UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica

https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRÓNICA

A RESEARCH VISION

Design and parallel implementation on Artix-7 FPGA of the SIMON32/64 Cryptosystem

Diseño e implementación en forma paralela en la FPGA Artix-7 del Criptosistema SIMON32/64

José Gabriel Rincón-González [1], *Andrés Hernández-Baquero* [2], *Fabián Velásquez-Clavijo* [3]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 20/05/2022
Recibido: 30/05/2022
Aceptado: 11/07/2022

Keywords:
Artix-7
Cryptosystem
Digital transformation
FPGA
Private key
SIMON

ABSTRACT

In the rise of digital transformation, information security is positioned in a prominent place because it provides confidence to society in the use of information technologies. For this reason, there is growing research in this area. One way to achieve this security is through private key cryptosystems such as SIMON. This work presents a parallel implementation in the Artix-7 FPGA of the SIMON32/64 cryptosystem to obtain an average delay time of 37.8 ns for the cipher and 54.1 ns for the decypher that allows real-time encryption applications.

Palabras clave:
Artix-7
Criptosistema
Transformación digital
FPGA
Clave privada
SIMON

RESUMEN

En el auge de la transformación digital, la seguridad de la información se posiciona en un lugar destacado porque brinda confianza a la sociedad en el uso de las tecnologías de la información. Por esta razón existe una creciente investigación en esta área. Una forma de conseguir esta seguridad es a través de criptosistemas de clave privada como SIMON. En este artículo se presenta una implementación en forma paralela en la FPGA Artix-7 del criptosistema SIMON32/64 para obtener un tiempo de retardo promedio de 37.8 *ns* para el cifrador y 54.1 *ns* para el descifrador que permiten las aplicaciones de cifrado en tiempo real.

1 Electronic Engineer, Universidad de los Llanos, Colombia. E-mail: jose.rincon.gonzalez@unillanos.edu.co

2 Electronic Engineer, Universidad de los Llanos, Colombia. MSc, PhD, University of Sao Paulo, Brazil. Professor at Universidad de los Llanos, Colombia. E-mail: wahernandezb@unillanos.edu.co

3 Electronic Engineer, Universidad de los Llanos, Colombia. MSc, University of EAFIT, Colombia. Professor at Universidad de los Llanos, Colombia. E-mail: fvelasquez@unillanos.edu.co

Cite this article as: J. G. Rincón-González, A. Hernández-Baquero, F. Velásquez-Clavijo, "Design and parallel implementation on Artix-7 FPGA of the SIMON32/64 Cryptosystem", *Visión Electrónica*, vol. 16, no. 2, pp. 194-203, 2022. https://doi.org/10.14483/22484728.20391

1. Introduction

IT security currently has great challenges, especially with the issue that has to do with cybersecurity. The cyberattack on SolarWinds in 2020 is an example of how unauthorized access to software source code can compromise the information of thousands of companies around the world [1], so it is required to take robust and efficient measures that are at the forefront to protect information against possible interceptions made by cybercriminals, who aim to access its content since they consider it as a vitally important asset.

On the other hand, the COVID-19 pandemic has suddenly caused a technological transformation in all social areas and in the way we live such as, for example, the internet of things and remote and collaborative work, however, the security policies of ordinary people have the potential to expose companies or organizations to great risks of cyber-attacks, therefore a contingency plan must be created to create a secure remote work environment and thus lessen possible unauthorized access to information [2].

Symmetric key cryptography is widely used to encrypt information in commercial applications, which allows it to be transmitted through an insecure channel while preserving its confidentiality and integrity. In 1974, the NBS, now NIST *(National Institute of Standards and Technology),* chose the algorithm proposed by IBM *(International Business Machines)* as the standard for symmetric encryption in commercial communications, a new development based on the Lucifer algorithm, with Horst Feistel as one of its inventors. A year later, in 1975, the NSA *(National Security Agency)* applied severe restrictions to this algorithm, reducing the 128-bit key of the original Lucifer to 56 bits, with the consequent criticism from experts, the DES. *(Data Encryption Standard)* was adopted as a standard and authorized for use in communications in 1976 [3]. Twenty years later, DES succumbed to a brute force attack on a network at the end of the 1990s, although some of its reduced variants could also be breached with a computer, as demonstrated by E. Biham and A. Shamir in [4] by applying differential cryptanalysis.

Based on the above-mentioned facts, in 1997 NIST opened a competition to select a new symmetric encryption algorithm and a year later, in 1998, the results of the most outstanding algorithms were published, including the scheme proposed by J. Daemen and V. Rijmen [5], which has been the Advanced Encryption Standard AES *(Advanced Encryption Standard)* since 2000, when NIST decided to select this algorithm. Rijmen [5], which has been the Advanced Encryption Standard (AES) since 2000 when NIST decided to choose this algorithm. In 2007 N. Velásquez and N. Pineda implemented the AES algorithm using an FPGA device obtaining good results in terms of performance and hardware resource consumption [6].

One of the advantages of block ciphers over stream ciphers is their high diffusion in the cryptogram elements, which is why most symmetric algorithms use this type of scheme.

A disadvantage is the low speed of encryption due to the need to process complete blocks of information, which has caused the emergence of new ultra-lightweight block ciphers such as present proposed by A. Bogdanov, L.R. Knudsen, G. Leander et. al. [7] in 2007.

There are several lightweight block ciphers such as LED developed by J. Guo, T. Peyrin, A. Poschmann et. al. [8] in 2011, this type of ciphers are intended for implementations in devices with few hardware resources. F. Velásquez and J. Castaño presented some applications of cryptographic algorithms on FPGA [9], [10]. In addition, FPGAs have an efficient use in data acquisition systems and the implementation of digital filters presented in the works of W. Enriquez, P. Nazate, O. Marcillo [11] and C. Hernandez, E. Jacinto [12].

Symmetric light block cryptography has been increasingly in demand in recent years as a result of the recent electronics industry in which devices for transmission, processing, and storage of confidential information are manufactured. These devices, due to their reduced area, have very few hardware resources and limited electrical power supply, so encrypting the information would imply that this process would be as efficient as possible and with less computational cost. Thus, in 2013, the NSA publishes the encryption SIMON and SPECK as a series of lightweight block cipher families, the term lightweight refers to the fact that this type of cipher requires less computing power in its execution. They are intended for devices with very low computational power (Efficiency) and have the capability of a suitable mode of operation (Security). SIMON is designed for optimal hardware performance and is based on a classical Feistel-like scheme [13]. In 2016 P. Maene and I. Verbauwhede performs research work in which they implement the block cipher algorithms AES, KATAN, PRESENT, PRINCE, RECTANGLE, SIMON, and SPECK using an ASIC device and a Virtex-6 FPGA.

The FPGA is programmed to execute the algorithms in a combinational circuit, i.e., they do not use the clock signal to execute their encryption and key expansion operations, but rather in a series of cascaded interconnected circuit blocks to obtain the cryptogram, however, this type of design has the disadvantage of generating a larger area occupation [14]. FPGA devices present certain advantages over other technologies, such as microcontrollers, since it provides more flexibility, reconfiguration features, low-cost designs, and efficient utilization of resources and hardware architecture [15].

A. Shahverdi, M. Taha, and T. Eisenbarth propose in 2017 a hardware implementation of the SIMON algorithm to prevent side-channel attacks and perform a comparison with the AES and PRESENT algorithm using an FPGA device and ASIC [16].

S. Sheikhpour, M. Hassani I, and A. Mahani in April this year address the application of the SIMON algorithm to obtain a configurable high-performance architecture to optimize the occupied area, paying attention to the power/energy ratio, considered an issue fundamental in low-resource devices [17]. The reach that the SIMON algorithm has had today is significant, so much so that it has been able to be applied as a cipher in Big Data, as demonstrated by A. Muthumari, J. Banumathi, S. Rajasekaran et. al. [18].

This paper presents an implementation of the SIMON cryptosystem encryption and decryption algorithm in its smallest configuration, i.e., *SIMON*32/64, programmed in VHDL language using the Vivado 2020.2 integrated development environment (IDE) and Digilent's Basys 3 development board which has an integrated Artix-7 FPGA. A theoretical presentation of the SIMON cryptosystem is given in section 2. Section 3 shows the design of the cryptosystem, Section 4 presents the results of the implementation and finally, Section 5 presents the conclusions of the work.

2. SIMON cryptosystem

The SIMON cryptosystem proposed by the NSA in 2013 [13] is composed of the key generation function and the Round Functions. The SIMON encryptor and decryptor use the same functions except that the decryptor uses the order of the generated keys in reverse.

The SIMON block cipher with a n-bit word (and thus a $2n$-bit block) is denoted as *SIMON* $2n$, where n must be 16, 24, 32, 32, 48 or 64 to form a block of two equal parts. *SIMON*$2n$ related to a word of mn bits as key size will be referred to as *SISIMON* $2n/mnn$. In the particular case of *SIMON* 32/64 the text block in the plane is 32-bit and uses a 64-bit key. The value of m, as will be seen later, will be the number of words into which the bit block of the key can be divided into parts of equal length.

Formally the encryption transformation R^k for an element $k \in GF(2)^n$ is defined as a round function for *SIMSIMON*$2n/mn$ dependent on a key k as a two-stage Feistel map, being.

$$R_k: GF(2)^n \times GF(2)^n \longrightarrow GF(2)^n \times GF(2)^n \quad (1)$$

The transformation relation, i.e., the operations of two elements of the field produce as a result another element of the same field. From the computational point of view, equation (1) can be expressed as follows

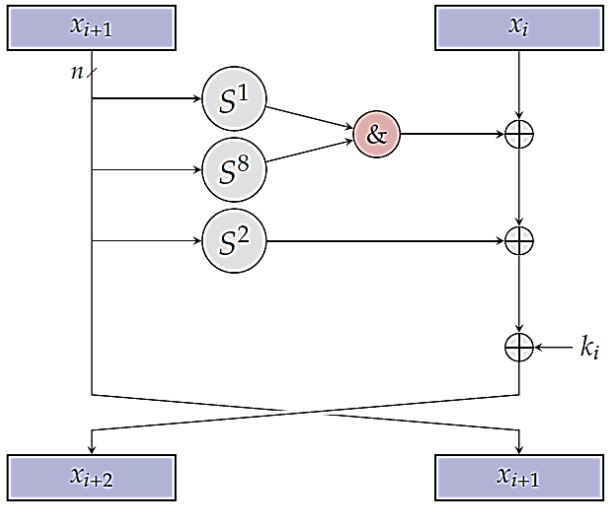
$$R_k(x, y) = (y \oplus f(x) \oplus k, x) \quad (2)$$

Where $f(x) = (Sx \& S^8 x) \oplus S^2 x$ and k is the round key. The decryptor is defined with the inverse transformation $R k^{-1}$ given by.

$$R_k^{-1}(x,y) = (y, x \oplus f(y) \oplus k) \quad (3)$$

Understand x, y under the SIMON context as the two blocks that make up the plaintext. The operations performed in the SIMON algorithm per round are summarized in Figure 1.

Figure 1. SIMON round function. [13]



The block diagram shown in Figure 1 presents the input data x partitioned into two blocks xi and $xi+1$ each of size n, corresponding to x and y in equation (2). Then $xi+1$ is shifted circularly to the left p cycles (s^p) and AND is done between the results of shifting one and eight cycles. Then a set of XOR operations is done between the results and the round key ki and finally, at the end of the round the concatenated output $xi+2$ is constructed with $xi+1$ returning as input to the next round. This process is repeated 32 times for the case of *SIMON*32/64 until the cryptogram is generated.

The key generation *(Key Schedules)* of the *SIMON*32/64 algorithm takes a 64-bit master key as a seed and from it generates a set of 4 16-bit words. The round key generation is shown in Figure 2.

In the above diagram blocks $ki+3$, $ki+2$, $ki+1$, and ki make up the master key in the initial round. The block $ki+3$ is shifted circularly to the right three cycles and an XOR is done with the block $ki+1$, this result is shifted circularly to the right one cycle and simultaneously an XOR is done with the block ki. Then a set of XOR operations between the results and the two constants c and $z0$ is performed, finally, the result is stored in $ki+3$, while $ki+3$ is stored in $ki+2$, $ki+2$ is stored in $ki+1$ and $ki+1$ is stored in ki which is the key generated in the round. This process is repeated 32 times for *SIMON*32/64. The value of the constant $z0$ is predefined according to the size of the block and the size of the key, in the case of *SIMON*32/64 $z0$ = 11111010001001010101100000111001001 10 [13].

3. Design and implementation of *SIMON*32/664.

The encryption and decryption were designed and implemented at the register transfer level (RTL) using VHDL and verified by dynamic simulations using Vivado Simulator. The design was synthesized and compiled using the software package for the Basys 3 Artix-7 FPGA with the synthesis tool in Vivado 2020.2. Finally, the occupied area results were observed with the synthesis reporting tool and the critical path with the timing summary reporting tool in Vivado 2020.2.

3.1. Cipher design and construction

The *SIMON*32/64 cipher block is a combinational circuit, the main component has as input the 32-bit plaintext x and the 64-bit master key k, while the output corresponds to the 32-bit ciphertext $R^k(x, y)$, as shown in Figure 3.

Figure 2. Key expansion function for m = 4. [13].

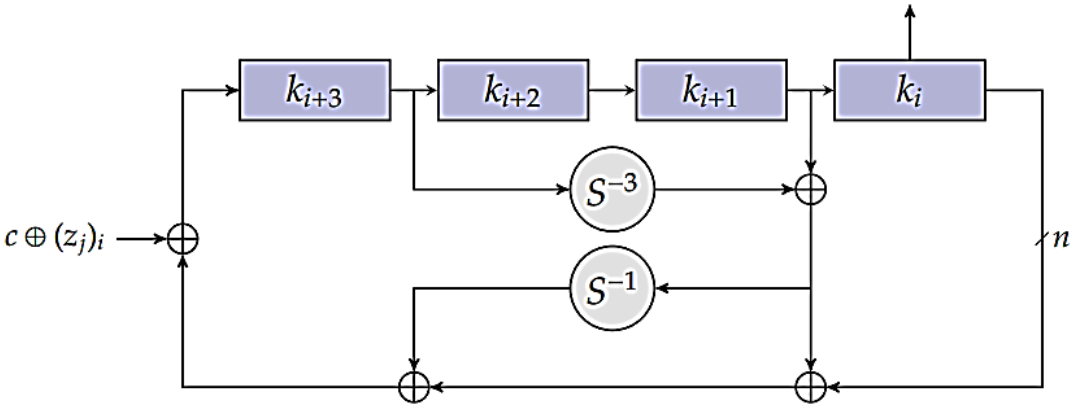
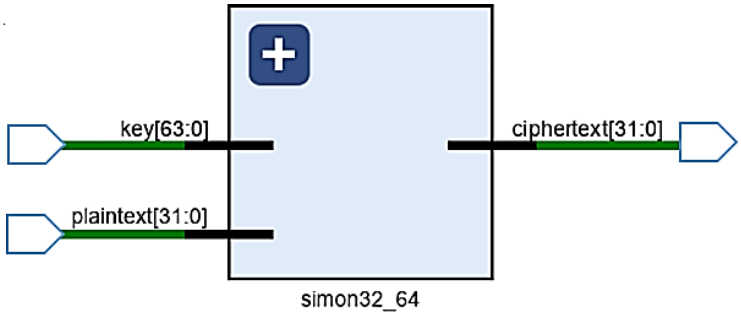


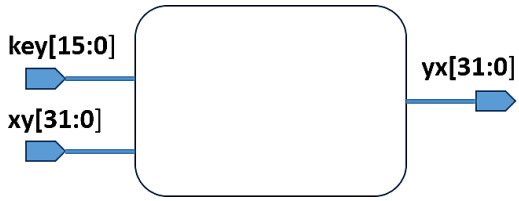
Figure 3. Diagram of the main block of the cipher.



Source: own.

A basic block that executes the function of a SIMON round was designed, implementing the algorithm shown in Figure 1, and obtaining the component shown in Figure 4. This component has as input the round key *ki* of 16 bits in length and a block *xy* that refers to the 2 words *xi* and *xi*+1 that make up the round plaintext state vector, on the other hand, the output vector *yx* is the state vector of the ciphertext generated in round *xi*+1 and *xi*+2.

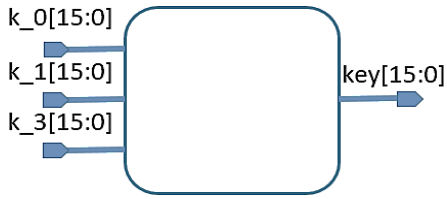
Figure 4. Basic block of round function.



Source: own.

According to Figure 2 the key expansion function used for each round of the words *ki*+3, *kki*+1, and *kki* to generate a new key. Figure 5 shows that 3 logical vectors of length 16 bits are input to the single component and one has a key of equal length as the output of the round.

Figure 5. SIMON key expansion function basic block.

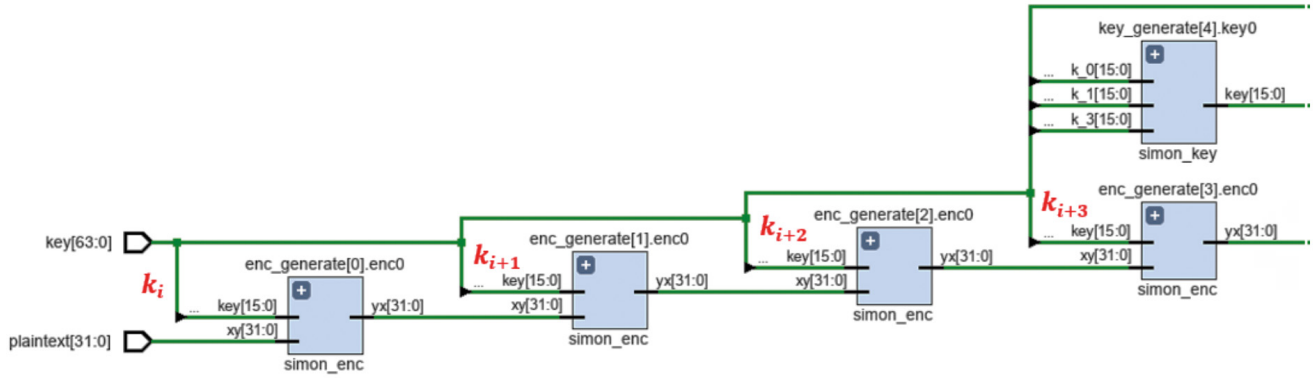


Source: own.

The two basic blocks were replicated and connected to create a sequence of blocks that generate a 32-bit long SIMON cryptogram. The first four words *kii*+3, *kii*+2, *ki*+1 and *ki* into which the 64-bit key is divided are the first input keys for the first four round functions as shown in Figure 6.

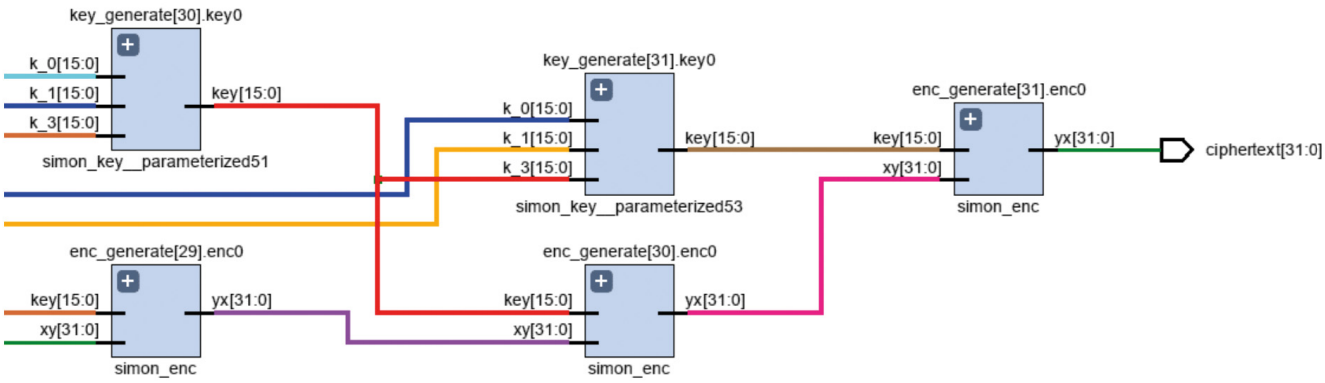
Starting with the fifth key, the remaining 28 basic blocks were connected in sequence to generate the 32 keys. As shown in Figure 7, each output value of a generated round function block is the input of the next cipher block, in which its key also enters simultaneously. In this way, the cryptogram was generated.

Figure 6. Sequence of connections for the SIMON32/64 encryption process.



Source: own.

Figure 7. Final connection sequence for the SIMON32/64 encryption process.

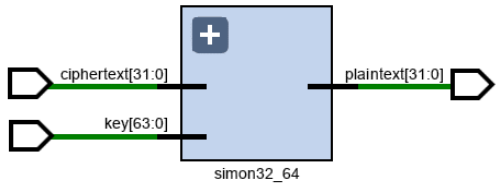


Source: own.

3.2. Decryptor design and construction

For the SIMON32/64 decryptor, the same basic blocks shown in Figure 4 and Figure 5 were used for the round cipher function and the key generation function respectively. In the main entity, 32 bits were declared for the ciphertext, 64 bits for the master key, and 32 bits for the plaintext as shown in Figure 8.

Figure 8. Diagram of the main block of the decryptor.



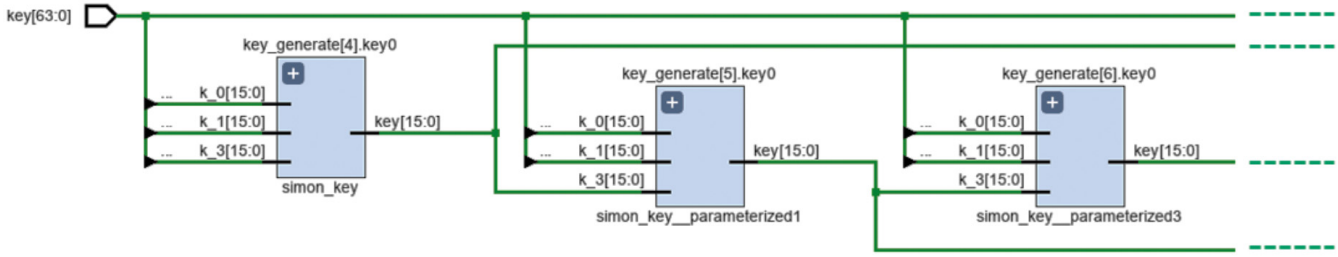
Source: own.

The combinational circuit for the decryptor was generated by also interconnecting a sequence of basic blocks of both functions, in this case, 28 blocks were connected as shown in Figure 9 for the keys from rounds 5 to 32.

For the decryption algorithm *SIMON*32/64 in the first round, the connection shown in Figure 10 was made, in which the block generating the key 32 was connected to the first round function block.

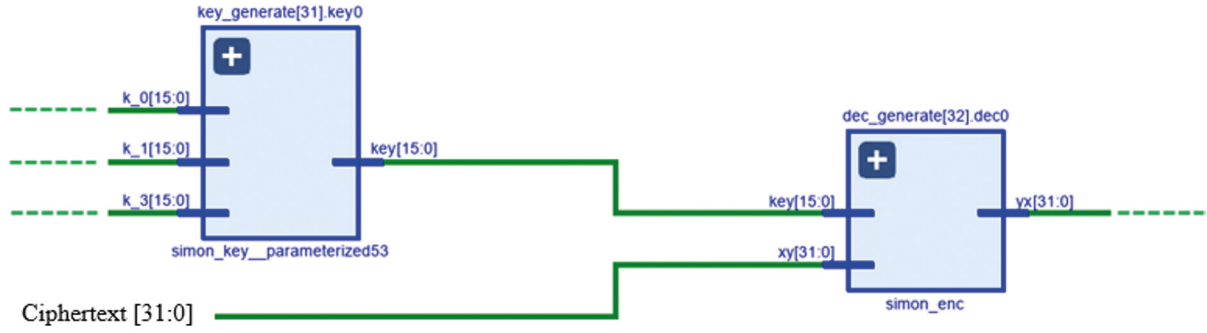
Figure 11 shows the decryptor using the keys in the opposite order to the cipher, so the last four round function blocks have as input the four words that make up the master key, generating the plaintext.

Figure 9. Connections of the decryptor key generation blocks.



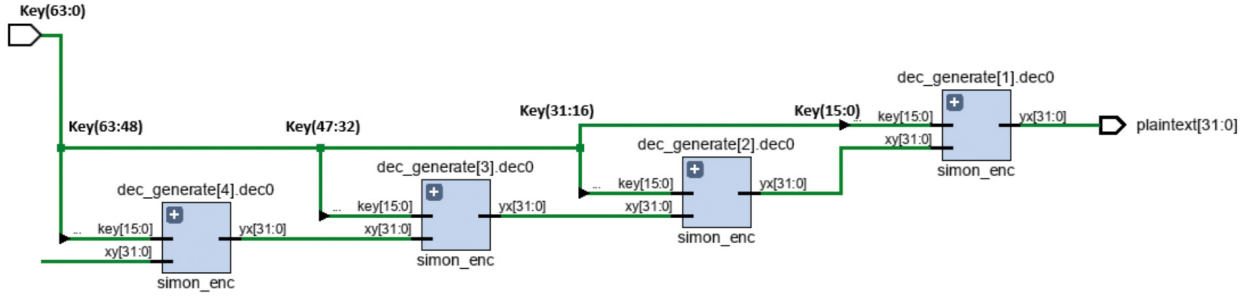
Source: own.

Figure 10. Connections for the first round of decoding.



Source: own.

Figure 11. Connections for the final rounds of decryption.



Source: own.

4. Results

To verify the operation of the cipher, a simulation file (*Test Bench*) was created using test vectors suggested in [13], values that are hexadecimal representations as follows:

Key: 1918 1110 0908 0100

Plaintext: 6565 6877

Ciphertext: c69b e9bb

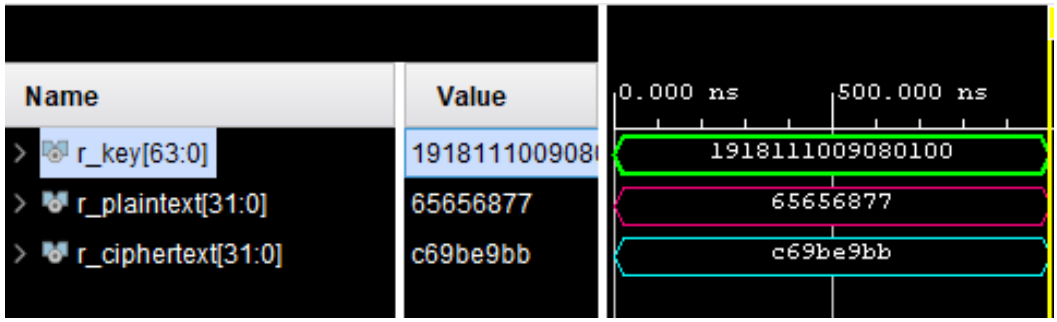
The simulation output data are shown in Figure 12, which shows that the suggested test values are obtained, verifying the correct operation of the cipher *SIMON*32/64.

The operation of the decryptor was verified similarly, obtaining the results shown in Figure 13, where the plain text is generated in an inverted manner.

Regarding the execution times for the encryptor, the simulator shows an average total delay time of 37.8 *ns* which implies a throughput of 846.56 Mbit/s. For the decrypter, the simulator shows an average total delay time of 54.1 *ns* which implies a throughput of 591.50 Mbit/s.

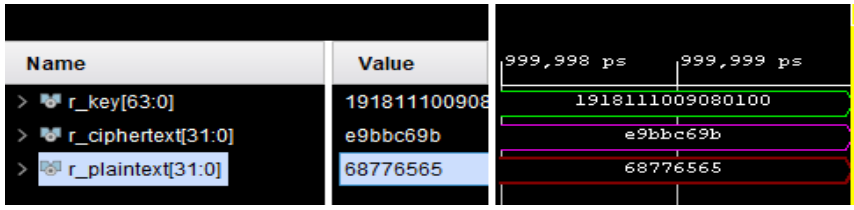
Table 1 shows a total of 976 *Slices* occupying 4.69% of the Artix-7 FPGA area for the cipher. It is observed that the design does not make use of registers, flip flops, and multiplexers, leaving space to implement other digital circuits.

Figure 12. SIMON32/64 cipher simulation data.



Source: own.

Figure 13. SIMON32/64 cipher simulation data decryptor.



Source: own.

Table 1. Slices are used in Artix-7 by the cipher.

Site Type	Used	Fixed	Available	Util%
Slice LUIs*	976	0	20800	4.69
LUT as Logic	976	0	20800	4.69
LUT as Memory	0	0	9600	0.00
Slice Registers	0	0	41600	0.00
Registers as Flip Flop	0	0	41600	0.00
Registers as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

Source: own.

Table 2 shows a total of 968 Slices occupied by the decryptor, being 4.65 % of the total area of the Artix-7 FPGA. Like the encryptor, it does not make use of registers, flip flops, or multiplexers.

Table 3 below shows some of the reported works and their area parameter used obtained in implementations of the cipher on an FPGA, contrasted with the results in the present work, taking into account that most of the implementations are designed in sequential form, that is why a lower area value is shown in some cases. As mentioned above, not many works report parallel implementations and do not show reports of the decryptor.

5. Conclusions

The implementation of the SIMON decryption block is a contribution of this work since most works focus on implementing encryption and round key generation. The encryption, key generation, and decryption systems were implemented in a purely combinational way, so it does not require the use of memory.

An important aspect of optimization is the reduced use of the area required to implement the SIMON32/64 cryptosystem since it only needs 976 and 968 slices for the encryptor and decryptor respectively. If we compare this amount of slices with the Artix-7, the cryptoprocessor occupation is of the order of 10% of the total area.

Table 2. Slices are used in Artix-7 by the decryptor.

Site Type	Used	Fixed	Available	Util%
Slice LUIs*	968	0	20800	4.65
LUT as Logic	968	0	20800	4.65
LUT as Memory	0	0	9600	0.00
Slice Registers	0	0	41600	0.00
Registers as Flip Flop	0	0	41600	0.00
Registers as Latch	0	0	41600	0.00
F7 Muxes	0	0	16300	0.00
F8 Muxes	0	0	8150	0.00

Source: own.

Table 3. Comparison of encryption with other implementations.

Author	Area (Slices)	Encryption	Platform
[9]	523	SIMON32/64	ARM SAGE-X v2.0
[19]	1404	SIMON96/96	SPARTAN 3E
[19]	434	SIMON96/96	VIRTEX-7
[11]	2057	SIMON32/64	Altera FPGA
[10]	960	SIMON32/64	Virtex 6
[12]	167	SIMON128/128	Spartan-3 xc3s50
[13]	139	SIMON48/96	Zynq
***	976	SIMON32/64	Artix-7
*** Results obtained in this article			

Source: own.

For the critical paths, an average delay of 37.8 *ns* was obtained for the cipher and 54.1 *ns* for the decryptor. These times allow the system to be used for real-time applications such as, for example, voice encryption.

Acknowledgments

To the General Direction of Research of the Universidad de Los Llanos for the support and funding of project C01-F02-010-2019 of the Macrypt research group that allowed the development of this work.

References

[1] M. Willett, "Lessons of the SolarWinds Hack", Survival (Lond)., vol. 63, no. 2, 2021. https://doi.org/10.1080/00396338.2021.1906001

[2] H. S. Lallie et al, "Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic", Comput. Secur. vol. 105, 2021. https://doi.org/10.1016/j.cose.2021.102248

[3] J. Aguirre, "Curso de seguridad informática y criptografía", vol. 3.1. 2003.

[4] E. Biham, A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", J. Cryptol., vol. 4, no. 1, 1991. https://doi.org/10.1007/BF00630563

[5] J. Daemen, V. Rijmen, "AES proposal: Rijndael", no. december, 1999.

[6] N. Velasquez, N. Pineda, "Design and Implementation of an AES-Rijndael Cryptoprocessor Prototype on FPGA", Universidad de Los Llanos, 2007.

[7] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, "PRESENT: An Ultra-Lightweight Block Cipher"

[8] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The LED block cipher", in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2011. https://doi.org/10.1007/978-3-642-23951-9_22

[9] F. Velásquez J. F. Castaño, "Cryptographic Implementations for Fpga", Rev. Visión Electron, vol. 5, no. 1, pp. 26-37, 2011.

[10] R. A. González Bustamante, R. Ferro Escobar, H. Vacca González, "Smart cities in collaboration with the internet of things, Vis. Electron., vol. 14, no. 2, pp. 185–195, 2020. https://doi.org/10.14483/22484728.16995

[11] P. D. Bonilla Nieto, J. S. Carrillo Sanabria, J. R. Camargo López, "Solar energy manager with PSOC5LP", Vis. Electron., vol. 13, no. 1, pp. 112–122, 2019. https://doi.org/10.14483/22484728.14426

[12] C. A. González González, F. Arévalo Tapias, J. Hernández Gutiérrez, "Análisis de seguridad en redes LPWAN para dispositivos IoT", Rev. Vínculos, vol. 16, no. 2, pp. 252–261, 2019. https://doi.org/10.14483/2322939X.15712

[13] L. W. Ray Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, "the Simon and Speck families of lightweight block ciphers", Natl. Secur. Agency, p. 42, 2013.

[14] P. Maene, I. Verbauwhede, "Single-cycle implementations of block ciphers", Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9542, pp. 131-147, 2016. https://doi.org/10.1007/978-3-319-29078-2_8

[15] S. Abed, R. Jaffal, B. J. Mohd, M. Alshayeji, "FPGA modeling and optimization of a SIMON lightweight block cipher", Sensors (Switzerland), vol. 19, no. 4, 2019. https://doi.org/10.3390/s19040913

[16] A. Shahverdi, M. Taha, T. Eisenbarth, "Lightweight Side Channel Resistance: Threshold Implementations of Simon", IEEE Trans. Comput. vol. 66, no. 4, pp. 661-671, 2017. https://doi.org/10.1109/TC.2016.2614504

[17] S. B. Basturk, C. E. Dancer, and T. McNally, "High-throughput Configurable SIMON Architecture for Flexible Security", Pharmacol. Res. p. 104743, 2020. https://doi.org/10.1016/j.mejo.2021.105085

[18] A. Muthumari et al., "High security for de-duplicated big data using optimal SIMON Cipher", Comput. Mater. Contin. vol. 67, no. 2, pp. 1863-1879, 2021. https://doi.org/10.32604/cmc.2021.013614

[19] W. Diehl, A. Abdulgadir, J. P. Kaps, K. Gaj, "Comparing the cost of protecting selected lightweight block ciphers against differential power analysis in low-cost FPGAs", Computers, vol. 7, no. 2, pp. 128-135, 2018. https://doi.org/10.3390/computers7020028