Contemporary Engineering Sciences, Vol. 10, 2017, no. 27, 1297 - 1307
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ces.2017.710153

Functional Comparison of the Present Algorithm on Hardware and Software Embedded Platforms

Edwar Jacinto Gómez

District University Francisco José de Caldas
Bogotá, Colombia

Fredy Martínez Sarmiento

District University Francisco José de Caldas
Bogotá, Colombia

Holman Montiel Ariza

District University Francisco José de Caldas
Bogotá, Colombia

Abstract

This article shows the implementation of the cryptographic algorithm "Present" in both software and hardware embedded platforms, for which measurements of the performance results of the implementation of the algorithm are performed on the different microcontrollers and also on FPGA type programmable logic devices. Encryption and decryption functions are performed using 80-bit key sizes, performing operation tests for the test vectors. In the microcontroller type software platforms, the use of program memory (FLASH), use of data memory (RAM) and throughput are taken into account; all of this keeping in mind the size of the processor bus and if the processor has pointers to functions or tables and the CPU speed measured in millions of instructions per second MIPS. As for hardware embedded platforms, in this case the throughput, it is significantly higher due to the

parallelism characteristic of the devices, but the reduction of the use of resources in terms of occupied Slices is sought (CLB-GE); These Slices or functional blocks are different for each company, which makes them difficult to compare.

Keywords: Block Ciphers, Lightweight algorithms, Cryptology, Embedded software/hardware

1 Introduction

In certain applications it is necessary to explore solutions using embedded processors [1], and hardware encryption devices [7], taking advantage of the advance in microelectronics, to achieve fast applications [4] looking for high processing speed, to achieve the real-time condition. In today's communication networks, data is required to travel safely giving a minimum throughput; for this, the algorithms must have efficient hardware-software implementations. Because of this reason, a versatile encryption algorithm [2] is applied, which is flexible, with low use of resources [3] [5, 6] and which is easy to implement regarding hardware and software.

In hardware implementations, fast applications are achieved where data traffic is greater and require real-time encryption [4]. In this case FPGA's are a very good alternative, thanks to being reconfigurable and their parallelism makes them optimal for massive and complex processes with good response times [3] [5, 6].

2 Methodology

2.1 Block Cipher

In this type of encryption, the information is organized in blocks of fixed size, regardless of the place it occupies in the binary chain, so that all the bits of the block are coded together, performing operations that try to eliminate the possible relationships between the encrypted text and the original message [8].

In all of the block ciphers a combination of operations is generated: ByteSub, ShiftRow, MixColum and AddRoundKey; these combinations are called round. The complexity of the algorithm depends on the combination of these operations and the rounds performed along with the size of the key used [9, 10].

2.2 Present Algorithm

PRESENT is a block cipher algorithm lightweight type highly used since it has easy implementation in both hardware and software [3, 5], see Fig. 1; its implementation in hardware can be done in some of the smallest FPGA's on the market, with a high throughput [6] and in bus microcontrollers from 8 bits to 32 bits with a few kilobytes of RAM.

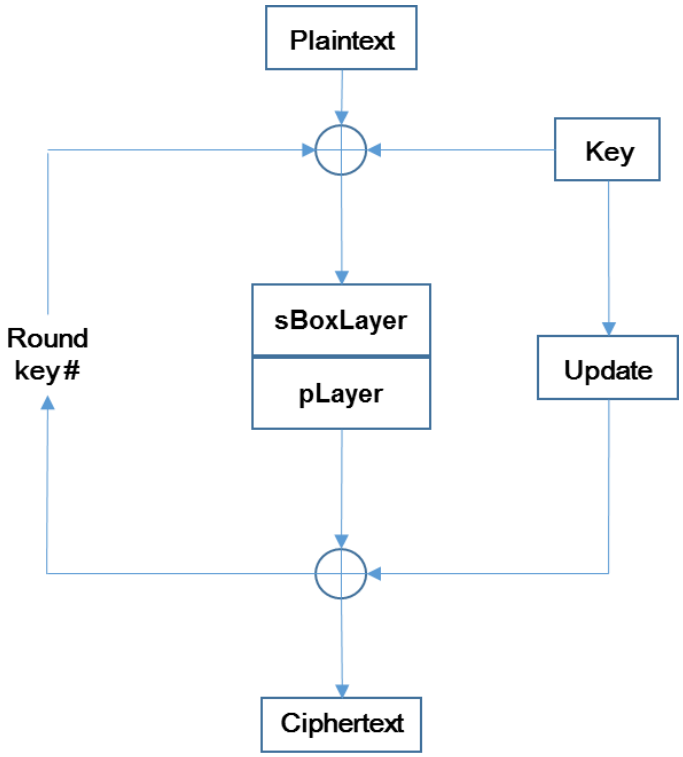

Figure 1: Structure of the Present algorithm

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

- **Byte substitution layer:** Consists of a non-linear substitution that is applied independently to each nibble of the state matrix. This substitution block is applied to 16 nibbles that complete 64 bits of information, which is the standard size of the cipher blocks.

- **Bit Permutation (pLayer):** It is a layer that mixes by means of a bitwise substitution a block of information of 64 bits, where the "i" bit of the round is moved to the position P(i).

- **Key expansion function (addRoundKey):** PRESENT can have keys of 80 or 128 bits in length, but for this design and implementation only a key of 80 bits will be taken into account, which will be stored in a K register of that size and will be listed K_ {79} K_ {78} .... K_0. But in each round only the 64 most significant bits of the new calculated key will be mixed after applying the key expansion function.

2.3 Embedded design platforms

The evolution of programmable digital devices has allowed a great leap in the possibilities of high-performance developments in low-cost devices [11]. Next, a

review of some of its characteristics is performed: programming and verification tools, market trends, advances in its architecture, performance, as well as the generation of standard code that can be used in subsequent designs.

2.4 ESL Design: Electronic System Level

The design of digital systems has had a leap in the complexity and capacity of the systems, which requires higher level tools and languages to be used being able to change the design dynamically. This is achieved with methodologies that manage certain levels of abstraction to describe these systems [11]; in this evolution, it is necessary to use languages as C / C ++ and the HW / SW co-design makes possible to reuse code and libraries in applications with high complexity. These tools allow to approach both hardware design and software design, providing the possibility that solutions can be made using both types of design.

2.5 Top-Down Design

In digital systems, one of the most used methodologies is to partition the design into smaller subsystems, starting from a functional specification until reaching the physical elements of the chosen technology. When defining the functional specification of the system, the subsystems are designed making verification and debugging them. The use of a design hierarchy allows you to partition the system into a series of subsystems of reduced complexity that can be designed by different work groups.

The Top-Down methodology allows the re-use of source code and the described system to be used as a subsystem in a more complex design. This technique requires the definition and storage of design libraries, as well as the reuse of generic parameterizable designs. The reuse of blocks allows to increase the reliability of a new design, by presenting tested and purified components, which reduces the possibility of errors and the development time.

2.6 Task planning

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. At that time, a Data-Path is performed, in which each of the subsystems is controlled by a finite state machine that will perform the synchronization of the global system.

2.7 Co-design

In the co-design it is required to have knowledge of the hardware and software embedded platforms, to generate the communication and verification channels between

the platforms, for this specific case, when developing algorithms of certain complexity in the previously mentioned platforms, tools or design methodologies are required for the implementation and generation of total design tests.

In this paper a VHDL Data-path was generated to describe each of the layers of the Present algorithm, controlled by a state machine. Then, the design of the cipher was controlled by a free Soft-Core, which was implemented in the FPGA and in which a code was run in C to perform verification tests of system operation, in addition to communication tasks both with the embedded software platform and with some tests performed with the use of a PC.

In the same way, an embedded software solution was made, emphasizing in making each of the layers described in standard functions, which allow the use of dynamic memory and re-use of the code, in such a way that at the end of the design the cipher could be used as a library called from another application that uses RTOS (Real Time Operating Systems).

3 Implementation and Results

A review was made of microprocessors, microcontrollers and PLDs that could support the PRESENT algorithm, taking into account their architecture, price, available compilers, supported programming languages, memory, speed and local acquisition facility.

3.1 Embedded software platforms

A series of comparisons were made between Parallax Inc. microcontrollers, Microchip, Atmel, Texas Intruments, Cypress Semiconductors and FreeScale which is now part of NXP, analyzing the speeds, CPU bus size, FLASH memory and RAM, analyzing several of their families with different configurations and peripherals; the specific analysis performed is shown in Table1.

Table 1. Microcontrollers Comparison

Microprocessor	**PIC16F**	**PIC18F**	**PIC32**	**MSP 430**	**TI Stellaris**	**FRDM-KL25Z**	**PSOC 4**
Language	ASM, C/C++	ASM, C/C++	ASM, C/C++	ASM, C/C++	ASM, C/C++	ASM, C/C++	ASM, C/C++
Supports ANSI C/C++	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Compiler	Mplab XC8, Picc CCS	Mplab C18, Picc CCS	Mplab XC32	IAR Kickstart, Code Composer	IAR Kickstart, Code Composer	Mbed	Psoc Creator, Psoc Designer

Table 1. (Continued): Microcontrollers Comparison

Architecture	RISC Microchip	RISC Microchip	MIPS	RISC	ARM Cortex M4	ARM Cortex M0	ARM Cortex M0
MIPS	5	10	124	16	40	48	48
Data bus width	8 bits	8 bits	32 bits	16 bits	32 bits	32 bits	32 bits
Memory	0.5-2Kb RAM 1-32Kb ROM	0.5-4Kb RAM 1-96Kb ROM	8K-32Kb RAM 32-512Kb ROM	0.5Kb RAM 8-16Kb ROM	hasta 32Kb RAM 256Kb ROM	0.5-2Kb RAM 1-16Kb ROM	16k b RAM 128Kb ROM
Max clock speed	20 Mhz	40 Mhz	80 Mhz	16 Mhz	80 Mhz	24 Mhz	48 Mhz
Development system	No	Yes	Yes	Yes	Yes	No	Yes
Packaging	DIP/SOIC/QFN	DIP/SOIC/QFN	TQFP/QFN/XBGA	DIP/SOIC	LQFP	DIP/SOIC/TQFP	SOIC/TQFP
Peripherals available	UART,SPI, I2C, ADC10Bit,PWM	PIC16 + CAN, USB	EQUAL PIC18F	UART,SPI, I2C, ADC10 bits, IrDA, ultra low power	EQUAL MSP430 + 8 UART, 4 I2C, 4 SPI, USB	6 Analog Blocks, 4 Digital Blocks, Cap Sense, Mosfet	Equal Psoc 1 + CAN, USB
Availability in the country	Yes	Yes	No	Yes	Yes	No	Yes
Chip price (COP)	5000-10000	13000-35000	5-10 USD	2500-34000	39000-82000	5000-20000	28000-62500

After this revision it is decided to carry out the implementation in several devices, in order to have measurement parameters in relation to the size of the processor bus. It is chosen to use 8 and 16 bits families of Microchip TM microcontrollers; this decision is made keeping in mind that the compiler is standard for C / C ++, besides that a quick verification is allowed thanks to the power of its simulators and development tools. Microchip is currently the most widely spread company in the world in 8 bits. In terms of 32-bit families, the ARM Cortex-M0 microprocessor is selected on the Mbed KL 25Z development card, due to the fact that the development time of the prototype is reduced by the ability of its online compiler that supports C / C ++ along with the handling of pointers, dynamic memory allocation, version control with mercurial/git and code formatting. Its cost, number of modules and ease of programming, surpass any other device in this range.

In all the block ciphers some operations or rounds are generated. The complexity and security of the algorithm depends on the combination of these basic operations and the number of rounds performed on the information to be encrypted. Each round consists of four operations based on uniform and invertible transformations that are called layers, which are designed to resist linear and differential cryptoanalysis.

It is possible to summarize the implementation in a compilable pseudocode in any embedded software platform, either 8, 16 or 32 bits in the following way:

```
generateRoundKeys()
  for i=1 to 31 do
  addRoundKEy(STATE,Ki)
 SBoxLayer(STATE)
 pLayer(STATE)
end for
addRoundKey(STATE,K32)
```

The code was tested for several platforms; with the test vector suggested by NIST in 8-bit microcontrollers (16FXXX and 18FXXX) with and without internal pointers, in 16-bit devices (24FJXXX) and in a 32-bit ARM Cortex-M0 architecture (KL25Z128). Said tests were carried out by transmitting the data through the UART of the processor and verifying the processing times.

Table 2. Performance parameters for embedded software platforms

Microcontroller	**Program Memory Flash (bytes)**	**Memory Data RAM (bytes)**	**Throughput bps**	**MIPS**
8 bits	2210	73	768	5
16 bits	2129	768	775,68	5
16 bits	3276	245	1203,2	10
32 bits	20,5 K	600	14976	48

The code of the cipher is written in C language, compatible with the different architectures, especially with those made for ARM 32-bit microcontrollers; this makes the work totally original.

3.2 Embedded platforms Hardware

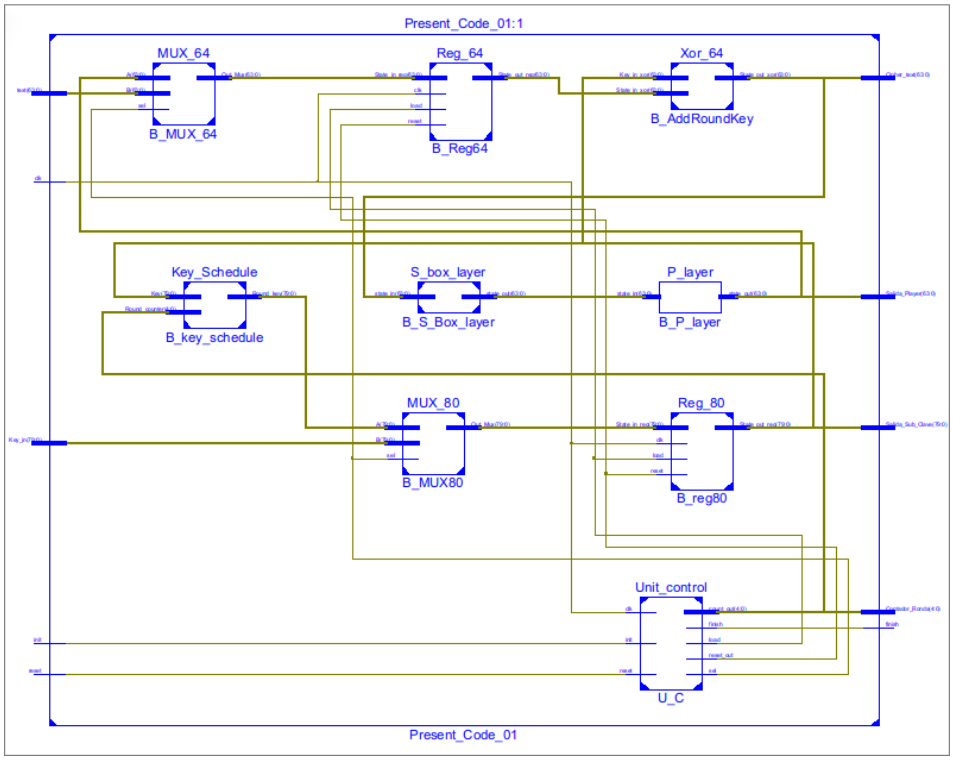

Figure 2: Real data-path of the cipher in VHDL

Regarding hardware embedded FPGA type platforms, the competition is only summarized to two companies, Xilinx and Altera with more than 90% of global sales. The design of the system must be done in the standard hardware description language, avoiding the use of IP's or any other feature that prevents the design from being implemented in any of the devices of the two companies. In this case it is decided to make a description of the block cipher using the VHDL hardware description language, which is standard for devices of any range, both Altera and Xilinx. For the elaborated development all the tests were carried out on the Spartan 3E of Xilinx. It should be noted that the resulting code is fully compatible with any FPGA of any range and manufactured by any company.

The implemented cipher was designed with the Top-Down methodology, generating a "core" code that can be included as a component in a higher-level design; Below are the blocks that were designed, tested and that implemented each of the functional parts of the Feistel network; Fig. 2 shows a block diagram, taken directly from the ISI tool of Xilinx.
Each of the blocks was described trying to use the least possible resource. Each block was simulated and tested, to guarantee the correct functioning of the system. After having fully described the hardware, it was proceeded to perform a simulation

of the system; this process is shown in Fig. 3. It is important to emphasize that being a highly complex algorithm and having no intermediate points in the test, the process of debugging and verification was quite long and wasteful. This task requires the re-design of each of the blocks and the control unit for each step.

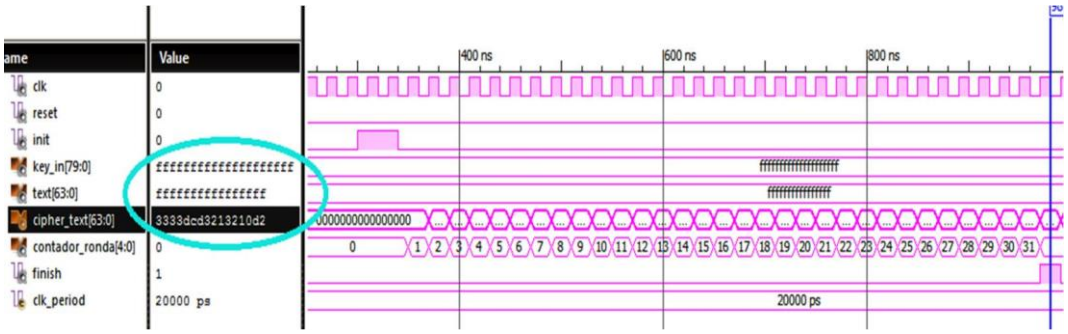

Figure 3: Step-by-step simulation of the hardware implementation

By having step-by-step simulation of the hardware, it was possible to verify each intermediate result of the embedded software design and perform debugging in each of the rounds. By having the complete implementation, a comparison was made of the hardware used and the hardware made by the people who proposed the algorithm; it is determined that it has an improvement of 26.89% regarding the use of the resource.

Table 2. Comparative table of some lightweight encryption algorithms implemented in Hardware

	Key size	**Block size**	**Cycles per block**	**Throughput at 100KHz (Kbps)**	**Logic process µm**	**Area GE**
PRESENT-80	80	64	32	200	0.18	1570
AES-128	128	128	1032	12.4	0.35	3400
HIGHT	128	64	34	188.2	0.25	3048
mCrypton	96	64	13	492.3	0.13	2681
Camellia	128	128	20	640	0.35	11350
DES	56	64	144	44.4	0.18	2309
DESXL	184	64	144	44.4	0.18	2168

4 Conclusions

The implementation in 16-bit and 32-bit microprocessors is completely unprecedented. There is no documentation of similar applications prior to this investigation. In addition, with the 32 bits a throughput is achieved that is usable in the applications of sensor networks and applications for which the algorithm is designed.

A fully standard C code was developed for families of 8, 16 and 32 bits that can also be used on computers. Taking into account that the 32-bit architecture was implemented on the most used family on the market today, the Cortex M0 of ARM, with a throughput of 234 messages per second (14.9 Kbps) that allows its use in real applications.

Regarding the development on hardware platforms, a code type CORE or library (software) totally standard for different families of devices was generated, which can be used in a simple way in real applications. The versatility and functionality of the generated tool ensures that the implementation is possible for FPGA manufacturers other than Xilinx.

References

[1] R. Beaulieu, D. Shors and J. Smith, The simon and speck block ciphers on AVR 8-bit microcontrollers, Chapter in *LightSec 2014 proceedings*, Springer Cham, Vol. 1, (2014) 1, 3–20. https://doi.org/10.1007/978-3-319-16363-5_1

[2] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks and L. Wingers, The simon and speck families of lightweight block ciphers, *Cryptology ePrint Archive*, Report 2013/404 (2013).

[3] A. Bogdanov, L. Knudsen, G. Leander and C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, C. Vikkelsoe, PRESENT: An Ultra-Lightweight Block Cipher, chapter 5, *Cryptographic Hardware and Embedded Systems - CHES 2007*, Springer-Verlag Berlin Heidelberg, 2007, 450–466. https://doi.org/10.1007/978-3-540-74735-2_31

[4] F. Chih-Peng and H. Jun-Kui, Implementations of high throughput sequential and fully pipelined AES processors on FPGA, *International Symposium on Intelligent Signal Processing and Communication Systems ISPACS 2007*, (2007), 353–356. https://doi.org/10.1109/ispacs.2007.4445896

[5] E. Kavun and T. Yalcin, RAM-based ultra-lightweight FPGA implementation of PRESENT, *Reconfigurable Computing and FPGAs (ReConFig) International Conference on 2011*, (2011), 280–285. https://doi.org/10.1109/reconfig.2011.74

[6] J. Pospiil and M. Novotny, Evaluating cryptanalytical strength of lightweight cipher PRESENT on reconfigurable hardware, *2012 15th Euromicro Conference on Digital System Design (DSD)*, (2012), 560–567. https://doi.org/10.1109/dsd.2012.53

[7] P. Yalla and J. Kaps, Lightweight cryptography for FPGAs, *International Conference* on *Reconfigurable Computing and FPGAs ReConFig '09,* (2009), 225–230. https://doi.org/10.1109/reconfig.2009.54

[8] N. Pineda and N. Velásquez, Diseño e Implementación de un Prototipo Criptoprocesador AES-Rijndael en FPGA, *Universidad de los Llanos*, Colombia, (2007).

[9] J. Attridge, An overview of hardware security modules, *SANS Institute InfoSec Read Room*, **1** (2002), 1–10.

[10] A. Aysu, E. Gulcan, P. Schaumont, SIMON says: Break area records of block ciphers on FPGAs, *IEEE Embed Syst Lett.*, **6** (2014), 37–40. https://doi.org/10.1109/les.2014.2314961

[11] A. Delgadillo, M. Guerrero and N. Pena, Diseno de un criptosistema para redes de sensores inalambricos WSN basado en MPSOC, *Universidad de los Andes*, (2008).

Received: November 9, 2017; Published: December 4, 2017