HASH FUNCTION OF CRYPTOGRAPHICALLY SECURE PSEUDORANDOM NUMBER GENERATOR FOR HARDWARE ROOT-OF-TRUST

ELAINE ONG EI LING

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic Systems)

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JULY 2021

DEDICATION

This project report is specially dedicated to my mother, who taught me that the best kind of knowledge to have is that which is learned for its own sake. She also always been my source of strength and inspiration when I’m facing hard time. She also like a father for me, the one who taught me that even the largest task can be accomplished if it is done one step at a time. It is such a blessing for you to be in my life.

ACKNOWLEDGEMENT

My deepest appreciation to all that involving and provide me the strength and possibility to complete this final year project successfully. I wish to express my sincere appreciation to my project supervisor, PM. Ir. Dr. Muhammad Nadzir Bin Marsono, for encouragement, guidance, critics, motivation, and friendship. He will always there to provide help whenever you need it. His constructive comments and thoughtful ideas had contributed a lot of positive outcomes to this project. Without his continued support and interest, this thesis would not have been the same as presented here.

A special gratitude to my beloved parents and family who had always supported and encouraged me throughout the though journey in Universiti Teknologi Malaysia (UTM). Their greatest love and motivation have brought me the strength to complete this master program, Not forgotten my beloved fellow friends for their supports. My sincere appreciation also extends to all my colleagues and others who have provide assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space. Thankful and grateful to have all of you in my life.

ABSTRACT

The hardware secure module is the common example of Root-of-Trust that used in cryptographic system, fundamentally it is generating, managing and provide protection to the cryptographic keys and performing cryptographic functions within its secure environments. A random number generator (RNG) is one of the critical components in hardware Root-off-Trust, since one of the most important elements in securing cryptographic system is the keys generation. A cryptographically secure pseudorandom number generator (CSPRNG) is a pseudorandom number generator with properties that suitable to be used in cryptography systems for the keys generation. There are few CSPRNG standardized under NIST SP 800-90A Rev.1 which is Hash_DRBG, HMAC_DRBG and CTR_DRBG. Both Hash_DRBG and HMAC_DRBG is hash based DRBG. This project is performing the study of Hash_DRBG algorithm and understand that the core of the DRBG is the hash function. All the internal process of Hash_DRBG is using hash function such as the instantiate process, reseeding process, and pseudorandom numbers generation process. Therefore, the selection of hash function to be used in the Hash_DRBG is important. There are few SHA family available such as SHA0, SHA1 and SHA2. Based on previous work, SHA0 and SHA1 family algorithm can be break by the generic attacks such as Brute Force attack, domain extender attack, poisoned block attach, etc. Therefore, SHA2 family is preferable in this project, under this family, SHA-256 is used. This is because SHA-256 can provide better robustness compare to SHA-512. In the framework of multiple cryptographic cores, two SHA-256 can perform better in term of higher throughput and lower internal state compare to SHA-512. This project will be focusing on SHA-256 algorithm and applied pipeline architecture on the algorithm to help on decreasing the critical path for better performance.

ABSTRAK

Modul keselamatan perkakasan adalah contoh biasa “Root-of-Trust” yang digunakan dalam sistem kriptografi, ia berfungsi untuk menghasilkan, mengurus dan memberikan perlindungan kepada kunci kriptografi dan melakukan fungsi kriptografi dalam persekitarannya. Penjana nombor rawak (RNG) adalah salah satu komponen penting dalam perkakasan “Root-of-Trust”, kerana salah satu elemen terpenting dalam memastikan keselamatan sistem kriptografi adalah penjanaan kunci. Penjana nombor pseudorandom kriptografi yang selamat (CSPRNG) adalah penjana nombor pseudorandom dengan sifat yang sesuai untuk digunakan dalam sistem kriptografi dalam penjanaan kunci. Terdapat beberapa CSPRNG yang diseragamkan di bawah NIST SP 800-90A Rev.1 iaitu Hash_DRBG, HMAC_DRBG dan CTR_DRBG. Hash_DRBG dan HMAC_DRBG adalah DRBG berasaskan hash. Projek ini memberi perhatian dalam kajian algoritma Hash_DRBG dan memahami bahawa teras DRBG adalah fungsi hash. Semua proses dalaman Hash_DRBG seperti proses instantiate, proses reseeding, dan proses penjanaan nombor pseudorandom menggunakan fungsi hash. Oleh itu, pemilihan fungsi hash yang perlu digunakan dalam Hash_DRBG adalah penting. Terdapat beberapa keluarga SHA yang tersedia seperti SHA0, SHA1 dan SHA2. Berdasarkan ujikaji sebelumnya, algoritma keluarga SHA0 dan SHA1 berjaya diserang oleh serangan generik seperti serangan “Brute Force”, serangan pemanjang domain, penyekat blok beracun, dll. Oleh itu, SHA-256 yg dibawah keluarga SHA2 menjadi pilihan dalam projek ini. Ini kerana SHA-256 dapat memberikan kekuatan yang lebih baik berbanding dengan SHA-512. Dalam kerangka pelbagai teras kriptografi, dua SHA-256 dapat menunjukkan prestasi yang lebih baik dari segi “throughput” yang lebih tinggi dan keadaan dalaman yang lebih rendah berbanding dengan SHA-512. Projek ini akan menumpukan pada algoritma SHA-256 dan seni bina saluran paip yang diterapkan pada algoritma untuk membantu mengurangkan jalan kritikal untuk prestasi yang lebih baik.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BRAM	-	Block Random Access Memory
CPU	-	Central Processing Unit
CSPRNG	-	Cryptographic Secure Pseudo-Random Number Generator
EN	-	Extract Number
FPGA	-	Field Programmable Gate Array
GN	-	Generate Number
HDL	-	Hardware Description Language
HSM	-	Hardware Secure Module
IC	-	Integrated Circuit
IG	-	Initialize Generator
LFSR	-	Linear Feedback Shift Register
LSB	-	Least Significant Bit
MMTG	-	Modified Mersenne Twister based on Chaotic Logistic Mapping
MSB	-	Most Significant Bit
MT	-	Mersenne Twister
MUX	-	Multiplexer
PM	-	Polynomial Modulator
PRNG	-	Pseudo-random Number Generator
RAM	-	Random Access Memory
RNG	-	Random Number Generator
ROM	-	Read-only-Memory
RoT	-	Root-of-Trust
RTL	-	Register Transfer Level
SoC	-	System-on-Chip
TGFSR	-	Twisted Generalized Feedback Shift Register
TPM	-	Trusted Platform Module
TRNG	-	True Random Number Generator

CHAPTER 1

INTRODUCTION

1.1 Research Background

In nowadays digital world, information security is getting crucial in the modern communications and computing system. From communication perspective, the security is mainly on protecting user sensitive or personal information, even in mandates the confidentially and integrity of communicated data, validating the communicating peers, identify and protecting the communications entities and especially in the protection of applications from malicious software. If looking from industrial sector, security is mainly needed to guarantee the machines is functioning properly and maintain it. This is to ensure it works safely and maintain profitability.

The hardware Root-of-Trust (RoT) is basically created or implemented as the foundation of trust that extent from System-on-Chip (SoC) to the operating system, applications and even to the external communications [1]. SoC is an integrated circuit (IC) that integrate the entire functional electronic or computer system onto it. As what it means by its name, the entire system is on a single chip, where it includes the central processing unit, internal memory, registers as well as input and output ports. Mainly hardware RoT is built by four basic element such as central processing unit (CPU), identification, key management, and encryption. How to have a good key management? This will directly depend on the random number generation. Especially in a cryptographic system, RoT always a trusted source because the cryptographic security is directly depending on the strengths of the secret keys that use for the data encryption and decryption and in performing the specific functions.

Therefore, random number generators (RNGs) are critical for the cryptographic application that need ultimate security since RNG is use as the key generation and the authentication protocols. The random numbers need to be nonreproducible, unpredictable, and having good statistical properties. Random number generator can be classified in two category there is a true random number generator (TRNG) and pseudo-random number generator (PRNG). TRNG is non-deterministic and claimed to have truly random as the randomness sources are implemented on hardware and this physical process is unpredictable. PRNG is different with TRNG, it will rely on the initial input value “seed” to produce deterministic, periodic sequence of numbers. This generator is claimed to be not provably random since the output can be predict when the “seed” is known [2], [3], [4].

1.2 Problem Statement

The PRNG that suitable for cryptographic systems is known as cryptographically secure pseudo-random number generators. The requirements for standard PRNG is also satisfied for cryptographic used PRNG, but the reverse is not true. For a PRNG to be the cryptographically secure pseudo-random number generator, there are two major requirements need to fulfill. First, it needs to satisfy the next-bit test. The person should not be able to predict the bit *k+1* even though he knows all the *k* bits from the initial state value of the PRNG. Secondly is that it requires to withstand the state compromise extensions. Let’s say if parts or all its state is known or somehow revealed by the attackers, it should be impossible for the attacker to reconstruct all previous random numbers prior to the evaluation [5].

From the literature review of previous work on cryptographic PRNG, the commonly used PRNG is the Mersenne Twister and the linear feedback shift register (LFSR) method. Mersenne Twister is been used in various simulation since it is popular for its long period, high performance, high equidistribution, highly random sequence and rapid number generation [6], [7], [8], [9]. Still, it has poor diffusion issue. Another frequently used PRNG is LFSR method, it is claimed to be fast and minimal computation. Besides, it is easy to be implement in either software or hardware [10]. But when it comes to cryptographic standpoint, LFSR is not consider as a cryptographic secure as the Berlekamp-Massey algorithm able to observed the 2n consecutive bits of the n-bit LFSR structure [11].

There are few standard CSPRNG algorithms that declare by NIST SP 800-90A Rev.1 which is the Hash_DRBG, HMAC_DRBG and CTR_DRBG. Both Hash_DRBG and HMAC_DRBG is uncontrovertible and proven, and both using hash function for the instantiate and reseeding. By having a high performance of hash function, the performance of the DRBG will be increases as well.

1.3 Research Aim and Objectives

This research is targeting to implement a hash function that supporting on the implementation of standard CSPRNG that can be used in cryptographic hardware RoT. The objectives of the research are as below:

To implement and design a hash function algorithm which targeting in encryption usage for cryptography at register transfer level (RTL) by using Verilog.

To validate and analyse the performance of the hash function that used in CSPRNG for SoC Root-of-Trust.

1.4 Report Outline

This research report consists of 5 chapters. The literature review of previous work is discussed in Chapter 2. The important elements in SoC hardware Root-of-Trust is discussed. The overview of random number generator is discussed and analyse which is the suitable random number generator to be used in cryptographic system.

In Chapter 3, the research methodology throughout this project is discussed. The architecture of the hash function that used with the CSPRNG under NIST SP 800-90A Rev. 1 standard will be discussed. The proposed design on the hash function will be explain in detail in this chapter.

Chapter 4 is about the result and discussion of the proposed design. This chapter will mainly be discussing the result obtained from the synthesis and simulation of the proposed design.

Lastly, the conclusion and future work discussion will be cover in Chapter 5. What does the gain from Chapter 4 and what can be done to improve the problems will be discussing here.

REFERENCES

[1] T. Root, “Using hardware secure modules to protect SoCs,” 2018.

[2] A. Yadav, “VCU Scholars Compass Design and Analysis of Digital True Random Number Generator Design and Analysis of Digital True Random Number Generator,” 2013.

[3] B. Yang, “True Random Number Generators for FPGAs,” no. September, 2018.

[4] M. Stipˇ, *Open Problems in Mathematics and Computational Science*, no. November 2014. 2014.

[5] S. Random and N. Generators, “Secure Random Generators.”

[6] A. Jagannatam, “Mersenne Twister – A Pseudo Random Number Generator and its Variants,” *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 2008, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.175.9735&amp;rep=rep1&amp;type=pdf.

[7] S. Chandrasekaran and A. Amira, “High performance FPGA implementation of the Mersenne Twister,” *Proc. - 4th IEEE Int. Symp. Electron. Des. Test Appl. DELTA 2008*, pp. 482–485, 2008, doi: 10.1109/DELTA.2008.113.

[8] Y. Li, J. Jiang, H. Cheng, M. Zhang, and S. Wei, “An efficient hardware random number generator based on the MT method,” *Proc. - 2012 IEEE 12th Int. Conf. Comput. Inf. Technol. CIT 2012*, pp. 1011–1015, 2012, doi: 10.1109/CIT.2012.208.

[9] L. Zhang, X. Zou, and B. Chen, “Modified mersenne twister based on chaotic logistic mapping strategy,” *Proc. - 2019 6th Int. Conf. Inf. Sci. Control Eng. ICISCE 2019*, pp. 783–788, 2019, doi: 10.1109/ICISCE48695.2019.00160.

[10] M. Han and Y. Kim, “Unpredictable 16 bits LFSR-based true random number generator,” *Proc. - Int. SoC Des. Conf. 2017, ISOCC 2017*, no. x, pp. 284–285, 2018, doi: 10.1109/ISOCC.2017.8368897.

[11] E. Dubrova, “A transformation from the fibonacci to the galois NLFSRs,” *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 5263–5271, 2009, doi: 10.1109/TIT.2009.2030467.

[12] "Security Subsystems for Systems-on- Chip (SoCs) Common concepts and usage paradigms of security subsystems."

[13] P. Maene, "Lightweight Roots of Trust for Modern Systems-on-Chip," no. October, 2019.

[14] D. DiCarlo, "Random Number Generation: Types and Techniques," *Sr. Honor. Theses*, 2012, [Online]. Available: https://digitalcommons.liberty.edu/honors/308.

[15] V. Van Der Leest, E. Van Der Sluis, G. J. Schrijen, P. Tuyls, and H. Handschuh, "Efficient implementation of true random number generator based on SRAM PUFs," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics),* vol. 6805 LNCS, pp. 300–318, 2012, doi: 10.1007/978-3-642-28368-0_20.

[16] E. Barker and J. Kelsey, "NIST SP 800-90A Revision 1 Recommendation for random number generation using deterministic random bit generators (revised)," *NIST Spec. Publ.*, no. Giugno, 2015.

[17] C. Jeong, "Cryptography Engine Design for IEEE 1609.2 WAVE Secure Vehicle Communication using FPGA," 2014, [Online]. Available: http://scholarworks.unist.ac.kr/bitstream/201301/10491/1/Cryptography Engine Design for IEEE 1609.2 WAVE Secure Vehicle Communication using FPGA.pdf.

[18] L. Baldanzi *et al.*, "Cryptographically secure pseudo-random number generator IP-core based on SHA2 algorithm," *Sensors (Switzerland),* vol. 20, no. 7, 2020, doi: 10.3390/s20071869.

[19] W. Penard and T. Van Werkhoven, "Chapter 1 On the Secure Hash Algorithm family," pp. 1–17.

[20] A. A. Maaita, H. A. A. Al Sewadi, A. K. Husain, and O. A. Hassan, "A Cryptographically Secure Multi-stage Pseudo-random Number Generator," no. May 2015, 2017, doi: 10.17148/IJARCCE.2015.4503.

[21] H. Mestiri, "Efficient FPGA Hardware Implementation of Secure Hash Function SHA-2," no. December 2014, pp. 9–15, 2015, doi: 10.5815/ijcnis.2015.01.02.

[22] S. Suhaili, "Design of High-Throughput SHA-256 Hash Function based on FPGA," 2017.

[23] M. Padhi, "An Optimized Pipelined Architecture of SHA-256 Hash Function,"

pp. 17–20, 2017.

[24] M. Clayton, M. Clayton, and K. Brown, “Federal Register/ Vol.67, No. 165/ Monday, August 26, 2002/ Notices,” vol. 67, no. 165, pp. 3–4, 2002.

[25] A. Publication, “Secure Hash Standard (SHS),” vol. 4, 2015.

[26] Xilinx, “ZCU106 Evaluation Board,” *Xilinx.com*, vol. UG1224, no. v1.4, pp. 1–134, 2019.

[27] F. Publication, “Archived Publication,” vol. 2, 2004, [Online]. Available: http://csrc.nist.gov/publications/PubsFIPS.html#fips180-4.