Research Journal of Applied Sciences, Engineering and Technology 10(8): 914-922, 2015
DOI:10.19026/rjaset.10.2447
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: January 31, 2015 Accepted: March 7, 2015 Published: July 20, 2015

Research Article

PRINCE IP-core on Field Programmable Gate Arrays (FPGA)

[1, 2]Yasir Amer Abbas, [1]Razali Jidin, [1]Norziana Jamil, [3]Muhammad Reza Z'aba and [1]Mohd Ezanee Rusli
[1]Center for Automation and Embedded Computing Systems (CAECS), College of Engineering, Universiti Tenaga Nasional, Selangor, Malaysia
[2]College of Engineering, Diyala University, Baquba, Diyala, Iraq
[3]MIMOS Berhad, Kuala Lumpur, Malaysia

Abstract: This study presents a high execution-speed and low-resource hardware IP-Core of PRINCE light weight block cipher on a Field Programmable Gate Arrays (FPGA). The new FPGA IP-core is to speed-up the performance of PRINCE, superseding software implementation that is typically slow and inefficient. The design of this IP core is based on concurrent concept in encrypting blocks of 64 bits data, in that each block is executed within one clock cycle, resulting in high throughput and low latency. Though this IP core encrypts data at high speed processing, it consumes relatively low power. The hardware design can allow encryption, decryption and key schedule to utilize identical hardware components, in order to reduce further the FPGA resources. This efficient PRINCE hardware architecture has been coded using Very High speed integrated circuit Hardware Description Language (VHDL). Also, a bus interface has been included as part of PRINCE IP core to allow it to communicate with an on-chip microprocessor. The IP core has been successfully synthesized, mapped, simulated and tested on an FPGA evaluation board. The test program that has been written in "C" to evaluate this IP-Core on a Virtex-403 FPGA board yields an encryption throughput of 2.03 Gbps or resource efficiency of 2.126 Mbps/slice.

Keywords: Block cipher, FPGA, IP-core, PRINCE, VHDL

INTRODUCTION

The emerging technology these days are in the forms of small or miniature devices, such as contactless smart cards, radio-frequency identification and sensor nodes, previously performed by bulky size of machines or devices. However, these devices come with several limitations such as low processing capability, small memory space availability and battery operated, thus shorter lifetime. These devices are referred to as resource constrained devices. In some information processing applications, these devices also perform cryptographic processing to protect data confidentiality (Deshpande *et al*., 2009).

The most widely used cryptographic algorithm is the Advanced Encryption Standard (AES) (Rouvroy *et al*., 2004; Daernen and Rijmen, 2002). The implementation of AES in information processing applications often appears in software form since most of the cipher's components can be implemented using lookup tables which are efficient in software. In order to further improve the speed of the cipher, it can be implemented on FPGA (Grabher *et al*., 2008; Guo *et al*., 2008; Sbeiti *et al*., 2009; Rolfes *et al*., 2008).

Despite these benefits, AES is not suitable to be implemented on resource-constrained devices such as in reconfigurable embedded system boards since it requires a large hardware footprint (Borghoff *et al*., 2012). Therefore, in order to provide data confidentiality for resource constrained devices, the use of lightweight block ciphers is seen as the potential solution.

Different lightweight block ciphers have been proposed with the goal of producing low hardware footprint or better performance in software (Doröz *et al*., 2014; Gielata *et al*., 2008; Standaert *et al*., 2007; Chodowiec and Gaj, 2003). The implementation results for a lightweight instruction set extension for bit-sliced AES software implementations was provided by (Grabher *et al*., 2008). Guo *et al*. (2008), the overall performance of a System-on-Chip (SoC) platform that uses PRESENT lightweight block cipher (Sbeiti *et al*., 2009), as a cryptographic co-processor has been investigated. Three different ultra-lightweight architecture of cryptographic algorithms and their stability have been presented by Rolfes *et al*. (2008). These designs have been implemented using ASIC. However, none of the implementation is mentioned above utilizing small area and small latency.

PRINCE is another example of a lightweight block cipher. It has been designed specifically with low latency in mind. PRINCE permits encryption of data

Corresponding Author: Yasir Amer Abbas, Center for Automation and Embedded Computing Systems (CAECS), College of Engineering, Universiti Tenaga Nasional, Selangor, Malaysia

within one clock cycle and a very competitive chip area compared to other known algorithms. A complete design in such algorithm needs to be implemented using innovative design choices. The number of rounds must be regular and shall produce shorter delays in hardware (Borghoff *et al*., 2012).

In this study, the IP-Core hardware with concurrent architectures design for PRINCE algorithm has been implemented on FPGA with XILINX Virtex-403. The FPGA Virtex-403 board exhibits high throughput of about 2.03 Gbps and low slices number of 956 for both encryption and decryption. To the best of our knowledge, this is the first time that PRINCE algorithm is implemented on an FPGA chip.

METERIALS

The PRINCE algorithm: PRINCE is a 64-bit Substitution-Permutation Network (SPN) lightweight block cipher which supports a 128-bit key (Borghoff *et al*., 2012). The cipher has 12 rounds at its core and each round function consists of the addition of a round-dependent constant and a fixed key, sixteen 4×4 parallel S-boxes and a linear diffusion. The first half of the 128-bit secret key is used as the pre- and post-whitening keys while the second half of the key is used directly in the round functions.

To perform decryption, the key is first XORed with a fixed value and the same circuit for encryption can be re-used. The overhead of performing decryption is therefore minimized. PRINCE is the first lightweight block cipher to be optimized with respect to latency (Doröz *et al*., 2014). Previous proposals focus mainly on having a small footprint in hardware. Encryption using PRINCE can be performed in just one clock cycle if an unrolled implementation is deployed. To achieve this and to also keep down the area requirement, the designers select an optimal S-box and keep the number of operations in the linear diffusion part to a minimum.

The Price core encryption is depicted in Fig. 1. The components of this cipher are highlighted as follows.

Key schedule: The 128-bit secret key is split into two halves: k_0 and k_1. The first half k_0 is used directly as the pre-whitening key. For post-whitening, another key denoted as $k_0' = (k_0 >>> 1) \oplus (k_0 >> 63)$ is used, which is a slight modification of k_0. Pre-and post-whitening refers to the addition of key content before and after a core cipher operation. In the case of PRINCE, the core operation is referred to as PRINCE core. The key k_1 is used directly in the key addition phase of the round functions $\mathcal{R}$ and $\mathcal{R}^{-1}$.

Round functions $\mathcal{R}$ and $\mathcal{R}^{-1}$: The round function consists the following: an XOR with a fixed key k_1, an XOR with a round-dependent constant RC, a S-Box layer S (and its inverse S^{-1}) and a linear diffusion M (and its inverse M^{-1}).

The round-dependent constant: The constants are defined as $RC_i \oplus RC_{11-i} = \alpha$ for $0 \le i \le 1$, with α = coac29b7c97c50dd (in hexadecimal).

S-box layer S (and its inverse S^{-1}): The S-box layer uses a mapping of 4 to 4-bit, as defined in Table 1.

Linear diffusion layer M (and its inverse M^{-1}): The linear layer XORs three input bits to produce a single output bit. Each output bit utilizes different input bits. It is designed in such a way as to maximize diffusion.

The middle involution: This component is composed of the functions SR^{-1}, M′ and SR. The functions SR and SR-1 are similar to the AES Shift Rows operation while M′ is a linear diffusion. The middle involution can be seen as a connector for the forward and inverse round functions. This has the effect that encryption with key k is equal to the decryption with key (k⊕α).

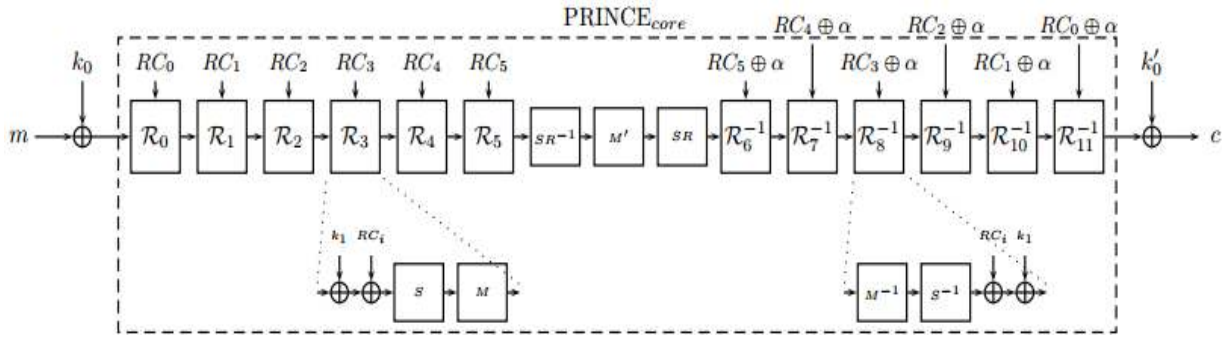

Fig. 1: The PRINCE core encryption (Borghoff *et al*., 2012)

Table 1: S-box and s-box inverse layer of prince (Borghoff *et al*., 2012)

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S (X)	B	F	3	2	A	C	9	1	6	7	8	0	E	5	D	4
X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S^{-1} (X)	B	7	3	2	F	D	8	9	A	6	4	0	5	E	C	1

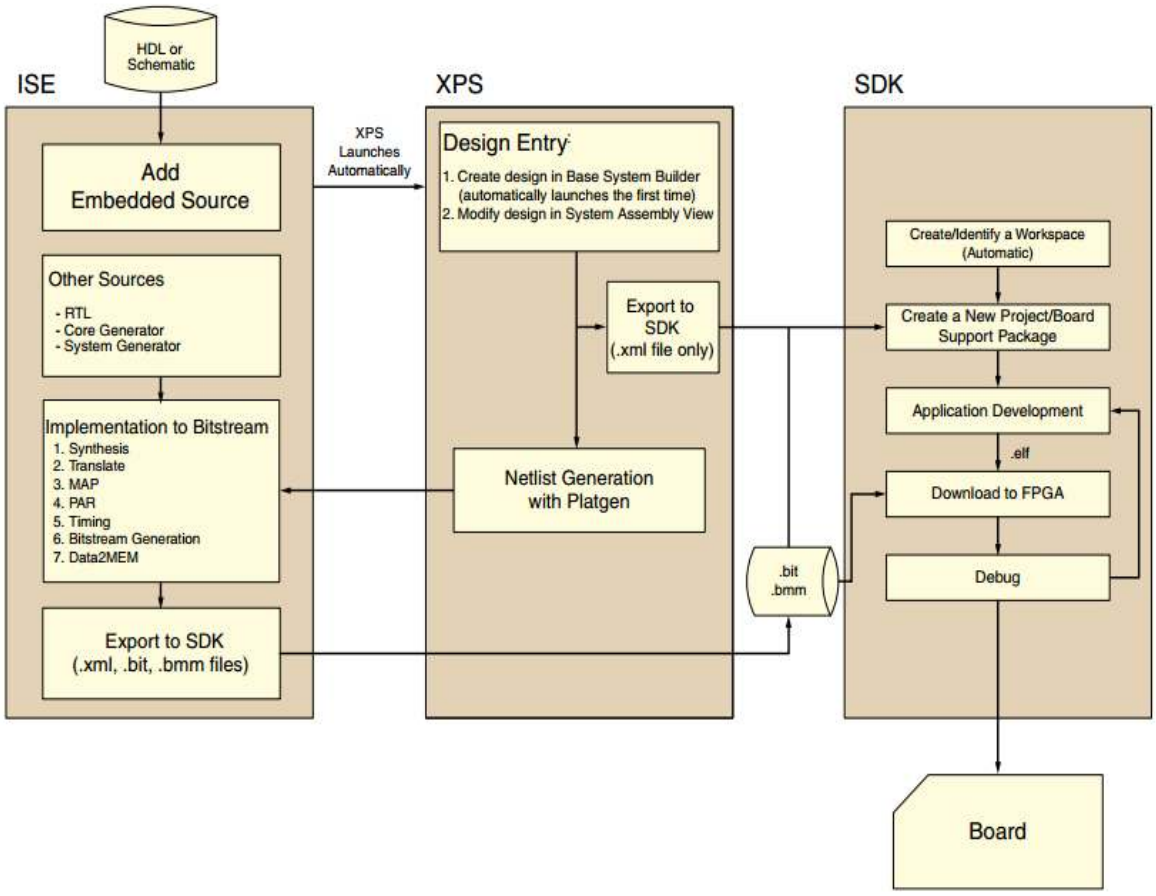

Fig. 2: Embedded design process flow (Xilinx EDK V14.5, 2013)

FPGA's embedded system design: The Xilinx Embedded Development Kit (EDK) system tool is deployed to design a complete embedded processor system for implementation on a Xilinx FPGA board. EDK is a component of Integrated Software Environment (ISE) design suite embedded and system editions. ISE is a Xilinx development system that is used to implement designs onto Xilinx programmable logic devices. The EDK consists of three main system tools. First is the Xilinx Platform Studio (XPS) system tool is used to develop embedded processor hardware. Second is the Software Development Kit (SDK), based on the Eclipse open-source framework, which using to develop the embedded software application. The SDK is also available as a standalone program. Third is Intellectual Property (IP) cores such as processors and peripherals libraries (Xilinx EDK V14.5, 2013). Figure 2, shows the embedded design process flow relevant to our project.

The embedded system development are divided into two main part; hardware and software, the hardware development uses the XPS to implement the PRINCE algorithm, processor and other required peripheral sinter connected via processor local buses. This hardware is then transferred or loaded to the software design part, the SDK as a Microprocessor Hardware Specification (MHS) files. The SDK is utilized to create different applications for the previously loaded hardware design, that programming languages options are C and C++ language.

METHODOLOGY

The design of PRINCE algorithm using VHDL: The main goal of using the proposed PRINCE block cipher algorithm has been described in Materials Section. Therefore, this cipher is realizable for different future broad applications to meet real-time requirement. The IP-Core hardware model design has to be completed first; The PRINCE algorithm is designed and implemented on FPGA. There is difference stages to completed System-on-Chips design, start from synthesize VHDL code, map, place and route, simulation, created IP-Core and then real time execution. Figure 3, shows the top level design for the data flow overview.

Figure 4, shows the interface of the block cipher in our design. The design involves of three ports, two ports for input the plaintext 64-bit and the key 128-bit. While, the third interface port represent the cipher text 64-bit output. The total numbers of the pins that used as input/output is 256 pins; therefore large FPGA boards

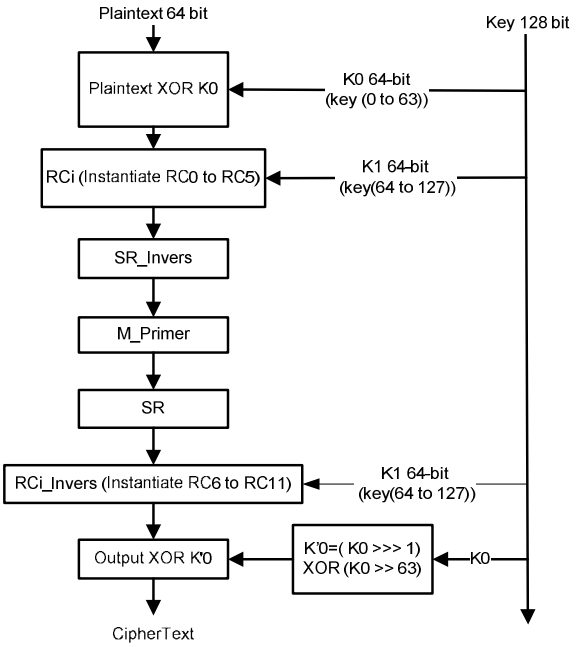

Fig. 3: The data flow of the PRINCE encryption unit

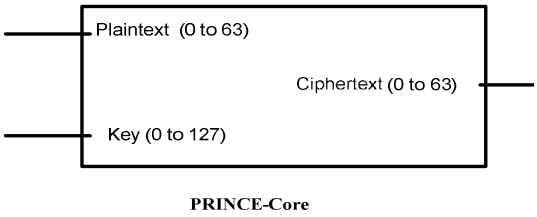

Fig. 4: The top module interface of the PRINCE

```
Entity K0dash is
Port
( In_K0_D : in  STD_LOGIC_VECTOR (0 to 63);
Out_K0_D : out  STD_LOGIC_VECTOR (0 to 63));
End K0dash;
Architecture Behavioral of K0dash is
Begin
Out_K0_D(0) <= In_K0_D(63);
Out_K0_D(1 to 62) <= In_K0_D(2 to 63);
Out_K0_D(63) <= In_K0_D(1) XOR In_K0_D(0);
        End Behavioral;
```

Fig. 5: The key schedule to find K0'

Virtex-403 are selected, because these boards have a big number of input/output pins as well as a large number of logic resources.

The main target of our design is low latency and low cost hardware implementation. During our model design, the lowest possible gate is investigated to get a low latency hardware component of PRINCE algorithm. PRINCE-Core consisted of many components such as: XOR 64-bit, S-Box Layer, M layer, M' layer, RC constant and Key schedule. The most expensive operation is S-box layer. In this design block RAM has not been used, instead all the components have been built by using Look-Up-Table (LUT) only. Therefore, the number of LUT is 32 slices for Virtex-403 resources.

As for the design of key schedule component with 63 shifts right operation, typically requires a long execution duration and large size area. Our approach for this key schedule is to create a custom circuit in the form of simple wiring route, with routing depending on the position of the input/output bits locations after and before the shift operation. This routing circuit requires only one slice of FPGA area of Virtex-403. Figure 5, shows the VHDL code for simple wiring route of key schedule entity or shift operation.

The concurrent model design with a short time delay, allows block cipher to encrypt the input data in all hardware components, within one clock cycle. The low area modification in our hardware architectures produces a great performance in terms of maximum frequency, throughput and occupied slices.

The PRINCE decryption unit is almost similar to encryption design. The data flow of decryption is shown in Fig. 6. The decryption unit requires the last round key in encryption unit to be the input key for the first round of decryption. These decryption keys are generated by XOR operation: key (K_1) with α or RC_{11} to get the new key (K_1) that to be fed in to the decryption unit.

PRINCE IP-core: In order to test the hardware-implemented PRINCE in a real environment set-up, bus slave interface with registers and address decoding is attached to the designed PRINCE entity to form a complete IP core. The bus slave interface allows the

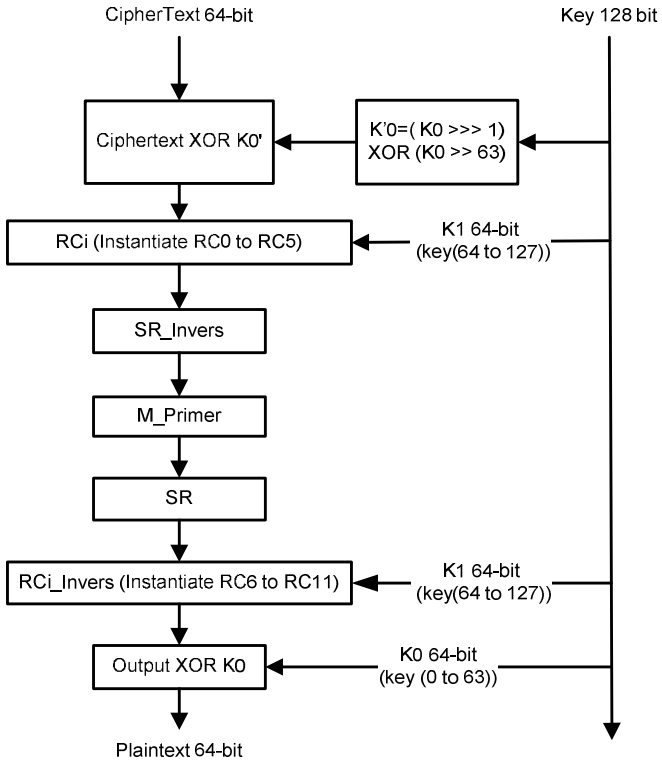

Fig. 6: The data flow of the PRINCE decryption unit

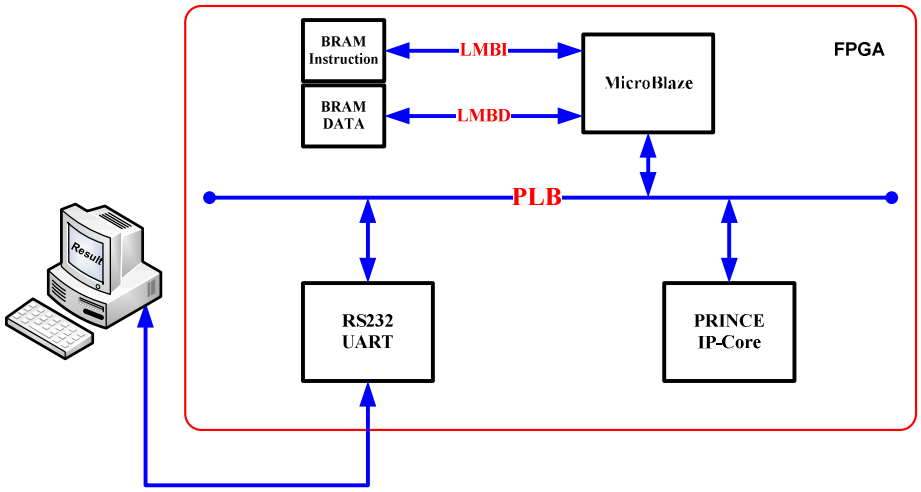

Fig. 7: Microprocessor (micro blaze processor) with PRINCE IP-core

PRINCE entity to receive and sent data from/to microprocessor via the processor local buses. This PRINCE IP has been tested within a complete microprocessor system realized on an FPGA chip, using XILINX XPS embedded system development tool. A complete microprocessor system consists of instantiated IP cores of Micro-Blaze soft-core, Block Random Access Memory (BRAM) and RS232 Asynchronous Receiver/Transmitter (UART). A test program written in C has been used to test the IP core. The UART is being used to connect to a terminal that to display messages of test results. Figure 7, shows the microprocessor system created on XILINX FPGA to test the PRINCE IP-Core. The test program has been written to deliver messages and keys to the PRINCE IP core and to read and display the produced cipher-texts on screen of terminal.

We have adopted the following steps to develop and test the IP core. First we use the ISE tools to develop the PRINCE encryption and decryption entities. After that

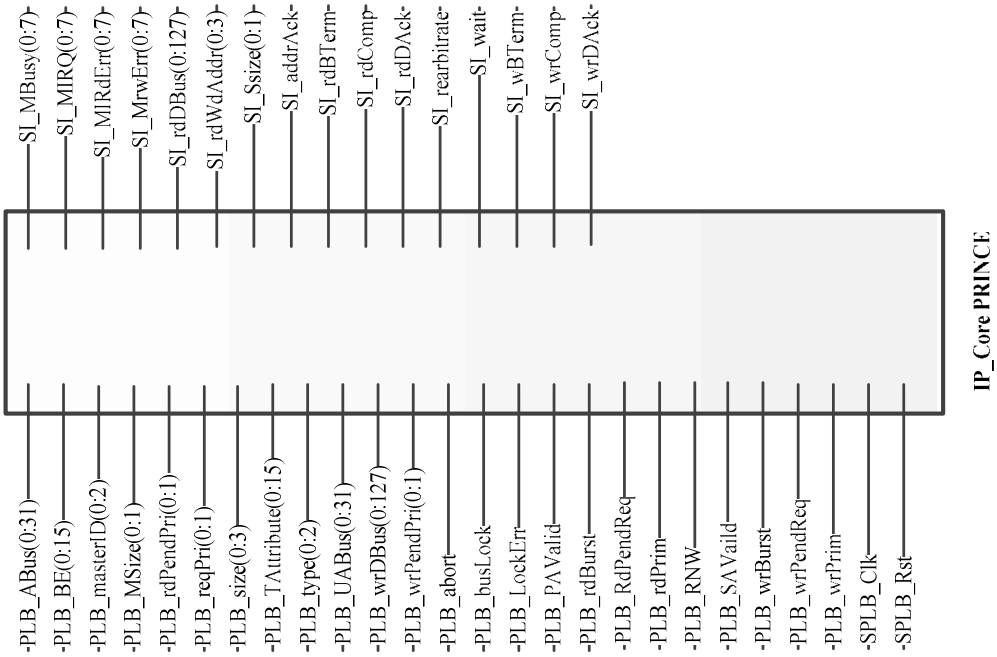

Fig. 8: RTL schematic for PRINCE IP-core

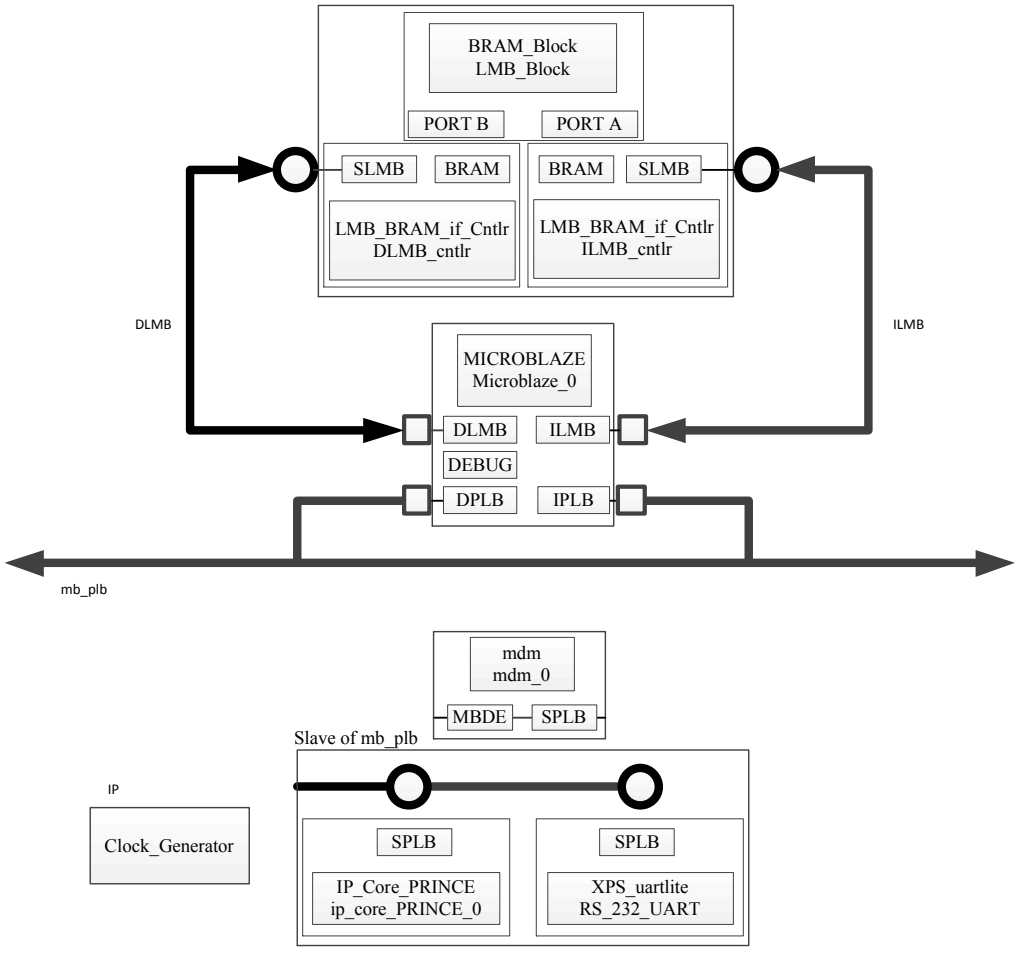

Fig. 9: The architectures of micro-blaze with IP-core

simulation tools has been used to test both of these entities. Subsequently XPS has been used to instantiate a "new slave-type" IP core. Then this new slave-type IP core is added with the encryption and decryption Entities (ISE) that have been developed earlier. In the next step the IP-core with attached slave interface has been tested to verify its syntax and then synthesized into bit-stream and download onto an FPGA chip. The RTL schematic for PRINCE IP-Core with ISE is depicted in Fig. 8.

Figure 9, shows XPS with IP cores of PRINCE, Micro-Blaze, Memory and UART that are interconnected via system bus (processor local bus). Except for Micro-Blaze, each IP core has its own addresses of 32-bit of system memory map. XPS enable the whole complete microprocessor system to be synthesized and implemented to produce bit-stream that can be downloaded onto FPGA. This on-chip micro-processor design is then to be transferred to another tool called System Development Kits (SDK) to prepare for test suite. The test suite is to be written in "C" program.

The steps needed within the SDK consist of hardware and software phases. Firstly the hardware design will be downloading onto the FPGA platform. The second phase is to prepare test program that can be written in C or C++. A simple C program is to assign the value of the 64-bit plaintext and the 128-bit key; therefore 6 registers with 32 bit have been used. The result produced from the cipher operation of the IP-Core requires another two 32 bits register as a ciphertext. Also the real time execution using this kind of hardware/software set-up has been performed for this design.

RESULTS AND DISCUSSION

The section is devoted to discuss the simulation results as well the real time validation. Both of the encryption and decryption functions have been coded in VHDL targeted for the XILINX Virtex-403 XC4VFX12 (Package FF668 with speed grade -10) FPGA. As for the synthesis and simulation tools,

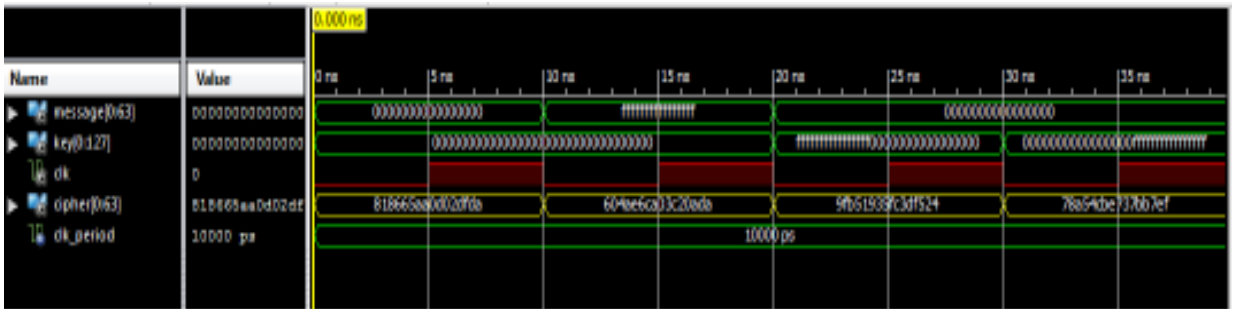

Fig. 10: The simulation of PRINCE core

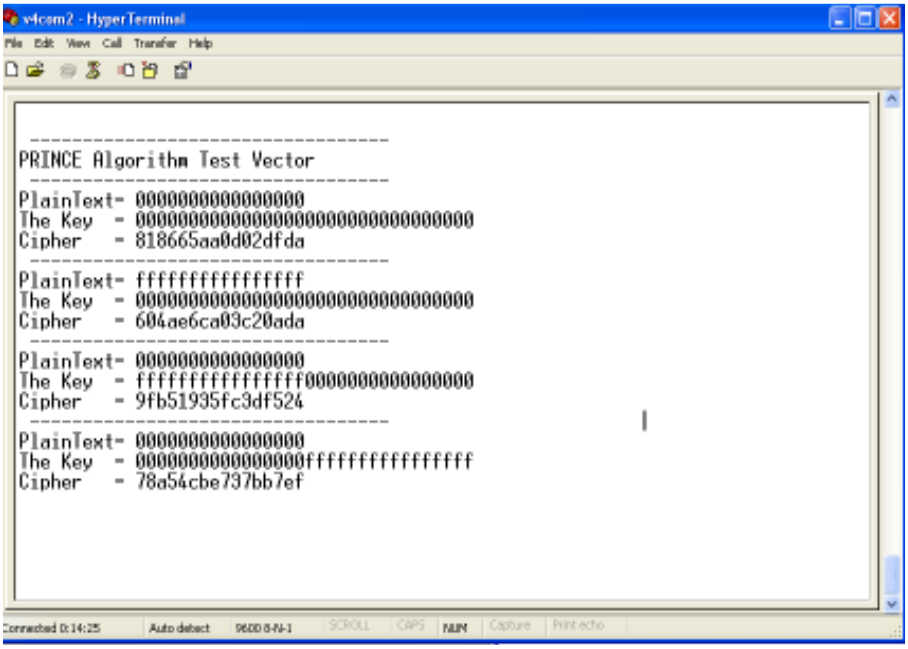

Fig. 11: Real-time hardware/software execution

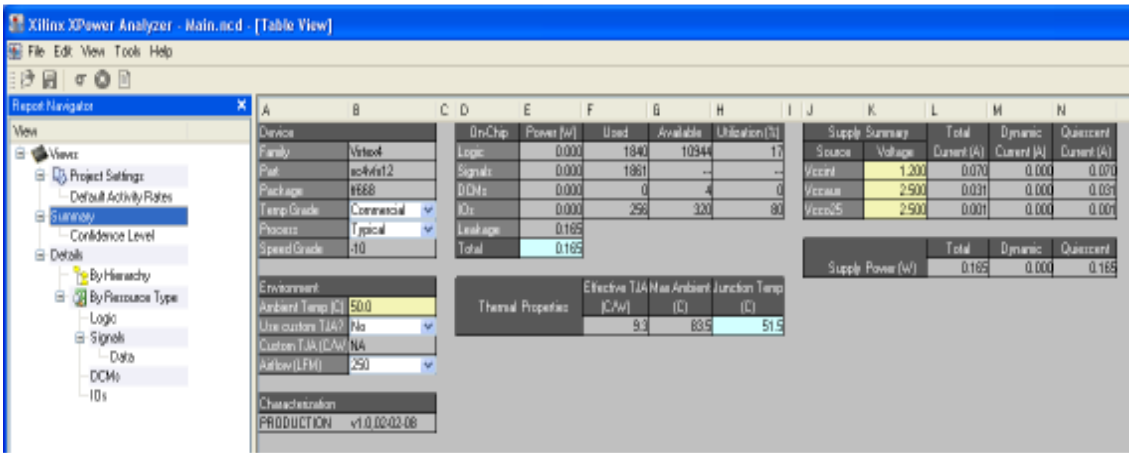

Fig. 12: Xilinx X power analysis for PRINCE IP-core using virtex-403

Table 2: Performance comparison of prince, present, SEA, ICEBERG and AES FPGA implementation

Algorithm	Block size	Device	Clock cycle	Max freq. (MHz)	Throughput (Mbps)	Total slices	Efficiency (Mbps/slice)	Power (watt)	Delay (ns)
PRINCE-128	64	Virtex-4FF668	1	31	2032	956	2.126	0.165	31.48
AES (Rouvroy *et al*., 2004)	128	Spartan-II XC2V40-6	32	123	358	1214	0.290	-	-
AES (Gielata *et al*., 2008)	128	Virtex4-4vlx200-10	11	165	-	77	-	-	-
AES (Chodowiec and Gaj, 2003)	128	Spartan-II XC2S30-6	32	60	166	522	0.320	-	-
SEA-128 (Mace *et al*., 2008)	128	Virtex-II XC2V4000	32	145	156	424	0.368	-	-
PRESENT-80 (Guo *et al*. 2008)	64	Spartan-IIIE XC3S500	32	-	-	271	-	-	-
ICEBERG (Standaert *et al*., 2007)	64	Virtex-II	32	-	1016	631	1.610	-	-
PRESENT-80 (Sbeiti *et al*., 2009)	64	Spartan-III XCS400-5	32	254	508	202	2.510	-	-

we have used Xilinx ISE V14.5 WebPack and ModelSimXE P.58f.

As for test simulation test vector, 64-bit hexadecimal plaintexts (0x0000000000000000 and 0xFFFFFFFFFFFFFFFF) and 128-bit key (0x000 ….. 0000, 0xFF….000, 0x000….FFF) in addition to other texts are fed into the hardware model. The cipher text 64-bit output is produced in one clock cycle. Figure 10, shows the test vector for some inputs and outputs in simulation graph.

Figure 11, shows the real time test of PRINCE IP core with the test suite program executing on Micro-Blaze processor, with test result of encrypting data being displayed on a hyper-terminal. The hyper-terminal executes on a PC that its serial port connected to the on-chip UART, in this case has been used as display or human interface. The hyper-terminal displays test vectors with cipher-texts have been generated by the PRINCE IP core as device on test.

The hardware IP-Core design of encryption and decryption model has low power consumption since it is of a small area design; the number of occupied slices used with Virtex-403 is 956 slices. The total power for proposed designed is 0.160 Watts. Figure 12, shows the power results using X-Power analysis tools of Xilinx ISE V14.5 WebPACK.

Several FPGA implementations of ASE and lightweight block ciphers have been evaluated for performance comparison. These different models have been implemented either for optimization the area and power consumption, or maximize data throughput by other related research projects. In addition, the proposed hardware design has been evaluated against other several existing FPGA implementation, utilizing different algorithms, namely, Scalable Encryption Algorithm (SEA) (Mace *et al*., 2008), ICEBERG algorithm (Standaert *et al*., 2007) PRESNT algorithms (Sbeiti *et al*., 2009). Our results, show high throughput, low power consumption and good efficiency with normal frequency speed as depicted in Table 2.

It is very important to note, that most AES implementations required a block RAM unit. While, our model design is working within one clock cycle and requires no block RAM unit. Furthermore, the proposed architecture is a concurrent design with acceptable number of slices; also with low power consumption.

There are different packages for each FPGA board with variety numbers of slices, LUT and input/output pin interface. This diversity makes the comparison even harder. Most AES implemented using Spartan family, whereas, SEA, ICEBERG and our design are implemented with Xilinx Virtex family. However, their execution performances have been with more than 10 clock cycle to produce output of the encryption cipher-texts.

PRINCE IP-core architecture has been designed to encrypt or decrypt the input data within one clock cycle, while having low latency and high speed. The investigations of PRINCE IP-CORE with Virtex-403 FPGA board indicates a high speed 31.765 MHz, throughput of 2.032 Gbps, low power consumption 0.165 and efficiency of 2.126 Mbps/slice.

CONCLUSION

This study presents the hardware and simulation of PRINCE algorithm on FPGA. The simulation and implementation includes VHDL code synthesis, mapping and simulation test suite. Meanwhile, the real-time hardware design evaluation platform consists of Micro-Blaze, PRINCE IP core and other IP cores. Concurrent design has been adopted to implement the PRINCE algorithm block cypher in FPGA to reduce clock cycle delay while to minimize FPGA area. The IP-Core hardware architecture has been designed to encrypt the 64 bit block cipher within one clock cycle.

This new PRINCE IP design that based on Virtex-403 FPGA platform produces 2.03 Gbps of throughput and consumes merely 0.165 W of power. It also has efficiency and latency of 2.126 Mbps and 31.48 ns, respectively. The design has produced promising results when compared to several other designs that its parallel architecture provides higher throughput, low power consumption and high efficiency. Notably, this low cost IP-Core PRINCE algorithm has the potential to be applied in a smart card and wireless devices.

ACKNOWLEDGMENT

This research is supported by Ministry of Education, Malaysia under Fundamental and Prototype Research Grant Scheme Phase 1/2013.

REFERENCES

Borghoff, J., A. Canteaut, T. Guneysu, E.B. Kavun, M. Knezevic, L.R. Knudsen, G. Leande *et al*., 2012. PRINCE-a low-latency block cipher for pervasive computing applications. In: Wang, X. and K. Sako (Eds.): ASIACRYPT 2012. LNCS 7658, International Association for Cryptologic Research, Springer, Berlin, Heidelberg, pp: 208-255.

Chodowiec, P. and K. Gaj, 2003. Very compact FPGA implementation of the AES algorithm. In: Walter, C.D. *et al*. (Eds.), CHES 2003. LNCS 2797, Springer-Verlag, Berlin, Heidelberg, pp: 319-33.

Daernen, J. and V. Rijmen, 2002. The Design of Rijndael, AES-the Advanced Encryption Standard. Springer-Verlag, Berlin, Heidelberg, pp: 238.

Deshpande, A.M., M.S. Deshpande and D.N. Kayatanavar, 2009. FPGA implementation of AES encryption and decryption. Proceeding of International Conference on Control, Automation, Communication and Energy Conservation (INCACEC, 2009), pp: 1-6.

Doröz, Y., A. Shahverdi, T. Eisenbarth and B. Sunar, 2014. Toward practical homomorphic evaluation of block ciphers using prince. In: Bohme, R. *et al*. (Eds.), FC 2014 Workshops. LNCS 8438, Springer-Verlag, Berlin, Heidelberg, pp: 208-220.

Gielata, A., P. Russek and K. Wiatr, 2008. AES hardware implementation in FPGA for algorithm acceleration purpose. Proceeding of International Conference on Signals and Electronic Systems (ICSES'08), pp: 137-140.

Grabher, P., J. Großsch and D. Page, 2008. Light-weight instruction set extensions for bit-sliced cryptography. In: Oswald, E. and P. Rohatgi (Eds.), CHES 2008. LNCS 5154, Springer-Verlag, Berlin, Heidelberg, pp: 331-345.

Guo, X., Z. Chen and P. Schaumont, 2008. Energy and performance evaluation of an FPGA-based SoC platform with AES and PRESENT coprocessors. In: Berekovic, M., N. Dimopoulos and S. Wong (Eds.), SAMOS 2008. LNCS 5114, Springer-Verlag, Berlin, Heidelberg, pp: 106-115.

Mace, F., E.X. Standaert and J.J. Quisquater, 2008. FPGA Implementation(s) of a scalable encryption algorithm. IEEE T. VLSI Syst., 16(2): 212-216.

Rolfes, C., A. Poschmann, G. Leander and C. Paar, 2008. Ultra-lightweight implementations for smart devices-security for 1000 gate equivalents. In: Grimaud, G. and F.X. Standaert (Eds.), CARDIS 2008. LNCS 5189, Springer-Verlag, Berlin, Heidelberg, pp: 89-103.

Rouvroy, G., F.X. Standaert, J.J. Quisquater and J.D. Legat, 2004. Compact and efficient encryption/decryption module for FPGA implementation O the AES rijndael very well suited for small embedded applications. Proceeding of the International Conference on Information Technology: Coding and Computing, (ITCC, 2004), pp: 583-587.

Sbeiti, M., M. Silbermann, A. Poschmann and C. Paar, 2009. Design space exploration of present implementations for FPGAS. Proceeding of the SPL 5th Southern Conference on Programmable Logic, pp: 141-145.

Standaert, F.X., G. Piret and G. Rouvroy and J.J. Quisquater, 2007. FPGA implementations of the ICEBERG block cipher franc. Integr. VLSI J., 8: 20-27.

Xilinx EDK V14.5, 2013. Embedded System Tools Reference Manual. Xilinx. Vol. 111.