Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

A High-Performance VLSI Architecture for the PRESENT Lightweight Cryptography

T. Pattalu Naidu
Research scholar
Department of Instrument Technology
Andhra University, Visakhapatnam India

Dr. A. Kamala Kumari
Assistant professor
Department of Instrument Technology,
Andhra University, Visakhapatnam India

***Abstract*—. In this paper, propose a high-performance and area-efficient VLSI architecture with 64-bit datapath for the PRESENT block cipher. The proposed architecture performs an integrated encryption/decryption operation for both 80-bit and 128-bit key lengths. The architecture is synthesized for the Spartan-III XCS400-5 FPGA device, available on the Xilinx platform. The results also highlight that PRESENT is well suited for high-speed and high-throughput applications. Especially its hardware efficiency, It has been observed that the proposed architecture utilizes 0.73% and 0.87% of FPGA slices for 80-bit and 128-bit key lengths, respectively. A throughput of 410 Mbps and power consumption is about 16 mW for both the key lengths.**

Keywords— Lightweight cryptography; PRESENT block cipher; Integrated encryption/decryption; VLSI architecture; FPGAs.

1. INTRODUCTION

THE UPCOMING ERA of pervasive computing will be characterized by many smart devices that—because of the tight cost constraints inherent in mass deployments—have very limited resources in terms of memory, computing power, and battery supply. Here, it's necessary to interpret Moore's law differently: Rather than a doubling of performance, we see a halving of the price for constant computing power every 18 months. Because many foreseen applications have extremely tight cost constraints—for example, RFID in tetrapacks—over time, Moore's law will increasingly enable such applications. Many applications will process sensitive health-monitoring or biometric data, so the demand for cryptographic components that can be efficiently implemented is strong and growing. For such implementations, as well as for ciphers that are particularly suited for this purpose, we use the generic term lightweight cryptography in this article.

Every designer of lightweight cryptography must cope with the trade-offs between security, cost, and performance. It's generally easy to optimize any two of the three design goals—security and cost, security and performance, or cost and performance; however, it is very difficult to optimize all three design goals at once. For example, a secure and high-performance hardware implementation can be achieved by side-channel-resistant architecture, resulting in a high area requirement, and thus high costs. On the other hand, it's possible to design a secure, low-cost hardware implementation with the drawback of limited performance[1]

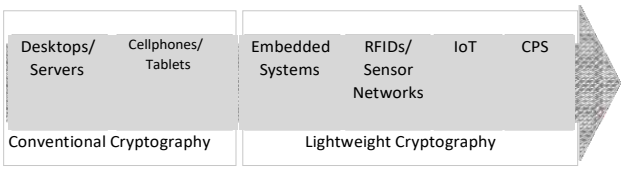

Fig.1. Deployment trend of ciphers in electronic systems.

As shown in the Fig. 1, Lightweight cryptography provides a solution tailored for resource-constrained devices and their efficient VLSI implementations, Recently, national institute of standards and technology (NIST) provided overview of lightweight cryptography and an outline of NIST's plan for standardizing the lightweight cryptographic algorithms [2]. Further, a detailed taxonomy of the lightweight block ciphers can be found in [3] and [4]. Systematic surveys of lightweight- cryptography ciphers and their software and hardware implementations with detailed description and related discussions can be found in [3], [4] and [1]. Here, it has been emphasized that efficient implementation of the ciphers are closely dependent on the selection of appropriate architecture, as they result in low implementation complexity and high- performance in actual realizations. To propose a new architecture for the lightweight cryptography, there is always trade-offs between the three prime objectives i.e. security, cost and performance, which is shown in Fig. 2

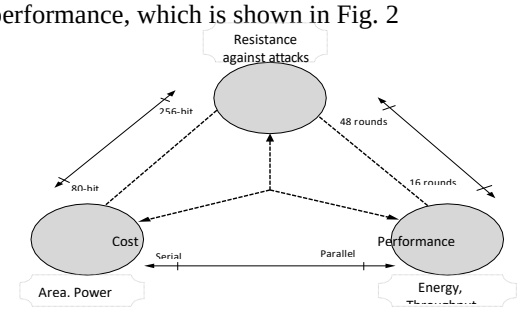

Fig. 2. Architectural trade-offs between security, cost and performance.

In this paper,propose a high-performance and area-efficient VLSI architecture for the PRESENT block cipher that completely integrates both encryption and decryption engines. The architecture has been implemented in the Xilinx Spartan-III XCS400-5 FPGA device [5]. The experimental results of the implementation show that the proposed architecture consumes a number of 126 slices for the 80-bit key and 150 slices for the 128-bit key lengths.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

Lightweight block cipher with a block size of 64 bits

PRESENT algorithm:

The PRESENT algorithm [6] is a symmetric block cipher that can process data blocks of 64 bits, using a key of length 80 or 128 bits. The cipher is referred to as PRESENT-80 or PRESENT-128 when using an 80-bit or 128-bit key respectively

PRESENT specific notations

$K_i = k^i_{63}\ldots k^i_0$ 64-bit round key that is used in round i

k^i_b: bit b of round key K_i

$K = k_{79}\ldots k_0$ 80-bit key register

k_b bit b of key register K

STATE: 64-bit internal state

b_i: bit i of the current STATE

w_i:4-bit word where $0 \le i \le 15$

PRESENT encryption
The PRESENT block cipher consists of 31 'rounds', i.e. 31 applications of a sequence of simple transformations. A pseudocode description of the complete encryption algorithm is provided in Figure 1, where STATE denotes the internal state.The individual transformations used by the algorithm are defined in[6]. Each round of the algorithm uses a distinct round key K_i ($1 \le i \le 31$), Two consecutive rounds of the algorithm are shown for illustrative purposes in Figure 5.

```
generateRoundKeys()
for i = 1 to 31 do
    addRoundKey(STATE,K_i)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE,K_32)
```

PRESENT decryption

The complete PRESENT decryption algorithm is given in Figure 4. The individual transformations used by the algorithm are defined in[6]. Each round of the algorithm uses a distinct round key K_i ($1 \le i \le 31$),

PRESENT transformations:

AddRoundKey

Given round key $\mathbf{K_i} = \mathbf{k^i_{63}\ldots k^i_0}$ for $1 \le i \le 32$ and current STATE $b_{63}\ldots b_0$,

AddRoundKey consists of the operation for $0 \le j \le 63$, $b_j \leftarrow b_j \oplus$.
S-BoxLayer
The non-linear **S-BoxLayer** of the encryption process of PRESENT uses a single 4-bit to 4-bit S-box S which is applied 16 times in parallel in each round. The S-box transforms the input x to an output $S(x)$ as given in hexadecimal notation in Table1
For **S-BoxLayer** the current STATE $b_{63}\ldots b_0$ is considered as sixteen 4-bit words $w_{15}\ldots w_0$ where $w_i = b_{4*i+3} \| b_{4*i+2}$

$\| b_{4*i+1} \| b_{4*i}$ for $0 \le i \le 15$ and the output nibble $S(w_i)$ provides the updated state values as a concatenation $S(w_{15}) \| S(w_{14}) \| \ldots \| S(w_0)$.

Inverse S-Boxlayer

The s-box used in the decryption procedure of present is the inverse of the 4-bit to 4-bit s-box s that is described and the inverse s-box transforms the input x to an output $s^{-1}(x)$ as given in hexadecimal notation in table 2.

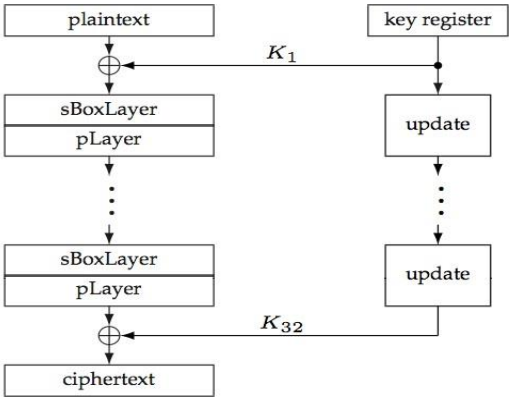

Figure 3— The encryption procedure of PRESENT

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

```
generateRoundKeys()
addRoundKey(STATE,K32)
for i = 31 downto 1 do
   invpLayer(STATE)
   invsBoxLayer(STATE)

   addRoundKey(STATE,Ki)
end for
```

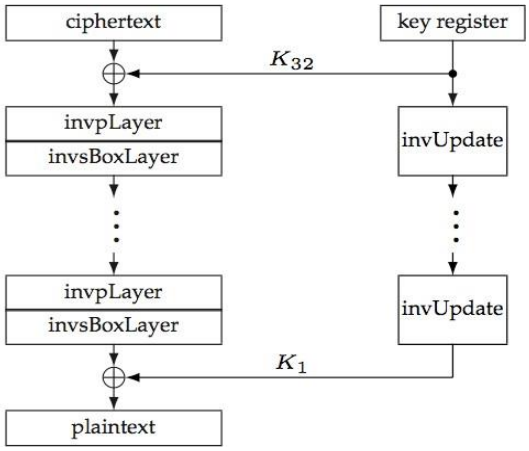

Figure 4 — The decryption procedure of PRESENT

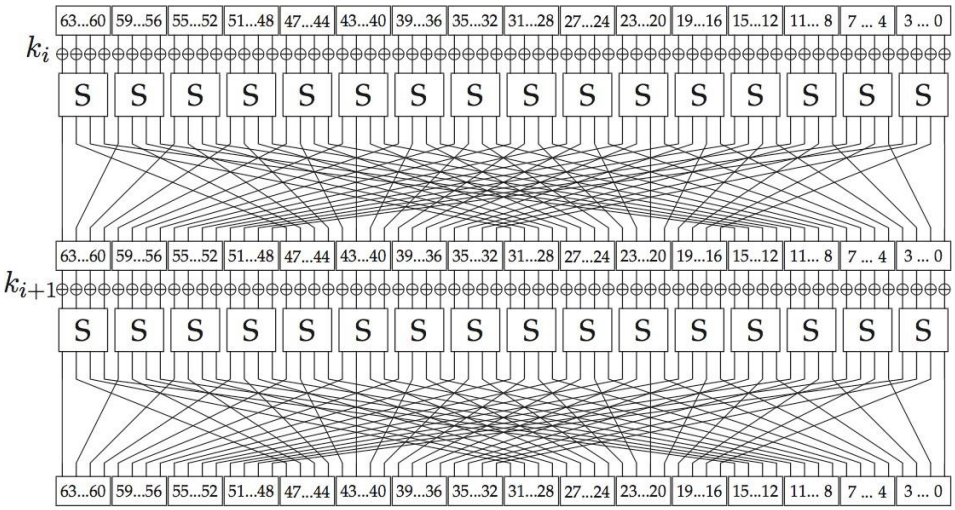

Figure 5 — Two rounds of PRESENT

P-Layer

The bit permutation **pLayer** used in the encryption routine of PRESENT is given by Table 3. Bit *i* of STATE is moved to bit position P(i).

Inv P-Layer

The inverse permutation layer **invpLayer** used in the decryption routine of PRESENT is given by Table 4.Bit i of STATE is moved to bit position $P^{-1}(i)$.

PRESENT key schedule

The key schedule. present can take keys of either 80 or 128 bits. However. we focus on the version with 80-bit keys. The user-supplied key is stored in a key register K and represented as k79k78 . . . k0. At round i the 64-bit round key Ki = κ63κ62 . . . κ0 consists of the 64 leftmost bits of the current contents of register K. Thus at round i we have that: 1. [k79k78 . . . k1k0] = [k18k17 . . . k20k19]
2. [k79k78k77k76] = S[k79k78k77k76]
3.[k19k18k17k16k15]=[k19k18k17k16k15]⊕ round_counter

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

TABLE-1:PRESET S-box

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-2: PRESENT inverse S-box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S^{-1}(x)$	5	E	F	8	C	1	2	D	B	4	6	3	0	7	9	A

TABLE-3:PRESENT Permutation Layer Box

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

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

TABLE-4:PRESENT Permuatation inverse layer Box

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

FPGA IMPLEMENTATION OF FPGA

The main design goals of the PRESENT block cipher described in PRESENT Algorithm, it is simplicity and high perfor mance/area ratio, so that all cipher components can be easily mapped in hardware. First, it describe our implementation of the encryption algorithm of PRESENT. The top level de- sign overview is shown in Fig. 6 and the interface of the ci- pher top module is shown in Fig. 7. As can be seen from the latter one our PRESENT-80 and PRESENT-128 entities have 212 and 270 I/O pins, respectively. We did not implement any I/O logic such as a UART interface in order to achieve implementation figures for the plain PRESENT core. The in- terface usually strongly depends on the target application.

It deliberately use additional I/O pins for a parallel key input. There are two reasons why we abandon the options of hard-coding the key inside the cipher module or implement- ing serial interface to supply the key to the algorithm. First, we want to reduce the control logic overhead to a minimum to be able to present the results reflecting the performance of the ciphering algorithm only. Secondly, most applications will us as an independent cipher module inside a larger top entity, so that the key can be supplied externally and in that perspective our implementation model offers the best flexibility.

Unfortunately, the low-cost Spartan-III XC3S200 FPGA has no package with more than 173 I/O pins [7]. Therefore we decided to move to the more advanced Spartan-III XC3S400 which features a package (FG456) with 264 I/O pins. Larger Spartan FPGAs such as the Spartan-III XC3S1000 feature even more I/O pins but also contain more logic resources. Since we focus on lightweight and low- cost implementations of PRESENT in this paper we chose the smallest possible device Spartan-III XC3S400 which is only slightly larger (and hence more expensive) than the Spartan- III XC3S200.

The entire cipher control logic was implemented as a 3-state finite-state machine. After reset the first round begins and the two inputs of the algorithm, plaintext and user-supplied key are read from the corresponding registers. The 64 and 80-bit multiplexers select the appropriate input depending on the value of the round counter, i.e. initial values for plaintext and key are valid only in round 1. Both 64- and 80-bit D-flip-flops are used for round synchroniza tion between the round function output and the output of the key schedule. Part of the round key is then XORed with the plaintext. Key schedule and round function run in parallel for each round

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

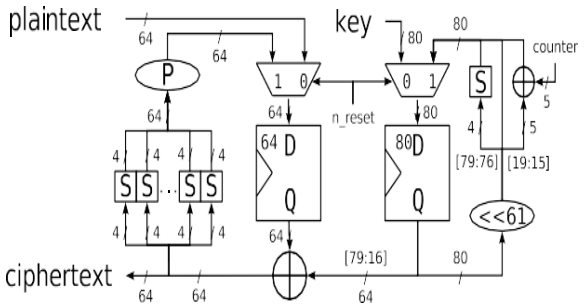

Fig. 6. The data path of an area-optimized version of thePRESENT-80 encryption unit

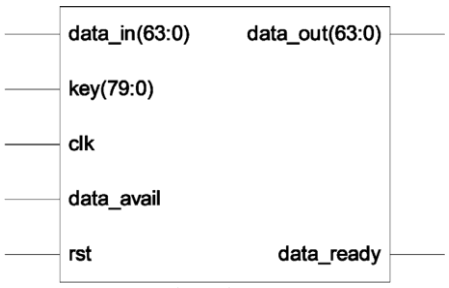

Fig. 7. Interface of the PRESENT-80 top module.

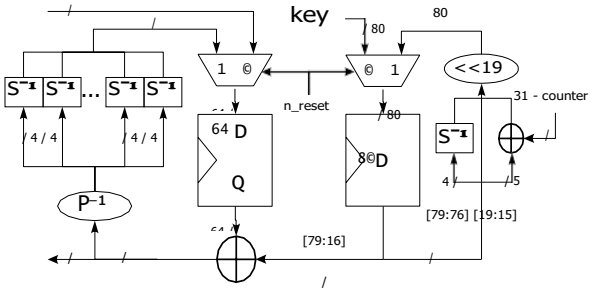

Fig. 8. The data path of an area-optimized version of the PRESENT-80 decryption unit

Implementation of both permutation and bit-rotation is very straightforward in hardware, which is a simple bit-wiring. The highly non-linear PRESENT S-Box function is the core of the cryptographic strength of the cipher, and is the only design component that takes a lion's share of both computational power and area. Two implementation options for the PRESENT S-Box were taken in consideration in order to optimize the efficiency of the cipher. Using Look-Up Ta- bles (LUTs) for bit substitution is the most obvious one and was implemented first. An alternative considered next was determining a minimal non-linear Boolean function

$$Si : F42\_ \rightarrow F2$$
$$(x3x2x1x0) \; \rightarrow y_i, \qquad 0 \leq i \leq 3$$

for each bit output of the PRESENT S-Box using only standard gates, i.e. AND, OR and NOT. A tool named espresso [8] helped us produce such minimal Boolean functions for the PRESENT S-Box.

Interestingly, in some cases this modification yielded performance boost in terms of max. frequency/throughput and area requirements measured in occupied slices. E.g., for PRESENT-80 with espresso-optimized S-Box ISE showed significant decrease in critical path delay due to routing as compared to the S-Box implementation with LUTs. From our results we conclude that espresso and its minimal Boolean functions can yield better resources uti- lization and may in some cases outpace ISE's internal syn- thesis mechanisms

The decryption unit of PRESENT is very similar to the encryption. The decryption data path is presented in Fig. 5. The first round of decryption requires the last round key of the encryption routine. For optimal performance we assume that this last round key is precomputed and available at the beginning of the decryption routine. The assumption is fair since we have to perform this step only once for multiple cipher texts.

We implemented both encryption and decryption functions in VHDL for the Spartan-III XC3S400 (Package FG456 with speed grade -5) FPGA core from Xilinx. We used Mentor Graphics ModelSimXE 6.2g for simulation purposes and Xilinx ISE v10.1.03 WebPACK for design synthesis.

Table 5 summarizes the performance figures for our im- plementations. All figures presented are from Post Place & Route Timing Report. To achieve optimal results both Syn- thesis and Place & Route Effort properties were set to High and Place & Route Extra Effort was set to continue on impossible.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

TABLE 5. Performance results for encryption and decryption of one data block with PRESENT for different key sizes and S-Box implementation techniques

Key size	enc/dec	S-box w/	#LUTs	#FFs	Total equiv. Slices	Max. freq. (MHz)	#CLK cycles	Throughput (Mbps)	Efficiency (Mbps/#Slices)
80	enc	espresso	253	152	176	258	32	516	2.93
		LUT	350	154	202	240	32	480	2.38
	dec	espresso	328	154	197	240	32	480	2.44
		LUT	328	154	197	238	32	476	2.42
128	enc	espresso	299	200	202	250	32	500	2.48
		LUT	300	200	202	254	32	508	2.51
	dec	espresso	366	202	221	239	32	478	2.16
		LUT	366	202	221	239	32	478	2.16

To compare the proposed design with an existing design available in the literature, the selected design metrics are: slice LUTs, registers and a total number of consumed slices. To perform a comparison at the architectural-level, the proposed integrated architecture is tuned to match the architectural capability of [9]. Therefore, for comparison, the key scheduling unit is implemented using *on-the-fly* mode rather than storing the computed keys in the BRAM. An architectural-level comparison between the proposed design and the design of [9] is given below.

*A. Architectural-level Comparison*The architecture presented in [9] is one of a few established ones that provides decryption operation for the FPGA. This architecture has been implementation on the Xilinx Spartan-IIIXC3S400 FPGA device. Thus, to perform a fair comparison of utilized device resources, we have targeted the same FPGA device and equal speed grade. Similar to [9],. The implementation has been performed for both 80-bit key length (*PRE_80*) and 128-bit key length (*PRE_128*). The synthesis results for both the architectures are compared and shown in Fig.9

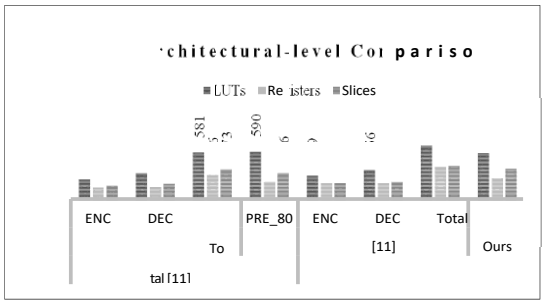

Fig.9Architectural-level comparison between the architecture of [10]

All the data presented in Fig. 9, are from the post place and route (*PnR*) report. It can be observed from the above figure that, in comparison to architecture [10], the proposed architecture with 80-bit key length (*PRE_80*) requires 12.6% lower FPGA slices and with 128-bit key length (*PRE_128*) consumes 9.7% lesser slices. By this, we can say that the proposed integrated architecture is capable of performing both the encryption (ENC) and decryption (DEC) by the same set of hardware, which is an essential requirement in any practical lightweight cipher-based system. Also, the integrated architecture consumes lesser slices in comparison to two separate modules for performing encryption and decryption. It can be noted that our design requires an extra clock cycle in comparison with [10] to perform the operations as we have considered the registered output.

TABLE6:performance on the Xilinx Spartan-III XC3S400 FPGA

Elements	Resource Utilization *PRE_80*	Resource Utilization *PRE_128*
Latency	33	33
Max. frequency (MHz)	215.42	212.13
Throughput (Mbps)	417.79	411.41
Efficiency (Mbps/#Slices)	3.32	2.74
Power (mW)	16.59	16.80

CONCLUSION

An integrated VLSI architecture for PRESENT lightweight block cipher is presented. The architecture supports both the encryption and decryption operations with 80-bit and 128-bit key lengths. The design is modeled in the VHDL language and synthesized in Xilinx Spartan-IIIXC3S400 FPGA device on ML-505 platform. The architecture utilizes 0.73% and 0.87% of FPGA slices for 80-bit and 128-bit key length, respectively. The throughput of the design is around 410 Mbps and power consumption isaround 16 mW for both the key lengths. The proposed architecture is area-efficient with high-performance capability for providing an adequate level of security under the resource- constrained environment for IoT and CPS applications.

ACKNOWLEDGMENTS

We thank Dr.A.kamalakumari,AssistantProfessor, k.chiranjeevi rao for their contributions to the development of Present. We also thank E.Govind for his assistance with software implementations.

REFERENCES

[1] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann and L. Uhsadel, "A survey of lightweight cryptography implementations," IEEE Design

[2] K. McKay, L. E. Bassham, M. S. Turan and N. W. Mouha, "NISTIR8114 - Report on Lightweight Cryptography," National Institute ofStandards and Technology (NIST), Gaithersburg, March 2017.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 01, January-2020

[3] A. Biryukov and L. Perrin, "Lightweight Block Ciphers,"[Online]:https://www.cryptolux.org/index.php/Lightweight_Block_Ciphers.
[4] B. J. Mohd, T. Hayajneh and A. V. Vasilakos, "A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues," Jour. of Network and Computer Appl., vol.
[5] B. J. Mohd, T. Hayajneh and A. V. Vasilakos, "A survey on lightweight block ciphers for low-resource devices: Comparativestudy and open issues," Jour. of Network and Compute Appl., vol.
[6] "Information technology – Security techniques – Part 2: Block ciphers," Jan. 2012
[7] X. Inc., "Spartan-3 FPGA Family Data Sheet," avail- able online via http://www.xilinx.com, June 2008
[8] N.A.,"Espresso,"availableonlineviahttp://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm, November 1994
[9] T. Good and M. Benaissa, "AES on FPGA from theFastest to the Smallest," in *Proceedings of CHES 2005*, pp. 427–440.
[10] M. Sbeiti, S. Michael, A. Poschmann and C. Paar, "Design space exploration of present implementations for FPGAS," in 5th Sout.Conf. on Prog. Logic, Sao Carlos, Brazil, pp. 141-145, 1-3 April 2009.
[11] P. Yalla and J. P. Kaps, "Lightweight cryptography for FPGAs," in IEEE Int'l Conf. on Reconfigurable Computing and FPGAs (ReConFig'09), Cancun, Mexico, pp. 225-230, 09 Dec. 2009.
[12] E. B. Kavun and T. Yalcin, "RAM-based ultra-lightweight FPGA implementation of PRESENT," in Int'l Conf. on Reconf. Computing and FPGAs (ReConFig'11), Cancum, Mexico, pp. 280-285, 30 Nov-02 Dec 2011.