International Journal of Future Generation Communication and Network
Vol. 14, No. 1, (2021), pp. 592-601

FPGA Implementation of Proficient Lightweight Architecture for Present Block Cipher

[1]Sunitha Tappari, [2] K. Sridevi
[1] G. Narayanamma institute of Technology and science
[2]GITAM University
*Corresponding author E-mail: sunitha.tappari@gmail.com

Abstract
Cryptography is a mathematical based technology and extremely rich to ensure that several concepts of the information security over a public channel. They have been deployed in diverse areas for every conceivable purpose. Present days the cryptography plays an essential role for secured data transmission. The proposed design has high performance Lightweight VLSI architecture for block cipher with 2x80-bit key. This architecture increases the throughput by reducing the latency up to 50% from the existing architectures, so that it can improve the performance of the designed architecture. Here it has been observed that the number of clock cycles of latency is 32 cycles. The FPGA with onboard clock frequency of 250 MHz achieved the throughput of 1Gbps. The entire architecture is designed using Verilog HDL. The simulation, synthesis and implementation on FPGA is done with Xilinx ISE design suite.
Keywords: Cryptography, Block cipher, FPGA, Lightweight.

1. Introduction

The modern-day cyber physical systems (CPS) and internet of things (IoT) infrastructures heavily rely on extensive deployment of tiny computing devices for sensing, computing, controlling and communication purposes [1], [2]. Scope of these devices is widespread; ranging from consumer items to virtually anything. These devices form a pervasive computing infrastructure with an intelligent ecosystem. Uninterrupted system availability, minimal power consumption, adequate level of data security, resource-efficient hardware architectures, low cost and quick time-to-market are the essential desirables of this ecosystem. Insatiable demands on the system design metrics make the system development task more complex and challenging. In emerging applications such as smart cities, smart grid, electronic bank transactions, digital locker, connected cars, etc., secure communication is utmost essential. It requires a mechanism, which ensures that unauthorized persons or machines cannot access the communicated information. For securing electronic data communication, cryptography plays an essential role. It is a technique which ensures secrecy and the authenticity of electronic data transfer in any insecure channel. In cryptography, the encryption operation is used to convert data into a secure form, known as cipher-text.

Cryptography is an art of science, distinguished an extensive history since from the time of the ancient Greeks! It aims to hide the information message through over a public or secret communication channel and that applies principles of mathematical algorithms and logic to design strong encryption methods are protected by large numbers of possible keys. Some cryptographic algorithms in use today are also safe to use in a real-world. The cryptographic process is used for authentication in many emerging applications such as in bank cards, wireless telephones, e-commerce, pay-TV, prepaid telephone cards, e-cash cards, etc. It is also used for making the access control in many systems such as car locks, lifts, metro-trains, electronic gadgets and many other form of embedded systems. In these omnipresent smart devices, there is always a need of high performance implementation of lightweight cryptographic algorithms for ensuring security in resource constrained environment. Hardware-based security solutions with symmetric key cryptography algorithms are ideally suited to meet the IoT security challenges for very low area and energy requirements [2]. Thus, resource-constrained hardware architectures of lightweight ciphers are very essential. A systematic survey of lightweight-cryptography ciphers and their software and hardware implementations can be found in [3]. In the survey it has been emphasized that efficient implementation of the ciphers are closely dependent on the selection of appropriate architectures as they result in low implementation complexity and high performance in actual realization. In the context of lightweight cryptography, ISO/IEC 29192-2 has standardized symmetric block cipher algorithm PRESENT in the year 2012 [4]. It provides adequate security goals along with hardware-oriented performance attributes which makes it a prominent choice for developing lightweight cryptographic applications [5]. FPGA-based platforms have been commonly deployed for architectural exploration, rapid prototyping and quick evaluation of area-performance tradeoffs across different set of architectures. In this context some of the architectures for the PRESENT cipher and their FPGA implementations have been described below.

2. Literature survey

A. Low area architecture:

The first area-optimized architecture of PRESENT was proposed by its creators in [5-7]. The optimization strategy for this architecture is to reduce the number of substitution boxes, creating a direct trade-off between utilized resources and latency. The corresponding hardware architecture is the one shown in Fig.1. There are two disadvantages in that proposal. First, the variation of size in the data path width to process the state requires additional logic for routing and control which in turn induces an area overhead that could reduce the efficiency of the solution. Second, the reduction of the substitution layer to decrease the resource consumption would increase the latency cycles which can be prohibitive for certain applications.

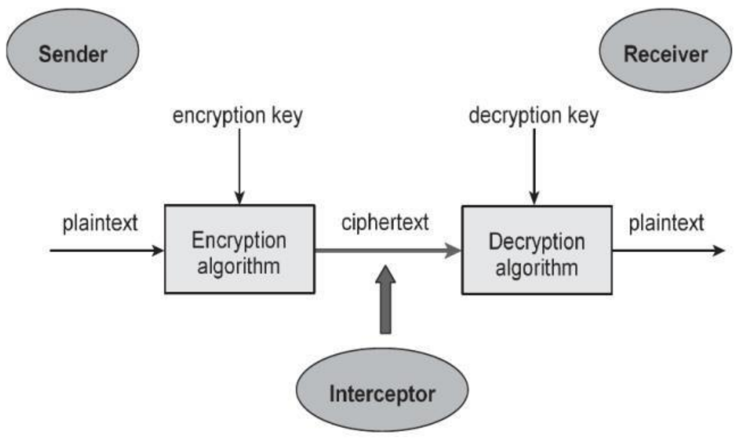

Fig.1 Block diagram of basic Encryption and Decryption

The latency for this design depends on the number of SBOX utilized. In the case where two SBOX are used, if the design considers all the required ports, two cycles are needed to take the input and produce the output, plus 8 × 31 cycles to process the state. In total, 250 cycles are necessary in a 2-SBOX configuration. In terms of area, the total count for the proposed design can be expressed as 2-input NAND gates (considered equivalent to 1 GE [8]). The cost of D-type Flip Flops (FF) and XOR gates is obtained through the equivalent circuit, considered to be of 5 and 4 GE, respectively. In silicon technology, the shifts and permutations are regarded to have no cost [9]. For the equivalence of a 4-bit substitution box, the reader can refer to the estimation provided in the original proposal of PRESENT [9], which is said to be of 28 GE approximately. If using two substitution boxes to process the data path this architecture can be constructed with: 149 FFs (745 GE) for the state, key, and counter registers; 69 XORs (276 GE) to add the round key with the state and the round counter with the round key; three substitution boxes (84 GE) for the data path and the key schedule; a 64-bit permutation (0 GE) for the state; and a 61-bit shift (0 GE) for the round key. In total 1,105 GE are needed, approximately.

B. Iterative Architecture:

This architecture is reported in [10]. This design uses 8-bit I/O data ports (8-bit for the plaintext, 8-bit for the key and 8-bit for the output) with a key size of 128-bit. The state is stored in a single 64-bit register and the key is stored in a single 128-bit register, both of these registers support multiple bit shifts and parallel inputs. This architecture is illustrated in Fig. 2. To process a 64-bit plaintext block, 16 cycles are required to load the data, plus 31 cycles of latency for encryption, and 8 cycles to produce the output, leading to a latency of 55 cycles. As appreciated in Fig. 3 this design utilizes 197 FF (985 GE), 77 XOR gates (308 GE), 18 SBOX (504 GE),

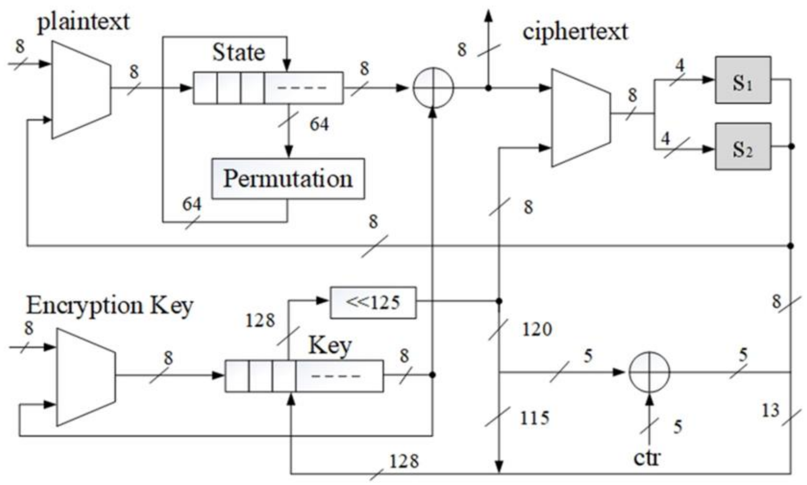

Fig. 2. Serial architecture PRESENT

a 64-bit permutation (0 GE), and a 61-bit shift (0 GE). That is, the design reported in [10] has a cost in area of 1797 GE approximately.

C. Serial Architecture:

Neil Hanley and Marie O'Neill (2012) [10]. This is an area optimized implementation of PRESENT using a 128-bit key [11]. The input and storage mechanisms for the state and key data work similarly to those of the iterative architecture. The optimization strategy is based on reducing the number of substitution boxes in the substitution layer to two. The substitution boxes in the key schedule are also replaced by those in the substitution layer. To achieve this replacement, 8 cycles per round are required to process the 64-bit state and during a 9th cycle in which the 64-bit permutation takes place, the round key is also updated. Fig.2 illustrates this proposal. This design requires 16 cycles to load the data, 279 cycles to process the state, and 8 cycles to produce the output, this is a total latency of 303 cycles. From Fig. 4 it can be noted that this design can be constructed using 197 FFs (985 GE), 13 XOR gates (52 GE), 2 SBOX (56 GE), a 64-bit permutation (0 GE), and a 125-bit shift (0 GE). This produces an approximate count of 1093 GE.

D. Serial Architecture with Boolean

SBOX – J. J. Tay et al. (2015) [11]:

In this proposal the authors claim to achieve a lightweight implementation of PRESENT by following the design of the serial architecture [11], and replacing the substitution boxes with a construction based on boolean logic [19]. The authors attempted to construct the PRESENT SBOX using logic gates. The design is achieved using Karnaugh mapping and factorization requiring 26 AND gates and 17 OR gates. This design has the same latency as the serial architecture. The reasoning for this optimization relies on the premise that a BRAM-based S-Box is rigid, so their proposal attempts to reduce the S-Box design through simplification of regular expressions. It is important to note that for FPGA technologies this kind of strategy would yield poor results, since in a conventional implementation process the synthesis tool tends to map the SBOX in the same way, regardless if it is described as a lookup- table or as a boolean construction. The estimation of gate equivalents for this architecture is similar to that of the serial architecture with the difference in the construction of the SBOX. Using the count of AND and OR gates provided by the authors, an SBOX designed this way would require 103 GEs. Then this design can be constructed using 197 FFs (985 GE), 13 XOR gates (52 GE), 2 SBOX (206 GE), a 64-bit permutation (0 GE), and a 125-bit shift (0 GE). This produces an approximate count of 1243 GE keying material in the architecture and to allow the synthesis tool to generate the combinational design that produces the round keys. This is interesting for this specific design since the width of the round key is reduced from 64 to 16-bit, which enables a reduction in the complexity of the combinational process. Under this approach, it is required to calculate the whole key set beforehand and to describe it as a ROM module. If it is specified that the FPGA cannot use memory blocks to implement this module, the synthesizer will be forced to use LUTs to create a combinatorial block capable of generating each one of the round keys required by the cipher. The main advantages of this design are: it has a reduced latency because the key is not entered to the circuit and the associated clock cycles (one for 80-bit keys and four for 128-bit keys) are avoided, and there is no need for extra registers to store the key since it can be read directly from the key space. This approach, however, can raise some security concerns as it is possible that side channel vulnerabilities allow the unauthorized retrieval of keying materials [12]–[14]. In this architecture the cipher's data path can be constructed using 80 FF (400 GE), 16 XOR (64 GE), 4 SBOX (112 GE), and a 16-bit permutation. This produces an estimate cost for the data path of 576 GE. It is difficult to estimate the total resource usage in GE for this design, since the key module is an architecture that can be considered as a black box generated by the synthesis tool.

3. Proposed Architectures:

The base design for the two architectures proposed in this paper follow the strategies reported in [15] in relation to the construction of the data path. The data path design is illustrated:

3.1. Pipeline process

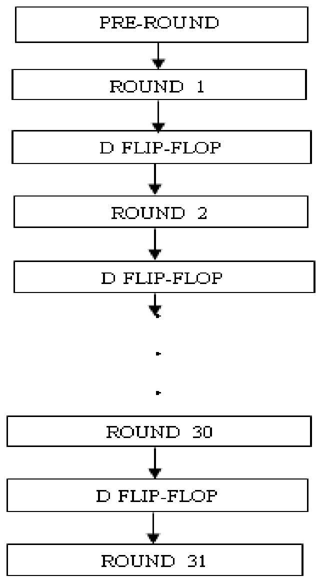

Fig 3: Flow diagram of pipeline process

Using pipeline process the optimization is achived in latency and improved the overall processing performance. This architectural approach allows the simultaneous execution of several instructions. Pipelining is transparent to the programmer; it exploits parallelism at the instruction level by overlapping the execution process of instructions.

Parallelization Process Operation:

This design has two control inputs namely key_load and data_load. When either of these inputs is high, at the rising edge of the clock signal (clk) the (required number of) bits present at data1,data2 are copied to the corresponding register i.e state1 and state2. In case of key_load all bits of data_i (which is 80 bits wide) are copied to key register (Key1,key2) (which is also 80 bits wide). In case of data_load, 64 rightmost bits (data1,data2[63:0]) of data1,data2 are copied to state registers. One other event that happens when data is loaded (i.e. data_load = '1') is that round_counter i.e round_counter1, round_counter2 is set to '1' (5'b00001) i.e. loading a new plaintext into the state register also resets the state machine, that is why this design does not have a reset signal. After loading the key and the plaintext, both key_load and data_load must be '0'. After setting the control inputs to '0', 32 clocks must be applied in order to data_o1,data_o2 contain a valid ciphertext. That is after the rising edge of the 32^{nd} clock signal, data_o has the correct ciphertext. So the complete encryption of plaintext requires 32 clocks. encrypting plaintexts will happen in parallel order using the same blocks and achieve faster cipher text generation at end of 32 clocks. This mechanism is being followed at processor level where two parallel processors will deal to do parallel operations with single control and helps to do multiple data process at same time. Following steps briefly summarize the operation of this design:

Load the key (key_load = '1', data_load = '0')

Load the plaintext (key_load = '0', data_load = '1')

Set control inputs to '0' (key_load = '0', data_load = '0')

Apply 32

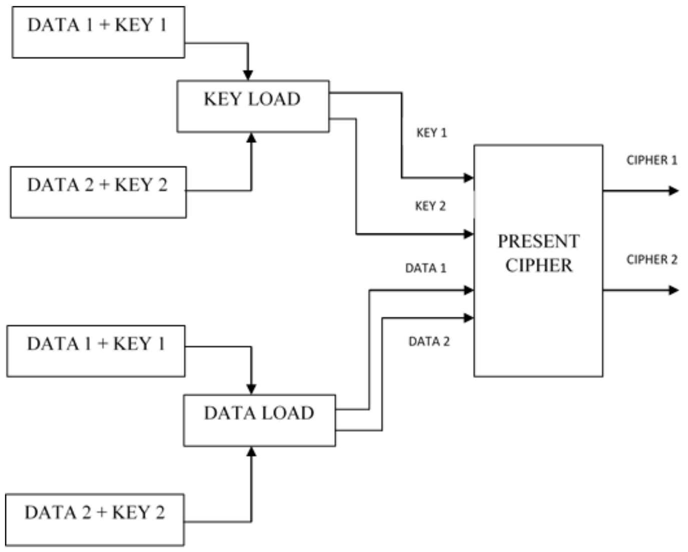

Fig: 4. Block diagram of the proposed Architecture

3.2. Present Block Cipher:

PRESENT is an example of an SP-network and consists of 31 rounds. The block length is 64 bits and based on the applications this design has 80-bit key length. Each round utilizes an XOR operation for total 31 round keys to introduce a round key *Ki* for $1 \leq i \leq 32$, where *K* 32 is used for post-whitening, a linear bitwise permutation and a non-linear substitution layer. The non-linear layer uses a single 4-bit S-box *S* which is applied 16 times in parallel in each round. The cipher is described in pseudo-code in Fig.5. and each stage is now specified in turn. We have assigned bits starting from zero with bit zero as on the right of a block or word.

```
generateRoundKeys()
for $i = 1$ to 31 do
    addRoundKey(STATE, $K_i$)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE, $K_{32}$)
```

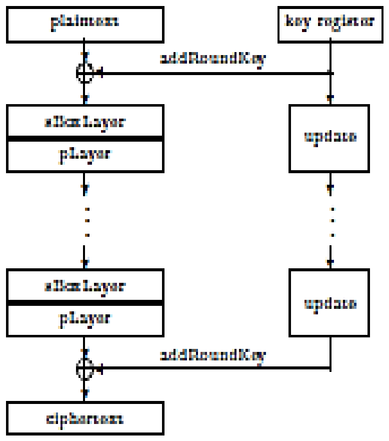

Fig.5. A top-level algorithmic description of *PRESENT*

5. Experimental results

A low- latency technique is added to the architecture to increase the performance and calculated the throughput. The maximum throughput (Thr) of an implementation is a function of the maximum operational frequency (F_{MAX}), the latency cycles required to process a block (LAT), and the block size (B_{SIZE}). It is calculated with Equation.

$$\text{Thr} = \frac{F_{max} \times B_{size}}{LAT}$$

The latencies of the proposed architecture are reduced by using parallelization method and obtained latency of the proposed design is 32 cycle, existing design has a latency of 64 cycle and reduced 50 percent latency in the proposed design. Fig.6illustrate the simulation results of the existing design. Fig.7 shows the simulation results of proposed design.

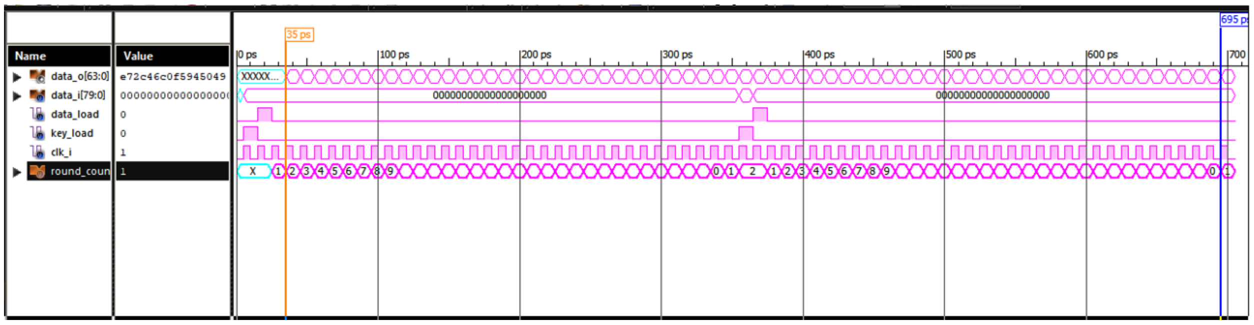

Fig 6. Two successive series inputs without parallelization, latency = 64 cycles

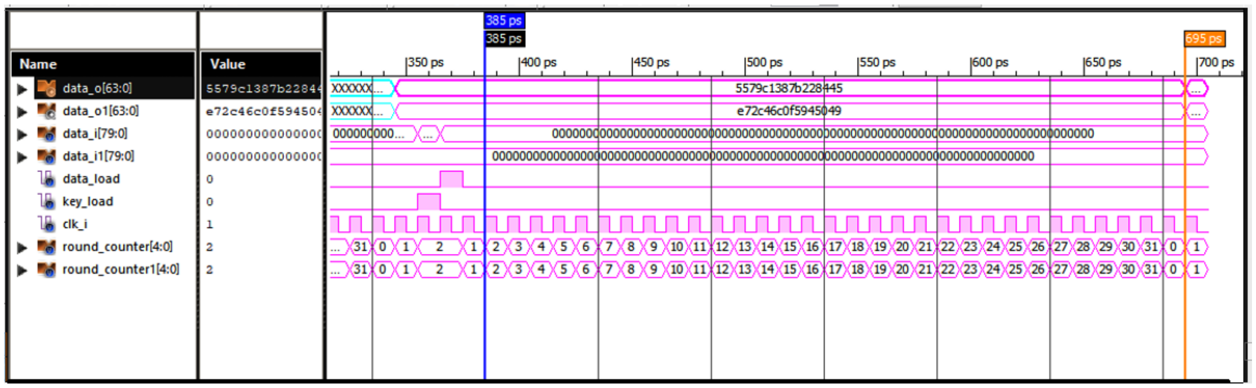

Fig 7. Two data inputs Parallelization Process, latency = 32 cycles

Table 1. Comparison results of different hardware architectures

Parameter	Proposed work	Existing [17]	Existing [18]
Key size	160	80	128
Area(SLC)	421	48	44
Latency(cycles)	32	136	303
Throughput(MHz)	1000	73.5	33.00

Table.1 shows the comparison of different hardware architectures. The latency of proposed design is very less when compare with the existing design and throughput rate is very high when compare with existing designs.

6. Conclusion

This paper presented a FPGA implementation of lightweight architecture for present block cipher of hardware architectures A 2 x 64-bit data-path architecture with 2x80-bit key schedule was presented. The simulation results show that the latency of the proposed design is very less when compared with all other existing designs. The throughput rate of the proposed design is very high when compare with the serial architecture and iterative architectures. The entire design is implemented on SPARTAN-6 XC6SLX16-3csg324 FPGA, with the help of Xilinx ISE 14.7 CAD tool.

References:

[1] United States National Intelligence Council, "Six Technologies with Potential Impacts on US Interests out to 2025," *Disruptive Civil Technologies*, 2008. [Online]. Available: http://www.dni.gov/

[2] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT Systems: Design Challenges and Opportunities," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 417–423.

[3] D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. Grossch¨adl, and A. Biryukov, "Triathlon of Lightweight Block Ciphers for the Internet of Things," *IACR Cryptology ePrint Archive*, vol. 2015, p. 209, 2015.[Online]. Available: http://eprint.iacr.org/2015/209

[4] M. Kneˇzevi´c, V. Nikov, and P. Rombouts, *Low-Latency Encryption – Is "Lightweight = Light + Wait"?* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 426–446.

[5] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds. Springer Berlin Heidelberg, 2007, vol. 4727, pp. 450–466. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74735-2 31

[6] "Specification for the Advanced Encryption Standard (AES)," FederalInformation Processing Standards Publication 197, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[7] M. Sbeiti, M. Silbermann, A. Poschmann, and C. Paar, "Design space exploration of PRESENT implementations for FPGAS," in *5th Southern Conference on Programmable Logic, 2009 – SPL*, April 2009.

[8] P. Yalla and J. P. Kaps, "Lightweight Cryptography for FPGAs," in *International Conference on Reconfigurable Computing and FPGAs, ReConFig '09*, Dec 2009, pp. 225–230.

[9] E. Kavun and T. Yalcin, "RAM-Based Ultra-Lightweight FPGA Implementation of PRESENT," in *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on*, Nov 2011, pp. 280–285.

[10] *ISO/IEC 29192-2:2012,* Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers, ISO/IEC Std., Jan. 2012.

[11] N. Hanley and M. ONeill, "Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers," in *2012 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Aug 2012, pp. 57–62. [12] J. J. Tay, M. L. D. Wong, M. M. Wong, C. Zhang, and I. Hijazin, "Compact FPGA implementation of PRESENT with Boolean S-Box,"
in *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*, Aug 2015, pp. 144–148.

[13] *ISO/IEC 14443-2*, Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 2: Radio frequency power and signal interface, ISO/IEC Std., Aug. 2010.

[14] SunithaTappari, K. Sridevi, Durga Rao Jenjeti, "Compact Lightweight Cryptographic Algorithmfor Optimizationof Resources" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-9 Issue-2S3, December 2019.

[15] Sunitha Tappari , K. Sridevi, "Review on lightweight hardware architectures for the crypt-analytics in FPGA" *International Journal of Engineering & Technology", 7 (3) (2018) 1888-1892.*

[16] Sunitha Tappari, K. Sridevi, "Resource Optimized Security Coding In Light Weight Security Protocol" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-10, August 2019.

[17] Carlos Andres Lara-Nino, Arturo Diaz-Perez, And Miguel Morales-Sandoval, Lightweight Hardware Architectures For The Present Cipher In FPGA, IEEE Transactions On Circuits And Systems–I: Regular Papers, Vol. 64, No. 9, September 2017.

[18] D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großschädl, and A. Biryukov, "Triathlon of lightweight block ciphers for the Internet of Things," in *Proc. NIST Lightweight Cryptogr. Workshop*, 2015.