iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.42143

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

FPGA Implementation of Rectangle Lightweight Block Cipher

Karishma Shaukat Sayyed[1], Prof. S. R. Ganolkar[2], Prof. S. O. Rajankar[3]
[1, 2, 3]*VLSI & Embedded System, SCOE, Pune, Maharashtra, India*

Abstract: ***Block ciphers are basic building blocks for network security. In recent years, designing a lightweight block cipher is the main goal of VLSI design engineers.***
In this paper, we have designed and verified the functionality of the RECTANGLE block cipher which is one of the lightweight block cipher using Modelsim simulator and implemented using Intel Quartus Prime 18.0 FPGA device. Using the bit-slice technique a RECTANGLE block cipher allows lightweight and fast implementations. The en-cryption architecture has two parts one is round transformation and the other is key scheduling.
RECTANGLE uses Substitution-Permutation network. It takes 64-bit plain text and an 80-bit key as an input and converts it into a 64-bit ciphertext.
There are three main advantages of using the RECTANGLE block cipher. First, it has a simple design. Second, it is very hardware friendly. By selecting the proper S-block RECTANGLE can achieve good security performance.
Index Terms: Lightweight Block Cipher, Block Ciphers, Encryption, Bit-slice technique, Round Transformation, Key Scheduling, Substitution Block, Permutation Block.

I. INTRODUCTION

In a world, full of cybercrimes and data misuse, information and network security is the need of the hour. Here cryptography plays an important role. Cryptography is nothing but a process that converts an ordinary plaintext into ciphertext and vice-versa. It is the method that protects information and communicates through the codes. Cryptography is classified into two categories first is the mode of operation and second Feistel network, substitution-permutation network. It can be done by symmetric key and asymmetric key.

Block ciphers are the primary integrant of cybersecurity as they are used to convert plaintext to ciphertext and back. But light-weighted block ciphers have substituted all the other conventional block ciphers because of its low computing resource, low memory usage, and power.

Symmetric cryptography utilizes only one key for encryption and decryption of data, while in the other hand asymmetric or public-key cryptography utilizes both public and private keys for encryption and decryption of data. Block cipher is a symmetric cipher process information block of various bits using constant mapping.

A lightweight block cipher is not the same as block ciphers as it used different algorithms that require less area, memory (RAM, ROM), power, etc. It gives a much simpler and faster result compared to the conventional ones. In this paper, we have verified the functionality of an architecture that has been designed and implemented using the Intel Quartus Prime 18.0 FPGA tool in the Cyclone IV device.

Our main objective in this project is to design and verify the functionality of a lightweight RECTANGLE block cipher which consists of various blocks. For this, we have designed different blocks such as S-blocks, Permutation block, 2:1 multiplexers, round counters, etc.

In this paper, we present a design of a RECTANGLE lightweight block cipher using 4x4 S-block. The paper contains the following sub-sections: in section II, a literature review is presented. In that, subsection A explains the rectangular block cipher. The architecture of RECTANGLE is discussed in subsection B again it has four parts, part 1 explains the subkey state and cipher state. Part 2 describes the round transformation methodology in the architecture. Subsequently, in part, the 3 S-block design has been discussed.

In part, 4 explains the key scheduling methodology in architecture. Simulation results are shown in subsection C. Section III will conclude all the work.

©IJRASET: All Rights are Reserved

II. LITERATURE REVIEW

Following observations are made from different papers.

In [1], The Author has implemented an L-block lightweight ci-pher. A full functional RTL is designed for L-block lightweight cipher which requires 32-clock cycles to encrypt a block of 64-bit message. In [2], the Author has proposed A Lightweight VLSI Architecture for the RECTANGLE cipher. He proposed an efficient lightweight block architecture for an extremely hardware constrained environment and further implementing ASIC on CMOS 180nm technology. In [3], the Author pro-posed a design of RECTANGLE light-weighted block cipher and RECTANGLE S-box and asymmetric design of the P layer for fast implementations by using bit-slice techniques. In [4], Survey on lightweight symmetric and asymmetric block ciphers is done. In [5], the author has synthesized and implemented an ultra-lightweight block cipher RECTANGLE, which has 64 bit and 80 or 128 bit for block and key, respectively. In paper [6], the authors described KLEIN as a new family of lightweight block cipher which is designed for resource-constrained devices such as wireless sensors and RFID tags. It has efficient software performance in sensor platforms, and also its hardware implementation is compact. In [7], the authors described HIGHT as a lightweight block cipher with a 64-bit block size and 128-bit key that provides a low resource hardware implementation. It requires 3048 gates implemented on a 0.25um technology. In [8], the present block cipher is implemented and the simulation of encryption and decryption is done using the blowfish algorithm so that it pro-vides great security between sender and receiver. In paper [9] architecture is based on 8-bit datapath and requires 48 clock cycles for processing of 64-bit plaintext and 128-bit user key and the performance is validated by using intellectual property of SoCs.Paper [10] is the deep comparison present and the latest developments in the field are also noted. Hardware and software implementations of block ciphers were examined. Security, cost and performance properties of all the different proposals were considered and a comparative analysis was presented.

A. Rectangle Block Cipher

A rectangular block cipher operates on 64-bit block size matrices (in 4x16 array) for ciphertext, plain text, and intermediate results along with 80-bit or 128-bit key. In this paper, we would be implementing hardware architecture with an 80-bit key for RECTANGLE lightweight block cipher on FPGA. RECTANGLE uses SP network cipher which consists of two layers- Substitution Layer(a Non-Linear layer) and Permutation Layer(a Linear layer). It requires 25 internal rounds of Key-scheduling. Each round consists of three stages:

1) *Add Round Key:* Each byte of state undergoes a bit-wise XOR operation with a round key.
2) *Mix Columns:* A non-linear parallel substitution operation in which each byte is being substituted with another byte-based on a look-up table.
3) *Shift Rows:* A transposition operation in which each row is left-shifted in a certain order by a certain number of steps.

After the 25th round, there is a final Add Round Key. We would be listing out all these operations in detail while we go further into the data pathway.

B. Rectangle Architecture

The RECTANGLE Lightweight Block Cipher architecture shown in fig. 1 utilizes a 64-bit datapath to get the encrypted cipher state. 16 S-Boxes operate parallelly as it is applied to all 64-bits. In Key scheduling, S-box substitution is applied to only 16-bits, so requires only four S-Boxes. The round Transformation and Key scheduling operations run parallelly. These all will be described in precisionl in the subsequent sections.

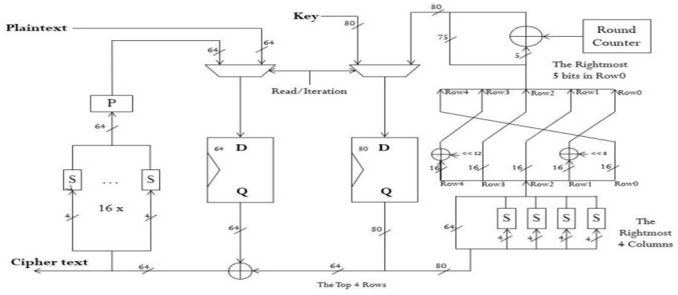


Fig. 1. Datapath Architecture of RECTANGLE Cipher.

1) *Sub-Key State and Cipher State:* One of the main reasons of naming the lightweighted cipher as RECTANGLE is that the block sizes can be pictured as a (4 x 16) array of bits. The cipher state consists of a 64-bit plaintext or ar 64-bit cipher text. let w_{15}, w_{14}.... w1,w0 represent the cipher state in first row. Similarly,w_{31} .....w_{17},w_{16} bits are arranged in second row and so on as shown below in Matrix1.

$$\begin{bmatrix} w_{15} & w_{14} & ..... & w_1 & w_0 \\ w_{31} & w_{30} & ..... & w_{17} & w_{16} \\ w_{47} & w_{46} & ..... & w_{33} & w_{32} \\ w_{63} & w_{62} & ..... & w_{49} & w_{48} \end{bmatrix}$$

Fig. 2. 64-bit Block Size (Matrix 1).

For convinience, a cipher text can also be represented in a two-dimensional array as shown in matrix2.
Also, a 64-bit key can be represented as (4 x 16) array known as sub-key state. The algorithm requires three steps

$$\begin{bmatrix} X_{0,15} & X_{0,14} & ..... & X_{0,1} & X_{0,0} \\ X_{1,15} & X_{1,14} & ..... & X_{1,1} & X_{1,0} \\ X_{2,15} & X_{2,14} & ..... & X_{2,1} & X_{2,0} \\ X_{3,15} & X_{3,14} & ..... & X_{3,1} & X_{3,0} \end{bmatrix}$$

Fig. 3. 64-bit Block Size in 2D (Matrix 2).

which is explained in the subsequent subsections.

PSEUDOCODE [2] :

```
generateRoundkeys()
  for (i=0 to 24)
  do
    addRoundkey(state,Ki);
    subcolumn(state);
    shiftrow(state);
  end;
addRoundkey(state,K25)
```

The first 64-bit of user key undergoes 25 internal rounds. The final Round key after 25th iteration, is used for post key-whitening (i.e. XORing before the first round and after the final round of encryption).

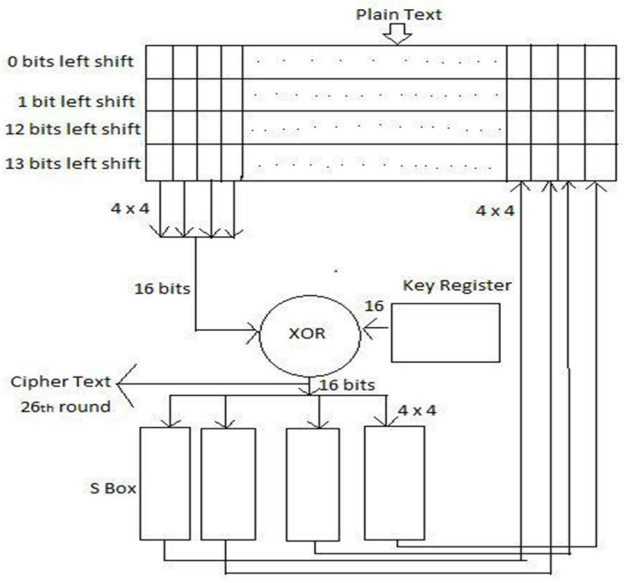


Fig. 4. Round Transformation Operation.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

2) *Round Transformation:* There are two 64-bit registers. One to store the round sub-key and the other for ciphertext. The ciphertext like we saw above is represented as a 4 x 16 array and at a time only 16-bits are processed sequentially. At every clock cycle, the leftmost four columns of the current state are being XORed with the correspondent subkey register.

This operation is called the Add Round Key. Then the S-Box used in the RECTANGLE algorithm behaves as a 4-bit x 4-bit S-box SCHEDULING. Let T_i be the 16-bit temporary state., for i= 1 to 12. The sub-column transformation can be evaluated in 12 steps:

1.T_1 = NOT X_1, 2. T_2= X_0 AND T_1, 3. T_3 = X_2 XOR X_3,

4. Y_0 = T_2 XOR T_3, 5. T_5= X_3 OR T_1,6.T_6 = X_0 XOR T_5,

7. Y_1 = X_2 XOR T_6, 8. T_8 = X_1 XOR X_2, 9.T_9 = T_3 AND T_6, 10.Y_3 = T_8 XOR T_9, 11. T_{11} = Y_0 OR T_8,

12. Y_2 = T_6 XOR T_{11}

The result of the XOR is put through parallel operation of (4 x 4) S-Box as illustrated below.

Input of to the S-Box(Matrix 3) is : column(j)= $X_{3;j}$, $X_{2;j}$, $X_{1;j}$, $X_{0;j}$; (for j = 0 to 15)

i.e.

$$\begin{bmatrix} X_{0,15} \\ X_{1,15} \\ X_{2,15} \\ X_{3,15} \end{bmatrix}$$

Fig. 5. S-Box Input (Matrix3).

and ouput of the S-Box (Matrix 4) is : column(j)= $Y_{3;j}$, $Y_{2;j}$, $Y_{1;j}$, $Y_{0;j}$; (for j= 0 to 15) i.e.

$$\begin{bmatrix} Y_{0,15} \\ Y_{1,15} \\ Y_{2,15} \\ Y_{3,15} \end{bmatrix}$$

Fig. 6. S-Box Output (Matrix4).

Similarly, the non-linear substitution is done for all the 16 columns parallelly using 16 (4 x 4) S-Boxes. The values are specified in hexadecimal that we will discuss in the next subsection. The final step of round Transformation includes Shift Row operation which is called as the Permutation layer. Row 0 is shifted by 0 bits (i.e. no rotation). Row 1 is left shifted by 1 bit. Row 2 is left shifted by 12 bits and the last row (i.e. row 3) is left shifted by 13 bits. Let Y_0, Y_1, Y_2, Y_3 be the four 16-bit inputs of shift-row operation and Z_0, Z_1, Z_2, Z_3 be the four 16-bit outputs. So,

Z_0 = Y_0;
Z_1 = left-shift Y_1 by 1 bit ;
Z_2 = left-shift Y_2 by 12 bits;
Z_3 = left-shift Y_3 by 13 bits;
So, these are the three steps of round transformation.

In the next clock cycle, 16 bits of cipher state is XORed with correspondent bits of subkey register. By the end of one clock cycle , the control signal is used to rotate the four rows over the conditions specified in shift row transformation and simultaneously store the values of the last S-Box output. This is known as Round Transformation.One clock cycle completes one round transformation. The control signal simultaneously loads the next round key into the sub-key register that is extracted from key-scheduling. After 25 rounds of 64-bit XOR of the current state with a sub-key state, the cipher state gives the final output.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

3) S-Box Design : S-box executes substitution operation. Here, we have taken a 4 x 4 S-Box (i.e. four inputs and four outputs). We have generated a look up table as shown in table1.

TABLE I
S-BOX LOOKUP TABLE

INPUT	OUTPUT
0	6
1	5
2	C
3	A
4	1
5	E
6	7
7	9
8	B
9	0
A	3
B	D
C	8
D	F
E	4
F	2

4) Key Scheduling: Key scheduling requires 80- bit (rep-resented as 5 x 16 array) key registers. It can be represented in matrix form as shown below in matrix 5.

$$\begin{bmatrix} v_{15} & v_{14} & \ldots\ldots & v_1 & v_0 \\ v_{31} & v_{30} & \ldots\ldots & v_{17} & v_{16} \\ v_{47} & v_{46} & \ldots\ldots & v_{33} & v_{32} \\ v_{63} & v_{62} & \ldots\ldots & v_{49} & v_{48} \\ v_{79} & v_{78} & \ldots\ldots & v_{65} & v_{64} \end{bmatrix}$$

Fig. 7. Key Scheduling Size (Matrix 5).

For convinience, it can also be represented in a two-dimensional array (in matrix 6) as $K_{i;15}$,..... $K_{i;1}$, $K_{i;0}$,for (i=0 to 4).

$$\begin{bmatrix} K_{0,15} & K_{0,14} & \ldots\ldots & K_{0,1} & K_{0,0} \\ K_{1,15} & K_{1,14} & \ldots\ldots & K_{1,1} & K_{1,0} \\ K_{2,15} & K_{2,14} & \ldots\ldots & K_{2,1} & K_{2,0} \\ K_{3,15} & K_{3,14} & \ldots\ldots & K_{3,1} & K_{3,0} \\ K_{4,15} & K_{4,14} & \ldots\ldots & K_{4,1} & K_{4,0} \end{bmatrix}$$

Fig. 8. Key Scheduling Size in 2D (Matrix 6).

At round i (total 25 rounds from i= 0 to 24), the sub-key K_i, contains the first four rows of key register (K_i = row3, row2, row1, row0). In the first step of key scheduling, S-Box substitution is applied to the bits intersecting the rightmost columns and the four uppermost rows. i.e. Input of to the S-Box (in matrix 7) is : column(j)= $K_{3;j}$, $K_{2;j}$, $K_{1;j}$, $K_{0;j}$; (for j = 0 to 3)
i.e. (for j=0)

$$\begin{bmatrix} K_{3,0} \\ K_{2,0} \\ K_{1,0} \\ K_{0,0} \end{bmatrix}$$

Fig. 9. S-Box Input in Key Scheduling (Matrix 7).

and ouput of the S-Box (in matrix 8) is : column(j)= $L_{3;j}$, $L_{2;j}$, $L_{1;j}$, $L_{0;j}$; (for j= 0 to 3) i.e. (for j=0)

$$\begin{bmatrix} L_{3,0} \\ L_{2,0} \\ L_{1,0} \\ L_{0,0} \end{bmatrix}$$

Fig. 10. S-Box Outputput in Key Scheduling (Matrix 8).

Similarly, the non-linear substitution is done for the first four columns parallelly using four (4 x 4) S-Boxes respectively. And the remaining 64 bits are passed through the wire. Next, one round of Fiestal transformation is used to the existing five rows which are also referred to as Permutation Layer.
i.e. $NewRow_0$ = (left-shift Row_0 by 8 bits) XOR with Row_1;
$NewRow_1 = Row_2$;
$NewRow_2 = Row_3$;
$NewRow_3$ = (left-shift Row_0 by 12 bits) XOR with Row_4;
$NewRow_4 = Row_0$
After executing this step, we move to the last step where the right-most five bits of $NewRow_0$ is XORed with a round constant RC_i (which is five bits). Each round of iteration has different values of round constants which are being generated by a five-bit LFSR (linear-feedback shift register).
i.e. ($L^0;4$, $L^0;3$, $L^0;2$, $L^0;1$, $L^0;0$) = ($K^0;4$, $K^0;3$, $K^0;2$,
$K_{0;1}$, $K_{0;0}$) XORed with (RC_i). The values of RC_i are as follows:
RC_0 = 0X01, RC_1] = 0X02,RC_2 = 0X04,RC_3 = 0X09,RC_4
= 0X12, RC_5 = 0X05, RC_6 = 0X0B, RC_7 = 0X16, RC_8
= 0X0C, RC_9 = 0X19,RC_{10} = 0X13,RC_{11} = 0X07, RC_{12}
= 0X0F,RC_{13} = 0X1F,RC_{14} = 0X1E,RC_{15} = 0X1C,RC_{16} =

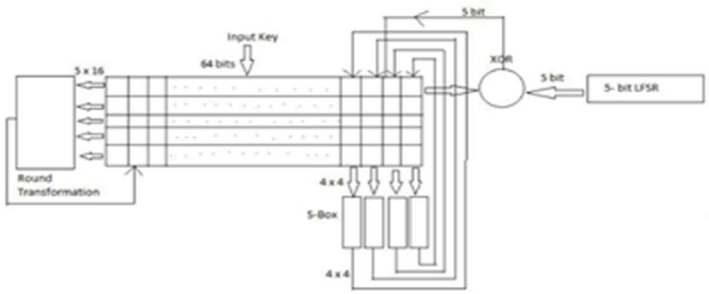


Fig. 11. Key Scheduling Operation.

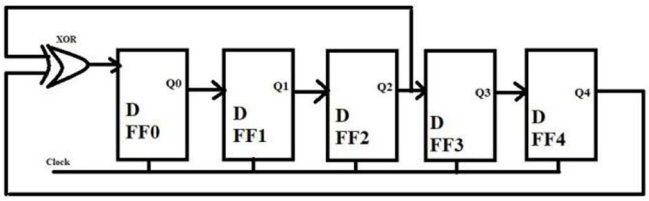


Fig. 12. Linear-feedback shift register for round count generation

0X18,RC_{17} = 0X11, RC_{18} = 0X03,RC_{19} = 0X06,RC_{20} = 0X0D,RC_{21} = 0X1B,RC_{22} = 0X17,RC_{23} = 0X0E,RC_{24} = 0X1D.
After 25 round cycles, K_{25} is finally extracted.

C. Simulation Results



Fig. 13. Output for key and plaintext.

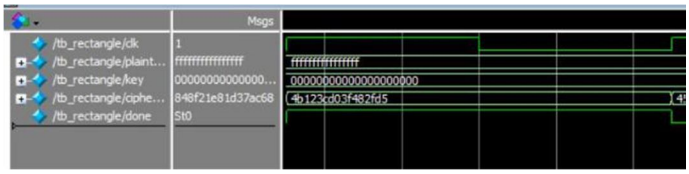


Fig. 14. Output for Random key and plaintext.

III. CONCLUSION

A RECTANGLE block cipher is a flexible lightweight (based on bit-slicing) algorithm that is used in many applica-tions such as RFID, WSNs, etc. From the simulation results.

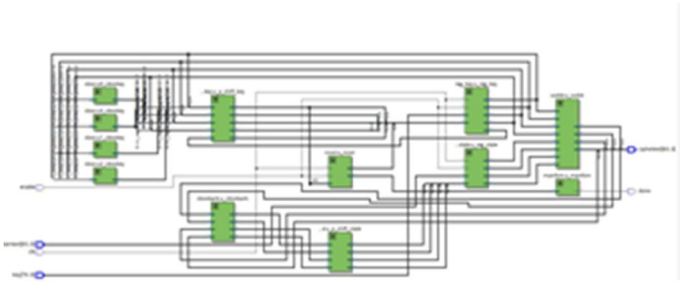


Fig. 15. RTL schematic for main module

Table II
Synthesis Report For Rectangular Block Cipher

Target Device used	Cyclone IV E
Logical Elements used	259
Total registers used	149
Pins used	211

TABLE. III
POWER ANALYSIS SUMMERY

Total Thermal Power Dissipation	187.10 mW
Core Dynamic Thermal Power Dissipation	1.58 mW
Core Static Thermal Power Dissipation	98.65 mW
I/O Thermal Power Dissipation	86.87 mW

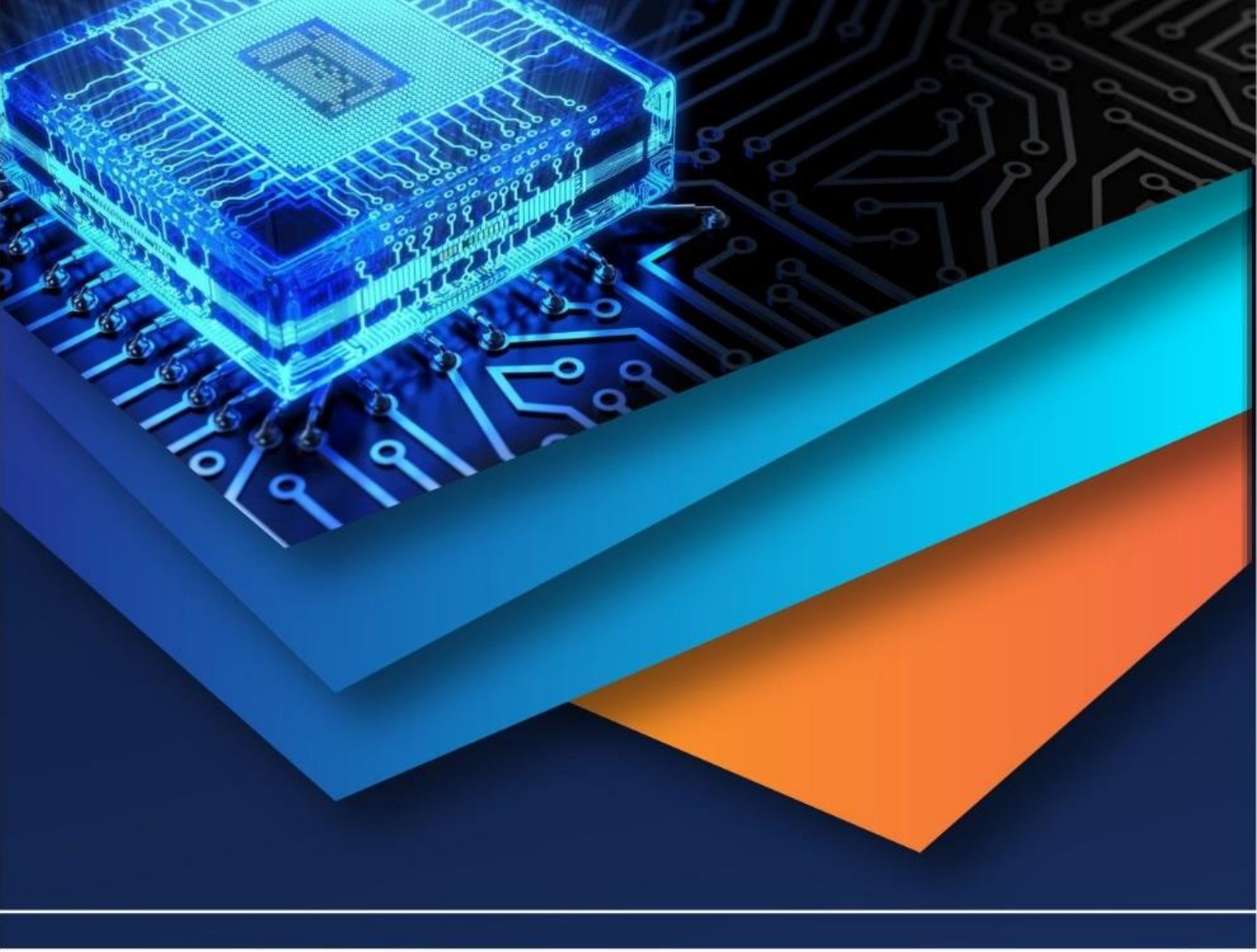
It operates at a maximum of 150MHz. The architecture has been implemented on the Intel Quartus Prime 18.0 device. It has the flexibility to choose different key sizes. It can trigger many problems in the field of cryptography and design. The P-Layer uses three left rotations, which not only reduces cost in hardware but also has software efficiency. Yet RECTANGLE is an interesting design, and there is always a further scope of security-based research in RECTANGLE, be it be in reducing the S-box used, using fewer clock cycles for each rotation, etc.

IV. ACKNOWLEDGMENT

We owe our deep gratitude to our mentor for his constant encouragement, support, and guidance throughout this journey. His keen supervision has not only helped us learn but grow our knowledge base to a new horizon.

REFERENCES

[1] Md. Nazmul Hasan, Md. Tariq Hasan, Rafia Nishat Toma, and Md. Maniruzzaman, “Fpga implementation of l-block lightweight block cipher,” March, 2017, doi: 10.1109/CEEICT.2016.7873062.
[2] Jai Gopal Pandey, Ayush Laddha, and Sashwat Deb Samaddar, “A lightweight vlsi architecture for RECTANGLE cipher and its Im-plementation on an FPGA,” 24th International Symposium on VLSI Design and Test (VDAT), Bhubaneswar, India, 2020, pp.16, doi: 10.1109/VDAT50263.2020.9190623.
[3] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede, “Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms,” Sci. China Inf. Sci. 58, 1–15 (2015).
[4] Hatzivasilis, G., Fysarakis, K., Papaefstathiou, I. et al., “ A review of lightweight block ciphers,” doi:10.1007/s13389-017-0160-y
[5] Soheil Feizi, Ali Nemati, Arash Ahmadi, and Vahab AI-din Makki, “A high-speed fpga implementation of a bit-slice ultra-lightweight block cipher, rectangle,” 2015 5th International Conference on Computer and Knowledge Engineering (ICCKE).
[6] Zheng Gong, Svetla Nikova, and Yee Wei Law. “KLEIN: a new family of lightweight block ciphers”, RFID Security and Privacy, 26-28 June 2011.
[7] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, K. Jeong, H. Kim and S. Chee, “HIGHT: A new block cipher suitable for low-resource device”, LNCS 4249, pp. 46-59, Springer-Verlag, 2006.
[8] Sweta K. Parmar, K.C. Dave, “Implementation of Data Encryption and Decryption Algorithm for Information Technology”, International Journal of Advances in Science Engineering and Technology, Volume-1, Issue- 2, Oct-2013.
[9] Jai Gopal Pandey, Tarun Goel, Mausam Nayak, Chhavi Mitharwal, Abhijit Karmakar, Raj Singh, “A High-performance VLSI Architecture of the PRESENT Cipher and its Implementations for SoCs”, IEEE, 2018.
[10] George Hatzivasilis, Konstantinos Fysarakis, Ioannis Papaefstathiou, Harry Manifavas, “A review of lightweight Block Ciphers”, DOI: 10.1007/s13389-017-0160, JCE, 2017.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)