International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, Number 12 (2020), pp. 5213-5218

Co-designed implementation of the PRESENT cryptographic algorithm for an ARM-based System on Chip

Edwar Jacinto Gómez [1], Fredy H. Martínez S.[2] and Fernando Martínez Santa [3]
Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá D.C, Colombia.

[1]*ORCID:* 0000-0003-4038-8137 [2]*ORCID*: 0000-0002-7258-3909
[3]*ORCID:* 0000-0003-2895-3084

Abstract

The complexity of current electronic systems allows to interconnect devices with different architectures in a System on Chip - SOC, performing data acquisition and processing tasks, as well as transmission and reception of information in IoT and/or IIoT intelligent network applications. To guarantee the security of the information in these systems it is possible to incorporate cryptographic processes in those applications, although these processes suppose a high expenditure of system resources for their execution, increasing the processing time in the creation of the same ones therefore it makes difficult its implementation in safe low-cost applications, the reason why the use of minimalist computer security algorithms like the ciphers by light blocks have arisen. This article presents the implementation of the lightweight block cipher algorithm called "PRESENT" by hardware-software coding in a low-cost device such as the PSoC® 5LP to reduce the time and number of resources used in the data encryption process, as well as execution times for the cryptographic algorithm.

Keywords: Co-designed, PRESENT, ultra-lightweight cipher, System on Chip.

I. INTRODUCTION

The complexity of current electronic systems allows interconnecting devices with different architectures in one SOC (System On Chip) or several integrated circuits , these devices having this great computing capacity can perform multiple tasks: acquire and process data [1][2], as well as transmit and receive information in an intelligent network IoT (Internet of things) and/or IIoT (Industrial Internet of things); these tasks should not occupy all the system processing time so that the tasks of acquisition and processing of information are performed satisfactorily[3][4].

These systems must guarantee a minimum of security in the transmission and reception of data [4][5], which is why it is required that cryptographic processes are performed in all IoT applications, and therefore seeks to improve the algorithms and methods of data encryption [6][7]. Cryptographic algorithms are a high expenditure of resources for the system, investing a lot of processing time in creating security mechanisms for information [8][9].

However, this is not a major problem for high-performance computer systems, but it makes it difficult to implement secure low-cost applications [10][11], so it requires the use of minimalist computer security algorithms such as lightweight block ciphers , This article shows the implementation of one of them called "PRESENT"[12], which performs the encryption process using 80 and 128-bit keys and processing blocks of up to 64 bits [13][14], all this implemented in an SoC of the Cypress company using coding techniques to lower the time and amount of resources used in the data encryption process[15][16].

Colombian research groups from the Universidad Distrital (Bogotá, Colombia) have done some work on cryptography[17], deepening in the implementation of light ciphers like CLEFIA and PRESENT on embedded platforms like FPGA and SoC[18][19].

Embedded applications that implement encryption algorithms require communication with a central computer that must do the tasks of storage and processing[19][20], in other words, security tasks typically fall to software made on this computer with greater computing power, these cryptographic algorithms implemented in the central computer are not optimized [21][22]. Cryptography applications and libraries are available in languages such as C/C++ and Python, which take AES (Advanced Encryption Standard) as their default block cipher. However, this article aims to make a code-designed hardware-software implementation [23][24], which improves execution times for the cryptographic algorithm PRESENT in a PSoC® 5LP, thus implementing a lightweight block cipher algorithm in a low-cost device[25][26].

II. METHOD

The implementation was done in a Cypress® PSoC® 5LP that has an ARM core, along with a field-configurable matrix type CPLD, the PSoC Creator is used as a development tool as an IDE; C++ is used for microcontroller processor programming and Verilog as a hardware description language. Next, the process of co-design is shown, starting with the internal architecture of the device used, then the block diagram of the implemented algorithm, and finally the parts of the hardware/software implementation are shown with their respective metrics and the improvement of the algorithm compared to previous works.

II.I SoC Architecture: Cypress PSoC

The implementation was done in a PSoC® 5LP Microcontroller, which has an ARM Cortex®-M3 core, some configurable analog and digital blocks, plus at least 256Kb of

FLASH program memory and at least 64 Kb of SRAM; but the important and significant thing about this low-cost device is its architecture, which allows to directly interconnect the processor bus with the reconfigurable digital blocks. The Cypress Creator IDE allows programming in C and C + + for the embedded software that runs on the Cortex ®-M3 and Verilog to make the description of the hardware of the blocks reconfigurable in the field.

The PSoC® 5LP Microcontroller has mixed hardware with an analog part, an ARM Cortex®-M3 core along with a field configurable array, with some basic blocks called UDB (Universal Digital Block array) interconnected by a system bus, as shown in Figure 1. These configurable digital blocks can be reprogrammed in different ways, with a series of tools, but in general, these tools generate a code in Verilog language.

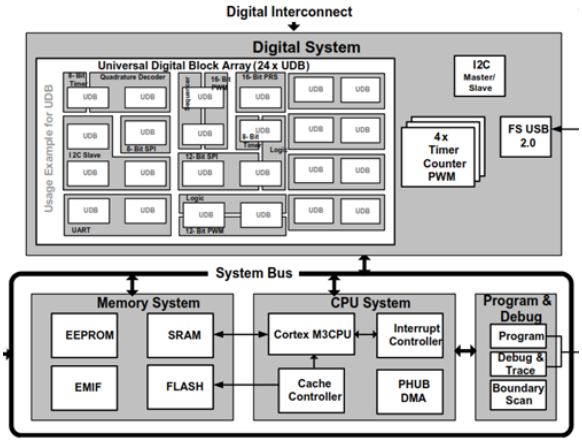

Fig. 1. Interconnection block diagram of the PSoC® 5LP digital blocks [27]

The UDB blocks are distributed in a programmable interconnection matrix. This matrix structure is homogeneous and allows flexibility in the interconnection of functions in the matrix. The matrix supports a complex range of flexible routing interconnections between UDBs and all interconnected digital parts of the system. The architecture of this interconnection matrix is shown in figure 2.

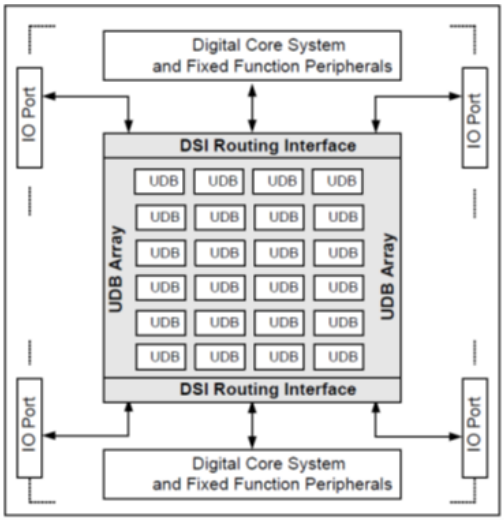

Fig. 2. Digital Programmable Architecture of the PSoC® 5LP [27]

II.II Block diagram of the block cipher: PRESENT

For the co-design process, it is required to know the internal architecture of the device perfectly, to perform the design partition. In this case, a previous work of analysis of the functional blocks of the block cipher algorithm must be performed, measuring the times and the amount of memory used in each block.

An implementation of the block cipher cryptographic algorithm PRESENT [1], which has a block size of 64-bits, using as possible key sizes 80-bits or 128-bits is one of the lightweight algorithms taken as a reference since it has a minimalist architecture, which could work up to a 4-bit processor since the word sizes and S-Box layer substitution tables are adapted to work with nibbles. In figure 3, the general block diagram of the cipher is shown.

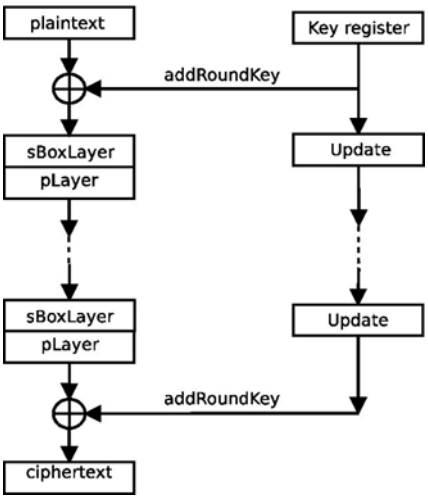

Fig. 3. PRESENT cipher general block diagram

Next, the operation of each of the layers of the algorithm is specified:

- **Byte substitution layer (S-box):** Consists of a non-linear substitution state matrix called S-box, which can describe as a simple table in C/C++ language. This substitution block is applied to 16 nibbles that complete 64 bits of information.

Table 1. sBox Layer Nibble Replacement Valuer [Hex]

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

- **Bit Permutation (pLayer):** This is a pure binary substitution layer, in which 64-bit blocks of information are exchanged bit by bit. This layer requires a series of frameworks to be performed in embedded software using the Cortex-M3 core of the microcontroller, and being purely binary it is the best candidate to be implemented using the logical digital matrix.

Table 2. pLayer Permutation Bit Position

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

- **Key expansion function (addRoundKey):** This is the layer in charge of updating the keys, which can be 80 or 128 bits in length, this design only a key of 80 bits will have implemented, each round the new calculated key will be mixed after applying the key expansion function.
-

II.III Co-design

After having each of the tasks of the subsystems, it is required the designer to carry out an execution planning scheme, making clear the instant of time when the actions will be executed. For hardware, it is required to make clear the function of each of the blocks and how to achieve the synchronization of them [23]. At that time, a Data-Path is performed, in which each of the subsystems is write in a C/C++ language and that will perform the synchronization of the global system with the Player layer which has been described in Verilog.

In the co-design it is required to have knowledge of the internal architecture of the device, to generate the communication and verification channels between the structural blocks of the PSoC® 5LP, for this specific case, when developing algorithms of certain complexity in the previously mentioned platform, tools or design methodologies are required for the implementation and generation of total design tests.

In previous works [24] 24]a full software implementation was made, where the entire cipher ran on the ARM Cortex®-M3 of the PSoC® 5LP, for this purpose some measurements of the processing times of the cipher were made, in addition to specific measurements of the amount of memory used per block and the amount of time measured in seconds of each of the structural blocks of the algorithm, These metrics can be seen in Figure 4, which presents the summary of results in a percentage of program memory usage on the left and the percentages of time used to perform each of the layers of the algorithm on the right.

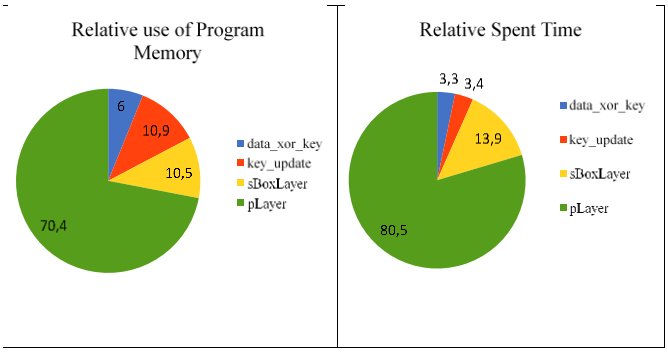

Fig. 4. PRESENT Ciphers metrics on the PSoC® 5LP. Percentage of memory used per layer (left) and runtime per layer (right) [28]

This analysis conducted in this previous work [29], clearly shows that the pLayer is the layer that most requires to be described or redesigned, so it is chosen as the layer that should be made in hardware using the logic available in the UDP's, for this we made a description of this layer hardware, making a thorough review of how to make the access to the processor bus to share the embedded software data with the digital programmable logic.

The hardware description is done classically, making the description of the permutation of the bits, complying with the indicated in Table 2, but to this hardware description must be added a line of code to capture the data from the software "cy_psoc3_control", which is a function that allows to read the data from a software register and write them in a UDP, it is important to clarify that in this way only bytes can be written, these bytes directly affect variables in the Verilog code, since the cipher handles a 64-bit word, you have to call this function 8 times, similarly the function "cy_psoc3_status" allows to read the data after passing through the hardware description to the ARM Cortex®-M3 kernel. Analog functions are performed in the software layer to read and write from the programmable digital layer.

III. RESULT

The results of the implementation of the PRESENT cryptographic algorithm using co-design, specifically performing the pLayer in hardware are shown in figure 5, these results were tabulated by comparing each of the layers of the cipher, verifying the percentage of memory usage on the left and the time used by the microcontroller in the encryption process on the right.

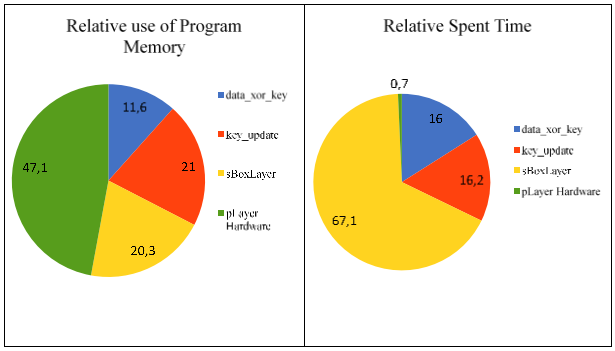

Fig. 5. PRESENT cryptographic algorithm using co-design, specifically performing the pLayer

In these graphs you can make several observations, on the left side you can see the significant change in memory usage, where before the pLayer layer used 70.4% of the application's memory and now it only uses 47.1% of the same, on the other hand, the most significant change is observed in the amount of time required by the algorithm to perform the pLayer, since before the vast majority of the processing time of the cipher 80.5% was to perform the permutations of this layer, but now only 0.7% of the time is required since the permutation of the bits required for this layer is done concurrently, the only time used is the time required by the processor to load data into the hardware layer.

To better understand the number of resources used in the cipher, table 3 is presented, where the measurements of the amount of program memory used in the whole cipher and each of its layers is shown. This metric is made by subtracting the memory used by the bootloader to load the program to the microcontroller.

Table 3. Use of program memory measured in bytes

	FLASH memory (bytes)	
	Software	**Codesign**
cipher	2136	976
data_xor_key	128	128
key_update	232	232
sBoxLayer	224	224
pLayer	1504	520

When implementing the pLayer using the SoC hardware only 45.69% of the FLASH memory is required, compared to the PRESENT implementation only described in software for the same device.

As for the amount of time used by the cipher, a measurement is made using the system clock called "sys_tick", using this tool the times of each of the layers were taken, in table 4 the results are shown, starting with the amount of time used to cipher a block of information and each of the layers, taking into account that this data is for the 31 rounds of the algorithm.

Table 4. Use of program memory measured in bytes

Execution time (Time milliseconds)		
	Software	**Co-design**
cipher	6.9875	1.44896
data_update	6.69178	
data_xor_key	0.232	0.232
key_update	0.23448	0.23448
sBoxLayer	0.97228	0.97228
pLayer	5.626	0.0102

You can see the decrease in time as the pLayer, which was optimized using programmable logic, since this layer is combinatorial only, requiring an exchange of a few bits in a D-type flip-flop, this makes the processing time required is only limited to the loading and reading of data in the hardware layer. In this case, the processing time decreases from 5,626 milliseconds to only 0,0102 milliseconds, which, makes the total cipher time decrease from 6,9875 milliseconds to 1,44896 milliseconds, that is, the processing time was reduced by 79.26% compared to the one used to cipher a block of information in the full software implementation.

IV. CONCLUSION

By performing a thorough analysis of the important metrics in an algorithm, it can be determined which of its components are susceptible to be improved, in this case, the use of FLASH memory is improved, it makes the algorithm go down from 2k bytes of program memory usage to only 1 K byte approximately.

In addition to this, the throughput of the application should be analyzed, so that it is a usable algorithm in application IoT or similar, in this case, by reducing the processing time of each of the blocks you get a system output of 44.19 Kbps, which makes it more than enough for this type of applications.

At the time that the algorithm was developed, it was thought to be implemented on a device that would cost only 10 USD or less, which makes it is by definition a minimalist application from the point of view of cost and the number of resources required, but at the time of this publication, the company Cypress® updated this family of devices from 24Mhz to 80Mhz, which makes the throughput increase significantly in a faster device with the same cost and architecture.

REFERENCES

[1] A. Bogdanov *et al.*, “PRESENT: An ultra-lightweight block cipher,” in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2007, pp. 450–466.

[2] Z. Gong, S. Nikova, and Y. W. Law, "KLEIN: a new family of lightweight block ciphers," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, 2011, pp. 1–18.

[3] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: an ultra-lightweight blockcipher," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2011, pp. 342–357.

[4] D. Engels, M.-J. O. Saarinen, P. Schweitzer, and E. M. Smith, "The Hummingbird-2 lightweight authenticated encryption algorithm," in *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, 2011, pp. 19–31.

[5] M. Alizadeh, M. Salleh, M. Zamani, J. Shayan, and S. Karamizadeh, "Security and performance evaluation of lightweight cryptographic algorithms in RFID," *Kos Island, Greece*, 2012.

[6] Y.-C. Lee, "Two ultralightweight authentication protocols for low-cost rfid tags," *Appl. Math. Inf. Sci.*, vol. 6, no. 2S, pp. 425–431, 2012.

[7] X. Bai, L. Jiang, Q. Dai, J. Yang, and J. Tan, "Acceleration of RSA processes based on hybrid ARM-FPGA cluster," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 682–688.

[8] S. Kerckhof, F. Durvaux, C. Hocquet, D. Bol, and F.-X. Standaert, "Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2012, pp. 390–407.

[9] B. K. B. Raju, A. Krishna, and G. Mishra, "Implementation of an efficient dynamic AES algorithm using ARM based SoC," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, 2017, pp. 39–43.

[10] E. Andreeva *et al.*, "APE: authenticated permutation-based encryption for lightweight cryptography," in *International Workshop on Fast Software Encryption*, 2014, pp. 168–186.

[11] B. Zhou, M. Egele, and A. Joshi, "High-performance low-energy implementation of cryptographic algorithms on a programmable SoC for IoT devices," in *High Performance Extreme Computing Conference (HPEC), 2017 IEEE*, 2017, pp. 1–6.

[12] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "Spongent: The design space of lightweight cryptographic hashing," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2041–2053, 2013.

[13] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and K. Rantos, "Lightweight cryptography for embedded systems–A comparative analysis," in *Data Privacy Management and Autonomous Spontaneous Security*, Springer, 2014, pp. 333–349.

[14] M. Grillo, W. Pereira, and Y. Cardinale, "Seguridad para la Autentificación, Cifrado y Firma en la Ejecución de Servicios Web Compuestos.(P. 106-118)," *Tekhné*, vol. 1, no. 18, 2017.

[15] H. Ning, H. Liu, and L. Yang, "Cyber-entity security in the Internet of things," *Computer (Long. Beach. Calif).*, vol. 46, no. 4, p. 1, 2013.

[16] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark: A lightweight hash," *J. Cryptol.*, vol. 26, no. 2, pp. 313–339, 2013.

[17] M. Mozaffari-Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5925–5932, 2013.

[18] V. Ruiz-Rosas, J. Forero-Casallas, and C. Bohórquez-Ávila, "Optimización topológica de un semirremolque tipo plataforma-Topological optimization of a platform semitrailer," *Rev. científica*, vol. 2, no. 19, pp. 56–63, 2014.

[19] R. De Clercq, L. Uhsadel, A. Van Herrewege, and I. Verbauwhede, "Ultra low-power implementation of ECC on the ARM Cortex-M0+," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, 2014, pp. 1–6.

[20] N. Arora and Y. Gigras, "FPGA implementation of low power and high speed hummingbird cryptographic algorithm," *Int. J. Comput. Appl.*, vol. 92, no. 16, 2014.

[21] E. E. Gaona García, S. L. Rojas Martínez, C. L. Trujillo Rodríguez, and E. A. Mojica Nava, "Authenticated encryption of pmu data," *Tecnura*, vol. 18, no. SPE, pp. 70–79, 2014.

[22] C. Franck, J. Großschädl, Y. Le Corre, and C. L. Tago, "Energy-Scalable Montgomery-Curve ECDH Key Exchange for ARM Cortex-M3 Microcontrollers," in *2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2018, pp. 231–236.

[23] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, 2015, pp. 1–6.

[24] H. Tschofenig and M. Pegourie-Gonnard, "Performance of state-of-the-art cryptography on ARM-based microprocessors," 2015.

[25] S. Belguith, A. Jemai, and R. Attia, "Enhancing data security in cloud computing using a lightweight cryptographic algorithm," in *The Eleventh International Conference On Autonomic and Systems*, 2015, pp. 98–103.

[26] A. K. Luhach, "Analysis of lightweight cryptographic solutions for Internet of Things," *Indian J. Sci. Technol.*, vol. 9, no. 28, 2016.

[27] A. T. Dust and G. Reynolds, "AN82156 Designing PSoC Creator Components with UDB Datapaths."

[28] F. Martínez Santa, E. Jacinto, and H. Montiel, "PRESENT cipher implemented on an ARM-based system on chip," *Commun. Comput. Inf. Sci.*, vol. 1071, pp. 300–306, 2019, doi: 10.1007/978-981-32-9563-6_31.

[29] G. Edwar Jacinto, A. Holman Montiel, and S. Fernando Martínez, "Implementation of the cryptographic algorithm 'present' in different microcontroller type embedded software platforms," *Int. J. Appl. Eng. Res.*, vol. 12, no. 19, pp. 8092–8096, 2017.