An Efficient VLSI Architecture for PRESENT Block Cipher and its FPGA Implementation

Jai Gopal Pandey, Tarun Goel*, Abhijit Karmakar

CSIR - Central Electronics Engineering Research Institute, Pilani, India
* Academy of Scientific & Innovative Research (AcSIR), CSIR-CEERI Campus

jai@ceeri.res.in, tarungoel.com@gmail.com,
abhijit@ceeri.res.in

Abstract. Lightweight cryptography plays an essential role for emerging authentication-based pervasive computing applications in resource-constrained environments. In this paper, we have proposed resource-efficient and high performance VLSI architectures for PRESENT block cipher algorithm for the two key lengths 80-bit and 128-bit, namely PRESET-80 and PRESENT-128. The FPGA implementations of these architectures have been done on LUT-6 technology based Xilinx Virtex-5 XC5VFX70T-1-FF1136 FPGA device. These architectures have latency of 33 clock cycles, run at maximum clock frequency of 306.84 MHz and provide throughput of 595.08 Mbps. They have been compared with the two different established architectures. It has been observed that the PRESENT-80 architecture consumes 20.3% lesser FPGA slices and there is gain of 25.4% in throughput. Similarly, the PRESENT-128 architecture requires 20.7% lesser FPGA slices alongwith a reduction in the latency by 27.7% and an overall increase of throughput by 69.1%.

Keywords: Lightweight cryptography, PRESENT, block cipher, VLSI architectures, FPGAs.

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 ciphertext.

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 alongwith 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.

1.1 Related Work

The architectural exploration for the PRESENT block cipher with serial, iterative and parallel variants has been given in [6]. Further, an investigation of the architectural design space exploration using Spartan-III FPGA can be found in [7]. An FPGA implementation of PRESENT cipher has been reported in [8] that uses 117 slices of the Xilinx Spartan-3 XC3S50 FPGA device. Here, a throughput of 28.46 Mbps has been obtained at the maximum frequency of 114 MHz. Two different RAM-based implementations of PRESENT cipher have been provided in [9]. In the first implementation, the substitution boxes have been realized within the FPGA slices, and in the second implementation they have been integrated into the same RAM block used for state storage. Here, the first design occupies 83 slices and the second design consumes 85 slices of Xilinx Spartan XC3S50 device. These realizations produce throughput of 6.03 Kbps and 5.13 Kbps at 100 KHz system clock respectively. In another implementation, 8-bit datapath based implementation of the PRESENT cipher has been provided in [10]. The design consumes 62 slices of the Xilinx Virtex-5 XC5VLX50 device and provides latency of 295 clock cycles with a throughput of 51.32 Mbps at the maximum frequency of

236.574 MHz. One of the implementations using 64-bit datapath, which utilizes 74 slices of the Xilinx Spartan-6 XC6SLX16-3CSG324C FPGA device has been provided in [11]. Here, at maximum clock frequency of 221.63 MHz and with 33 clock latency, a throughput of 221.63 Mbps has been obtained. Similar to this a 64-bit datapath based architecture has been synthesized on 87 slices of the Xilinx Virtex-5 XC5VLX50 FPGA device in [12]. Here latency of 47 clocks, maximum clock frequency 221.64 MHz and a throughput of 341.64 Mbps have been reported. In the following section we provide contributions of this paper.

1.2 Our Contributions

In this paper we propose an efficient VLSI architecture for PRESENT block cipher. Based on the key length of 80-bit and 128-bit, we provide two different variants of the architecture. These architectures are represented as *Proposed_80* and *Proposed_128* respectively. In both the architectures, the required S-box(*S*) is realized by an area-optimized combinational logic datapath. These architectures are synthesized on Xilinx Virtex-5 XC5VFX70T-1-FF1136 FPGA device. The first architecture (with 80-bit key) utilizes 0.51% FPGA slices and the second architecture (with 128-bit key) uses 0.62% of FPGA slices. Here, both the architectures have latency of 33 clock cycles, runs at maximum clock frequency of 306.84 MHz and provides throughput of 595.08 Mbps.

Further, the proposed architectures are compared with the established architectures of [11] and [12] with 6-input look-up table (LUT-6) technology based FPGAs [13]. Here, the standard platform and the devices with the same device family alongwith the same speed grades are considered for comparisons. For this purpose, the architectures are synthesized on two different FPGA devices of LUT-6 based technology. These devices are Xilinx Spartan-6 XC6SLX16-3CSG324C [11] and Virtex-5 XC5VFX50 [12] FPGAs. By experimental results it is found that in comparison to the architecture of [11], the *Proposed_80* consumes 20.3% lesser FPGA slices and there is a gain of 25.4% in throughput. Similarly, in comparison to 128-bit key architecture [12], it requires 20.7% lesser FPGA slices and reduction in the latency by 27.7% thus, an overall increase of throughput by 69.1% is observed.

Rest of this paper is organized as follows: in Section 2, an overview of the PRESENT algorithm is given. Section 3 is used to describe the proposed architecture for the PRESENT cipher alongwith the detail description of various basic building blocks. Section 4 is used to provide experimental results and comparisons with the established architectures. Finally, conclusions are drawn in Section 5.

2 The PRESENT Algorithm

The PRESENT algorithm operates on block size of 64-bit. It supports two key lengths of 80-bit and 128-bit. The algorithm is based on the SP-network and consists of 31 rounds [5]. Each of the 31 rounds consist of an XOR operation, which is required to introduce a round key K_i for $0 \le i \le 31$ in which K_{31} is used for post-whitening operation. In addition to that, there is a linear bitwise permutation layer and a non-linear

substitution layer based operation. The non-linear layer uses a single 4-bit S-box which is applied 16-times in parallel in each rounds. The algorithm requires mainly four functions, which are: Key Scheduling, AddRoundKey, *sboxlayer* and *p-layer* [5]**.**

3 An Architecture for PRESENT Block Cipher

The proposed architecture for PRESENT block cipher is shown in Fig. 1. To save area and compute-time we have considered iterative type of architecture. Here, 64-bit datapath is chosen that provides optimal trade-off in terms of area, power and latency. The three main components of the architecture are: encryption engine, key scheduling and a controller. The key scheduling block takes 80-bit or 128-bit input key and generates thirty one round keys for the thirty one individual rounds of the cipher.

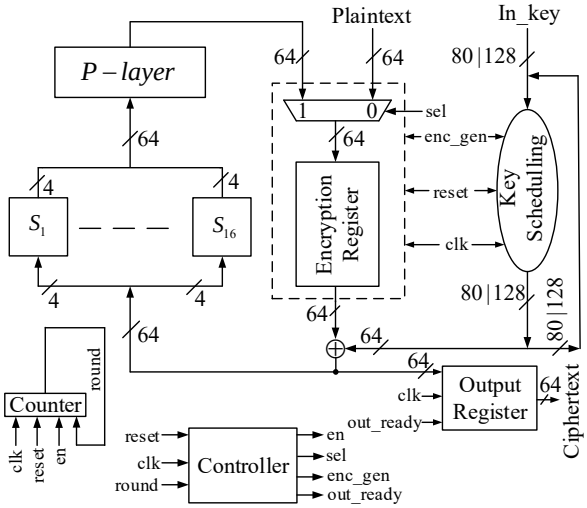

Fig. 1. Proposed architecture for the PRESENT cipher.

The datapath of the architecture consists of a set of flip-flops, registers, multiplexers and XOR gates. The permutation/expansion is a simple bit-transposition, which requires only simple wirings. The main building blocks of the architecture have been arranged in different subsections as described below.

3.1 Datapath of PRESENT Architecture

The 64-bit datapath based architecture as shown in the Fig. 1 is capable of performing the encryption operation with a key register. The architecture consists of a 64-bit *Encryption Register*, which is used to store the internal states of encryption operation. A 64-bit state register is used to store the internal state alongwith an 80-bit register (*Proposed_80*) or 128 bit key register (*Proposed_128*) for storing the intermediate round key. In addition to that one 64-bit and another 80-bit (or 128 bit) multiplexers are used

to switch the data between load phase and round computation phase. The datapath contains *sboxlayer* (16 S-boxes) and one S-box (*Proposed_80*) or two S-boxes (*Proposed_128*) for key scheduling. Alongwith this, one 64-bit XOR gate, 5-bit XOR gate and a 5-bit up-counter are also used.

In the proposed architecture the inputs and outputs are registered. For this, 64-bit register is used to get the ciphertext at the output. By this the output gets synchronized with the last round. The latency can further be reduced by one more clock cycle if we do not want the output to be registered. However, the register is added to reduce the control logic and for synchronization of output with the last round. After completion of all the rounds, registered output is available after thirty three clock cycles. The main advantage of this architecture is that there is a reduction in the latency alongwith efficient utilization of hardware resources. In the Fig. 1, round keys are computed on-the-fly which are used for mixing with the state.

Plaintext is loaded in the first clock cycle. In the next clock cycle multiplexer switches the data and then for next 31 cycles all intermediate states are computed. Data is available at the *Encryption Register* and is mixed with intermediate round key i.e., XORed with first 64-bit of round key. Further, the mixed state is passed to *sboxlayer* for state processing, which provides 64-bit data concurrently to *p-layer* and subsequently, it is passed to the *Encryption Register* through the multiplexer. In the last clock cycle, ciphertext is available at output register. Thus, here, a total of 1+31+1=33 clock cycles are required to encrypt a single block of 64-bit plaintext.

3.2 Design of the sboxlayer

To achieve high performance, high throughput and area efficient design of *sboxlayer*, numerous approaches can be considered. Three main deign approaches for *sboxlayer* are: look up table (LUT) based approach [7] and [12], RAM-based approach [9], and combinational logic based approach [7], [10]. The LUT based design offers a shorter critical path, however, there is some associated delay in high speed pipelined designs. Using RAM-based design a single S-box requires 16×4 bit size of memory. For a reasonably fast implementation of the encryption operation, on an average sixteen S-boxes are required. Thus, the memory requirement raises to $(16 \times 4) \times 16$ bits, that is, equal to 1-Kb, which is a sufficiently large amount of memory. To minimize the delay and area requirements another way is to design the *sboxlayer* circuits using combinational logic optimization directly from its logic properties. In the proposed architecture the S-box(S) is realized by an area-optimized combinational logic circuit.

3.3 Architecture for the Key Scheduling Process

Key processing unit works on-the-fly with each round. A 64-bit register is used to store the round key, the first leftmost 64-bit of the key register is XORed with the intermediate state. At the first clock, input key is loaded to the key register as shown in Fig. 2. As per the Fig. 2(a), following three steps are performed for key scheduling with 80-bit key.

(a.) The output of the key register is rotated by 61 bits to the left, i.e.,
$k_{79}k_{78}.....k_1k_0 \Rightarrow k_{18}k_{17}...k_1k_0k_{79}k_{78}...k_{20}k_{19}$

(b.) First 4-bit is passed through S-box as,
$[k_{79}k_{78}k_{77}k_{76}] \Rightarrow S[k_{79}k_{78}k_{77}k_{76}]$

(c.) The value of counter is XORed with 5-bit of key,
$k_{19}k_{18}k_{17}k_{16}k_{15} \Rightarrow k_{19}k_{18}k_{17}k_{16}k_{15} \oplus round\_counter$

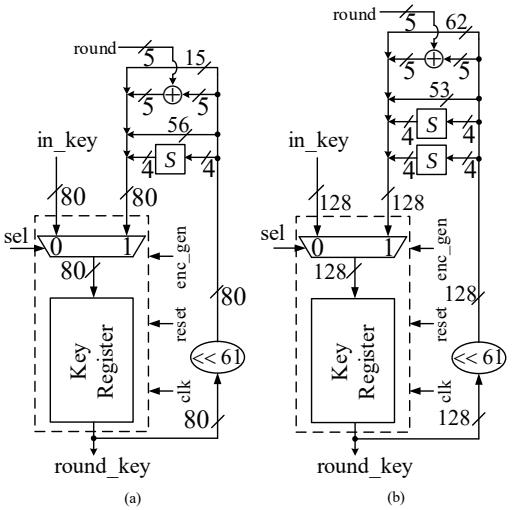

Fig. 2. The key-scheduling process in the PRESENT cipher (a) with 80-bit input key (b) with 128-bit input key.

Similarly, with 128-bit key, the following three steps is performed for key scheduling as shown in Fig. 2(b).

(a.) $k_{127}k_{126}.....k_1k_0 \Rightarrow k_{66}k_{65}...k_1k_0k_{127}k_{126}...k_{68}k_{67}$

(b.) $[k_{127}k_{126}k_{125}k_{124}] \Rightarrow S[k_{127}k_{126}k_{125}k_{124}]$ and $[k_{123}k_{122}k_{121}k_{120}] \Rightarrow S[k_{123}k_{122}k_{121}k_{120}]$

(c.) $k_{66}k_{65}k_{64}k_{63}k_{62} \Rightarrow k_{66}k_{65}k_{64}k_{63}k_{62} \oplus round\_counter$

3.4 Controller for Encryption Operation

A controller, as shown in Fig. 3 is designed to generate various required control signals for controlling the key generation process and the encryption engine. The controller generates four control signals which are *en*, *enc_gen*, *sel* and *out_ready*. There are three states in the FSM, in state S_0 the counter is enabled through *en* signal and the plaintext with key is loaded when *sel*='0'. In state S_1 the multiplexers are switched as *sel* gets logic '1'. The *enc_gen* signal is used to start the intermediate operations by enabling the encryption and key registers. The state remains in S_1 until counter value reaches 31. Then, the state is switched to state S_2 where counter is disabled through *en*='0' and

out_ready signal becomes logic '1'. In the next cycle ciphertext is available through output register.

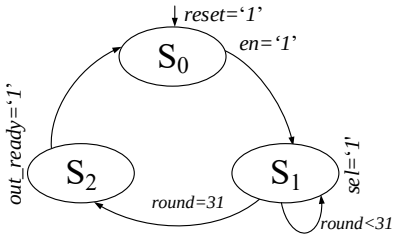

Fig. 3. FSM for the PRESENT cipher.

4 Experimental Results and Discussions

The proposed architectures are implemented in the VHDL language, and synthesized using Xilinx Design Suite 14.7 for the Virtex-5 XC5VFX70T-1-FF1136 FPGA device on Xilinx ML-507 platform. The device utilization is given in Table 1.

Table 1. Device utilization summary for Xilinx Virtex-5 XC5VFX70T-1-FF1136 FPGA.

Elements	**Available Resources**	**Resource Utilization** ***Proposed_80***	**Resource Utilization** ***Proposed_128***
Slice LUTs	44800	218	266
Slice Registers	44800	215	263
Total Slices	11200	57	69
Bonded IOBs	640	210	258
Latency	-	33	33
Max.freq.(MHz)	-	306.84	306.84
Throughput (Mbps)	-	595.08	595.08

Here, the architecture *Proposed_80* utilizes 0.51% FPGA slices and the architecture *Proposed_128* utilizes 0.62% of FPGA slices. The IOBs utilization in the *Proposed_80* architecture is 32.81% and in *Proposed_128* architecture it is 40.31%. Here, both the architectures have the latency of 33 clock cycles, runs at maximum clock frequency of 306.84 MHz, consumes 23.75 mW power and provide throughput of 595.08 Mbps.

To compare the architectures with the existing designs, the selected design metrics are: slice LUTs, registers and total number of consumed slices. Performance of the design is evaluated in terms of latency, maximum frequency and throughput. Throughput is computed using formula, $throughput = (\max.\, freq. \times total\ no.\ of\ bits)/latency$. To compare the work with the implementation of [11] and [12] which uses LUT-6 technology based FPGAs, the design have been synthesized for two different Xilinx devices namely, Spartan-6 XC6SLX16-3CSG324C [11] and Virtex-5 XC5VFX50 [12].

Architectural comparisons between *Proposed_80* and [11] is given in Table 2. It can be observed that in comparison to the implementation reported in [11], the proposed architecture requires 20.3% lower FPGA slices and the LUT-FF pair utilization is 96%. Alongwith the efficient utilization of the device resources, the performance of the architecture has also improved. In comparison to [11], the proposed architecture is able to work on an increased maximum frequency by 25.4%, and thus, there is gain in the throughput by 25.4% as per.

Table 2. Comparison of resource utilization between proposed architecture and architecture [11] on Xilinx Spartan-6 XC6SLX16-3CSG324C FPGA device.

Elements	**Available Resources**	**Architecture (PRE) [11]**	***Proposed_80***
Slice LUTs	9112	229	221
Slice Registers	18224	136	224
Total Slices	2278	74	59
Latency	-	33	33
Max. freq.(MHz)	-	221.63	278.00
Throughput (Mbps)	-	429.83	539.15

In the second implementation, in line with [12], we have used 128-bit key size. The synthesis results of the implementations are shown in Table 3. As evident from the table, the proposed architecture requires 20.7% lesser FPGA slices in comparison to the architecture of [12]. Alongwith LUT-FF pair utilization by 96%, we also get improvements in the performance. In the proposed architecture the latency has also reduced by 27.7% and there is an increase of 22.3% in the maximum frequency. The decrease in latency and increase in the maximum frequency result in an overall increase of 69.1% in the throughput which is a significant improvement.

Table 3. Comparison of resource utilization between proposed architecture and architecture [12] on Xilinx Virtex-5 XC5VFX50 FPGA device.

Elements	**Available Resources**	**Architecture (Iterative)** [12]	***Proposed_128***
Slice LUTs	28800	285	271
Slice Registers	28800	200	263
Total Slices	7200	87	69
Latency	-	47	34
Max. freq. (MHz)	-	250.89	306.84
Throughput (Mbps)	-	341.64	577.58

The Virtex-5 XC5VLX50 device as considered in [12] offers 220 I/O pins. For the purpose of comparison with [12], the128-bit key is brought into the datapath using two clock cycles, thus, incurring an increased latency of 1 clock. As per the iterative architecture of [12], 8-bit input has been supplied to 64-bit datapath at a time, thus requiring a total of 16 clock cycles to bring the 128-bit key. Apart from this, it requires 8 more

clock cycles for providing 64-bit ciphertext output. In comparison, the proposed architecture completes the processing from plaintext to ciphertext in 34 clock cycles.

The design have also been compared with some other popular implementations using LUT-6 technology across different FPGA devices. The 64-bit implementations of PRESENT that are considered for comparison are: iterative architecture [10] denoted as *Iterative:Tay*; basic implementation (PRE) [11], denoted as *PRE:Nino*; the area-optimized design of [11] as *PRE_O1:Nino*; the serial implementation of [12] as *Serial:Hanely*; and the iterative realization of [12] as *Iterative:Hanely*.

A comparison of latency vs. number of slices consumed is shown in Fig. 4. In comparison to the latest implementation of *PRE:Nino* [11], a reduction in slice count is observed. Also, there is reduction in both the latency and number of used slices with respect to *Iterative:Hanely* [12]. The implementation of *Iterative:Tay* uses slightly less slices in comparison to the *Proposed_128* however, there is an increase in the latency which can be observed in the Fig. 4.

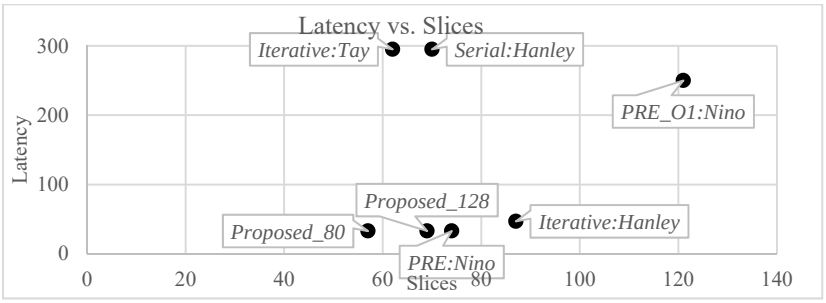

Fig. 4. Comparison of latency vs. consumed slices for different PRESENT implementations using LUT-6 based FPGA devices.

Also, a comparison of throughput (Mbps) vs. number of occupied slices is shown in Fig. 5. It is found that in comparison to both *PRE:Nino* [11] and *Iterative:Hanely* [12], the proposed architecture requires lowest number of slices and provides highest throughput (Mbps).

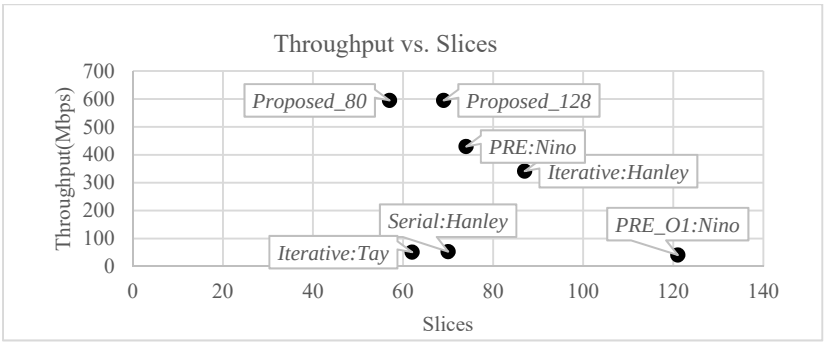

Fig. 5. Comparison of throughput vs. consumed slices for different PRESENT implementations using LUT-6 based FPGA device.

5 Conclusion

In this paper, we have presented two efficient VLSI architectures for PRESENT block cipher with key sizes of 80-bit and 128-bit. The proposed architectures efficiently utilize the FPGA slices for providing data security under the resource-constrained environment. The design has been modeled in VHDL language and the architectures have been synthesized in the Xilinx Virtex-5 XC5VFX70T-1-FF1136 FPGA device. The presented architectures consume only 57 and 69 FPGA slices respectively. In comparison to other existing implementations, the proposed architectures shows improvement in terms of hardware resources and provide high throughput, which makes them amenable for utilizing in lightweight cryptography applications.

References

1. Lee, E.A., Seshia, S.A.: Introduction to Embedded Systems – A Cyber-Physical Systems Approach. 1st edn. LeeSeshia.org (2011).
2. Xu, T., Wendt, J.B., Potkonjak, M.: Security of IoT systems: Design challenges and opportunities. In: IEEE/ACM Int'l Conf. on Computer-Aided Design, pp. 417–423. IEEE, San Jose, California (03 Nov. 2014).
3. Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., Uhsadel, L. A survey of lightweight-cryptography implementations. IEEE Design & Test of Computers, 24(6), 522–533 (2007).
4. ISO/IEC 29192-2,*"Information tech.-Security techniques-Part 2:Block ciphers,"* 2012.
5. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer (2007).
6. Rolfes, C., Poschmann, A., Leander, G., Paar, C.: Ultra-lightweight implementations for smart devices–security for 1000 gate equivalents. In: Int'l Conf. on Smart Card Research and Advanced Applications, pp. 89–103. Springer, London, UK (08–11 Sep. 2008).
7. Sbeiti, M., Michael, S., Poschmann, A., Paar, C.: Design space exploration of present implementations for FPGAS. In: 5th Southern Conf. on Programmable Logic (SPL), pp. 141–145. IEEE, Sao Carlos, Brazil (1–3 April 2009).
8. Yalla, P., Kaps, J.P.: Lightweight cryptography for FPGAs. In: Int'l Conf. on Reconf. Computing and FPGAs (ReConFig'09), pp. 225–230. IEEE, Cancun, Mexico (09 Dec. 2009).
9. Kavun, E.B., Yalcin, T.: RAM-based ultra-lightweight FPGA implementation of PRESENT. In: Int'l Conf. on Reconfigurable Computing and FPGAs (ReConFig'11), pp. 280–285. IEEE, Cancum, Mexico (30 Nov–2 Dec 2011).
10. Tay, J.J., Wong, M.L.D., Wong, M.M., Zhang, C., Hijazin, I.: Compact FPGA implementation of PRESENT with Boolean S-Box. In: 6th Asia Symp. on Quality Electronic Design (ASQED), pp. 144–148. IEEE, Kula Lumpur, Malaysia (04 Aug. 2015).
11. Lara-Nino, C.A., Morales-Sandoval, M., Diaz-Perez, A.: Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher. In: 2016 Euromicro Conf. Digital System Design (DSD), pp. 646–650. IEEE, Limassol, Cyprus (31 Aug. 2016).
12. Hanley, N., O'Neill, M.: Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers. In: IEEE Computer Society Annual Symp. on VLSI (ISVLSI), pp. 57–62. IEEE, Amherst, MA, USA (19–21 Aug. 2012).
13. Cosoroaba, A., Rivoallon, F. (2006). Achieving higher system performance with the Virtex-5 family of FPGAs, Xilinx WP245 (Vol. 1).