Received: August 12, 2021. Revised: September 15, 2021. 507

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

# Low Area FPGA Implementation of SP-box PRESENT Cryptography with MS-CLCG Key Generation

**Srikanth Parikibandla[1]*** **Sreenivas Alluri[1]**

[1]*Department of Electronics and Communication Engineering,*
*GITAM Deemed to be University, Visakhaptnam, India*
* Corresponding author's Email: psrikanth.bt@gmail.com

**Abstract:** Nowadays, the Lightweight cryptography algorithm is mostly used for security level purposes, because it's designed with the help of basic arithmetic logic blocks. The basic blocks are perfectly placed in respective modules to process the encryption with effective manner. The fundamental shifting based PRESENT algorithms are frequently used to help the hackers to retrieve the data. To overcome this issue, Separation With NOT (SWN) with Srikanth Parikibandla box (SP-box) architecture is used in this research work. Due to the usage of modified Register Transfer Level (RTL) structure, the process and the confidentiality of the PRESENT encryption architecture is improved in a significant manner. The Modified Single Coupled Linear Congruential Generator (MS-CLCG) architecture is used to generate the random number, which act as a key for encryption as well as decryption. For every clock cycle, MS-CLCG architecture generates the random value to perform the encryption. The SWN-SP architecture occupied 56 counts of LUTs, 33 count of Flip-flops and 30 counts of slices, which are less compared to the existing SRS-PRESENT architecture.

**Keywords:** Lightweight cryptography, Modified single coupled linear congruential generator, PRESENT algorithms, Separation with not and Srikanth Parikibandla box.

## 1. Introduction

In modern days, the utilization of computerized information is increasing day by day in various applications such as industrial applications, medical applications etc. The security of data becomes the major concern, when the information is transmitted over the networks. Cryptography is an emerging technology which ensures security as well as authentication to the data by encryption and decryption operations. In encryption and decryption operation, the data are converted into cipher text and vice versa by the key scheduling algorithm [1-3]. The ciphers are categorized into two types based on the number of key generations such as asymmetric cipher and symmetric cipher. The asymmetric cipher uses two keys (Private and public keys) and symmetric cipher uses one shared private key for both encryption and decryption process [4]. The applications such as Radio-frequency Identification (RFID), Wireless Nano sensors and Internet of Things (IoT) are mainly focused on security as well as resource utilization. The Lightweight ciphers with symmetric cipher play a vital role in resource constrain applications [5]. The different types of light weight ciphers are MIDORI cipher [6], Advanced Encryption Standard (AES) [7], SIMON, SPECK, PRESENT and KHUDRA [8], etc.

This research is mainly focused on the PRESENT lightweight block cipher to attain resource efficient hardware for IoT applications. The PRESENT cipher is standardized by ISO\IEC in 2012 and this cipher is based on Substitution Permutation Network (SNP) with a key length of 80/128 bits and support 64-bit input data [9]. The processing system of the PRESENT cipher performs in round-based method which takes 31 rounds to run. The three primary operations in each round are AddRoundkey, Substitution layer (S-box) and Permutation layer (P-box) [10]. This research

extends based on the comparative analysis of the previous works to improve the resource efficiency and security. Nedjah and Macé, implemented the designs of AES and Scalable Encryption Algorithm (SEA) on the FPGA platform and these architectures gives better performance in terms of resource constrain but the throughput efficiency is considerably low [11, 12]. The KATAN cipher architecture [13] was synthesized in FPGA to evaluate the performance in terms of block size, a number of rounds and variable keys. In [14], Kaur and Sakthivel proposed Hamming bird algorithm which was implemented in both FPGA and ASIC to evaluate the area efficiency. In [15] Nino proposed a novel architecture for PRESENT cipher which achieved low cost hardware, but it failed maintain the energy consumption. This SWN-SP is developed to overcome the limitations of existing PRESENT cipher architectures. The major contribution of the proposed method is given as follows,

- Modified PRESENT architecture with various key generation helps to improve the system security and the FPGA performances.
- SWN with SP-box architecture is designed to change the structure of basic PRESENT architecture.
- MS-CLCG architecture provides the random numbers for every clock cycle to perform the encryption. Due the randomness, the confidential data is not taken by the hackers.
- Conventional PRESENT architecture required more registers to store the intermediate data. But, there is no need of registers in the proposed work. So, the LUT, flip flops, and slices counts are reduced in the proposed work.

The organization of the paper is given as follows: The literature survey is provided in Section 2, the problem statement of the PRESENT architecture is given in Section 3, the SWN-SP architecture explanation along with MS-CLGG key generation are explained in Section 4. The results and discussion of the proposed method are explained in Section 5. The conclusion and future scope are provided in Section 5.

## 2. Literature review

Haider [16] proposed a Low Cost Self-Test (LC-ST) architecture which integrated with PRESENT cipher to achieve low cost hardware components. The light weight block PRESENT cipher was implemented with self-test ability to test the pattern generation and the X-compactor to test the response. This architecture was implemented on different Xilinx FPGAs to evaluate the performance of an area and throughput. The experimental results showed that this self-test architecture occupied only 23% of area because it reduced the required additional resources to test pattern generation. This proposed architecture achieved high throughput of 14% and improved fault coverage than the other self-testing designs. But, the process of FSM occupied more LUTs (304) and it required 32 round of latency which increases the system complexity.

Rashidi [17] proposed the HIGHT and PRESENT lightweight block cipher to achieve high throughput and low cost hardware. The $2^8$ module of the HIGHT algorithm gives low critical path delay when it implemented with parallel prefix adder. The PRESENT cipher with S-box structure and loop unrolling technique was used to reduce the logic gate counts and latency respectively. The glitch filter was provided to reduce the power consumption of this architecture. The Virtex-5 and Spartan-3 FPGAs evaluated the performance, such as throughput, execution time and area. However, the proposed architecture occupied 10 registers and 8 F functions to store the intermediate data which caused more Slices (206).

Sravya [18] proposed an ultra-lightweight design of PRESENT block cipher for improved Security and hardware planning. In this paper, 64bit plain text and 80bit cipher texts was considered. Mainly this paper focused on reducing the size of the substitution box (i.e S-box) in which the operation was carried out in 4X2 blocks where each block was divided with 8 bits. As a result, it reduced the round of operations to 16 which in turn reduced the time taken to produce the output by 1.310ns and also utilized a minimum 1% area. Even though the speed of the operation was increased, in the timing report of 8 bit S-box it requires more time period of about 2.202ns when compared to 4 bit S-box of about 1.746ns.

Pandey [19] proposed architecture based on a PRESENT Light Weight Block Cipher (LW-BC) to achieve high efficiency hardware components. The on-the-fly (OTF) architecture was implemented with 80/128-bits key length of PRESENT cipher which compute the intermediate keys and perform encryption and decryption operation. The Xilinx Virtex-5 FPGA was used to evaluate the performance of this architecture such as energy consumption, maximum frequency, and power dissipation. The experimental results showed that this architecture achieved high throughput of 20Mbps per slice at the clock frequency of 458MHz. Two BRAM has been designed to perform the basic key scheduling process which occupied more LUT

count (281). Due to the usage of BRAM, the static power of the entire architecture was increased to 1043mW.

Patranabis [20] proposed the implementation of the PRESENT cipher with security strategies to achieve better security by resisting the Light weight Side Channel and Fault Attacks (LW-SCFA). The fault space transformation was used to resist the fault in linear layers. The S-box structure with modified transparency order and cipher-dependent multi round shuffling were used to refresh the mask periodically and provide better security. Due to the manual key generation process, the control unit required more number of LUTs (263) and the key values were not updated automatically for every clock cycle.

## 3. Problem statement

In recent days, security is one of the major processes to secure the confidential data from the unauthorized persons and the following problems have been occurred in encryption standards.

- The basic PRESENT architecture and it is a fundamental operation helps to the hackers to get the idea of the architecture. If the hackers know the process of the basic PRESENT architecture, they can retrieve the original data easily [19].
- Normally, the key is mandatory for performing the encryption and decryption process. During the PRESENT architecture encryption, the key values are given by manually, which is too difficult for all the input samples [20].
- Key scheduling process is easy to hack due to the usage of standard RTL module structure.

**Solution:**

To overcome the aforementioned problems, separation with NOT gate based SP-box is used in this research work. SP-box act as a confusion property which helps to secure the data from the adversary. Because, user only defined the images which pixel values are stored in the SP-box. So, it is too difficult to identify the stored values which improve the system security. This proposed RTL helps to design the new PRESENT architecture model. Due to the modification in the PRESENT architecture, it is too difficult to identify the architecture flow. Moreover, the MS-CLGG architecture is used to generate the random number, which is acting as a key for encryption as well as decryption. The random number has generated for every clock cycle as well as the encryption has performed for every clock cycle. With the help of SP-box, the key scheduling architecture is designed to perform the cryptography operation. The detailed explanation of the SWN-SP is given in the following section.

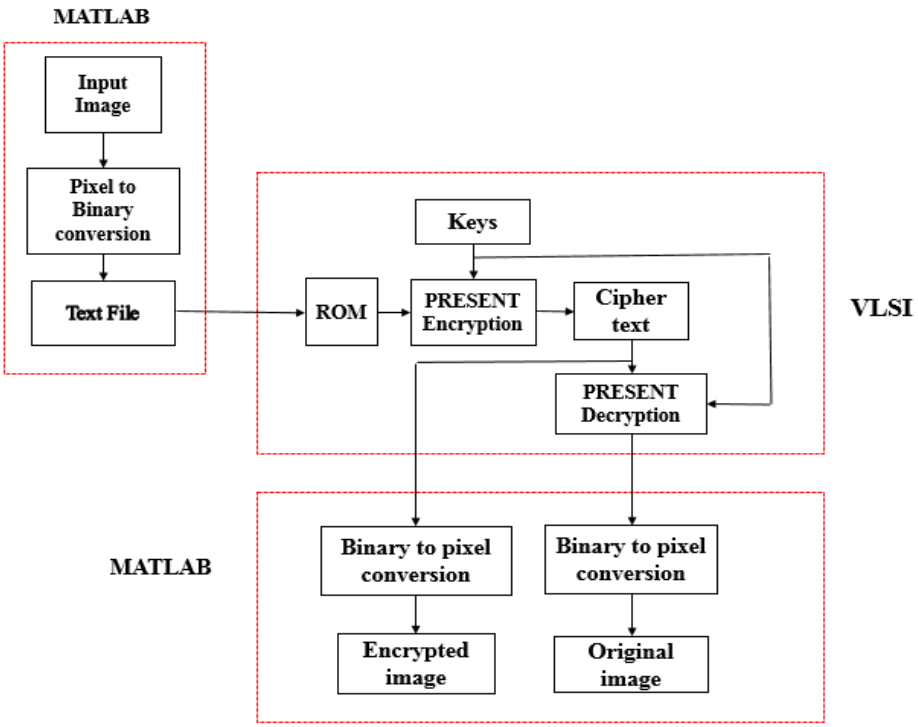



Figure. 1 Overall process of the encryption and decryption

## 4. SWN-SP architecture

The main objective of this SWN-SP method is to improve the security level with unpredictable key generation. The digital image is taken as an input and then the pixels of the image are converted into binary format by the MATLAB tool. The stored binary conversions in the text file are read by the Read Only Memory (ROM). The PRESENT encryption algorithm is used to encrypt the binary format with the generation of keys. The encrypted keys are in the form of cipher text and the decryption operation is performed with the help of same keys which are used for encryption. The binary format from the decryption process is converted into pixels to get the original image and encrypted image is also acquired from the cipher text. The following Fig. 1 shows the overall process of the encryption and decryption.

The 16-bits plaintext is truncated as 4-bits which is given as input to the Multiplexers when the counter value is '0'. The multiplexer output is processed by the Separation with NOT and Srikanth Parikibandla box (SP-Box) which is concatenate as 16-bits. The keys are generated by the Modified Single Coupled Linear Congruential Generator (MS-CLCG) and it is given to the modified key schedule. The XOR operation is performed in the 16-bits SP-Box and modified key schedule output. Then the S-Box and P-Box process the XOR output and given as input to the multiplexers when the counter value is changed to '1'. The multiplexer input is selected based on the counter value which is designed from '0' to '3'. The Proposed PRESENT cipher architecture takes four rounds to generate cipher text for a 16-bits plaintext which is given in the Fig. 2 [15].

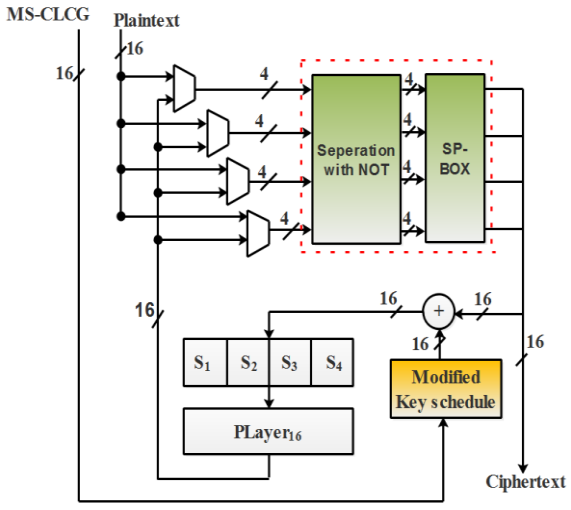



Figure. 2 Proposed architecture of the PRESENT cipher

The PRESENT cipher architecture is modified to improve the security level by the use of Separation with NOT and SP-Box designs. In Separation with NOT, the outputs of multiplexer are truncated and inverted by the inverter. This inverted output is concatenated, which generated the modified 4-bits. The SP-Box contains the default value of the author's 4X4 image pixel value. The architecture of Separation with NOT and SP-Box is given in Fig. 3.

The input of the SWN module is given in Eq. (1),

$$a\,[3:0] = \{a[3], a[2], a[1], a[0]\} \tag{1}$$

where, a [3:0] – Input of the SWN module,
a [0] – 1st LSB bit of SWN input,
a [1] – 2nd LSB bit of SWN input,
a [2] – 3rd LSB bit of SWN input,
a [3] – 4th LSB bit of SWN input.

These inputs are performing the NOT operation which are given in the following Eqs. (2) to (5),

$$I_0 = \sim a[0] \tag{2}$$

$$I_1 = \sim a[1] \tag{3}$$

$$I_2 = \sim a[2] \tag{4}$$

$$I_3 = \sim a[3] \tag{5}$$

where, $I_0, I_1, I_2, I_3$ – Inverted output of 4 bit.

All the inverted outputs are performed the concatenation operation and the results are given in Eq. (6),

$$I_{out} = \{I_3, I_2, I_1, I_0\} \tag{6}$$

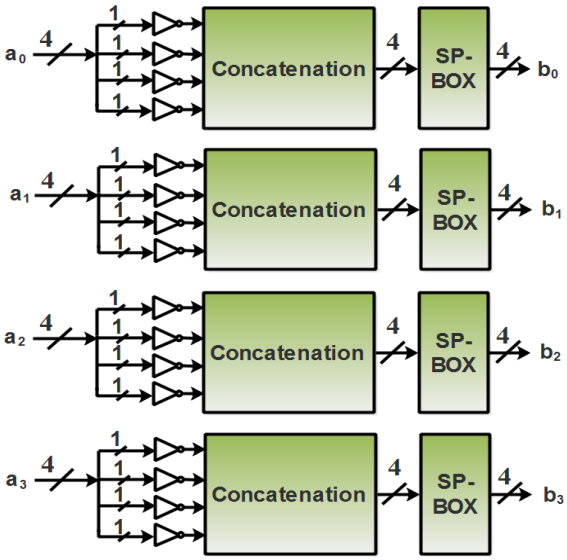



Figure. 3 Architecture of separation with NOT and SP-box

Figure. 4 SP-box image

Table 1. SP-box values

| **Input** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| Output | F | 7 | E | 6 | D | 5 | C | 4 |
| **Input** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| Output | B | 3 | A | 2 | 9 | 1 | 8 | 0 |

where, $I_{out}$ – Concatenated Inverted output.

This $I_{out}$ values are performed the SP box operation which explanation is given as follows:

The SP-box image is shown in Fig. 4 which is read in MATLAB software. In MATLAB software, the image size is represented as 3X3. This size of the image pixel values is performed the "dec2bin" operation to convert the pixel values into binary values. These binary values are helping to generate the SP-box module. The SP-box values are given in Table 1.

The previous stage $I_{out}$ value is connected to the input of the SP-box and previous stage output and SP-box inputs are designed to perform the 4 bits of data. Based on the input values, the SP-box creates the 4 bit of output data which is represented as the output of SWN module architecture. The output of SWN is given in Eq. (7),

$$SWT_{out} = SPbox\,(I_{out}, SP_{out}) \tag{7}$$

where, $I_{out}$ – input of the SP-box,

$SP_{out}$ – Output of the SP-box,

$SWT_{out}$ – Output of the SWT module

The SP-box has designed inversely in the process of decryption. For decryption, this modified architecture is implemented in reverse order. In this research work, the symmetric key process is implemented to secure the data. For performing decryption, the same key is used in encryption side. The confusion property plays an important role in the SP-box. The process of MS-CLCG key generation is shown in Fig. 5. Initially, the value X is generated randomly to the connected next logical block MUX. Counter is used to choose the selection line for the MUX logical block.

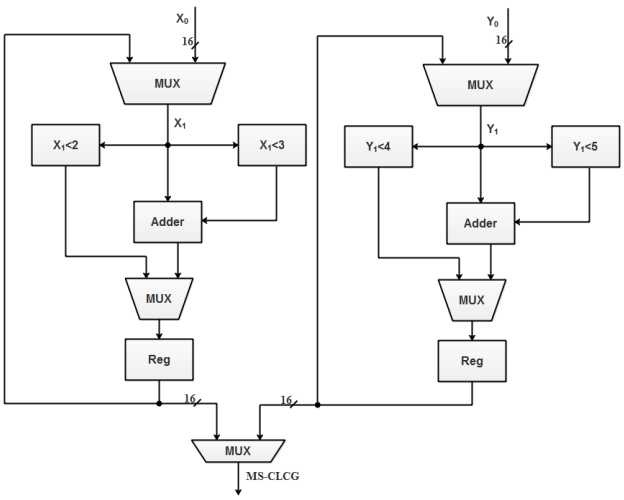



Figure. 5 Architecture of MS-CLCG

If the count value is zero, the selection line "0" is chosen by MUX as well as if the count value is one, the selection line "1" is selected by MUX. The initial MUX output is given in Eq. (8) and this MUX output is going to perform the shifting operation (<<2, <<3) and these outputs are given in Eqs. (9) and (10),

$$X\_MUX1_{out} = sel\,?\,X :\ X\_MUX2_{out} \tag{8}$$

$$X\_Shift_1 = X\_MUX1_{out} \ll 2 \tag{9}$$

$$X\_Shift_2 = X\_MUX1_{out} \ll 3 \tag{10}$$

where, $X\_MUX1_{out}$– Output of the MUX1,

sel - Selection line; X – Random number; $X\_MUX2_{out}$ – Output of the MUX2; $X\_Shift_1$ – Left shift output by 2; $X\_Shift_2$ – Left shift output by 3.

The shift output is performed the addition operation with initial stage of the MUX output value. Based on the selection line (sel), the second MUX is used to select the addition results or shift1 results in the output terminal. The addition and MUX2 outputs are given in Eqs. (11) and (12),

$$X\_addition = X\_MUX1_{out} + X\_Shift_2 \tag{11}$$

$$X\_MUX2_{out} = sel\,?\,X\_addition: X\_Shift_1 \tag{12}$$

where, X_addition – Adder output.

The same process is performed for the "Y" terminal side which generated the MUX and addition values based on the previous process occurred in "X" terminal. The generation of "Y" terminal side values are explained in Eqs. (13) to (17),

$$Y\_MUX1_{out} = sel\,?\,Y :\ Y\_MUX2_{out} \tag{13}$$

$$\text{Y\_Shift}_1 = \text{Y}_1 \ll 4 \quad (14)$$

$$\text{Y\_Shift}_2 = \text{Y}_1 \ll 5 \quad (15)$$

$$\text{Y\_addition} = \text{Y\_MUX1}_{\text{out}} + \text{Y\_Shift}_2 \quad (16)$$

$$\text{Y\_MUX2}_{\text{out}} = \text{sel}\,?\,\text{Y\_addition} : \text{Y\_Shift}_1 \quad (17)$$

where, $\text{Y\_MUX2}_{\text{out}}$ – output of the "Y" terminal MUX.

At final stage, the X and Y terminal outputs are performed the final stage of MUX operation to produce the MS-CLCG random output which is given in Eq. (18),

$$\text{MS} - \text{CLCG}_{\text{out}} = \text{sel}\,?\,\text{X\_MUX2}_{\text{out}} : \text{Y\_MUX2}_{\text{out}} \quad (18)$$

where, $\text{MS} - \text{CLCG}_{\text{out}}$ – Pseudo Random number output. If the selection line is 1, $\text{X\_MUX2}_{\text{out}}$ is delivered in the$\text{MS} - \text{CLCG}_{\text{out}}$. Else, $\text{Y\_MUX2}_{\text{out}}$ is delivered in the $\text{MS} - \text{CLCG}_{\text{out}}$ . Normally, the random number is generated with the help of "$random" function in the conventional works. This true random number also identified due to the repetition values. So, the pseudo random number generation process is implemented in this research work. With the help of MS-CLGG architecture, the pseudo random number is generated which is considered as key for encryption as well as decryption. This mathematical logic operation helps to generate the different key values for every clock cycle. Additionally, there is no usage of the registers to store the intermediate data which helps to reduce the components count. Due to the less combination blocks in the proposed work, the LUT, flip flops, slices, power consumption and speed of the proposed PRESENT architecture is improved. The speed of the proposed work is decided based on the operating frequency of the design model. Due to the usage of MS-CLCG, the frequency of the proposed architecture is high which helps to process the entire system effectively. Moreover, the encryption and decryption application also implemented in this research work which helps to secure the multimedia data from adversary.

## 5. Simulation setup

The SWN-SP architecture was implemented using 4GB RAM with 3.30 GHz, i3 processor, and 500GB hard disk. The PRESENT algorithm was implemented in Xilinx 14.4 and simulated in ModelSim 10.5 software to verify the timing diagram. Verilog language has been used to write the coding for each and every FPGA module. Xilinx 14.4 ISE software is used to evaluate the FPGA performances and taking RTL schematic of each module. The MATLAB r2018a software is used to read the input image and retrieve the output images.

### 5.1 Results and discussion

The SWN-SP architecture is compiled in Xilinx ISE software to perform synthesis and implementation design. In the synthesizing process, the coding has been compiled and it helps to show the RTL schematic diagram.

The technology schematic is also possible to view which contains a different kind of RTL schematics. After performing the implementation process, all the FPGA performances are calculated and are given in Table 2 to 4.

Table2. Hardware utilization of proposed PRESENT in Virtex 6 FPGA

| FPGA performances | Total resources | Occupied resources | % of utilization |
|---|---|---|---|
| Number of slice registers | 93,120 | 33 | 1% |
| Flip Flops | 46,560 | 33 | 1 % |
| Number of slice LUTs | 46,560 | 56 | 1 % |
| Number of used as a logic | 46,560 | 52 | 1 % |
| Slices | 11,640 | 28 | 1% |
| Bonded IOB | 240 | 35 | 14% |

Table 3. Hardware utilization of proposed PRESENT in Virtex 6 xc6vlx75tl Low power FPGA

| FPGA performances | Total resources | Occupied resources | % of utilization |
|---|---|---|---|
| Number of slice registers | 93,120 | 33 | 1% |
| Flip Flops | 46,560 | 33 | 1 % |
| Number of slice LUTs | 46,560 | 52 | 1 % |
| Number of used as a logic | 46,560 | 50 | 1 % |
| Slices | 11,640 | 31 | 1% |
| Bonded IOB | 240 | 35 | 14% |

Table 4. Hardware utilization of proposed PRESENT in Virtex 7 xc7vx330t FPGA

| FPGA performances | Total resources | Occupied resources | % of utilization |
|---|---|---|---|
| Number of slice registers | 408000 | 33 | 1% |
| Flip Flops | 204000 | 33 | 1 % |
| Number of slice LUTs | 204000 | 56 | 1 % |
| Number of used as a logic | 204000 | 54 | 1 % |
| Slices | 51000 | 44 | 1% |
| Bonded IOB | 600 | 35 | 5% |

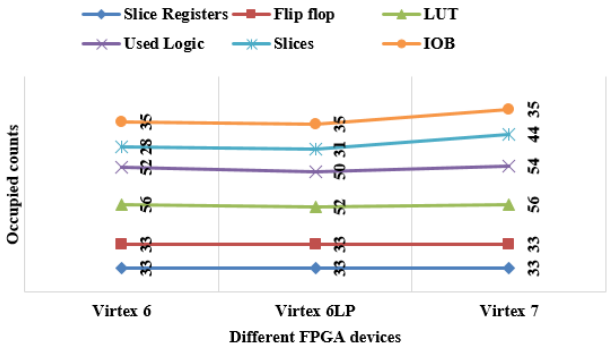



Figure. 6 FPGA performances for different Virtex device

The SWN-SP architecture occupied 33 slice registers among 93,120 available resources. The utilization percentage is mentioned as 1. The flip flop storage element requires 33 counts among the 46,560 total elements. The utilization percentage of the flipflops is 1. The LUT is used in the entire architecture for predefined storage purposes and key scheduling module. So, an overall 56 LUT counts has used in this architecture among 46,560 counts. The utilization of the slices is 35 from 11,640 counts. The bonded IOB utilization count is 35 among 240. The utilization percentage of bonded IOB is 14%. The FPGA performances are evaluated for three different FPGA devices such as Virtex 6, Virtex 6 Low power (LP) and Virtex 7 which are tabulated in Table 2, 3 and 4. The pictorial representation of the FPGA performances is presented in Fig. 6.

This image is taken from Xilinx ISE software using "X-power analyser". The analysis of power, throughput, and delay of the proposed design is given in Table 5. From the synthesized report, the delay values are evaluated for the different FPGA devices. The throughput of the system is evaluated using Eq. (19),

Table 5. Analysis of power, throughput, and delay for proposed system

| FPGA devices | Power (W) | Throughput | Delay (ns) |
|---|---|---|---|
| Virtex 6 – xc6vcx75t | 1.293 | 1.316 | 3.032 |
| Virtex 6 – xc6vlx75tl | 1.065 | 1.043 | 3.832 |
| Virtex 7 – xc7vx330t | 0.178 | 1.56 | 2.56 |

$$Throughput = Number\ of\ bits\ per\ one\ clock\ cycle \times Highest\ clock\ speed \quad (19)$$

where, Number of bits per one clock cycle - 4

The highest clock speed is evaluated using Eq. (20).

$$Highest\ clock\ speed = \frac{1}{delay} \quad (20)$$

### 5.2 Comparative analysis

The comparison of the SWN-SP architecture with conventional encryption architecture is given in

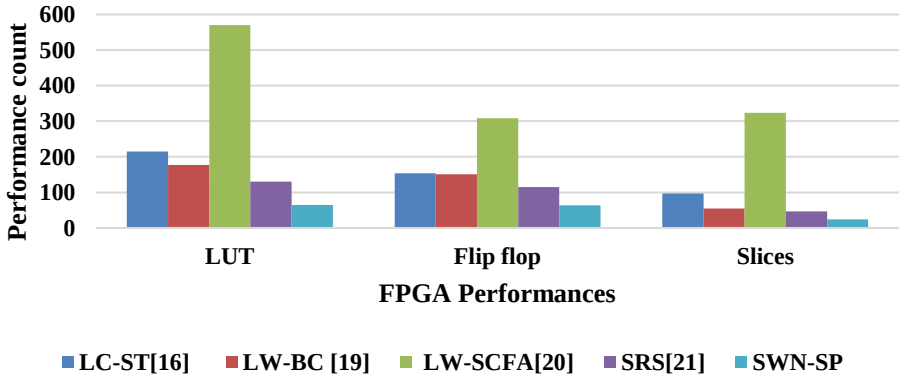



Figure. 7 Comparison of FPGA performances for different methodologies

Table 6. The conventional encryption architectures are implemented in different FPGA devices such as Spartan 6 XC6SLX16 [16], Kintex 7 XC7K325T [16], Spartan 3 XC3S400 [19], Virtex-5 XC5VLX50 [20] and Virtex 6 XC6VCX75T [21]. In [16], LC-ST PRESENT architecture was implemented to perform the encryption algorithm. The key scheduling module has been designed using shifter and FSM design which occupies more area to design the entire architecture.

In [19], LW-BC-PRESENT cryptographic algorithm was designed to improve the latency performance. But, these RTL designs are occupied more area, power and less key sensitivity. In [20], LW-SCFA architecture was implemented for the data security. In the process of spatial redundancy, two registers are used for fault transformation. In this design, the key values are generated manually which caused more system complexity. In [21], the PRESENT algorithm was designed with default architecture and traditional key scheduling. Due to the usage of more logical blocks, all the conventional algorithms were occupied more hardware to perform the encryption process. Moreover, some of the existing algorithms were not discussed about decryption and image cryptographic applications. To overcome the conventional method drawbacks, proposed SWN-SP architecture is implemented in this research work. In the proposed work, the intermediate registers are not required to store the intermediate data. Due to this process, the hardware utilization and power consumption of the proposed work is reduced when compared to the conventional works. With the help of MS-CLCG key generation, the randomness of the key also increased which helps to perform the cryptography with high security. The pictorial representation of comparative analysis is shown in Fig.7. From this graph, it is clears that the LUT, flip flops, and slices are reduced more in SWN-SP architecture compared to the conventional architectures [16, 19, 20, 21]. The output waveform of the proposed architecture is shown in Fig. 8. Generally, all the sequential circuits required clock signals for storage purposes. Enable and reset signals are used to perform the encryption operation at a particular stage. From the MATLAB, images are converted to the binary values (y,y1) which are concatenated to produce the plain text. This plain text performs the PRESENT encryption with SRS key value. The reverse process of the encryption is performed to generate the decryption operation. The encryption and decryption output is stored in enc_out and dec_out variable which is given to the MATLAB to retrieve the cipher and output image.

The input of the color image is shown in Fig. 9 (a) which is performed the "RGB to gray" pre-processing to generate the grey scale image shown in Fig. 9 (b). From FPGA output, the text file is generated which is given to the MATLAB software. By using binary to decimal function, all the stored binary values are converted to the pixel value. Each 8 bit of data holds a single pixel value. With the help of encryption and decryption text files, cipher and decrypted images are taken which are shown in Fig. 10 (a) and (b).

### 5.3 Quantitative and cryptographic analysis

From the Decryption output, the PSNR, MSE, and SSIM performances are evaluated for the SWN-SP and SWN-SP-KST methodology which values are given in the table 7. The quantitative analysis are evaluated while comparing the decryption output with the input image. From the table 7, it is clear that the quantitative analysis of the proposed design meets the sufficient values which helps to retrieve

Table 6. Comparison of FPGA analysis for the different architectures

| Target Family | Device | SG and P | Architecture | LUT | Flip flop | Slices | Frequency (MHz) | Power (W) |
|---|---|---|---|---|---|---|---|---|
| Spartan 6 | XC6SLX16 | -3 and CSG324C | LC-ST [16] | 225 | 156 | 68 | 254.63 | 2.962 |
| | | | **SWN-SP** | **89** | **52** | **34** | **298.14** | **2.451** |
| Kintex 7 | XC7K325T | -2 and FFG900 | LC-ST-PRESENT [16] | 215 | 154 | 97 | 214.36 | 2.682 |
| | | | **SWN-SP** | **58** | **39** | **33** | **254.12** | **2.124** |
| Spartan 3 | XC3S400 | -5 and PQ208 | LW-BC [19] | 350 | 154 | 202 | 190.47 | 2.364 |
| | | | **SWN-SP** | **89** | **52** | **34** | **225.21** | **2.140** |
| Virtex 5 | XC5VLX50T | -2 and FF1136 | LW-BC [19] | 177 | 151 | 55 | 264.21 | 2.142 |
| | | | **SWN-SP** | **69** | **58** | **24** | **301.24** | **1.984** |
| Virtex 5 | XC5VLX50 | -2 and FF1136 | LW-SCFA[20] | 570 | 308 | 324 | 105.1 | 1.897 |
| | | | **SWN-SP** | **65** | **64** | **24** | **311.24** | **1.364** |
| Virtex 6 | XC6VCX75T | -1 and FF484 | SRS[21] | 130 | 115 | 47 | 90.26 | 1.597 |
| | | | **SWN-SP** | **56** | **33** | **28** | **329.79** | **1.293** |

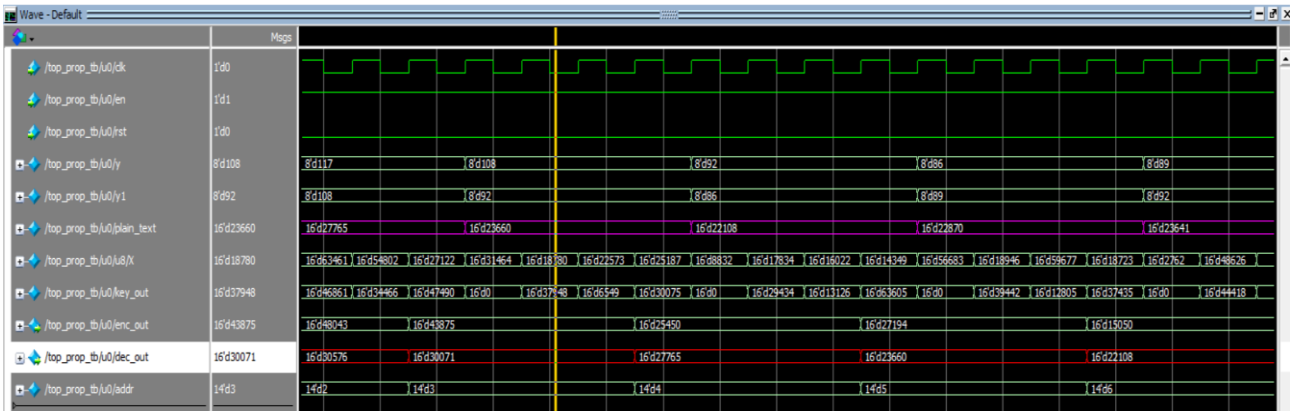
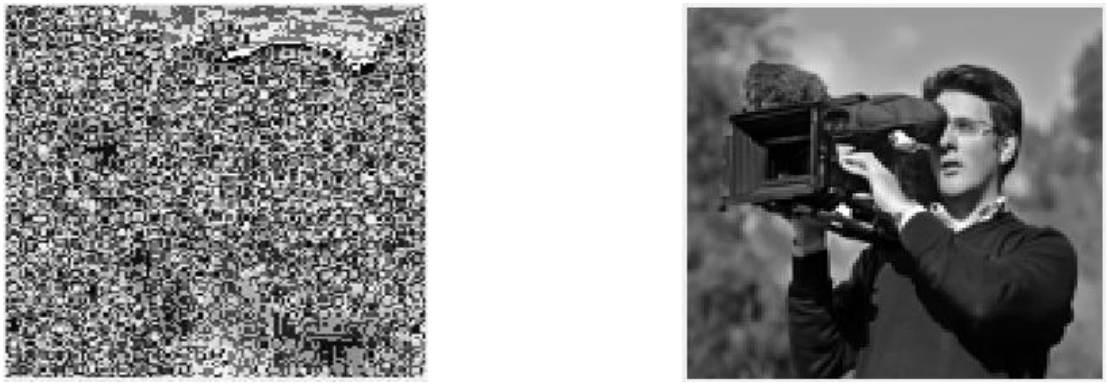



Figure. 8 Modelsim output waveform

(a) (b)

Figure. 9: (a) input color image and (b) input gray scale image



(a) (b)

Figure. 10: (a) encryption image and (b) decryption image

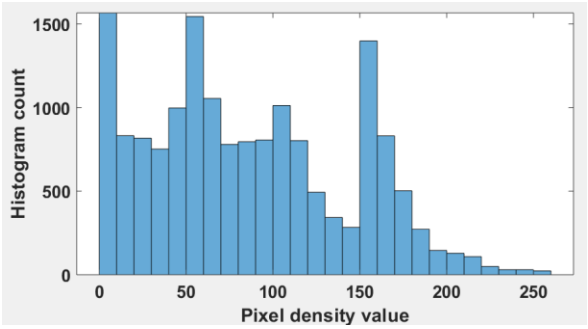
Table 7. Quantitative analysis of proposed design

| Methodology | PSNR | MSE | SSIM |
|---|---|---|---|
| SWN-SP | 40.246 | 37.47 | 0.913 |
| SWN-SP-KST | 38.24 | 40.24 | 0.891 |

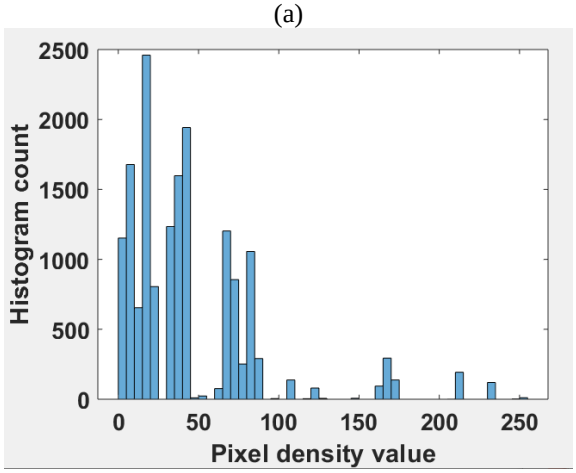
the original date with less noise. If the Key Sensitive Test (KST) performed in the proposed model, the quantitative analysis performance also reduced due to the degradation in the output image.

The cryptographic analysis of the proposed work is evaluated based on the Histogram analysis and key sensitive test operation.

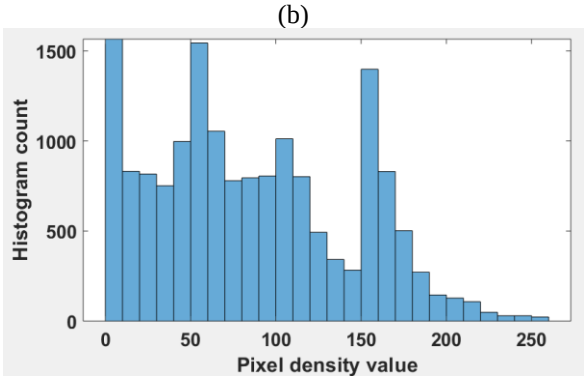
The histogram of the input image, encryption image, decryption image is shown in Fig. 11 (a), (b), and (c) respectively. From the histogram analysis, it is clear that the proposed design retrieve the maximum pixel values in the decryption side. Moreover, the key sensitive test process is performed in this research work. If the single bit of key is changed in the decryption side, the decryption image quality is degraded which is shown in Fig. 12. So, the proposed work is very sensitive in the key generation and operation.




(a)




(b)




(c)

Figure. 11: (a) histogram of input image, (b) histogram of encryption image, and (c) histogram of decryption image

Figure. 12 Key sensitive test image

## 6. Conclusion

In general, the PRESENT algorithm contains the shifting and fundamental key scheduling process. This basic RTL schematic structure requires more hardware utilization and it can be hacked by the adversary and there was no proper confidentiality for the fundamental PRESENT architecture. To overcome these issues, the separation with NOT module is used in the PRESENT encryption algorithm. The SP-box is also used in the PRESENT architecture to improve the design performances. In the conventional algorithms, the key values are needed to provide manually for the encryption. But, MS-CLCG architecture is used in this research work to generate the random number of every clock cycle. Due to the randomness, the key values are not possible to predict by unknown persons. This SWN-SP architecture occupied 56 counts of LUT, 33 counts of Flip flop and 30 counts of slices which are less compared to the SRS-PRESENT architecture. The SWN-SP architecture consumed 1.293W power consumption which is reduced compared to the conventional architectures. In the future, different kind of encryption algorithm can be used for the security purpose and optimized key generation will be used to improve the hardware utilization of the entire architecture. Moreover, Spartan 6 FPGA hardware can be used to do the experiments.

## Conflicts of Interest

The authors declare no conflict of interest.

## Author Contributions

The paper background work, conceptualization, methodology, dataset collection, implementation, result analysis and comparison, preparing and editing draft, visualization have been done by first author. The supervision, review of work and project administration, has been done by second author.

## References

[1] S. Madhavapandian and P. MaruthuPandi, “FPGA implementation of highly scalable AES algorithm using modified mix column with gate replacement technique for security application in TCP/IP”, *Microprocessors and Microsystems*, Vol. 73, pp. 102972, 2020.

[2] C. Li, D. Lin, J. Lü, and F. Hao, “Cryptanalyzing an image encryption algorithm based on autoblocking and electrocardiography”, *IEEE MultiMedia*, Vol. 25, pp. 46-56, 2018.

[3] N. Thangamani and M. Murugappan, “A lightweight cryptography technique with

random pattern generation", *Wireless Personal Communications*, Vol. 104, pp. 1409-1432, 2019.

[4] S. E. Abed, R. Jaffal, B. J. Mohd, and M. Alshayeji, "FPGA modeling and optimization of a Simon lightweight block cipher", *Sensors*, Vol.19, No. 4, p. 913, 2019.

[5] C. A. L. Nino, A. D. Perez, and M. M. Sandoval, "Energy and area costs of lightweight cryptographic algorithms for authenticated encryption in WSN", *Security and Communication Networks*, 2018.

[6] A. Aghaie, M. M. Kermani, and R. Azarderakhsh, "Fault diagnosis schemes for low-energy block cipher Midori benchmarked on FPGA", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 25, pp. 1528-1536, 2016.

[7] M. James and D. S. Kumar, "An implementation of modified lightweight advanced encryption standard in FPGA", *Procedia Technology*, Vol. 25, pp. 582-589, 2016.

[8] R. Sadhukhan, S. Patranabis, A. Ghoshal, D. Mukhopadhyay, V. Saraswat, and S. Ghosh, "An evaluation of lightweight block ciphers for resource-constrained applications: Area, performance, and security", *Journal of Hardware and Systems Security*, Vol. 1, pp. 203-218, 2017.

[9] S. Dogan, E. Akbal, T. Tuncer, and U. R. Acharya, "Application of substitution box of present cipher for automated detection of snoring sounds", *Artificial Intelligence in Medicine*, 117, p.102085, 2021.

[10] L. Dalmasso, F. Bruguier, P. Benoit, and L. Torres, "Evaluation of SPN-based lightweight crypto-ciphers", *IEEE Access*, Vol. 7, pp. 10559-10567, 2019.

[11] N. Nedjah, L. D. M. Mourelle, and C. Wang, "A parallel yet pipelined architecture for efficient implementation of the advanced encryption standard algorithm on reconfigurable hardware", *International Journal of Parallel Programming*, Vol. 44, pp. 1102-1117, 2016.

[12] F. Macé, F. X. Standaert, and J. J. Quisquater, "FPGA implementation (s) of a scalable encryption algorithm", *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 16, pp. 212-216, 2008.

[13] B. J. Mohd, T. Hayajneh, K. M. A. Yousef, Z. A. Khalaf, and M. Z. A. Bhuiyan, "Hardware design and modeling of lightweight block ciphers for secure communications", *Future Generation Computer Systems*, Vol. 83, pp. 510-521, 2018.

[14] R. Sakthivel, "VLSI Implementation of Lightweight Cryptography Algorithm", *Advances in Systems Science and Applications*, Vol. 16, pp. 95-101, 2016.

[15] C. A. L. Nino, A. D. Perez, and M. M. Sandoval, "Lightweight hardware architectures for the present cipher in FPGA", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 64, pp. 2544-2555, 2017.

[16] Z. Haider, K. Javeed, M. Song, and X. Wang, "A Low-Cost Self-Test Architecture Integrated with PRESENT Cipher Core", *IEEE Access*, Vol. 7, pp. 46045-46058, 2019.

[17] B. Rashidi, "High-throughput and lightweight hardware structures of HIGHT and PRESENT block ciphers", *Microelectronics Journal*, Vol. 90, pp. 232-252, 2019.

[18] G. Sravya, M. O. Kumar, G. M. Sheeba, K. Jamal, and K. Mannem, "Hardware lightweight design of PRESENT block cipher", *Materials Today: Proceedings*, 33, pp. 4880-4886, 2020.

[19] J. G. Pandey, T. Goel, and A. Karmakar, "Hardware architectures for PRESENT block cipher and their FPGA implementations", *IET Circuits, Devices & Systems*, Vol. 13, pp. 958-969, 2019.

[20] S. Patranabis, D. B. Roy, A. Chakraborty, N. Nagar, A. Singh, D. Mukhopadhyay, and S. Ghosh, "Lightweight design-for-security strategies for combined countermeasures against side channel and fault analysis in IoT applications", *Journal of Hardware and Systems Security*, Vol. 3, pp. 103-131, 2019.

[21] P. Srikanth and S. Alluri, "Lorentz chaotic system key generation with Low area FPGA implementation using PRESENT security algorithm", *International Journal of Recent Technologies and Engineering*, Vol. 8, 2019.