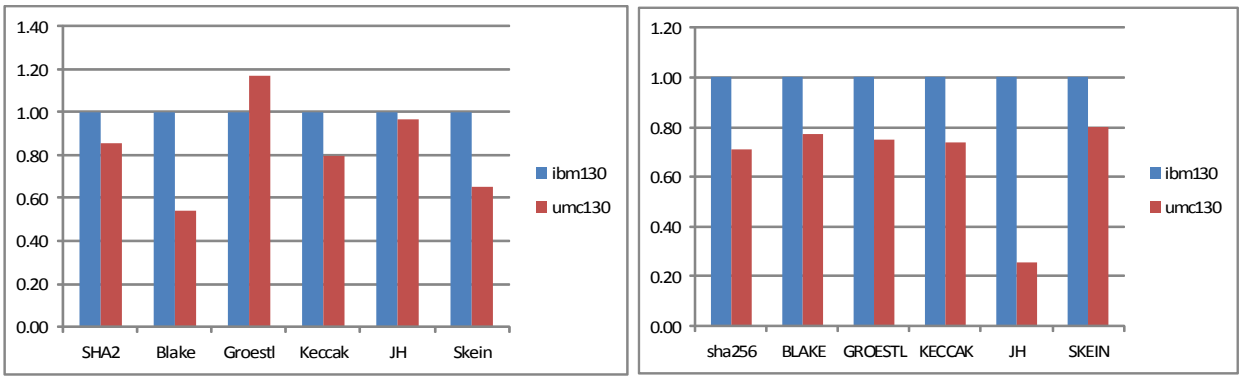
Figure 5.43: Performance of area using different libraries

The area with UMC library always has a better performance than that of IBM. The maximum difference between two libraries among VT's candidates is Blake, which shows a 30% difference. While among GMU's candidates, the maximum difference lies in the candidate of JH, which has a difference of 36%.

- **Performance in Power**

Figure 5.44 shows the performance in power. Only Groestl has more power consumption when implemented with UMC as compared with IBM. And, the maximum difference between these two libraries lies in the candidate of Blake, which has a 46% decrease.

Among GMU's candidates, all candidates have a better performance in power when implemented with UMC library. The maximum difference is JH, which has a huge decrease, 74%, compared to IBM library.

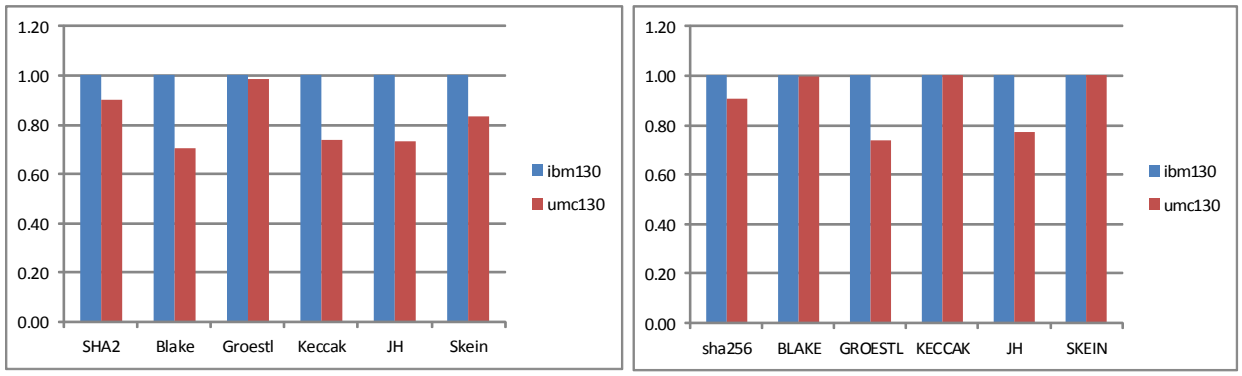


(a) VT (b) GMU

Figure 5.44: Performance of power using different libraries

- **Performance in Throughput**

Figure 5.45 shows the comparison of throughput:



(a) VT (b) GMU

Figure 5.45: Performance of throughput using different libraries

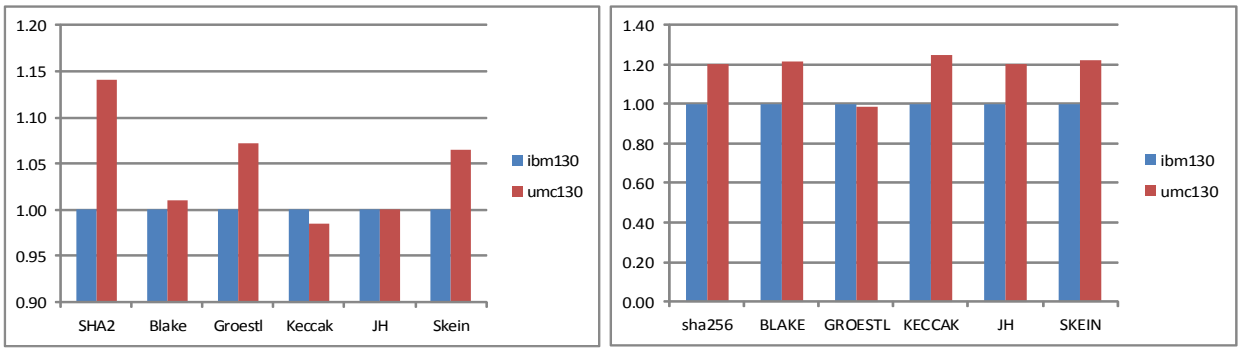
For VT's candidates, all candidates implemented with UMB library has less throughput than the one with IBM. That means IBM library has better standard cells designed at the performance of throughput. Groestl almost has the same performance on IBM and UMC. And the maximum difference between the two sets of implementations is Blake, which has a 30% decrease with UMC library.

For GMU's candidates, no candidate has a larger throughput with the implementation of UMC than the one with IBM. There is no effect on the performance of throughput for Blake, Keccak, and Skein, which means these three candidates will not be influenced by the choose

the different libraries as long as the same technology node is applied. And, the maximum difference between IBM and UMC is the candidate of Groestl, which has 26% decrease.

- **Performance in Throughput/Area**

Figure 5.46 is the comparison for the throughput/area:



(a) VT (b) GMU

Figure 5.46: Performance of throughput/area using different libraries

For VT's candidates, only Keccak has less throughput/area when implemented with IBM library. And, the maximum increase is from Greostl, which has 7% increase except SHA-2.

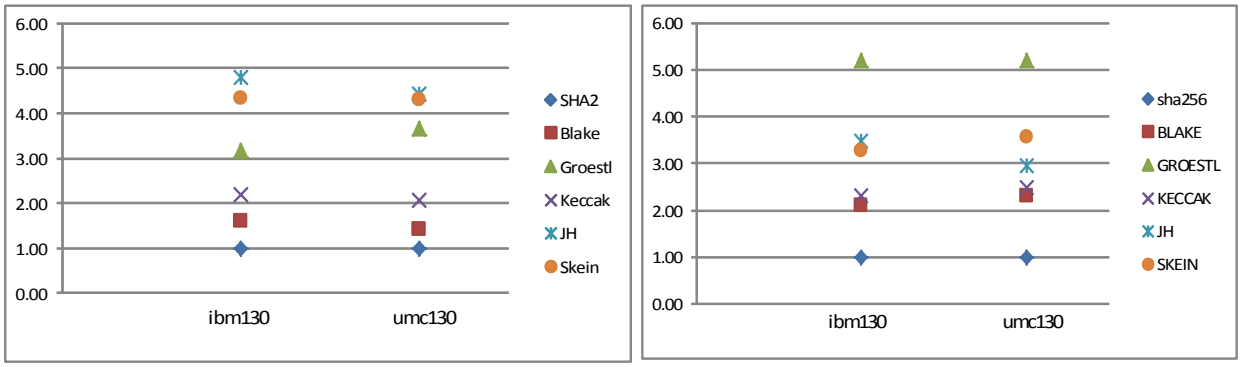
For GMU's candidates, except Greostl, all other candidates have a better performance when implemented with UMC library. The difference between candidates is larger than that of VT, with a 25% in Keccak.

Inter Comparison

Table B.2 has a summary of the normalization for all candidates from VT and GMU to SHA2. The comparison has been done mainly in area, power, throughput, and throughput/area, which can been seen in the following sub-sections.

- **Performance in Area**

Figure 5.47 shows the performance in area on IBM and UMC libraries. For VT's candidates, all the candidate have a larger implementation area for both IBM and UMC library than that of SHA2. The maximum difference is 4.79 times at JH for IBM and 4.44 times at JH for UMC. The minimum difference is 1.60 times at Blake for IBM and 1.41 times Blake for UMC.

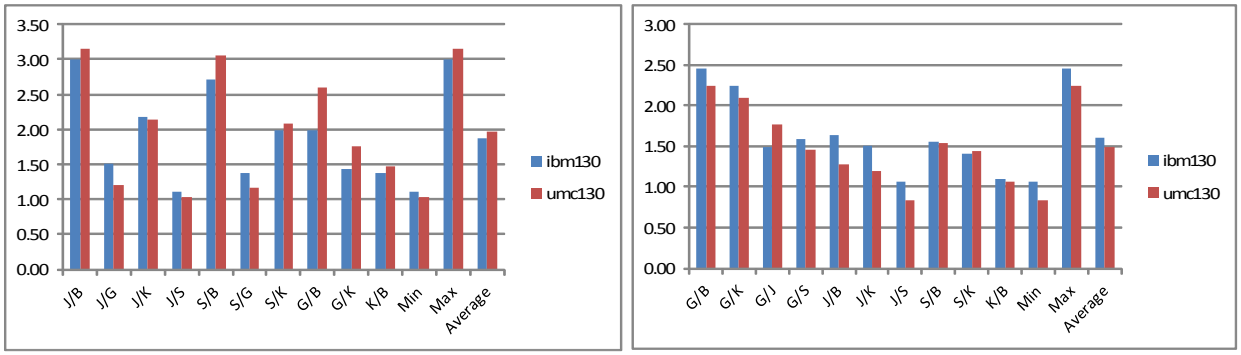


(a) VT (b) GMU

Figure 5.47: Area variation using different libraries

For GMU's candidates, the area performance has the same trend for both IBM and UMC library. And the maximum increase compared to SHA2 is 5.21 times at Greostl for both IBM and UMC libraries. The minimum increase is 2.12 times for IBM and 2.32 times for UMC.

The variation between candidates can also be found in Figure 5.48:



(a) VT (b) GMU

Figure 5.48: Relative area variation using different libraries

The maximum variation lies between JH and Blake for VT's candidates, which is around 3 times variation. While for GMU's candidates, the maximum variation is between Groestl and Blake, which shows around 2.5 times variation.

- **Performance in Power**

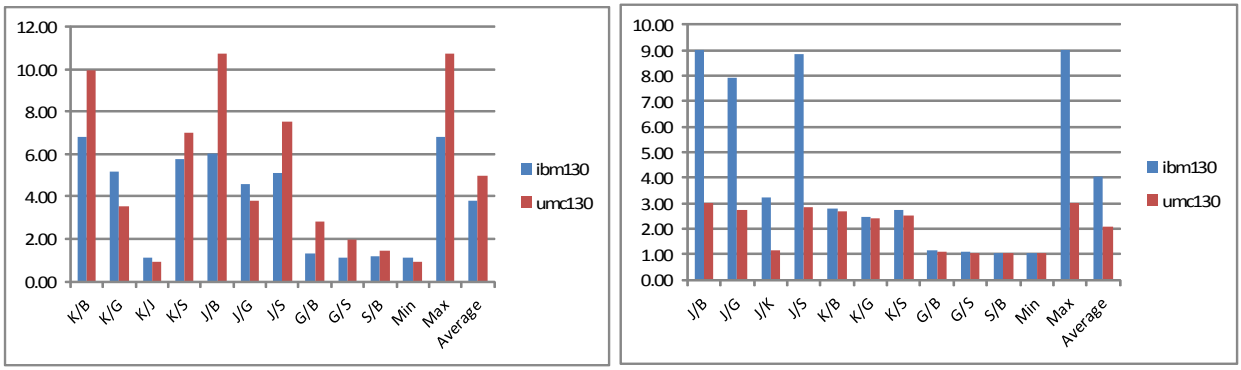
In Figure 5.49, the discussed relative difference can be found by comparison.

(a) VT (b) GMU

Figure 5.49: Power comparison using different libraries

Only Keccak and JH have worse power performance, which is 2 times more power consumption than the comparison base, SHA2 for VT's candidats. In GMU's candidates, JH has 5.4 times and 2.4 times more power consumption for IBM and UMC library respectively.

In Figure 5.50, the relative variation between candidates can be found:



(a) VT (b) GMU

Figure 5.50: Relative power variation using different libraries

For VT's candidates, the maximum variation for IBM library is between Keccak and Blake, which is around 7 times variation, and 11 times between JH and Blake for UMC library.

- **Performance in Throughput**

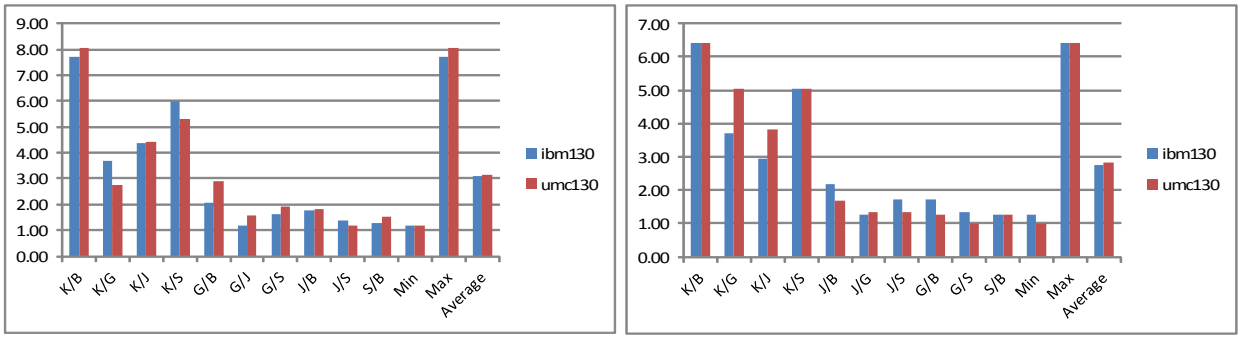
The comparison for the ability of processing data input, which is termed as throughput, is shown in Figure 5.51:

(a) VT (b) GMU

Figure 5.51: Throughput comparison using different libraries

Among all the candidates from both VT and GMU, all the candidates have a better performance in throughput than SHA2. Keccak has an about 7 times better performance than SHA2.

So, expectably, the relative variation between candidates is shown in Figure 5.52:



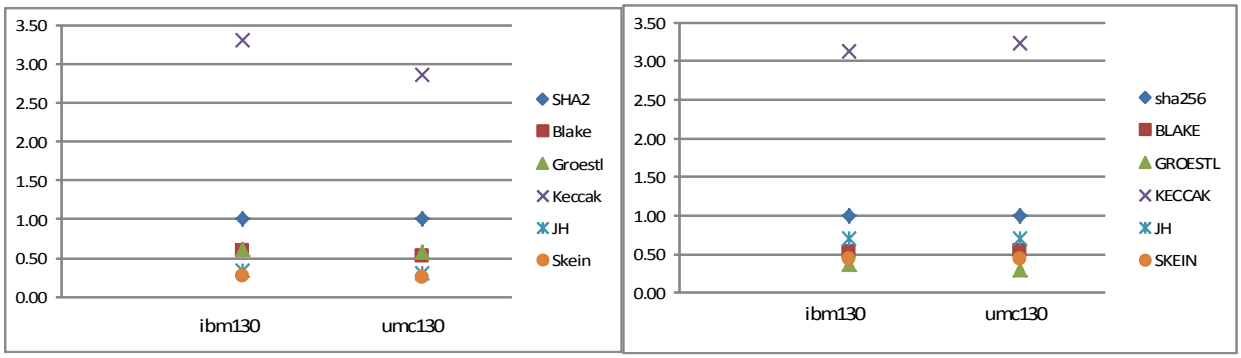
(a) VT (b) GMU

Figure 5.52: Relative throughput variation using different libraries

- **Performance in Throughput/Area**

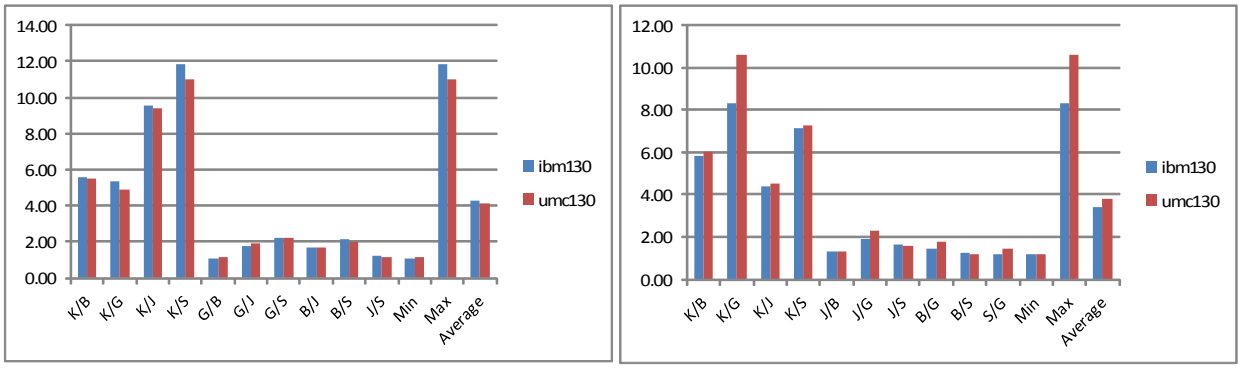
The performance of throughput/area can be found in Figure 5.53. Keccak has far larger throughput/area value as compared to the remaining candidates. This rule is applied to the candidates of both VT and GMU.

The variation between Keccak and the other candidates is much larger than the remaining candidate pair, which is shown in Figure 5.54.



(a) VT (b) GMU

Figure 5.53: Throughput/area variation using different libraries



(a) VT (b) GMU

Figure 5.54: Relative throughput/area variation using different libraries

5.4.2 Implementation at Maximum Frequency

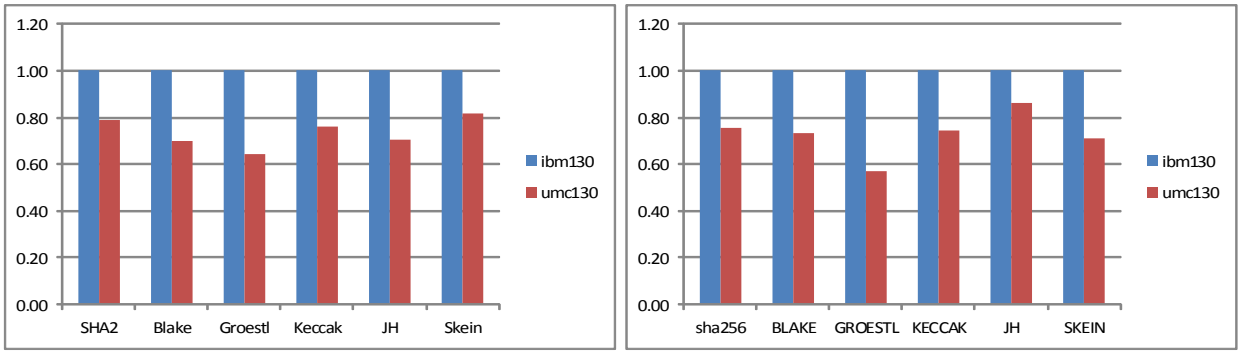
The point for maximum frequency is the same point with the maximum throughput, which shows the ability to process data input. So, the detailed analysis for the implementation at the maximum frequency becomes highly valuable.

Intra Comparison

The normalization of each value is performed as in Table B.3.

- **Performance in Area**

The area at the constraint of maximum frequency is shown in Figure 5.55:



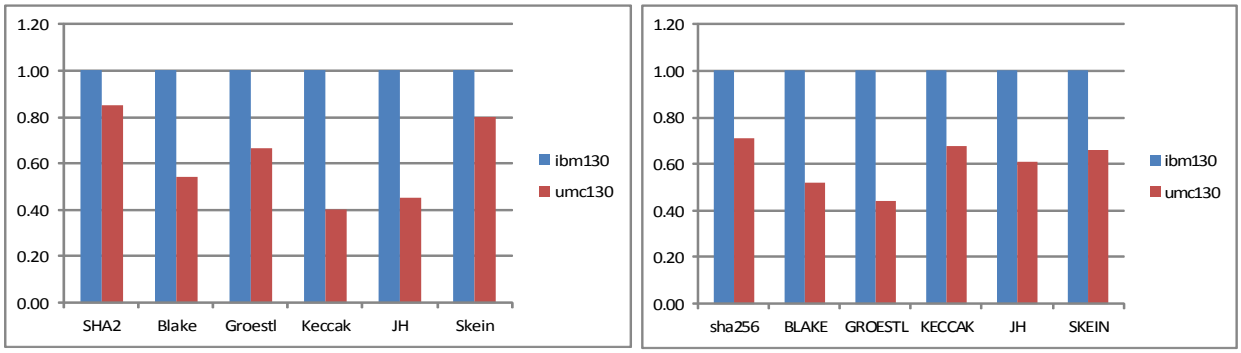
(a) VT (b) GMU

Figure 5.55: Area comparison using different libraries

The maximum decrease is Groestl, which has a decrease of 35% for VT's candidates, while for GMU's candidates, the maximum decrease is 43% with the same candidate, Groestl.

- **Performance in Power**

Figure 5.56 shows the comparison for power with different libraries.



(a) VT (b) GMU

Figure 5.56: Power comparison using different libraries

The performance of power for each candidate with UMC library is much better than that of IBM library. Keccak of VT has the maximum percentage decrease, 60%, while in GMU's candidates, Groestl has the maximum percentage decrease of 56%.

- **Performance in Throughput**

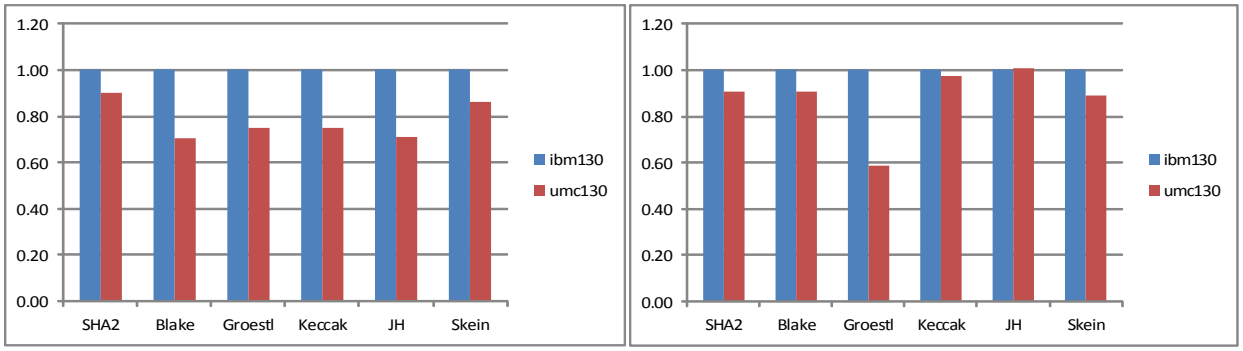


Figure 5.57: Throughput comparison using different libraries

The uniformity of decrease in the performance with UMC library is applied to all the candidates, which is shown in Figure 5.57. The maximum decrease for VT is 30% for Blake, while the one for GMU is 41% for Greostl.

- **Performance in Throughput/Area**

The Figure 5.58 shows the comparison in throughput/area for each candidate.

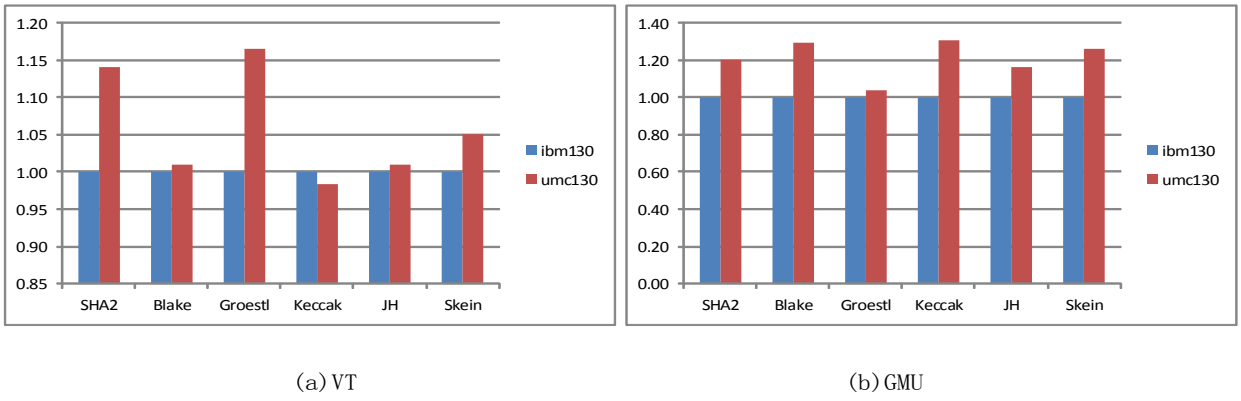


Figure 5.58: Throughput/area comparison using different libraries

Because of the impact from the area, although the throughput of each candidate implemented with UMC library is smaller than that of IBM lirary, the ratio between throughput to area with UMC library shows a much better performance. The Groestl of VT has a much higher throughput/area ratio, while for GMU, the candidate of Keccak shows the maximum difference.

Inter Comparison

The summary of the normalization to the implementation of SHA2 can be found in Table B.4.

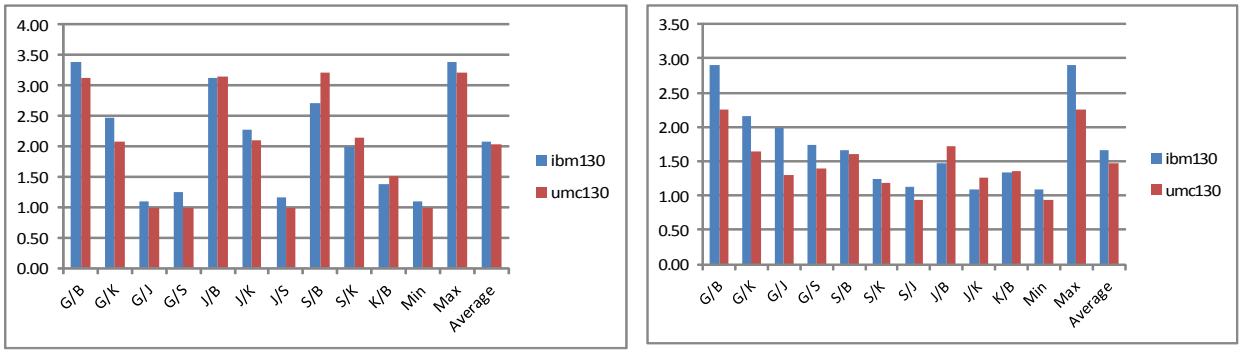
- **Performance in Area**

The performance of area at the constraint of maximum frequency is shown in Figure 5.59:

(a) VT (b) GMU

Figure 5.59: Area variation using different libraries

At this stage, all the candidates from VT and GMU have a larger area than that of SHA2. The relative variation between candidate pair is shown in Figure 5.48:



(a) VT (b) GMU

Figure 5.60: Relative area variation using different libraries

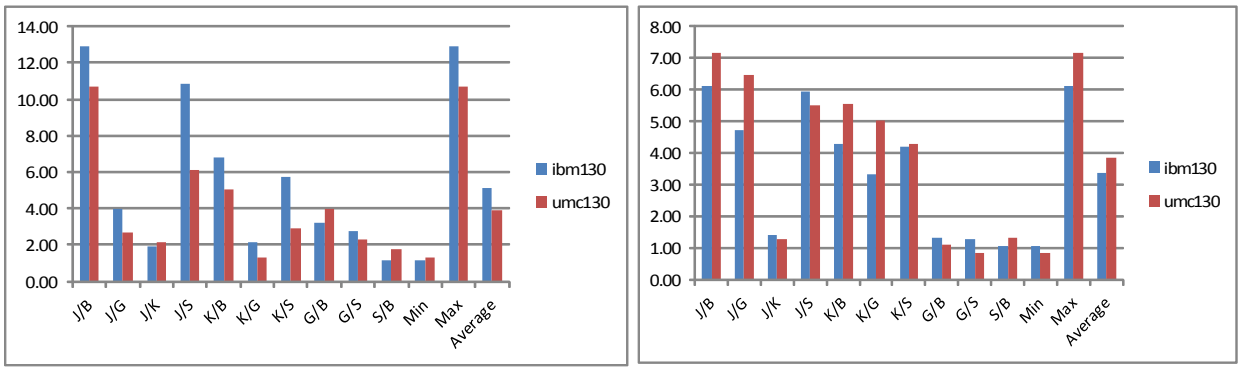
The Figure 5.48 is quite self-explained for the relationship between candidate pairs. The candidate pair of Groestl and Blake have the largest variation, which is 3 times for VT, and 2.5 times for GMU.

- **Performance in Power**

(a) VT (b) GMU

Figure 5.61: Power variation using different libraries

For both VT and GMU, JH has much more power consumption than others, that means, the algorithm of JH is designed to consume more power when implemented at the constraint of maximum frequency. After that it comes to the candidate of Keccak, which show little effect from the choice of libraries.

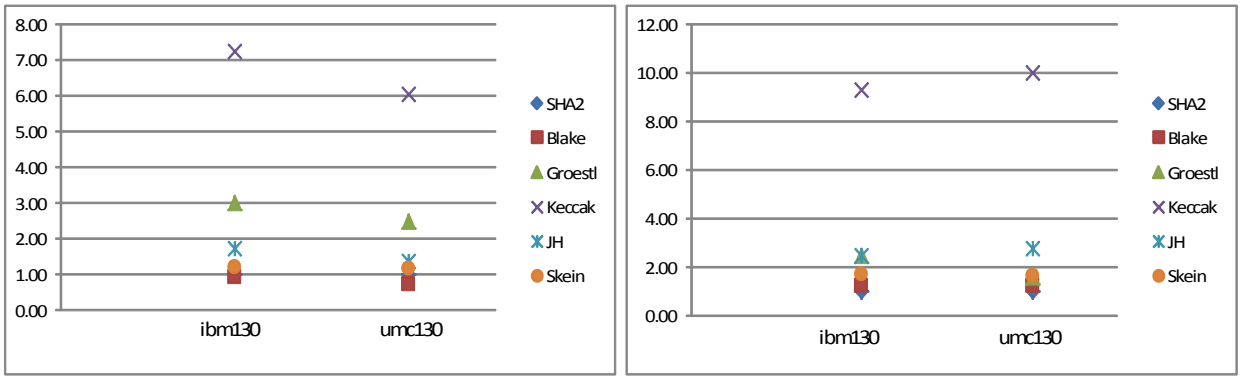


(a) VT (b) GMU

Figure 5.62: Relative power variation using different libraries

The largest variation among candidate pairs lie in the combination of JH and Blake. It indicates around 10 times variation for VT and about 7 times for GMU.

- **Performance in Throughput**

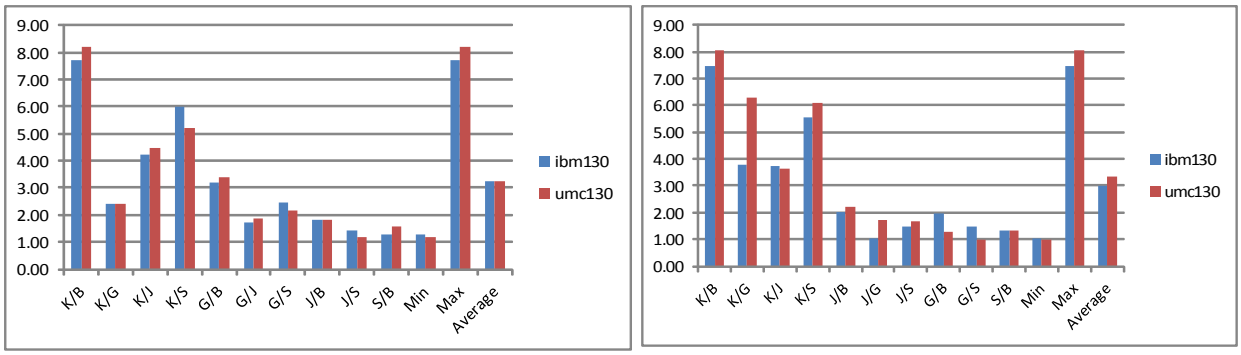


(a) VT (b) GMU

Figure 5.63: Throughput variation using different libraries

The performance of throughput in the above figure shows a simpler rule, which is Keccak has a far larger throughput no matter what library is under use, or what RTL architecture is being implemented. It shows an approximation of 6 times larger than SHA2 for VT, and even about 10 times larger than SHA2 for GMU.

The relationship for each candidate pair can show more information about the variation between candidates, which can be found in Figure 5.64:

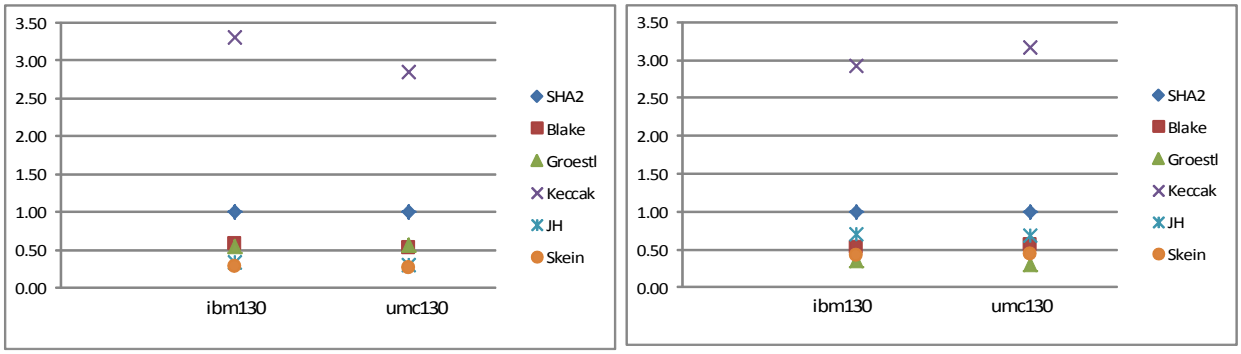


(a) VT (b) GMU

Figure 5.64: Relative throughput variation using different libraries

- **Performance in Throughput/Area**

When we consider the aspect of throughput/area, the same result with throughput can be obtained from Figure 5.65:

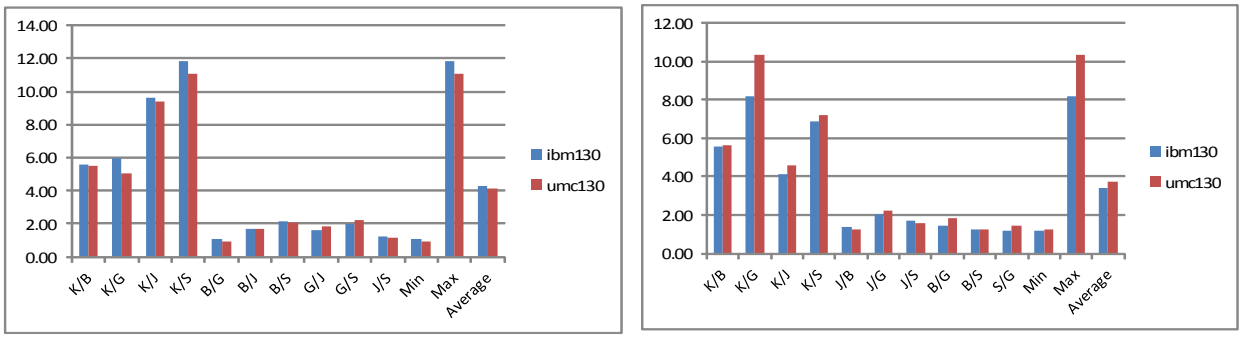


(a) VT (b) GMU

Figure 5.65: Throughput/area variation using different libraries

Among all the candidates, only JH have a better performance than SHA2 in terms of throughput/area, and is almost 3.5 times better than SHA2 for both VT and GMU.

The variation of each candidate pair is shown below:



(a) VT (b) GMU

Figure 5.66: Relative throughput/area variation using different libraries

As expected, the variation between JH and any other candidate is much larger than the remaining pairs, which can be proved in Figure 5.66.

5.4.3 Implementation at Minimum Frequency

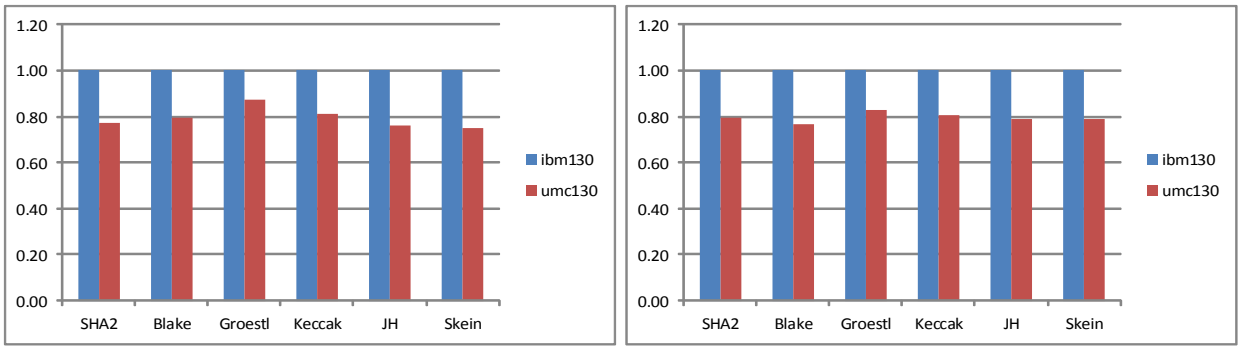
The implementation at the minimum frequency shows the implementation with the worst standard cells in terms of timing constraints. Through the analysis in this section, we can get to know more detailed performance when implementing an algorithm with the least effort for the optimization.

Intra Comparison

A summary for all the normalization to the comparison based is listed in Table B.5.

- **Performance in Area**

After the implementation at the constraint of minium frequency, the effect of the different choice on libraries is shown in Figure 5.67:



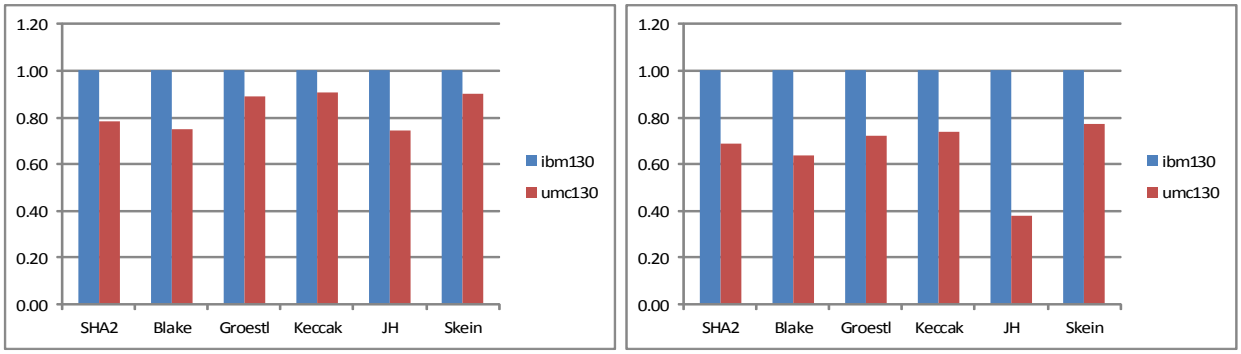
(a) VT (b) GMU

Figure 5.67: Area comparison within candidate using different libraries

All the candidate implemented with UMC 130nm library have less area than that with IBM 130nm library . And also, the percentage of decrease almost stay at the same line.

- **Performance in Power**

The performance of power share a uniformity of decrease when implemented with UMC library. And JH has the most percentage decrease compared with other candidates.

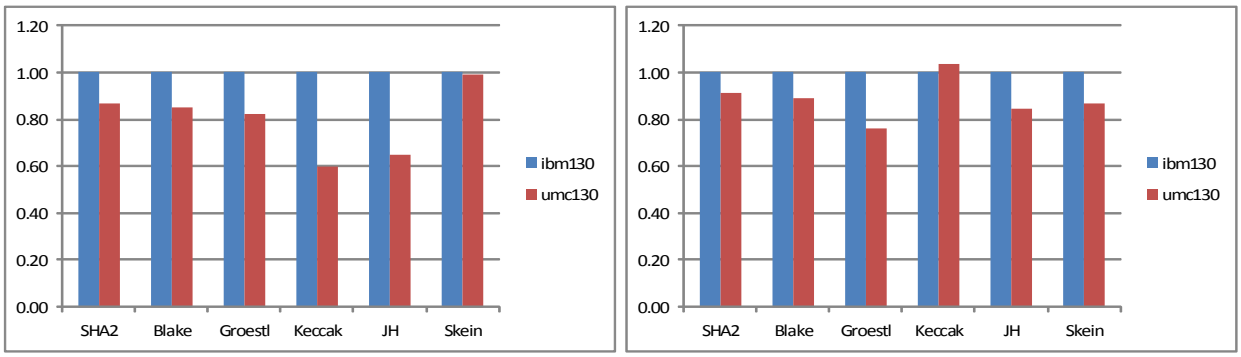


(a) VT (b) GMU

Figure 5.68: Power comparison within candidate using different libraries

- **Performance in Throughput**

The performance of throughput can be found in Figure 5.69:



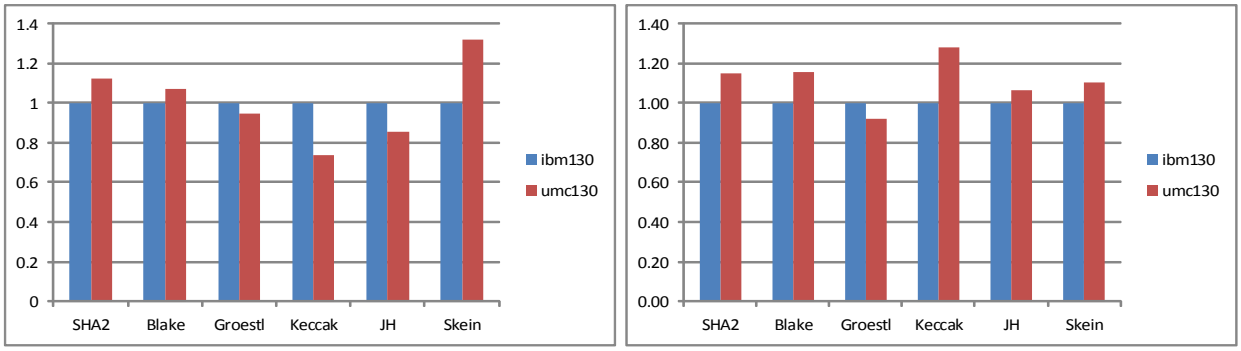
(a) VT (b) GMU

Figure 5.69: Throughput comparison within candidate using different libraries

All the other candidates except Keccak of GMU share the same uniformity of decrease when implemented with UMC library.

- **Performance in Throughput/Area**

Figure 5.70 shows the comparison for throughput/area. VT's candidates, Groestl, Keccak, and JH show a decrease when implemented with UMC library. While in GUM's candidates, only Groestl has shown the decrease. Skein of VT has a increase percentage of 32%, while Keccak of GMU shows the maximum increase percentage of 28%.



(a) VT (b) GMU

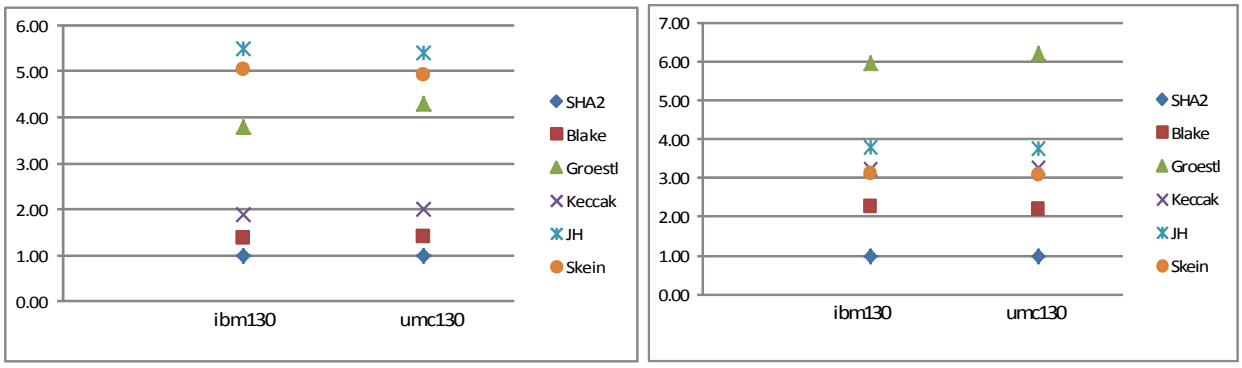
Figure 5.70: Throughput/area comparison within candidate using different libraries

Inter Comparison

The normalization to the comparison base, SHA2, is concluded in Table B.6.

- **Performance in Area**

Figure 5.71 shows the quantity relationship between candidates in one specific library.

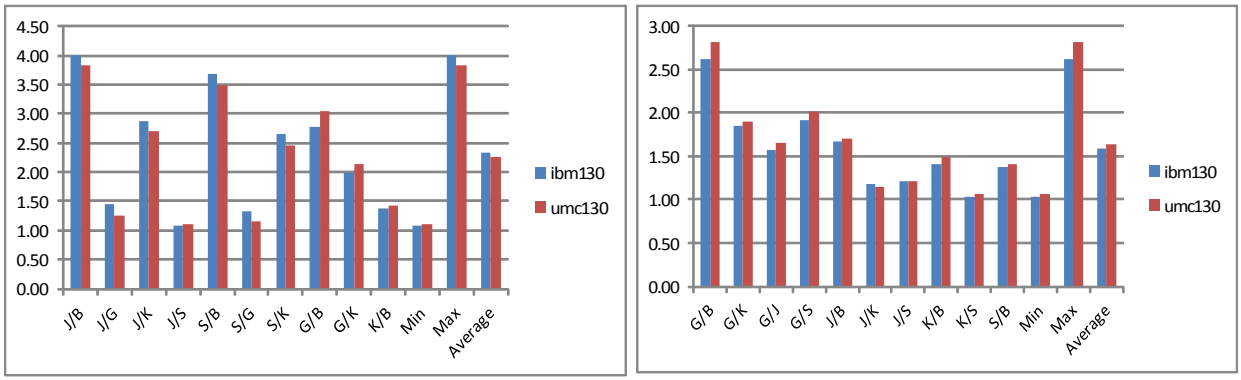


(a) VT (b) GMU

Figure 5.71: Area comparison using different libraries

In the set of candidates of VT, JH has the worst performance for the implementation with both IBM and UMC libraries. In the set candidates of GMU, Groestl has the worst performance also for both IBM and UMC libraries.

In order to seek the quantity relationship between candidates for both libraries, Figure 5.72 is shown below:



(a) VT (b) GMU

Figure 5.72: Relative area variation using different libraries

The maximum variation between candidates of VT is the candidate pair of JH and Blake, which is around 4 times for both IBM and UMC libraries. The maximum variation between candidates of GMU is the candidate pair of Groestl and Blake, which is about 2.5 times variation for both libraries.

- **Performance in Power**

The quantity relationship between candidates is shown in Figure 5.73:

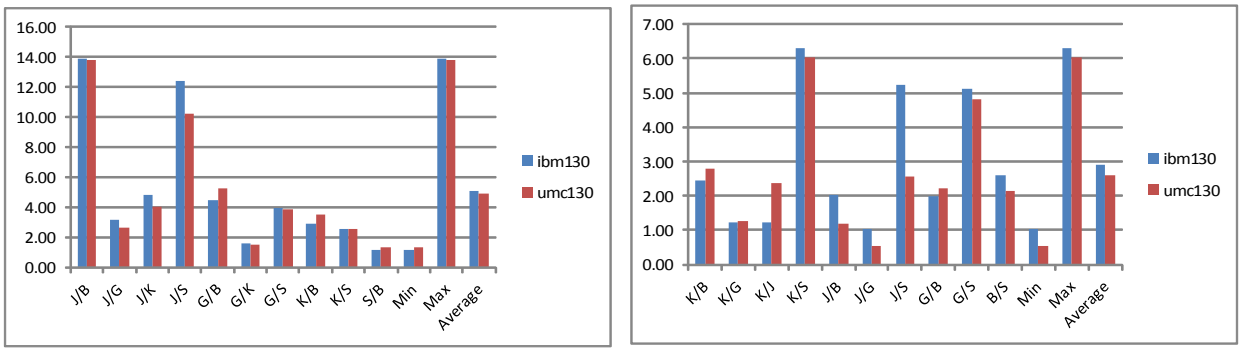
(a) VT (b) GMU

Figure 5.73: Power comparison using different libraries

For VT's candidates, Blake and Skein consume less power compared to that of SHA2, JH has the most power consumption, which is about 6.5 times of SHA2.

For GMU's candidates, only Blake consumes less power than SHA2, among other candidates, Keccak, the most power consumption, takes around 3 times of the power SHA2 consumes.

Figure 5.74 shows the relative variation for each candidate pair:



(a) VT (b) GMU

Figure 5.74: Relative power variation using different libraries

Among the candidate pairs of VT, the variation between JH and Blake is about 14 times, which is the largest. While in the candidate pairs of GMU, the largest variation lie between Keccak and Skein, which is around 6 times.

- **Performance in Throughput**

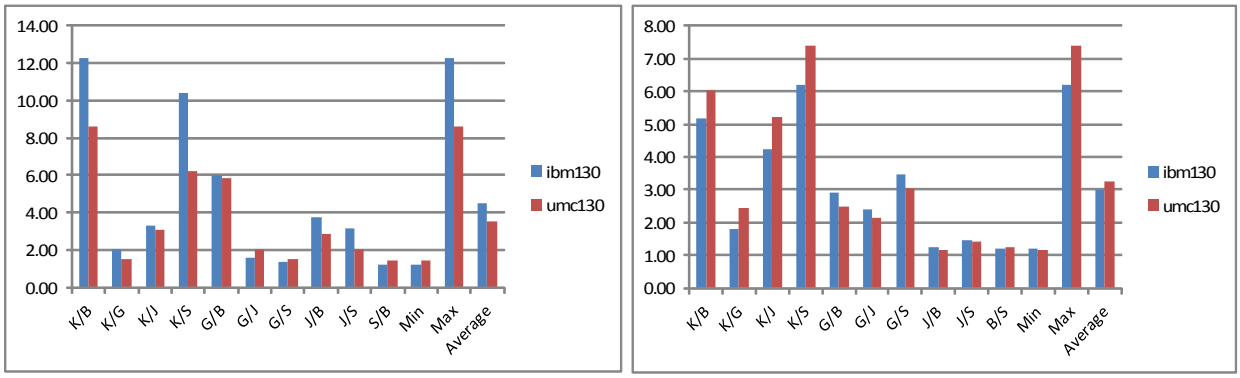
The performance of throughput in quantity relationship is shown in Figure 5.75:

(a) VT (b) GMU

Figure 5.75: Throughput comparison using different libraries

The candidates of VT and GMU have the uniformity that all the candidates have better performance than SHA2 in throughput.

To continue to see the relationship between candidates in a specific library, Figure 5.76 is provided below:

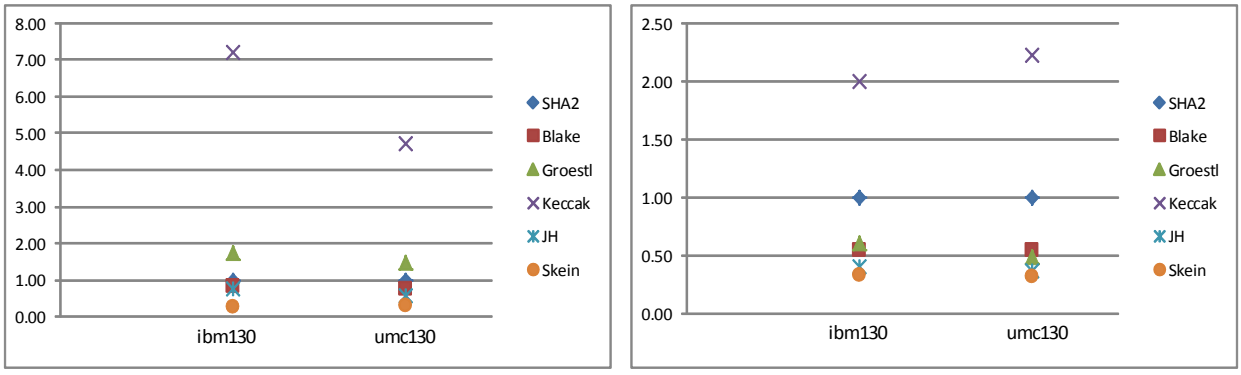


(a) VT (b) GMU

Figure 5.76: Relative throughput variation using different libraries

The maximum variation of VT's candidates lies in the pair of Keccak and Blake, which are 12 times for IBM and 8 times for UMC. The maximum variation of GMU's candidates is between the pair of Keccak and Skein, which are 6 times for IBM and 7.5 tiems for UMC library respectively.

- **Performance in Throughput/Area**

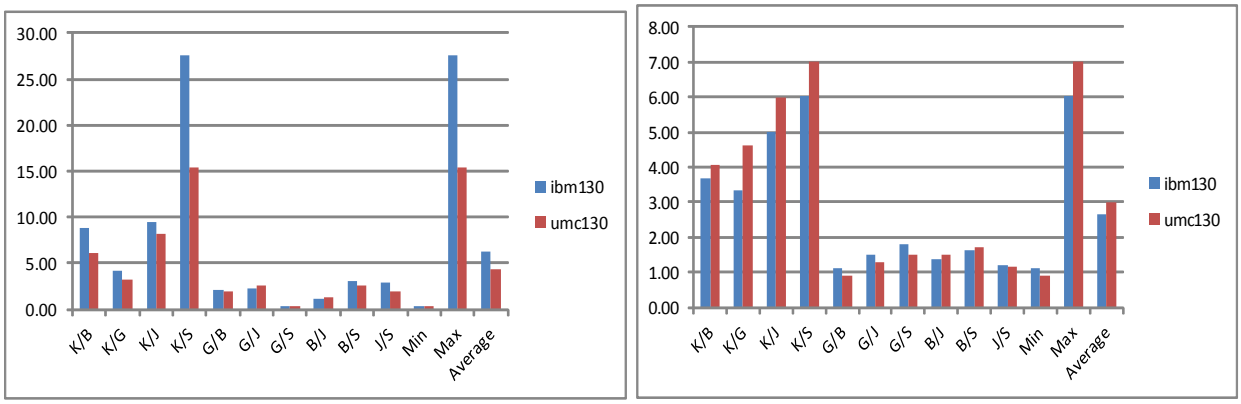


(a) VT (b) GMU

Figure 5.77: Throughput/area comparison using different libraries

The candidates of VT and GMU share two uniformities: first, both Blake and Skein give out less throughput/area than SHA2, second, Keccak from both VT and GMU have the highest throughput/area compared to that of SHA2.

The pair of Keccak and Skein has an extremely large variation value, which is around 30 times for IBM library and 15 times for UMC library.



(a) VT (b) GMU

Figure 5.78: Relative throughput/area variation using different libraries

5.5 Technology Effect Analysis

In this section, the effects from choosing different technology nodes will be discussed. In order to achieve this goal, a series of UMC libraries with different technologies, including UMC 65nm, 90nm, 130nm, and 180nm have been used. In order to do the technology effect analysis, three different implementations, which are implementation at the maximum throughput/area, maximum frequency and minimum frequency will be introduced.

5.5.1 Implementation at Maximum Throughput/Area

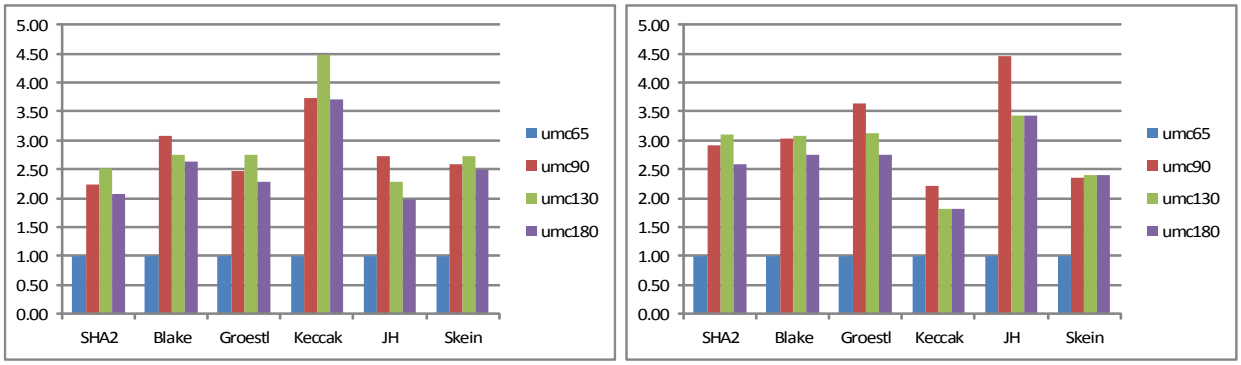
Intra Comparison

The normalization for all different performance has been done to the implementation with UMC 65nm library. A summary of all the relative values are shown in Table C.1.

- **Performance in Area**

The effect on the choice of different technologies of library can be found in Figure 5.79:

All the implementations of the libraries have larger implementation area than that of the comparison base. The change tendency of area does not have a correlation with the the feature size of different technologies.

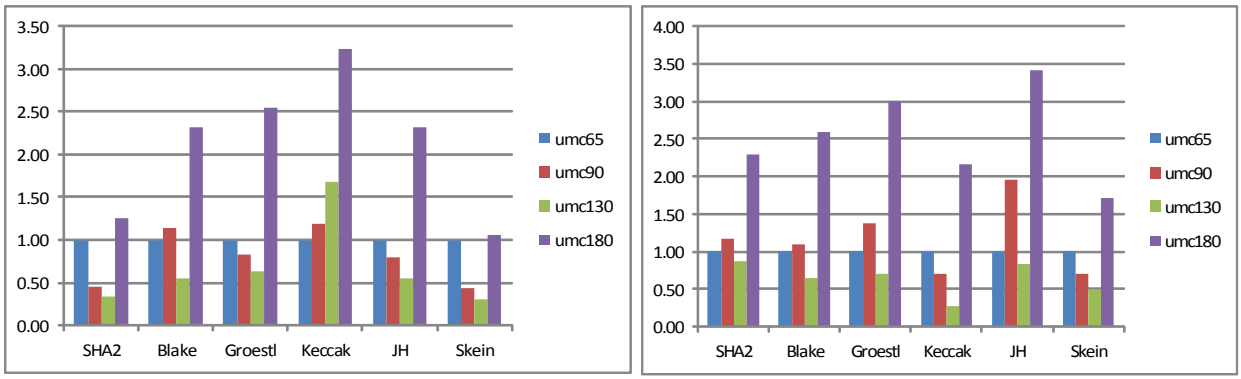


(a) VT (b) GMU

Figure 5.79: Area comparison using different technologies

- **Performance in Power**

Figure 5.80 shows the performance in power of different technology choice:



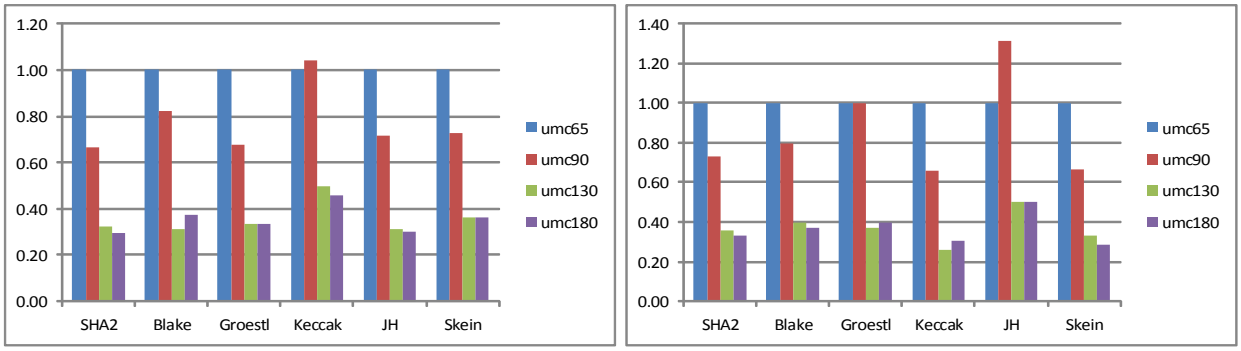
(a) VT (b) GMU

Figure 5.80: Power comparison using different technologies

All the implementations of different candidates with UMC 180nm have a extremely large power consumption. That means the UMC 180nm library is designed not specially aimed at the optimization of power.

- **Performance in Throughput**

The performance of throughput can be found in the Figure 5.81:

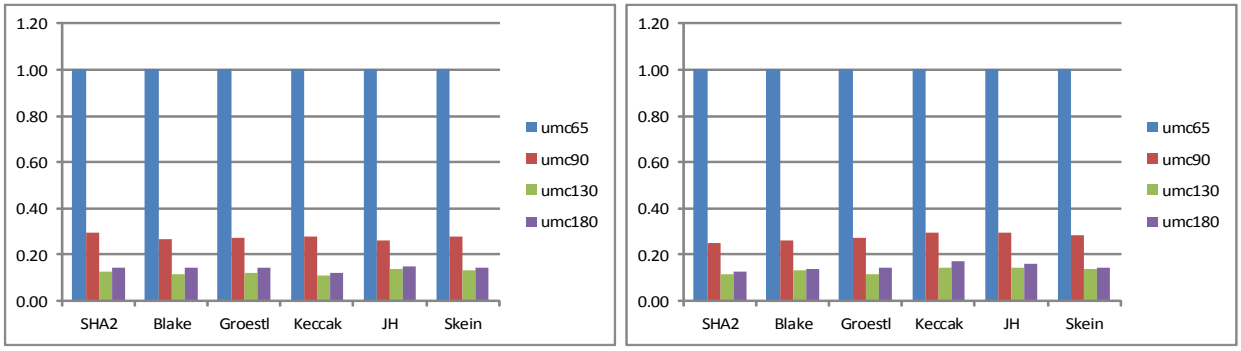


(a) VT (b) GMU

Figure 5.81: Throughput comparison using different technologies

- **Performance in Throughput/Area**

Figure 5.82 shows the performance in throughput/area.



(a) VT (b) GMU

Figure 5.82: Throughput/area comparison using different technologies

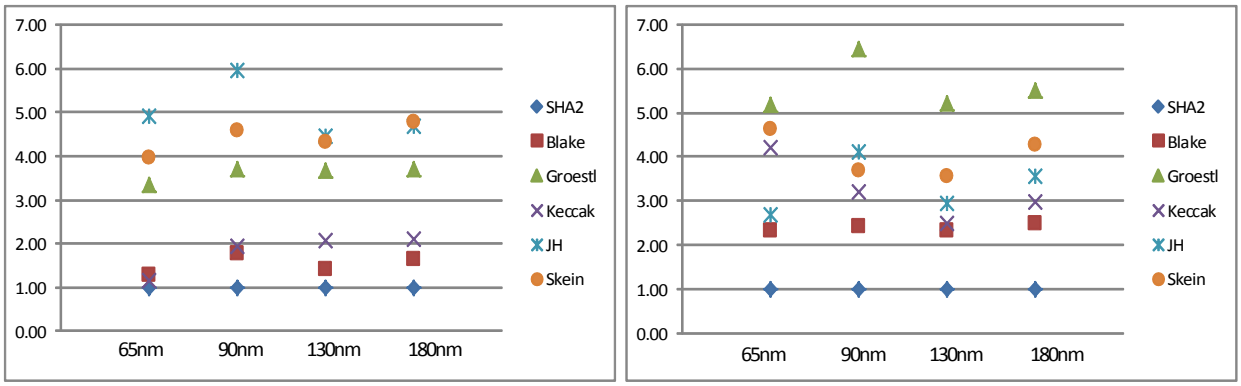
The implementation with UMC 65nm always has the best performance in throughput/area no matter which algorithm is implemented, or what RTL architecture is under taken.

Inter Comparison

Inter comparison compares the performance between candidate under one specific technology. Table C.2 shows all the relative values after being converted to the comparison base.

- **Performance in Area**

The relative relationship in quantity between candidates is shown in Figure 5.83:

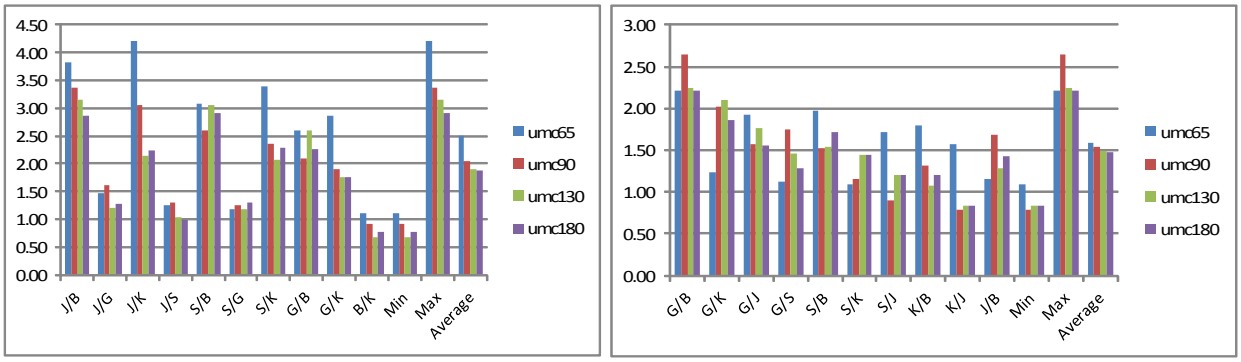


(a) VT (b) GMU

Figure 5.83: Area comparison using different technologies

All the candidates have a larger area than that of SHA2 for both VT and GMU. JH from VT shows the largest area for all candidates with different technology nodes. While for GMU's candidates, Greostl has the largest area with different technology nodes.

The relationship between different candidate pair is shown in Figure 5.84:



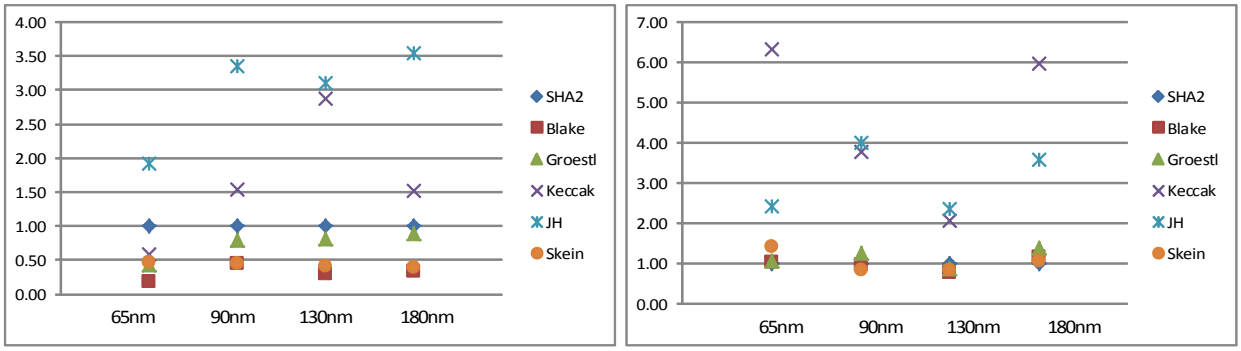
(a) VT (b) GMU

Figure 5.84: Relative area variation using different technologies

For GMU's candidates, Greostl and Blake always have maximum variation for each technology node.

- **Performance in Power**

Figure 5.85 shows the performance in power:

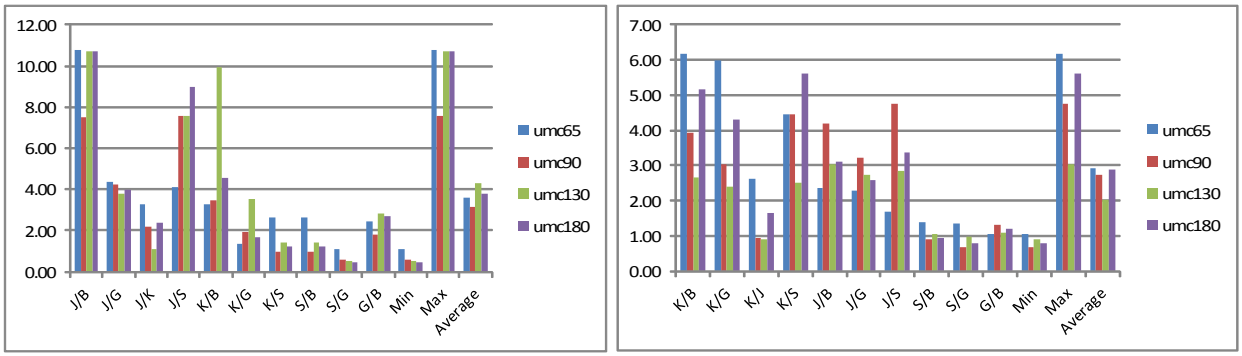


(a) VT (b) GMU

Figure 5.85: Power comparison using different technologies

For the candidates of VT, JH always has the largest power consumption for each technology node. But, the situation becomes more random when it comes to GMU's candidates.

The relative variation between candidate pairs are shown in Figure 5.86:



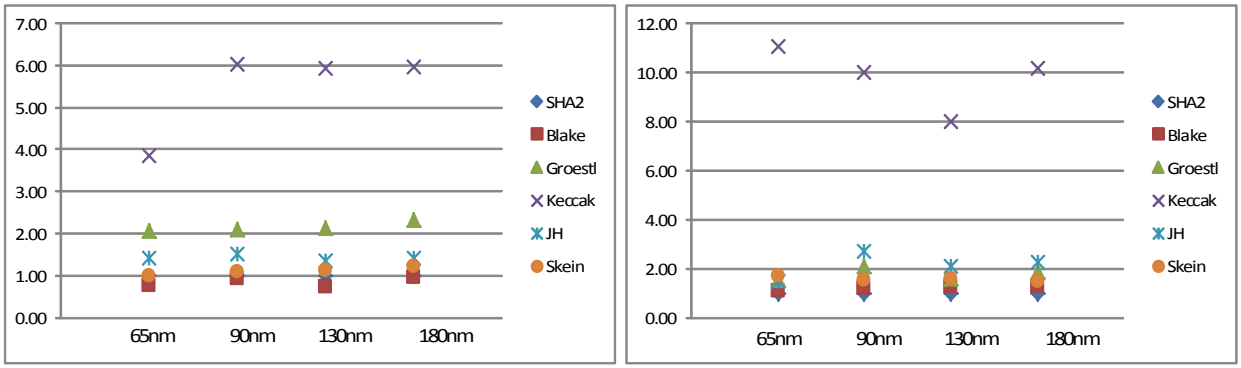
(a) VT (b) GMU

Figure 5.86: Relative power variation using different technologies

For VT's candidates, the maximum variation lies between the candidate pair of JH and Blake for different technology nodes. While for GMU's candidates, the maximum variation is between the candidate pair of Keccak and Blake.

- **Performance in Throughput**

Figure 5.87 shows the performance in throughput:

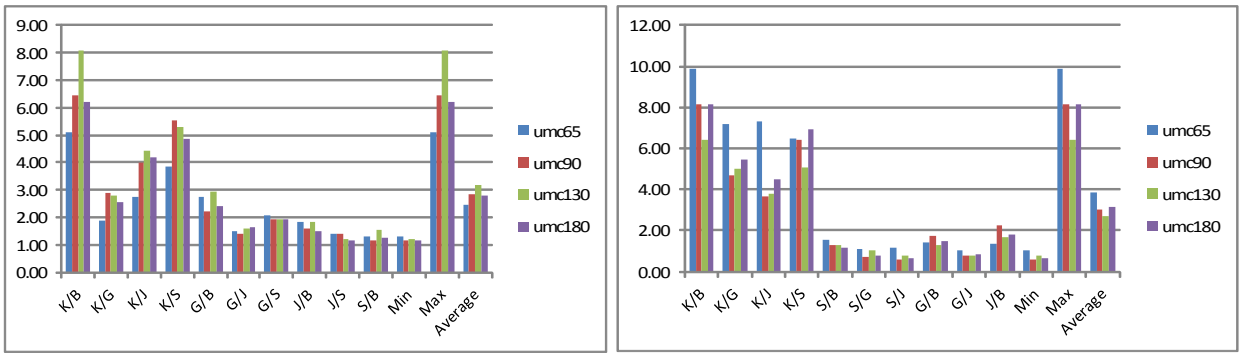


(a) VT (b) GMU

Figure 5.87: Throughput comparison using different technologies

Keccak always has the largest throughput, and except the Blake of VT for all technology nodes, and all the other implementations have a larger throughput than that of the comparison base.

The figure 5.88 is for relative variation:



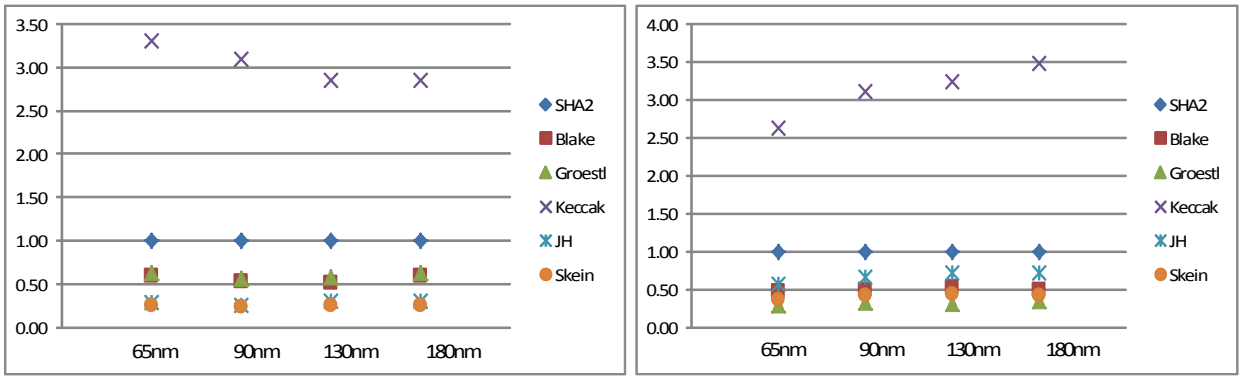
(a) VT (b) GMU

Figure 5.88: Relative throughput variation using different technologies

The candidate pair of Keccak and Blake always has the maximum variation for the implementations on different technology nodes.

- **Performance in Throughput/Area**

The comparison between candidates of throughput/area in quantity is shown in Figure 5.89 below:

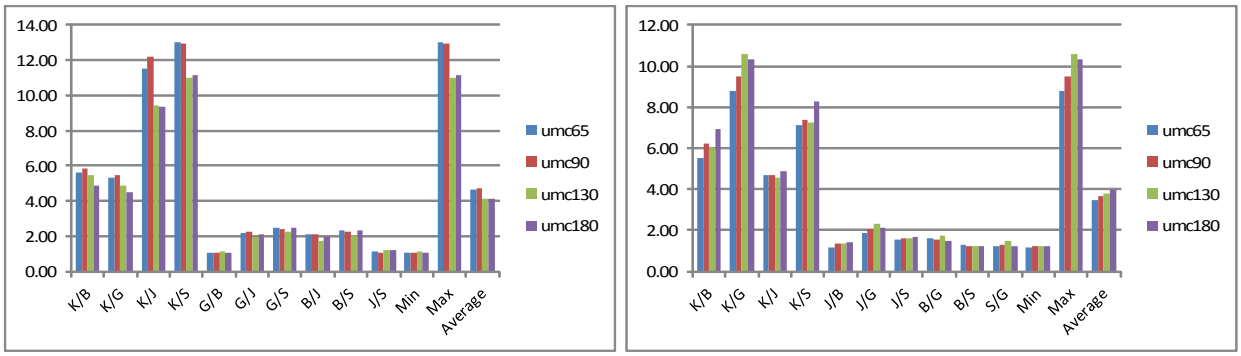


(a) VT (b) GMU

Figure 5.89: Throughput/area comparison using different technologies

Except the candidate of Keccak, all the other candidates from both VT and GMU have worse performance in the aspect of throughput/area for all technology nodes.

For the analysis of the candidate pairs, the relative variation between candidates is shown in the Figure 5.90:



(a) VT (b) GMU

Figure 5.90: Relative throughput/area variation using different technologies

As expected, the maximum variation among candidate pairs lies between Keccak and Skein, which are the largest and the smallest in throughput/area respectively.

5.5.2 Implementation at Maximum Frequency

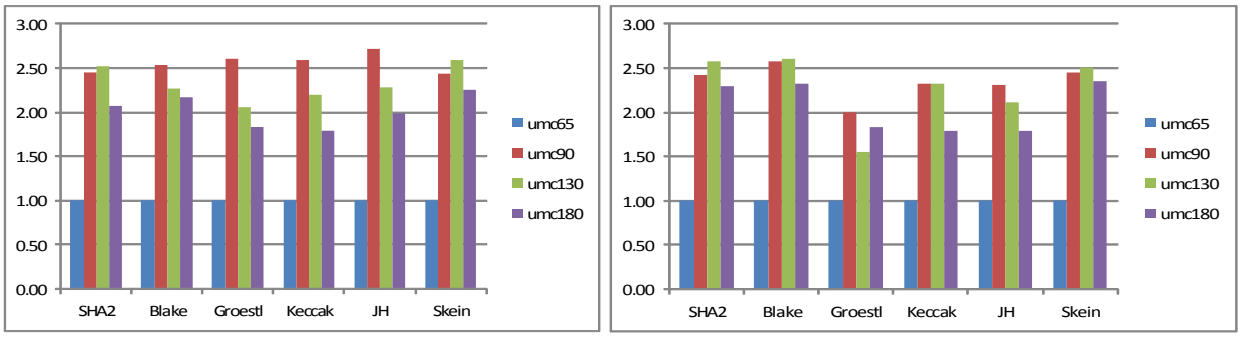
The implementation at the maximum frequency, as what has been done in the previous experiments, is the implementation focused on the optimum timing.

Intra Comparison

In this section, the results for the intra comparison will be shown. The Table has summarized the normalization for the intra comparison, which is the comparison within a specific candidate for different libraries.

- **Performance in area**

In terms of the area, the performance is shown in Figure 5.91:



(a) VT (b) GMU

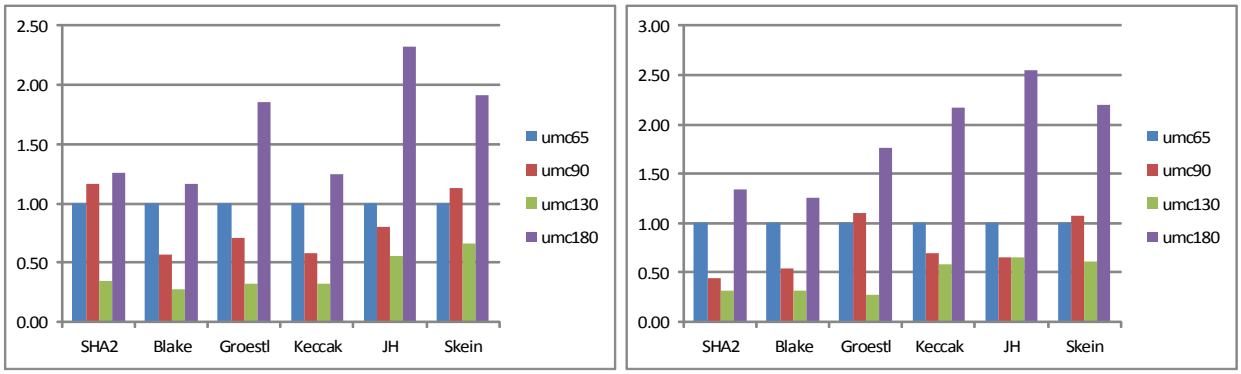
Figure 5.91: Area Comparison using different technologies

Except for UMC 65nm which always has the least area, UMC 180nm has spent more effort on area. Then it comes to the UMC 130nm. After that, we can find UMC 90nm has the worst performance in area, and the largest area is JH with UMC 90nm.

- **Performance in power**

The comparison of power for the intra comparison is shown in Figure 5.92.

The implementation with UMC 180nm has the most power consumption, while UMC 130nm has the least for all candidates. Among VT's candidates, JH has the maximum difference between 180nm and 130nm, which shows 1.3 times than that of 130nm. And for the performance of GMU's candidates, the same conclusion is applied. The implementation with



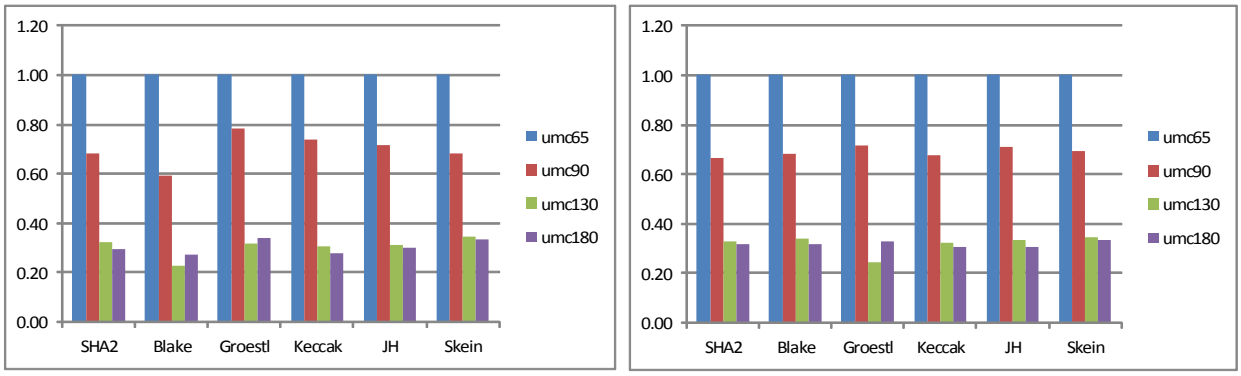
(a) VT (b) GMU

Figure 5.92: Power Comparison using different technologies

UMC 180nm has the most power consumption for all the candidates and UMC 130nm has the least power consumption. The maximum difference among GMU's candidates is JH compared to the implementation using UMC 180nm and UMC 130nm, which has twice the power consumption.

- **Performance in throughput**

The Figure 5.93 shows the performance in throughput for each candidate:



(a) VT (b) GMU

Figure 5.93: Throughput Comparison using different technologies

The implementation with UMC 65nm for all candidates has the maximum operation frequency. After that, it comes to the one with UMC 90nm. The maximum difference for VT's candidates lies between the implementation of UMC 65nm and UMC 130nm, which shows a 79% decrease. The maximum difference between difference is between the implementation of UMC 65nm and UMC 130nm, which also has 79% decrease.

- **Performance in throughput/area**

When it comes to the aspect of throughput/area, we can expect the results by combining the performance in area and throughput, which can be verified in Figure 5.94:

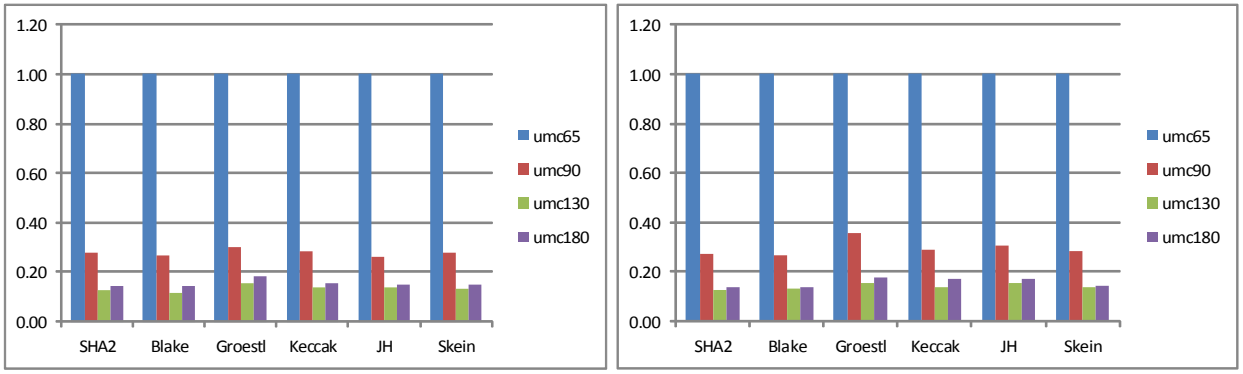


Figure 5.94: Throughput/Area Comparison using different technologies

By observing the figure above, we can summarize that the performance of the implementation with UMC 65nm is far better than all the other candidates. As we can see, the maximum difference between VT's candidates is Blake, which shows 89% decrease between the implementation with UMC 65nm and UMC 130nm. For comparison, the maximum difference between GMU's candidates is also Blake, which shows an 87% decrease.

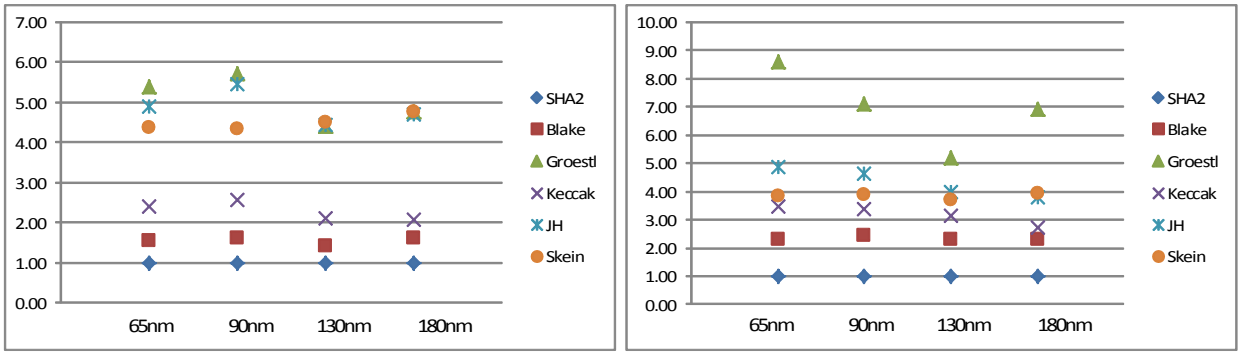
Inter Comparison

The normalization for the absolute values of each algorithm is summarized as in Table C.4.

- **Performance in area**

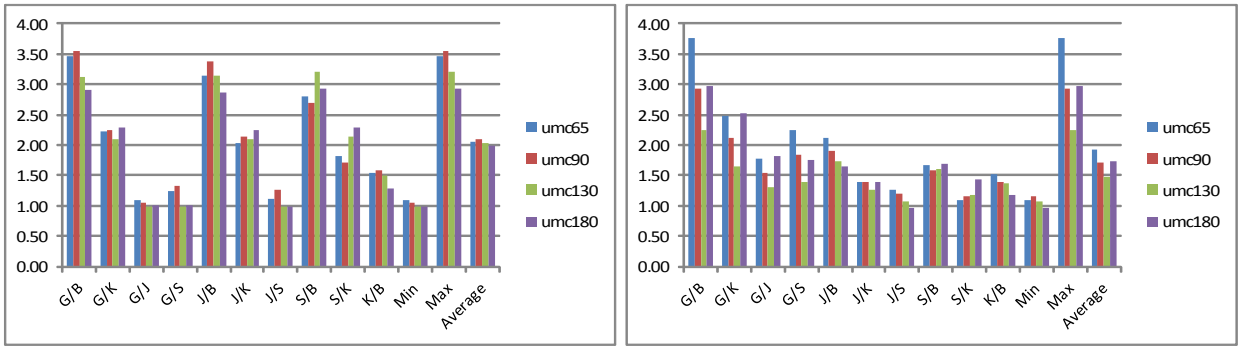
Figure 5.95 shows the performance in area. For VT's candidates, all the performance of area of each candidate is larger than that of SHA2. Using UMC 65nm and UMC 90nm, Groestl has the largest area. For GMU's candidates, we can find that each candidate has a larger area compared to SHA2. Greostl always has the largest area.

The variation between candidates can be found in Figure 5.96. The maximum variation between candidates is between Groestl and Blake for both VT and GMU.



(a) VT (b) GMU

Figure 5.95: Area comparison using different technologies



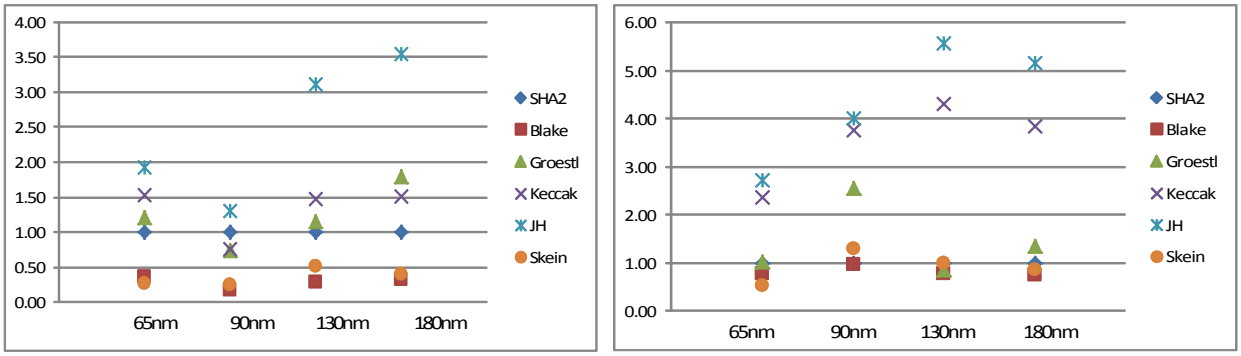
(a) VT (b) GMU

Figure 5.96: Relative area variation using different technologies

- **Performance in power**

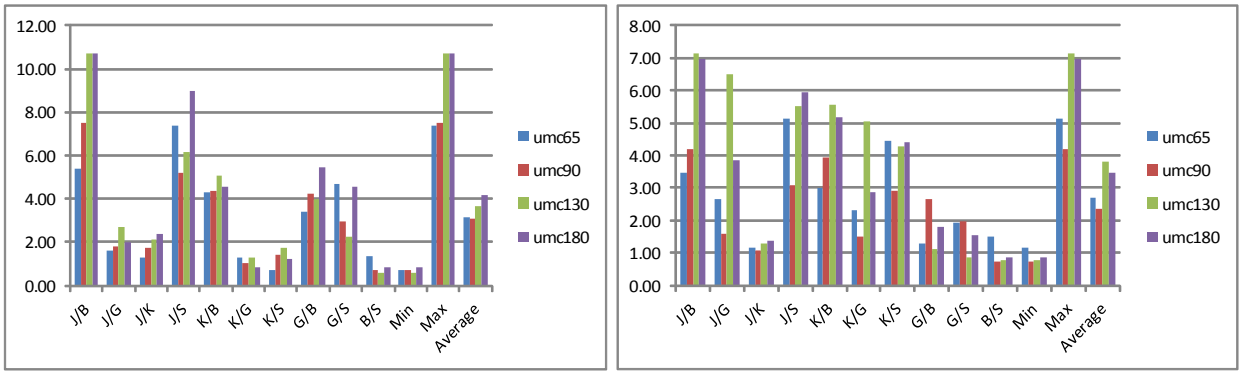
Only JH has more power consumption than that of SHA2 for VT's candidates. When it is referred to the candidates of GMU, JH always has the most power consumption.

Figure 5.98 shows the relative variation between candidates.



(a) VT (b) GMU

Figure 5.97: Power comparison using different technologies



(a) VT (b) GMU

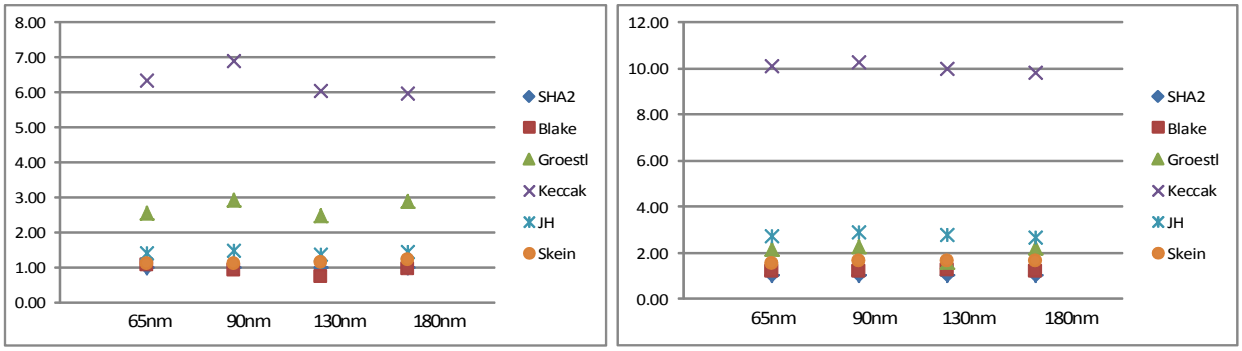
Figure 5.98: Relative power variation using different technologies

- **Performance in throughput**

Figure 5.99 shows the performance in throughput. For candidates from VT and GMU, Keccak always has a faster speed when implemented at the maximum frequency. While, all the other candidates have relatively close performance in throughput.

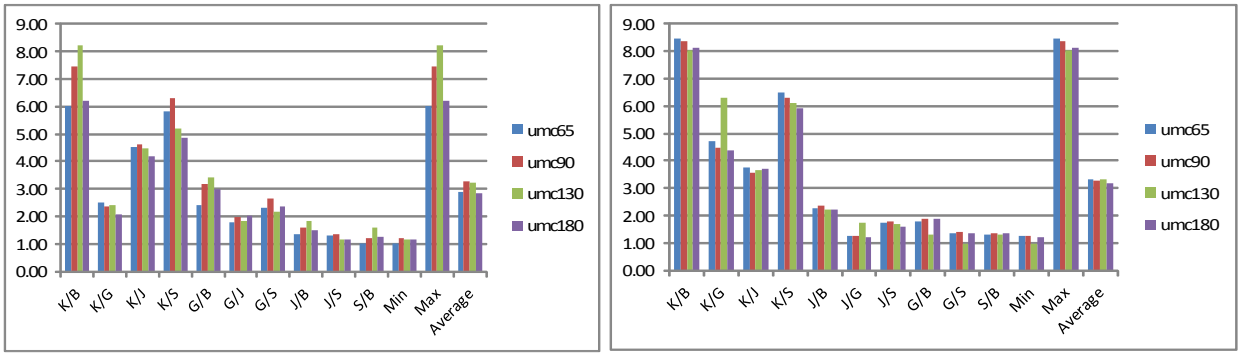
The relative difference can be found in Figure 5.100:

By concluding from the figure, we can get that, for the two sets of candidates, the candidate pair of Keccak and Blake shows the maximum difference when implemented at maximum frequency.



(a) VT (b) GMU

Figure 5.99: Throughput comparison using different technologies



(a) VT (b) GMU

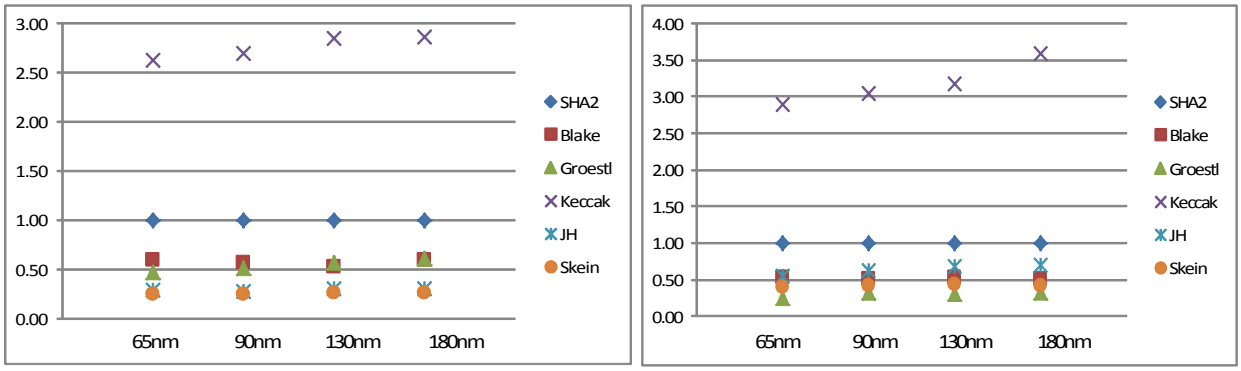
Figure 5.100: Relative throughput variation using different technologies

- **Performance in throughput/area**

Figure 5.101 shows the performance of the throughput/area.

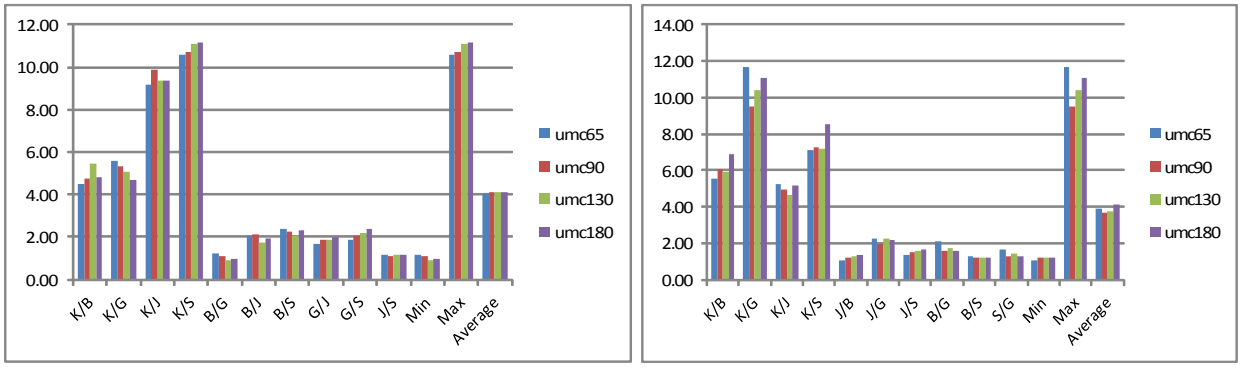
By analyzing Figure 5.101, we can see, all the candidates from VT and GMU have the perfect uniformity. That is the candidate of Keccak has a much larger throughput/area than all the other candidates, and all the candidates, except Keccak, have a relatively smaller throughput/area for different technologies.

The relative difference between candidates is shown in Figure 5.102. In VT's candidates, the maximum difference lies in the candidate pair of Keccak and Skein, while the maximum difference for GMU's candidates is between the candidate pair of Keccak and Greostl.



(a) VT (b) GMU

Figure 5.101: Throughput/Area comparison using different technologies



(a) VT (b) GMU

Figure 5.102: Relative throughput/Area variation using different technologies

5.5.3 Implementation at Minimum Frequency

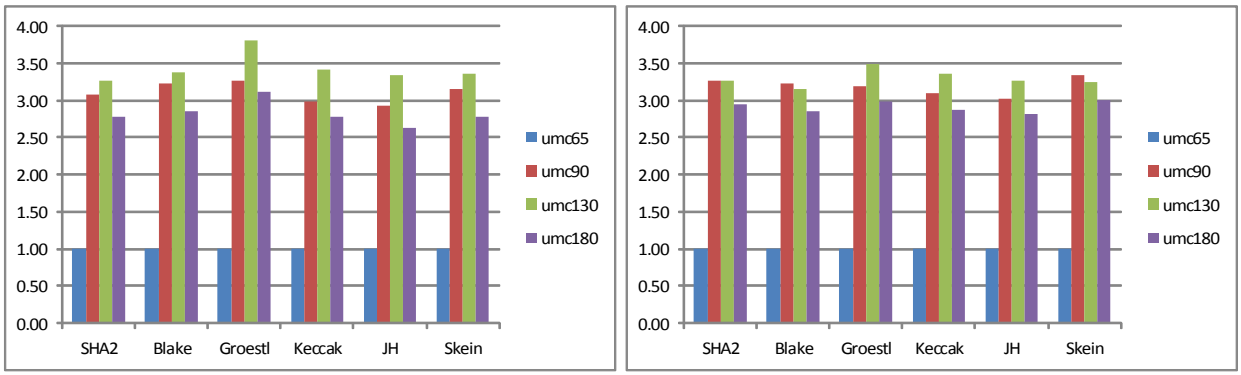
After the exploration and analysis for the implementation at the maximum throughput/area and maximum frequency, this section has provided a platform for the discussion of the performance for the implementation at minimum frequency for different technology nodes. The implementation at minimum frequency is the implementation with the worst cells, with respect to delay, available in the standard cell libraries. Two points of comparison will be presented in following discussion: intra comparison and inter comparison.

Intra Comparison

Table C.5 is the summary for the normalization which can function as the database for the comparison.

- **Performance in area**

The performance of area for each algorithm is shown in Figure 5.103:



(a) VT (b) GMU

Figure 5.103: Area comparison using different technologies

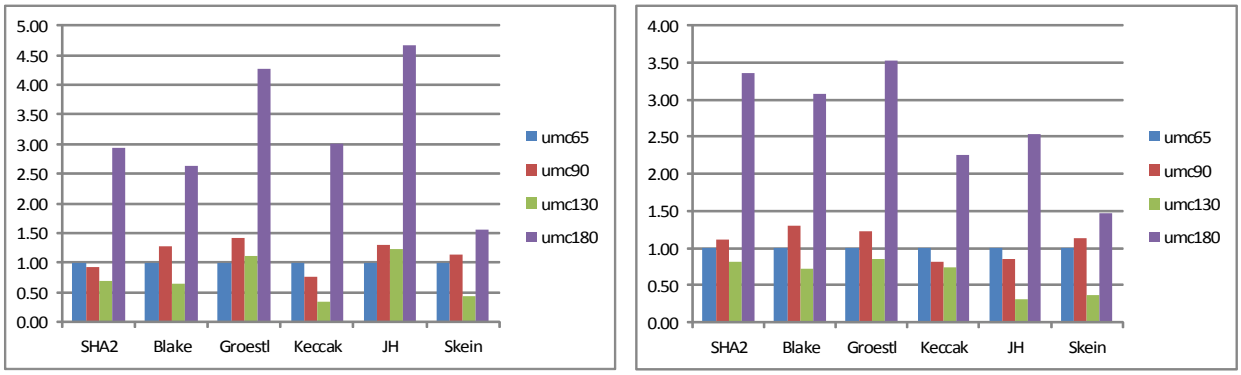
UMC 65nm always shows the smallest area. The maximum change due to the technology change is Groestl, which is 3.8 times and 3.5 times difference between UMC 65nm and 130nm for VT and GMU's implementation, respectively.

- **Performance in power**

The performance of power is shown in Figure 5.104. It is obvious to see that implementations of all algorithms using UMC 180nm is much larger than all the others. That means the performance of power with the library of UMC 180nm is not ideal.

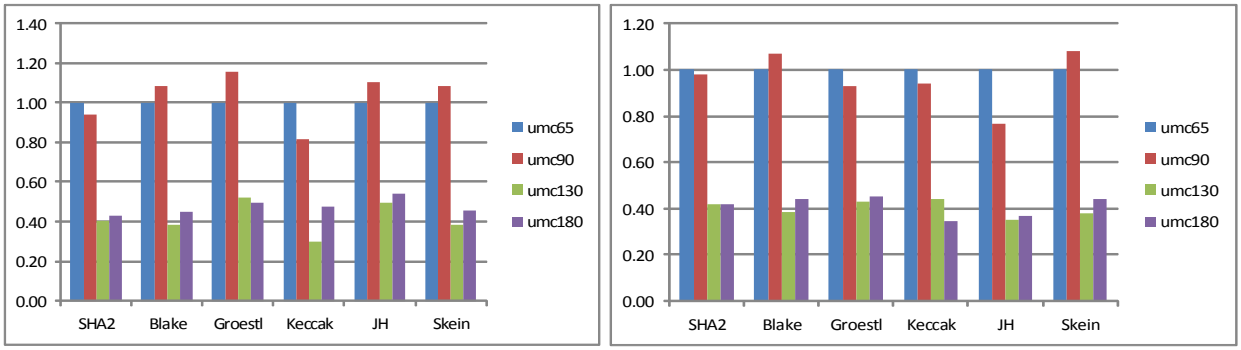
- **Performance in throughput**

Figure 5.105 shows the performance in throughput. For implementations at the minimum frequency, the technology node of 65nm has almost the same performance with 130nm, which are much better than others.



(a) VT (b) GMU

Figure 5.104: Power comparison using different technologies

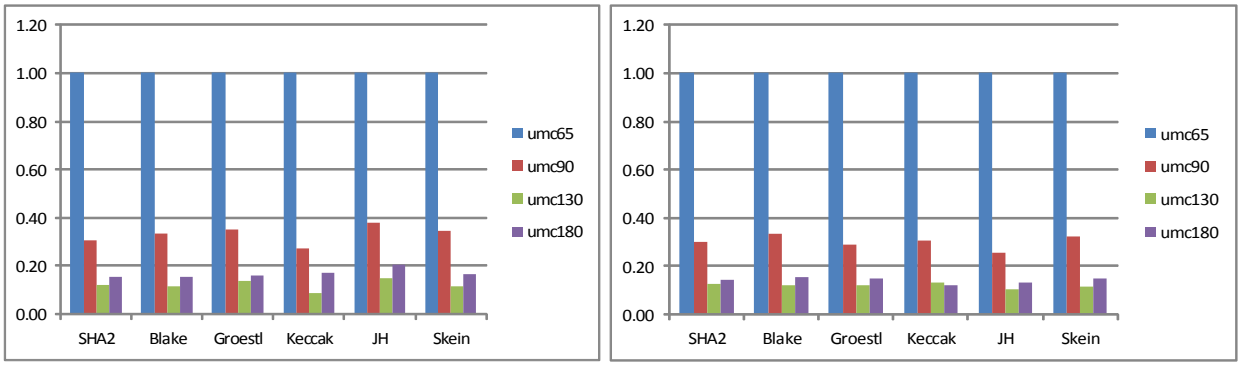


(a) VT (b) GMU

Figure 5.105: Throughput comparison using different technologies

- **Performance in throughput/area**

By combining the performance of area and throughput, we can see in Figure 5.106, the implementation with 65nm is much better in the aspect of throughput/area.



(a) VT (b) GMU

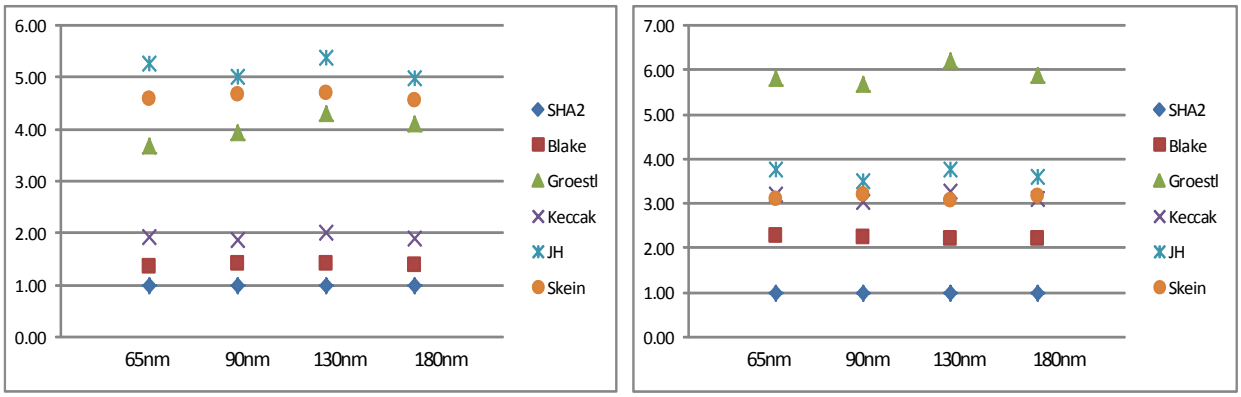
Figure 5.106: Throughput/Area comparison using different technologies

Inter Comparison

The normalization for the inter comparison is included in Table C.6.

- **Performance in area**

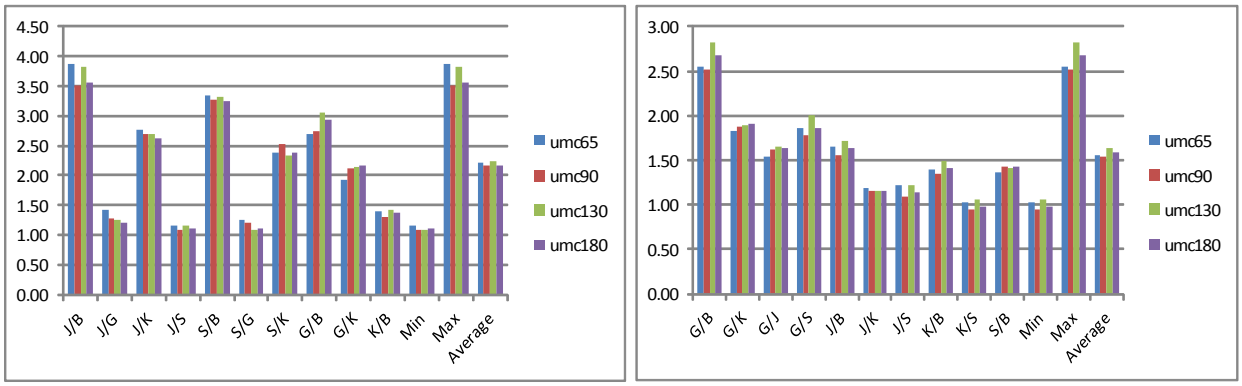
The histogram for area is shown in Figure 5.107.



(a) VT (b) GMU

Figure 5.107: Area comparison using different technologies

From Figure 5.107, it is seen that all the candidates from both VT and GMU have a larger area than that of SHA2. The maximum area is found in JH and Greostl for VT and GMU, respectively, which shows the effect on the implementation from different technology nodes.



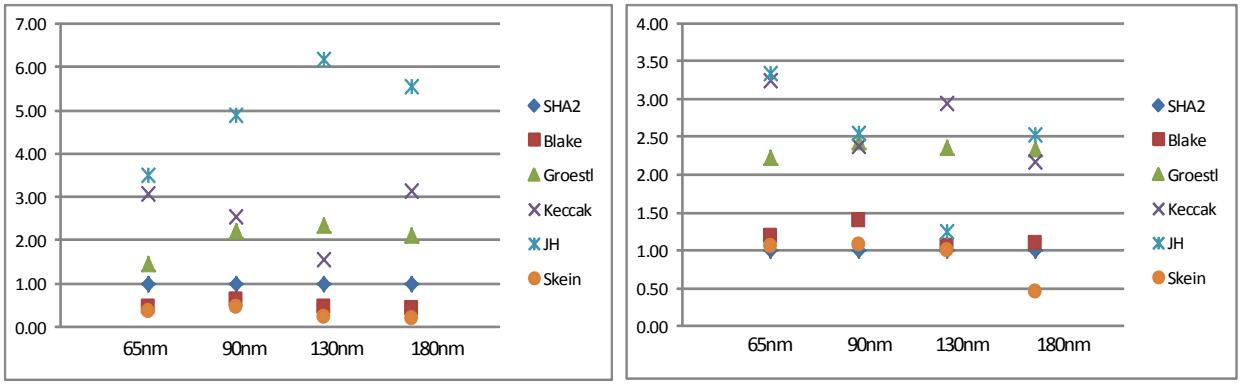
(a) VT (b) GMU

Figure 5.108: Relative area variation using different technologies

The maximum difference amongst candidates are between JH and Blake for VT, and Groestl and Blake for GMU. Even for different technology nodes, the conclusion remains the same for the two architectures from VT and GMU.

- **Performance in power**

When it comes to the power performance, Figure 5.109 is provided:

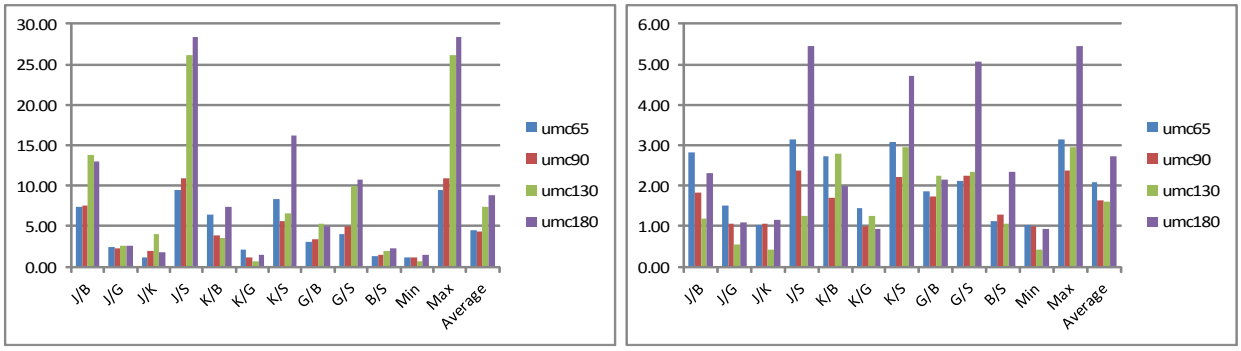


(a) VT (b) GMU

Figure 5.109: Power comparison using different technologies

For VT's architecture, the implementation with 65nm show relative small variation between candidates, and JH always has the most power consumption. For GMU's architecture, except the implementation with 130nm, JH has the most power consumption. All the implementations on different technology nodes shows a relative small variation between candidates.

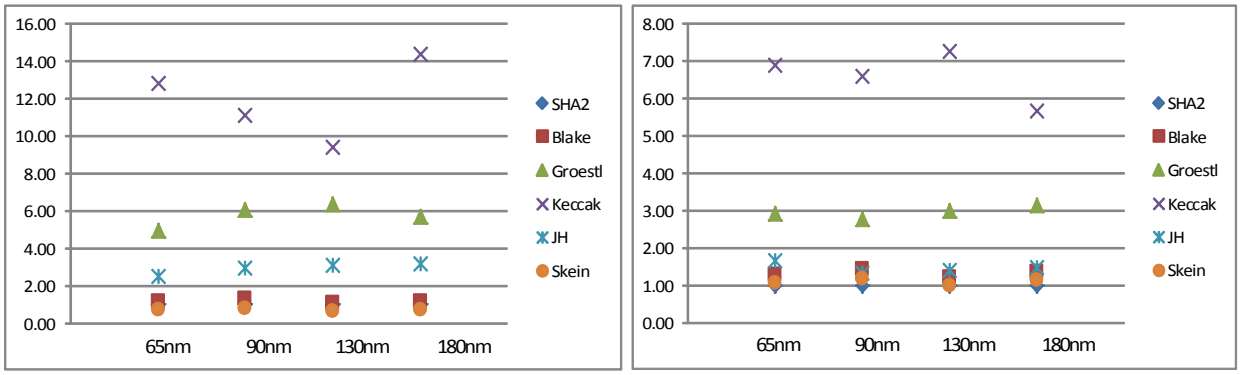
As we can see, the relative difference between candidates is shown in Figure 5.110:



(a) VT (b) GMU

Figure 5.110: Relative power variation using different technologies

- **Performance in throughput**



(a) VT (b) GMU

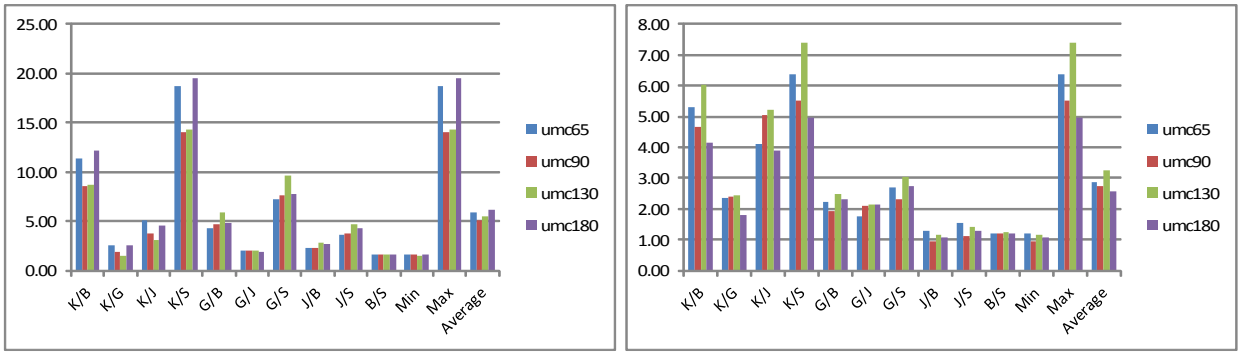
Figure 5.111: Throughput comparison using different technologies

When comparing the performance in throughput, we can find, the candidates from VT and GMU have a perfect uniformity for the maximum throughput.

In order to seek the relative difference between candidates, Figure 5.112 is provided. As we can expect, the maximum difference between candidates is between Keccak and Skein, for both VT and GMU.

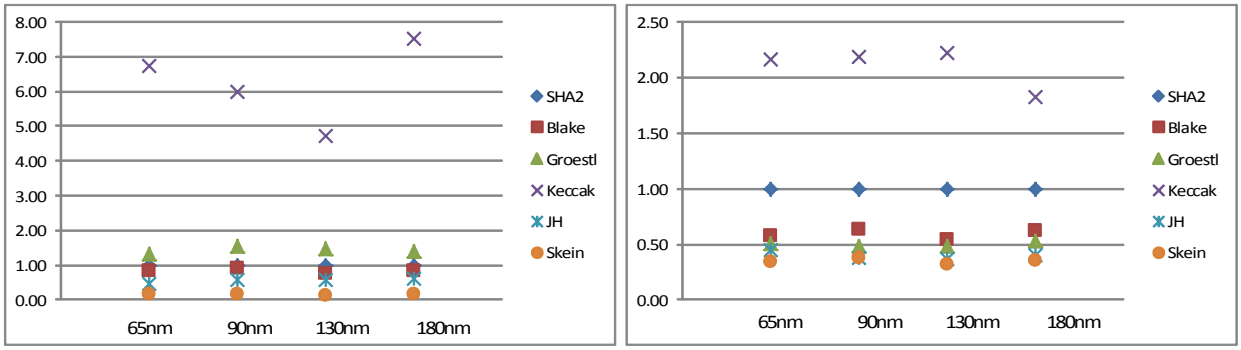
- **Performance in throughput/area**

Figure 5.113 shows the performance of throughput/area for each algorithm.



(a) VT (b) GMU

Figure 5.112: Relative throughput variation using different technologies

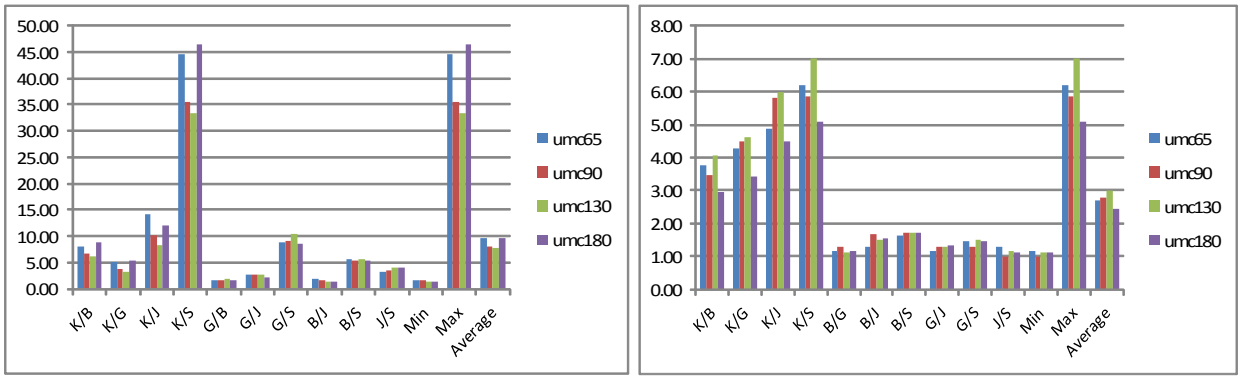


(a) VT (b) GMU

Figure 5.113: Throughput/area comparison using different technologies

At this time, Keccak from both VT and GMU has the largest throughput/area as shown in the figure above. While the performance of other candidates in GMU shows a relative even worse performance than the comparison base, SHA2.

The relative difference is shown in Figure 5.114. The candidate pairs from VT have a relatively large variation. The maximum difference is shown between Keccak and Skein for both VT and GMU's candidates.



(a) VT (b) GMU

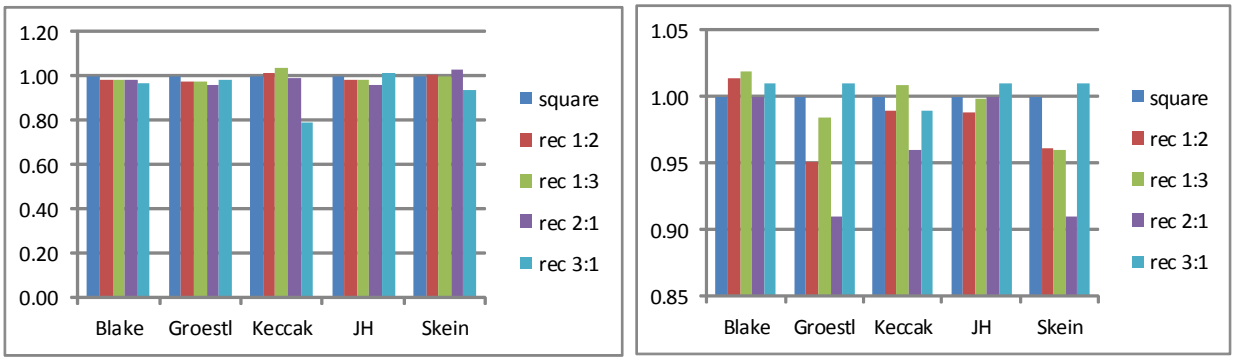
Figure 5.114: Relative throughput/area variation using different technologies

5.6 Floorplan Shape Effect

5.6.1 Intra Comparison

The summary for the normalization is shown in Table D.1.

- **Performance in area**

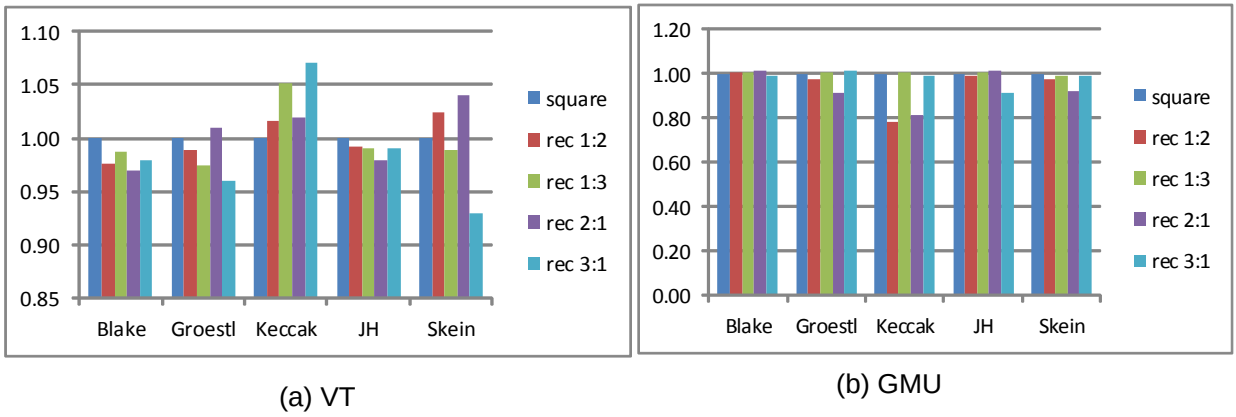


(a) VT (b) GMU

Figure 5.115: Area comparison for different floor plan shapes

From Figure 5.115, we can have a concept of how the different floor plan shape influence on the performance of area for each algorithm. For VT's candidate, Keccak has the most sensitivity for the floor plan shape, which result in a 3% increase in area for "rec 1:3". For GMU's candidates, Groestl has the most sensitivity for the floor plan shape, which has a 500% decrease for "rec 1:2".

- **Performance in power**



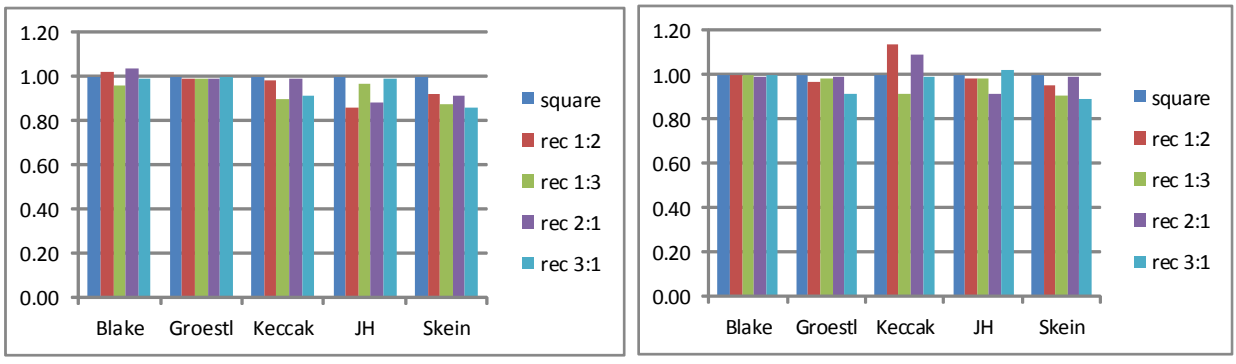
(a) VT (b) GMU

Figure 5.116: Power comparison for different floor plan shapes

Although VT's candidates have their preference for the specific shapes, while the difference is trivial. The maximum difference is 5% for Keccak. When observing the performance of Keccak of GMU, we can see, the difference is much larger, which shows 24% between "rec 1:2" with others.

- **Performance in throughput**

The performance of throughput is shown in Figure 5.117:



(a) VT (b) GMU

Figure 5.117: Throughput comparison for different floor plan shapes

For VT's candidates, both JH and Skein have a large difference when choosing different floor plan shapes. While for GMU's candidates, the most obvious change lies in KeccaK, which shows 23% increase when choosing "rec 1:2".

- **Performance in throughput/area**

The performance for throughput/area is shown in Figure 5.118.

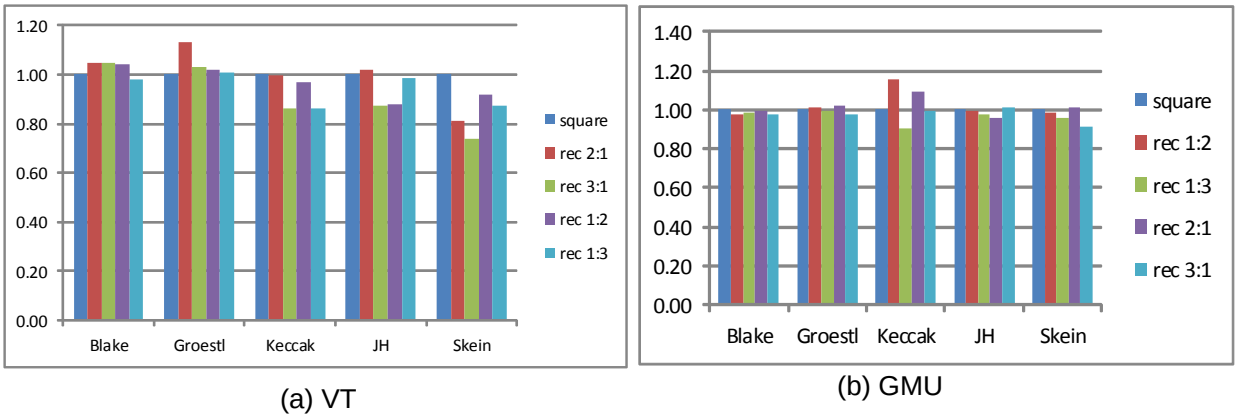


Figure 5.118: Throughput/area comparison for different floor plan shapes

By observing, all the candidates have their own preference for a specific shape, while the difference is relatively small in terms of throughput/area.

5.6.2 Inter Comparison

Table D.2 is a summary for the normalization to the performance of Blake.

- **Performance in area**

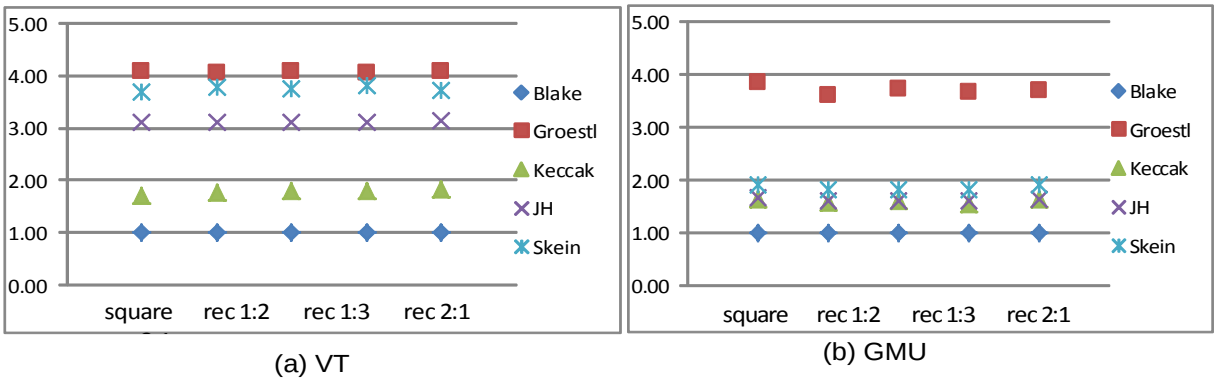


Figure 5.119: Area comparison for different floor plan shapes

The area comparison between candidates on a specific floor plan shape is shown in Figure 5.119. From Figure 5.119, we can see that for both VT and GMU's candidates, Blake always has the least implementation area and Groestl always has the most implementation area.

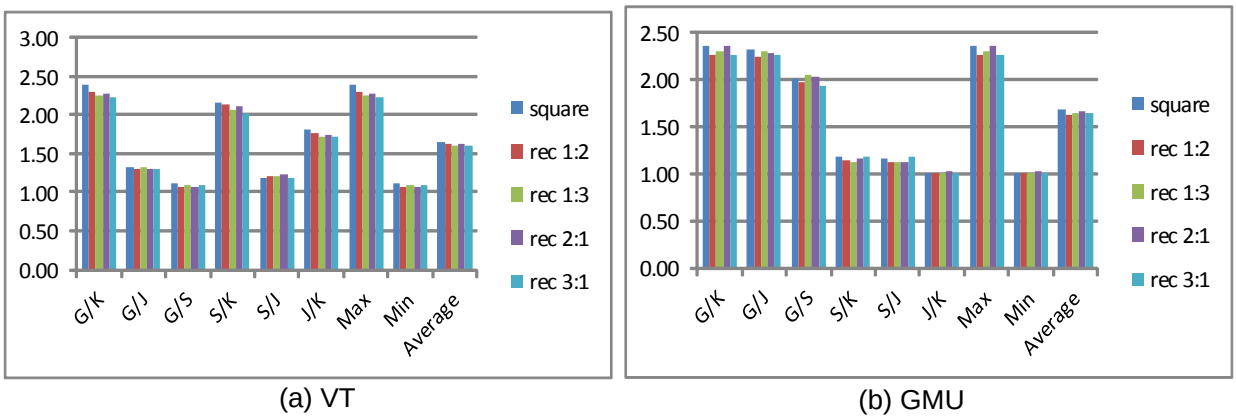


Figure 5.120: Relative area variation for different floor plan shapes

As we expect, because the maximum and minimum area lies in Groestl and Blake respectively, so the maximum difference between candidates is between the candidate pair of Groestl and Blake, for both VT and GMU's algorithms.

- **Performance in power**

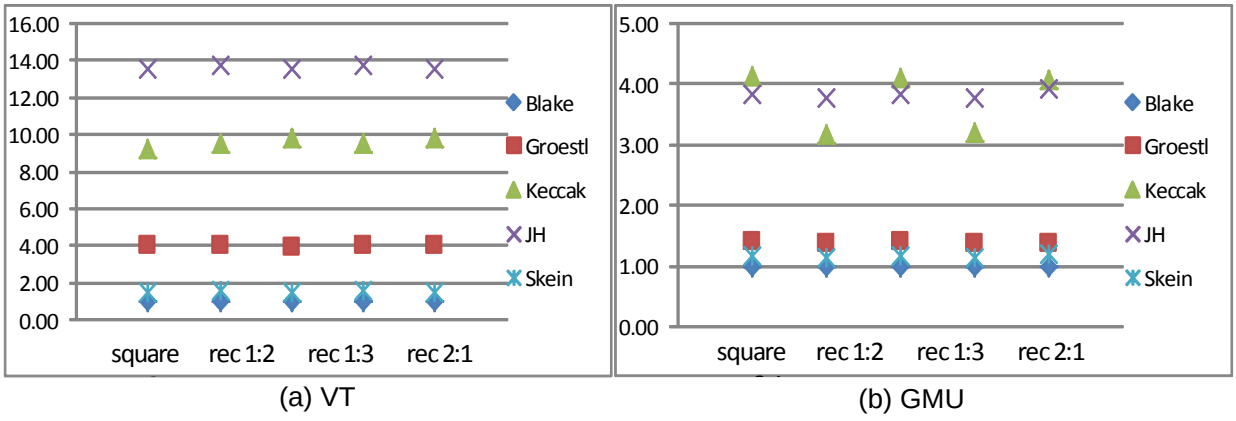
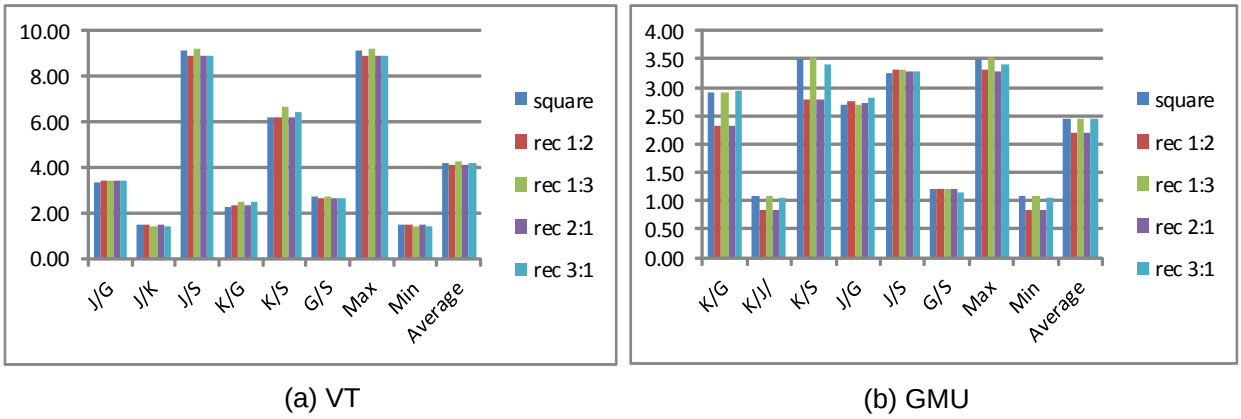


Figure 5.121: Power comparison for different floor plan shapes

The performance of power for each candidate can found in Figure 5.121. We can conclude that Blake again always consumes least power compared with other candidates of VT and GMU. In VT's candidates, JH has the most power consumption. In GMU's candidates,

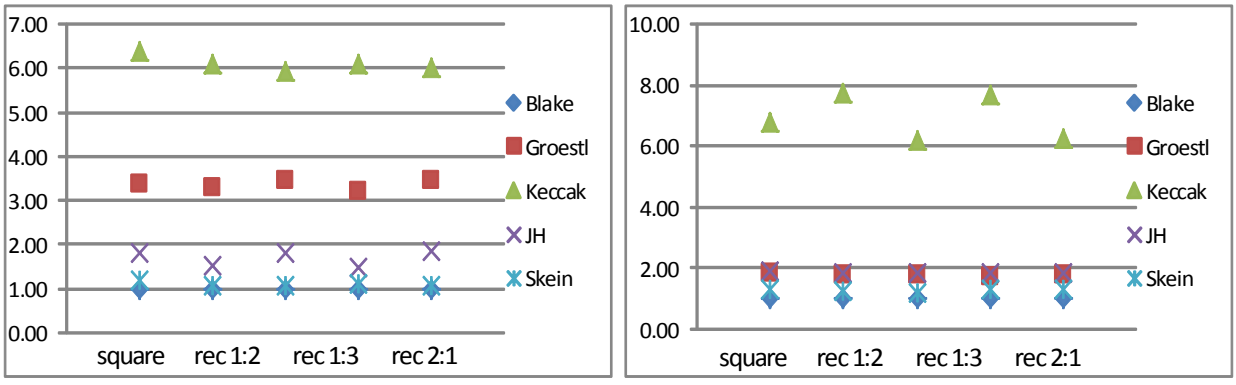
Keccak has the most power consumption for square and "rec 1:3", and JH has little more power consumption for "rec 1:2". The relative difference for power between candidates is shown in Figure 5.122.



(a) VT (b) GMU

Figure 5.122: Relative power variation for different floor plan shapes

- **Performance in throughput**



(a) VT (b) GMU

Figure 5.123: Throughput comparison for different floor plan shapes

By analyzing Figure 5.123, Keccak from both VT and GMU performs much better than other candidates. Although candidate Blake has least area, it also sacrifices the performance in producing the digests.

For all VT and GMU's candidates, Keccak and Skein always have the maximum difference when comparing the performance between candidates.

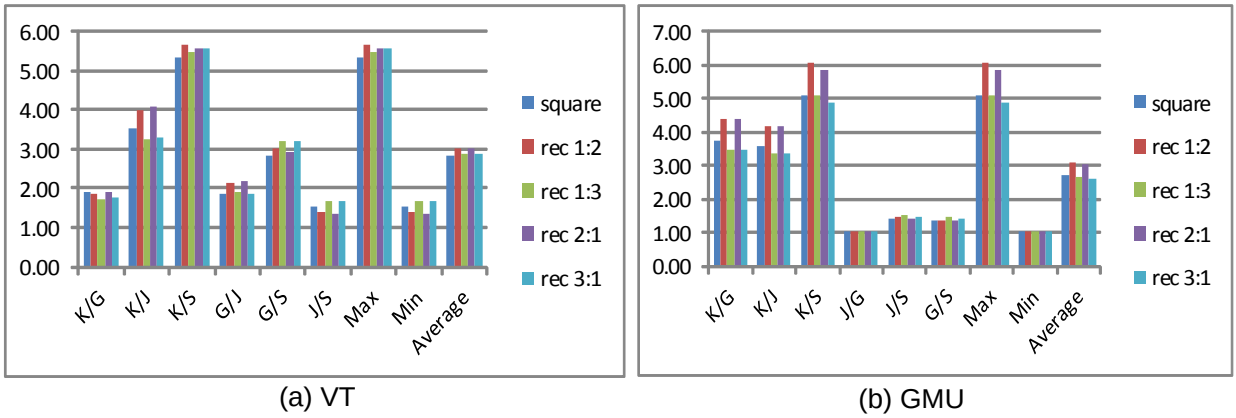


Figure 5.124: Relative throughput variation for different floor plan shapes

- **Performance in throughput/area**

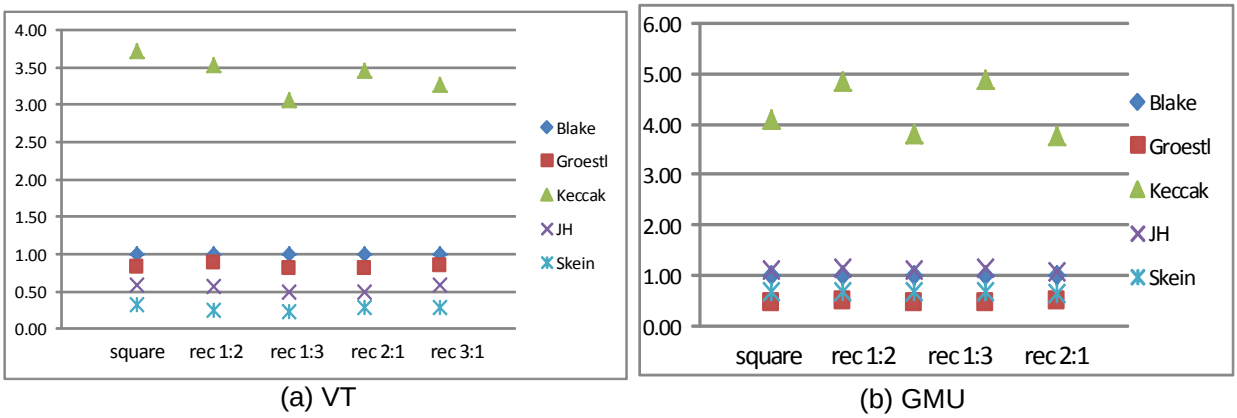


Figure 5.125: Throughput/area comparison for different floor plan shapes

The performance of throughput/area is shown in Figure 5.125. When comparing the candidates in terms of throughput/area, the conclusion that Keccak always has the best performance can be easily drawn. All other candidates are much competitive to each other with regards to throughput/area.

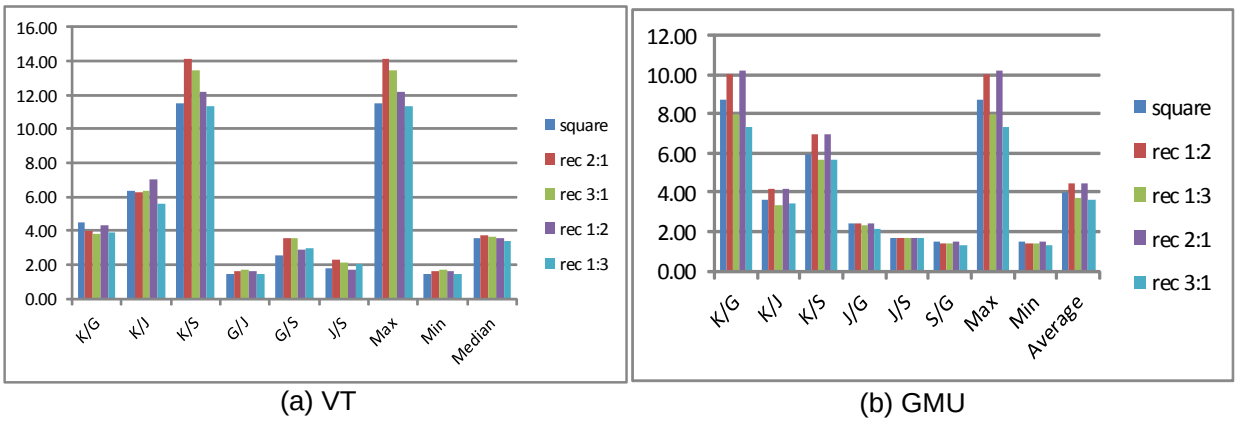


Figure 5.126: Relative throughput/area variation for different floor plan shapes

The minimum throughput/area for VT's candidates is Skein, and Greostl for GMU's. So, the maximum difference for VT's candidates is between Keccak and Skein, while it's between Keccak and Greostl for GMU.

Chapter 6

Conclusion

In this thesis, through the performance analysis for SHA3 finalists with an ASIC implementation, the effects of the performance on ASIC implementation from different constraint factors have been extensively explored. Basically, the approach has been realized through the following steps:

At the beginning of the thesis, the motivation behind this thesis research has been discussed, as well as the introduction for the secure algorithms, the definition of hash function, and the general hardware components for the implementation of a secure algorithm. After that, the history for the SHA3 competition was introduced in order to lay a clear foundation for the future discussion in this thesis.

Later, to introduce the focus of this thesis to an ASIC implementation, the ASIC implementation flow has been introduced. Post-synthesis and post-layout analysis have been the focus through out the thesis. Several CAD tools and several important implementation factors have been discussed. After that, the library characteristics as well as the implementation characteristics of the standard cells has been explored. This is to make a better understanding of the effects on the performance from different constraints.

Based on the discussion for the basic knowledge, chapter four begins with the introduction for the comparison methods, which includes intra comparison (horizontal comparison) and inter comparison (vertical comparison). Continuously, the organization of the experiments, different constraints, and different environment setup have been explained.

Finally, in chapter five, fair and comprehensive comparisons have been conducted. Based on the methodology introduced in chapter four, the comparison has been done through four aspects: synthesis constraints effects, libraries effects, technology node effects, and floor plan shape effects.

Overall, through the discussion of this thesis, we can come to a clear picture of how different implementation factors influence the ASIC performance of secure algorithms.

6.1 Future Work

Because of the limitation for the source of standard cell libraries, this research is only based on a limited number of libraries. In order to seek for a comprehensive comparison, implementations with more source of libraries can be done in the future.

After the analysis for the ASIC performance in the post-synthesis and post-layout stage, the influence from different implementation factors has been understood. Because all of these analysis are built from the simulation and tool analysis, in order to explore the performance of each algorithm on the real devices, it would be more valuable to compare the performance of each algorithm on the real chip.

Appendix A

Normalization for Constraint Effect Analysis

Table A.1: Intra comparison at maximum throughput/area for constraint effect analysis

Note: 1. Freq is frequency. 2. FA is frequency and area. 3. FP is frequency and power. 3. FAP is frequency, area and power

Arch	Cand Name	Area (Normalized to Freq)				Power (Normalized to Freq)				Throughput (Normalized to Freq)				Throughput/Area (Normalized to Freq)			
		Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP
VT	SHA2	1	0.97	1.01	0.99	1	0.96	0.93	0.03	1	1.00	1.04	1.06	1	1.03	1.03	1.07
	Blake	1	1.12	1.16	1.12	1	1.18	1.11	1.18	1	1.25	1.24	1.25	1	1.11	1.07	1.11
	Groestl	1	0.98	1.00	0.99	1	0.96	0.96	0.95	1	1.00	1.00	1.00	1	1.02	1.00	1.01
	Keccak	1	0.97	1.00	1.01	1	0.94	0.96	0.96	1	1.02	1.02	1.02	1	1.04	1.02	1.01
	JH	1	1.00	1.03	1.01	1	0.95	1.00	1.00	1	1.04	1.07	1.04	1	1.04	1.04	1.02
	Skein	1	0.93	1.01	0.96	1	0.94	0.88	0.88	1	1.00	1.00	1.00	1	1.08	0.99	1.04
GMU	SHA2	1	0.98	1.00	0.93	1	1.01	0.64	0.63	1	1.02	0.99	0.93	1	1.04	0.99	1.00
	Blake	1	0.98	1.00	0.99	1	0.99	0.99	0.98	1	1.00	0.99	1.00	1	1.02	0.99	1.01
	Groestl	1	0.75	0.77	0.77	1	0.71	0.71	0.70	1	0.80	0.80	0.80	1	1.06	1.04	1.04
	Keccak	1	0.98	1.26	0.99	1	0.99	0.99	0.99	1	1.00	1.00	1.27	1	1.02	1.01	1.01
	JH	1	0.99	1.35	1.01	1	1.00	0.99	0.99	1	1.00	1.00	1.33	1	1.01	0.99	0.99
	Skein	1	0.94	0.99	0.98	1	0.98	1.02	1.01	1	1.00	1.00	1.00	1	1.07	1.01	1.02

Table A.2: Inter comparison at maximum throughput/area for constraint effect analysis

Note: 1. Freq is frequency. 2. FA is frequency and area. 3. FP is frequency and power. 3. FAP is frequency, area and power

Arch	Cand Name	Area (Normalized to SHA2)				Power (Normalized to SHA2)				Throughput (Normalized to SHA2)				Throughput/Area (Normalized to SHA2)			
		Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP
VT	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	1.41	1.63	1.62	1.59	0.29	0.35	0.34	0.37	0.73	0.91	0.87	0.86	0.52	0.56	0.54	0.54
	Groestl	3.67	3.71	3.64	3.67	0.82	0.82	0.84	0.84	2.14	2.14	2.05	2.01	0.58	0.58	0.56	0.55
	Keccak	2.08	2.09	2.06	2.11	2.88	2.81	2.96	2.96	5.93	6.02	5.79	5.67	2.85	2.88	2.81	2.69
	JH	4.44	4.58	4.54	4.53	3.11	3.08	3.33	3.33	1.35	1.40	1.39	1.32	0.30	0.31	0.30	0.29
	Skein	4.31	4.14	4.31	4.17	0.41	0.40	0.39	0.39	1.12	1.12	1.08	1.06	0.26	0.27	0.25	0.25
GMU	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	0.57	0.63	0.55	0.62	0.78	0.76	1.20	1.19	1.24	1.22	1.25	1.34	0.54	0.53	0.53	0.54
	Groestl	0.50	0.49	0.48	0.53	1.18	0.83	1.31	1.30	2.15	1.68	1.74	1.86	0.32	0.33	0.34	0.34
	Keccak	2.16	2.18	2.22	1.83	2.06	2.03	3.21	3.22	8.00	7.83	8.09	11.00	3.23	3.17	3.27	3.26
	JH	0.44	0.37	0.37	0.41	2.34	2.32	3.64	3.66	2.10	2.06	2.12	3.02	0.71	0.69	0.71	0.70
	Skein	0.35	0.37	0.32	0.36	0.82	0.79	1.31	1.31	1.59	1.55	1.60	1.71	0.45	0.45	0.45	0.46

Table A.3: Intra comparison at maximum frequency for constraint effect analysis

Note: 1. Freq is frequency. 2. FA is frequency and area. 3. FP is frequency and power. 3. FAP is frequency, area and power

Arch	Cand Name	Area (Normalized to Freq)				Power (Normalized to Freq)				Throughput (Normalized to Freq)				Throughput/Area (Normalized to Freq)			
		Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP
VT	SHA2	1	0.97	1.01	0.99	1	0.96	0.93	0.93	1	1.00	1.04	1.06	1	1.03	1.03	1.07
	Blake	1	1.12	1.16	1.12	1	1.18	1.11	1.18	1	1.25	1.24	1.25	1	1.11	1.07	1.11
	Groestl	1	0.99	0.99	0.99	1	0.97	0.98	0.97	1	1.01	1.00	1.01	1	1.02	0.97	0.98
	Keccak	1	0.99	1.03	1.03	1	0.96	0.98	0.95	1	1.01	1.01	1.01	1	1.02	0.97	0.98
	JH	1	1.00	1.03	1.01	1	0.95	1.00	1.00	1	1.04	1.07	1.04	1	1.04	1.04	1.02
	Skein	1	0.89	1.05	1.00	1	0.77	0.91	0.91	1	0.97	0.99	0.97	1	1.10	0.94	0.97
GMU	SHA2	1	0.98	1.00	0.99	1	1.01	1.00	0.98	1	1.02	0.99	0.96	1	1.04	0.99	0.97
	Blake	1	0.98	1.02	0.99	1	0.99	0.99	0.99	1	1.00	1.00	1.00	1	1.02	0.98	1.01
	Groestl	1	1.04	1.06	1.04	1	1.00	1.03	1.01	1	1.00	1.01	1.02	1	0.96	0.95	0.97
	Keccak	1	0.99	0.99	0.99	1	0.98	0.99	0.98	1	1.02	1.01	1.02	1	1.03	1.02	1.03
	JH	1	0.98	1.00	1.00	1	0.99	0.99	1.02	1	1.01	1.00	1.02	1	1.03	1.00	1.02
	Skein	1	1.00	1.02	1.02	1	1.00	1.04	1.04	1	1.03	1.02	1.02	1	1.03	0.99	1.00

Table A.4: Inter comparison at maximum frequency for constraint effect analysis

Note: 1. Freq is frequency. 2. FA is frequency and area. 3. FP is frequency and power. 3. FAP is frequency, area and power

Arch	Cand Name	Area (Normalized to SHA2)				Power (Normalized to SHA2)				Throughput (Normalized to SHA2)				Throughput/Area (Normalized to SHA2)			
		Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP
VT	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	1.43	1.63	1.62	1.59	0.29	0.35	0.34	0.37	0.73	0.91	0.87	0.86	0.52	0.56	0.54	0.54
	Groestl	4.41	4.51	4.35	4.39	1.15	1.16	1.21	1.19	2.49	2.52	2.40	2.34	0.56	0.56	0.55	0.53
	Keccak	2.12	2.16	2.17	2.19	1.46	1.47	1.54	1.50	6.02	6.08	5.82	5.73	2.85	2.82	2.68	2.62
	JH	4.44	4.58	4.54	4.53	3.11	3.08	3.33	3.33	1.35	1.40	1.39	1.32	0.30	0.31	0.30	0.29
	Skein	4.51	4.14	4.71	4.54	0.51	0.40	0.49	0.49	1.15	1.13	1.10	1.06	0.26	0.27	0.23	0.23
GMU	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	2.32	2.31	2.37	2.33	0.97	0.95	0.96	0.99	1.24	1.22	1.25	1.30	0.54	0.53	0.53	0.56
	Groestl	6.62	7.04	7.07	6.99	1.58	1.57	1.63	1.64	2.15	2.11	2.20	2.27	0.32	0.30	0.31	0.32
	Keccak	3.16	3.19	3.14	3.16	4.31	4.20	4.27	4.33	10.00	9.97	10.24	10.59	3.17	3.12	3.26	3.36
	JH	4.00	3.98	4.00	4.02	5.55	5.46	5.51	5.79	2.75	2.71	2.78	2.91	0.69	0.68	0.69	0.72
	Skein	3.72	3.78	3.82	3.86	1.01	0.99	1.05	1.07	1.64	1.65	1.69	1.75	0.44	0.44	0.44	0.45

Table A.5: Intra comparison at minimum frequency for constraint effect analysis

Note: 1. Freq is frequency. 2. FA is frequency and area. 3. FP is frequency and power. 3. FAP is frequency, area and power

Arch	Cand Name	Area (Normalized to Freq)				Power (Normalized to Freq)				Throughput (Normalized to Freq)				Throughput/Area (Normalized to Freq)			
		Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP
VT	SHA2	1	0.97	1.00	1.00	1	0.96	0.91	0.91	1	0.98	0.95	0.95	1	1.01	0.94	0.95
	Blake	1	0.98	1.03	1.03	1	0.95	0.88	0.92	1	1.00	0.97	1.02	1	1.01	0.95	1.00
	Groestl	1	0.99	1.00	1.00	1	0.97	0.94	0.94	1	0.94	0.92	0.92	1	0.95	0.91	0.91
	Keccak	1	0.98	1.00	1.00	1	0.72	0.79	0.68	1	0.72	0.82	0.83	1	0.74	0.81	0.83
	JH	1	0.97	0.99	0.99	1	0.79	0.46	0.46	1	0.88	0.58	0.64	1	0.91	0.59	0.65
	Skein	1	0.96	0.99	0.97	1	0.94	0.88	0.88	1	1.00	1.00	1.00	1	1.04	1.01	1.03
GMU	SHA2	1	0.98	1.00	1.00	1	1.00	0.88	0.88	1	0.99	0.88	0.88	1	1.01	0.87	0.88
	Blake	1	1.00	0.99	0.99	1	1.01	0.93	0.93	1	1.00	0.98	0.98	1	1.01	0.99	0.99
	Groestl	1	0.98	1.01	1.01	1	0.98	0.97	0.97	1	1.00	1.00	1.00	1	1.02	0.99	0.99
	Keccak	1	0.97	0.98	0.98	1	0.98	0.79	0.79	1	1.00	0.92	0.92	1	1.04	0.95	0.95
	JH	1	0.99	1.03	1.03	1	1.00	1.02	1.02	1	1.00	1.01	1.01	1	1.01	0.97	0.97
	Skein	1	0.98	1.00	1.00	1	0.99	0.99	0.99	1	1.00	1.00	1.00	1	1.02	1.00	1.00

Table A.6: Inter comparison at minimum frequency for constraint effect analysis

Note: 1. Freq is frequency. 2. FA is frequency and area. 3. FP is frequency and power. 3. FAP is frequency, area and power

Arch	Cand Name	Area (Normalized to SHA2)				Power (Normalized to SHA2)				Throughput (Normalized to SHA2)				Throughput/Area (Normalized to SHA2)			
		Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP	Freq	FA	FP	FAP
VT	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	1.41	1.43	1.44	1.44	0.45	0.44	0.43	0.45	1.09	1.11	1.12	1.17	0.77	0.78	0.78	0.81
	Groestl	4.29	4.39	4.29	4.30	2.34	2.37	2.42	2.41	6.34	6.12	6.13	6.13	1.48	1.39	1.43	1.43
	Keccak	2.00	2.02	2.00	2.00	2.58	1.95	2.25	1.94	9.41	6.79	8.09	8.21	4.70	3.45	4.04	4.10
	JH	5.39	5.40	5.30	5.31	6.17	5.08	3.12	3.12	3.10	2.80	1.90	2.10	0.58	0.52	0.36	0.40
	Skein	4.92	4.89	4.85	4.79	0.60	0.59	0.58	0.58	1.51	1.55	1.59	1.59	0.31	0.32	0.33	0.33
GMU	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	2.20	2.25	2.18	2.18	2.20	2.25	2.18	2.18	1.20	1.22	1.35	1.35	0.55	0.54	0.62	0.62
	Groestl	6.20	6.22	6.23	6.22	6.20	6.22	6.23	6.22	2.99	3.04	3.40	3.40	0.48	0.49	0.55	0.55
	Keccak	3.27	3.23	3.17	3.17	3.27	3.23	3.17	3.17	7.25	7.35	7.61	7.62	2.22	2.28	2.40	2.40
	JH	3.75	3.80	3.87	3.87	3.75	3.80	3.87	3.87	1.47	1.48	1.68	1.68	0.39	0.39	0.43	0.43
	Skein	3.71	3.70	3.69	3.70	3.71	3.70	3.69	3.70	1.65	1.67	1.88	1.88	0.45	0.45	0.51	0.51

Appendix B

Normalization for Library Effect Analysis

Table B.1: Intra comparison at maximum throughput/area for library effect analysis

Arch	Cand Name	Area (Normalized to IBM130)		Power (Normalized to IBM130)		Throughput (Normalized to IBM130)		Throughput/Area (Normalized to IBM130)	
		IBM130	UMC130	IBM130	UMC130	IBM130	UMC130	IBM130	UMC130
VT	SHA2	1	0.79	1	0.85	1	0.90	1	1.14
	Blake	1	0.70	1	0.54	1	0.70	1	1.01
	Groestl	1	0.92	1	1.17	1	0.98	1	1.07
	Keccak	1	0.75	1	0.79	1	0.74	1	0.99
	JH	1	0.73	1	0.97	1	0.73	1	1.00–
	Skein	1	0.78	1	0.65	1	0.83	1	1.07
GMU	SHA2	1	0.75	1	0.71	1	0.91	1	1.20
	Blake	1	0.82	1	0.77	1	1.00–	1	1.21
	Groestl	1	0.75	1	0.75	1	0.74	1	0.98
	Keccak	1	0.80	1	0.74	1	1.00–	1	1.25
	JH	1	0.64	1	0.26	1	0.77	1	1.21
	Skein	1	0.82	1	0.80	1	1.00–	1	1.22

Table B.2: Inter comparison at maximum throughput/area for library effect analysis

Arch	Cand Name	Area (Normalized to SHA2)		Power (Normalized to SHA2)		Throughput (Normalized to SHA2)		Throughput/Area (Normalized to SHA2)	
		IBM130	UMC130	IBM130	UMC130	IBM130	UMC130	IBM130	UMC130
VT	SHA2	1	1	1	1	1	1	1	1
	Blake	1.60	1.41	0.45	0.29	0.94	0.73	0.59	0.52
	Groestl	3.16	3.67	0.60	0.82	1.96	2.14	0.62	0.58
	Keccak	2.20	2.08	3.09	2.88	7.25	5.93	3.30	2.85
	JH	4.79	4.44	2.74	3.11	1.66	1.35	0.35	0.30
	Skein	4.35	4.31	0.54	0.41	1.21	1.12	0.28	0.26
GMU	SHA2	1	1	1	1	1	1	1	1
	Blake	2.12	2.32	0.71	0.78	1.13	1.24	0.53	0.54
	Groestl	5.21	5.21	0.81	0.86	1.95	1.59	0.37	0.31
	Keccak	2.32	2.47	1.98	2.06	7.25	8.00	3.12	3.23
	JH	3.48	2.96	6.43	2.34	2.47	2.10	0.71	0.71
	Skein	3.28	3.56	0.73	0.82	1.44	1.59	0.44	0.45

Table B.3: Intra comparison at maximum throughput for library effect analysis

Arch	Cand Name	Area (Normalized to IBM130)		Power (Normalized to IBM130)		Throughput (Normalized to IBM130)		Throughput/Area (Normalized to IBM130)	
		IBM130	UMC130	IBM130	UMC130	IBM130	UMC130	IBM130	UMC130
VT	SHA2	1	0.79	1	0.85	1	0.90	1	1.14
	Blake	1	0.70	1	0.54	1	0.70	1	1.01
	Groestl	1	0.65	1	0.67	1	0.75	1	1.17
	Keccak	1	0.76	1	0.40	1	0.75	1	0.98
	JH	1	0.70	1	0.45	1	0.71	1	1.01
	Skein	1	0.82	1	0.80	1	0.85	1	1.05
VT	SHA2	1	0.75	1	0.71	1	0.91	1	1.20
	Blake	1	0.73	1	0.52	1	0.91	1	1.29
	Groestl	1	0.57	1	0.44	1	0.59	1	1.04
	Keccak	1	0.75	1	0.67	1	0.98	1	1.31
	JH	1	0.86	1	0.61	1	1.01	1	1.17
	Skein	1	0.71	1	0.66	1	0.89	1	1.26

Table B.4: Inter comparison at maximum throughput for library effect analysis

Arch	Cand Name	Area (Normalized to SHA2)		Power (Normalized to SHA2)		Throughput (Normalized to SHA2)		Throughput/Area (Normalized to SHA2)	
		IBM130	UMC130	IBM130	UMC130	IBM130	UMC130	IBM130	UMC130
VT	SHA2	1	1	1	1	1	1	1	1
	Blake	1.60	1.41	0.45	0.29	0.94	0.73	0.59	0.52
	Groestl	5.40	4.41	1.46	1.15	2.98	2.49	0.55	0.56
	Keccak	2.20	2.12	3.09	1.46	7.25	6.02	3.30	2.85
	JH	4.99	4.44	5.86	3.11	1.71	1.35	0.34	0.30
	Skein	4.35	4.51	0.54	0.51	1.21	1.16	0.28	0.26
GMU	SHA2	1	1	1	1	1	1	1	1
	Blake	2.38	2.32	1.05	0.78	1.24	1.24	0.52	0.56
	Groestl	6.89	5.21	1.37	0.86	2.45	1.59	0.35	0.31
	Keccak	3.19	3.16	4.53	4.31	9.30	10.00	2.92	3.17
	JH	3.48	4.00	6.43	5.55	2.47	2.75	0.71	0.69
	Skein	3.95	3.72	1.08	1.01	1.67	1.64	0.42	0.44

Table B.5: Intra comparison at minimum throughput for library effect analysis

Arch	Cand Name	Area (Normalized to IBM130)		Power (Normalized to IBM130)		Throughput (Normalized to IBM130)		Throughput/Area (Normalized to IBM130)	
		IBM130	UMC130	IBM130	UMC130	IBM130	UMC130	IBM130	UMC130
VT	SHA2	1	0.77	1	0.78	1	0.87	1	1.12
	Blake	1	0.79	1	0.75	1	0.85	1	1.07
	Groestl	1	0.87	1	0.89	1	0.83	1	0.95
	Keccak	1	0.81	1	0.91	1	0.60	1	0.73
	JH	1	0.76	1	0.75	1	0.65	1	0.86
	Skein	1	0.75	1	0.90	1	0.99	1	1.32
GMU	SHA2	1	0.79	1	0.69	1	0.91	1	1.15
	Blake	1	0.77	1	0.64	1	0.89	1	1.16
	Groestl	1	0.83	1	0.72	1	0.76	1	0.92
	Keccak	1	0.81	1	0.74	1	1.03	1	1.28
	JH	1	0.79	1	0.38	1	0.84	1	1.07
	Skein	1	0.79	1	0.77	1	0.87	1	1.10

Table B.6: Inter comparison at minimum throughput for library effect analysis

Arch	Cand Name	Area (Normalized to SHA2)		Power (Normalized to SHA2)		Throughput (Normalized to SHA2)		Throughput/Area (Normalized to SHA2)	
		IBM130	UMC130	IBM130	UMC130	IBM130	UMC130	IBM130	UMC130
VT	SHA2	1	1	1	1	1	1	1	1
	Blake	1.37	1.41	0.47	0.45	1.11	1.09	0.81	0.77
	Groestl	3.80	4.29	2.06	2.34	6.66	6.34	1.75	1.48
	Keccak	1.90	2.00	1.33	1.55	13.68	9.41	7.20	4.70
	JH	5.48	5.39	6.46	6.17	4.15	3.10	0.76	0.58
	Skein	5.05	4.92	0.52	0.60	1.32	1.51	0.26	0.31
GMU	SHA2	1	1	1	1	1	1	1	1
	Blake	2.27	2.20	1.13	1.05	1.23	1.20	0.54	0.55
	Groestl	5.95	6.20	2.23	2.35	3.58	2.99	0.60	0.48
	Keccak	3.21	3.27	2.75	2.95	6.40	7.25	1.99	2.22
	JH	3.78	3.76	2.27	1.25	1.51	1.39	0.40	0.37
	Skein	3.12	3.09	0.43	0.49	1.03	0.98	0.33	0.32

Appendix C

Normalization for Technology Effect Analysis

Table C.1: Intra comparison at maximum throughput/area for technology effect analysis

Arch	Cand Name	Area (Normalized to UMC65nm)				Power (Normalized to UMC65nm)				Throughput (Normalized to UMC65nm)				Throughput/Area (Normalized to UMC65nm)			
		65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm
VT	SHA2	1	2.24	2.51	2.08	1	0.46	0.34	1.26	1	0.67	0.33	0.30	1	0.30	0.13	0.14
	Blake	1	3.08	2.76	2.64	1	1.14	0.55	2.33	1	0.82	0.31	0.38	1	0.27	0.11	0.14
	Groestl	1	2.48	2.76	2.29	1	0.83	0.64	2.55	1	0.67	0.34	0.33	1	0.27	0.12	0.15
	Keccak	1	3.74	4.47	3.71	1	1.19	1.68	3.24	1	1.04	0.50	0.46	1	0.28	0.11	0.12
	JH	1	2.72	2.28	1.99	1	0.80	0.55	2.32	1	0.72	0.31	0.30	1	0.26	0.14	0.15
	Skein	1	2.60	2.74	2.50	1	0.43	0.30	1.06	1	0.73	0.36	0.36	1	0.28	0.13	0.14
GMU	SHA2	1	2.92	3.11	2.59	1	1.18	0.86	2.30	1	0.73	0.36	0.33	1	0.25	0.12	0.13
	Blake	1	3.04	3.08	2.75	1	1.10	0.65	2.60	1	0.80	0.40	0.37	1	0.26	0.13	0.14
	Groestl	1	3.64	3.13	2.75	1	1.38	0.70	3.00	1	1.00	0.37	0.37	1	0.27	0.12	0.15
	Keccak	1	2.22	1.82	1.82	1	0.70	0.28	2.17	1	0.66	0.26	0.31	1	0.30	0.14	0.17
	JH	1	4.45	3.42	3.42	1	1.95	0.84	3.42	1	1.32	0.50	0.50	1	0.30	0.15	0.16
	Skein	1	2.34	2.40	2.40	1	0.70	0.50	1.72	1	0.67	0.33	0.29	1	0.28	0.14	0.15

Table C.2: Inter comparison at maximum throughput/area for technology effect analysis

Arch	Cand Name	Area (Normalized to UMC65nm)				Power (Normalized to UMC65nm)				Throughput (Normalized to UMC65nm)				Throughput/Area (Normalized to UMC65nm)			
		65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm
VT	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	1.28	1.77	1.41	1.63	0.18	0.45	0.29	0.33	0.76	0.94	0.73	0.96	0.59	0.53	0.52	0.59
	Groestl	3.34	3.70	3.67	3.69	0.44	0.80	0.82	0.89	2.07	2.10	2.14	2.34	0.62	0.57	0.58	0.63
	Keccak	1.17	1.95	2.08	2.09	1.59	1.54	2.88	1.51	3.86	6.03	5.93	5.96	3.30	3.09	2.85	2.85
	JH	4.90	5.95	4.44	4.69	1.92	3.36	3.11	3.54	1.41	1.51	1.35	1.43	0.29	0.25	0.30	0.30
	Skein	3.96	4.58	4.31	4.77	0.47	0.44	0.41	0.40	1.00	1.09	1.12	1.22	0.25	0.24	0.26	0.26
GMU	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	2.34	2.44	2.32	2.48	1.02	0.96	0.78	1.16	1.12	1.22	1.24	1.25	0.48	0.50	0.54	0.50
	Groestl	5.17	6.45	5.21	5.50	1.06	1.24	0.86	1.39	1.55	2.12	1.59	1.86	0.30	0.33	0.31	0.34
	Keccak	4.21	3.20	2.47	2.97	6.33	3.76	2.06	5.97	11.07	9.98	8.00	10.16	2.63	3.12	3.23	3.48
	JH	2.68	4.10	2.96	3.55	2.42	4.00	2.34	3.59	1.51	2.72	2.10	2.27	0.56	0.67	0.71	0.71
	Skein	4.61	3.70	3.56	4.27	1.42	0.84	0.82	1.06	1.71	1.56	1.59	1.47	0.37	0.42	0.45	0.42

Table C.3: Intra comparison at maximum throughput for technology effect analysis

Arch	Cand Name	Area (Normalized to UMC65nm)				Power (Normalized to UMC65nm)				Throughput (Normalized to UMC65nm)				Throughput/Area (Normalized to UMC65nm)			
		65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm
VT	SHA2	1	2.45	2.51	2.08	1	1.17	0.34	1.26	1	0.68	0.33	0.30	1	0.28	0.13	0.14
	Blake	1	2.53	2.27	2.17	1	0.57	0.28	1.16	1	0.59	0.23	0.27	1	0.27	0.11	0.14
	Groestl	1	2.60	2.06	1.83	1	0.71	0.32	1.85	1	0.78	0.32	0.34	1	0.30	0.16	0.18
	Keccak	1	2.59	2.20	1.80	1	0.58	0.33	1.24	1	0.74	0.31	0.28	1	0.28	0.14	0.16
	JH	1	2.72	2.28	1.99	1	0.80	0.55	2.32	1	0.72	0.31	0.30	1	0.26	0.14	0.15
	Skein	1	2.43	2.59	2.26	1	1.13	0.67	1.92	1	0.68	0.34	0.33	1	0.28	0.13	0.15
GMU	SHA2	1	2.42	2.57	2.30	1	0.44	0.32	1.34	1	0.66	0.33	0.31	1	0.27	0.13	0.14
	Blake	1	2.57	2.60	2.33	1	0.54	0.32	1.26	1	0.68	0.34	0.32	1	0.26	0.13	0.14
	Groestl	1	2.00	1.55	1.84	1	1.10	0.27	1.76	1	0.71	0.24	0.33	1	0.36	0.16	0.18
	Keccak	1	2.33	2.33	1.79	1	0.70	0.59	2.17	1	0.68	0.33	0.31	1	0.31	0.16	0.17
	JH	1	2.31	2.12	1.80	1	0.65	0.66	2.54	1	0.71	0.33	0.31	1	0.31	0.16	0.17
	Skein	1	2.45	2.51	2.35	1	1.07	0.61	2.19	1	0.69	0.35	0.33	1	0.28	0.14	0.14

Table C.4: Inter comparison at maximum throughput for technology effect analysis

Arch	Cand Name	Area (Normalized to UMC65nm)				Power (Normalized to UMC65nm)				Throughput (Normalized to UMC65nm)				Throughput/Area (Normalized to UMC65nm)			
		65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm
VT	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	1.56	1.62	1.41	1.63	0.36	0.17	0.29	0.33	1.06	0.92	0.73	0.96	0.59	0.57	0.52	0.59
	Groestl	5.39	5.73	4.41	4.76	1.22	0.74	1.15	1.79	2.53	2.91	2.49	2.90	0.47	0.51	0.56	0.61
	Keccak	2.41	2.55	2.12	2.09	1.53	0.76	1.46	1.51	6.34	6.87	6.02	5.96	2.62	2.69	2.85	2.85
	JH	4.90	5.45	4.44	4.69	1.92	1.31	3.11	3.54	1.41	1.48	1.35	1.43	0.29	0.27	0.30	0.30
	Skein	4.38	4.35	4.51	4.77	0.26	0.25	0.51	0.40	1.09	1.09	1.16	1.22	0.25	0.25	0.26	0.26
GMU	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	2.29	2.44	2.32	2.32	0.79	0.96	0.78	0.74	1.19	1.22	1.24	1.21	0.52	0.50	0.54	0.52
	Groestl	8.62	7.12	5.21	6.90	1.02	2.55	0.86	1.34	2.13	2.29	1.59	2.24	0.25	0.32	0.31	0.32
	Keccak	3.49	3.36	3.16	2.73	2.36	3.76	4.31	3.82	10.07	10.24	10.00	9.78	2.89	3.05	3.17	3.59
	JH	4.86	4.65	4.00	3.80	2.72	4.00	5.55	5.16	2.70	2.88	2.75	2.65	0.55	0.62	0.69	0.70
	Skein	3.82	3.87	3.72	3.92	0.53	1.29	1.01	0.87	1.56	1.62	1.64	1.65	0.41	0.42	0.44	0.42

Table C.5: Intra comparison at minimum throughput for technology effect analysis

Arch	Cand Name	Area (Normalized to UMC65nm)				Power (Normalized to UMC65nm)				Throughput (Normalized to UMC65nm)				Throughput/Area (Normalized to UMC65nm)			
		65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm
VT	SHA2	1	3.08	3.27	2.79	1	0.93	0.69	2.95	1	0.94	0.40	0.43	1	0.31	0.12	0.15
	Blake	1	3.22	3.38	2.85	1	1.27	0.65	2.63	1	1.08	0.39	0.45	1	0.34	0.11	0.16
	Groestl	1	3.27	3.81	3.11	1	1.43	1.12	4.26	1	1.16	0.52	0.50	1	0.35	0.14	0.16
	Keccak	1	2.99	3.42	2.77	1	0.77	0.35	3.02	1	0.82	0.30	0.48	1	0.27	0.19	0.17
	JH	1	2.93	3.34	2.64	1	1.31	1.23	4.67	1	1.11	0.50	0.54	1	0.38	0.15	0.20
	Skein	1	3.15	3.35	2.77	1	1.13	0.44	1.55	1	1.08	0.39	0.46	1	0.34	0.12	0.17
GMU	SHA2	1	3.26	3.27	2.95	1	1.12	0.81	3.36	1	0.98	0.42	0.42	1	0.30	0.13	0.14
	Blake	1	3.22	3.16	2.86	1	1.31	0.71	3.07	1	1.07	0.39	0.44	1	0.33	0.12	0.15
	Groestl	1	3.19	3.50	2.99	1	1.22	0.85	3.53	1	0.93	0.43	0.45	1	0.29	0.12	0.15
	Keccak	1	3.09	3.35	2.87	1	0.82	0.73	2.25	1	0.94	0.44	0.35	1	0.30	0.13	0.12
	JH	1	3.03	3.27	2.81	1	0.85	0.30	2.54	1	0.77	0.35	0.37	1	0.25	0.11	0.13
	Skein	1	3.35	3.25	3.00	1	1.14	0.37	1.47	1	1.08	0.38	0.44	1	0.32	0.12	0.15

Table C.6: Inter comparison at minimum throughput for technology effect analysis

Arch	Cand Name	Area (Normalized to UMC65nm)				Power (Normalized to UMC65nm)				Throughput (Normalized to UMC65nm)				Throughput/Area (Normalized to UMC65nm)			
		65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm	65nm	90nm	130nm	180nm
VT	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	1.37	1.43	1.41	1.40	0.48	0.64	0.45	0.43	1.14	1.30	1.09	1.18	0.83	0.91	0.77	0.85
	Groestl	3.69	3.92	4.29	4.11	1.45	2.22	2.34	2.10	4.93	6.06	6.34	5.70	1.34	1.55	1.48	1.39
	Keccak	1.91	1.86	2.00	1.90	3.08	2.53	1.55	3.15	12.84	11.14	9.41	14.34	6.77	6.00	4.70	7.53
	JH	5.28	5.02	5.39	4.99	3.49	4.89	6.17	5.53	2.51	2.95	3.10	3.15	0.48	0.59	0.58	0.63
	Skein	4.57	4.68	4.69	4.55	0.37	0.45	0.24	0.19	0.69	0.79	0.66	0.74	0.15	0.17	0.14	0.16
GMU	SHA2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Blake	2.28	2.25	2.20	2.20	1.19	1.39	1.05	1.09	1.30	1.42	1.20	1.36	0.57	0.63	0.55	0.62
	Groestl	5.80	5.80	5.68	6.20	5.89	2.23	2.43	2.35	2.34	2.92	2.76	2.99	3.13	0.50	0.49	0.48
	Keccak	3.19	3.02	3.27	3.10	3.25	2.38	2.95	2.17	6.88	6.60	7.25	5.67	2.16	2.18	2.22	1.83
	JH	3.77	3.51	3.76	3.59	3.35	2.54	1.25	2.53	1.67	1.31	1.39	1.46	0.44	0.37	0.37	0.41
	Skein	3.11	3.20	3.09	3.16	1.06	1.08	1.00	0.46	1.08	1.19	0.98	1.14	0.35	0.37	0.32	0.36

Appendix D

Normalization for Technology Effect Analysis

Table D.1: Intra comparison for post-layout analysis

Arch	Candidate Name	Area (Normalized to Square)					Power (Normalized to Square)					Throughput (Normalized to Square)					Throughput/Area (Normalized to Square)				
		Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1	Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1	Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1	Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1
VT	Blake	1	0.98	0.98	0.98	0.97	1	0.98	0.99	0.97	0.98	1	1.02	0.96	1.04	0.99	1	1.04	0.98	1.04	0.98
	Greostl	1	0.97	0.98	0.96	0.98	1	0.99	0.97	1.01	0.96	1	0.99	0.99	0.99	1.00	1	1.02	1.01	1.02	1.01
	Keccak	1	1.01	1.04	0.99	0.79	1	1.02	1.05	1.02	1.07	1	0.98	0.89	0.99	0.91	1	0.97	0.86	0.97	0.86
	JH	1	0.98	0.98	0.96	1.01	1	0.99	0.99	0.98	0.99	1	0.86	0.97	0.88	0.99	1	0.88	0.99	0.88	0.99
	Skein	1	1.01	1.00	1.03	0.94	1	1.02	0.99	1.04	0.93	1	0.92	0.87	0.91	0.86	1	0.92	0.87	0.92	0.87
GMU	Blake	1	1.01	1.02	1.00	1.01	1	1.01	1.01	1.01	0.99	1	1.00	1.00	0.99	1.00	1	0.98	0.98	0.99	0.98
	Groestl	1	0.95	0.98	0.91	1.01	1	0.98	1.00	0.91	1.01	1	0.97	0.98	0.99	0.91	1	1.01	0.99	1.02	0.98
	Keccak	1	0.99	1.01	0.96	0.99	1	0.78	1.00	0.81	0.99	1	1.14	0.91	1.09	0.99	1	1.15	0.91	1.09	0.99
	JH	1	0.99	1.00	1.00	1.01	1	0.99	1.01	0.92	0.99	1	0.98	0.98	0.91	1.02	1	1.00	0.98	0.96	1.01
	Skein	1	0.96	0.96	0.91	1.01	1	0.97	0.99	0.92	0.99	1	0.95	0.91	0.99	0.89	1	0.99	0.96	1.01	0.91

Table D.2: Inter comparison for post-layout analysis

Arch	Candidate Name	Area (Normalized to Square)					Power (Normalized to Square)					Throughput (Normalized to Square)					Throughput/Area (Normalized to Square)				
		Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1	Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1	Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1	Square	Rec1:2	Rec1:3	Rec2:1	Rec3:1
VT	Blake	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Groestl	4.09	4.06	4.09	4.07	4.09	4.03	4.08	3.98	4.06	4.01	3.37	3.27	3.45	3.21	3.44	0.82	0.81	0.85	0.81	0.85
	Keccak	1.71	1.77	1.81	1.79	1.83	9.19	9.56	9.80	9.54	9.8	6.36	6.11	5.92	6.08	6.01	3.71	3.45	3.26	3.45	3.26
	JH	3.11	3.12	3.11	3.11	3.13	13.53	13.74	13.59	13.76	13.58	1.81	1.53	1.81	1.48	1.83	0.58	0.49	0.58	0.49	0.58
	Skein	3.69	3.79	3.75	3.8	3.71	1.48	1.55	1.48	1.55	1.52	1.19	1.08	1.08	1.09	1.08	0.32	0.28	0.29	0.28	0.29
GMU	Blake	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Groestl	3.85	3.61	3.72	3.67	3.7	1.42	1.37	1.42	1.38	1.39	1.82	1.76	1.79	1.75	1.81	0.47	0.49	0.47	0.48	0.51
	Keccak	1.63	1.59	1.61	1.56	1.63	4.12	3.19	4.10	3.2	4.08	6.77	7.72	6.19	7.7	6.23	4.11	4.85	3.79	4.89	3.76
	JH	1.65	1.61	1.62	1.61	1.63	3.83	3.77	3.83	3.78	3.92	1.88	1.85	1.84	1.85	1.85	1.13	1.15	1.12	1.16	1.08
	Skein	1.92	1.82	1.81	1.81	1.91	1.18	1.14	1.16	1.15	1.20	1.33	1.27	1.21	1.31	1.28	0.69	0.69	0.67	0.70	0.66

Bibliography

[1] F. Rodríguez-Henríquez, N. A. Saqib, A. Díaz-Pèrez, and C. K. Koc, *Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[2] K. Kobayashi, J. Ikegami, M. Knezevic, E. Guo, S. Matsuo, S. Huang, L. Nazhandali, U. Kocabas, J. Fan, A. Satoh, I. Verbauwhede, K. Sakiyama, and K. Ohta, "Prototyping platform for performance evaluation of sha-3 candidates," in *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, june 2010, pp. 60 –63.

[3] G. Xu, H. Sinan, N. Leyla, and S. Patrick, "Fair and Comprehensive Performance Evaluation of 14 Second Round SHA-3 ASIC Implementations," in *The Second SHA-3 Candidate Conference*, August 2010.

[4] X. Guo, M. Srivastav, S. Huang, D. Ganta, M. Henry, L. Nazhandali, and P. Schaumont, "Pre-silicon characterization of nist sha-3 final round candidates," in *Digital System Design (DSD), 2011 14th Euromicro Conference on*, 31 2011-sept. 2 2011, pp. 535 –542.

[5] M. Henry, S. Griffin, and L. Nazhandali, "Fast simulation framework for subthreshold circuits," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*, may 2009, pp. 2549 –2552.

[6] N. Sklavos, "Towards to sha-3 hashing standard for secure communications: On the hardware evaluation development," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 10, no. 1, pp. 1433 –1434, jan. 2012.

[7] L. Han and B. Guoqiang, "Hardware implementation analysis of sha-3 candidates algorithms," in *Solid-State and Integrated Circuit Technology (ICSICT), 2010 10th IEEE International Conference on*, nov. 2010, pp. 266 –268.

[8] D. Lee, "Hash function vulnerability index and hash chain attacks," in *Secure Network Protocols, 2007. NPSec 2007. 3rd IEEE Workshop on*, oct. 2007, pp. 1 –6.

[9] D. Webster and M. Lukowiak, "Versatile fpga architecture for skein hashing algorithm," in *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, 30 2011-dec. 2 2011, pp. 268 –273.

[10] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[11] D. Markovic, C. Wang, L. Alarcon, T.-T. Liu, and J. Rabaey, "Ultralow-power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237 –252, feb. 2010.