A Hardware Implementation of Simon Cryptography Algorithm

Soheil Feizi
Department of Electronics Eng.
Islamic Azad University of Science and Research
Kermanshah, Iran
s.f31391@yahoo.com

Arash Ahmadi
Department of Electronics Eng.
Islamic Azad University of Science and Research
Kermanshah, Iran
aahmadi@razi.ac.ir

Ali Nemati
Department of Electronics Eng.
Islamic Azad University of Science and Research
Kermanshah, Iran
ali_nemati_1989@yahoo.com

***Abstract*- Nowadays, the security of information exchange and communication such as authentication, confidentiality, and privacy are essentially important. Cryptographic algorithms are basic components of the security. Encryption algorithms are classified into various types such as block and stream ciphers. In constrained environments and embedded systems such as RFID, lightweight and low cost cryptographic algorithms are required. In June 2013 American National Security Agency (NSA) proposed a new block cipher family named Simon and declared that the aim of Simon design is to fill the gap for secure, flexible, and analyzable and to perform exceptionally well across the full spectrum of lightweight applications. In this paper, we analysis the Simon block cipher family and then simulate and synthesize this algorithm. Implementation results of the Simon cryptography algorithm on a FPGA are presented.**

Keywords: Lightweight Cryptography, Simon Algorithm, High Level Synthesis, Field Programmable Gate Array (FPGA).

I. INTRODUCTION

Radio Frequency Identification (RFID) is a ubiquitous wireless technology which lets objects to be identified automatically. This technology is envisaged to play a significant role in different applications such as labeling products and supply chain management, object tracking, patients monitoring in healthcare facilities and electronic passports. Exerting RFID tags can streamline many applications and provide many benefits but at the same time, privacy of the customers and security of the system should be taken into account against unwanted issues which may arise from using the new technology. Some potential customers are currently reluctant to use RFID embedded products because of the privacy issues [6].

During last decade, lightweight cryptography has provided security solutions for highly constrained devices and embedded systems. A large number of research works have already been done in this domain, which have specifically achieved optimized block ciphers suitable for lightweight cryptographic applications, for instance KATAN, PICOLLO and several others [2-3]. As RFID devices become broadly utilized in different applications, it is vital to provide various levels of security, associating to the application. The problem of devising appropriate cryptographic algorithms for constrained devices has been widely examined. The main approach to this issue is to investigate different techniques to decrease the number of gates required on the devices to implement alterations of the standard algorithms, while not impacting too much on the algorithm security [6].

Different applications have different security requirements, and in any case to achieve the desired level of security, protocols should be developed. Each block cipher is adaptively to initial cryptographic principles and we expect that any lightweight protocol can be based on a block cipher with the suitable data block and key size.

Although some standard cryptographic algorithms such as Advance Encryption Standard (AES) have been proposed for lightweight applications, choosing AES is not a good option for highly constrained environments [9]. Only few number of the block ciphers can be implemented on a small circuit with minimal power and area requirements and some others can be embedded on microcontrollers with limited flash, SRAM, and/or power availability [1]. Obviously, any effort to optimize the performance of a parameter on one side will have a negative impact on the performance of other parameters on the other side meaning that the tradeoff law is in place.

In 2013 NSA proposed a new family of highly optimized block cipher Simon that has flexibility and superior performance both in hardware and software environments especially on hardware devices. The Simon algorithm has a variety of data blocks and key sizes which can be used for different implementations, thus, the users can coordinate security requirements and specific applications with algorithm. To increase the flexibility, The NSA's experts have designed Simon block cipher family with several different block and key sizes, which provides 10 algorithms as shown in Table I. Each of these algorithms have been optimized for specific applications, for example block sizes of 48 or 96 bits are optimal for some electronic product code (EPC) applications [1].

Security has a direct relation with key size, meaning that with the key-length increase, the level of security grows.

978-1-4799-5487-2/14/$31.00 ©2014 IEEE

TABLE I. SIMON BLOCK AND KEY SIZE

Block Size	Key Size
32	64
48	72,96
64	96,128
96	96,144
128	128,192,256

On the other hand, one should choose appropriate block and key sizes to match the required level of security for the application.

II. SIMON ALGORITHM DESCRIPTION

The Simon block cipher with $2n$-bit block and mn-bit key is denoted Simon$2n/mn$ where n must be 16, 24, 32, 48, or 64 and m must be 2, 3 or 4. For example, Simon32/64 refers to the version of Simon acting on 32-bit plain text blocks and using a 64-bit key. This algorithm uses Feistel network structure [1].

For Simon$2n$ encryption it is required to use three operations on n-bit words:

1. XOR
2. AND
3. Left and Right Circular Shifts, S^j and S^{-j} by j bits, respectively.

For $k \in GF(2)^n$, the Simon$2n$ round function is $R_k: GF(2)^n \times GF(2)^n \rightarrow GF(2)^n \times GF(2)^n$ defined by

$$R_k(x, y) = (y \oplus f(x) \oplus k, x) \quad (1)$$

where $f(x) = (Sx \,\&\, S^8x) \oplus S^2x$ and k is the round key [1]. The inverse of the round function, used for decryption, is

$$R^{-1}_k(x, y) = (y, x \oplus f(y) \oplus k) \quad (2)$$

The encryption map is the composition $R_{k(T-1)} \circ \ldots \circ R_{k(1)} \circ R_{k(0)}$, where T is the number of rounds. In Fig. 1, we observe the Simon round function $R_{k(i)}$ on the two sub-cipher (x_{i+1}, x_i) at the i^{th} step of the process [1].

The Simon key schedules use one of the five sequence z_j (j = 0 to 4) with this aim that to make separation between different versions of Simon having the same block size. Therefore, the Simon algorithm's designers provide these sequences z_0, z_1 with period 31 and z_2, z_3, z_4 with period 62 [1]. Z_0 is defined below:

$z_0 = (z_0)_0 (z_0)_1 (z_0)_2 \ldots =$
11111010001001010110000111001 10…
where $(z_j)_i$ is the i^{th} bit of z_j.

The method of z_0 to z_4 sequences generation is given in [1].

Let $c = 2^n - 4 = (2^n - 1) \oplus 3 = 0xff \cdots fc$. For Simon$2n$ with m key words $(k_{m-1} \ldots k_1, k_0)$ and constant sequence z_j, round keys are generated by

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1} & \text{if } m = 2, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2} & \text{if } m = 3, \\ c \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}) & \text{if } m = 4, \end{cases} \quad (3)$$

for $0 < i < T - m$.

The key schedule for m=4 is shown in Fig. 2, and the version of constant sequence z_j is expressed in Table II. The key words k_0 to k_{m-1} are used as the first m round keys [1].

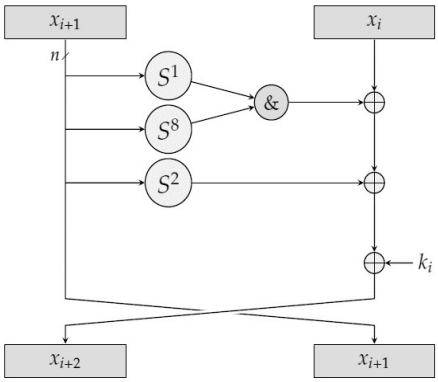

Figure 1. Simon round function [1]

The pseudo code of the Simon algorithm is shown below:

```
- - - - - - - - - - - - - Key expansion - - - - - - - - - - - - -
for i = m to T-1
TEMP <= S^-3 k[i-1]
if (m = 4) TEMP <= TEMP ⊕ k[i-3]
TEMP <= TEMP ⊕ S^-1 TEMP
k[i] <= k[i-m] ⊕ TEMP ⊕ z[j][(i-m) mod 62] ⊕ c
end for
- - - - - - - - - - - - - - - encryption - - - - - - - - - - - - - -
for i = 0 toT-1
TEMP <= x
x <= y ⊕ (Sx & S^8x) ⊕ S^2x ⊕ k[i]
y <= TEMP
end for
cipher <= x|y (attach x to y )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

TABLE II. SIMON PARAMETERS [1]

block size 2n	key size mn	word size n	key words m	const seq	rounds T
32	64	16	4	z_0	32
48	72	24	3	z_0	36
	96		4	z_1	36
64	96	32	3	z_2	42
	128		4	z_3	44
96	96	48	2	z_2	52
	144		3	z_3	54
128	128	64	2	z_2	68
	192		3	z_3	69
	256		4	z_4	72

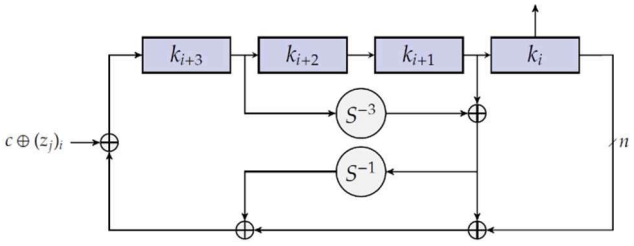

Figure 2. The Simon key expansion for m = 4 [1]

III. SYNTHESIS

Synthesis and circuit design can be divided into different phases. The circuit can be expressed at different levels of design. Level design is divided into the following sections: 1. System level, 2. Behavioral level, algorithmic level or high level, 3.Architecture or RTL level, 4.Logic level, 5. Circuit or transistor level, 6.Physical or layout level [5].

Here, we use the high-level (Behavioral level) synthesis. High-level synthesis consists of the following subsections:

1. Compilation
2. Transformation
3. Scheduling
4. Selection or Allocation
5. Binding or Assignment
6. Output Generation (hardware design with control unit and sequencer) [5].

To start the simulation and synthesis steps, we have chosen one of the Simon's algorithm (Simon 32/64). According to Table II, it is explicit that data block length = 32 bit, key length = 64 bit, m = 4, n = 16, T = 32 and $z_j = z_0$. Also, an example of this type of algorithm is considered as follows [1]:

Key: $(1918\ \ 1110\ \ 0908\ \ 0100)_{16}$

Plaintext: $(6565\ \ 6877)_{16}$

Ciphertext: $(c69b\ \ e9bb)_{16}$

At first, we have written and simulated HDL code of Simon32/64 algorithm for above example. Simulation results are shown in Fig. 4. Then, Data Flow Graph (DFG) for the algorithm is drawn and changes have been applied such a way that each operation round of the algorithm is done in 5 Control Step (CS). Also, Sub-operations have been located in such a way that in each control step at most two ALUs, two shift-registers and one register are required.

Allocation is further divided into sub-tasks, such as functional unit allocation, memory unit allocation and interconnect allocation. In required resource allocation, we assigned the number of functional units of each type for performing operations, memory units (registers) for storing data values and interconnects for data transportation. Resource allocation and binding may share resources so that the same hardware can be used to execute different operations. Finally optimal state for scheduling, allocation and binding is shown in Fig. 5.

The proposed trend of the Simon algorithm execution is depicted in flow chart Fig. 3.

Similarly, scheduling and allocation and binding can be repeated for m=2 and 3, however, it is easier compared with m = 4. In Fig. 5, j<< means a circular shift to left by j bits and j>> means a circular shift to right by j bits. Another noteworthy point in Fig. 5 is that x_i is transferred to the y_{i+1} without change in CS5. According to the Simon 32/64, m is 4. So, 4 round keys are derived from main key. Therefore, the encryption is done 4 times from i=0 to 3. Obviously, the new round key must be generated when i=4 which is done in key expansion part. The key expansion part must be done for i=4 to 31 and the round key will be obtained for each i, then the encryption part must be done with the obtained round key. There are two ways ahead doing this task. One way is dividing the scheduling into seven control steps (as shown in Fig. 6), so that the generated key is used in the same encryption round and since the scheduling is done in 28 rounds and therefore whole encryption operation is performed in (4×7 + 28×7) = 224 CS. The other way is dividing the scheduling into five control steps based on which total rounds will be 29 so the whole operation will be (4×5 + 29×5) = 165 CS. It is clear that 29 rounds with five CS is more optimized in comparison with 28 rounds with seven CS in terms of time, given that resources are used in both cases are the same.

In next step, designed hardware to run the Simon algorithm is discussed as depicted in Fig. 7. This hardware includes ten Muxes, two ALUs, two registers and two shift-registers. Designed sequencer for this hardware is shown in Fig. 8. This sequencer includes two AND gates, one counter and a very simple controller.

According to the number of control steps, counter determines which outputs of the controller are '0' and which are '1'. Sequencer gives necessary instruction execution time of the hardware as presented in Table III.

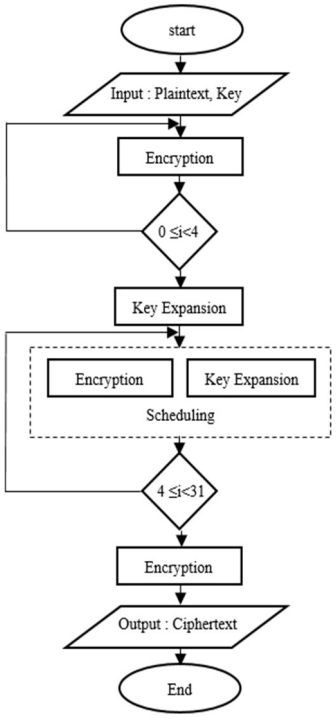

Figure 3. Proposed trend of the Simon algorithm

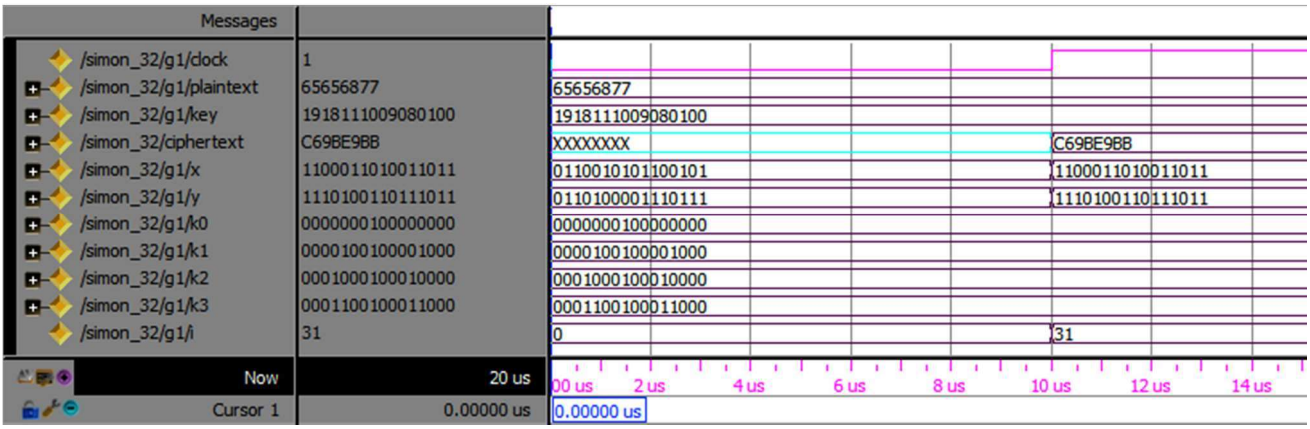

Figure 4. Simulation Results

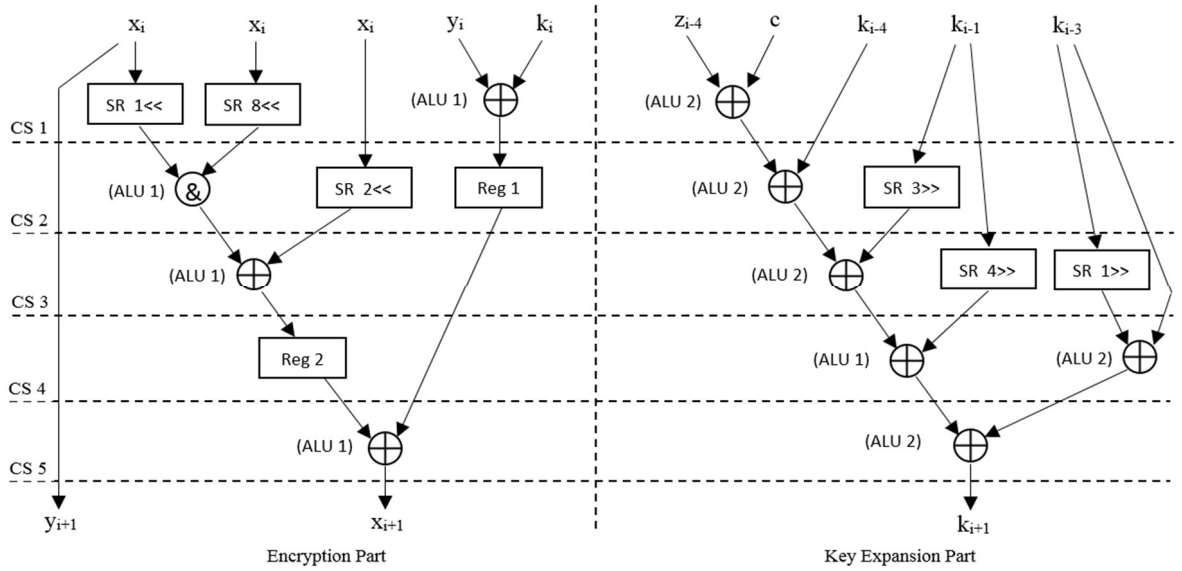

Figure 5. Scheduling, Allocation and Binding with 5 CS

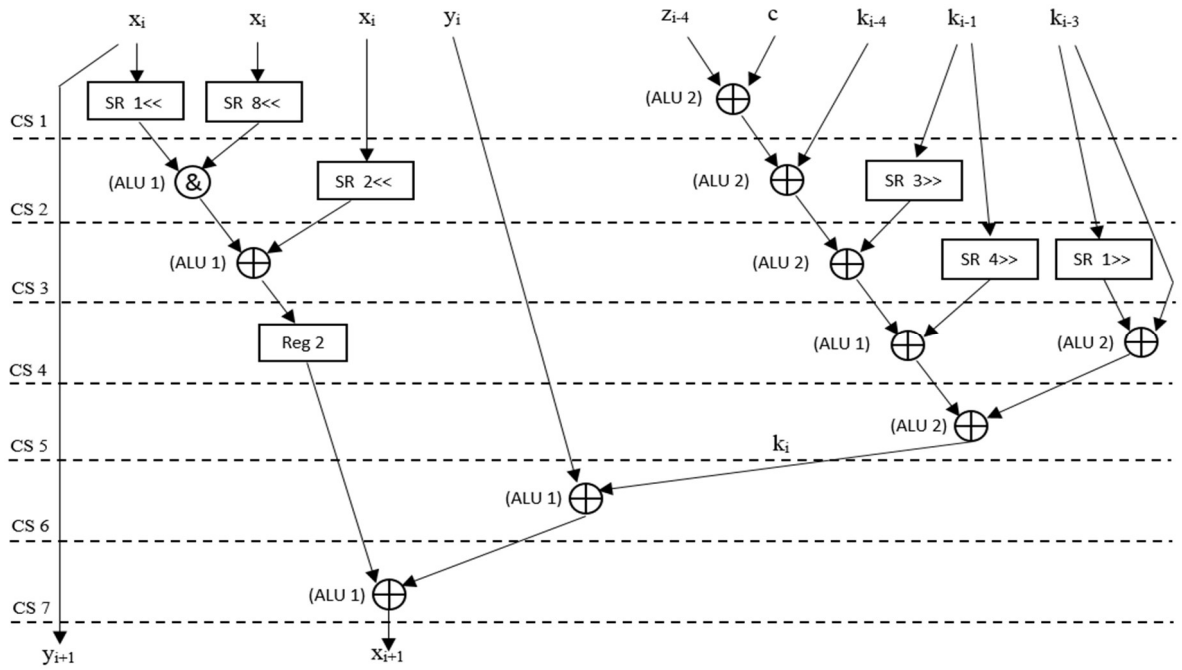

Figure 6. Scheduling, Allocation and Binding with 7 CS

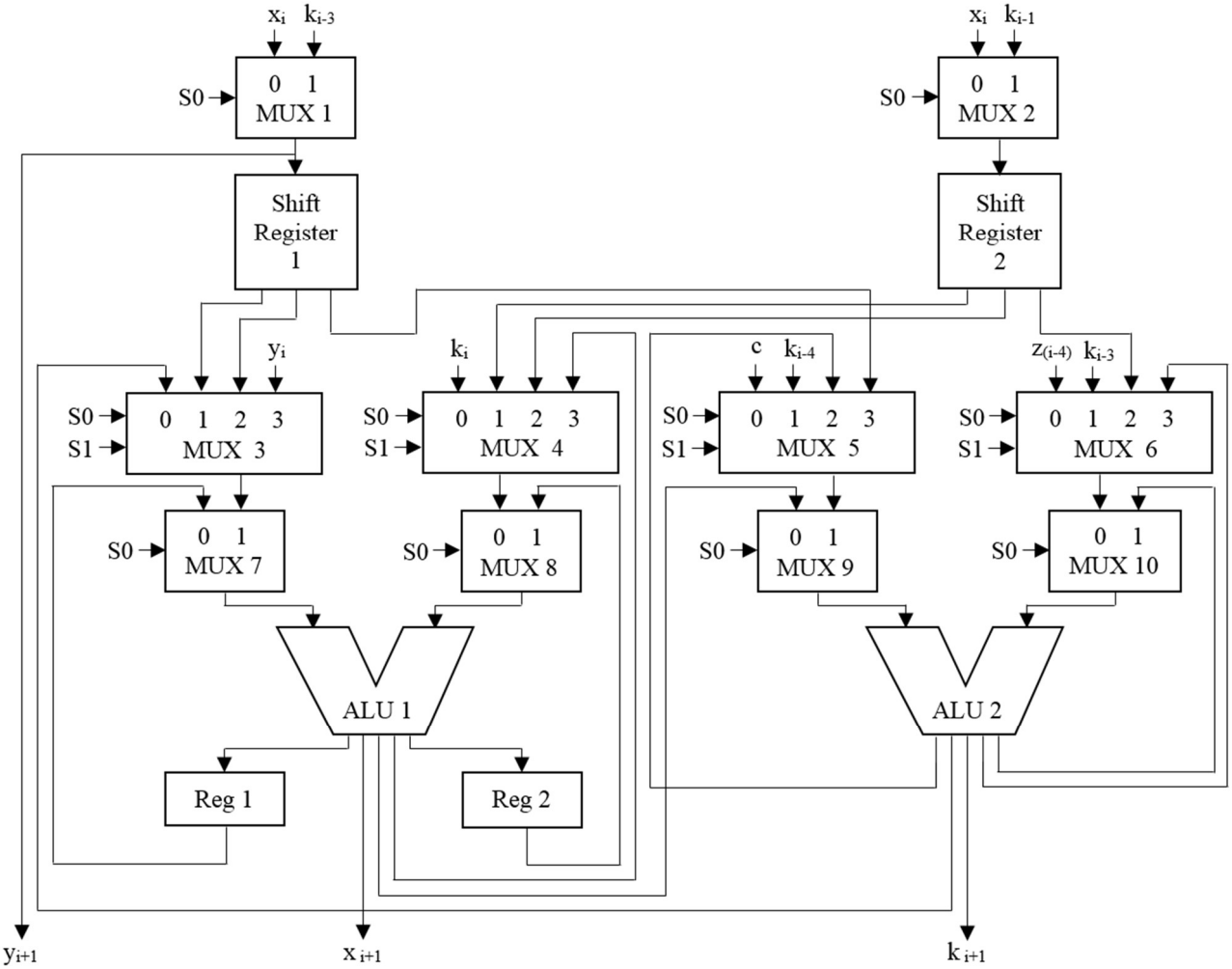

Figure 7. Hardware Design

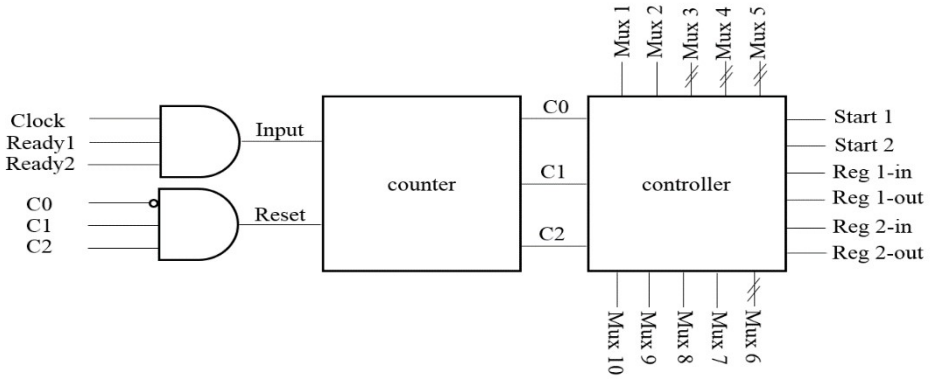

Figure 8. Sequencer Unit

TABLE III. SEQUENCER UNIT PARAMETERS

Control Step			Output Array																			
C2	C1	C0	Reg2 out	Reg2 in	Reg1 out	Reg1 in	Start 2	Start 1	Mux 10	Mux 9	Mux 8	Mux 7	Mux 6 S1	Mux 6 S0	Mux 5 S1	Mux 5 S0	Mux 4 S1	Mux 4 S0	Mux 3 S1	Mux 3 S0	Mux 2	Mux 1
0	0	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	1	0	0
0	1	0	0	0	0	1	1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	0
0	1	1	0	0	0	0	1	1	0	1	0	1	0	1	1	0	1	1	1	0	1	1
1	0	0	0	1	0	0	1	1	0	1	0	1	1	0	1	1	1	0	0	0	x	x
1	0	1	1	0	1	0	1	1	1	0	1	0	x	x	x	x	x	x	x	x	x	1
1	1	0	Reset																			

IV. IMPLEMENTATION

The synthesizer cad tool is an important tool that prepares the post synthesis analysis and produces the hardware platform. After executing the syntax checking and RTL or technology design from the synthesizer cad tool, the software fetch the code directly. The Xilinx synthesis technology synthesizes VHDL code to make Xilinx files with .ngr and .ngc extensions after the functional simulation. [8,22]. We implemented the Simon32/64 algorithm using Xilinx ISE development tools on FPGA model Virtex-5 XC5VFX200T. The results and resource utilization of the design is summarized in Fig. 9. Also synthesis estimation is displayed in Fig. 10.

Some comparisons on utilization between Simon, Hummingbird and X-TEA cryptography algorithms is depicted in Table IV to show its efficiency for resource limited hardware.

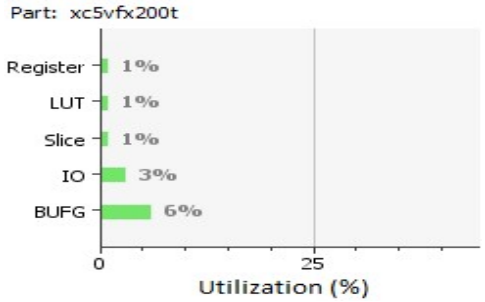

Figure 9. Implementation Result

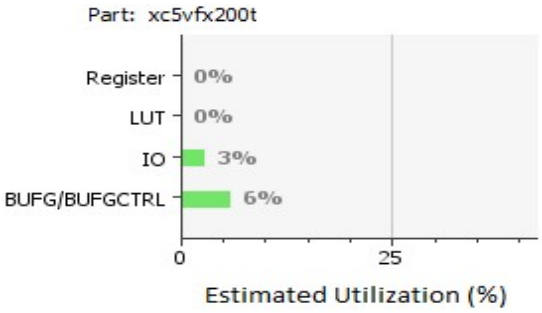

Figure 10. Synthesis Estimation Result

TABLE IV. COMPARISONS ON UTILIZATION

Algorithm	IO	LUT	Reg	BUFG	Max Freq. (MHz)
HB [22]	278	3504	4242	8	152.905
X-TEA [12]	133	634	307	–	120.3
This work	32	177	182	2	523.56

V. CONCLUSION

Simon's algorithm is a lightweight flexible algorithm for using in RFID systems and wireless sensor networks (WSNs) and deployment capabilities in many other applications as well. Due to the flexibility of the algorithm used in various applications and environments, data and key length can be selected with different sizes. In Simon 32/64, data length and key length are small and also, due to having too few resources as shown in our implementation results, it is very suitable for lightweight application and embedded systems.

REFERENCE

[1] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, Louis Wingers, "The simon and speck of lightweight block ciphers," National Security Agency 9800 Savage Road, Fort Meade, MD 20755, USA, June 2013.

[2] C. D. Cannière, O. Dunkelman, and M. Kneževi´c, "KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers,"Springer-Verlag, 2009 [In CHES 2009, Lecture Notes in Computer Science No. 5747, pages 272–88.]

[3] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai, "Piccolo: An Ultra-Lightweight Blockcipher," Sony Corporation 1-7-1 Konan, Minato-ku, Tokyo 108-0075, Japan, 2011.

[4] M. Zwolinski, "Digital System Degin with VHDL,"2nd Edition, 2004.

[5] Giovanni De Micheli, "Synthesis and optimization of digital circuits," 1994.

[6] Ehsan Vahedi,Vincent W.S. Wong, Ian F. Blake, "An Overview of Cryptography," Chapter 5, University of British Columbia, Canada, 2014.

[7] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, "Handbook of Applied Cryptography", CRC Press, August 2001, pp. 202-203.

[8] Xilinx, Inc., Virtex-5 User Guide, available at www.xilinx.com.

[9] Martin Feldhofer, Sandra Dominikus and Johannes Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm," Proc. Workshop on Cryptographic Hardware and Embedded Systems - CHES 2004, Cambridge (Boston), Ma., USA, Springer, vol. 3156, August 2004, pp. 357-370.

[10] Thomas Eisenbarth, Sandeep Kumar, Christof Paar, Axel Poschmann and Leif Uhsadel, "A Survey of Lightweight Cryptography Implementations," IEEE Design & Test of Computers, vol. 24, no. 6, November 2007, pp. 522-533.

[11] Paris Kitsos, Nicolas Sklavos, Maria Parousi, Athanassios N. Skodras, "A comparative study of hardware architectures for lightweight block ciphers", Elsevier Journal, Computers and Electrical Engineering 38 (2012) 148–160.

[12] Issam Damaj, Samer Hamade and Hassan Diab, "Efficient Tiny Hardware Cipher under Verilog", Proceedings of the 2008 High Performance Computing & Simulation Conference.

[13] FIPS, "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197.

[14] FIPS, "Data Encryption Standard", Federal Information Processing Standards Publication 46.

[15] Mi Lu, "Arithmetic and Logic in Computer Systems", Texas A&M University, Published by John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

[16] M. Hell, T. Johansson, A. Maximov, W. Meier, "The Grain family of stream ciphers", New Stream Cipher Designs - The eSTREAM Finalists, LNCS 4986, pp. 179–190, Springer, 2008.

[17] C. D. Canniere, B. Preneel, "Trivium", New Stream Cipher Designs – The eSTREAM Finalists, LNCS 4986, pp. 244–266, Springer, 2008.

[18] T. Isobe, "A single-key attack on the full GOST block cipher", FSE'11 , LNCS 6733, pp. 290–305, 2011.

[19] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, S. Chee, "HIGHT: A new block cipher suitable for low-resource device", CHES'06 , LNCS 4249, pp. 46–59, Springer-Verlag, 2006.

[20] G. Leander, C. Paar, A. Poschmann, K. Schramm, "New lightweight DES variants", FSE'07 , LNCS 4953, pp. 196–210, Springer-Verlag, 2007.

[21] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher", CHES'07 , LNCS 4727, pp. 450–466, Springer-Verlag, 2007.

[22] Nikita Arora, Yogita Gigras, "FPGA Implementation of Low Power and High Speed Hummingbird Cryptographic Algorithm", International Journal of Computer Applications (0975 – 8887) Volume 92 – No.16, April 2014.