Lightweight Hardware Architectures for PRESENT Cipher in FPGA

Suresh.H[1], R Vignesh Chandrasekhar[2]
ME VLSI Design[1], Assistant Professor[2]
Department of Electronics and Communication, REC Coimbatore

Abstract **- Lightweight symmetric ciphers have gained interest in constrained computing due to the increasing demand for security services as in the Internet of Things. This paper discusses the hardware implementations of PRESENT, a standardized lightweight cipher designed to overcome part of the security issues in extremely constrained conditions. The most representative realizations of this cipher are reviewed and two novel designs are presented. Using the same implementation conditions, the two new proposals and three state-of-the-art designs are evaluated and compared using area, performance, energy, and efficiency as metrics. In particular, this design results to be adequate in regards to energy-per-bit and throughput-per-slice.**

Keywords **- cipher, IoT , PRESENT**

I.INTRODUCTION

The Internet of Things (IoT) is said to revolutionize the way in which individuals and organizations interact with the physical world and among themselves. According to IoT is regarded as an extension of Internet to the real world of physical objects, usually associated with such terms as “ambient intelligent”, “ubiquitous network”, and “cyber- physical system”. It has been cataloged as one of the six disruptive technologies with potential impacts on US interests out to 2025, which denotes its relevance. Everyday smart objects could become information-security risks, and the IoT could distribute those risks more widely than the conventional Internet. These security risks are the central issues that may delay the development and adoption of IoT applications. This has motivated the study of several options to guarantee trust, security, and privacy under this domain. However, it is particularly difficult to support security and privacy in the IoT . One reason of this is due to the large amount of sensitive data in the network (military, health care, financial, among others); another is due to the limited computational capabilities of the computing devices in the network, which are much more vulnerable to physical attacks and are character- ized by having low computational resources. These limitations must be taken into account during the design of privacy and security solutions for the IoT.

Since typical devices in IoT (i.e., sensors nodes in a Wireless Sensor Network) are equipped with low end micro-controllers with small word sizes and slow oscillators, software-friendly lightweight primitives are desired. However, other representative devices in IoT as the RFID tags usually do not have a software-programmable processor, requiring to realize a cryptography-based solution only through hardware implementations. Moreover, the majority of these devices use limited power sources to the point where it is required to rely on energy harvesting, power optimization techniques, and novel transmission technologies. Therefore it is difficult to provide cryptographic solutions for constrained environments. In order to realize the full potential of hardware based security for the IoT, very significant research and engineering issues have to be addressed in novel and creative ways. The use of FPGAs is of particular interest for the development of hardware systems. The nature of frequently changing and evolving security protocols necessitates the use of devices with reconfigurable capabilities. Symmetric cryptography is interesting for constrained devices due to the nature of its operations which are usually hardware-friendly. When implemented efficiently enough so as to comply with the scarce resources of the IoT devices it is said to play a major role on the security of smart objects. In 2012 ISO/IEC standardized the symmetric block cipher PRESENT, a lightweight cipher intended for constrained applications. Several hardware implementations of PRESENT This paper discusses the hardware implementation issues of the cipher PRESENT, having as main contributions:

1) Two novel designs of PRESENT cipher aiming at reducing implementation size and energy consumption, considering the key generation mechanism in the design.
2) The experimental evaluation of five different PRESENT designs, three from related works plus the two proposed in this paper, is presented and discussed. This evaluation considers not only area and performance but also energy and efficiency as metrics for comparison purposes.

The source files of the five designs evaluated are made publicly available to the community with agreement of all the authors as an effort to impulse the culture of transparency in the scientific method for this field of study.

II. THE PRESENT CIPHER

Present is a symmetric ultra-lightweight block cipher designed by ISO/IEC as “block cipher” suitable for lightweight cryptography, which is tailored for implementation in constrained environments. The cipher is based on a Substitution Permutation Network (SPN), with a round-based processing system. PRESENT supports 64-bit input data blocks and key sizes of 80 and 128 bits. To improve the security of hardware implementations the keying material can be generated using hardware primitives such as Physical

Unsolvable Function. In PRE SE NT, the state is a one-dimensional array of 64 bits that supports shift operations and parallel access over the data. The input key is processed internally to generate a round key for each of the 31 total rounds. The cipher uses three basic operations over the state, which is a structure that contains the plaintext and is modified at each round to produce confusion and diffusion over the data

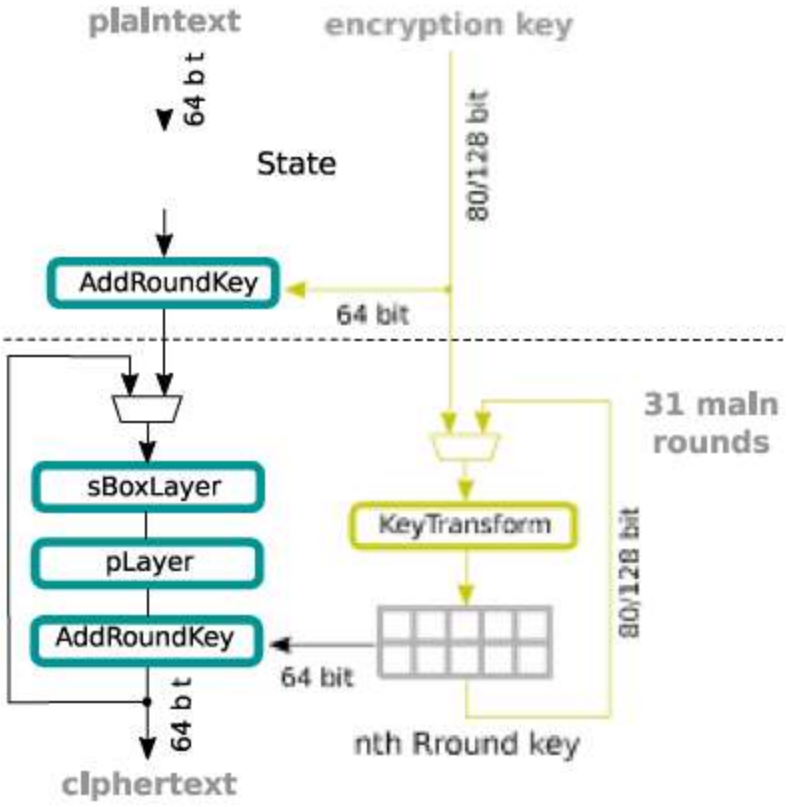


Fig. 1. Encryption procedure of PRES ENT

The basic block diagram of PRESENT shown in Fig. 1
Closely follows its algorithm specification. A 64-bit data path enables the execution of an entire round in a single cycle, requiring in turn sixteen 4-bit substitutions (SBOX) and a 64-bit permutation. This design is derived directly from the algorithm specification and the latency is equivalent to the number of rounds.

III. LIGHTWEIGHT HARDWARE

This section is a review of optimized hardware architectures for PRESENT reported in the literature. For each design the basic outline, latency, and estimated implementation size is provided. The two PRESENT designs proposed in this paper are also described. Thus, a total of seven different hardware architectures for PRESENT are discussed in this section.
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. 2.

There are two disadvantages in that proposal. First, the vari
ation 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.

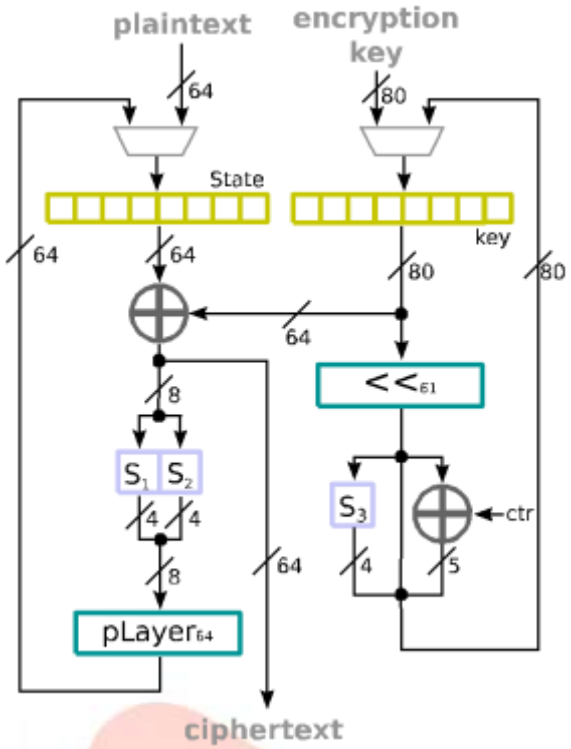


Fig. 2. Area-optimized implementation of PRESENT

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. 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 . For the equivalence of a 4-bit substitution box, the reader can refer to the estimation provided in the original proposal of PRESENT, 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.

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. 3.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.

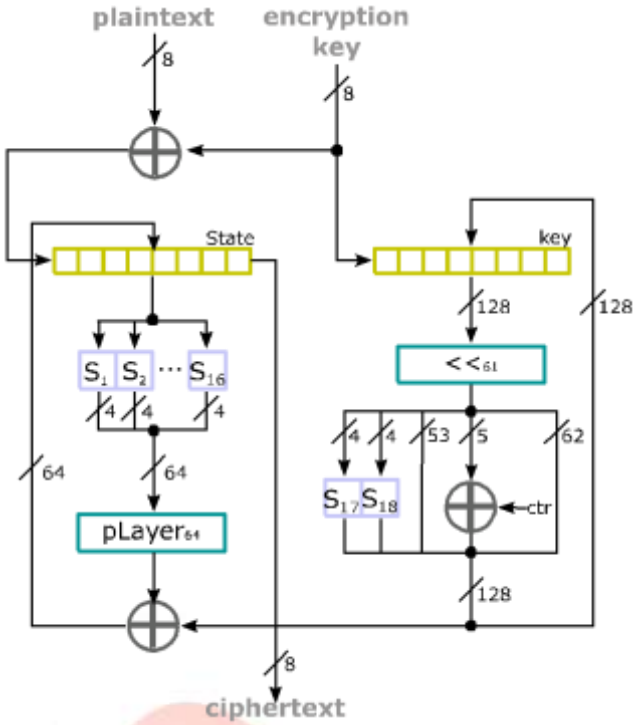


Fig.3. Iterative architecture for PRES E NT

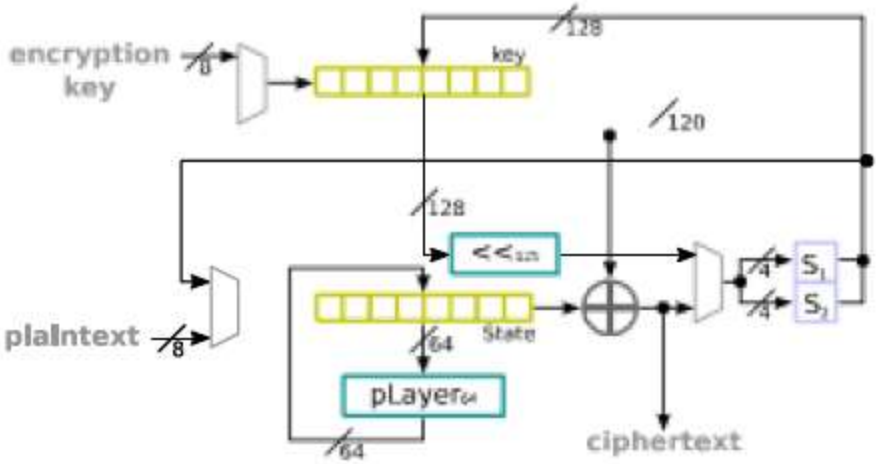


Fig. 4. Serial architecture for PRES E NT

To achieve a lightweight implementation of PRE SE NT by following the design of the serial architecture, and replacing the substitution boxes with a construction based on Boolean logic. The authors attempted to construct the PRE SE NT 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 look- up-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.

In this design, the aim is to reduce the data path width considering both, the substitution layer (sBox Layer) and the permutation layer (pLayer). The reduction of the substitution layer follows the conventional approach, so the data path can be adjusted to any width divisible by four, that is, the total input bits in a PRE SE NT's substitution box. The reduction of width in the permutation layer is achieved thanks to a pattern in the structure of the function itself. With that reduction, the substitution layer can take a width of 16-bit too, thus requiring only four substitution boxes. Fig. 5 illustrates the data path of the 16-bit architecture for PRE SE NT. In this proposal the authors claim to achieve a lightweight implementation of PRE SE NT by following the design of the serial architecture, and replacing the substitution boxes with a construction based on boolean logic.

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. 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.

In this design, the aim is to reduce the data path width considering both, the substitution layer (sBox Layer) and the permutation layer (p Layer). The reduction of the substitution layer follows the conventional approach, so the data path can be adjusted to any width divisible by four, that is, the total input bits in a PRE SE NT's substitution box. The reduction of width in the permutation layer is achieved thanks to a pattern in the structure of the function itself.

Only two data ports are required in this design, 16-bit to take in the plaintext and 16-bit to produce the output. To input the data, 4 cycles are consumed, 124 cycles are then used to process the state, and finally 4 more cycles are required to produce the output, giving a total latency of 132 cycles. This work presents an attractive data path, but the key generation is not properly addressed.

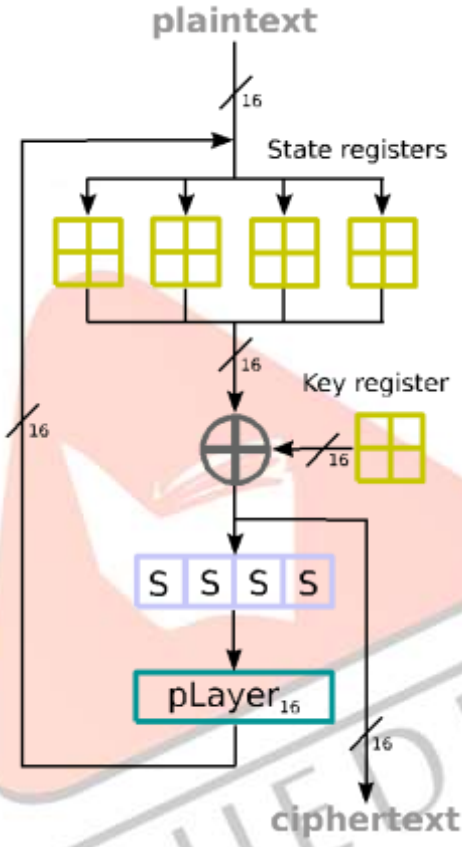


Fig. 5. 16-bit architecture for the PRES E NT cipher

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 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. 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.The base design for the two architectures proposed in this paper follow the strategies reported in [20] in relation to the construction of the data path. The data path design is illustrated in Fig. 6.

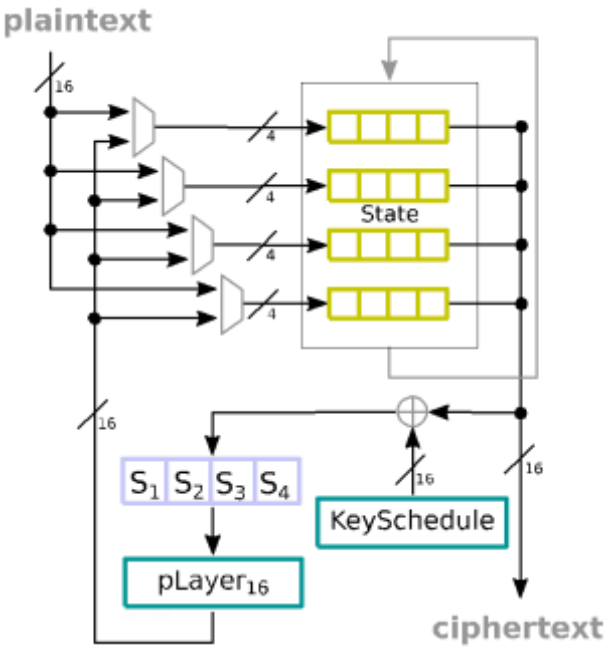


Fig. 6. Data path for the PRESENT architecture proposed in this work

This is an area optimized implementation of PRESENT using a 128-bit key. 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 the64-bit state and during a 9th cycle in which the 64-bit permutation takes place, the round key is also updated. Fig. 4 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 1.

IV. EXPERIMENTAL EVALUATION

A. Configurations

To compare the implementation results with the state of the art architectures under fair conditions, the source files from [18] and [19] were requested to synthesize all the designs for the same device and with the same implementation parameters and tools. Only the source files for the iterative and serial architectures from [18] were provided by the authors. Having access to the source files of the design in [20] and together with the source of the proposed designs in this paper, five different PRESENT designs were implemented, evaluated and analyzed.

TABLE I

SUMMARY OF DIFFERENT HARDWARE ARCHITECTURES FOR PRESENT

Year	Ref.	Name	Key size	Area (GE)	Latency (Cycles)	Count
2007	[24]	Basic	80	1570	33	
2007	[24]	Low-area	80	1135	250	898
2008	[18]	Iterative	128	1797	55	1566
2008	[18]	Serial	128	1029	302	1054
2010		Serial + b 1 SBOX	128	1243	303	1054
2014		16-bit	80/128	576[b]	132	528[b]
2015	This	16-bit + 80 b t keys	80	989	133	926
2016	This	16-bit + 128-bit keys	128	1257	136	1194

257

Each one of the evaluated architectures included I/O mecha- nisms that allow to use the hardware module as an independent core or in an integrated system. Each implementation was made targeting a specific FPGA board to generate the physical

constraints file. All the source files and relevant data is available at http://www.tamps.cinvestav.mx/~hardware under a free software license. The five configurations of PRESENT (summarized in Table II) are:

1) Iterative Architecture (C1): It is the architecture reported in [18] that closely follows the specifications of the algorithm presented in [24]. The source files for this implementation were provided by its creators [18].

2) Serial Architecture (C2): This is an area optimized implementation of PRE SE NT using a 128-bit key. The source files for this implementation were also provided by its creators [18].

3) 16-Bit Architecture (C3): This configuration is based of the approach where the round keys are used to generate a key module which is embedded in the architecture as combinational logic. This design is akin to that presented in [20] and the implementation was revamped with ideas presented in [18].

4) 16-Bit Architecture With 80-Bit Keys (C4): This design Represents the design proposed in this work using keys of 80-bit.

5) 16-Bit Architecture With 128-Bit Keys (C5): This architecture consists of the area optimized data path proposed in Section III with a key schedule to process 128-bit keys.

B. Environment

The five PRESENT architectures to be evaluated were syn- the sized for Xilinx FPGAs using the ISE Design Suite 14.7 As computing platform, a set of low cost development boards were considered. To study the architectures in LUT-4 FPGAs the Spartan-3 (xc3s200-5ft256) and Virtex-4 (xc4vlx25-12ff668) were used. In regards to LUT-6 technology the Spartan-6 (xc6slx16-3csg324) and Virtex-5 (xc5vlx50t-3ff1136) were used. As computing platform, a set of low cost development boards were considered. To study the architectures in LUT-4 FPGAs the Spartan-3 (xc3s200-5ft256) and Virtex-4 (xc4vlx25-12ff668) were used. In regards to LUT-6 technology the Spartan-6 (xc6slx16-3csg324) and Virtex-5 (xc5vlx50t-3ff1136) were used. Newer FPGAs such as the 7 series were not considered since according to the manufacturer the Configurable Logic Blocks in these FPGAs are similar to those used in FPGAs of the 6 series such as the Spartan 6, whichLow-resource implementations usually do not use the maxi- mum possible frequency, but limit it to a lower frequency, as is the case in RFID applications where a frequency of 13.56MHz is used. might lead to similar implementation results in regards to the power estimation depends on both, the size, and the signal activity of the implementation. The energy will then be directly affected by the latency if all the architectures are evaluated using a constant operational frequency. Since all the evaluated implementations have the same block size, the energy-per-bit has a linear proportion to the energy. This metric is useful to compare the results provided against those of ciphers with different block sizes however.

TABLE III

RESOURCE USAGE AND PERFORMANCE RESULTS FOR THE FIVE ARCHITECTURES UNDER EVALUATION

Work	Design	(bit)	KEY (bit)	FF	LUT	SLC	FMAX (MHz)	LAT (cycles)			Thr/SLC (Kbps/Slice)
xc6slx16-3csg324											
[18]	C1	64	128	200	202	58	160.21	**55**			
[18]	C2	64	128	200	157	**44**	164.23	300			
[20]	C3	64	128	**73**	**147**	48	206.40	132			
This work.	C4	64	80	153	170	48	**257.40**	135			
This work.	C5	64	128	201	220	61	210.66	136			
xc3s200-5ft256											
[18]	C1	64	128	200	381	191	179.95				
[18]	C2	64	128	200	258	131	177.34				
[20]	C3	64	128	**73**	280	153	120.71				
This work.	C4	64	80	153	**215**	**124**	**213.81**				
This work.	C5	64	128	201	264	151	194.63				
xc5vlx50t-3ff1136											
[18]	C1	64		200	283	88	271.67		**316.12**		
[18]	C2	64		200	237	72	289.69		61.19		
[20]	C3	64		**73**	**182**	75	321.96		156.19		
This work.	C4	64		153	190	**67**	**542.30**		260.96		
This work.	C5	64		201	239	73	431.78		203.19		
xc4vlx25-12ff668											
[18]	C1	64	128		382	192	284.33			**15.78**	**82.18**
[18]	C2	64	128		258	131	288.52			2.86	21.81
[20]	C3	64	128		279	151	223.51			6.57	42.54
This work.	C4	64	80		**215**	**124**	**375.66**			6.53	52.62
This work	C5	64	128		265	152	364.56			6.38	41.98

* Using a frequency of 13.56 MHz.

TABLE IV

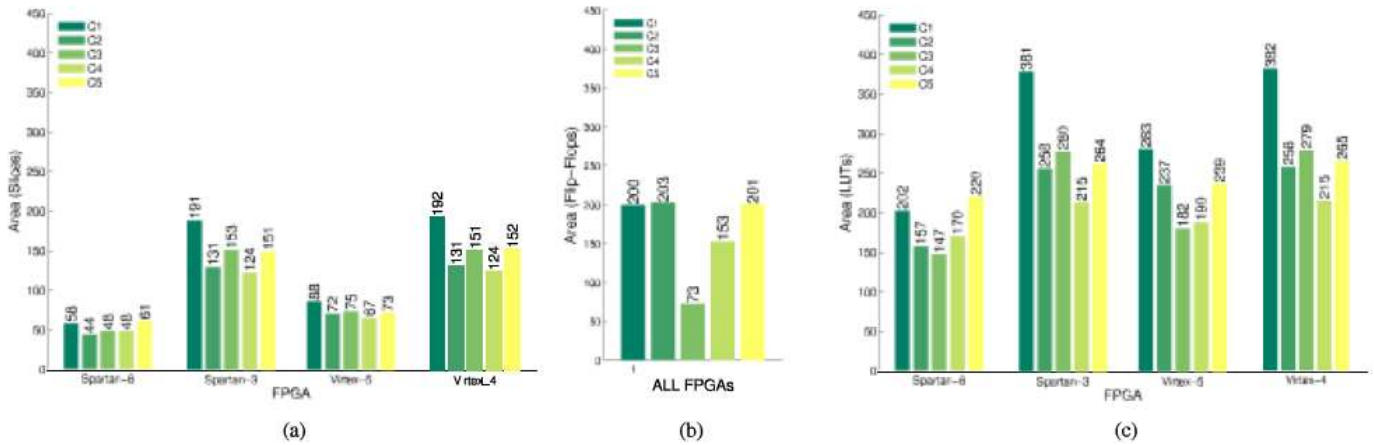
POWER AND ENERGY CONSUMPTION RESULTS FOR THE FIVE ARCHITECTURES UNDER EVALUATION

Work	D	STATE (bit)	KEY (bit)			Dynamic POW (mW)			ENG/bit (nJ/bit)
[18]	C1	64	128						
[18]	C2	64	128						
[20]	C3	64	128						
This work	C4	64	80						
This work	C5	64	128						
[18]	C1	64							
[18]	C2	64							
[20]	C3	64							
This work	C4								
This work	C5								
[18]	C1		128						
[18]	C2		128						
[20]	C3		128						
This work	C4	64	80						
This work	C5	64	128						
		64							

V. RESULTS

Table III presents the performance and resource usage results for the five architectures under evaluation in the four FPGAs devices. Table IV presents the power and energy consumption results.

The resource usage for all the architectures is presented for the four FPGAs selected as implementation platform, this metric is illustrated in Fig. 9. The results are consistent for both LUT-4 FPGAs. In the case of the LUT-6 platforms it can be noted how implementations in the Spartan-6 FPGA use less LUT elements, which derives in lower slice counts than those of the implementations in the Virtex-5 FPGA. This is due to slight variations in the slice architecture for both FPGAs require low resource usage but also have high performance constraints determined by the application. The later enables a comparison across all the implementations which can be useful for systems that require low resource usage and can accept compromises in the performance. The frequency of 13.56 MHz is utilized since it is appropriate for RF applications, which is the case of some IoT transmitters. This metric will be used to discuss the performance results. The throughput calculated using a constant frequency can also be matched with the energy consumption analysis, which is calculated using the same frequency. From the maximum frequency it can be noted that the results depend not only on the implementation, but also



(a) (b) (c)

Fig. 9. Resource usage for the different configurations in the different FPGAs utilized. (a) Slices. (b) Flip-Flops. (c) LUTs.

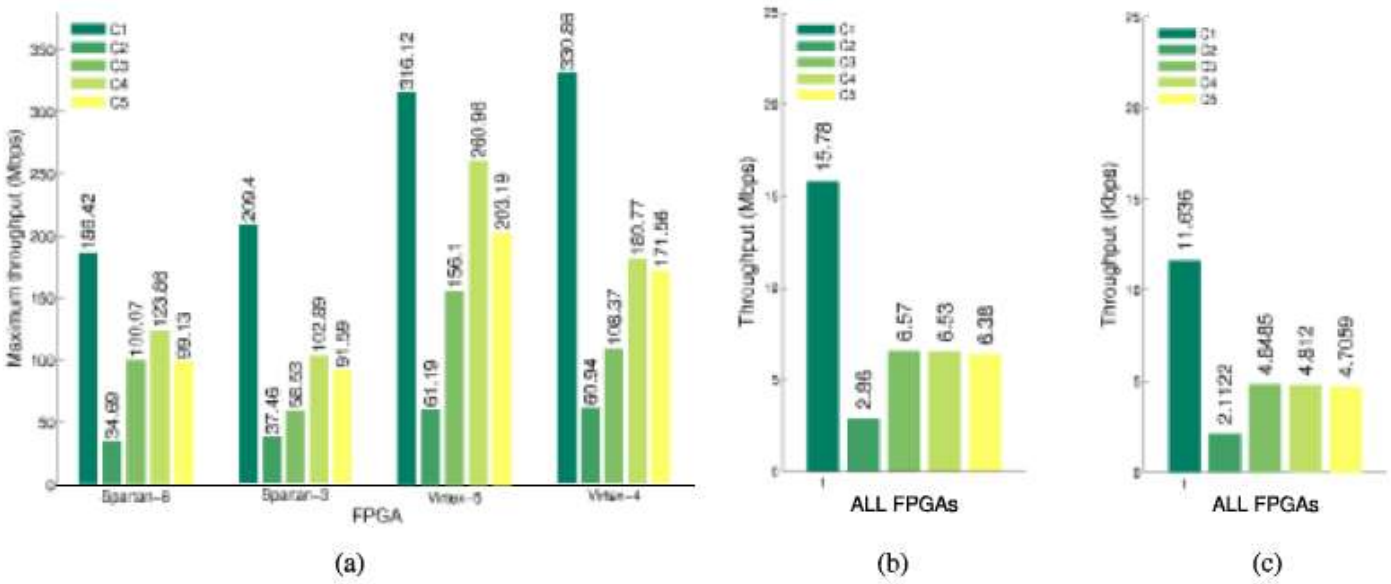


Fig. 10. Throughput for the five PRES E NT designs, using different FPGAs and different operational frequencies. (a) Using the maximum frequency. (b) At 13.56 MHz. (c) At 100 KHz.

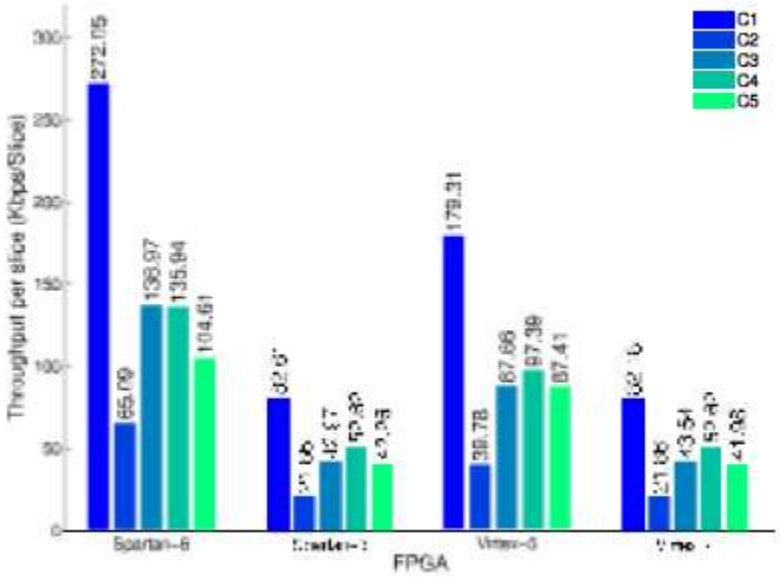


Fig. 11. Throughput per slice for the different configurations in the different FPGAs utilized.

on the underlying FPGA platform. The performance compar- ison for the different implementations is shown in Fig. 10 using the maximum frequency of the implementation, the fre- quency recommended in [32] of 13.56MHz, and the frequency of 100KHz, which is commonly used and reported in related works.

The throughput per slice is a metric utilized to illustrate the efficiency of the architectures when it is desired to study the trade-off between the area reduction and the performance of the implementations. In this case the non-optimized implemen- tation of PRE SE NT will have the maximum efficiency, and the area-optimized architectures can be ranked from this reference. Note that the implementation with a reduced key size should be compared having in mind that it also features a security trade-off. Fig. 11 illustrates the throughput per slice for the different configurations.

The analysis performed for each architecture delivers power and temperature estimations based on a user defined opera- tional frequency and temperature. It was determined to use a frequency of 13.56 MHz for all the studies and the default operation temperature. Since the goal of this experiment is to study the energy consumption, only the power results were used. In most of them it is shown how the static power remains constant across the different implementations for the same FPGA board. Regarding the dynamic power, it can be noted how it changes depending on the switching activity of the circuit. The total power is the sum of the static and dynamic power. The power analysis demonstrates how selecting the appropriate FPGA board can deliver a change with a signif- icance of an order of magnitude. A graphic comparison for

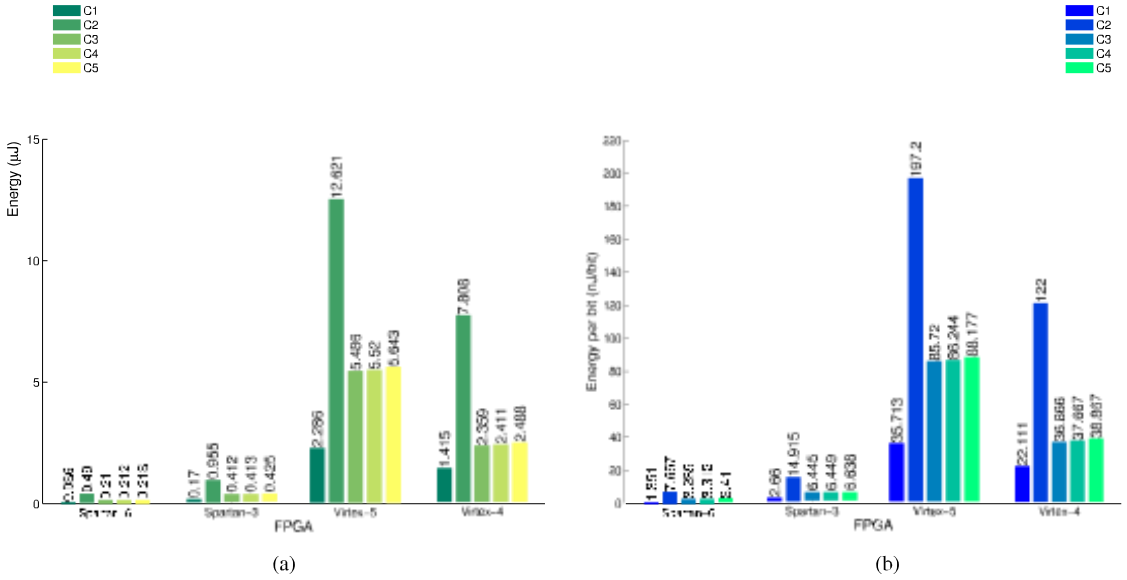


Fig. 12. Results of the energy analysis for the different configurations in the different FPGAs utilized. (a) Energy consumption. (b) Energy-per-bit.

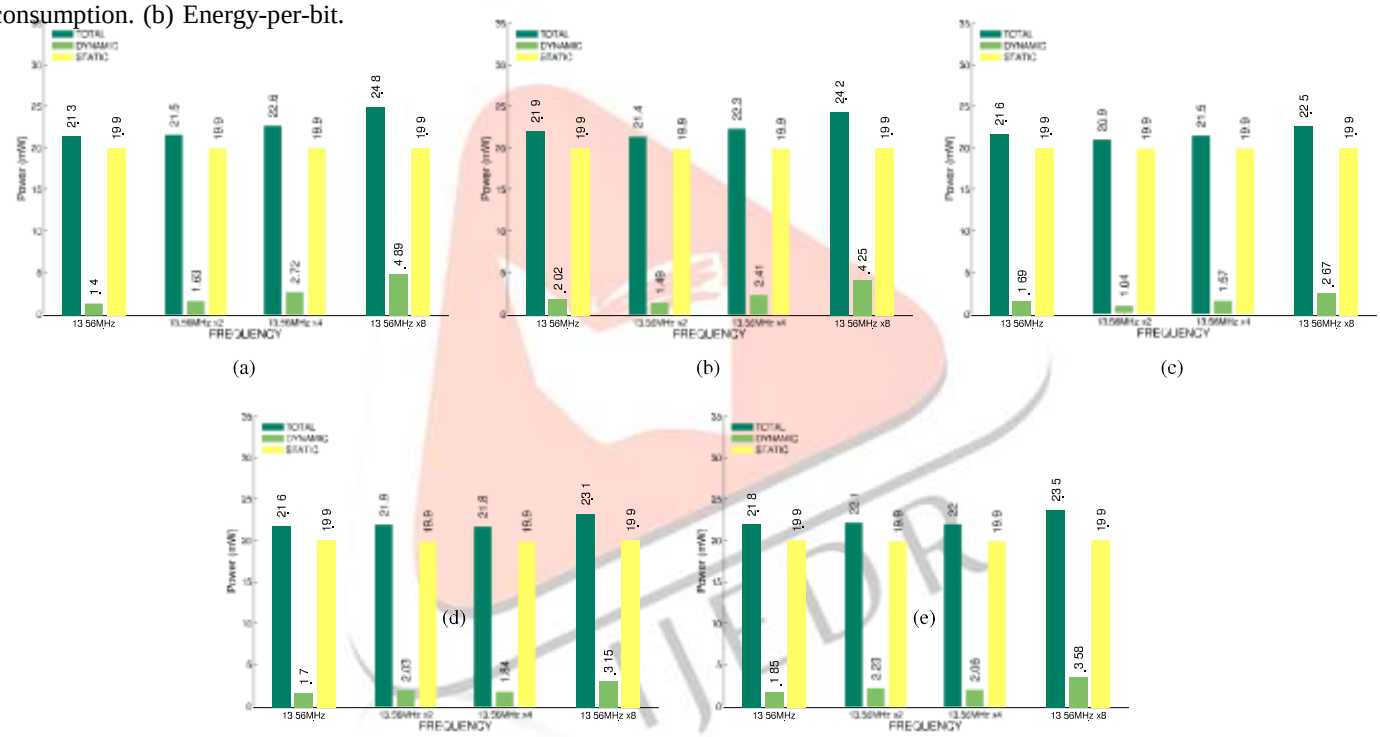


Fig. 13. Power as a function of the operational frequency for the five PRES E NT architectures evaluated in the Spartan-6 FPGA. (a) Architecture C1. (b) Architecture C2. (c) Architecture C3. (d) Architecture C4. (e) Architecture C5.

the energy consumption for the different implementations is shown in Fig. 12a. The energy per bit metric, used in this work also as an efficiency measurement, represents the energy cost associated to process a single bit of the message. Since in this particular case all the architectures under study have the same state size, the energy per bit will be directly related to the energy. This can be of interest to compare these results with those obtained from implementations of algorithms with different state size. In Fig. 12b the energy per bit for the different implementations is presented. Some application scenarios may require to operate not at 13.56MHz but at a multiplier of this frequency. This however can represent an increment in the power consumption of the architecture.

VII. CONCLUSIONS

This paper presented a comparison of hardware architec- tures for the PRE SE NT cipher. Two alternatives have been studied to generate the round keys required by the algorithm. A 16-bit datapath architecture with 128-bit key schedule was presented and can be compared directly to relevant works in the literature. A 16-bit datapath architecture with 80-bit key schedule was developed for applications where an area/security trade-off can be established.Experimental results for the proposed architectures and the most relevant implementations of PRE SE NT in the state of the art were obtained. The results presented are derived from a fair experimentation. The different evaluations were conducted following well defined methods such that it is possible to replicate the presented results using the source files for all the architectures analyzed in this work. The source files were released

with permission from the authors under a free software license. Of the architectures reviewed the iterative design (C1) achieves the best results in performance and energy consumption. In contrast, the serial architecture(C2) produces the lowest implementation size but registers the worst mea- surements for performance and energy consumption. From the architectures proposed in this paper, the one that utilizes 128-bit keys (C5) features an efficient trade-off, in terms of throughout-per-slice and energy-per-bit for small applications that also have performance constraints such as IoT nodes. The architecture using 80-bit keys (C4) is a good alternative for applications that require small implementation area with good performance, at the cost of a smaller key size. All the implementations for the architectures evaluated in this work have been published so that anyone interested can replicate the experimental results presented and use the designs under a free software license.

REFERENCES

[1] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT systems: Design challenges and opportunities," in Proc. IEEE/ACM Int. Conf. Comput.- Aided Design (ICCAD), Piscataway, NJ, USA, Nov. 2014, pp. 417–423.
[2] Z. Zou et al., "A low-power and flexible energy detection IR-UWB receiver for RFID and wireless sensor networks," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 7, pp. 1470–1482, Jul. 2011.
[3] SRI Consulting Business Intelligence, Disruptive Civil Technologies: Six Technologies With Potential Impacts on US Interests Out to 2025. [Online]. Available: https://fas.org/irp/nic
[4] T. Macaulay, "Introduction—The Internet of Things," in RIoT Control: Understanding and Managing Risks and the Internet of Things. Boston, MA, USA: Morgan Kaufmann, 2017, ch. 1, pp. 1–26.
[5] D. Giusto, A. Iera, G. Morabito, and L. Atzori, The Internet of Things. New York, NY, USA: Springer, 2010, pp. 1–32.
[6] C. Alippi and C. Galperti, "An adaptive system for optimal solar energy harvesting in wireless sensor network nodes," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 6, pp. 1742–1750, Jul. 2008.
[7] A. A. R. Haeri, M. G. Karkani, M. Sharifkhani, M. Kamarei, and A. Fotowat-Ahmady, "Analysis and design of power harvesting circuits for ultra-low power applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 64, no. 2, pp. 471–479, Feb. 2017.
[8] M. Kneževic´, V. Nikov, and P. Rombouts, "Low-latency encryption— Is 'lightweight = light + wait?"' in Cryptographic Hardware and Embedded Systems. Berlin, Germany: Springer, 2012, pp. 426–446.
[9] C. J. Mcivor, M. Mcloone, and J. V. Mccanny, "Hardware elliptic curve cryptographic processor over GF(p)," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 9, pp. 1946–1957, Sep. 2006.
[10] T. Good and M. Benaissa, "Very small FPGA application-specific instruction processor for AES," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 7, pp. 1477–1486, Jul. 2006.
[11] N. Smyth, M. McLoone, and J. V. McCanny, "WLAN security proces- sor," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 53, no. 7, pp. 1506–1520, Jul. 2006.
[12] 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.
[13] Information Technology—Security Techniques—Lightweight Cryptography—Part 2: Block Ciphers, document ISO/IEC 29192-2, Jan. 2012.
[14] X. Guo, Z. Chen, and P. Schaumont, "Energy and performance eval- uation of an FPGA-based SoC platform with AES and PRESENT coprocessors," in Embedded Computer Systems: Architectures, Model- ing, and Simulation. Berlin, Germany: Springer, 2008, pp. 106–115. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-70550-5_12
[15] M. Sbeiti, M. Silbermann, A. Poschmann, and C. Paar, "Design space exploration of PRESENT implementations for FPGAS," in Proc. 5th Southern Conf. Program. Logic (SPL), Apr. 2009, pp. 141–145.