2016 Euromicro Conference on Digital System Design

Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher

Lara-Nino, Carlos Andres; Morales-Sandoval, Miguel and Diaz-Perez, Arturo
Centro de Investigacion y Estudios Avanzados del IPN
Unidad Tamaulipas
Ciudad Victoria, Tamaulipas. Mexico.
{clara, mmorales, adiaz}@tamps.cinvestav.mx

***Abstract*—This paper presents a novel FPGA-based design for the lightweight block cipher PRESENT and its implementation results. The proposed design allows to study area-performance trade-offs and thus constructing smaller or faster implementations. When optimized by area, the proposed design exhibits smaller latency and fewer FPGA resources than representative related works in the literature.**

I. INTRODUCTION

The conceived ecosystem of the Internet of Things (IoT) consists of smart objects, intelligent interconnected devices that communicate among themselves thus enabling applications in Intelligent Transportation Systems (ITS), business/process management, assisted living and e-health. In 2008, the National Intelligence Council of the United States (US) presented a report on Disruptive Civil Technologies [1] mentioning IoT as one of the six technologies with potential impacts on US interests out to 2025. In that document, the authors also reflect that everyday objects could become information-security risks, and the IoT could distribute those risks more widely than the conventional Internet.

According to [2], the greatly reduced quantitative bounds on acceptable area and energy metrics will require qualitative breakthrough in design and optimization techniques. The same authors have claimed that hardware-based security is ideally well suited to answer IoT security requirements. In [3], Symmetric Key Cryptography is called to play a major role in the security area of the IoT, designed and implemented efficiently enough so as to comply with the scarce resources of the IoT devices. As typical devices in IoT (i.e., sensors nodes in a WSN) are equipped with low end micro-controllers with small word sizes and slow oscillators, software-friendly lightweight primitives are desired. A such solution, as mentioned in [4], is a balancing act between several aspects as cryptographic strength, area implementation costs, execution speed and power consumption.

This work focuses on the design of a hardware architecture for the lightweight symmetric key cipher PRESENT [5]. As the targeted applications are in the IoT context, the main goal is to obtain a design with a small area implementation.

A. Related Work

Several authors have elaborated on the need to create area-optimized security architectures for constrained environments. Some follow as strategy the optimization of standardized general purpose ciphers as the Advanced Encryption Standard (AES) [6]. Others try to optimize lightweight algorithms like PRESENT [5]. These proposals are targeted to different hardware architectures, mainly Application-Specific Integrated Circuits (ASIC) and Field Programmable Gate Arrays (FPGA). This work focuses on the latter because it is a platform commonly used for evaluating cryptographic algorithms, taking advantage of rapid prototyping to explore area-performance trade-offs of hardware designs under study. The same FPGA design can be later used to produce a more efficient product delivered as an Application Specific Integrated Circuit (ASIC), with possibly less area and power consumption.

In 2009, Sbeiti *et al.* [7] presented an implementation of PRESENT evaluating *efficiency* as a compromise between performance and area. Their results demonstrated that PRESENT was not only a good cipher for low-cost hardware applications but also for high-performance applications. That work was implemented on a Spartan-III FPGA achieving a cost of 176 slices for LUT-4 technology. Also in 2009, Yalla and Kaps [8] proposed novel optimization techniques for implementing block ciphers in the Spartan3 FPGA, using PRESENT as case study. They devised a 16-bit datapath architecture by reducing the width of the substitution layer and then used semi shift-registers to handle the permutation layer. That design had a cost of 117 slices in a LUT-4 FPGA. In 2011, Bilge and Yalcin [9] developed two architectures for the PRESENT cipher exploiting the usage of the embedded block RAMs in Spartan3 FPGAs. That work also showcased interesting access patterns for reducing the permutation layer of the cipher with an estimated implementation size of 183 slices. After the standardization of PRESENT in 2012 [10], Hanley and O'Neill presented a hardware comparison of the standardized ciphers featuring as implementation platforms the Virtex-V and Virtex-II FPGAs [11]. As a result, the authors delivered two of the most compact FPGA architectures for PRESENT and the one with the highest throughput per slice reported to

978-1-5090-2817-7/16 $31.00 © 2016 IEEE
DOI 10.1109/DSD.2016.46

that date. Their results recorded a resource usage of 87 and 70 slices for their LUT-6 implementations. In 2015, Tay *et al.* created an 8-bit hardware architecture for PRESENT, intended to be compact for IoT applications implemented on a Virtex-V FPGA [12]. The main feature in that work is the usage of optimized substitution boxes based on boolean logic. Their results reported the smallest implementation of PRESENT in FPGA registered to date with a slice count of 69 for LUT-6 technology.

B. Contributions

The main contribution in this work is a novel architecture for the PRESENT cipher that is well suited for hardware implementations. A novel datapath for the entire algorithm is provided, with unified width of 16-bit and the introduction of four 16-bit full registers. Also, part of the final permutation is not hardwired but realized by memory-addressing operations during the processing of the 64-bit input data. These features in the datapath lead to a hardware realization with reduced latency and area resources.

The rest of the paper is organized as follows. Section II briefly reviews the PRESENT algorithm. Section III elaborates on the design of the novel hardware architecture for PRESENT. Section IV describes the applied methodology for the experimental work. Section V discusses the obtained implementation results and provides comparisons against related works. Finally, Section VI concludes this work and gives an outline of possible future work.

II. THE PRESENT CIPHER

PRESENT is one of the symmetric ultra-lightweight blocks cipher standardized in 2012 by ISO/IEC as "block ciphers suitable for lightweight cryptography, which are tailored for implementation in constrained environments" [10]. 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. The input key is considered to be a set which contains a roundkey for each one of the 31 total rounds. The cipher uses three basic operations to produce confusion and diffusion over the input data:

– `AddRoundKey`: adds the state to a 64-bit word from the roundkey using finite field arithmetic.
– `sBoxLayer`: effectuates a 4-bit to 4-bit substitution using sixteen Substitution Box (SBOX).
– `pLayer`: applies bit level shifts over the state.

Figure 1 shows the diagram for the encryption process previously described. For a detailed specification of PRESENT the reader could refer to [5].

III. A NOVEL HARDWARE ARCHITECTURE FOR PRESENT

The basic architecture of PRESENT follows closely its algorithm specification. A 64-bit datapath enables the

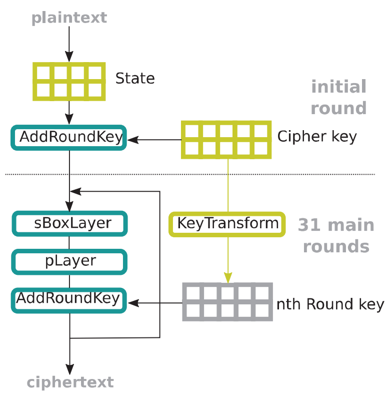


Fig. 1. PRESENT's encryption procedure.

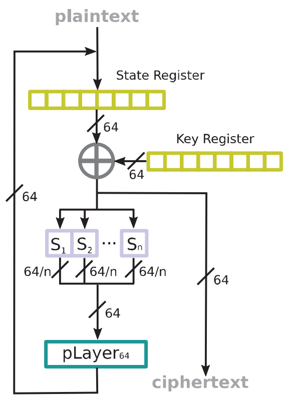


Fig. 2. The datapath of an area-optimized version of PRESENT-80 in [5].

execution of an entire round in a single cycle, requiring in turn an extensive number of 4-bit substitutions (SBOX) and a 64-bit permutation. In this case, the low area is achieved solely based on the algorithm specification and the latency equals the number of rounds. An area optimization strategy to this architecture is to reduce the number of substitution boxes creating a direct trade-off between utilized resources and latency. That approach was firstly outlined in [5] and the corresponding hardware architecture is the one shown in Figure 2.

However, because of the datapath width variation proposed in [5], it requires additional logic for routing and control which subsequently induces area overhead that can reduce efficiency.

In this work, the aim is to reduce the datapath width as a whole. This implies to reduce the width of both, the substitution layer (sBoxLayer) and the permutation layer (pLayer). The reduction of the substitution layer follows the suggested optimization in [5], adjusted to any width divisible by four, that is, the total input bits in a PRESENT's SBOX. Width reduction of the permutation layer is achieved following a similar strategy to that first presented in [8] and detailed later in [9], which consists in exploiting the regularity in the 64-bit permutation. By using an access pattern it is possible to shrink down the width of the permutation from 64-bit to 16-bit, as illustrated in Figure 3. With that reduction in the permutation layer, the substitution layer can use a data bus of 16-bit too, thus requiring only four substitution boxes.

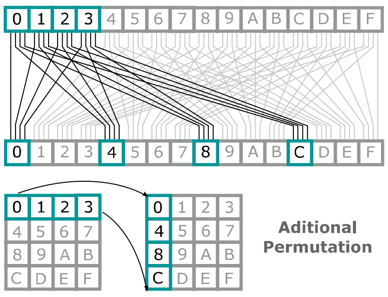

0 1 2 3 4 5 6 7 8 9 A B C D E F
0 1 2 3 4 5 6 7 8 9 A B C D E F
Aditional Permutation

Fig. 3. PRESENT permutation scheme in [9].

A unified 16-bit datapath reduces the additional logic demanded by a variable datapath width, as in [5]. 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. Moreover, having a smaller datapath width also reduces the number of resources consumed in both the substitution layer, requiring less substitution boxes, and the permutation later, requiring less wires. However, the reduction strategy for the permutation layer is not straightforward. The new datapath processes the data in blocks of 16-bit words. Therefore, if these data are stored in a register using the natural order of processing, an additional permutation is required. This was solved in [8] by using two 64-bit semi shift-registers to perform all the permutations, as it is shown in Figure 4.

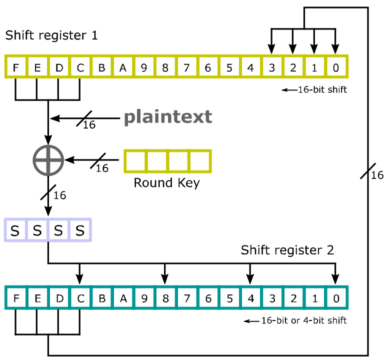


Fig. 4. 16-bit datapath of PRESENT in [8].

This work proposes to use four 16-bit full registers instead of the two 64-bit semi shift-registers used in [8]. Instead of hardwiring the additional permutation, we use memory-addressing operations. Experimental results showed that there is no significant increment in area due to the use of full registers. Also, this change allows to execute the initial permutation sequentially and the additional permutations in parallel. As a consequence, the latency is reduced.

Figure 5a illustrates the new proposed datapath for PRESENT with the four full registers. That figure allows to distinguish four processing threads that concurrently do writing operations over the four 16-bit full registers. Figure 5b shows a hardware-oriented implementation optimized by area, which is the sequential version resulting from removing three of the processing lines shown in Figure 5a. This architecture is able to process, in case of the datapath in Figure 5b, 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 4 cycles per round. Since 4 cycles are used to load the input data and 4 cycles are invested to generate the output, 132 cycles are required to encrypt a 64-bit word.

This work considers that the whole set of keys is pre-calculated and stored in a secure memory. Having the keys stored beforehand enables the partitioning of the memory space, switching from a 32x64 memory block to a 128x16 memory. This memory space partition favours the implementation of the architecture on FPGAs with LUT-6 technology. Although other works [11] have computed the PRESENT's set of keys on the fly, it was experimentally demonstrated that the cost in terms of hardware of that approach is almost the same than storing the keys in memory.

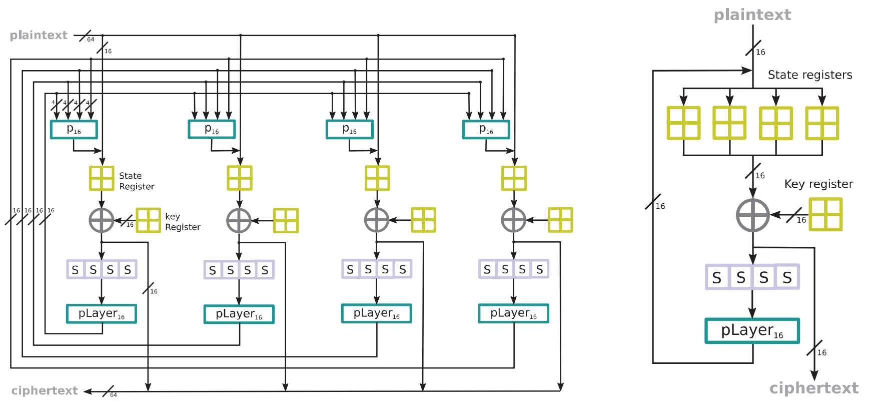
IV. METHODS

A. Environment

The architectures for PRESENT discussed in the previous section were modeled using VHDL and synthesized for Xilinx FPGAs using the Design Suite 14.7. A family of low cost development boards were used as computing platforms: a) the NEXYS2 board from Digilent, equipped with an FPGA Xilinx Spartan3E XC3S500-5FG320 was used to characterize the architectures when implemented in FPGAs with LUT-4 technology; b) the NEXYS3 board from Digilent, equipped with an FPGA Xilinx Spartan6 XC6SLX16-3CSG324 was used to characterize the architectures implemented in FPGAs with technology LUT-6.

B. Synthesis criteria

The synthesis process for all of the implementations was configured with *Area* as *Optimization Goal* and *High* as *Optimization Effort*. To provide results independent of the platform 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. The synthesis, map and place and route processes were verified multiple times in different computers and no variation in the results was found.



(a) Parallel version with four processing lines for high performance implementations.

(b) Sequential pipelined version for low-area implementations.

Fig. 5. Novel hardware architectures for the PRESENT cipher.

C. Configurations

Three architectures of PRESENT were constructed and evaluated:

- **PRE**: A basic architecture that closely follows the specifications of the algorithm presented in [5]. This implementation was made to serve as a reference for comparing the optimization strategies proposed in this work.
- **PRE_O1**: An area-optimized architecture that considers only the reduction of the substitution layer using two 4-bit substitution boxes, as proposed in [5].
- **PRE_O2**: An area-optimized architecture that follows all the premises presented in this work, to construct a low-area architecture, using the datapath shown in Figure 5b.

Each of the three PRESENT architectures includes I/O mechanisms that allows to use the hardware module as an independent core or in an integrated system. The implementations **PRE** and **PRE_O1** were created only to have a fair comparison with the novel design proposed (**PRE_O2**) under the same implementation conditions: FPGA devices, programming style, synthesis process and tools.

D. Metrics

The main metric used to evaluate the hardware designs is the area consumption, expressed as amount of slices, flip-flops, and LUTs consumed by the designs. Performance is evaluated and expressed in terms of latency and throughput. The throughput is computed using Equation 1.

$$throughput = \frac{frequency \times bits}{latency} \tag{1}$$

As presented in [11], low-resource implementations usually do not use the maximum possible frequency, but limit it to 13.56MHz as pointed out in [13].

V. Results

The implementation results in the FPGA Spartan3E XC3S500-5FG320 (LUT-4) are presented in Table I whereas Table II presents the implementation results in the FPGA Spartan6 XC6SLX16-3CSG324 (LUT-6). These tables also include the implementation results of most of the representative related works targeting low-area designs for PRESENT.

PRE was designed closely following the description in [5]. By using a 64-bit register and I/O ports it was possible to achieve a latency of 33 cycles, which is the minimum possible. The fact that this design has no changes in the datapath width, as in **PRE_O1**, and no additional operations are required, as in **PRE_O2**, makes this design simpler. Due to these factors this implementation turned out to be the one with the highest throughput per slice registered to date for the PRESENT algorithm, to the best of our knowledge. This result is not the product of any optimization technique, but the result of the lightweight properties of the algorithm itself.

PRE_O1 follows a common optimization technique based on the reduction of the number of substitution boxes. However, only 4 LUTs are required for a single substitution box in PRESENT. This optimization technique produces an overhead in the control logic and requires additional multiplexers to pass data through the reduced substitution layer.

The architecture proposed in this paper, **PRE_O2**, demonstrates that it can be attractive for specific scenarios. This design features a reduced-width datapath, which implies that the number of I/O ports is also reduced. The optimizations applied aimed to have a reduction in the number of Flip-Flops required, however this reflects only slightly in the Slice count. In terms of latency the design achieves better results than other State of the Art area-optimized implementations [11], [12].

TABLE I
AREA-OPTIMIZED BLOCK CIPHERS IMPLEMENTED USING LUT-4 TECHNOLOGY.

	Year	Reference	Algorithm	Platform	Datapath	Flip-Flops	LUTs	Slices	Frequency (MHz)	Latency	Throughput (Mbps)	Throughput[3] (Mbps)	Throughput[3]/Slice (Kbps/Slice)
1	2009	[7]	PRE	Spartan-III XC3S400 FG456-5	64	152	253	176	258.00	32	516.00	27.12	154.09
2	2009	[8]	PRE	Spartan-III XC3S50-5	16	114	159	117	113.89	256	28.47	3.39	28.97
3	2011	[9]	PRE	Spartan-III XC3S50	16	-	-	183[1]	129.77	-	-	-	-
4	2012	[11]	PRE Serial	Virtex-II XC2VP7 FG456-5	64	203	258	131[2]	114.17	295	24.77	2.94	22.46
5	2012	[11]	PRE Iterative	Virtex-II XC2VP7 FG456-5	64	200	303	155[2]	155.76	47	212.10	18.46	119.13
6	2016	This work.	PRE	Spartan3E-500 FG320-5	64	136	476	265	192.23	33	372.82	26.30	99.24
7	2016	This work.	PRE_O1	Spartan3E-500 FG320-5	64	145	524	271	141.26	250	36.16	3.47	12.81
8	2016	This work.	PRE_O2	Spartan3E-500 FG320-5	16	98	478	256	132.19	132	64.09	6.57	25.68

1 Estimated.
2 Without considering I/O logic.
3 Using a frequency at 13.56MHz.

TABLE II
AREA-OPTIMIZED BLOCK CIPHERS IMPLEMENTED USING LUT-6 TECHNOLOGY.

	Year	Reference	Algorithm	Platform	Datapath	Flip-Flops	LUTs	Slices	Frequency (MHz)	Latency	Throughput (Mbps)	Throughput[2] (Mbps)	Throughput[2]/Slice (Kbps/Slice)
1	2012	[11]	PRE Serial	Virtex-5 XC5VLX50 FF324-1	64	203	237	70[1]	245.76	295	53.31	2.96	42.03
2	2012	[11]	PRE Iterative	Virtex-5 XC5VLX50 FF324-1	64	200	285	87[1]	250.89	47	341.63	18.46	212.24
3	2015	[12]	PRE	Virtex-5 XC5VLX50	64	201	222	62	236.57	295	51.32	2.94	47.75
4	2016	This work.	PRE	Spartan-6 XC6SLX16 CSG324C	64	136	229	74	221.63	33	429.83	26.30	355.38
5	2016	This work.	PRE_O1	Spartan-6 XC6SLX16 CSG324C	64	137	308	121	160.13	250	40.99	3.47	28.69
6	2016	This work.	PRE_O2	Spartan-6 XC6SLX16 CSG324C	16	89	226	69	172.92	132	83.84	6.57	95.28

1 Without considering I/O logic.
2 Using a frequency at 13.56MHz.

VI. CONCLUSIONS AND FUTURE WORK

This paper presented a novel hardware design for the PRESENT cipher aiming at obtaining a low-cost design in terms of area. The proposed architecture features a reduced-width datapath well suited for platforms with reduced number of ports.

The experimental results demonstrate that the proposed design (**PRE_02**) outperforms the optimization proposed by [5] and still performs on par with State of the Art alternatives in terms of area and performance. Using a LUT-6 FPGA as implementation platform, the proposed architecture achieves a Flip-Flop count 56% smaller than the architecture with the smallest slice count [12], and the latency is reduced by 55%.

The **PRE** version implemented in the Spartan6 (LUT-6 FPGA) resulted to be the most efficient implementation of PRESENT registered to date. This implementation features a minimal latency and reduced area which are critical for the efficiency when the operation frequency follows standardized specifications.

Further studies are required to verify the nature of the relationship between the number of latency cycles and the energy consumed by the proposed designs. There is also room for improvement in the use of implementation technologies as the HLS falls short when it comes to small designs.

REFERENCES

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

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

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

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

[5] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds. Springer Berlin Heidelberg, 2007, vol. 4727, pp. 450–466. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74735-2_31

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

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

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

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

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

[11] N. Hanley and M. ONeill, "Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers," in *2012 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Aug 2012, pp. 57–62.

[12] J. J. Tay, M. L. D. Wong, M. M. Wong, C. Zhang, and I. Hijazin, "Compact FPGA implementation of PRESENT with Boolean S-Box," in *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*, Aug 2015, pp. 144–148.

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