ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DIGITAL DESING AND IMPLEMENTATION OF GENERAL PURPOSE CONTROL AND COMMUNICATION MODULES FOR ACTIVE SHIELD

M.Sc. THESIS

Ömer Faruk BİRGÜL

Department of Electronics and Communication Engineering

Electronics Engineering Programme

JULY 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DIGITAL DESING AND IMPLEMENTATION OF GENERAL PURPOSE CONTROL AND COMMUNICATION MODULES FOR ACTIVE SHIELD

M.Sc. THESIS

Ömer Faruk BİRGÜL
(504211220)

Department of Electronics and Communication Engineering

Electronics Engineering Programme

Thesis Advisor: Prof. Dr. Müştak Erhan YALÇIN

JULY 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

AKTİF KALKAN İÇİN GENEL AMAÇLI KONTROL VE HABERLEŞME MODÜLLERİNİN SAYISAL TASARIMI VE UYGULANMASI

YÜKSEK LİSANS TEZİ

Ömer Faruk BİRGÜL
(504211220)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Müştak Erhan YALÇIN

TEMMUZ 2024

Ömer Faruk BİRGÜL, a M.Sc. student of ITU Graduate School student ID 504211220 successfully defended the thesis entitled "DIGITAL DESING AND IMPLEMENTATION OF GENERAL PURPOSE CONTROL AND COMMUNICATION MODULES FOR ACTIVE SHIELD", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Müştak Erhan YALÇIN** Istanbul Technical University	.....
Jury Members :	**Doç. Dr. Onur FERHANOĞLU** Istanbul Technical University	.....
	Dr. Öğr. üyesi Tuba Ayhan MEF University	.....

Date of Submission : **24 May 2024**
Date of Defense : **23 July 2024**

To my lovely wife and my family,

Preface

First of all, I want to express my gratitude to Prof. Dr. Müştak Erhan Yalçın for his endless support on this journey. I also want to thank Dr. Emre Göncü and Emrah Abtioğlu for supporting me when I realized the project.

Finally, my lovely wife and my family, have been always with me and supported me all the time.

Part of this project is realized in the university-industry collaboration project with İTÜ, TÜBİTAK, and Procenne with 5210073 numbered project.

July 2024 Ömer Faruk BİRGÜL

TABLE OF CONTENTS

Page

ABBREVIATIONS

AES	**:** Advanced Encryption Standard
CBC	**:** Cipher Block Chaining
FIB	**:** Focused Ion Beam
FPGA	**:** Field Programmable Gate Array
FSM	**:** Finite State Machine
GDS	**:** Graphical Design System
HDL	**:** Hardware Description Language
IC	**:** Integrated Circuit
IV	**:** Initial Vector
LUT	**:** Lookup Table
NIST	**:** National Institute of Standards and Technology
PRNG	**:** Pseudo Random Number Generator
SPI	**:** Serial Peripheral
SPN	**:** Substitution-permutation Network
SW-RST	**:** Software Reset
TRNG	**:** True Random Number Generator

SYMBOLS

m **:** NUMBER_OF_BUSES
n **:** BUS_WIDTH

LIST OF TABLES

LIST OF FIGURES

DIGITAL DESING AND IMPLEMENTATION OF GENERAL PURPOSE CONTROL AND COMMUNICATION MODULES FOR ACTIVE SHIELD

SUMMARY

At present, relationship between humanity and digital technology is getting more more stronger. People increasingly manage their lives online. The increasing percentage of usage of digital systems brings risks. Some of the risks are in banking and digital citizenship applications. Because of the value of information in these areas, they have crucial importance for security applications. One of the security applications is the Hardware Security Module (HSM). Hardware Security Modules have crucial importance, especially in the realm of digital banking.

An active shield is adopted in these security modules as a protection layer. An active shield that has the feature of detecting and resisting microprobing attacks from attackers is one of the main components of the Hardware Security Module.

In this thesis, a fully parametrized active shield digital design for integrated circuit (IC) is implemented. This design has seven subblocks: Serial Peripheral Interface (SPI), shield control, Pseudo-Random Number Generator (PRNG), shield driver, shield receiver, shield bus select, comparator, and register. All these modules except SPI are parametric; therefore, any dimension of the maze of the shield can be used with this digital design. The shield digital design has SPI ports and an alarm output port. Data coming over SPI is written to the register via the shield control module. Also, the saved data in the register is read by the SPI. A pseudo-random number generator generates random numbers to use in the maze of shields. The shield driver transmits random numbers to the maze of shields and the shield receiver receives random numbers from the maze of shields. The shield bus select selects the correct bus on the maze of shields. The comparator compares values from both the maze of shields and a pseudo-random number generator. In the design, when there is an attack on the active shield, an alarm triggers. In this study, the attack is detected by the mismatch between the values from the maze of shield and values from the pseudo-random number generator. The alarm triggers when an attack occurs.

The previous studies on random number generators are analyzed for use in the active shield design. The system has to be lightweight, so it is decided to use pseudo-random number generators instead of true random number generators. Lightweight ciphers are used in cipher block chaining (CBC) mode to generate pseudo-random number generators. For that purpose, lightweight ciphers in the first round of National Institute of Standards and Technology (NIST) lightweight cryptography standardization were scrutinized. The five ciphers with the smallest area are selected. These five ciphers were realized in Python as pseudo-random number generators in cipher block chaining mode. Random numbers generated with these pseudo-random number generators were

tested with the NIST random number test suite. Also, runtime in Python to produce random numbers is measured for comparison. SIMON cipher was selected for use in the active shield design for random number generation according to the NIST test and runtime.

In this study, four parameters are used in the design of the active shield. Half of the parameters are obtained from SIMON parameters. These parameters ensure that every size of the active shield can be controlled by the shield's digital design. All designs except SPI are parametric. All designs in the active shield except SPI were designed in this thesis.

Implemented designs are tested in field programable gate arrays (FPGAs), and the attack to shield is simulated. It is observed that an attack happened. This test with FPGA proves that the implemented design is working as expected.

To observe the effects of parameters in the design, different configured versions of parameters were synthesized, and the results show that SIMON block size has a direct relationship with the size of the shield digital design.

AKTİF KALKAN İÇİN GENEL AMAÇLI KONTROL VE HABERLEŞME MODÜLLERİNİN SAYISAL TASARIMI VE UYGULANMASI

ÖZET

Teknolojinin gelişmesiyle günümüzde neredeyse tüm hayatımız dijital hale gelmiştir. Bu dijital hale gelen hayatımıza doğrudan etkili olan finans sektörü ile dijital vatanlaşlık uygulamaları hayatımızı oldukça kolaylaştırmaktadır. Ancak bu kolaylık beraberinde güvenlik risklerini de getirmektedir. Bu riskler çok çeşitli olabililmektedir.

Bankacılık uygulamalarında ve dijital vatandaşlık uygulamalarında donanım güvenliği, bu riskler dolayısıyla öne çıkmaktadır. Donanımlara yapılan saldırılar çeşitli şekillerde gerçekleştirilebilmektedir. Bu saldırılaran donanımlardan birisi olan entegre devreler, içerisinde barındırdığı önemli verilerin açığa çıkmasının getireceği olumsuz sonuçlar sebebiyle yüksek öneme sahiptir.

Entegre devrelere çeşitli şekillerde saldırılar yapılabilmektedir. Bunlar bozucu olmayan ve bozucu saldırılar şeklinde iki ana kategoriye ayrılabilir. Bozucu olmayan saldırılar da pasif bozucu olmayan saldırılar ve aktif bozucu olmayan saldırılar olarak ikiye ayrılabilir. Bozucu olmayan saldırıların genel özelliği, saldırılan entegre devreye fiziksel bir hasar vermeden içerisindeki veri almaktır. Bozucu saldırıların genel özelliği ise saldırı yapılan entegre devreye fiziksel zarar verilerek yapılan saldırı olmasıdır. Bozucu saldırılara örnek olarak mikroprobing saldırısı gösterilebilir.

Mikroprobing saldırısında bir entegre devreye, mikroskobik boyuttaki problarla temas edilir ve içerisindeki veriler okunmaya çalışılır. Bu saldırıyı gerçekleyebilmek için, entegre devrenin paketinin çeşitli yöntemlerle aşındırılarak içerisinde bulunun metal hatlara fiziksel erişimin sağlanabilmesi gereklidir. Buna karşı savunma yöntemi olarak da aktif kalkan kullanılmaktadır.

Aktif kalkan, entegre devreye dışarıdan gelen mikroskobik saldırıları tespit edebilme kabiliyeti sayesinde, Donanımsal Güvenlik Modüllerinin ana bileşenlerinden birisi olmaktadır. Donanımsal Güvenlik Modülleri (HSM), günümüzde özellikle dijital bankacılığın yükselişiyle bu sektörde sektöründe yüksek öneme sahiptir. Bu modüllerde fiziksel koruma katmanı olarak aktif kalkan kullanılmaktadır.

Bu tezde, tüm özellikleri parametrik hale getirilmiş bir kırmık üstü aktif kalkan tasarımı yapılmıştır. Bu sayede ihtiyaca göre istenilen büyüklükte kalkan tasarımı yapılabilmektedir. Kalkanın sayısal tasarımında; kontrol, sözde rastgele sayı üreteci, hat seçici, sürücü, alıcı, karşılaştırıcı ve kaydedici blokları mevcuttur. Bu bloklar, sistemin düzenli çalışabilmesi adına birbiriyle uyumlu ve en önemlisi parametrik olarak tasarlanmıştır.

Aktif kalkan tasarımında kullanılmak üzere rastgele sayı üretimi ihtiyacını karşılabilmek için literatürde bulunan rastgele sayı üreteciler incelenmiştir. Sistemin hafif

olması arzulandığından, gerçek rastgele sayı üreteciler yerine sözde rastgele sayı üreteciler kullanılmaya karar verilmiştir. Sözde rastgele sayı üretimi ise, hafif şifreleyicileri CBC yöntemiyle kullanarak gerçekleştirildi. Bu amaçla, NIST hafif şifreleme standardizasyon sürecinin birinci turundaki şifreleyiciler incelenmiş ve alan tüketimi en az olan 5 adet şifreleyici olarak SIMON, SPECK, PRINT cipher, LRBC, TinyJAMBU seçilmiştir. Seçilmiş olan 5 şifreleyici ile Pyhon ile rastgele sayılar üretecek şekilde gerçeklenmiş ve bu üretilen sayılar NIST testine tabi tutulmuştur. Ayrıca, tüm rastgele sayı üreticiler ile eşit uzunlukta bir veya sıfır ürettirilerek çalışma süreleri ölçülmüştür. Bu ölçüm 100 farklı rastgele giriş ile tekrarlanarak çalışma sürelerinin ortalamaları alınmıştır. çalışma süresi ve NIST testi sonucu karşılaştırma metriği olarak alınmıştır. Testlerin sonucunda; SIMON 42dk çalışma süresi ve %6 kalma oranı, SPECK 47dk çalışma süresi ve %10 kalma oranı, PRINTcipher 463dk çalışma süresi ve %11 kalma oranı, LRBC 155dk çalışma süresi ve %100 kalma oranı, TinyJAMBU 1000dk çalışma süresi ve %100 kalma oranı değerlerine sahip olmuştur. Bu metriklere göre, SIMON en uygun olarak seçilmiştir.

Aktif kalkanın tasarımında SIMON şifreleyicisinin blok genişliği ve anahtar genişliği kombinasyonları temel alınarak SPI devresi hariç diğer bütün devreler parametrik olarak tasarlanmıştır. Kontrol bloğu, sistemin SPI girişinden gelen verileri kaydediciye iletir ve sistemin durumunu kontrol ederek gerekli şekilde yönlendirmeler yapar. Sözde rastgele sayı üreteci, kalkanın metal hat kısmına iletilecek rastgele sayıları üretir. Hat seçici, kalkan üzerindeki metal hatların hangilerinin seçileceğini belirler. Sürücü, sözde rastgele sayı üretecinden aldığı rastgele sayıları hat seçicinin değerine göre kalkan metal hatlarına iletir. Alıcı, kalkan metal hatlarından gelen veriyi hat seçicinin değerine göre karşılaştırıcıya iletir. Karşılaştırıcı, metal hatlardan gelen rastgele sayılar ile sözde rastgele sayı üretecinden gelen rastgele sayıları karşılaştırır. Karşılaştırma sonucunda eğer eşitsizlik varsa bu durumda alarm tetiklenir. Son olarak SPI, kalkan sayısal tasarımı ile dış dünyanın haberleşmesi amacıyla kullanılmıştır.

Sistemin parametrik olması amacıyla çeşitli parametreler kullanılmıştır. Bu parametrelerin iki tanesi SIMON şifreleyicisinden gelir. Diğer iki tanesi, kullanıma uygunluk amacıyla çalışma esnasında belirlendi. SIMON blok uzunluğu, parametrelerden bir tanesidir. Bu değer, SIMON şifreleyicisinin blok uzunluğudur ve şifreleyiciye verilebilecek düz metin uzunluğunu belirtir. SIMON anahtar uzunluğu da SIMON blok uzunluğuna benzer şekilde çalışır. Bu değer, SIMON şifreleyicisinin anahtar uzunluğudur ve şifreleyiciye verilebilecek anahtar uzunluğunu belirtir. Hat genişliği, bir seferde sürülebilecek olan metal sayısıdır ve bu değer en yüksek SIMON blok uzunluğu olabilir. Hat sayısı, metal hatta bir seferde sürülebilecek hatlardan kaç tane olduğunu belirtir. Bu değer sonsuza kadar çıkabilir. Hat sayısı ile hat genişliğini çarptığımız zaman, kalkan metal hattında toplam kaç farklı yol olduğunu belirtir.

Kalkan sayısal tasarımı ile üretilmiş entegre devrenin, bir saldırıdan sonra devre dışı kalmaması, tekrar tekrar kullanılıp test yapılabilmesi amacıyla maskeleme sistemi gelişitirlmiştir. BU maskeleme sistemi, yapılan saldırıların kaydını tutar ve saldırı yapılmış bu metal yolların sürekli alarm üretmesini engeller. Tüm bu kayıtlı bilgileri okuyabilmek için alarm okuma sistemi kurulmuştur. Alarm okuma sistemi aracılığıyla, önceki saldırılarda hangi metal yolunun zarar gördüğü belirlenebilir.

Tamamlanmış kalkan sayısal tasarımı, FPGA üerinde test edilmiş ve saldırı benzeri sistem kurularak tasarımın doğru çalışıp çalışmadığı incelenmiştir. Yapılan testler sonucunda, FPGA üzerindeki sayısal tasarımın doğru çalıştığı tespit edilmiştir.

Parametrik tasarımın etkilerini incelemek amacıyla parametreler farklı kombinasyonlarda ayarlanarak sentez yapılmıştır. Bu sentezler sonucunda, SIMON blok uzunluğu arttıkça sistemin doğrusal büyüdüğü gözlenmiştir. Aynı şekilde hat sayısı artırıldığında da doğrusal bir büyüme gözlenmiştir.

1. INTRODUCTION

The recent trend indicates a substantial shift towards a digital lifestyle, with a significant portion of our time now being devoted to the online world, which is digital. There are two digital areas, banking and citizenship operations. Digitalization makes people's lives better. Daily duties are digitalizing and becoming online recently. Becoming online allows people to do their tasks faster and easier. But, there is an important drawback of digitalization: Security. Governments, banks, and even homes are getting digitalized. When digitalization is getting popular, security concerns stand out more. Security breaches can cause physical and financial harm as well as psychological harm. In these areas, communication and storing data is crucial. The security breaches cause physical and financial harm as well as psychological harm.

Semiconductor technology provides people with digitalization. The integrated circuit (IC) is the main component of the semiconductor technology. The security of integrated circuits for transferring or storing data is also very critical. Since the data in the IC is very valuable, it appeals to attackers. Attackers have generated several methods to attack microchips while designers develop their security.

1.1 Methods of Attacks

There are several methods of attack on integrated circuits. These methods can differ according to the purpose of the attack. These purposes are getting crucial information or getting cryptographic keys from integrated circuits. There are two major categories of attacks: invasive attacks and non-invasive attacks [1]. Non-invasive attacks can be categorized as active and passive non-invasive attacks [2].

Passive non-invasive attacks are types of attacks in which the attacker does not influence the running IC. Electromagnetic analysis attacks and power analysis attacks are two examples of passive non-invasive attacks [2].

Active non-invasive attacks influence IC, but physical structures remain unchanged. Fault injection attacks, focused ion beam (FIB) attacks, temperature variation attacks, and power glitching attacks are some examples of active non-invasive attacks [2].

Invasive attacks influence the physical structure of IC. In those attacks, attackers try to obtain information by changing the IC structure. Compared with other types of attacks, invasive attacks are expensive because high-quality equipment is needed to apply invasive attacks to IC. Decapsulation, modification, and microprobing are three examples of invasive attacks [2].

In this thesis, active shield design is implemented against microprobing attacks. There is a microscopic probe used in the micro probing attacks, and this microprobe reads the values of metal lines inside the IC [3].

1.2 Purpose of Thesis

In this thesis, a scalable and lightweight digital design of an active shield is implemented and tested in an FPGA against microprobing attacks. The main purpose of this design is to adapt to different bit widths of shield maze structures. Implemented digital design can drive up to thousands of bits of shield maze structure. This scalability gives flexibility in shield design and reduces the limits of the reusability of the shield. Due to its lightness, the design can be used in very small areas.

In the Figure 1.1, a simplified version of the shield can be seen. Shield Control Digital Design will be implemented in this thesis. Shield Digital Design will be coded in Verilog, and SPI will be used as an IP [4]. Turgut and Kurt designed and implemented the maze structure of the shield project [5] [6].

1.3 Literature Review

There are several protection methods. Some of these methods are active shield, driving of shield, and generating complex mazes. Some studies only propose new methods to produce complex maze structures and algorithms.

Brias and et. al. focused on the structure of the maze in their studies. Their research shows that when the maze is made more complex, the rate of unsuccessful attacks

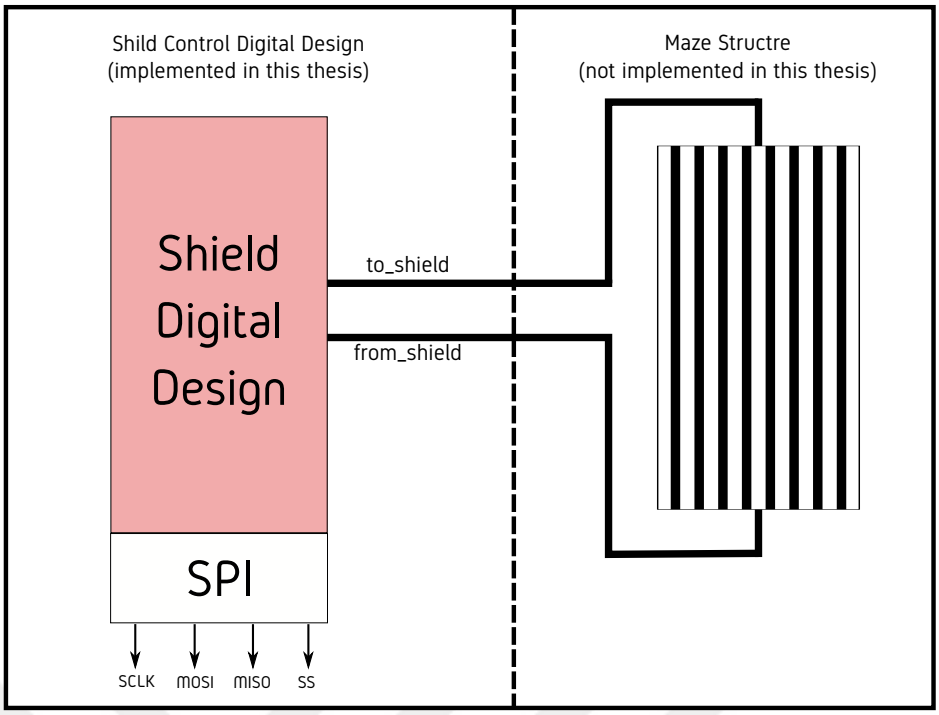


Figure 1.1 : Simplified view of shield digital design and maze structure.

increases, and if there are two layers in the maze, all attacks fail [7]. Increasing the complexity of the maze, it is aimed to protect unprotected areas against FIB attacks.

In another study, the maze structure is not solid but it is reconfigurable [8]. Therefore complexity of the maze will be increased by changing the maze constantly. The complexity of the maze is a measure of the ability to resist attack.

Mun and et. al. design a solid maze structure that has randomly changing input lines [1]. In their study, the complexity is increased by changing inputs of the maze structure.

There are also studies in the literature that only worked on the maze structure [9] [10] [11] [12]. However, there is no study analyzing the digital design of the shield.

1.4 Organization of Thesis

In this thesis, the digital design of the active shield against micro probing attacks, implementation, and necessary pseudo-random number generator (PRNG) production are reviewed. The first chapter has an introduction, methods of attacks, the purpose of the thesis, and the literature review. The lightweight ciphers, the general PRNG

structure, the PRNG generation by using the selected ciphers, and the comparison of these PRNG's explained in section two. The third chapter explains the design, operation, and implementation of active shield digital design explained in depth module by module, and also the scalability of the system by using parameters. In the fourth chapter, all results of this work are concluded.

2. LIGHTWEIGHT CIPHER BASED PRNG

2.1 PRNG Structure

Random number generators are used for generating random numbers by using several methods either deterministic or random noises from unpredictable variables of the physical environment. There are two types of random number generators according to their origin: true random generators (TRNG) and pseudo-random generators (PRNG). True random generators use the randomness of the physical world to generate random numbers. They are expected to be mathematically random and unpredictable. Pseudo-random numbers are generated according to mathematical equations. They are deterministic, and results can be calculated if inputs are known.

There are several methods to create the PRNG. In this thesis, the cipher block chaining (CBC) mode is used to create the PRNG. In the CBC mode, the lightweight ciphers are used sequentially. The Figure 2.1 shows that the operation of CBC mode. At the beginning, the initial vector (IV) as the first plaintext and the key are given to the first cipher. The first cipher produces the first encrypted output. This output is the first random number. The second cipher uses the first random number as the plaintext and same the key, and it produces the second random number. This process continues until it is interrupted. Interrupt can be caused by a user or another process in the system.

2.2 Lightweight Ciphers

Ciphers are structures designed to transform plain text into encrypted text using a specific key. Lightweight ciphers are characterized by their minimal area usage and low power consumption. This study conducted a comprehensive review of the literature to identify a suitable lightweight cipher for producing PRNG. During the review, ciphers were analyzed based on their respective areas, with gate equivalent area being used as the primary comparison metric.

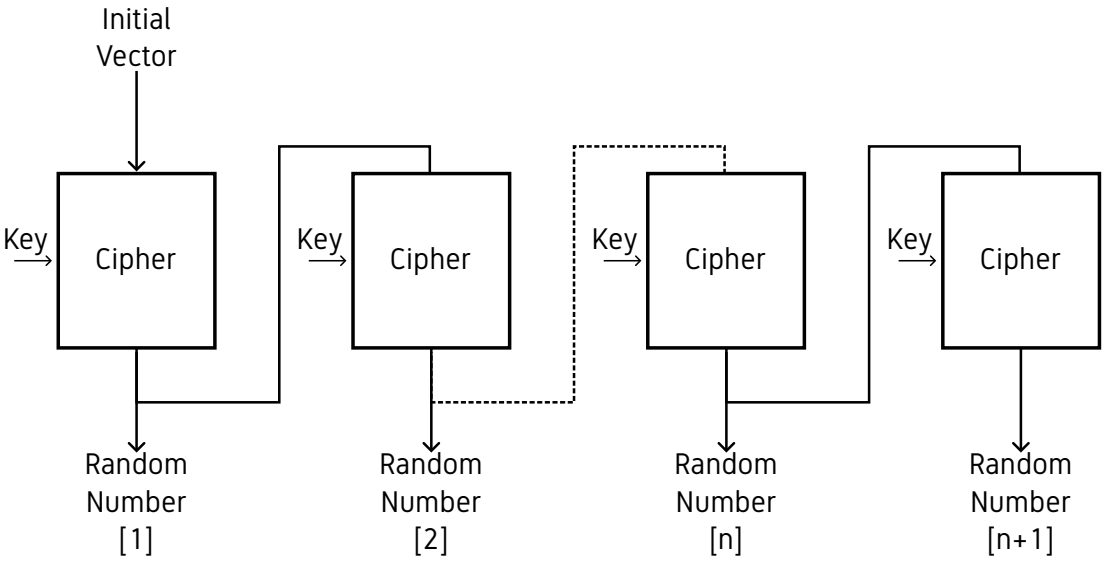


Figure 2.1 : CBC mode to produce PRNG.

As a starting point, the surveys about lightweight ciphers are reviewed. The Table 2.1 shows a comparison including the Advanced Encryption Standard (AES) and the other lightweight ciphers [13]. According to this comparison, the PRESENT cipher is smaller than the others in terms of the area.

Table 2.1 : Result of the implementation of the ciphers.

Cipher	Mode E,D,ED	Area 2 [μm]	fmax [MHz]	Latency [cycles]	Throughput [Mbps]	Power [mW]	Energy [pJ per bit]
AES	E	17921	444	12	4740	13,5	2,9
Nr = 1	D	20292	377	22	2195	10,6	4,8
	ED	24272	363	≈17	≈2997	≈12,6	≈4,4
NOEKEON	E	8011	1149	18	8173	15,0	1,8
Nr = 1	D	10431	1075	19	7243	14,1	1,9
	ED	10483	1075	≈18,5	≈7445	≈15,35	≈ 2,1
HIGHT	E	6524	641	19	2159	6,3	2,9
Nr = 2	D	6524	645	19	2173	6,3	2,9
	ED	8217	540	19	1820	6,1	3,3
ICEBERG	E	11377	699	17	2632	10,7	4,0
Nr = 1	D	11359	699	17	2632	10,7	4,0
	ED	11408	689	17	2596	10,6	4,0
KATAN	E	6231	952	17	3585	8,1	2,7
Nr = 16	D	8616	666	33	1292	9,8	6,1
	ED	12609	473	≈25	≈1347	≈12,7	6,4
PRESENT	E	5024	1123	17	4230	9,3	2,2
Nr = 2	D	6060	1041	33	2020	8,9	4,4
	ED	8213	884	≈25	≈2523	≈12,6	4,7

In the other paper for surveying the lightweight ciphers for the embedded systems shows that the WG-8 cipher is better in terms of the area in the Table 2.2 [14].

Table 2.2 : Result of the implementation of the ciphers.

Cipher	Key size (bits)	Block size (bits)	IV (bits)	Latency (cycles / block)	Throughput at 100 KHz (MBps)	Tech (μm)	Area (GE)	FOM
Better is				Lower	Higher		Lower	Higher
WG-8 [15]	80	11	80	1	1100	0.65	3942	0.00884
Enocoro-128 [16]	128	1	64	1	800	0.09	4100	0.00594
WG-8 [15]	80	1	80	1	100	0.65	1786	0.00391
Enocoro-80 [16]	80	1	80	1	—	0.09	2700	—
AES [17]	128	128	—	226	56.64	0.35	2400	0.00122
Salsa20 [18]	256	32	64	2	99	0.13	12126	0.00008
HC-256 [18]	256	—	256	—	—	0.13	>> 524000	—

One of the study examined a total of 14 distinct ciphers, which encompassed the evaluation of SIMON alongside its associated algorithm, SPECK [19]. The comparison of these ciphers in terms of the gate equivalent can be seen at the Figure 2.2.

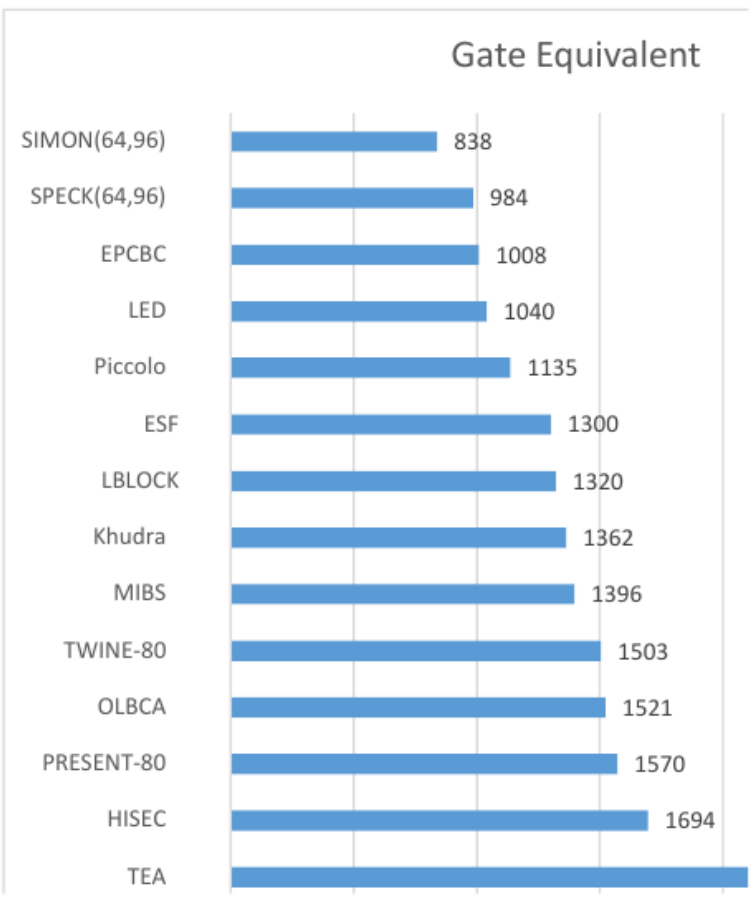


Figure 2.2 : Result of the implementation of the ciphers.

Another survey reviewed the 5 different ciphers including SIMON with SIMON's different block and key size [20]. The results of this work can be seen at the Table 2.3.

Table 2.3 : Result of the implementation of the ciphers.

Block Ciphers	Block Size	Key Size	Area (GEs)	Throughput	No. of Rounds	Made of
PRESENT	128	80/128	1000	High	32	SPN
KATAN	32/48/ 64	80	802	Low	254	Feiste
Humming Bird	16	256	2159	High	4	SPN
SIMON	32	64	739	High	32	Feistel
	48	72/96	809		36	
	64	96/1289	958		42/44	
	96	96/144	955		52/54	
	128	128/192/256	1234		68/69/72	
RECTANGLE	64	80	1600	High	25	SPN

There is a study that the LRBC was proposed and also 12 different ciphers were compared including SIMON, SPECK, PRINT, and LRBC itself [21]. Table 2.4 shows that LRBC, PRINT, SPECK, and SIMON have minimum gate equivalent areas.

Table 2.4 : Result of the implementation of the ciphers.

Ciphers	Block size (bit)	Key size (bit)	Structure	Round	Logic process (μm)	Max. frequency (KHz)	Area (GE)
AES [22]	8	128	SPN	16	0.13	153,000	3100
PRESENT [23]	64	80	SPN	32	0.18	100	1570
LED [24] [25]	64	128	SPN	48	0.13	78,130	3407
PRINCE [26]	64	128	SPN	12	0.13	100	3491
TEA [27]	64	128	Feistel	32	0.35	100	1140
QTL [28]	64	64	Feistel, SPN	16	0.18	100	1026
RECTANGLE [29]	64	80	SPN	26	0.13	100	1599.5
SIMON [30]	32	64	Feistel	32	0.13	100	523
SPECK [30]	32	64	Feistel	22	0.13	100	580
SIMECK [31]	32	64	Feistel	32	0.065	100	488
PRINT	48	80	SPN	48	0.18	100	402
Proposed LRBC	16	16	Feistel, SPN	24	0.065	100	258.9

There is the lightweight cryptography standardization process organized by NIST [32]. In this process, 57 ciphers are submitted to NIST and some of them are selected to round 2 of this process. This selection criterion depends on resistance to attacks, performance, and cost [33]. The application-specific integrated circuit (ASIC) benchmark of these candidates on the round 2 shows TinyJAMBU has the lowest area and this comparison can be seen at the Figure 2.3 [34].

As a result of this review, the ciphers; SIMON, SPECK, PRINTcipher, LRBC, and TinyJAMBU were selected to implement as PRNG in Python language.

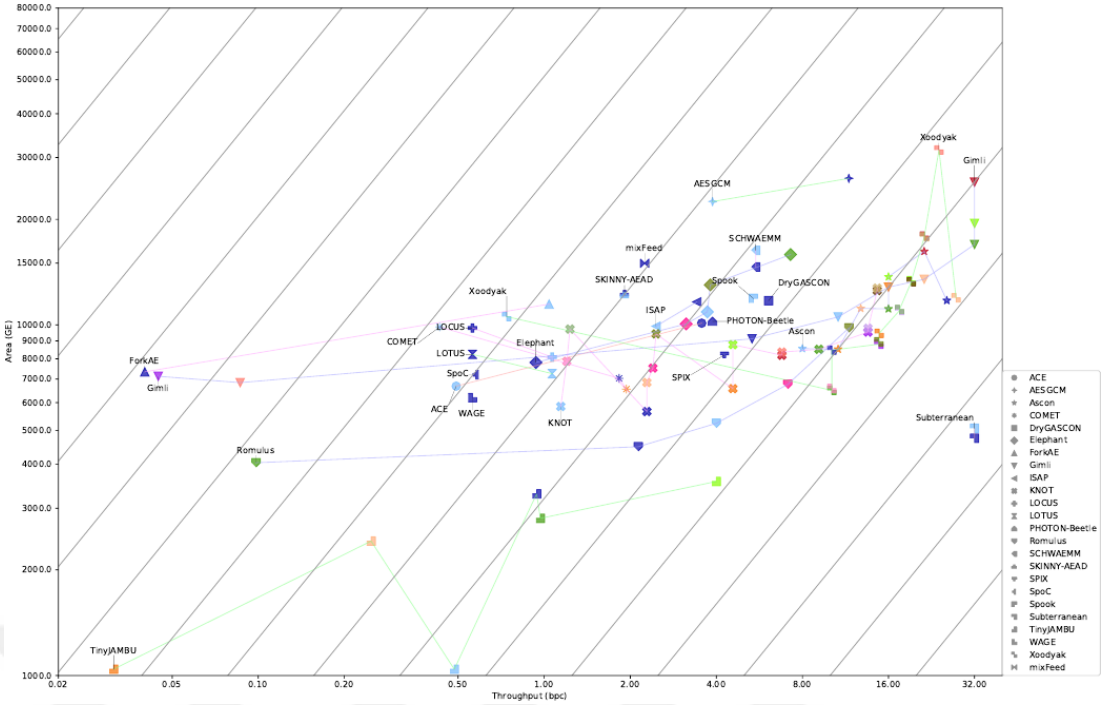


Figure 2.3 : Result of the implementation of the ciphers.

2.3 Generated PRNGs with Selected Lightweight Ciphers

The five pseudo-random number generators (PRGNs) were implemented with the selected lightweight ciphers in the Python language in order to test the randomness of produced numbers. Following the utilization of these PRNGs, the random numbers generated at each iteration of the PRNG operation in cipher block chaining (CBC) mode are assembled into individual files containing binary representations (consisting of ones and zeros) for each PRNG used.

2.3.1 Testing randomness of random numbers

There are several methods to determine the randomness of given numbers. In this thesis, NIST test is used to check produced numbers whether they are random or not [35]. It is a test suite with 15 different tests and it tests the given numbers statistically. These 15 tests are the frequency (monobit) test, block frequency test, runs test, longest run test, rank test, FFT test, nonoverlapping template test, overlapping template test,

universal test, linear complexity test, serial test, approximate entropy test, cumulative sums test, random excursions test, and random excursions variant test.

While these tests run, larger sequences are divided into small equal sections. All tests run in this section individually and then generate total results for sequence. If a test passes in all sections, it can be accepted as the test passed for all sequences.

2.3.2 SIMON

The SIMON cipher is a block cipher that utilizes the Feistel network [30]. It is specifically designed for hardware implementations.

The SIMON cipher uses the CBC mode to generate a pseudo-random number generator (PRNG) for producing random numbers. Furthermore, a SIMON-based PRNG has been implemented in Python, and NIST tests have been applied to the generated random numbers to assess their randomness. The results of the NIST test indicate that the SIMON-based PRNG is proficient in producing random numbers.

2.3.3 SPECK

SPECK belongs to the same family as SIMON but is designed as specialized software rather than hardware [30]. Despite being specialized in software, it is suitable for hardware implementation.

The SPECK cipher utilizes the CBC mode, same as SIMON, to create a PRNG for generating random numbers. The PRNG, implemented in Python, has successfully undergone NIST testing, confirming its ability to produce random numbers.

2.3.4 PRINTcipher

The PRINTcipher is a type of block cipher suitable for hardware implementation [36]. It uses a substitution-permutation network (SPN) and comes in two block width options: 48 or 96. When using a 48 block width, the key width is 80 and there are 48 rounds. With a 96 block width, the key width is 160 and there are 96 rounds.

The PRINTcipher uses CBC mode to generate a pseudo-random number generator (PRNG) implemented in Python. NIST testing has confirmed that the PRINTcipher-based PRNG is capable of producing random numbers.

2.3.5 LRBC

LRBC is a type of block cipher that uses SPN and Feistel [21]. These ciphers consist of 24 rounds. The key width and the plain text width is 16.

LRBC has implemented CBC mode in order to generate PRNG for its operations. The LRBC-based PRNG is developed using Python and has undergone NIST testing to assess the randomness of the generated numbers. The results from the NIST testing indicate that the LRBC-based PRNG is not capable of producing sufficiently random numbers.

2.3.6 TinyJAMBU

TinyJAMBU is a version of the JAMBU cipher which is a block cipher [37]. TinyJAMBU utilizes CBC mode to create a PRNG for generating random numbers. The PRNG based on TinyJAMBU is implemented in Python, and a NIST test has been conducted to assess the randomness of the generated numbers. According to the NIST test results, the TinyJAMBU-based PRNG is not capable of producing random numbers.

2.4 Comparison of PRNG's Based on Lightweight Ciphers

Two metrics have been established to compare cipher-based pseudo-random number generators (PRNGs). The first metric is the result of the NIST test, and the second is the runtime of each PRNG when producing random numbers of the same length. The NIST test result is the most critical factor, as a failure means the PRNG unusable for secure digital design. The runtime of Python refers to the time taken for the software to run on the computer, providing insight into the number of operations involved in the PRNG when generating random numbers.

Each PRNG produced different length of random number in a round. For comparing these random numbers, total length of the produced numbers from each PRNG should be same. Therefore, common multiple of length of produced random numbers for total length is determined. This value, denoted as 6291456, is chosen as the total bit length for each PRNG to produce random numbers. A numerically high value is selected to ensure significant input for the NIST test. Subsequently, each PRNG is executed 100 times and the results are individually assessed. The total time taken to produce 100 random numbers is considered as the runtime for each PRNG.

The Table 2.5 shows the results of NIST tests and runtimes for each cipher-based PRNG. According to this table, SIMON is the best both for NIST results and runtime. SPECK which is in the same family as SIMON follows SIMON in second place. PRINTcipher is in third place for NIST test results but the runtime of this cipher is lower than SIMON and SPECK. LRBC and TinyJAMBU failed in the NIST test, therefore they are in the last place.

Table 2.5 : Results of 100 different random numbers and each of them 6.291.456-bit length vector.

Cipher	PT-Block	Key-Block	Runtime	Failure Rate
SIMON	128	128	42 min	6%
SPECK	64	64	47 min	10%
PRINTcipher	48	48	463 min	11%
LRBC	16	16	155 min	100%
TinyJAMBU	32	128	1000 min	100 %

3. DESIGN AND IMPLEMENTATION OF THE ACTIVE SHIELD

In the process of the active shield digital design, the system is designed and implemented by considering several requirements. These requirements are; system should be secure, system should be scaleable, system should be lightweight, system should be reusable.

The definition of a secure system is resistance against microprobing attacks. Therefore PRNG is generated using SIMON which is a lightweight cipher. Random numbers are produced to operate the shield system by using SIMON-based PRNG.

The definition of a scaleable system is resizing the shield digital design without any effort. A scalable system makes the shield design ready to produce. The shield digital design is scaleable according to SIMON parameters and the desired line width of the shield maze structure.

To provide a lightweight system, a lightweight cipher was selected to generate PRNG which can produce random numbers.

The definition of a reusable system is the usability of the IC after an attack. A masking structure was added to the system to provide reusability and all the attacks were saved in the memory.

The system architecture is designed to realize all the requirements. This system architecture explains the subblocks and the interactions between these subblocks. These subblocks are the submodules in the architecture and realized in Verilog HDL language: Shield control, PRNG, shield bus select, shield driver, shield receiver, comparator, register. Top module is the shield in the Verilog HDL language. All submodules is inside of the top module. Connections between submodules are made in the top module.

After designing the system architecture, it is ensured that the system-dependent parameters scale the shield digital design. Therefore design parameters are determined.

A communication interface is included in the system to connect IC to the world. SPI communication interface is used for the communication. SPI is not designed in this thesis. As a verified IP, a selected SPI is utilized to achieve communication [4].

After the architecture design process, the system realized on FPGA by using the SystemVerilog HDL language. Input-output ports of FPGA are implemented to simulate the shield maze structure. Microprobing attack is simulated by breaking cables at random times on the input-output ports of the FPGA.

3.1 Scalebility and Parameters

The most important feature of the active shield digital design in this thesis is scaleability. The active shield digital design can be simulated and synthesized in various sizes. The scaleability feature is applied by using the parameters explained below. These parameters have the desired values with the certain limitations.

`SIMON_BLOCK_SIZE` and `SIMON_KEY_SIZE` are from SIMON cipher [30]. `SIMON_BLOCK_SIZE`, is the block size of SIMON cipher and `SIMON_KEY_SIZE` is the key size of SIMON cipher. These two values have multiple combinations. All combinations are shown at the Table 3.1.

`BUS_WIDTH` is the width of the drive bus for the shield maze for one cycle. This value should be lower than `SIMON_BLOCK_SIZE`. `NUMBER_OF_BUSES` is the bus number of the shield maze. The product of `BUS_WIDTH` and `NUMBER_OF_BUSES` equals to the total line in the maze structure. If the last bus has a lower number of bus width from `BUS_WIDTH` the following procedure is applied. In that procedure, all buses transmit their values normally except the last bus. The last bus has the lower number of width, extra values from produced random numbers are tranferred to comparator directly for comparing purposes. This directly tranferred values and values from the receiver are concatenated before comparison. This procedure provides that the width value of shield maze can be of non power of two.

Table 3.1 : All possible values of the parameters.

Parameters	**Values**									
SIMON_BLOCK_SIZE	32	48	48	64	64	96	96	128	128	128
SIMON_KEY_SIZE	64	72	96	96	128	96	144	128	192	256
BUS_WIDTH	1 to SIMON_BLOCK_SIZE									
NUMBER_OF_BUSES	1 to ∞									

3.2 Architecture of The Digital Design of The Active Shield

The digital design of the shield consists of seven sub-blocks. These blocks are shield control, PRNG, shield bus select, shield driver, shield receiver, comparator, register, and SPI. Shield control is the main control block of shield digital design, and handles the datapath and state machine of shield digital design. PRNG produces random numbers for maze structure and it uses SIMON cipher to produce random numbers. Shield bus select is responsible for selecting the current bus in the maze structure according to the bus number and the order. The shield driver is responsible for sending random numbers to the maze structure with a determined frequency. The shield receiver takes random numbers from the maze structure. The comparator is responsible for comparing random numbers before the maze structure and after the maze structure, and it triggers the alarm if they are not the same values. Register saves the initial vector, key, random numbers, alarm position, status, and configuration information.

There is an SPI interface to communicate with the outside world. SPI is an interface between shield digital design and a computer to send key and initial vector to shield digital design and read information from shield digital design. All blocks except SPI, designed and implemented in this thesis.

There are two clock domains in the shield digital design. One of them is the SPI clock and the other is the system clock. Three registers are used for solving the clock domain crossing issue in the design.

The Figure 3.1 shows all subblocks of shield digital design. The grey subblocks are designed in this thesis. SPI is not designed in this thesis but it is also used to create the shield digital design.

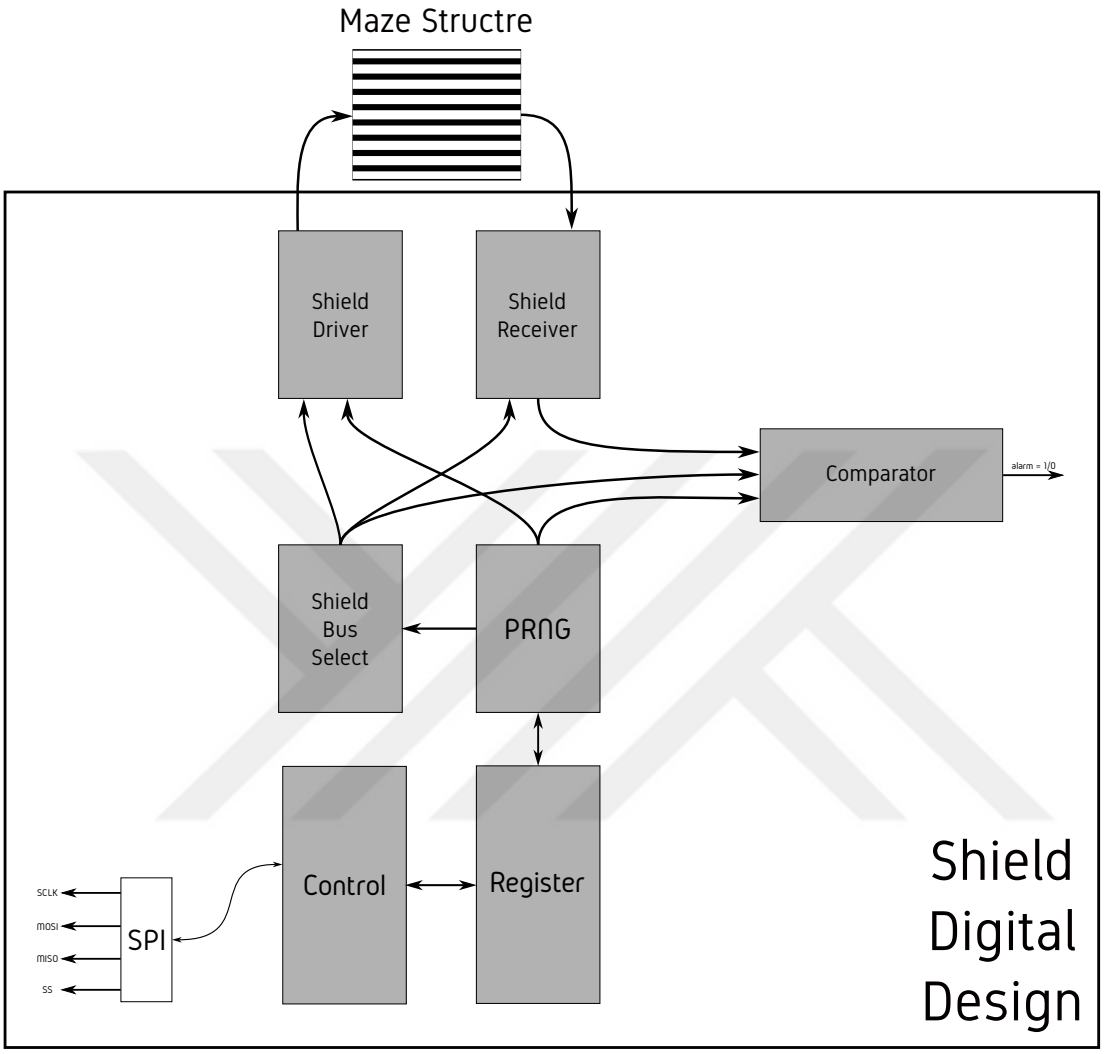


Figure 3.1 : Architecture of the shield digital design with maze.

3.2.1 Shield control

The shield digital design is controlled by control sub-block. After loading IV and the key to register by using control, the run command is sent via SPI to the sixth bit of the configuration register. After receiving the run command, PRNG starts. Until an attack occurs, PRNG works constantly.

The finite state machine (FSM) in the Figure 3.2 manages the flow of data between SPI and the register. The FSM has three states and these states show operation mode, every state corresponds to an operation mode in the section 3.3.

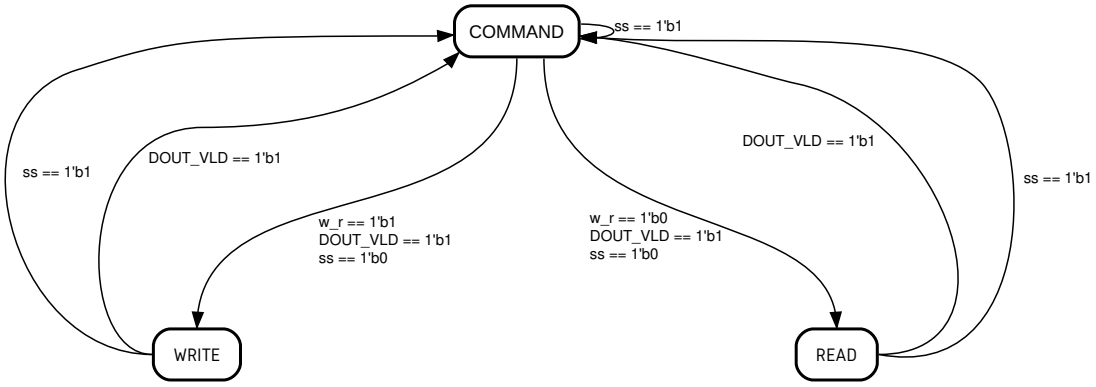


Figure 3.2 : FSM of control in shield digital design.

3.2.2 PRNG

PRNG design consists of one SIMON cipher and FSM to run this SIMON as PRNG. The Figure 3.3 shows that SIMON gets the values of key and IV from the register file in the first round of PRNG. In the following rounds, the same key was operated but produced a random number from the previous round used as plaintext of SIMON. PRNG is started by the sixth bit of the configuration register. If the alarm is triggered, the alarm stops the operation of the PRNG.

3.2.3 Shield bus select

The maze structure of the shield consists of multiple metal lines. Every line can be considered as a one-bit width wire. The width of the maze structure of the shield can change from an IC to an IC. In this thesis, because of this variable width the maze structure of the shield is divided into buses when it is operating. The methodology of the division is parametric and controlled by `NUMBER_OF_BUSES` and `BUS_WIDTH` parameters which are explained in the section 3.1. In the maze structure of the shield, not all buses operate at the same time. When sending random numbers to the maze structure of the shield, only one bus is actively used. Other buses are inactive at that

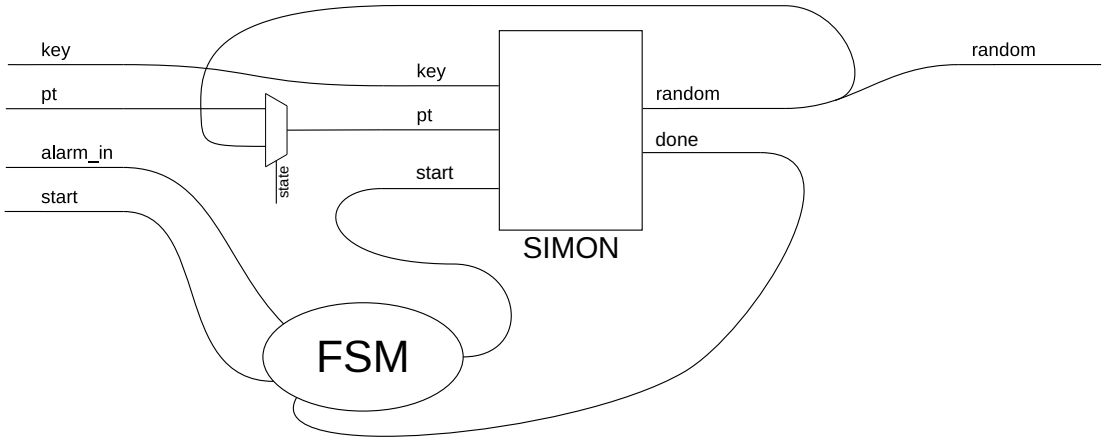


Figure 3.3 : PRNG with SIMON.

time. Shield bus select is the block that determines which bus is actively in use. The duration of the changing buses is predetermined and this duration should be lower than human operator speed using the micro probing device. The Figure 3.4 shows shield bus select diagram and *m* stands for `NUMBER_OF_BUSES`.

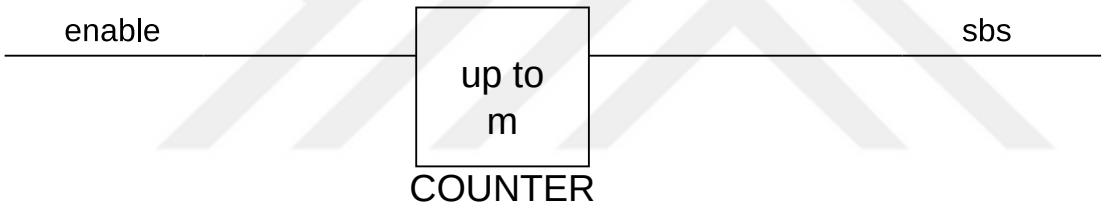


Figure 3.4 : Shield bus select diagram.

3.2.4 Shield driver

The shield driver sends produced random numbers into the PRNG to the maze structure of the shield according to the bus information that comes from the shield bus select. The Figure 3.5 shows the shield driver diagram, *n* stands for `BUS_WIDTH` and *m* stands for `NUMBER_OF_BUSES`.

3.2.5 Shield receiver

The shield receiver gets random numbers from the maze structure of the shield according to the bus information coming from the shield bus select. After receiving the

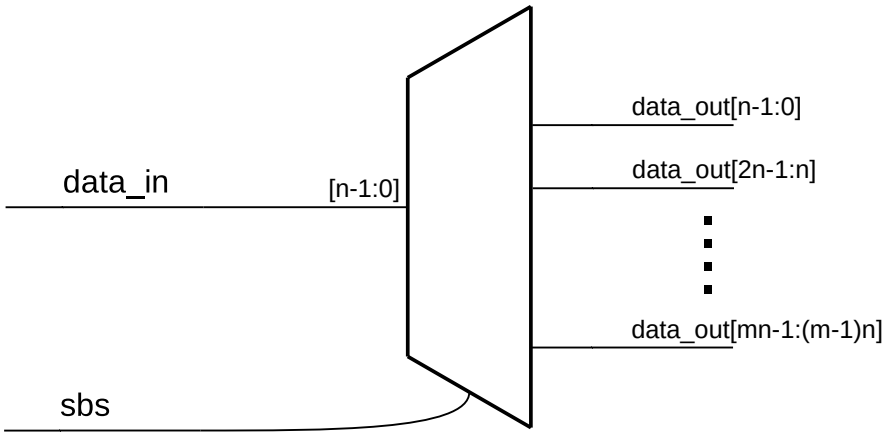


Figure 3.5 : Shield driver diagram.

numbers, the numbers are delivered to the comparator. The Figure 3.6 shows shield receiver diagram, *n* stands for `BUS_WIDTH` and *m* stands for `NUMBER_OF_BUSES`.

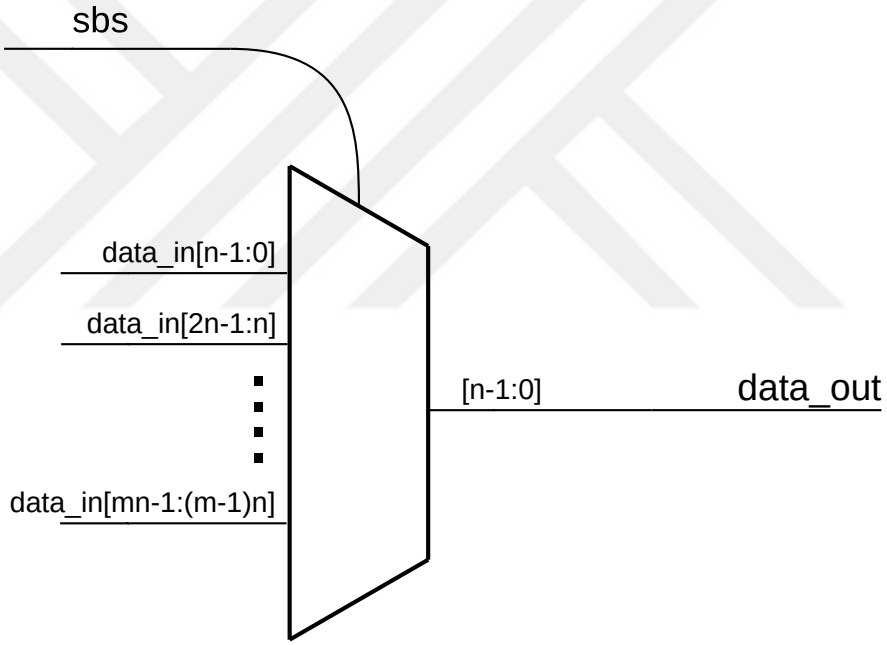


Figure 3.6 : Shield receiver diagram.

3.2.6 Comparator

The comparator is responsible for comparing both sides of the maze structure of the shield. The input side of the maze structure of the shield gets the numbers from PRNG and these numbers are also conveyed to the comparator. The output side of the maze structure of the shield is handled by the shield receiver and received numbers are

conveyed to the comparator. These two numbers are compared in this block and if is there any mismatch triggers the alarm. The Figure 3.7 shows comparator sub-block including masking and alarm read operation.

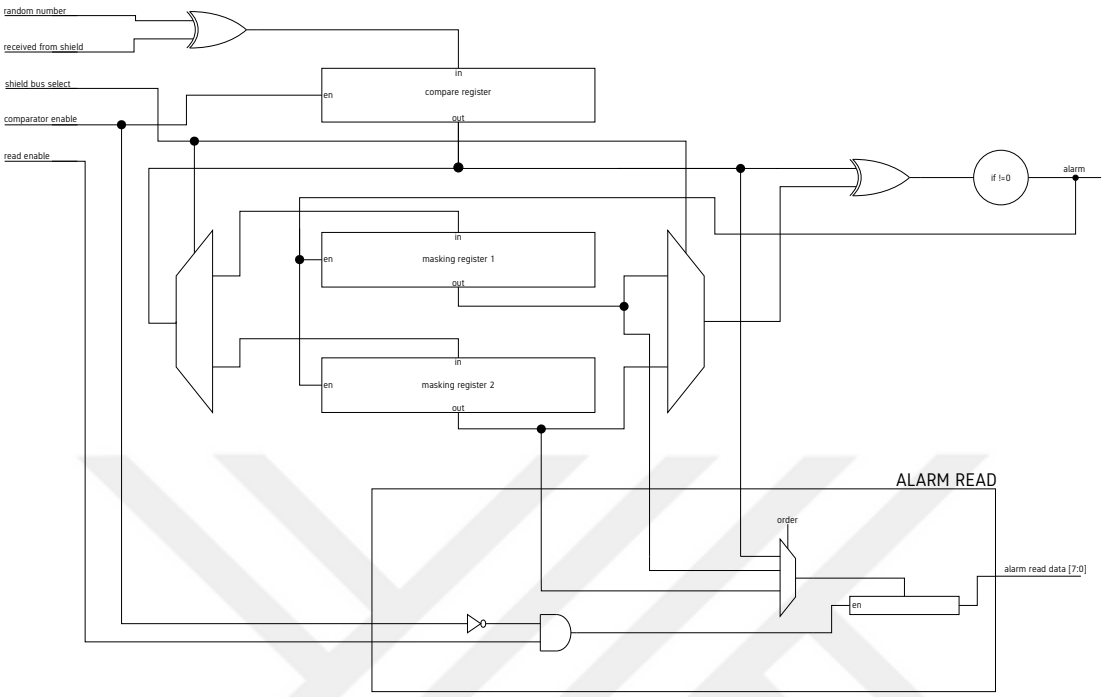


Figure 3.7 : Comparator of the shield digital design.

3.2.6.1 Masking previous attacks

In the digital design of the shield, the expected produced IC should be used multiple times after an attack. Therefore, a masking system is designed to save previous attack information. This masking system provides reusability for the IC of active shield after attacks.

After an attack, it is expected that at least one wire on the maze structure of the shield would become unconnected anymore. The information on the position of unconnected one wire or multiple wires needs to be saved in the registers. For that purpose, this information is recorded in the comparator in alarm read registers.

At the beginning of the operation, the comparator register and masking registers are empty, since there is not any previous attacks. After every random number generation, a corresponding comparing operation is executed. When a random number

is produced, comparator_enable signal is set to 1. This enables providing comparison of numbers from PRNG and maze structure of shield.

After every attack, the line becomes broken permanently. This broken line cannot trigger an alarm since it is masked. Therefore two layers of comparison operation is executed. The first random numbers from PRNG and the maze structure of the shield are compared and the result is saved into the compare register. After that, the compare register and corresponding masking register are compared according to shield bus selection. The result triggers an alarm if there is a mismatch.

After an attack, the first comparator_enable signal initiates a comparison that result as an alarm. After one clock cycle, information on the broken line in the maze structure of the shield is saved into the corresponding masking register according to the shield bus select value. The Figure 3.7 shows masking registers and the relationship between compare registers. The width of the compare register and the masking registers is BUS_WIDTH. The number of the masking registers is NUMBER_OF_BUSES.

3.2.6.2 Alarm read

When an attack occurs on the maze structure of the shield, it is expected to fault in the operation on at least one line of the maze structure of the shield. It is important to know the position of this line in the maze structure of the shield. Therefore, there is a register to save those positions and masking positions from previous attacks. The alarm read register is a special type of register that gets the position of the alarm according to the order of reading of compare register and masking register 1, masking register 2, and masking register n. The alarm read register is separated from the register file and it cannot be readable with SPI. When *alarm read* is enabled, values in the alarm register are transferred to the register file and it is read with 8 bits for every transfer.

For reading attack positions, it is read the last address of the register via SPI. When this address is read, it triggers read enable signal. This signal activates the alarm read part of the comparator.

3.2.7 Register

The register file is the main storage of the active shield digital design. The width of the register file is 8-bit and the maximum depth of register file is 128-bit. Actual dept depends on `SIMON_BLOCK_SIZE` and `SIMON_KEY_SIZE`, the Table 3.2 shows all of the actual depths of the registers. The all registers are readable and writeable via SPI. Memory map of register file can be seen in the Figure 3.8.

Table 3.2 : All base addresses (B.A.) and last addresses (L.A.) for each register area (addresses are in decimal).

SIMON		IV registers		key registers		random number registers		reserved area		alarm read
block size	key size	B.A.	L.A.	B.A.	L.A.	B.A.	L.A.	B.A.	L.A.	
32	64	2	5	6	13	14	17	18	30	31
48	72	2	7	8	16	17	22	23	30	31
48	96	2	7	8	19	20	25	26	30	31
64	96	2	9	10	21	22	29	30	30	31
64	128	2	9	10	25	26	33	34	62	63
96	96	2	13	14	25	26	37	38	62	63
96	144	2	13	14	31	32	43	44	62	63
128	128	2	17	18	33	34	49	50	62	63
128	192	2	17	18	41	42	57	58	62	63
128	256	2	17	18	49	50	65	66	126	127

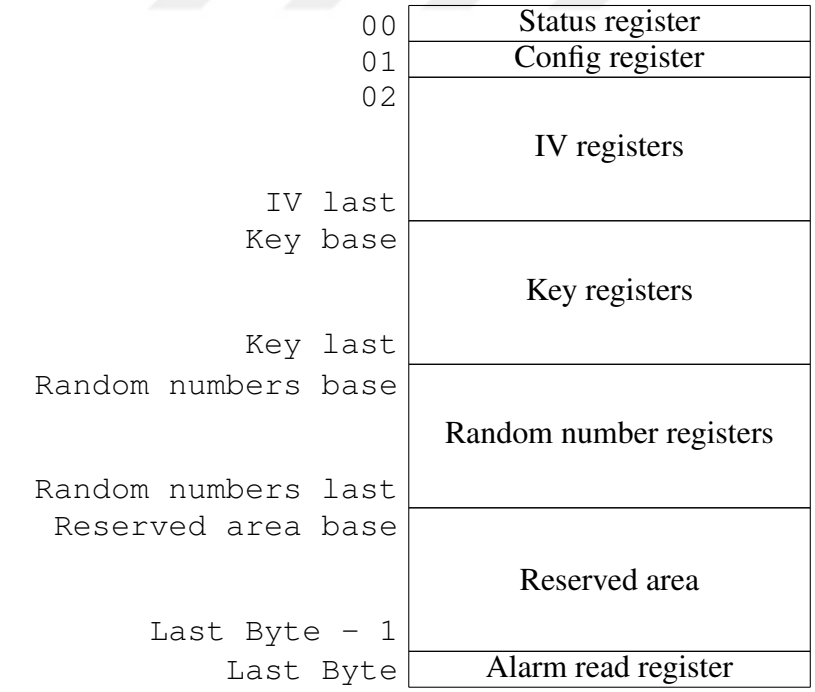


Figure 3.8 : Memory map of register.

3.2.7.1 Status register

The address of status register is 0. It keeps the information about the current status. The register field 7 of the status register keeps the alarm information of the system. The remaining fields are reserved.

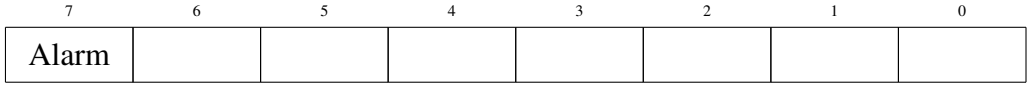


Figure 3.9 : Status register.

3.2.7.2 Configuration register

The address of the configuration register is 1. The register field 7 of the status register usign for the software reset. The register field 6 of the status register is using for starting PRNG. The remaining fields are reserved.

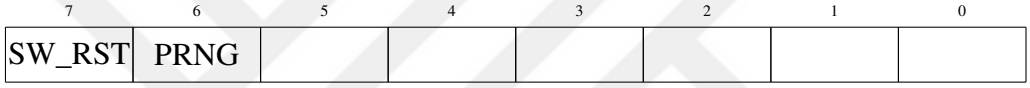


Figure 3.10 : Config register.

3.2.7.3 IV registers

The base address is 2 and the offset is `SIMON_BLOCK_SIZE/8` for the IV registers. The number of the registers in the IV registers is `SIMON_BLOCK_SIZE/8`. It keeps the IV of the SIMON cipher as a text.

3.2.7.4 Key registers

The base address is `SIMON_BLOCK_SIZE/8+2` and the offset is `SIMON_KEY_SIZE/8` for the key registers. The number of the registers in the key registers is `SIMON_KEY_SIZE/8`. It keeps the key of the SIMON cipher as a text.

3.2.7.5 Random number registers

The base address is `SIMON_BLOCK_SIZE/8+SIMON_KEY_SIZE/8+2` and the offset is `SIMON_BLOCK_SIZE/8` for the random number registers. The number of

the registers in the random number registers is `SIMON_BLOCK_SIZE/8`. It keeps the produced random numbers of SIMON cipher as a text.

3.2.7.6 Alarm read register

The last byte of the register keeps the alarm position. The reading operation of the alarm read register explained in the subsubsection 3.2.6.2. When reading this register, it triggers the alarm reading process.

3.3 Communication Interface

The SPI is the interface between the shield's digital design and the outside world. It is used for both reading and writing purposes. All registers in the register are readable and writeable via SPI.

The Table 3.3 shows the transfer modes and the corresponding received data and the response from shield digital design. There are three transfer modes for communication in the shield digital design implementation. These transfer modes are command mode, write mode, and read mode.

The transfer mode is controlled by the FSM in the control in the shield digital design; a diagram of FSM is depicted in the Figure 3.2. The first transfer is always in the command mode. The following transfer is in write mode or read mode; it is controlled by the most significant bit of the received data in the command mode. After a write mode or read mode, the next mode is always the command mode.

In the command mode, both address information and write or read information are received. The most significant bit of the received data indicates write or read information, and the least significant 7 bit indicates the address. When the most significant bit is 1, the following transfer is a write transfer. When the most significant bit is 0, the following transfer is a read transfer. In command mode, the response is always in the in the status register.

In the write mode, the received data is used for writing to registers according to the address received from the previous command mode. In the write mode, the response is always in the status register.

Table 3.3 : Transfer modes of SPI and purpose of transfers.

Transfer Mode	Received Data	Response
Command	W/R[7], Address[6:0]	Status Register
Write	Write data	Status Register
Read	Don't care	Read Data

In the read mode, the received data is not important and is not used. But the response reads data from the registers according to the address received from the previous command mode.

3.4 Operation Stages

There are four stages of operation in the shield. The first stage is the data input stage, the second stage is the random number generation stage, the third stage is the operating shield stage, and the fourth stage is the attack. These stages are set up using the configuration register. The Figure 3.11 shows the flow of the operation. The details of the operation explained below.

1. Data Input: This stage is used for the data input to the registers. The initial vector (IV) and the key which produces the random numbers are data inputs for the registers. The IV and the key are loaded via SPI. The total length of the data depends on the SIMON configuration.

2. Random Number Generating: When all the IV and the key values are collected, the random number generation can start. Asserting the sixth bit of the configuration register initiates this stage. To initiate PRNG, this register field should be asserted. After the first run, it should be deaserted. This stage can be used only to start the PRNG. After starting the PRNG, it runs constantly until an alarm is triggered.

3. Operating Shield: After a random number is generated, the operating shield starts to run. Until there is an attack, this stage and the Random Number Generating stage operate constantly. Conveying produced numbers to the maze structure of the shield and comparing the random numbers from PRNG and the maze structure of the shield take place in this stage.

4. Defense: This stage only operates when there is an attack. In this stage, an alarm is generated and the operation of PRNG is stopped. When there is an attack, the location of the attack is saved in this stage.

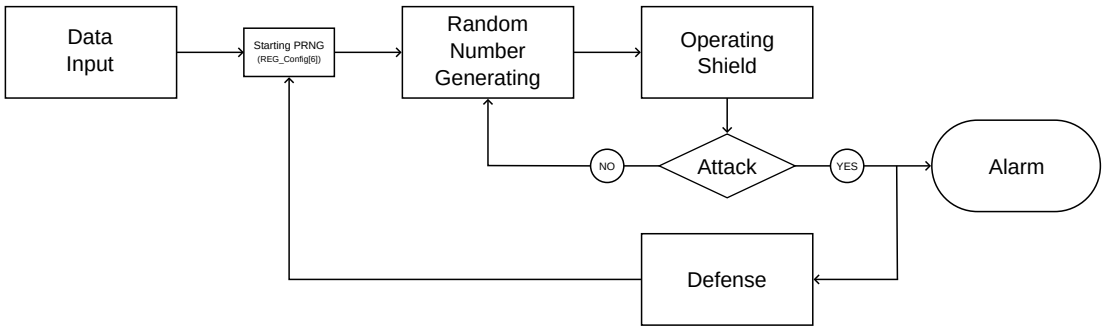


Figure 3.11 : Operation stages of the digital design of the active shield.

4. RESULTS AND CONCLUSION

4.1 Results

The active shield design is simulated in AMD, formerly known as Xilinx, Vivado 2022.2 [38]. It is implemented into the FPGA. There are three steps to verify the active shield digital design. At first, the produced random numbers are compared with random numbers which generated by the PRNG designed in the Python. Secondly, attack is simulated in the Xilin Vivado. At last, all digital design implemented in the FPGA and the attack scenario are simulated in the real world. After all steps, the IC with active shield digital design is produced with the technology of TSMC.

4.1.1 Produced random numbers in simulation

To test the correctness of the random numbers generated by the design, a comparison with random numbers generated with Python was made. A design with block size 128, key size 256, bus width 128 and number of bus 5 was simulated with AMD Vivado. A PRNG with the same block size and the key size values was also run with Python.

The Table 4.1 shows description of the names in the Figure 4.1, the Figure 4.2, the Figure 4.3, the Figure 4.4, the Figure 4.5, and the Figure 4.6. These names are the signals of the active shield digital design. A signal refers to a type of variable in HDL.

The Table 4.2 shows the produced random numbers for sample for the indices 0, 1, 2, 98, 99, 100, 101, 1125, 1126, 1127, and 1128. The first index in the generated random numbers in Python is zero.

The Figure 4.1 and the Figure 4.2 show the generated random numbers in the simulation for the indices 0, 1, and 2. The Figure 4.3 and the Figure 4.4 show the generated random numbers in the simulation for the indices 98, 99, 100, and 101. The Figure 4.5 and Figure 4.6 show the generated random numbers in the simulation for the indices 1125, 1126, 1127, and 1128.

There are eleven values corresponding to the sample eleven indices. The eleven values obtained from the generated numbers by PRNG in Python were compared to the eleven values obtained from the generated numbers by the active shield digital design simulation result. It is proven that these values were the same.

Table 4.1 : Descriptions of the names of the waveforms.

Name	Description
rst	Reset of the system.
clk	Clock of the system.
DIN[7:0]	Data input of the SPI from the testbench.
from_shield[2:0]	3-bits of the bus from the shield maze to the active shield digital design.
shield_mask[2:0]	3-bits mask of the bus between the active shield digital design and the shield maze for the attack injection purpose.
to_shield[2:0]	3-bits of the bus from the active shield digital design to the shield maze .
alarm	System alarm output.
shield_chip_to_shield [639:0]	All of the bus from the active shield digital design to the shield maze .
shield_chip_from_shield [639:0]	All bits of the bus from the shield maze to the active shield digital design.
write_pt[127:0]	Loaded value of the IV in the active shield digital design.
write_key[255:0]	Loaded value of the key in the active shield digital design.
read_ct[127:0]	Readed value of the cipher text in the active shield digital design.

Table 4.2 : Random numbers of the selected indices generated by PRNG in Python.

Index of random number	Random number values in hexadecimal format
0	7EECD851F4C5FBEF3FD15B6EB450505C
1	B3A8D9D788DCC4348CA5D747D9BB5487
2	730BAFBAA109581F6B780D300FB82879
98	0CFDB385C5921698309E9B0195CE9D4E
99	8E21948AFBFF4185100399EB3E3BD9D3
100	FB445C87BEC3D4CF0E1368839C20BA26
101	6044890771110B31F3B9DBBDF7475D7B
1125	208FD2AD34E54BE8E9BB49AD5BEF4109
1126	4E3EFE390222397797AD265ECD5CB1F6
1127	D12041EB7683FD47DCF374C2DEC3391B
1128	A5711AAA114A8A02753978BD35E49B8C

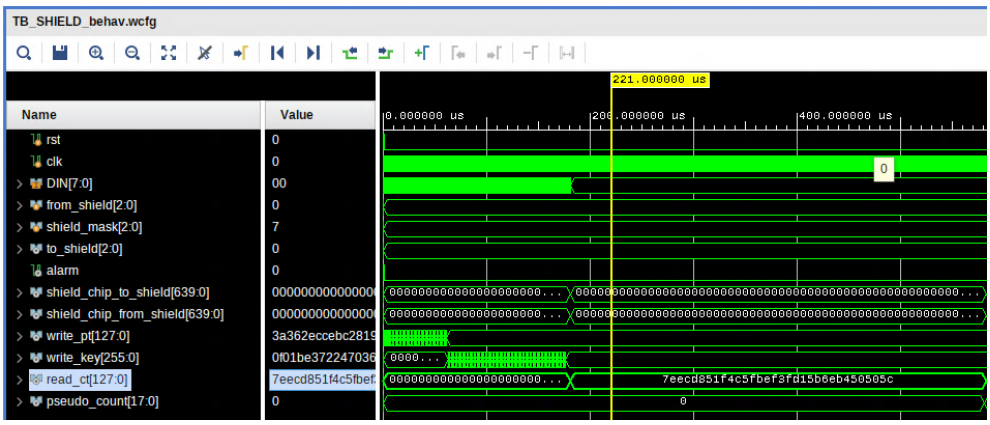


Figure 4.1 : Simulation waveform of produced random number for the index 0 in the Vivado.

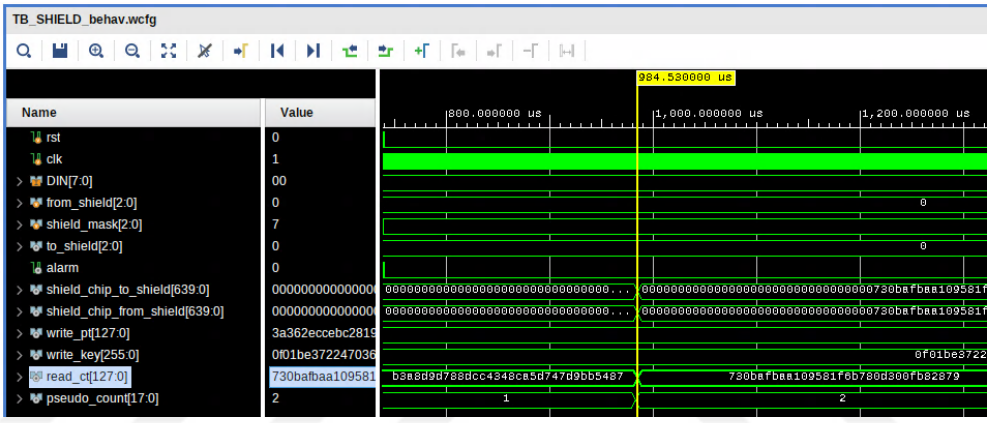


Figure 4.2 : Simulation waveform of produced random number for the indices 1, and 2 in the Vivado.

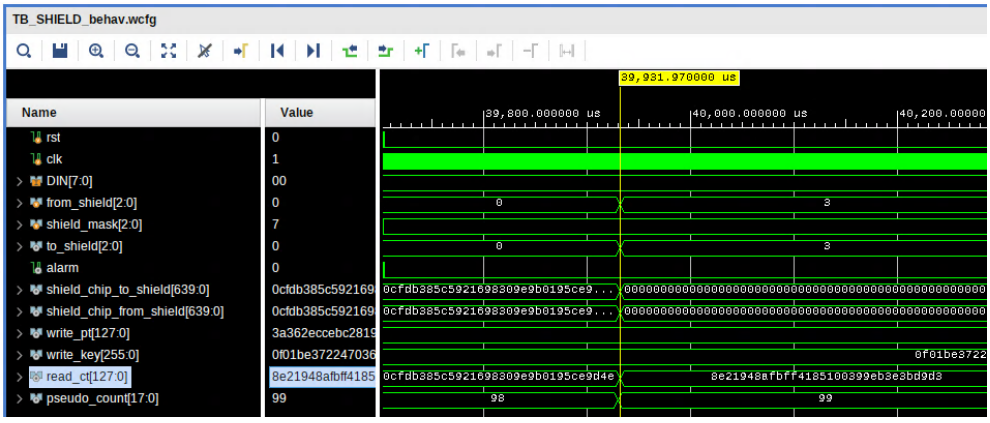


Figure 4.3 : Simulation waveform of produced random number for the indices 98, and 99 in the Vivado.

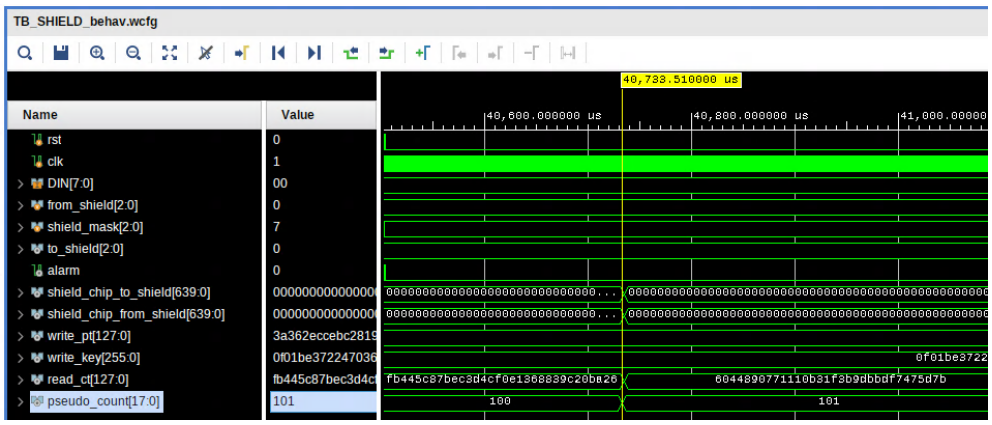


Figure 4.4 : Simulation waveform of produced random number for the indices 100, and 101 in the Vivado.

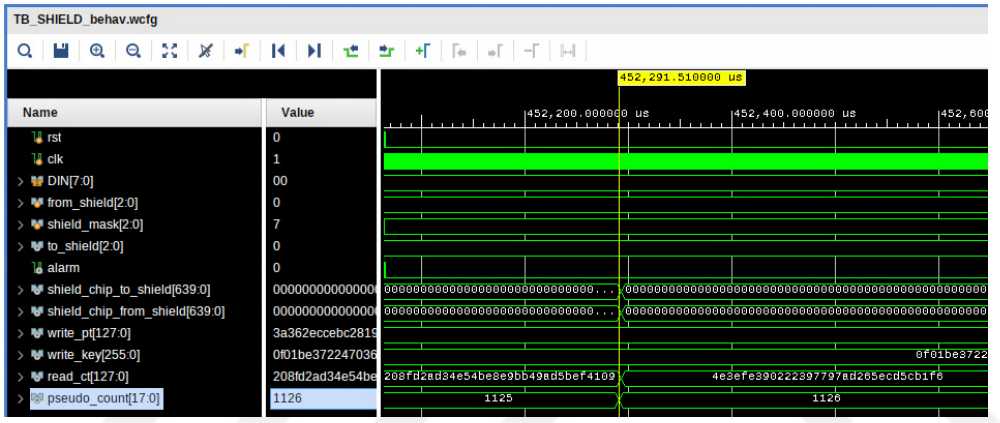


Figure 4.5 : Simulation waveform of produced random number for the indices 1125, and 1126 in the Vivado.

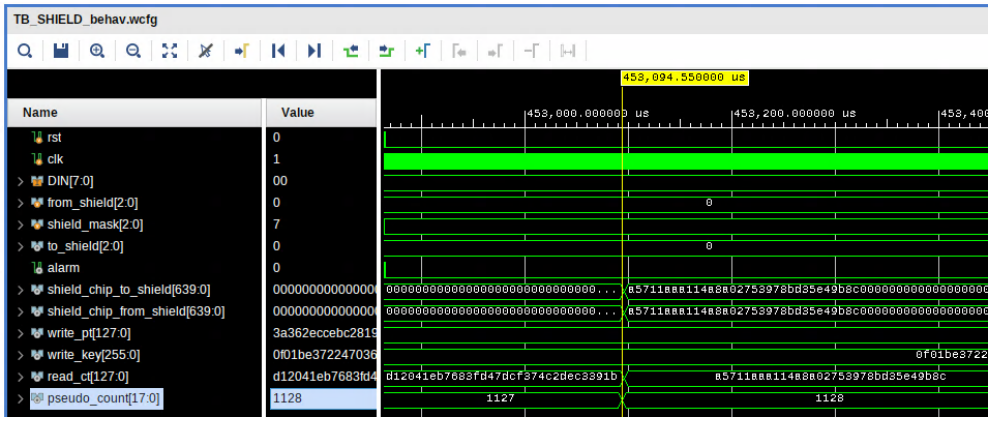


Figure 4.6 : Simulation waveform of produced random number for the indices 1127, and 1128 in the Vivado.

4.1.2 Simulation of the attack

After validating of the produced random numbers in the simulation, the attack is simulated. In order to simulate the attack, AMD Vivado has been utilized. The Figure 4.7 and the Figure 4.8 shows the scenario. The scenario is started with the loading of the IV and the key to the design via the SPI in the test bench. When the time is `5,1837ms`, the first attack on the active shield digital design is executed. Since the `NUMBER_OF_BUSES` is 5, detecting the first attack is delayed. Following a brief period of comparative analysis, the shield comparator identified the attack at `8,61295ms`. After detection of the attack, at the `8,61295ms`, the shield comparator triggers the alarm.

At the `15,18910ms`, the start command is send via SPI. After the start command, the system continues to work normally until the second attack. At the `30,19110ms`, the second attack applied. Following a brief period of comparative analysis, the shield comparator identified the second attack at `35,26735ms`. After detection, at the `35,26735ms` the shield comparator triggers the alarm. After certain period time, at the time `50,19110ms`, the second start command is send via SPI. After the start command, system continues to work normally.

This simulation proves that the active shield digital design is working as expected. It can detect the attack, after an attack it can continue to work as before. Previous attacks is saved in the registers and they are masked to not trigger an alarm again.

4.1.3 FPGA iplementation

The active shield digital design is implemented into AMD Artix 7 FPGA (part: xc7a100tcsg324-1) [39]. The implemented design is embedded into the FPGA and the shield maze structure imitated by the jumper cables that are connected to the IO ports of the FPGA. An Arduino Mega is used as the SPI master for sending values to the FPGA [40]. The designed system is set as the Figure 4.9 and the Figure 4.10. The Figure 4.10 depicts the FPGA working.

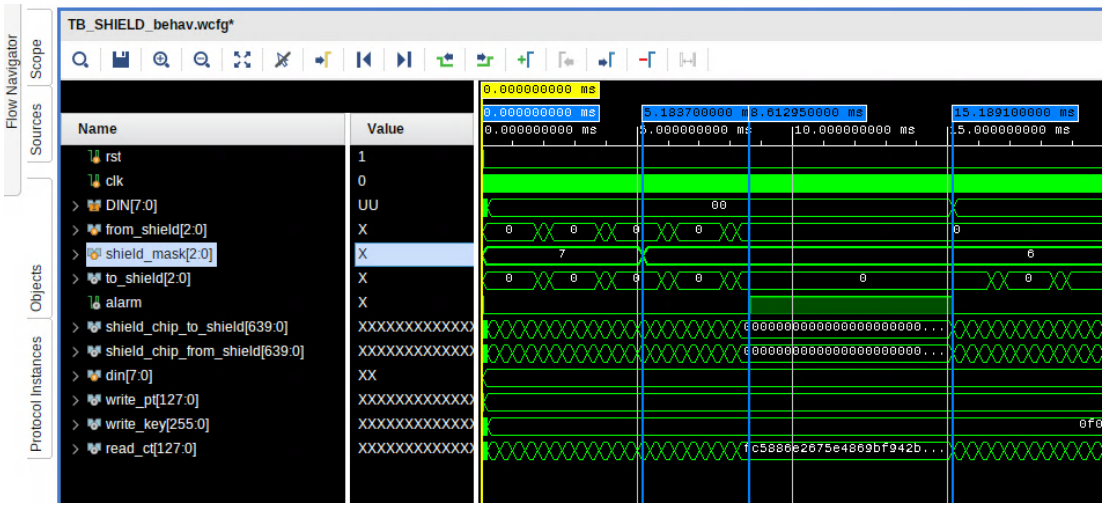


Figure 4.7 : Simulation waveform of attack in the Vivado, first half of the waveform.

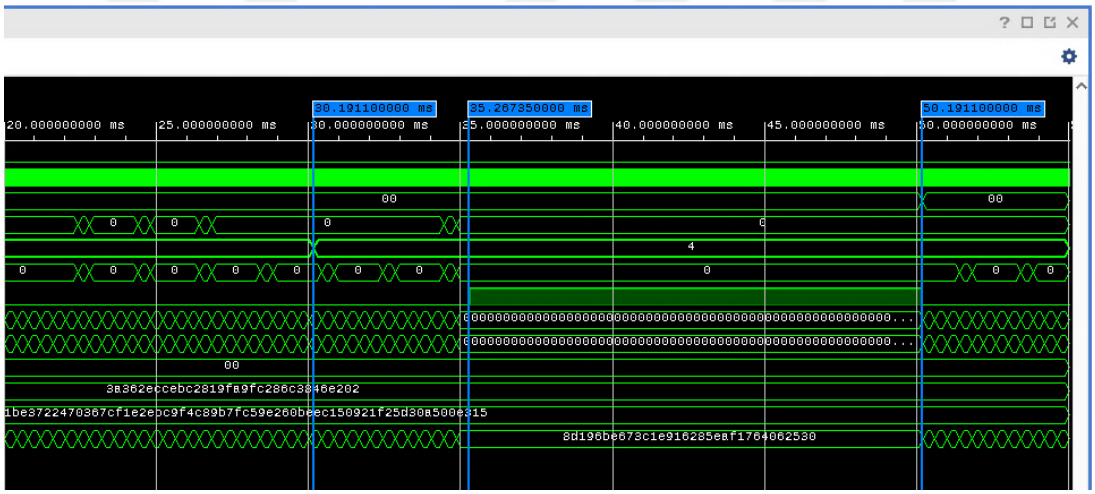


Figure 4.8 : Simulation waveform of attack in the Vivado, second half of the waveform.

The system started, and values were loaded via SPI using Arduino. While the initial value and the key are loading, the blue light in the FPGA is activated. The 4.10(a) shows the loading phase of the system.

After everything is loaded, the system runs in normal mode. While everything is running normally, FPGA led turns to green. The 4.10(b) shows the running of the system.

The attack in the FPGA implementation, it is imitated by removing one of the jumper cables. While system working, when the attack is happened the FPGA led turns red. The 4.10(c) shows the the attack into system. Removed jumber cable can be seen at the top left corner of the 4.10(c).

This test shows implementation of the active shield digital design in an FPGA is working as expected. System is running and when there is an attack system triggers the alarm.

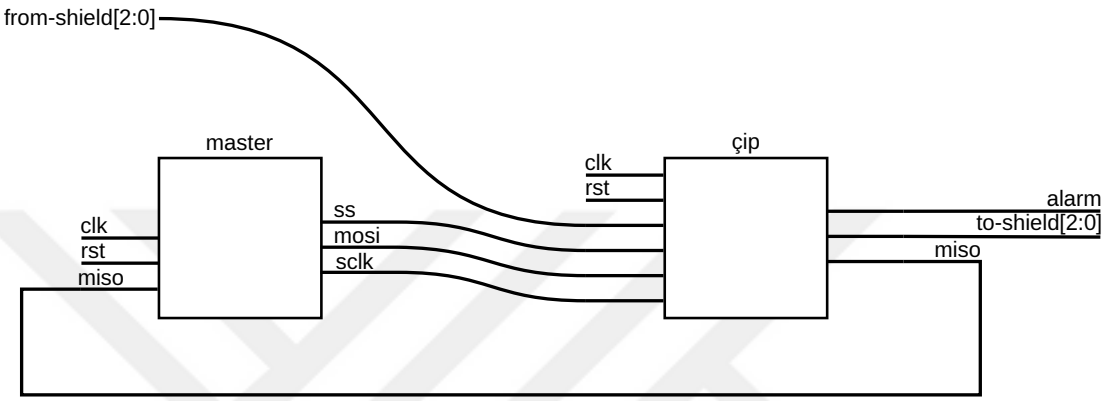
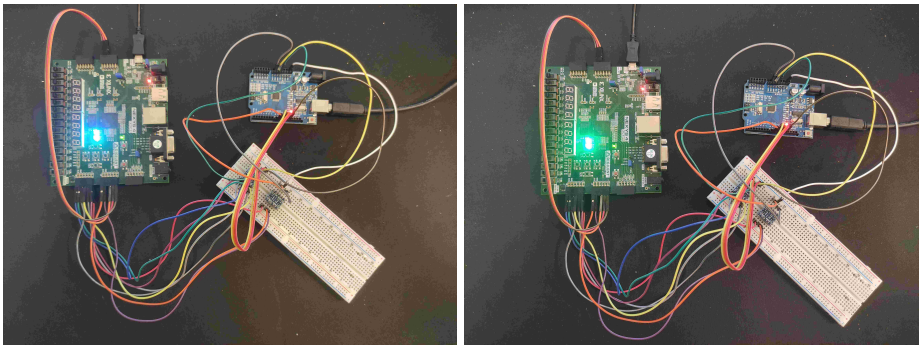


Figure 4.9 : Model for testing, master is Arduino.

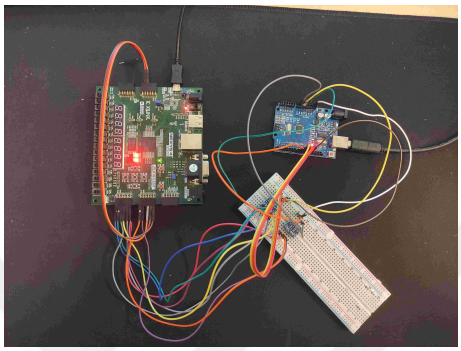
4.1.4 Utilization in the FPGA

Understanding the scaleability of shild digital designs, every configuration of SIMON synthesized in Vivado and the Table 4.3 shows the number of LUTs and number of register values for every possible SIMON configuration. SIMON configuration is the combination of possible `SIMON_BLOCK_SIZE` and `SIMON_KEY_SIZE`. All values and combinations can be seen in the Table 3.1.

The Figure 4.12, the Figure 4.13, the Figure 4.14, the Figure 4.15, the Figure 4.16, the Figure 4.17, the Figure 4.18, the Figure 4.19, the Figure 4.20, and the Figure 4.21 shows the relationship between `NUMBER_OF_BUSES` and LUT and register number for each combination of SIMON parameters. In these figures, `BUS_WIDTH` is equal to `SIMON_BLOCK_SIZE`. These figures show that there is a direct relationship between number of register and `NUMBER_OF_BUSES`. There is a relationship between number of LUT and `NUMBER_OF_BUSES` but this relationship is not strictly linear.



(a) Loading the IV and the key. (b) System running normally.



(c) Attacked to the system.

Figure 4.10 : FPGA implementation running phases.

The Figure 4.11 shows the relationship between the combinations of SIMON parameters and LUT and register number. This figure shows that the most important parameter effects on count of LUT and register is `SIMON_BLOCK_SIZE`. `SIMON_KEY_SIZE` has also effect but compared to `SIMON_BLOCK_SIZE` it is minimal.

4.1.5 Production of the IC

At the end, GDSII was generated from the active shield digital design in Cadence Virtuoso [41]. This GDSII file includes the maze structure of shield and shield digital design. The Figure 4.22 shows generated GDSII file in Cadence Virtuoso.

After generation of GDSII file, the active shield digital design producted in TSMC. The Figure 4.23 shows producted IC's. The Figure 4.24 shows the microscope screen which shows produced IC with active shield digital design.

Table 4.3 : LUT and register numbers of synthesised design with number of bus equals to 1, 16, 64, and 128.

# of Bus	SIMON Block Size	SIMON Key Size	# of LUT	# of Registers
1	32	64	823	1686
1	48	72	917	1878
1	48	96	941	1926
1	64	96	1023	2103
1	64	128	1166	2423
1	96	96	1329	2711
1	96	144	1351	2807
1	128	128	1543	3129
1	128	192	1575	3257
1	128	256	1854	3898
16	32	64	1559	2171
16	48	72	1990	2607
16	48	96	2014	2655
16	64	96	2323	3074
16	64	128	2466	3394
16	96	96	3080	4165
16	96	144	3102	4261
16	128	128	3746	5066
16	128	192	3777	5194
16	128	256	4047	5835
64	32	64	3423	3720
64	48	72	4580	4929
64	48	96	4578	4977
64	64	96	7645	6168
64	64	128	7812	6488
64	96	96	10480	8802
64	96	144	10503	8899
64	128	128	10343	11248
64	128	192	10376	11376
64	128	256	10566	12017
128	32	64	7551	5783
128	48	72	9524	8019
128	48	96	9548	8067
128	64	96	12322	10290
128	64	128	12486	10610
128	96	96	18091	14986
128	96	144	18114	15083
128	128	128	21975	19493
128	128	192	22008	19621
128	128	256	22199	20260

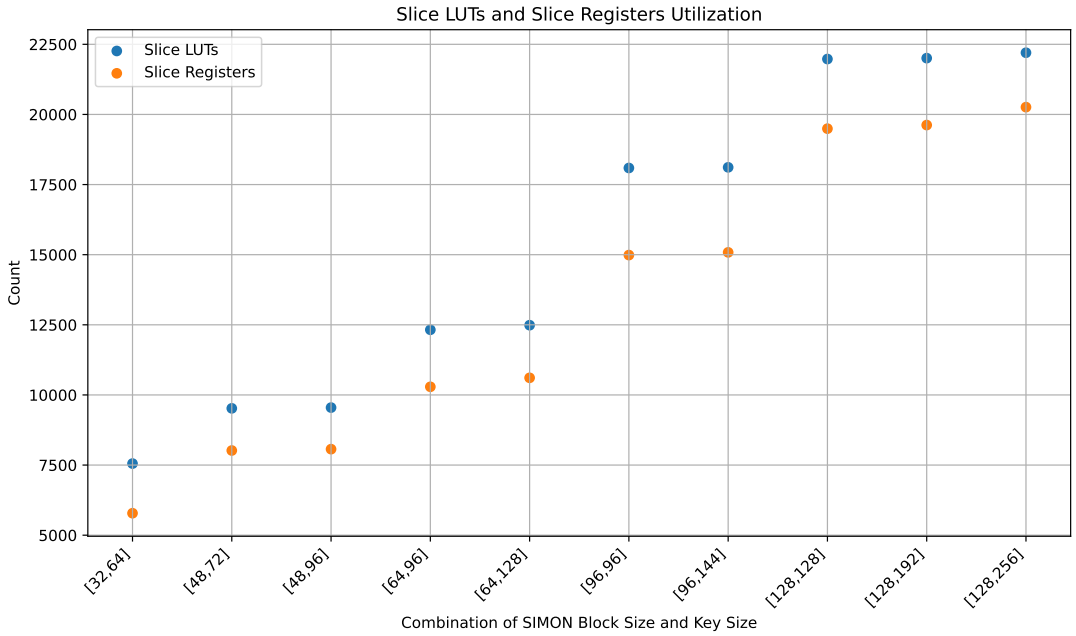


Figure 4.11 : Number of register and LUT changed by the SIMON block size and key size.

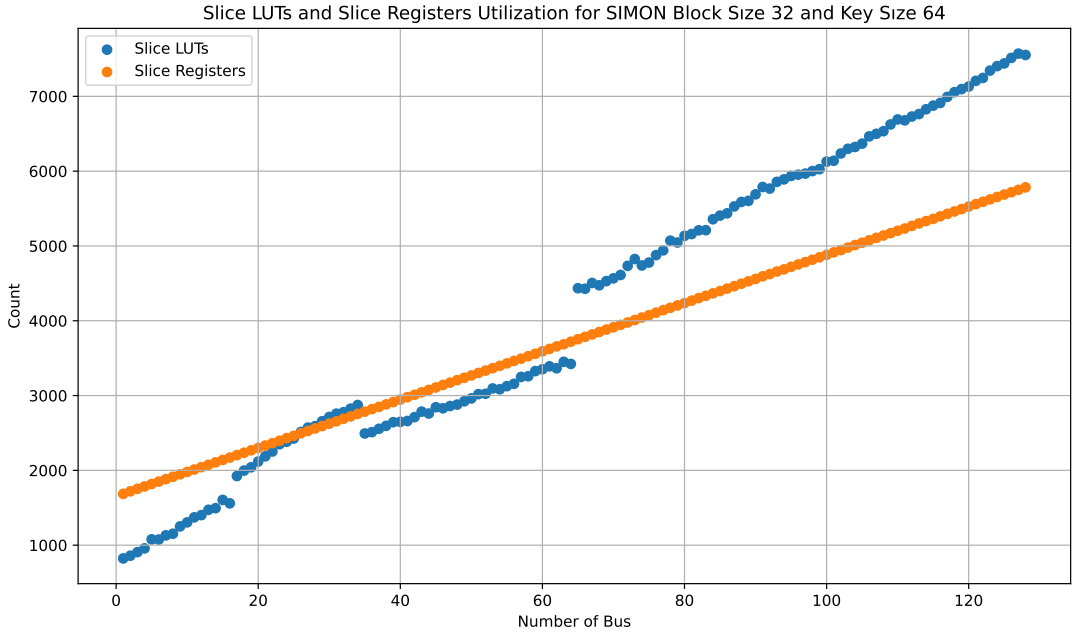


Figure 4.12 : Number of register and LUT changed by the number of buses, SIMON block size is 32 and key size is 64.

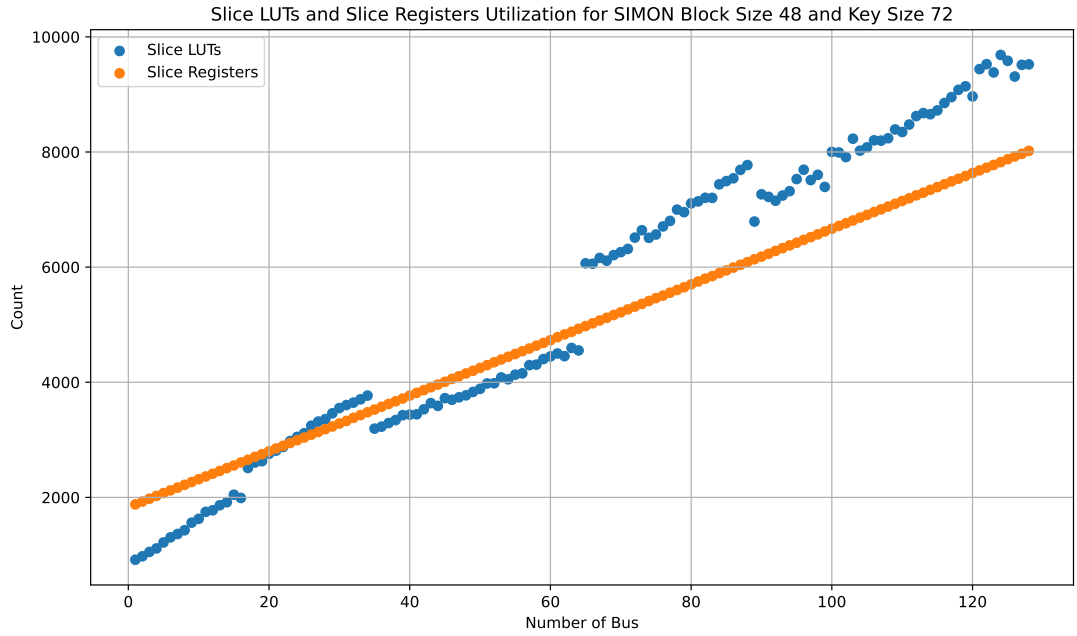


Figure 4.13 : Number of register and LUT changed by the number of buses, SIMON block size is 48 and key size is 72.

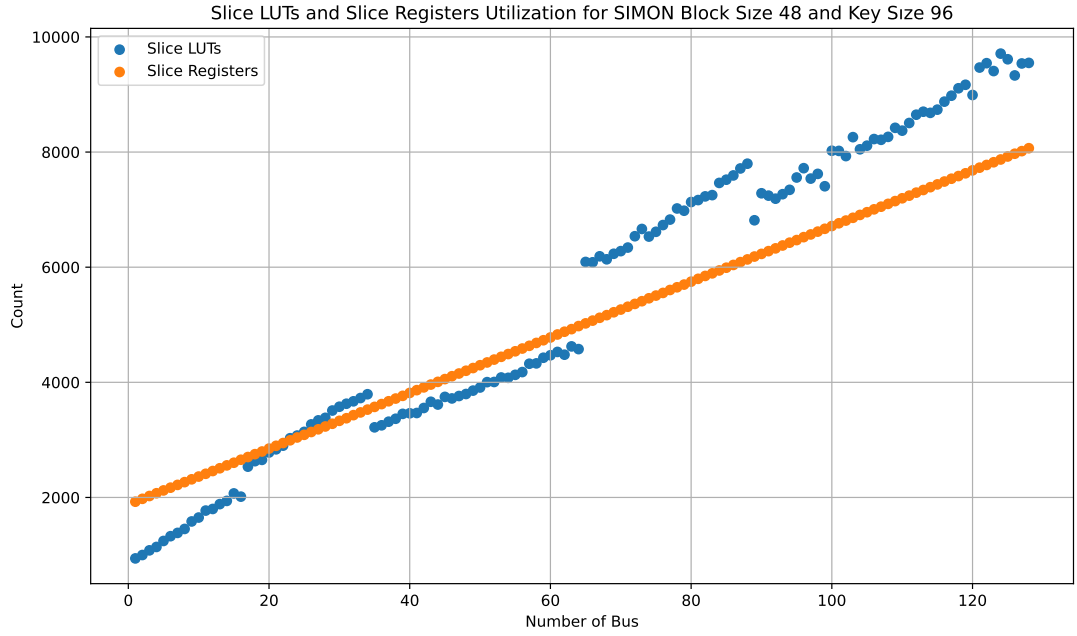


Figure 4.14 : Number of register and LUT changed by the number of buses, SIMON block size is 48 and key size is 96.

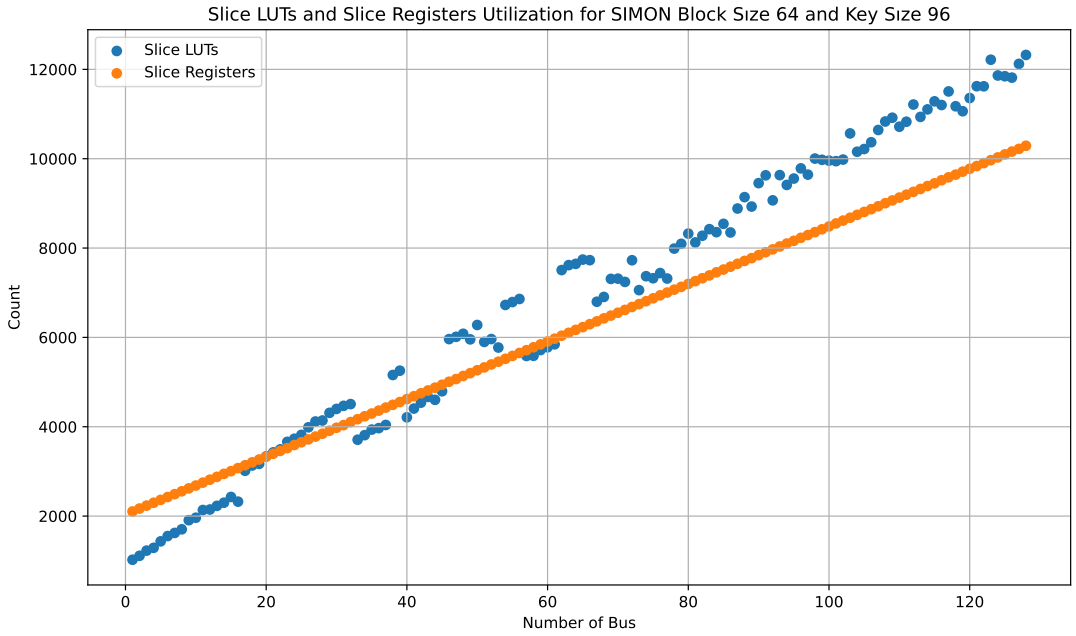


Figure 4.15 : Number of register and LUT changed by the number of buses, SIMON block size is 64 and key size is 96.

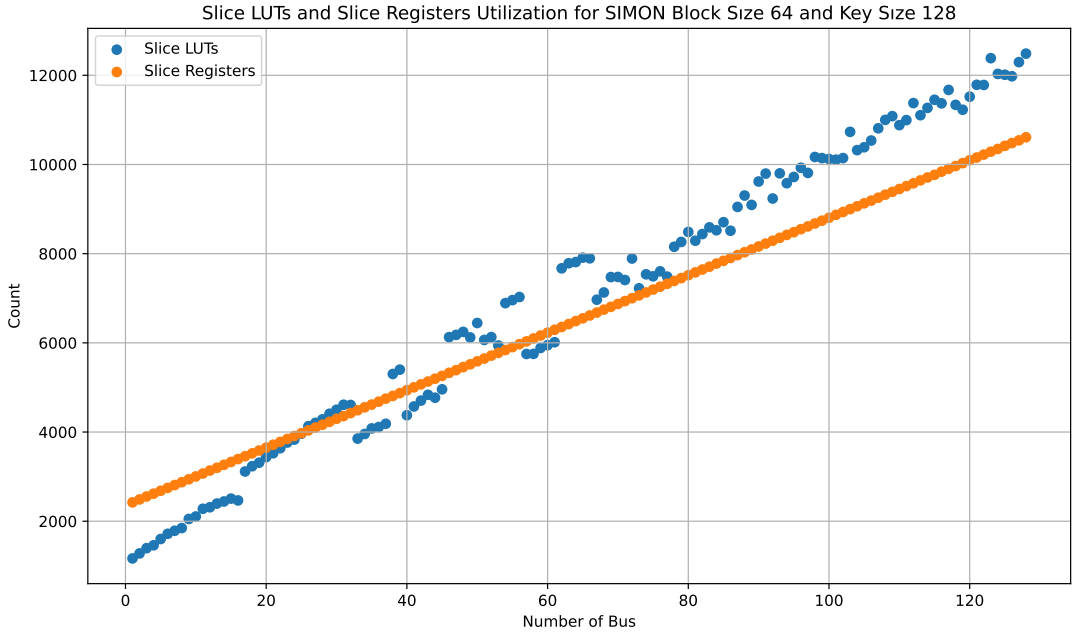


Figure 4.16 : Number of register and LUT changed by the number of buses, SIMON block size is 64 and key size is 128.

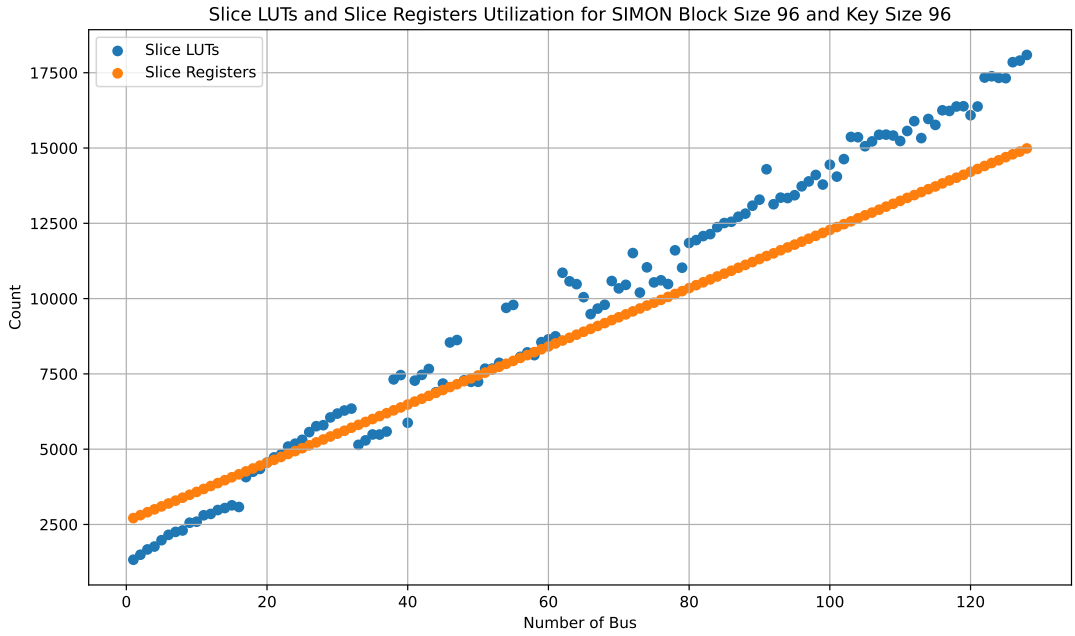


Figure 4.17 : Number of register and LUT changed by the number of buses, SIMON block size is 96 and key size is 96.

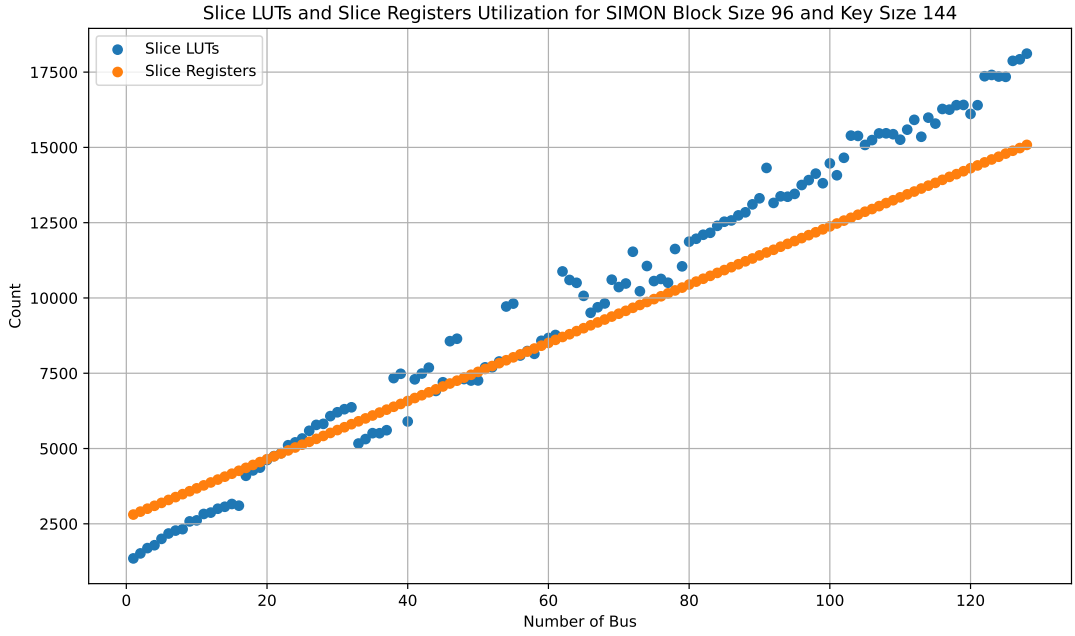


Figure 4.18 : Number of register and LUT changed by the number of buses, SIMON block size is 96 and key size is 144.

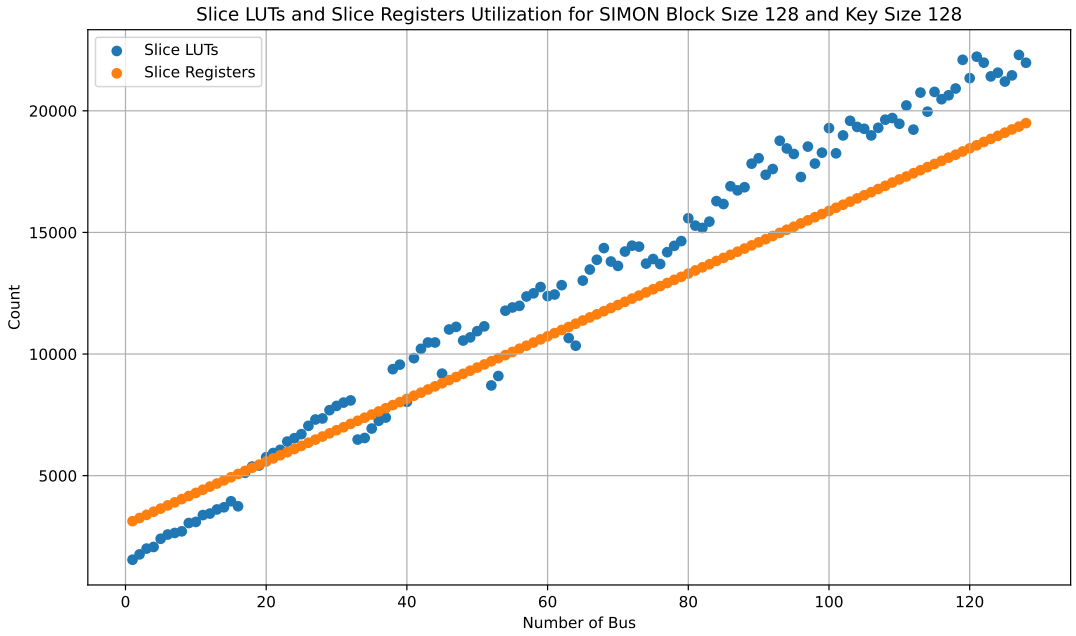


Figure 4.19 : Number of register and LUT changed by the number of buses, SIMON block size is 128 and key size is 128.

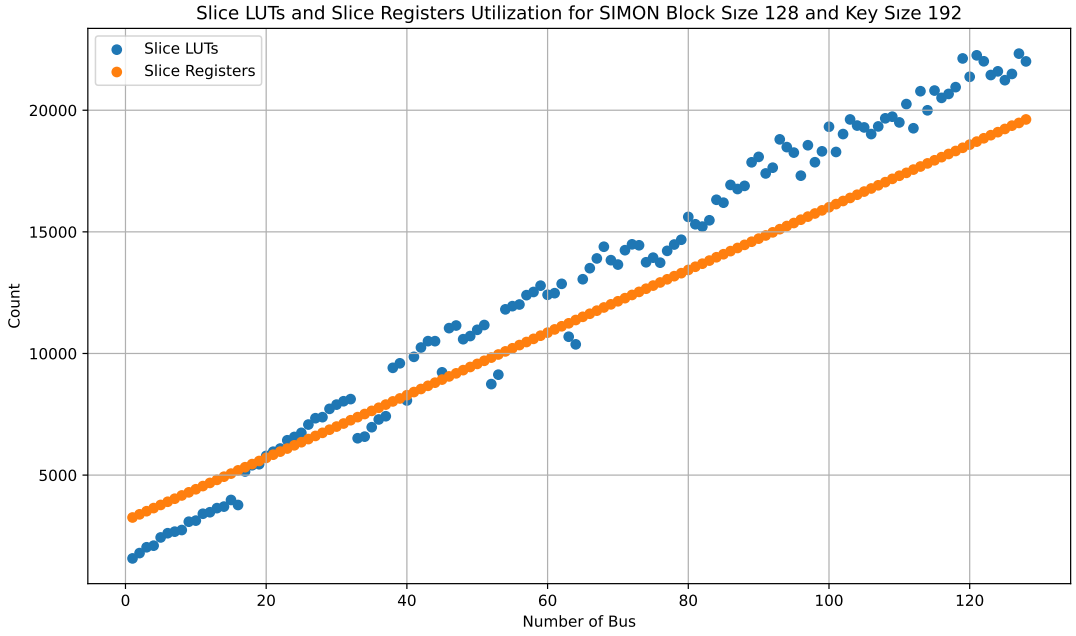


Figure 4.20 : Number of register and LUT changed by the number of buses, SIMON block size is 128 and key size is 192.

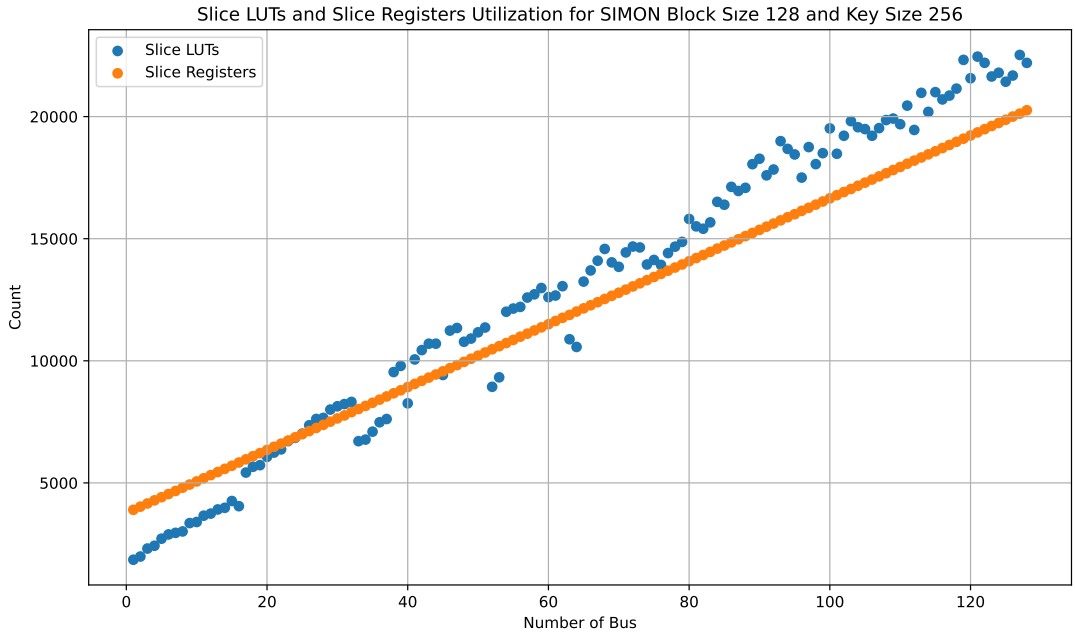


Figure 4.21 : Number of register and LUT changed by the number of buses, SIMON block size is 128 and key size is 258.

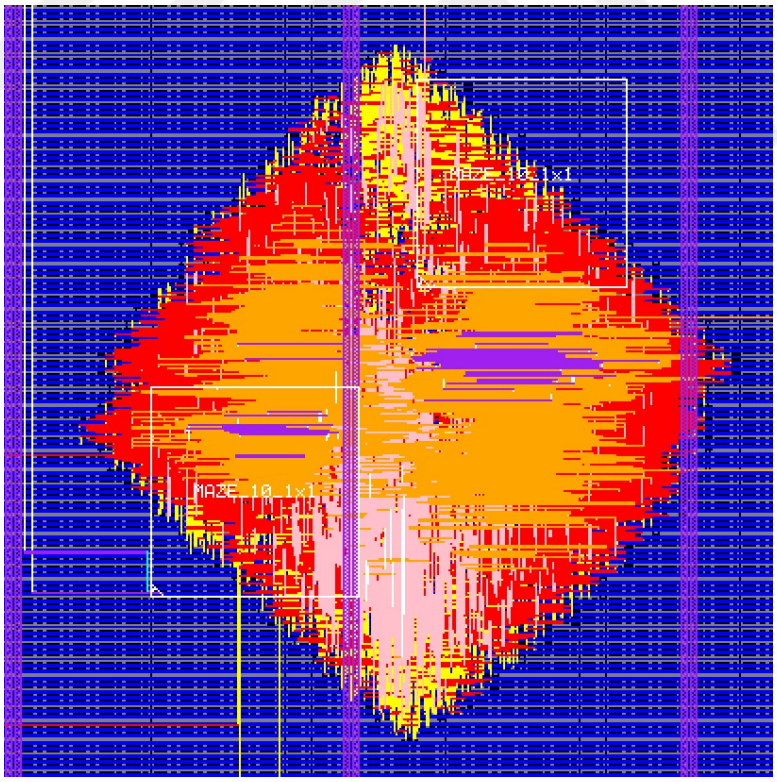


Figure 4.22 : GDSII image of shield digital design.

Figure 4.23 : Produced IC with the active shield digital design.

Figure 4.24 : Looking into IC with microscope.

4.2 Conclusion

In conclusion, the aim of this thesis, designing of a parametric active shield digital design is achieved. There are several outputs obtained from this study.

One of the output from this work is the testing of the randomness of the PRNGs generated by using the lightweight ciphers. These ciphers are block ciphers. They differ from each other based on their block size and algorithm. These differences affect random numbers produced from these PRNGs. These differences are shown on the Table 2.5.

The Table 2.5 shows the results of the five different PRNGs based on different ciphers coded in the Python language. According to these results, random numbers generated by SIMON PRNG have %6 failure rate, SPECK has %10 failure rate, and PRINTcipher has %11 failure rate. The random numbers generated by LRBC PRNG and TinyJAMBU PRNG have %100 failure rate. It shows that these ciphers are not suitable to generate PRNG in CBC mode.

SIMON, SPECK, and PRINTcipher have similar results on randomness. But the second metric is the runtime of generating the 6291456-bit length vector. The runtime of SIMON PRNG is 42 minutes, that of SPECK PRNG is 47 minutes, and that of PRINTcipher PRNG is 463 minutes.

All these results show that SIMON, SPECK, or PRINTcipher are suitable ciphers to generate PRNG. The results are getting better from PRINTcipher, SPECK and SIMON respectively.

The simulation with AMD Vivado shows that the active shield digital design can produce random numbers in each configuration. Attack simulations show that the active shield digital design is working as expected since it can detect the attacks and trigger the alarms. FPGA implementations shows that the active shield digital design is working succesfully in FPGA with attack scenarious.

Utilization shows the relationshipship between parameters of the design and number of the register and number of the lookup table (LUT). There is a linear relationshipship

between number of register and parameters. There is relationship between number of LUT and parameters but this relationshipship is not linear.

The IC with the active shield digital design is produced by the TSMC successfully.

Five different lightweight cipher is implemented in this study. This study can be improved by using more lightweight ciphers, or other different lightweight ciphers than the ones used in this study can be used. The PRNGs generated by other lightweight ciphers then the ones that have been used in this study can have less failure rate in NIST test. Besides, different modes can be utilised in the design of the PRNG. The PRNGs that have designed with different modes can have lower failure rate compared to the PRNGs that is designed with the CBC mode.

The number of the parameters can be increased in further studies. Increasing the number of parameters may provide more flexibility to the system. The optimisation studies can be studied to make the occupied area smaller. Thus, the design has the same speed but the occupied area may become smaller. A method, other than the used method in comparator subblock, can be implemented in future studies.

REFERENCES

[1] **Mun, Y.**, **Kim, H.**, **Lee, B.**, **Han, K.**, **Kim, J.**, **Kim, J.H.**, **Choi, B.D.**, **Kim, D.K. and Ko, H.** (2019). Secure Integrated Circuit with Physical Attack Detection based on Reconfigurable Top Metal Shield, *JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE*, *19*(3), 260–269.

[2] **Vidaković, M. and Vinko, D.** (2023). Hardware-Based Methods for Electronic Device Protection against Invasive and Non-Invasive Attacks, *Electronics*, *12*(21), 4507.

[3] **Wang, H.**, **Shi, Q.**, **Nahiyan, A.**, **Forte, D. and Tehranipoor, M.M.** (2020). A Physical Design Flow Against Front-Side Probing Attacks by Internal Shielding, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *39*(10), 2152–2165.

[4] **Cabal, J.** *Spi-Fpga: SPI Master and SPI Slave for FPGA Written in VHDL*, https://github.com/jakubcabal/spi-fpga.

[5] **Turgut, R.R. and Yalçın, M.** (2024). *Maze-Based Shield Design to Protect ICs Against Invasive Hardware Attacks*.

[6] **Kurt, C. and Başkaya, F.** (2024). *Integration of a Maze Shield Into Digital ASIC Design Flow for Hardware Attack Resistant ICs*.

[7] **Briais, S.**, **Cioranesco, J.M.**, **Danger, J.L.**, **Guilley, S.**, **Naccache, D. and Porteboeuf, T.** (2012). Random Active Shield, *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp.103–113.

[8] **Feng, T.**, **Li, K.**, **Qin, Y. and Yu, S.** (2016). A Dynamic Active Shield against Shield Rerouting Attack, *International Journal of High Performance Computing and Networking*, *9*(5-6), 462–469.

[9] **Jin, R.**, **Yuan, Y.**, **Xin, R.**, **Gan, J.**, **Hu, X.**, **Yu, Y.**, **Li, N. and Zhao, Y.** (2018). Active Shield Design for Security Chip in Smart Grid, ***F. Li, T. Takagi, C. Xu and X. Zhang**, editors, Frontiers in Cyber Security*, Communications in Computer and Information Science, Springer, Singapore, pp.273–281.

[10] **Wang, K.**, **Gu, Y.**, **Zhou, T. and Chen, H.** (2019). Multi-Pair Active Shielding for Security IC Protection, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *38*(12), 2321–2329.

[11] **Xin, R.**, **Yuan, Y.**, **He, J.**, **Li, Y. and Zhao, Y.** (2019). High-Efficient Generation Algorithm for Large Random Active Shield, *Science China Information Sciences*, *62*(3), 39108.

[12] **Xin, R.**, **Yuan, Y.**, **He, J.**, **Zhen, S. and Zhao, Y.** (2020). Random Active Shield Generation Based on Modified Artificial Fish-Swarm Algorithm, *Computers & Security*, *88*, 101552.

[13] **Kerckhof, S.**, **Durvaux, F.**, **Hocquet, C.**, **Bol, D. and Standaert, F.X.** (2012). Towards Green Cryptography: A Comparison of Lightweight Ciphers from the Energy Viewpoint, ***E. Prouff and P. Schaumont**, editors, Cryptographic Hardware and Embedded Systems – CHES 2012*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp.390–407.

[14] **Manifavas, C.**, **Hatzivasilis, G.**, **Fysarakis, K. and Papaefstathiou, Y.** (2016). A Survey of Lightweight Stream Ciphers for Embedded Systems, *Security and Communication Networks*, *9*(10), 1226–1246.

[15] **Fan, X.**, **Mandal, K. and Gong, G.** (2013). WG-8: A Lightweight Stream Cipher for Resource-Constrained Smart Devices, ***K. Singh and A.K. Awasthi**, editors, Quality, Reliability, Security and Robustness in Heterogeneous Networks*, Springer, Berlin, Heidelberg, pp.617–632.

[16] *Enocoro : Hitachi Crypto Technology : Hitachi*, `https://www.hitachi.com/rd/yrl/crypto/enocoro/`.

[17] **Moradi, A.**, **Poschmann, A.**, **Ling, S.**, **Paar, C. and Wang, H.**, (2011). Pushing the Limits: A Very Compact and a Threshold Implementation of AES, Advances in Cryptology – EUROCRYPT 2011, volume6632, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.69–88.

[18] **Good, T. and Benaissa, M.** Hardware Results for Selected Stream Cipher Candidates.

[19] **Hosseinzadeh, J. and Bafghi, A.G.** (2017). Evaluation of Lightweight Block Ciphers in Hardware Implementation: A Comprehensive Survey, *arXiv:1706.03878 [cs]*, `1706.03878`.

[20] **Philip, M.A. and Vaithiyanathan** (2017). A Survey on Lightweight Ciphers for IoT Devices, *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, pp.1–4.

[21] **Biswas, A.**, **Majumdar, A.**, **Nath, S.**, **Dutta, A. and Baishnab, K.L.** (2020). LRBC: A Lightweight Block Cipher Design for Resource Constrained IoT Devices, *Journal of Ambient Intelligence and Humanized Computing*.

[22] *Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core | IEEE Conference Publication | IEEE Xplore*, https://ieeexplore.ieee.org/abstract/document/1690090.

[23] **Bogdanov, A.**, **Knudsen, L.R.**, **Leander, G.**, **Paar, C.**, **Poschmann, A.**, **Robshaw, M.J.B.**, **Seurin, Y. and Vikkelsoe, C.**, (2007). PRESENT: An Ultra-Lightweight Block Cipher, **P. Paillier and I. Verbauwhede**, editors, Cryptographic Hardware and Embedded Systems - CHES 2007, volume4727, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.450–466.

[24] **Guo, J.**, **Peyrin, T. and Poschmann, A.**, (2011). The PHOTON Family of Lightweight Hash Functions, Advances in Cryptology – CRYPTO 2011, volume6841, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.222–239.

[25] **Guo, J.**, **Peyrin, T.**, **Poschmann, A. and Robshaw, M.**, (2011). The LED Block Cipher, Cryptographic Hardware and Embedded Systems – CHES 2011, volume6917, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.326–341.

[26] **Borghoff, J.**, **Canteaut, A.**, **Güneysu, T.**, **Kavun, E.B.**, **Knezevic, M.**, **Knudsen, L.R.**, **Leander, G.**, **Nikov, V.**, **Paar, C.**, **Rechberger, C.**, **Rombouts, P.**, **Thomsen, S.S. and Yalçın, T.**, (2012). PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications, Advances in Cryptology – ASIACRYPT 2012, volume7658, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.208–225.

[27] **Abdelhalim, M.B.**, **El-Mahallawy, M. and A. Elhennawy, M.A.** (2013). Design and Implementation of an Encryption Algorithm for Use in RFID System, *International Journal of RFID Security and Cryptography*, *2*(1), 51–57.

[28] **Li, L.**, **Liu, B. and Wang, H.** (2016). QTL: A New Ultra-Lightweight Block Cipher, *Microprocessors and Microsystems*, *45*, 45–55.

[29] **Zhang, W.**, **Bao, Z.**, **Lin, D.**, **Rijmen, V.**, **Yang, B. and Verbauwhede, I.** (2014). *RECTANGLE: A Bit-slice Lightweight Block Cipher Suitable for Multiple Platforms*.

[30] **Beaulieu, R.**, **Shors, D.**, **Smith, J.**, **Treatman-Clark, S.**, **Weeks, B. and Wingers, L.** (2013). The SIMON and SPECK Families of Lightweight Block Ciphers, **Technical Report404**.

[31] **Yang, G.**, **Zhu, B.**, **Suder, V.**, **Aagaard, M.D. and Gong, G.** (2015). The Simeck Family of Lightweight Block Ciphers, ***T. Güneysu and H. Handschuh**, editors, Cryptographic Hardware and Embedded Systems – CHES 2015*, Springer, Berlin, Heidelberg, pp.307–329.

[32] **Computer Security Division, I.T.L.** (2017). *Round 1 - Lightweight Cryptography | CSRC | CSRC*, https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[33] **Sönmez Turan, M.**, **McKay, K.**, **Çalık, Ç.**, **Chang, D. and Bassham, L.** (2019). Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process, **Technical ReportNIST Internal or Interagency Report (NISTIR) 8268**, National Institute of Standards and Technology.

[34] **Aagaard, M.D. and Zidaric, N.** (2021). ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process, **Technical Report049**.

[35] **Rukhin, A.**, **Soto, J.**, **Nechvatal, J.**, **Smid, M.**, **Barker, E.**, **Leigh, S.**, **Levenson, M.**, **Vangel, M.**, **Banks, D.**, **Heckert, N.**, **Dray, J.**, **Vo, S. and Bassham, L.** (2010). A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, **Technical ReportNIST Special Publication (SP) 800-22 Rev. 1**, National Institute of Standards and Technology.

[36] **Knudsen, L.**, **Leander, G.**, **Poschmann, A. and Robshaw, M.J.B.**, (2010). PRINTcipher: A Block Cipher for IC-Printing, Cryptographic Hardware and Embedded Systems, CHES 2010, volume6225, Springer Berlin Heidelberg, Berlin, Heidelberg, pp.16–32.

[37] **Wu, H. and Huang, T.** (2021). TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms (Version 2), 40.

[38] *Vivado Overview*, `https://www.xilinx.com/products/design-tools/vivado.html`.

[39] *AMD Artix 7 FPGAs*, `https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/artix-7.html`.

[40] *Arduino Mega 2560 Rev3*, `https://store.arduino.cc/products/arduino-mega-2560-rev3`.

[41] *Virtuoso Layout Suite | Cadence*, `https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/layout-design/virtuoso-layout-suite.html`.

CURRICULUM VITAE

Name SURNAME: Ömer Faruk BİRGÜL

EDUCATION:

- **M.Sc.:** 2024, Istanbul Technical University, Faculty of Electrical and Electronics Engineering , Electronics and Communication Engineering
- **B.Sc.:** 2021, Istanbul Technical University, Faculty of Electrical and Electronics Engineering , Electronics and Communication Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2024 ASIC Digital Design Engineer at Yongatek
- 2023-2024 ASIC Digital Design Engineer at Synopsys
- 2021-2022 Research Engineer at ITU Embedded Systems Laboratory

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Birgül Ö. F.**, Yalçın M. E. (2024). Fully Parametric Active Shield Design. *International Graduate Research Symposium*, May 8-10, 2024 Istanbul, Turkey.