International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 08 | August - 2022 Impact Factor: 7.185 ISSN: 2582-3930

FPGA IMPLEMENTATION OF PRESENT ALGORITHM FOR IOT APPLICATION

Dr.P. Rajasekar[1], P.Jeevan Kumar Reddy[2],P.Srinadh[2], T.Sasikanth[2],V.Phaneendra Nath[2], Y. Dileep Kumar[2].

[1]Professor, Department of ECE, Narayana Engineering College, Gudur, AP, 524101.
[2]UG Student, Department of ECE, Narayana Engineering College, Gudur, AP,524101.
[2]Palaganijeevan5@gmail.com

---***---

Abstract – IoT is a global network with dynamic capabilities that use standard communication protocols. Security is the process of securing data as they are transmitted and exchanged among objects which can reduce the limitations. Cryptography is an important algorithm means in security that can be defined as converting plaintext into ciphertext.The algorithm of cryptography provides a high-level security but does not concern itself in hardware requirements like power consumption, computation cost, and memory overhead. To solve these challenges, special algorithms are designed and called light Weight Cryptography to be suitable for resource constrained devices. The goal of light Weight Cryptography is to reduce the overall implementation cost of traditional encryption, through several aspects such as area, throughput, latency, and power and energy consumption. Recently, several algorithms have been proposed for LWC. Comparing with different circumstances, we observed that PRESENT has much competence and many advantages. Several basic attacks PRESENT but all know attacks are ineffective. Given the importance of IoT security in this project, we are focusing on improving the S-Box implementation for essential measures such as power consumption and performance using Xilinx tool and spartan 7 and Artix 7 device.

Software Tools:

- Xilinx Vivado 2020.2
- Quartus Prime and Modelsim

Key Words:Cryptography, Lightweight Cryptography, SBOX, PBOX, Key Rounding, PRESENT Algorithm

1.INTRODUCTION

IoT is a technology in which a large number of devices are controlled over the Internet through programs or directly by consumers. The possibility of cyber-attacks or malicious attacks are highly common and sometimes it is uncontrollable if objects are connected via internet. The malicious attacks can directly affect the whole physical world. With the increasing demand for the Internet of Things (IoT) and its applications in today's markets and industry, the importance of security in device-to-device communication between edge nodes is becoming more and more critical.

The Cryptography is the field of cryptographic methods for securing the information and communication techniques where the plaintext is transformed into cipher text using a key generated by cryptographic algorithm. Implementing a conventional cryptographic environment would be impractical due to the constraints in the IoT device such as power dissipation, area and cost. The standard cryptographic algorithms can be of larger, very slow, and consumes more power.

Therefore, security solution at hardware level requires algorithm with small footprint which all comes under the energy budget. Lightweight cryptography gives a solution to security implementation in hardware level. These cryptographic methods specially established for embedded systems. The usage of IOT devices has increased worldwide due to lightweight engineering used in IOT devices, it also provides secure end-to-end communication under computation, limited memory and low power consumption.

2.AIM AND OBJECTIVE

To secure the information in this type of application, it is necessary to choose a scheme, where it is clear how the information will be encrypted, what kind of algorithm, how to share the secret, and generally the type of cryptographic scheme to use. In this case, we choose a symmetric encryption scheme, in which there is a single secret key, known to the sender and the receiver, this type of encryption has characteristics and advantages, and it has a very short execution time, which makes it available for use in high-speed applications and/or applications in which devices using low computational power.

3.LITERATURE SURVEY

This section provides the discussion on various existing researches in cryptographic technologies for security enhancement at hardware level.

The light weight block cipher PRESENT implementation on FPGAs was presented in research work of Sbeiti et al. The minimal hardware design of PRESENT cipher algorithm was exhibited. The outcome of the scheme produced more efficiency, so that PRESENT is well suitable for high throughput and high-speed presentations.

The S-box architecture for secure data and implementation of cryptographic hardware was presented in work of Rahaman et al. In this method C-testable S-box was invited for data encryption which is most complex block in hardware implementation so S-box structure was divided into a Read-Muller form and these are tested by using BIST circuit. The proposed architecture was efficiently evaluated against S-box functionality.

The FPGA implementation of the Ultra lightweight cipher PRESENT algorithm was evaluated in research work of Kavun and Yalcin. In the first design Sboxes were utilized within the slices and next design these all combined into the same RAM box used for state storage, which all more suitable for lightweight applications. The outcome of the proposed method produced reasonable throughput is 6.03kbps and 5.13kbps at 100 kHz, low cost and low area.

The work of Yalla and Kaps introduced lightweight cryptography for FPGAs. In this method block cipher independent optimization technique was presented for Xilinx Spartan3 FPGAs which all put on to the PRESENT and HIGHT lightweight cryptographic algorithm. With the analysis of proposed scheme, the ratio of throughput and area in PRESENT gives 240kbps/ slice and HIGHT with 720kbps/slice.

The system integration of AES and PRESENT coprocessors was performed in Guo et al. where the system outline analysis was performed by simulating the system model over FPGA based System on Chip (SoC). The outcomes of the system performance, energy and implementation were estimated for both PRESENT and AES which suggests that PRESENT less energy efficient than AES in a lightweight block cipher with lesser security level.

In this paper, we produced the SBOX implementations in a different coding style and compare them for power analysis.

4.PRESNET BLOCK CIPHER

Different lightweight block ciphers include KLEIN, LBlock, PRESENT, HIGHT, Piccolo, SPECK, and AES. Among these block ciphers, the PRESENT block cipher has a compact nature for hardware implementation and serves as a benchmark for the new hardware-oriented block ciphers, and its efficiency is higher.

The substitution box (S-box) is the only nonlinear part and essential constituent of different lightweight block cipher algorithms. During the process of encryption, it creates confusion in plaintext. For the improvement of the PRESENT algorithm, one S-box is chosen among 16 good S-boxes. It is shown that the PRESENT algorithm provides more security than the fixed present S-box.

The proposed design can be implemented using Verilog code and is simulated by ModelSim simulator and synthesized by Quartus prime tool. The performance and power metrics are estimated after logical synthesis, map, and place and route compilation by Xilinx Vivado2020.2 on the Xilinx Spartan-7 (devices). Here in this proposed algorithm, we mainly focused on Modelling SBox in different styles and the designs for state-of-the-art are evaluated and compared, by using Power, and utilization.

Present Algorithm:

The block cipher PRESENT is a Substitution Permutation Network with a block size of 64 bits and two key lengths of 80 and 128 bits are supported. We recommend the version with 80-bit keys for the applications we have in mind.

Present Encryption:

Each round of PRESENT consists of three stages: key addition, substitution layer, and bit-wise permutation layer.

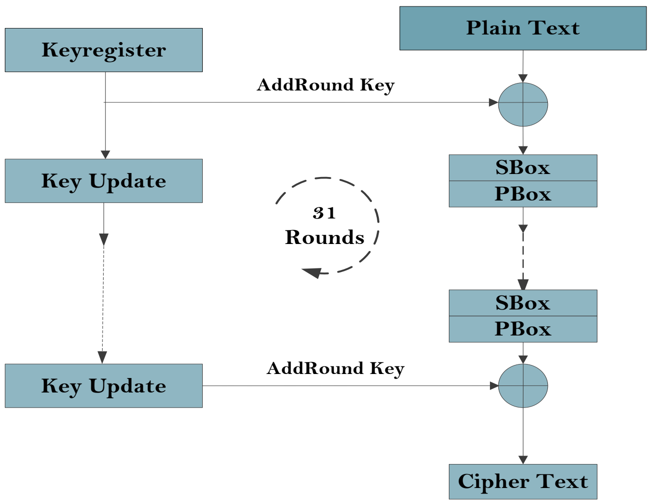

Fig 4.1: PRESENT encryption process

This encryption technique uses the Substitution-Permutation idea and contains 31 rounds. XOR operation is performed on each round to generate round key K_i for $0 < I < 31$.

There is a permutation layer (P-Box) and substitution layer (S-Box) based operation. A 4-bit S-box is used 16 times in parallel for each round in the substitution layer. The four functions included in this algorithm are s-box layer and p-layer, key scheduling, and add round key. 80 bit is given to the key scheduling block which generates 31 round keys for 31 individual rounds.

Step 1. Generate RK ( )

Step 2. for n=1,2,3, ……. 31

addRK(STATE, K_n)

SBox(STATE)

PBox(STATE)

Step 3. addRK(STATE, K_{32}); where RK is Round Key

Note:Consider Plain text as STATE and Kn as Key for Nth round

AddRoundKey(RK):

It is amongst the most vital blocks because it generates new keys for each round; yet, there was only an 80-bit implementation for this implementation, for that reason, a vector K of 80 positions was generated, as it can be seen as $K_{79}K_{78}...K_0$. In each round only the most significant 64 bits of the new K_i key of the next round will be mixed, how the rotation must be made is shown below:

$K_i = K_{63}K_{62}...K_0 = K_{79}K_{78}...K_{16}$………………..(1)

Then the following operations must be performed, in the order in which they have been presented: Bit rotation of the input key:

$[K_{79}K_{78}...K_0] = [K_{18}K_{17}...K_{20}K_{19}]$……………....(2)

Substitution by using S-Box for the 4 bits K_{78} to K_{76} of the key:

$[K_{79}K_{78}k_{77}K_{76}] = S[K_{79}k_{78}K_{77}K_{76}]$……………..…..(3)

Using the XOR technique, nibble adding or combining K_{19} to K_{15} of the key with the round counter:

$[K_{19}K_{18}K_{17}K_{16}K_{15}] = [K_{19}K_{18}K_{17}K_{16}K_{15}]$ ^ $RoundCounter$…………………………………......(4)

SBox:

The S-box used at present is a 4-bit input and 4-bit output

of each 16 S-Boxes.

The following table shows the action of the S-Box in the hexadecimal notation:

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 4.1: S-box from PRESENT algorithm

For SBoxLayer the 64 bits in state b_{63} . . . b_0 will be made into sixteen 4-bit words w_{15} . . . w_0

PBox:

It is a layer that combines a 64-bit information block employing bit-to-bit substitution, with Bit i of the round being shifted to the P position (i); The order of these substitutions is displayed in the table below.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Table 4.2: P-box from PRESENT algorithm

5.SIMULATION RESULTS AND DISCUSSION

SIMULATION RESULTS:

The system design is performed by using Verilog code and implemented on Artix-7, Spartan7, Artix-7 Low Voltage FPGA board. The verification of the designed system is verified by using a design supporting tool like Intel FPGA Quartus Prime and is simulated by using Mentor Graphics ModelSim and power simulation Is done by using Xilinx vivado 2020.2 Here we mainly focus to compare the Power analysis of SBOX using different Coding styles.

SBOX is 4-bit width which is given 4-bit input and it generates 4-bit output.

Note that during the design phase of PRESENT we use the same S-box 16 times rather than having 16 different S-boxes and this eases a further serialization of the design. The simulation results obtained for SBOX for PRESENT CIPHER are shown in figure 6.1.

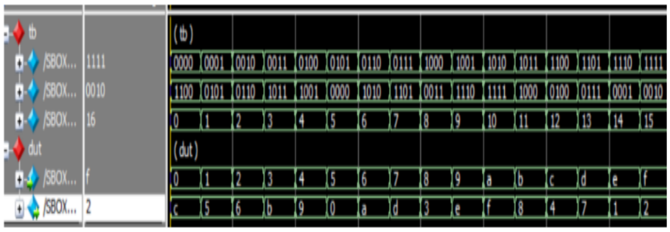

Figure 5.1: Simulation results for 4-bit SBOX

The simulation results obtained from the KEY Rounding (80bit key) are given. If the 31st cycle reaches high as k load then reaches low in the next cycle. The process begins with the parallel key updating process. Once the key updating process is completed, the obtained Key output (Kout) is generated on simulation results.

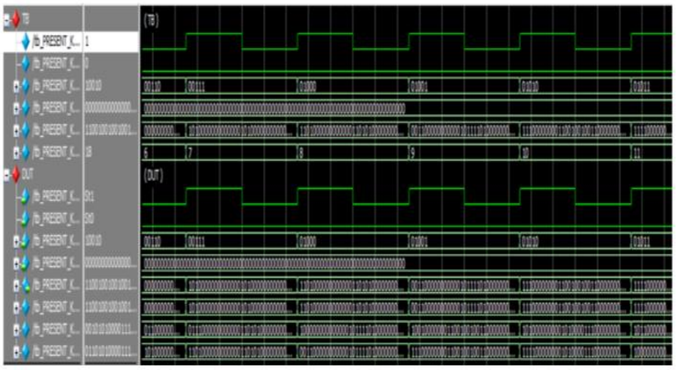

Figure 5.2: Simulation results for Key Updating

```
key = 00000000000000000000 round = 00000 out = xxxxxxxxxxxxxxxxxxxx
key = 00000000000000000000 round = 00000 out = 00000000000000000000
key = 00000000000000000000 round = 00001 out = c0000000000000008000
key = 00000000000000000000 round = 00010 out = 50001800000000010000
key = 00000000000000000000 round = 00011 out = 60000a00030000018000
key = 00000000000000000000 round = 00100 out = b0000c00014000620000
key = 00000000000000000000 round = 00101 out = 900016000180002a800c
key = 00000000000000000000 round = 00110 out = 0001920002c000330005
key = 00000000000000000000 round = 00111 out = a000a0003240005b8006
key = 00000000000000000000 round = 01000 out = d000d4001400064c000b
key = 00000000000000000000 round = 01001 out = 30017a001a80028480c9
key = 00000000000000000000 round = 01010 out = e01926002f4003550050
key = 00000000000000000000 round = 01011 out = f00a1c0324c005ed806a
key = 00000000000000000000 round = 01100 out = 800d5e014380649e00bd
key = 00000000000000000000 round = 01101 out = 4017b001abc028768c93
key = 00000000000000000000 round = 01110 out = 71926802f600357f050e
key = 00000000000000000000 round = 01111 out = 10a1ce324d005ec786af
key = 00000000000000000000 round = 10000 out = 20d5e21439c649a80bd8
key = 00000000000000000000 round = 10001 out = c17b041abc4287304935
key = 00000000000000000000 round = 10010 out = c926b82f6083578150e6
key = 00000000000000000000 round = 10011 out = 6a1cd924d705ec19eaf0
key = 00000000000000000000 round = 10100 out = bd5e0d439b249aeabd83
key = 00000000000000000000 round = 10101 out = 07b077abc1a8736e135d
key = 00000000000000000000 round = 10110 out = 426ba0f60ef5783e0e6d
key = 00000000000000000000 round = 10111 out = 41cda84d741ec1d52f07
key = 00000000000000000000 round = 11000 out = f5e0e839b509ae8fd83a
key = 00000000000000000000 round = 11001 out = 2b075ebc1d0736adb5d1
key = 00000000000000000000 round = 11010 out = 86ba2560ebd783ade6d5
key = 00000000000000000000 round = 11011 out = 8cdab0d744ac1d777075
key = 00000000000000000000 round = 11100 out = 1e0eb19b561ae89b83ae
key = 00000000000000000000 round = 11101 out = d075c3c1d6336acddd13
key = 00000000000000000000 round = 11110 out = 8ba27a0eb8783ac96d59
key = 00000000000000000000 round = 11111 out = 6dab31744f41d7008759
```

Figure 5.3: Transcript view for Key updating

The simulation results obtained from the PRESENT cipher model (80bit key) are shown. If the 31st cycle reaches high as k load then reaches low in the next cycle. Later, d load is extended to logic high and the encryption of the data input (din) process begins with the parallel key updation process. Once the key update process is completed, the obtained data output (dout) is generated on simulation results.

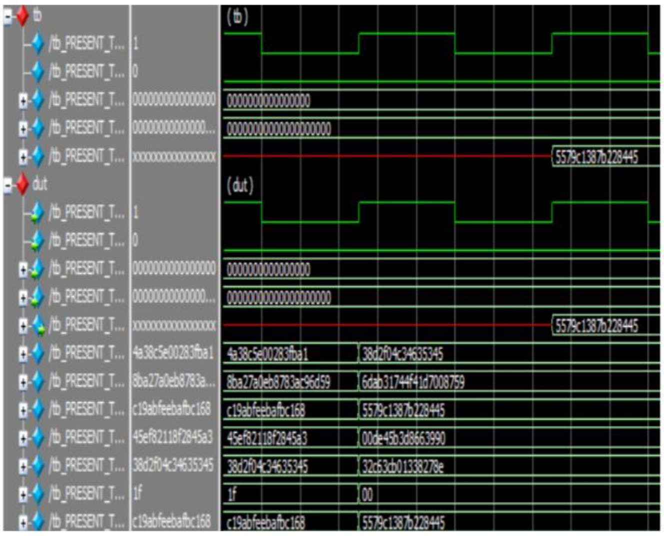

Figure 5.4: Present-80 Simulation results with D_{in} and Key given as “0”

```
load = 0 round = xxxxx plain_text = xxxxxxxxxxxxxxxx, Key = xxxxxxxxxxxxxxxxxxxx
load = 1 round = xxxxx plain_text = xxxxxxxxxxxxxxxx, Key = xxxxxxxxxxxxxxxxxxxx
load = 0 round = 00001 plain_text = 0000000000000000, Key = 00000000000000000000
load = 0 round = 00010 plain_text = ffffffff00000000, Key = c0000000000000008000
load = 0 round = 00011 plain_text = 80ff00ffff008000, Key = 50001800000000010000
load = 0 round = 00100 plain_text = 4036c837b7c88c09, Key = 60000a00030000018000
load = 0 round = 00101 plain_text = 73c2cd26b6192359, Key = b0000c00014000620000
load = 0 round = 00110 plain_text = 41d7be58531e4446, Key = 900016000180002a800c
load = 0 round = 00111 plain_text = 182ef861ad62fd1c, Key = 0001920002c000330005
load = 0 round = 01000 plain_text = 0ea0a5b67effc5a4, Key = a000a0003240005b8006
load = 0 round = 01001 plain_text = bba0b848a113e080, Key = d000d4001400064c000b
load = 0 round = 01010 plain_text = fa943423a9142338, Key = 30017a001a80028480c9
load = 0 round = 01011 plain_text = 69f2e22d63684d54, Key = e01926002f4003550050
load = 0 round = 01100 plain_text = 548a4b63c330a59d, Key = f00a1c0324c005ed806a
load = 0 round = 01101 plain_text = d75f955fa228e4ca, Key = 800d5e014380649e00bd
load = 0 round = 01110 plain_text = 442558641038 41f9, Key = 4017b001abc028768c93
load = 0 round = 01111 plain_text = e2cc9004363f6c12, Key = 71926802f600357f050e
load = 0 round = 10000 plain_text = c36682c5cd375421, Key = 10a1ce324d005ec786af
load = 0 round = 10001 plain_text = 597db55cc2a5d9b6, Key = 20d5e21439c649a80bd8
load = 0 round = 10010 plain_text = e67ce40e71b8b713, Key = c17b041abc4287304935
load = 0 round = 10011 plain_text = 751df6d6807b5b59, Key = c926b82f6083578150e6
load = 0 round = 10100 plain_text = b948414e23332c93, Key = 6a1cd924d705ec19eaf0
load = 0 round = 10101 plain_text = 5b75890dcfb3d563, Key = bd5e0d439b249aeabd83
load = 0 round = 10110 plain_text = 5679203168278f5a, Key = 07b077abc1a8736e135d
load = 0 round = 10111 plain_text = 17c377c413fa45a3, Key = 426ba0f60ef5783e0e6d
load = 0 round = 11000 plain_text = 262a2de73b5f3ecd, Key = 41cda84d741ec1d52f07
load = 0 round = 11001 plain_text = d3a053128b4d7bb3, Key = f5e0e839b509ae8fd83a
load = 0 round = 11010 plain_text = 7db29209c28a20fa, Key = 2b075ebc1d0736adb5d1
load = 0 round = 11011 plain_text = 62050c9940f400b9, Key = 86ba2560ebd783ade6d5
load = 0 round = 11100 plain_text = 65d50da21fbcc09f, Key = 8cdab0d744ac1d777075
load = 0 round = 11101 plain_text = 6a50663c540d862f, Key = 1e0eb19b561ae89b83ae
load = 0 round = 11110 plain_text = c79b8ff00a48df35, Key = d075c3c1d6336acddd13
load = 0 round = 11111 plain_text = 4a38c5e00283fba1, Key = 8ba27a0eb8783ac96d59
Ciphertext = xxxxxxxxxxxxxxxx
Ciphertext = 5579c1387b228445
```

Figure 5.5: Transcript view for Present-80 Simulation results

Synthesis Results:

As said SBOX is implemented in different methods like using Multiplexer, LUT based and using only logic gates. The RTL of the proposed models of SBOX is shown below figures.

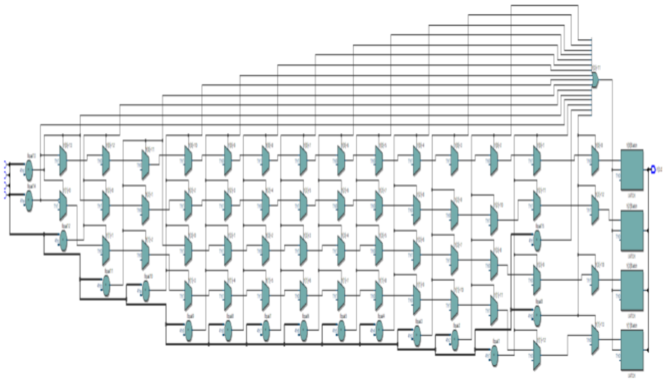

Figure 5.6: RTL for SBOX using Multiplexer

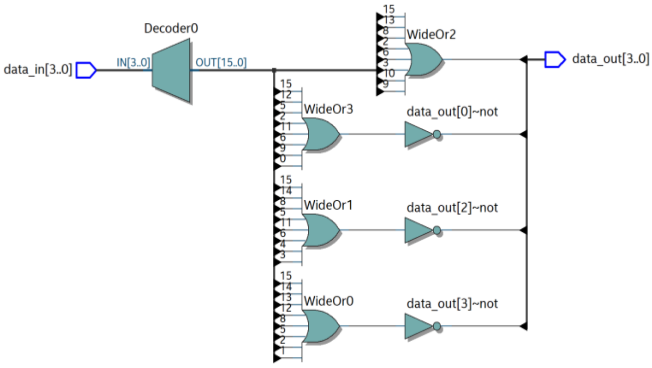

Figure 5.7: RTL for SBOX using LUTs

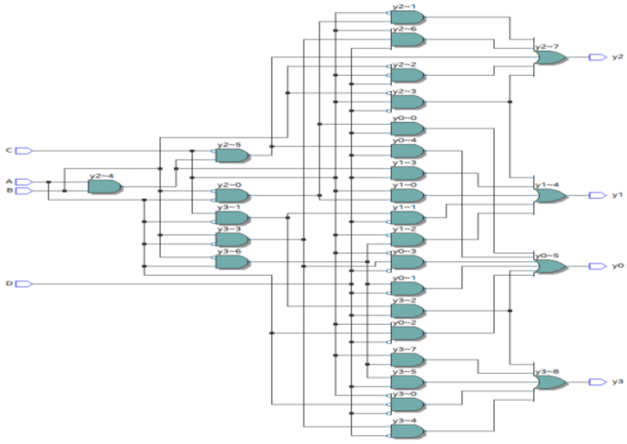

Figure 5.8: RTL for SBOX using Logic Gates

The RTL for key Rounding is shown in below figure 6.9

The execution of the system gives significant results respective to outputs. The RTL of the proposed PRESENT cipher model is given below, which contains 64bit data input (din), 80bit key input, clock (CLK), d load, k load, and yields a corresponding output of 64bit (d_{out}).

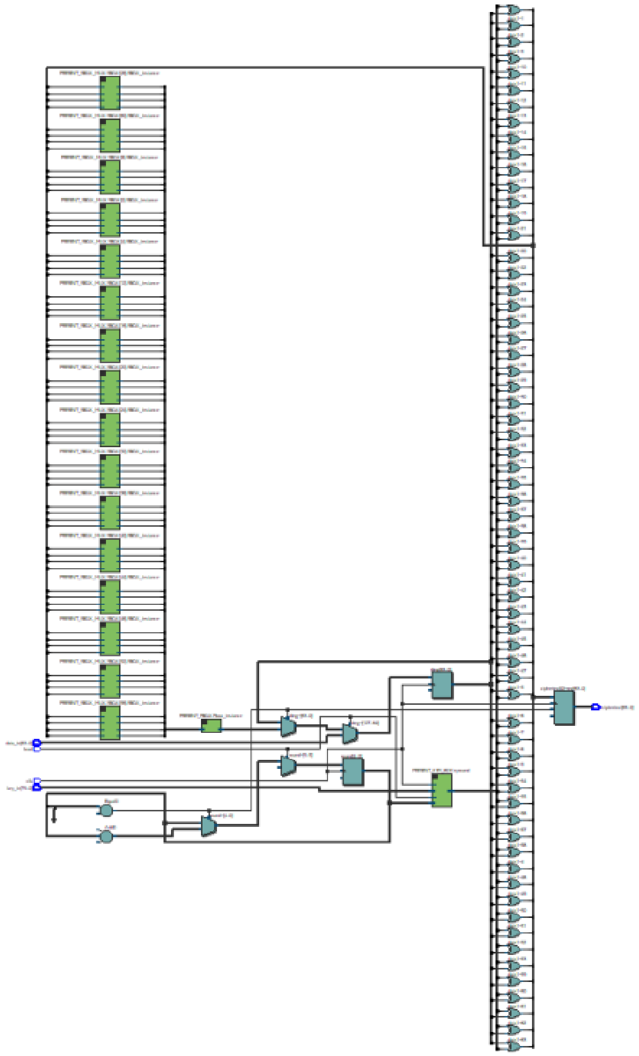

Figure 5.10: Top module of PRESENT cipher model

Power Analysis:

The Power analysis of the proposed cipher model is performed by comparing the proposed models of SBOX and PRESENT Block cipher using these SBOX. The target device utilized for comparison proposed PRESENT cipher model is Artix-xc7a50tcsg324-2L and spartan-xc7a50csga324-2.

Fig 5.11: Power Analysis for SBOX using Gates

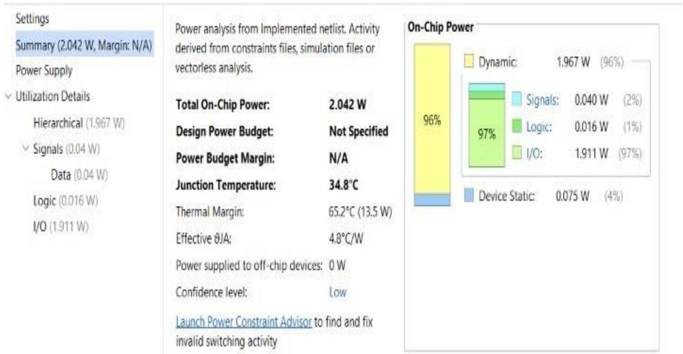

Fig 6.12: Power Analysis for SBOX using LUT's

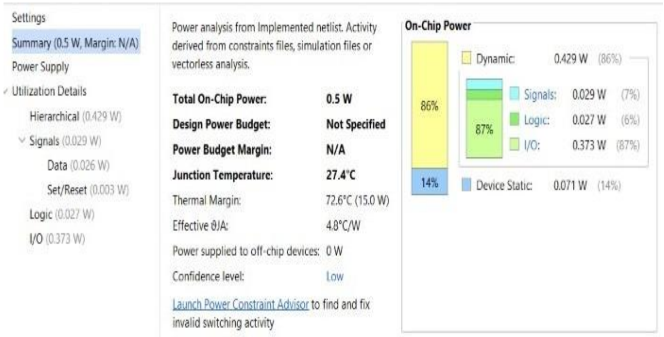

Fig 5.13: Power Analysis for SBOX using LUT's

The comparative results are shown in the below tables.

Power (mW)	SBOX Gate Model	SBOX LUT Model	SBOX MUX Model
Total on chip power	2042	2040	500
Heirarcal or Dynamic	1967	1965	429
Signals	40	39	29
Data	40	39	26
Logic	16	15	27
I/O	1911	1911	373
Static	75	75	71

Table 5.1: Comparison of Power between different SBOX Models

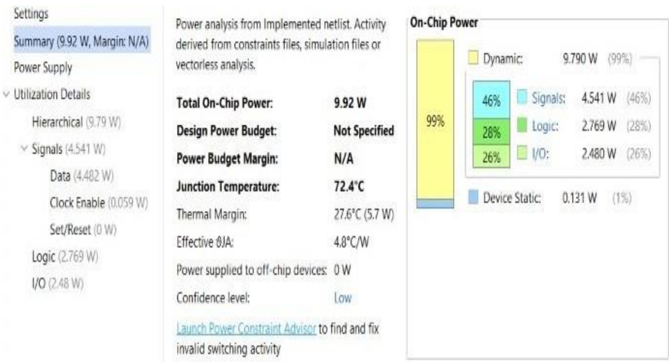

Fig 5.14: Power Analysis for PRESENT Top using Gated SBOX

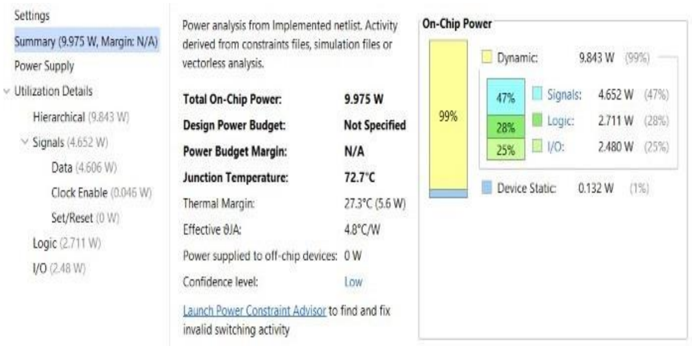

Fig 5.15: Power Analysis for PRESENT Top using LUT SBOX

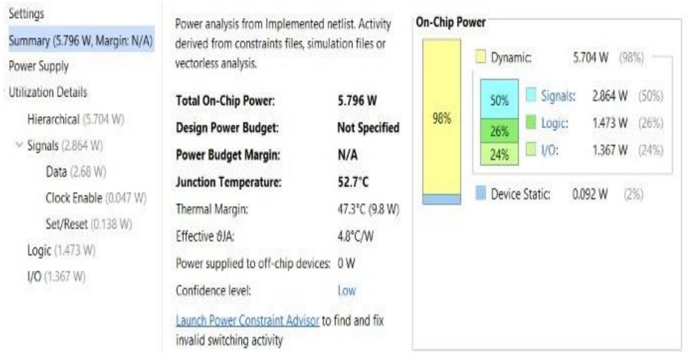

Fig 5.16: Power Analysis for PRESENT Top using MUX SBOX

The comparative results for PRESENT Block cipher Top Modules are shown in the below tables.

Power (mW)	Top using Gated SBOX	Top using LUT SBOX	SBOX MUXed SBOX
Total on chip power	9920	9975	5796
Heirarcal or Dynamic	9790	9843	5704
Signals	4541	4652	2864
Data	4482	4606	2680
Logic	2769	2711	1473
I/O	2480	2480	1367
Static	131	132	92

Table 5.2: Comparition of Power between PRESENT TOP Modules

6. CONCLUSIONS

This paper introduces the hardware architecture implementation of the PRESENT cipher model using different models of SBOX. The design of the PRESENT cipher model is performed by using Verilog programming language on Xilinx Vivado 2020.2 platform implemented over Artix-7 and spartan FPGA for encryption through an 80bit key module. In the proposed PRESENT cipher model, a 64bit data path enables the entire round execution in a single cycle. This execution operation requires a 64bits permutation layer as well as sixteen S-box layers of 4bits. In this paper, the 80bit key-based PRESENT cipher model using different SBOX implementations is compared to analyse the power between each and achieved up to x4 times reduction.

FUTURE SCOPE

Future Work As technology is emerging people are looking for more speed in addition to power and memory in future, we can focus on implementing PRESENT algorithm Using Pipelining method in different stages so that we can achieve more speed. Considering the emerging technology, the ASIC devices are also gradually coming under more resource constraints devices due to huge requirement of security in these devices in future we research to implement the PRESENT Block Cipher algorithm in ASIC design and try to verify in chip level.

REFERENCES

1. Santosh Pandurang Jadhav." Towards Light Weight Cryptography Schemes for Resource Constraint Devices in IoT". Published in Journal of Mobile Multimedia, Article 5, volume 15, Page 91-110, 2019.

2. Sumit Singh Dhanda, Brahmjit Singh and Poonam Jindal." Lightweight Cryptography: A Solution to Secure IoT". Springer, 2020.

3. HengamehDelfan Azari, Dr. Prashant V Joshi. "AN EFFICIENT IMPLEMENTATION OF PRESENT CIPHER MODEL WITH 80 BIT AND 128 BIT KEY OVER FPGA BASED HARDWARE ARCHITECTURE". International Journal of Pure and Applied Mathematics Volume 119, Article No. 14, 2018.

4. Bogdanov, A. et al." PRESENT: An Ultra-Lightweight Block Cipher". vol 4727. Springer, Berlin, Heidelberg, 2007.

5. Suzan Sallam, Babak D. Beheshti." A Survey on Lightweight Cryptographic Algorithms". IEEE, 2018.

6. Bharathi R; Bhagya R; Anjan K Koundinya." FPGA Based Lightweight Encryption Algorithm for Cyber Security Applications". IJCSN Journal Volume 9, Issue 4, 2020. http://ijcsn.org/IJCSN-2020/9-4/FPGA-Based-Lightweight-Encryption-Algorithm-for-Cyber-Security-Applications. pdf

7. Gaoli Wang, ShaohuiWang."Differential Fault Analysis on PRESENT Key Schedule". IEEE, 2011.

8. Amir Fotovvat, Gazi M. E. Rahman, SeyedShahimVedaei, Khan A. Wahid." Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Node". IEEE INTERNET OF THINGS JOURNAL, VOL. 8, NO. 10, MAY 15, 2021.

9. Edwar Jacinto G, Holman Montiel A and Fernando Mart´ınez S." Implementation of the cryptographic algorithm "Present" in different microcontroller type embedded software platforms". International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8092-8096.

10. Vishal A. Thakor, Mohammad Abdur Razzaque, Muhammad R. A. Khandaker." Lightweight Cryptography Algorithms for Resource-Constrained

IoT Devices: A Review, Comparison and Research Opportunities". IEEE Access (Volume: 9), 2021.

11. P.Rajasekar, H Mangalam (2021), "Design and implementation of power and area optimized AES architecture on FPGA for IoT application", Circuit World, ISSN 0305-6120 Vol. 47 No. 2, pp. 153-163 https://doi.org/10.1108/CW-04-2019-0039
12. P.Rajasekar & H.Mangalam (2015),"Design and Implementation of Low Power Multistage AES S Box", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 19 (2015) pp 40535-40540
13. U.Ambili, P.Rajasekar, (2017), "Area and Power Optimized Lightweight PRESENT-GRP Cryptography Algorithm",International Journal of Emerging Innovations in Science and Technology(ISSN:2348-439X) Volume 3, issue4.
14. P.Rajasekar & H.Mangalam (2016),"Efficient FPGA implementation of AES 128 bit for IEEE 802.16e Mobile WiMax standards", Circuits and System,2016, Volume 7, Page 371-380.
15. N. Lavanya , Shaik Mehanaaz, N.Blalji, Dr.P.Rajasekar (2019), Low Power and Area Optimised Present GRP Cryptography Algorithm, Journal of Analysis and Computation (JAC) , ISSN 0973-2861, Volume XII, Issue I, Jan-June 2019
16. Penchalaiah P, Rajasekar P, Srinivas Viswanth V and Ramesh Reddy(2020), "An Efficient Multi-User Hierarchical Authenticated Encryption Using Simultaneous Congruence For Highly Secure Data",International Journal of Future Generation Communication And Networking, ISSN/eISSN 2233-7857/2207-9645, 30 Jun 2020, vol. 13, no.2 , pp. 1-10, 10.33832/ijfgcn.2020.13.2.01
17. Imdad, M., Ramli, S.N., Mahdin, H." An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys". Symmetry 2022, 14, 604.,