Lightweight Hardware Architectures for the Present Cipher in FPGA

Carlos Andres Lara-Nino, Arturo Diaz-Perez, and Miguel Morales-Sandoval

***Abstract*— In recent years, the study of lightweight symmetric ciphers has gained interest due to the increasing demand for security services in constrained computing environments, such as in the Internet of Things. However, when there are several algorithms to choose from and different implementation criteria and conditions, it becomes hard to select the most adequate security primitive for a specific application. This paper discusses the hardware implementations of PRESENT, a standardized lightweight cipher called to overcome part of the security issues in extremely constrained environments. 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. From this wide experimental evaluation, to the best of our knowledge, new records are obtained in terms of implementation size and energy consumption. In particular, our designs result to be adequate in regards to energy-per-bit and throughput-per-slice.**

***Index Terms*— PRESENT cipher, IoT, lightweight cryptography, low-area, low-energy.**

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 [1]. According to [2], 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 [3], 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 [3]. These security risks are the central issues that may delay the development and adoption of IoT applications [4]. 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 [5]. 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 characterized 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 [6], power optimization techniques [7], and novel transmission technologies [2]. Therefore it is difficult to provide cryptographic solutions for constrained environments. A such solution, as mentioned in [8], must be a balancing act between several aspects as cryptographic strength, area implementation costs, execution speed and power consumption. Hardware-based security has been pointed out as ideal for the IoT [1], requiring a qualitative breakthrough in design and optimization techniques for the proposed security architectures. The use of hardware-based cryptographic solutions intrinsically can provide significant security improvements over software solutions [9]. 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. As it is mentioned in [10] it is no longer true that FPGAs are only used for prototyping, that even if their inclusion in production models seems expensive, they bring additional advantages such the ability to update the design and reduce time to market. Furthermore, as pointed out in [11] 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 [12]. In 2012 ISO/IEC standardized the symmetric block cipher PRESENT [13], a lightweight cipher intended for constrained applications. Several hardware implementations of PRESENT have been reported since its creation [14]–[20]. Particularly,

Manuscript received December 7, 2016; revised February 24, 2017; accepted March 20, 2017. This work was supported by CONACYT under Grant 393070. This paper was recommended by Associate Editor A. S. Vincentelli.

The authors are with the Center for Research and Advanced Studies of the National Polytechnic Institute of Mexico, Campus Tamaulipas, Victoria City 87130, Mexico. (e-mail: clara@tamps.cinvestav.mx; adiaz@tamps.cinvestav.mx; mmorales@tamps.cinvestav.mx).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2017.2686783

in [18] and [19] optimization goal is the reduction of the area size, reporting the most compact designs known to their respective dates. However, not only area restrictions should be taken into account; hardware implementations must also consider energy consumption and performance issues. In [20] a novel architecture was proposed to reduce the implementation size and the latency in aims to achieve a compact design with low energy consumption.

A. Contributions

This paper discusses the hardware implementation issues of the cipher PRESENT, having as main contributions:

1) Two novel designs of PRESENT 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.

B. Structure of the Paper

The rest of the paper is organized as follows. Section II briefly reviews the PRESENT algorithm. Section III reviews the most relevant hardware architectures for PRESENT reported in the literature and elaborates on the design of the hardware architecture for PRESENT described in this paper. Section IV describes the methodology for the experimental work. Section V provides the experimental results and the comparisons of the architectures under evaluation. Section VI discusses the implementation results for the designs proposed in this work. Finally, Section VII outlines the conclusions of this work.

II. The PRESENT Cipher

PRESENT is a symmetric ultra-lightweight block cipher standardized in 2012 by ISO/IEC as "[block cipher] suitable for lightweight cryptography, which is tailored for implementation in constrained environments" [13]. 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 Unclonable Functions [21]–[23]. In PRESENT, the state is a 1-D array of 64 bits that supports shift operations and parallel access over the data. The input key is processed internally to generate a roundkey 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:

- `AddRoundKey`: adds the state to a 64-bit word from the roundkey using finite field arithmetic.

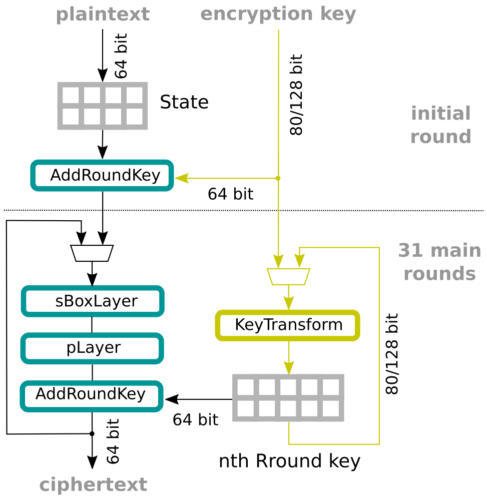


Fig. 1. Encryption procedure of PRESENT. The plaintext stored in a state is modified along 31 rounds using substitution and permutation operations to add confusion and diffusion to the data. At each round, a key is derived from the one given as input and combined with the state.

- `sBoxLayer`: effectuates a 4-bit to 4-bit substitution in the state using a Substitution Box (SBOX) with 16 values.
- `pLayer`: applies bit level shifts over the state.

For a detailed specification of PRESENT the reader could refer to [24].

The basic block diagram of PRESENT shown in Fig. 1 closely follows its algorithm specification. A 64-bit datapath 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 Architectures of PRESENT

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.

A. Low-Area Architecture – Andrey Bogdanov et al. (2007) [24]

The first area-optimized architecture of PRESENT was proposed by its creators in [24]. 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 variation of size in the datapath 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.

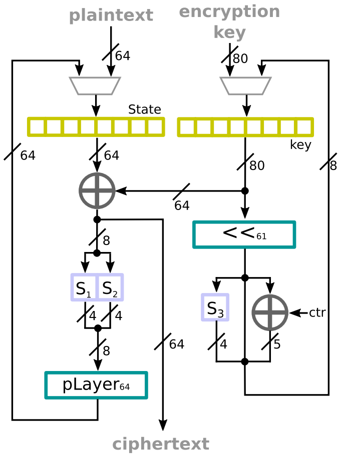


Fig. 2. Area-optimized implementation of PRESENT as proposed in [24]. The main feature of this design is the use of a small number of substitution boxes to reduce resource usage.

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 [25]). 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 [24]. For the equivalence of a 4-bit substitution box, the reader can refer to the estimation provided in the original proposal of PRESENT [24], which is said to be of 28 GE approximately.

If using two substitution boxes to process the datapath 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 datapath 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 – Neil Hanley and Marie O'Neill (2012) [18]

This architecture is reported in [18]. 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. 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.

As appreciated in Fig. 3 this design utilizes 197 FF (985 GE), 77 XOR gates (308 GE), 18 SBOX (504 GE), a 64-bit permutation (0 GE), and a 61-bit shift (0 GE). That is, the design reported in [18] has a cost in area of 1797 GE approximately.

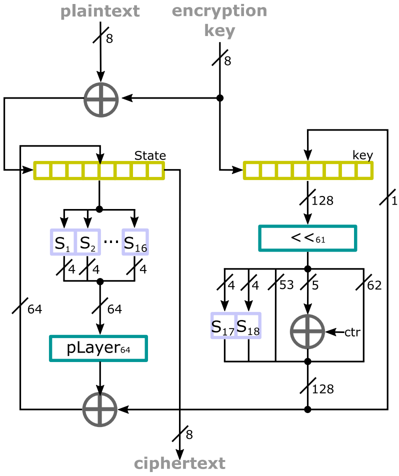


Fig. 3. Iterative architecture for PRESENT as proposed in [18]. This architecture follows closely the specification of PRESENT, featuring high performance and efficiency.

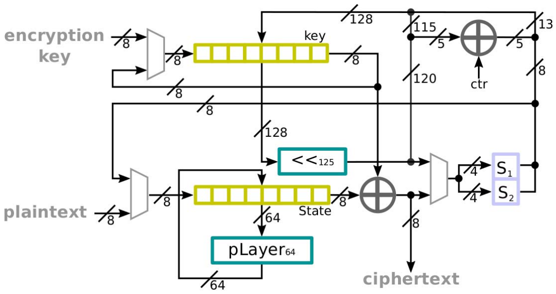


Fig. 4. Serial architecture for PRESENT as proposed in [18]. By minimizing the number of substitution boxes and reducing parts of the datapath to be 8-bit wide, this design achieves a reduction in its implementation size; however, its performance is affected as result of that trade-off.

C. Serial Architecture – Neil Hanley and Marie O'Neill (2012) [18]

This is an area optimized implementation of PRESENT using a 128-bit key [24]. 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. 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 1093 GE.

D. Serial Architecture With Boolean SBOX – J. J. Tay et al. (2015) [19]

In this proposal the authors claim to achieve a lightweight implementation of PRESENT by following the design of the serial architecture [18], 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 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.

E. 16-Bit Architecture – Andres Lara et al. (2016) [20]

In this design, the aim is to reduce the datapath width considering both, the substitution layer (sBoxLayer) and the permutation layer (pLayer). The reduction of the substitution layer follows the conventional approach, so the datapath can be adjusted to any width divisible by four, that is, the total input bits in a PRESENT's substitution box. The reduction of width in the permutation layer is achieved thanks to a pattern in the structure of the function itself. This strategy was first observed in [16] and detailed later in [17]. Such strategy consists in exploiting the regularity of the 64-bit permutation. Using said pattern it is possible to shrink down the width of the permutation layer from 64-bit to 16-bit. With that reduction, the substitution layer can take a width of 16-bit too, thus requiring only four substitution boxes. Fig. 5 illustrates the datapath of the 16-bit architecture for PRESENT.

Lara-Nino *et al.* [20] claim that a unified 16-bit datapath reduces the additional logic demanded by a variable datapath width, which is observed in conventional optimizations. That is because transferring data from a wide word to a smaller one or vice versa requires additional shift registers and multiplexers which increase the latency and the area usage.

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 datapath, but the key generation is not properly addressed. It is proposed to store all the 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 can not 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.

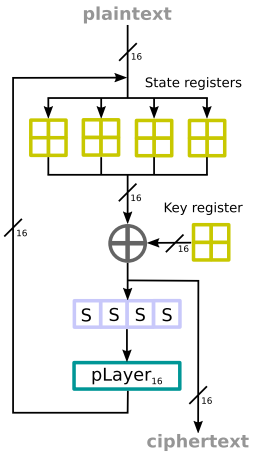


Fig. 5. 16-bit architecture for the PRESENT cipher as proposed in [20]. This architecture has a datapath with constant width of 16-bit and reduced implementation size with a moderate compromise to its performance. The key mechanism is not addressed in that work.

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 [26]–[28].

In this archtiecture the cipher's datapath 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 datapath 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.

F. Proposed Architectures

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

To implement the state storage, ideas were taken from [20] by using four 16-bit registers, and from [18] by implementing these registers with both parallel and shift access. To input the data, the 64-bit block is partitioned in four 16-bit words. Each word is then distributed in the four registers as 4-bit words. The content in the registers is shifted to the right to input the data. This helps to reduce the size of the MUXs

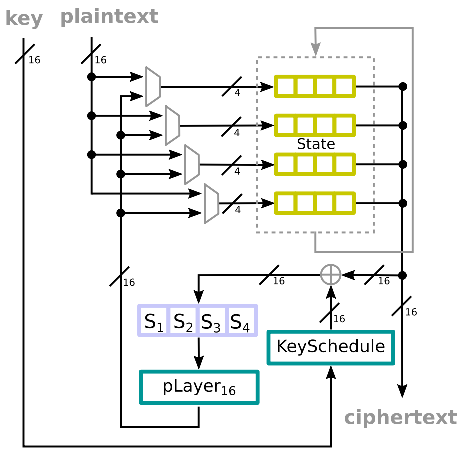


Fig. 6. Datapath for the PRESENT architecture proposed in this work. This design features a constant datapath of 16-bit with improved data registers and key mechanism for 80-bit and 128-bit keys. For simplicity, the key generation mechanism is shown as a small block.

required to load the data into the registers. At each round of the algorithm the part of the state to be processed is made of the four right-most bits in each state register. These 16-bit are XORed with the appropriate 16-bit round key word and then processed by the substitution and permutation layers. The 16-bit words of the state are then stored as 4-bit words in the state registers, whose content is then shifted to the right. In this case, the additional permutation is not required to be hardwired but realized by updating the content of the registers at the end of each round. This change allows to execute the initial permutations sequentially and the additional permutation in parallel. As a consequence, the latency is reduced by 1 cycle per round.

One of the differences of the architectures proposed in this paper with the work reported in [20] is the key generation scheme. For the generation of the round-keys this work explores two alternatives:

- to process 80-bit input keys and use standard logic to produce each round key.
- to process 128-bit input keys and use standard logic to produce each round key.

The first alternative is based on the original proposal of the cipher. The authors of PRESENT in [24] designed the algorithm with a relative short key to be used in constrained environments that could forfeit some security in benefit of reduced implementation size. In this case, the key schedule is implemented using four 20-bit registers that store the initial key and update its value each round. In each round the 16-bit key consists of the right-most four bits in the four key registers. The content of the registers is shifted to the right to extract the adequate key for the 16-bit state word being processed. At each round, 64 bits out of the 80 bits contained in the registers are used. The round key is updated at the end of each round in parallel with the data processing so that no additional cycles are required. Five cycles are necessary to divide the input key in words of 16-bit, however. This design is shown in Fig. 7.

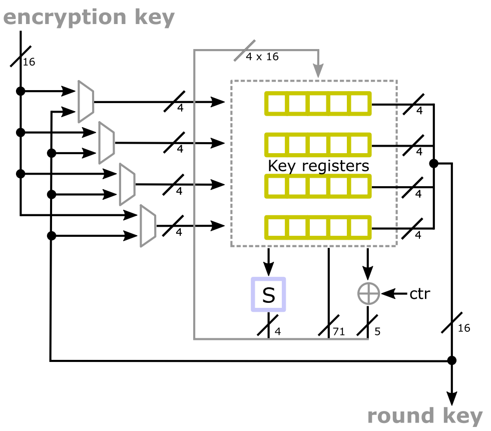


Fig. 7. Key schedule for the PRESENT cipher using 80-bit keys.

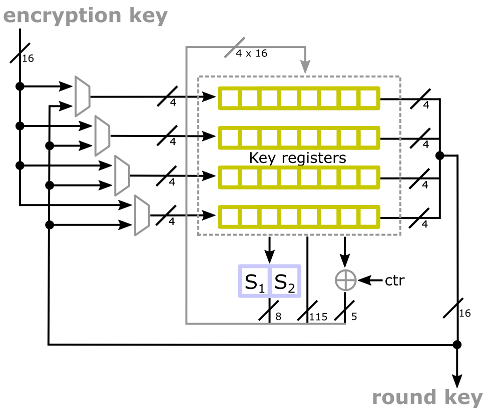


Fig. 8. Key schedule for the PRESENT cipher using 128-bit keys.

If a higher security level is desired, the original proposal of PRESENT also considers a key schedule to process 128-bit input keys. To implement this mechanism, four 32-bit registers are used. The input key is introduced in words of 16-bit, which are then partitioned into 4-bit words, and each one is introduced to one of the 32-bit registers. The contents of the registers are shift to the right to obtain the input data. This process to input the data is identical to the one used with 80-bit keys, with the difference that for a 128-bit key, 8 input cycles are required. At each round, the actual 16-bit word of the key is made of the rightmost four bits in the key registers, in this case only 64-bit out of the 128-bit contained in the registers are used. As it happens with the 80-bit keys, the 128-bit round key is updated at the end of each round. Fig. 8 illustrates this key generation scheme. Note that the key schedule shown in Figures 7 and 8 is intended to serve as the "KeySchedule" module illustrated in Figure 6.

The proposed architecture for PRESENT is able to process the four 16-bit words of the state in four cycles. The extra permutation over the four 16-bit full registers is executed in the fourth cycle. This strategy delivers a latency of four cycles per round. To input the plaintext, four cycles are required, and if a key is required to be introduced, then a 80-bit key will require an additional cycle while a 128-bit key will need four additional cycles. Since four cycles are also required to produce the ciphertext output, the proposed architecture will take 133 or 136 cycles to process a 64-bit plaintext block, depending on the nature of the key generation process.

In case that the cipher module is used in application scenarios involving exposure to insecure environments (such as in Wireless Sensor Network motes), it can be considered incorporating the proposed architecture with masking techniques [29] and energy-oriented protection against side channel attacks [30] to improve its resilience against physical attacks.

For this design, the cipher's datapath requires four 16-bit registers realized with 64 FF with a cost of 320 GE, a key addition step formed of 16 XOR with a cost of 64 GE, four substitution boxes that sum 112 GE to the count, and the permutation layer which has no cost. The control requires 7 FF for the round counter and 2 FF for the state machine, both with a cost of 45 GE. From this, the 16-bit datapath has an estimate cost of 541 GE.

The key schedule for 80-bit keys requires four 20-bit registers with a cost of 400 GE, a five XOR addition of the round counter with a cost of 20 GE, one substitution box with cost of 28 GE, and key shifts with no cost. If this key scheme is used, the final cost for the architecture will be approximately 989 GE.

If a 128-bit key is used, the key schedule will need four 32-bit registers which costs 640 GE, five XOR with a cost of 20 GE for the round counter addition, two substitution boxes with a cost of 56 GE, and key shifts with no cost. Under this configuration, the architecture will have a cost of 1257 GE.

G. Summary

All the different hardware architectures of PRESENT discussed in this section are summarized in Table I. The number of IO ports, key size, estimated resource consumption in GE, and latency cycles for each implementation are registered. It can be noted how the architectures proposed in this paper achieve smaller or comparable implementation size estimations while also featuring latency reductions with regards to the most optimized works reported in the literature (Low-area [24], Serial [18], Serial + boolean SBOX [19]). This is believed to help reducing the energy consumption of the proposals. Table I also presents the circuit count for the different architectures under review. The circuit count is obtained as the sum of all the independent gates in the design and assuming that in aggressive processes, NAND and XOR gates are often the same size.

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)	Circuit Count
2007	[24]	Basic	80	1570[a]	33	-
2007	[24]	Low-area	80	1105	250	898
2012	[18]	Iterative	128	1797	55	1566
2012	[18]	Serial	128	1093	303	1054
2015	[19]	Serial + bool SBOX	128	1243	303	1054
2016	[20]	16-bit	80/128	576[b]	132	528[b]
2016	This work.	16-bit + 80-bit keys	80	989	133	926
2016	This work.	16-bit + 128-bit keys	128	1257	136	1194

[a] Results presented in [24].
[b] The key generation module is not considered

TABLE II
EXPERIMENTAL CONFIGURATIONS

Year	Ref.	Architecture	Key size	Conf.
2012	[18]	Iterative	128	C1
2012	[18]	Serial	128	C2
2016	[20]	16-bit	80/128	C3
2016	This work.	16-bit + 80-bit keys	80	C4
2016	This work.	16-bit + 128-bit keys	128	C5

Each one of the evaluated architectures included I/O mechanisms 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 PRESENT 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 datapath proposed in Section III with a key schedule to process 128-bit keys.

B. Environment

The five PRESENT architectures to be evaluated were synthesized 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. Newer FPGAs such as the 7 series were not considered since according to the manufacturer [31] the Configurable Logic Blocks in these FPGAs are similar to those used in FPGAs of the 6 series such as the Spartan 6, which might lead to similar implementation results in regards to resource usage.

C. Synthesis Criteria

The synthesis process for all the designs was configured with `Area` as *Optimization Goal* and `High` as *Optimization Effort*. To provide results independent of the platform and achieve a fair comparison it was decided not to use the FPGA embedded RAM/ROM resources by disabling the corresponding flags in the HDL options of the implementation process. If the RAM/ROM resources in the FPGA were used, the configuration C3 is the only one suitable to exploit this feature without hardware description modifications.

D. Metrics

The metrics for evaluating the PRESENT hardware designs in FPGA are area, performance and energy consumption. *Slices* are used as area units, but the results in LUT and Flip-Flop usage are also presented and discussed. Performance is expressed in both the *latency* exhibited and the throughput achieved by each of the hardware architectures under evaluation. In terms of energy, the calculation of the total power dissipated by the architectures and the energy required to process a plaintext block are provided.

Two derived metrics are used to estimate the efficiency of the architectures under evaluation: the throughput-per-slice which is a relation of the implementation size and the performance, and the energy-per-bit, which is the estimate of energy spent to process the state.

The resource usage for each implementation is obtained from the implementation reports generated by the ISE Design Tools 14.3. For each implementation, the number of flip flops (FF), look-up tables (LUT), and slices (SLC) is provided. It is important to note that two of the FPGAs utilized have LUT-4 technology and the other two have LUT-6 technology, so a direct comparison between these is not appropriate.

The maximum throughput (Thr) of an implementation is a function of the maximum operational frequency (FMAX), the latency cycles required to process a block (LAT), and the block size (BSIZE). It is calculated with Equation 1.

$$\text{Thr} = \frac{\text{FMAX} \times \text{BSIZE}}{\text{LAT}} \tag{1}$$

The throughput-per-slice (Thr/SLC) is computed using Equation 2.

$$\text{thr/SLC} = \frac{\text{Thr}}{\text{SLC}} \tag{2}$$

Low-resource implementations usually do not use the maximum possible frequency, but limit it to a lower frequency, as is the case in RFID applications where a frequency of 13.56MHz is used [32]. By having a common frequency to evaluate all the architectures implemented in this paper, it is possible to achieve a fair comparison in terms of throughput-per-slice. In this case the throughput calculated at a frequency of 13.56MHz is denoted Thr* and calculated using Equation 3.

$$\text{Thr}^* = \frac{13.56\text{MHz} \times \text{BSIZE}}{\text{LAT}} \tag{3}$$

The tool Xilinx XPower Analyzer 14.3 was utilized to estimate the static, dynamic, and total power (POW) of each implementation. Each estimation was made using the design file (NCD) generated by the ISE Design Tools, the physical constraints file (PCF) derived from the specific FPGA board for each implementation, and the simulation activity file (SAIF) generated with the Post-Place & Route Simulation Model simulated with ISim 14.3 for an operation frequency of 13.56MHz. By using these files as input, the analysis tool reported `high` overall confidence levels for all the estimations.

The energy (ENE) spent by an implementation to process a single block, considering an operational frequency of 13.56 MHz, is calculated according to Equation 4.

$$\text{ENE} = \frac{\text{POW} \times \text{LAT}}{13.56 \text{ MHz}} \tag{4}$$

Then Equation 5 is used to estimate the energy-per-bit.

$$\text{ENE/bit} = \frac{\text{ENE}}{\text{BSIZE}} \tag{5}$$

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.

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 yielding different results depending on the implementation strategies.

Regarding performance, the throughput was calculated for both, the maximum operational frequency of each architecture, and for a constant operational frequency of 13.56 MHz across all the platforms. The former can be of use to systems that

TABLE III
RESOURCE USAGE AND PERFORMANCE RESULTS FOR THE FIVE ARCHITECTURES UNDER EVALUATION

Work	Design	STATE (bit)	KEY (bit)	FF	LUT	SLC	FMAX (MHz)	LAT (cycles)	Thr (Mbps)	Thr* (Mbps)	Thr*/SLC (Kbps/Slice)
xc6slx16-3csg324											
[18]	C1	64	128	200	202	58	160.21	**55**	**186.42**	**15.78**	**272.05**
[18]	C2	64	128	203	157	**44**	164.23	303	34.69	2.86	65.09
[20]	C3	64	128	**73**	**147**	48	206.40	132	100.07	6.57	136.97
This work.	C4	64	80	153	170	48	**257.40**	133	123.86	6.53	135.94
This work.	C5	64	128	201	220	61	210.66	136	99.13	6.38	104.61
xc3s200-5ft256											
[18]	C1	64	128	200	381	191	179.95	**55**	**209.40**	**15.78**	**82.61**
[18]	C2	64	128	203	258	131	177.34	303	37.46	2.86	21.86
[20]	C3	64	128	**73**	280	153	120.71	132	58.53	6.57	42.97
This work.	C4	64	80	153	**215**	**124**	**213.81**	133	102.89	6.53	52.62
This work.	C5	64	128	201	264	151	194.63	136	91.59	6.38	42.26
xc5vlx50t-3ff1136											
[18]	C1	64	128	200	283	88	271.67	**55**	**316.12**	**15.78**	**179.31**
[18]	C2	64	128	203	237	72	289.69	303	61.19	2.86	39.78
[20]	C3	64	128	**73**	**182**	75	321.96	132	156.10	6.57	87.66
This work.	C4	64	80	153	190	**67**	**542.30**	133	260.96	6.53	97.39
This work.	C5	64	128	201	239	73	431.78	136	203.19	6.38	87.41
xc4vlx25-12ff668											
[18]	C1	64	128	200	382	192	284.33	**55**	**330.86**	**15.78**	**82.18**
[18]	C2	64	128	203	258	131	288.52	303	60.94	2.86	21.86
[20]	C3	64	128	**73**	279	151	223.51	132	108.37	6.57	43.54
This work.	C4	64	80	153	**215**	**124**	**375.66**	133	180.77	6.53	52.62
This work	C5	64	128	201	265	152	364.56	136	171.56	6.38	41.98

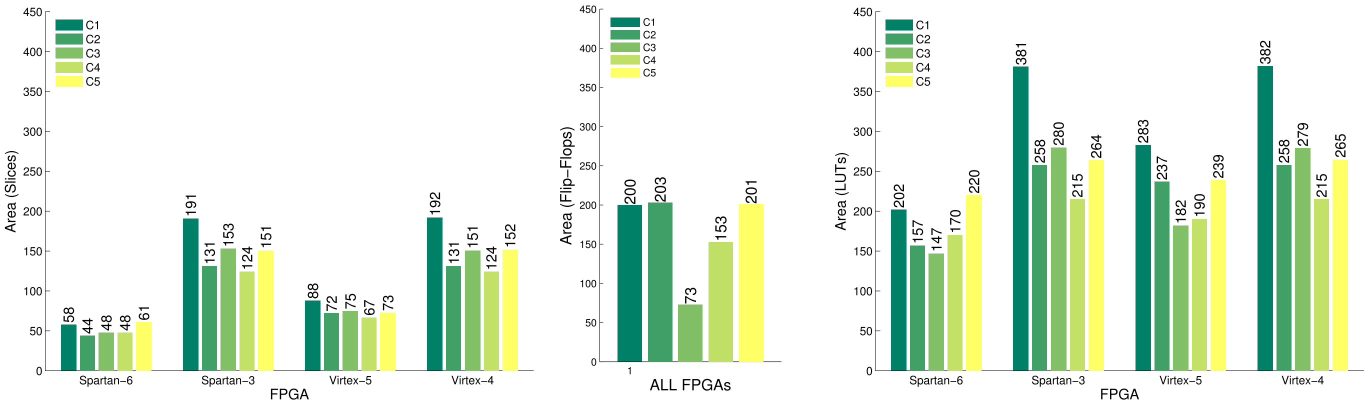
* Using a frequency of 13.56 MHz.

TABLE IV
POWER AND ENERGY CONSUMPTION RESULTS FOR THE FIVE ARCHITECTURES UNDER EVALUATION

Work	Design	STATE (bit)	KEY (bit)	LAT (cycles)	Static POW (mW)	Dynamic POW (mW)	Total POW (mW)	ENE* (μJ)	ENE*/bit (nJ/bit)
xc6slx16-3csg324									
[18]	C1	64	128	**55**	**19.91**	**1.40**	**21.31**	**0.086**	**1.351**
[18]	C2	64	128	303	**19.91**	2.02	21.93	0.490	7.657
[20]	C3	64	128	132	**19.91**	1.69	21.6	0.210	3.285
This work.	C4	64	80	133	**19.91**	1.70	21.61	0.212	3.312
This work.	C5	64	128	136	**19.91**	1.85	21.76	0.218	3.410
xc3s200-5ft256									
[18]	C1	64	128	**55**	**40.99**	**0.98**	**41.97**	**0.170**	**2.660**
[18]	C2	64	128	303	**40.99**	1.73	42.72	0.955	14.915
[20]	C3	64	128	132	**40.99**	1.38	42.37	0.412	6.445
This work.	C4	64	80	133	**40.99**	1.09	42.08	0.413	6.449
This work.	C5	64	128	136	**40.99**	1.37	42.36	0.425	6.638
xc5vlx50t-3ff1136									
[18]	C1	64	128	**55**	**560.04**	3.47	563.51	**2.286**	**35.713**
[18]	C2	64	128	303	560.06	4.74	564.8	12.621	197.196
[20]	C3	64	128	132	**560.04**	3.53	563.57	5.486	85.720
This work.	C4	64	80	133	**560.04**	2.71	562.75	5.520	86.244
This work.	C5	64	128	136	**560.04**	**2.63**	**562.67**	5.643	88.177
xc4vlx25-12ff668									
[18]	C1	64	128	**55**	333.44	15.44	348.88	**1.415**	**22.111**
[18]	C2	64	128	303	333.46	15.97	349.43	7.808	122.001
[20]	C3	64	128	132	**232.90**	**9.48**	**242.38**	2.359	36.866
This work.	C4	64	80	133	232.97	12.81	245.78	2.411	37.667
This work.	C5	64	128	136	233.01	15.01	248.02	2.488	38.867

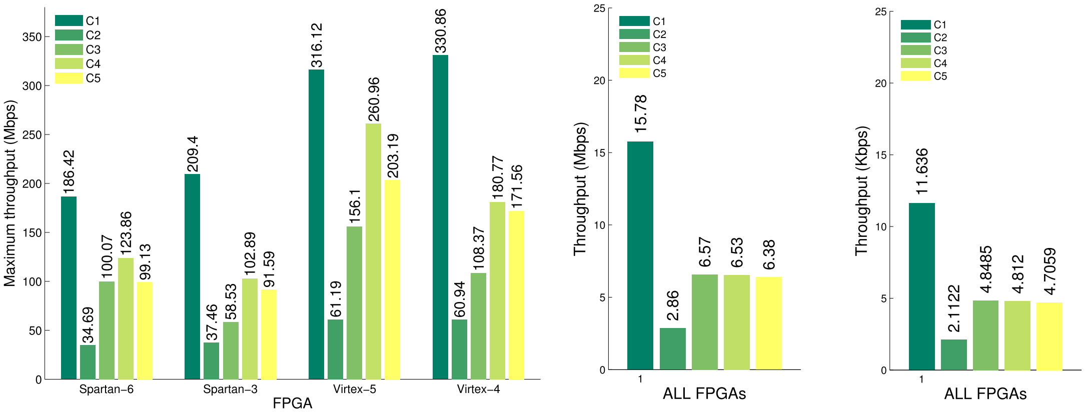
* Using a frequency of 13.56 MHz.

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.



(a) (b) (c)

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

on the underlying FPGA platform. The performance comparison for the different implementations is shown in Fig. 10 using the maximum frequency of the implementation, the frequency 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 implementation of PRESENT 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 operational 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 significance of an order of magnitude. A graphic comparison for

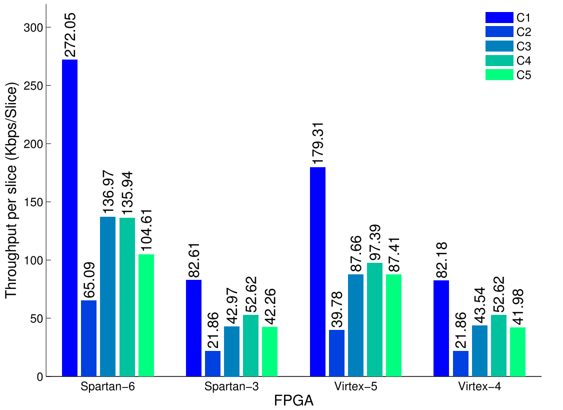
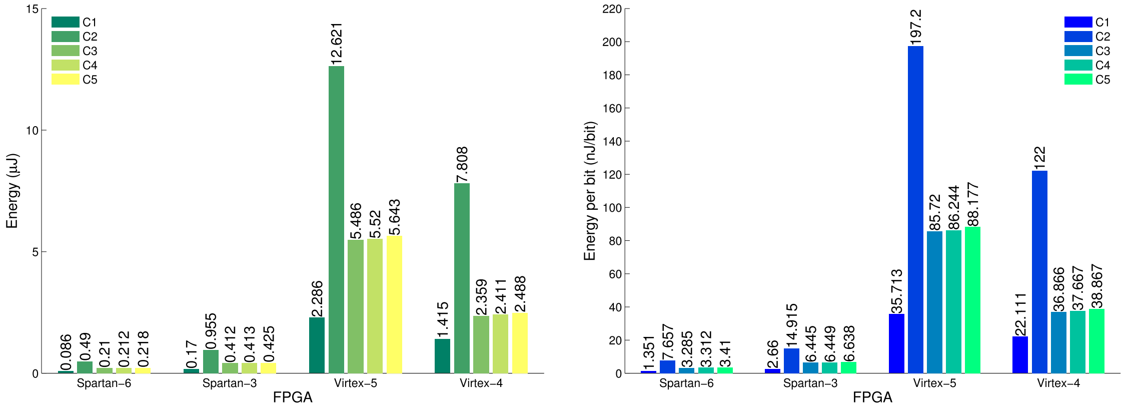
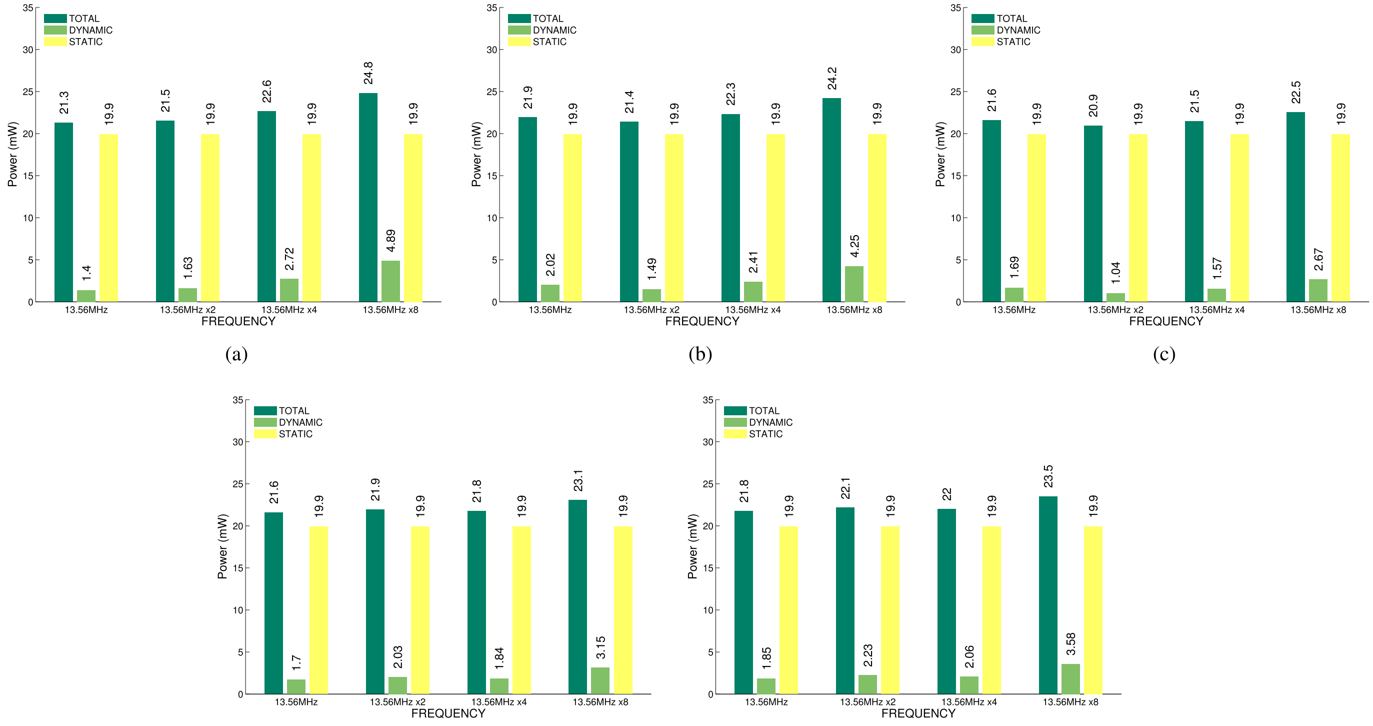


Fig. 11. Throughput per slice for the different configurations in the different FPGAs utilized.



(a) (b)

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



(a) (b) (c)

(d) (e)

Fig. 13. Power as a function of the operational frequency for the five PRESENT 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. Figure 13 illustrates the power estimation for ×1 ×2, ×4, and ×8 multipliers of 13.56 MHz for the different configurations. The Spartan-6 FPGA was selected to run this experiment since it obtained the best results in regards to power efficiency.

VI. Discussion of the Proposed Architectures

This section presents a brief evaluation of the two proposed PRESENT architectures in relation to the designs retrieved from the literature. Area, energy, and performance are compared and averaged when possible, to determine if the achieved results are platform-independent and the impact these results have.

A. C4 – 16-Bit Architecture With Key Size of 80-Bit

1) Area: This architecture (C4) has a reduction in the resource usage for both LUT-4 FPGAs of 35% if compared to the iterative architecture (C1), and uses 5% less resources if compared to the serial architecture (C2). For the Spartan-6 FPGA, C4's slice count is smaller than that of C1 by 17%, but its not smaller than C2. Using the Virtex-5 FPGA this design has smaller slice count than both C1 and C2 by 24% and 7% respectively.

2) Performance: The throughput for C4 is reduced by 59% with regards to the throughput of C1, and the throughput of the C2 is reduced by 56% compared to C4.

3) Energy: Statistically C4 requires 2.25 times the energy of C1 for all the FPGA boards utilized. This in turn uses in average 56% of the energy spent by C2. The consumption profile for C4 is similar to those of the other 16-bit architectures (C3, C5).

C4 features reduced costs in resource usage and energy consumption but the encryption key is smaller. For three of the FPGAs under study C4 achieves the smallest slice count with reasonably high performance. From the two models proposed in this paper, C4 features the highest efficiency in throughput per slice and the lowest energy consumption. Applications that can use a smaller security level and require both small size and low energy consumption can benefit of this design.

B. C5 – 16-Bit Architecture With With Key Size of 128-Bit

1) Area: This architecture (C5) achieves smaller resource usage than the iterative design (C1) by 21% when implemented in LUT-4 FPGAs. The serial architecture (C2) outperforms C5's slice count by 15%. The implementation results for C5 are not favorable for the Spartan-6 FPGA which features LUT-6 technology. In contrast, if the Virtex-5 FPGA is used as implementation platform, the C5's slice count is on par to that of C2 and is smaller than that of C1 by 17%.

2) Performance: C5 has a throughput reduction of 60% compared to C1, while C2 has a reduction of 55% in relation to this design.

3) Energy: C5 consumes in the average 2.31 times the energy of C1, which represents statistically 58% of the energy required by C2.

C5 is an alternative to the proposals in the state of the art (C1, C2) which can be compared in equal grounds due to having a common security level. Compared to C1, it has a smaller slice count for 3 out of 4 FPGA boards analyzed, considering the case of the Spartan-6 FPGA which can be considered as an outlier. Compared to C2, this design features higher performance and efficiency in both throughput per slice and energy per bit. This alternative is attractive for applications with area and performance constraints, that also require high security.

VII. Conclusions

This paper presented a comparison of hardware architectures for the Present 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 Present 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 measurements 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.

Acknowledgment

The authors would like to thank the authors of [18] for providing their source codes and agreeing to make them publicly available.

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. Knežević, 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 processor," *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 evaluation of an FPGA-based SoC platform with AES and PRESENT coprocessors," in *Embedded Computer Systems: Architectures, Modeling, 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.

[16] P. Yalla and J.-P. Kaps, "Lightweight cryptography for FPGAs," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs (ReConFig)*, Dec. 2009, pp. 225–230.

[17] E. B. Kavun and T. Yalcin, "RAM-based ultra-lightweight FPGA implementation of PRESENT," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs (ReConFig)*, Nov. 2011, pp. 280–285.

[18] N. Hanley and M. O'Neill, "Hardware comparison of the ISO/IEC 29192-2 block ciphers," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Aug. 2012, pp. 57–62.

[19] 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 *Proc. 6th Asia Symp. Quality Electron. Design*, Aug. 2015, pp. 144–148.

[20] C. A. Lara-Nino, M. Morales-Sandoval, and A. Diaz-Perez, "Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher," in *Proc. Euromicro Conf. Digit. Syst. Design*, Aug./Sep. 2016, pp. 646–650.

[21] T. Addabbo, A. Fort, M. Di Marco, L. Pancioni, and V. Vignoli, "Physically unclonable functions derived from cellular neural networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 12, pp. 3205–3214, Dec. 2013.

[22] Y. Cao, L. Zhang, S. S. Zalivaka, C.-H. Chang, and S. Chen, "CMOS image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 11, pp. 2629–2640, Nov. 2015.

[23] M. Wan, Z. He, S. Han, K. Dai, and X. Zou, "An invasive-attack-resistant PUF based on switched-capacitor circuit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 8, pp. 2024–2034, Aug. 2015.

[24] A. Bogdanov *et al.*, "PRESENT: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 4727. Berlin, Germany: Springer, 2007, pp. 450–466.

[25] H. Kaeslin, *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication*, 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[26] A. Moradi, A. Barenghi, T. Kasper, and C. Paar, "On the vulnerability of FPGA bitstream encryption against power analysis attacks: Extracting keys from Xilinx Virtex-II FPGAs," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2011, pp. 111–124. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046722

[27] M. Nassar, Y. Souissi, S. Guilley, and J.-L. Danger, "'Rank correction': A new side-channel approach for secret key recovery," in *Security Aspects in Information Technology*. Berlin, Germany: Springer, 2011, pp. 128–143. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24586-2_12

[28] S. Gao, H. Chen, W. Wu, L. Fan, W. Cao, and X. Ma, "My traces learn what you did in the dark: Recovering secret signals without key guesses," in *Topics in Cryptology*. Cham, Switzerland: Springer, 2017, pp. 363–378. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-52153-4_21

[29] J. D. Golic, "Techniques for random masking in hardware," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 2, pp. 291–300, Feb. 2007.

[30] W. Yu and S. Köse, "A voltage regulator-assisted lightweight AES implementation against DPA attacks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 8, pp. 1152–1163, Aug. 2016.

[31] (Sep. 2016). *7 Series FPGAs Configurable Logic Block*. [Online]. Available: www.xilinx.com

[32] *Identification Cards—Contactless Integrated Circuit Cards—Proximity Cards—Part 2: Radio Frequency Power and Signal Interface*, document ISO/IEC 14443-2, Aug. 2010.

Carlos Andres Lara-Nino is a Ph.D. student at the Center for Research and Advanced Studies of the National Polytechnic Institute of Mexico. He is a Mechatronics Engineer with a masters degree in Computer Science. His main area of interest is the design of Digital Systems. Currently his research is focused on the implementation of highly optimized cryptographic algorithms for constrained environments.

Arturo Diaz-Perez is currently a full time Professor and the Head of Center for Research and Advanced Studies Tamaulipas campus, Mexico. His research areas are information security, hardware level description of dynamic programming algorithms, high-level synthesis of digital systems, and recurrence equations for parallel programming. He co-authored the book *Cryptographic Algorithms on Reconfigurable Hardware*.

Miguel Morales-Sandoval received the Ph.D. degree from the National Institute for Astrophysics, Optics, and Electronics, Mexico, in 2008. He is currently a Computer Science Researcher with special interests on data security, cryptography, and embedded systems. His research interests focus on the development of hardware/software security schemes for networked embedded systems and for the cloud.