Journal of Engineering and Applied Sciences 13 (9): 2638-2646, 2018
ISSN: 1816-949X

A Comparative Study of Smart Card Design with Memory Ciphering System on Arm-Based FPGA

[1]Wira Firdaus Yaakob, [2]Hazleen Aris and [1]Jahariah Sampe
[1]Institute of Microengineering and Nanoelectronics, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
[2]Institute of Informatics and Computing in Energy, Universiti Tenaga Nasional, 43000 Kajang, Selangor, Malaysia

Abstract: Memory ciphering system is a mechanism to secure the data in non-volatile memories of the system using standard encryption module and other security protections. Memory ciphering system in a smart card consists of three important units. Advanced Encryption Standard (AES) cipher, Random Number Generator (RNG) and Scrambler/Descrambler. It is built inside the Memory Management Processing Unit (MMPU) for securing data transactions with the smart card memories. This study presents the results of a comparative study performed between Xilinx's and Intel's (previously Altera) Advanced RISC Machines (ARM) based Field Programmable Gate Array (FPGA) in prototyping smart card design with memory ciphering system. The smart card design is implemented in Xilinx's Zynq-7000 XC7020-1-CLG484 and Intel's Cyclone V System-on-a-Chip (SoC) 5 CSEMA 5 F 31 C6N devices. The objective of this study is to identify the optimum FPGA platform for the prototype. The comparative study between the two ARM-based FPGA implementations is explained in terms of logics utilization and time requirements. The memory ciphering system in the smart card is capable to complete in 40 nsec that is a single CPU clock cycle of the smart card. Results obtained showed that the implementation in the Intel Cyclone V SoC has the least utilized logics and the highest maximum frequency which are 8.313 slices and 195 MHz, respectively. Since, the smart card prototyping in FPGA is the prerequisite for its Application Specific Integrated Circuit (ASIC) implementation, findings from this study serve as a good reference for enhancing secure smart card performance especially for logic optimization in ASIC.

Key words: Smart card, ARM-based FPGA ASIC, SoC, ID, implementation, results

INTRODUCTION

Smart card is an Integrated Circuit (IC)-based card that contains embedded ICs with microprocessor and internal memories or other intelligence logics with internal memory only. It can have direct contact with a smart card reader or remote contact using radio frequency interface (Mohammed *et al*., 2004). Smart card has made life a lot easier in many applications such as national identification, healthcare, entertainment and banking (Rossudowski *et al*., 2010). Figure 1 shows a typical smart card microcontroller design. It has high memory capacity and low power consumption. It provides 4 kB External Random Access Memory (XRAM) for fast read/write access. Other Non-Volatile Memories (NVMs) such as 64 KB Read-Only Memory (ROM) and 16 KB read/write Electrically Erasable Programmable Read-Only Memory (EEPROM) are also provided (Yaakob *et al*., 2014).

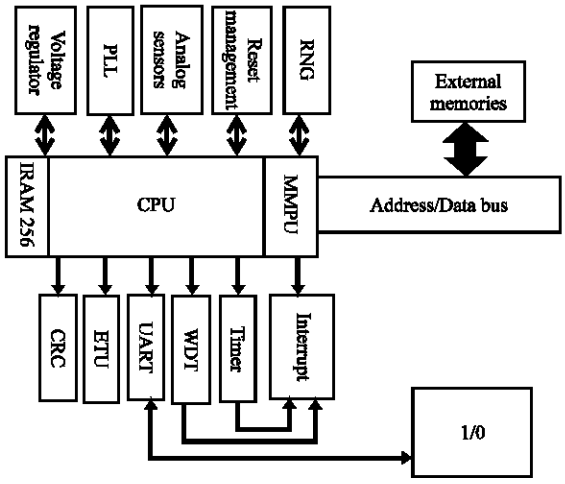

Fig. 1: Smart card design architecture

Due to its crucial application combined with the rise of more advanced attacks on smart cards, security issue

Corresponding Author: Wira Firdaus Yaakob, Institute of Microengineering and Nanoelectronics, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

is of major concern (Ren and Wu, 2013). One of the attack methods is 'sniffing' of transaction data between smart card reader and smart card NVM. Thus, it is very important to have a memory ciphering system that protects the data from being seen as plain text by attacker (Markantonakis *et al*., 2009). The memory ciphering system is a smart card security protection system at the circuit level. Other security layers that are incorporated together for smart card data protection are silicon, operating system and application levels (Guillou *et al*., 2001).

Most of the memory ciphering systems today are mainly proposed for hard disk encryption. For a smart card, the Integrated Circuit (IC) size is restricted and the area is usually dominated by the NVM modules. It is merely not practical to enlarge further the NVM size for more complex encryption and decryption process because on smart cards, NVM sizes are usually restricted. It is also not possible to embed high speed processing module in order to achieve high performance because it requires lots of memory storage and logic elements. Therefore, a method that is low-cost, security proven, high performance and less complexity is needed. In our previous research (Yaakob *et al*., 2017), a new memory ciphering system is proposed to achieve high security with rapid encryption and decryption processes without enlarging the NVM and embedding high speed processing module. Evaluation results showed that the system is capable to achieve throughput of 9085 (Mbits/sec) and 40 nsec ciphering time.

Other than security issues, smart card design industry is also very competitive. Rapid smart card design is essential to shorten time to market. Therefore, it is important to have an early smart card design prototype in Field Programmable Gate Array (FPGA). Prototype the design in FPGA also helps to optimize the smart card in the Application Specific Integrated Circuit (ASIC) implementation later on because it helps to emulate the design in real-time, early software development and early design debugging. Design optimization, especially in terms of logic area and design throughput, helps to reduce the design cost and enhance design performance (Jyrwa and Paily, 2009). As a result, the design can achieve fast time-to-market due to shortened development and debugging time with a very low cost and high performance.

In this research work, Xilinx and Intel FPGA platforms have been chosen for the smart card prototyping. The two platforms are two well-known key players in the FPGA industry (Druyer *et al*., 2015). The system is developed using Verilog Hardware Description Language (HDL) and embedded inside the Memory Management Processing Unit (MMPU). The functionality of the HDL design were simulated using Mode lsim TM. Then, the design are synthesized, placed and routed using Vivado TM for Xilinx flow and Quartus 2 TM for Intel FPGA flow. Finally, the implemented design are downloaded into Intel FPGA and Xilinx FPGA devices for real-time testing with the software applications. Results of the FPGA implementations in the Xilinx and Intel FPGA are being compared in terms of resource utilization and time requirements. In our experiment, it was found that the design implementation in the Intel FPGA gave the best result with the least resource utilization and the shortest time requirements.

MATERIALS AND METHODS

The proposed memory ciphering system of the smart card as illustrated in Fig. 2, comprises three major units, Random Number Generator (RNG), Advanced Encryption/Decryption System (AES) and Scrambler/Descrambler.

It is developed in the MMPU of the smart card between the CPU and the NVMs. The system is designed to complete the whole ciphering process within a single CPU clock cycle that is 40 nsec. RNG unit prototyped in the FPGA device is a Pseudo RNG (PRNG) (Oksar *et al*., 2011). Its function is to generate random keys for the AES cipher unit (Messerges *et al*., 2002). There are 4 steps in the PRNG issuance flow as indicated in Fig. 3. There are also two Linear Feedback Shift Registers (LFSR) that reside in the RNG unit. The RNG unit provides 128 bits random number on each warm or cold reset or whenever seed generation instruction is driven by the software.

The second main unit is the AES-128 cipher as shown in Fig. 4. The unit is based on the AES Rijndael algorithm and is compliant to Federal Information Processing Standard Publication (FIPS PUB) 197 (Chi-Feng *et al*., 2003). It is a symmetric block cipher that encrypts plaintext. It has nine rounds of transformation for 128-bit key. Each round comprises the operations byte substitution, shift row, mix column and key addition.

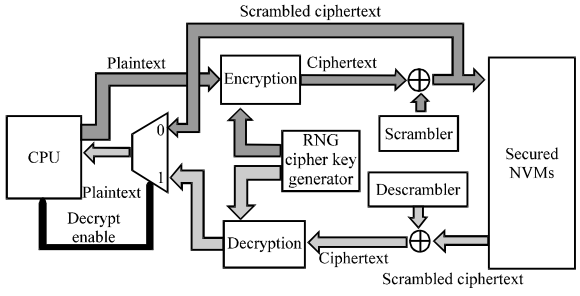

Fig. 2: The proposed smart card memory ciphering system

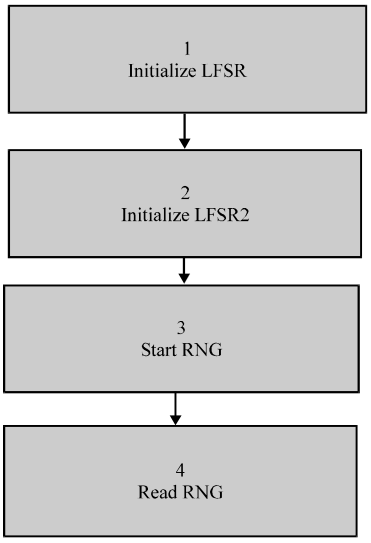

Fig. 3: RNG flow

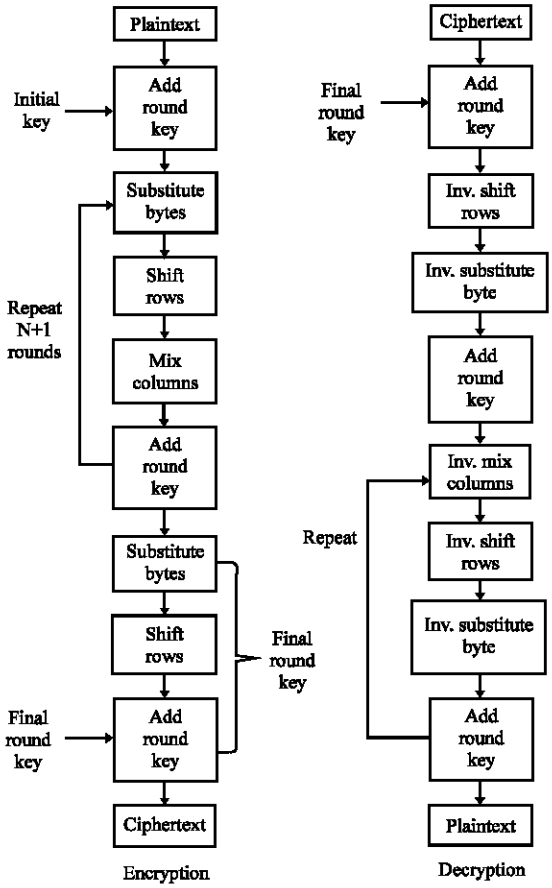

Fig. 4: AES-128 Rijndael algorithm

Byte substitution is the operation that runs independently on each byte of the state using a substitution table thatis called as S-box. Shift row operation runs by shifting bytes of the last three rows of the state cyclically while the first row is not shifted. Mix column operation is the transformation process that runs on the state, column-by-column.

Every column is treated to be a four-term polynomial. The final operation for each round that is key addition, operates by adding the round key to the state by a simple bitwise OR. Other than key size 128 bits that is used in this study, Rijndael also supports for 192 and 256 bits key to encrypt plaintext blocks that are 192 bit and 256 bit, respectively.

The third main unit is the scrambler/descrambler. The scrambling process helps to complicate the AES-128 cipher by mixing a cipher text from the AES-128 with scrambled data. The scrambled data is a block 128 bit data that relies on the user input PIN. The descrambling is the reverse of the scrambling process. Without the correct user input PIN keys data from NVMs are not allowed to be descrambled back into cipher text. Thus, the secured data cannot be decrypted back by the AES-128 cipher into the plaintext.

For smart card functional verification, a general purpose Operating System (OS) that is conforming to the ISO 7816 standard has been developed and programmed into the program memory. The key features of the OS are such as its IO contact is following the ISO7816-3 T = 01, its file system management based on ISO7816-4, flexible system configuration, a centralized anti-tearing mechanism and several commands can be linked in an all or-nothing transaction session. At the early communication stage of the smart card with a host, the smart card appears operative by transmitting an Answer to Reset (ATR) data to the host.

Several basic OS commands are used for functional verification such as select file, get challenge/get random, get data, put data, directory info, create file, update binary, read binary, read record, append record and increase/decrease. The OS commands follow the ISO7816 4 interindustry commands for interchange. Descriptions of the OS commands used are provided.

Select file command sets a current file within a logical channel. The following commands may implicitly apply to this current file. The get challenge command triggers a challenge issuing by the smart card for use in a procedure that relates to security. The smart card responses with an 8-byte random number and holds this number for use in a following mutual authenticate command. The get random command is alike but that generated random value is neither restricted to 8 bytes nor it is retained by the ICC.

It is able to be used by an interface device for random number generationcapabilities. The get data command sends the value of a system parameter that is defined in class and sub-class.

The directory info command returns information about a file and its sub-files. The create file of Master File (MF) command initiates the creation of a file. When creating an MF it results in the initialization of the file system. When creating a Dedicated File (DF) or Elementary File (EF), the file is placed immediately under Current DF (or MF). Once, the command is successfully completed, the created file is turned to be the current file. The update binary command triggers the update of bits that are already present in an EF with the given bits in the Application Protocol Data Unit (APDU) command. The EF must be of type transparent. When the command has a Short File Identifier (SFI) it selects the file implicitly and sets it as current EF upon successful.

The read binary command responses (all or portion of) the EF content with a transparent structure. The read record command returns the contents of the specified record of an EF which must be of type linear fixed, cyclic or purse. Note that it is not required to read the full record length. Records are always read starting at off set 0. The append record command initiates the setting of record number 1 in an EF cyclic construct there by overwriting the oldest record.

The increase/decrease command only applies to an EF of type purse. The command initiates a subtraction (addition) of the input amount from the value held in (logical) record 1 and the resulting value is appended to the file. On successful completion, the new value is held in (logical) record 1 together with any additional transaction data. When the number of bytes to be updated is larger than the record size, the command returns 6C XXh where XX equals the record size.

For the FPGA realization, the smart card design with the memory ciphering system has been implemented in Xilinx's Zynq-7000 FPGA (XC7020-1-CLG484) and Intel's Cyclone V SoC (5CSEMA5F31C6N). Vivado software has been used to implement the smart design in the ARM based Xilinx FPGA device. Its design flow is illustrated in Fig. 5. On the other hand, Quartus II software has been used for the implementation on the Intel's ARM-based FPGA device. The quartus 2 design flow is shown in Fig. 6 (Intel, 2010).

Design implementation using xilinx 's zynq -7000: The smart card design logics including the analogue and memory interfaces are developed in Verilog Hardware Description Language (HDL). The constraints and pin assignments have been set after the successful

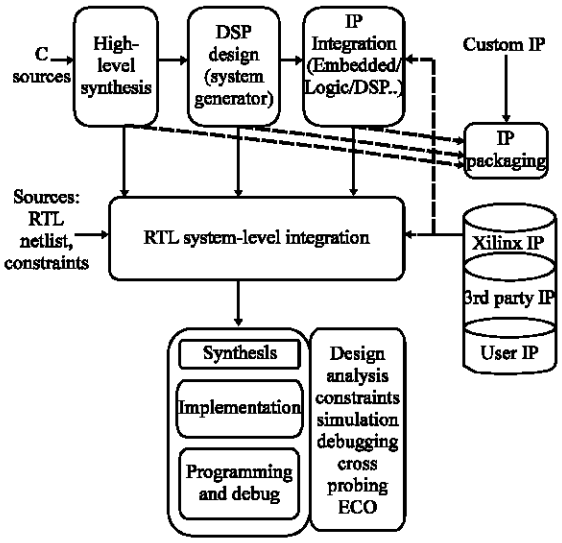

Fig. 5: Vivado high-level design flow

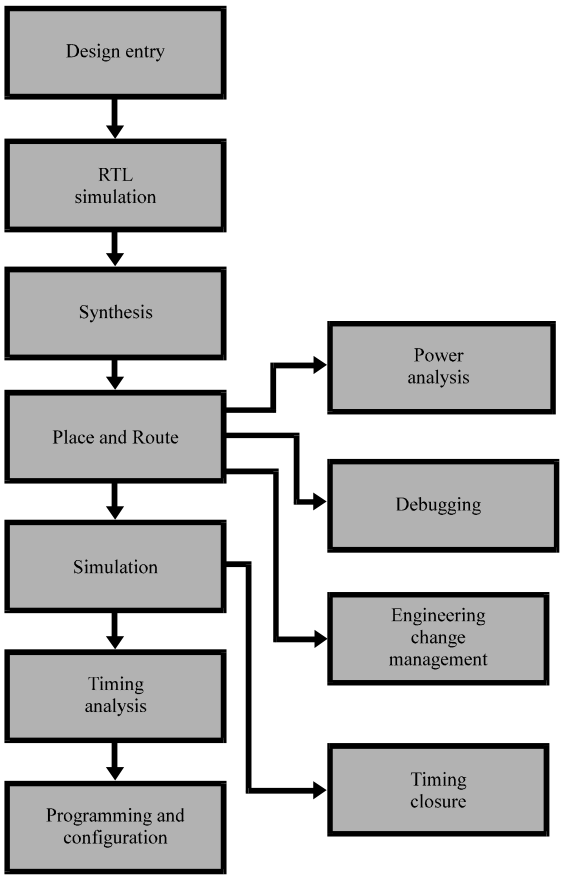

Fig. 6: Quartus 2 design flow

functional verification. The resource utilization summary of the smart card design is shown in Table 1. Its statistic

Table 1: Post-implementation resource utilization report of the smart card design that is provided by the Xilinx

Resource	Utilization	Available	Utilization (%)
Slice LUT	10.927	53.200	20.54
Slice register (FF)	2.484	106.400	2.33
BRAM	37.5	140	26.79
DSP48	3	220	1.36

Table 2: Resource utilization summary for the smart card design implementation on intel cyclone V

Resource	Utilization	Available	Utilization (%)
Slice LUT	8,313 (3,260 ALM)	81,779 (32,070 ALM)	10.17
Slice register (FF)	2,533	128,280	1.97
M10K blocks	120	397	30.22
DSP blocks	2	87	2.30

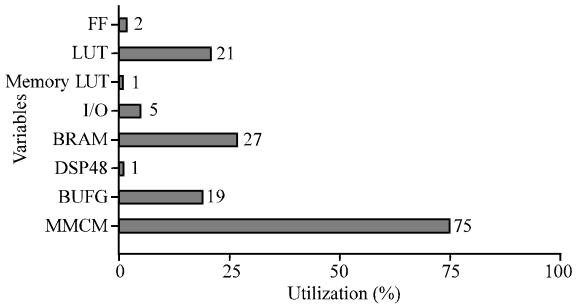

Fig. 7: Post-implementation resource utilization report of the smart card design that is provided by the Vivado Software

result is shown in the Fig. 7. The implementation that includes the Register Transfer Level (RTL) analysis, synthesis, translate, map and place and route processes have been processed, successfully. The operating system is converted from hex to Xilinx Coefficient (COE) file format before being included in the program memory of the smart card. The corresponding programming file has been generated and programmed into the FPGA XC7020-1-CLG484 device.

Design implementation using Intel's Cyclone V SoC: The similar smart card system has been implemented in the intel cyclone V SoC, 5CSEMA6F31C6 device. The functional simulation is done using Modelsim simulator.

The OS file is converted into intel memory Initialization File (MIF) file format before being inserted in the program memory. The compilation processes include analysis and synthesis, fitting (place and route), timing analysis and EDA netlist writing. Its resource usage summary is shown in Table 2.

RESULTS AND DISCUSSION

The comparison results of the ARM-based FPGA implementations are shown in Table 3. The Adaptive Logic Module (ALM) utilization percentage in the intel FPGA is much less than Look-Up Table (LUT) utilization in the Xilinx FPGA. The difference is about 10.37%. This means that the logic slices are reduced twice in the Intel FPGA fabric. This is due to the fact that in ALM architecture less area is utilized for more combinational logic thus, gaining a higher logic density than a standard four-input LUT (Intel, 2005). Equation 1 is the conversion formulae from ALM into LUT slice, normalized relative to logic capacity for these types of FPGA devices. The equation means that the logic capacity of one Intel Cyclone V SoC ALM is equivalent to that of 2.55 Xilinx Zynq SoC slices. From Table 3, too it can be seen that the percentage of utilized registers which are mainly flip-flops in the intel FPGA is 0.36%> the Xilinx FPGA.

Table 3: Comparison of the smart card design 's resources between Xilinx Zynq SoC and Intel Cyclone V SoC

Resource	Xilinx Zynq SoC (xc7z020-1clg484)	Intel Cyclone V SoC (5CSEMA6F31C6)
Slice LUT/ALM (%)	20.54	10.17
Slice register (%) (FF)	2.33	1.97
BRAM/M10K (%)	37.5	30.22
Max. Freq (Fmax) (MHz)	104	195
Combinational path delay	10.81	13.74

$$\text{Number of LUT slices} = 2.55*\text{ALM} \quad (1)$$

The intel FPGA is also less than the Xilinx FPGA in terms of memory block utilization that is 7.28%. This means that the memory prototyping for the smart card design is more optimized in the intel FPGA than the Xilinx FPGA. The intel FPGA utilizes configurable M10K block for memory prototyping while the Xilinx FPGA utilizes configurable Block Random Access Memory (BRAM). The FMax of ARM-based Intel FPGA is 91 MHz greater than Xilinx's ARM-based FPGA. The FMax is a clock frequency that is achieved without Internal Set Up (TSU) and hold (TH) time violations. On the other hand, the Xilinx FPGA's worst path delay is 2.93 nsec less than the Intel FPGA. The path delay in the Intel FPGA implementation is higher than the Xilinx FPGA due to the longer routing path.

The memory ciphering simulation results using the software Modelsim are exhibited in Fig. 8 and 9. Figure 8 shows the total time taken for one cycle of the memory ciphering system is about a single CPU clock cycle that is 40 nsec. The rapid ciphering system helps to improve the security of the stored data in the smart card memories without incurring very much delay. Figure 9 illustrates the successful decryption at the end of the memory ciphering system cycle to get back the original plaintext. The process of securing the original plaintext has been described in Fig. 2, previously. For basic OS commands verification in the smart card, its functional simulation

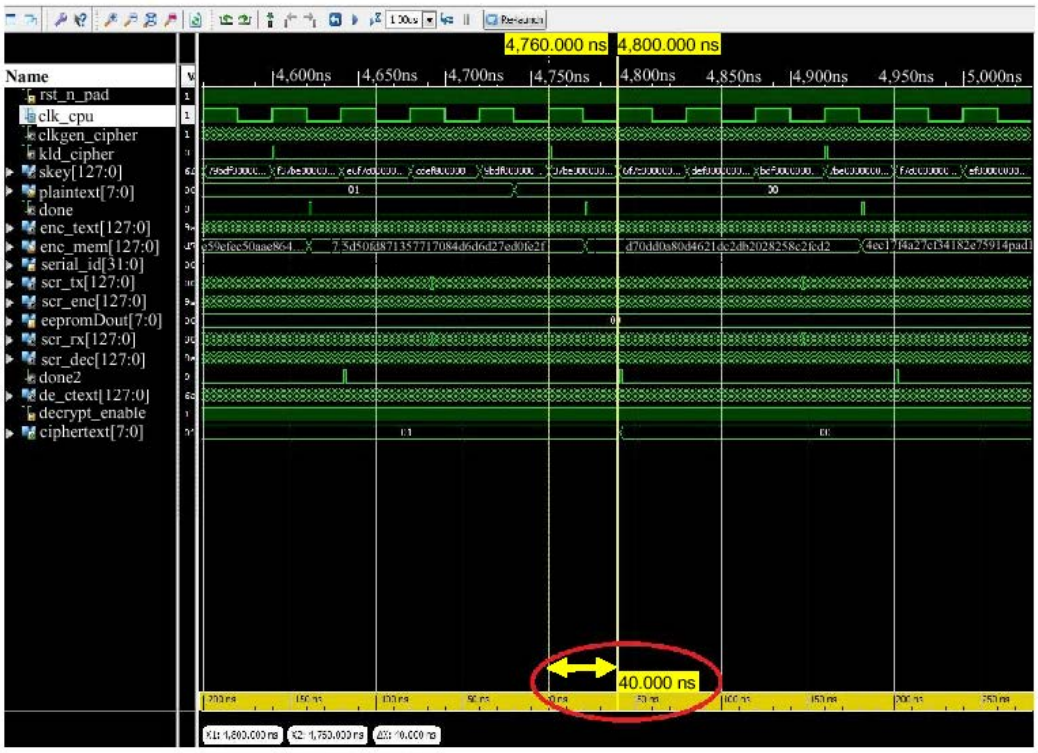

Fig. 8: The total ciphering time is a single CPU clock cycle, i.e., 40 nsec

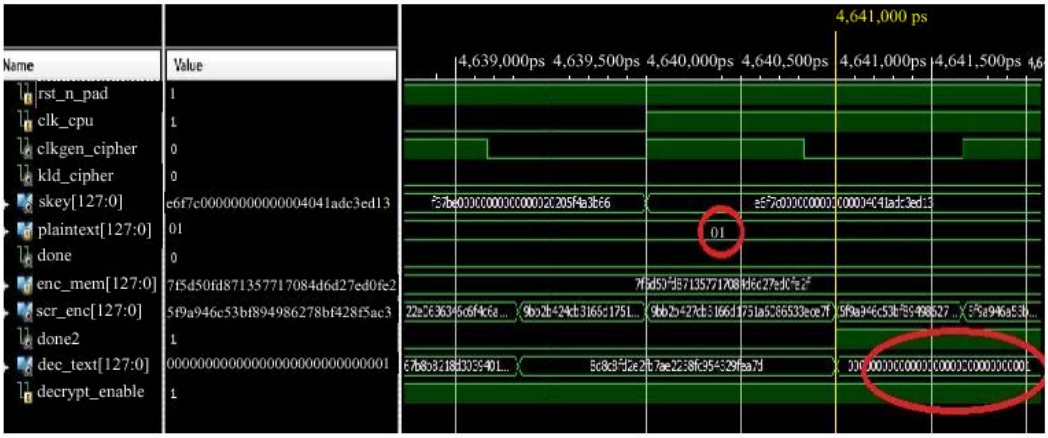

Fig. 9: Secured data is successfully decrypted to its plaintext value i.e. Hex 01

results are shown in Fig. 10 and 11. The OS commands are read using the memory ciphering system by the CPU from the smart card program memory.

A s can be seen from Fig. 10, the ATR reading result as shown in the figure's waveform and execution log at the bottom of the figure is as expected. The ATR reading verification is important in order to ensure the smart card is functioning. Without getting the ATR reading result, the rest of OS command verifications cannot be done (Choudary *et al*., 2012). The expected ATR reading result is in hexadecimal 3B 9618004 D79434 F 5370H. The ATR indicates also the operation characteristics of the smart card. Figure 11 shows the result of select MF and get response commands. Prior sending the get response command, the select MF command is sent first to the smart card. The select MF command is 00 A4 0000 h 02 3 F00H. As a result, the achieved get response result is as shown in the Fig. 11 that Is 6 F 0783023 F 00820138 H. The result is similar with the expected result that has been tested in real-time with a reader as shown in Fig. 12. The

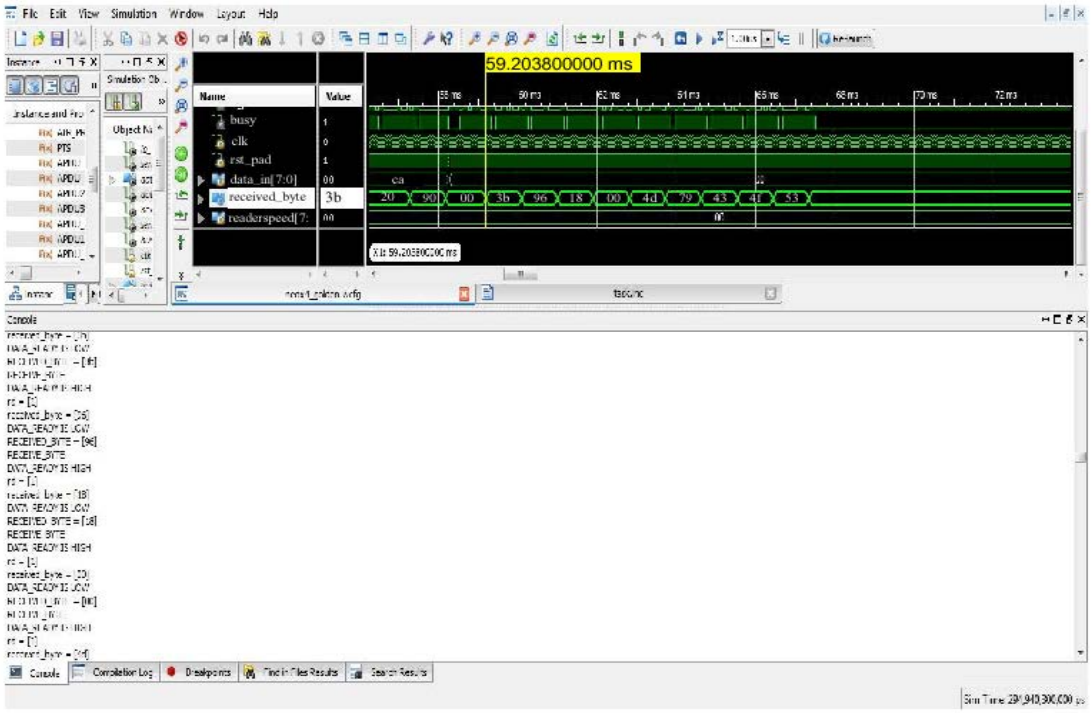

Fig. 10: Functional simulation result of the smart card's ATR reading command

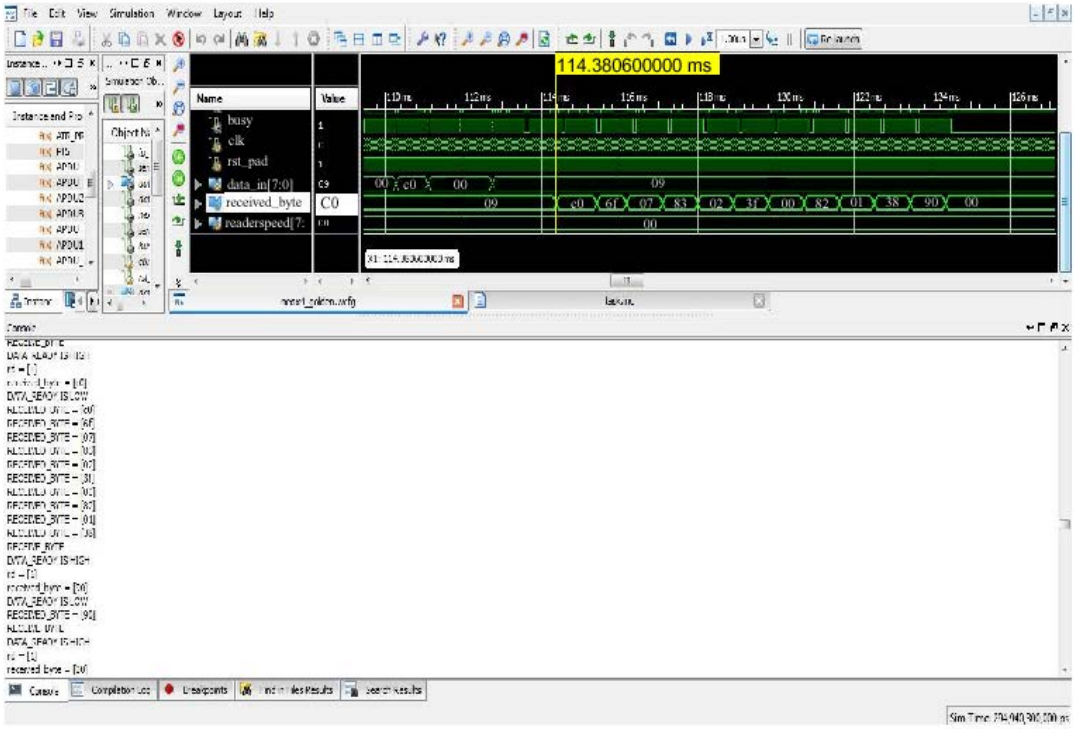

Fig. 11: Functional simulation result of the smart card's SELECT MF and GET RESPONSE commands

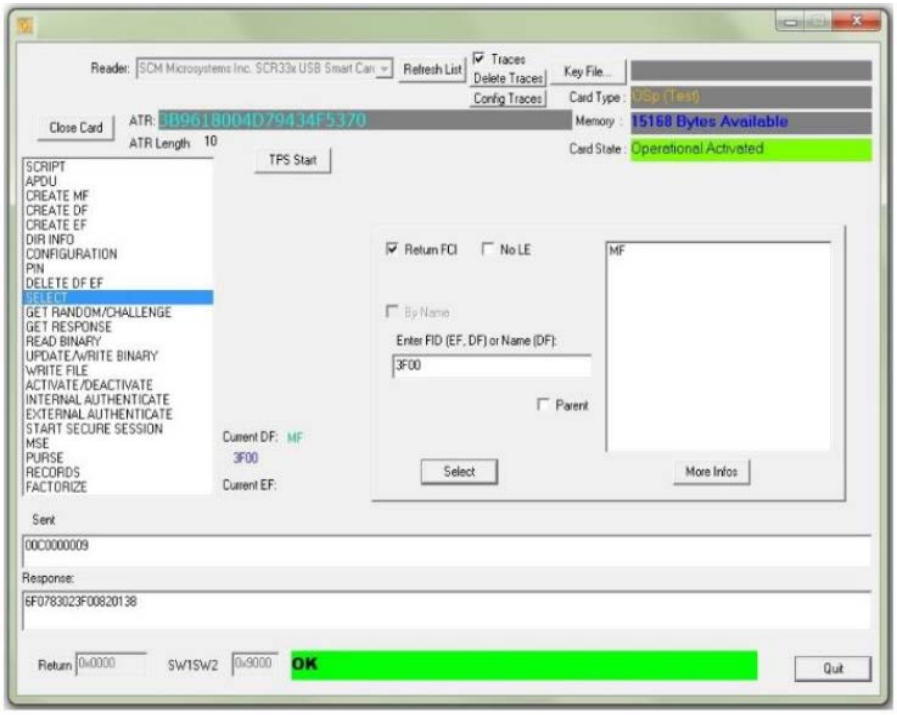

Fig. 12: Real-time testing with a reader of the ATR reading, SELECT MF and GET RESPONSE commands

APDU command of get response i s 00 C 000000 9 H.The Status Word (SW) 9000 H after the get response's result means there is no response error of the smart card. The SW definition is following the ISO 7816 standard.

CONCLUSION

Design of memory ciphering system for a smart card has been implemented on the ARM-based FPGA from the two leading FPGA industry players. The objective of the study is to identify the optimum FPGA platform for the secured smart card prototype between Xilinx's Zynq-7000 XC7020-1-CLG484 and Intel's Cyclone V SoC 5 CS EMA 5 F31 C6N. The smart card memory ciphering system is developed by using RNG key generator, AES-128 cipher and Scrambler/Descrambler. The smart card with the memory ciphering system has been designed in Verilog HDL and simulated using Vivado and Modelsim software. After the successful simulation, the smart card is implemented and programmed in the FPGA devices using Vivado v 2014. Software for Xilinx and Quartus 2 13.0.1 for Intel. Finally, a real-time testing with a host is done using a smart card reader and a smart card software application. From the FPGA implementation results in Table 3, it shows the smart card with the memory ciphering system on the ARM-based Intel platform has achieved the smallest area due to the least number of the total hardware used. The platform has also attained the highest maximum frequency that is the best time requirement. This finding can be a good reference for other smart card or SoC design implementations on the SoC-based type FPGA platform in achieving design optimization with high performance as it helps to optimize the logic implementation in ASIC for silicon realization.

RECOMMENDATION

Future research includes integrating the memory ciphering system with a 32 bit CPU and sensors for biometric features.

REFERENCES

Chi-Feng, L., K. Yan-Shun, C. Hsia-Ling and Y. Chung-Huang, 2003. Fast implementation of AES cryptographic algorithms in smart cards. Proceedings of the IEEE 37th Annual International Conference on Carnahan Security Technology, October 14-16, 2003, IEEE, Taipei, Taiwan, ISBN:0-7803-7882-2, pp: 573-579.

Choudary, O.S., 2012. The smart card detective: A hand-held EMV interceptor. Master Thesis, University of Cambridge, Cambridge, England.

Druyer, R., L. Torres, P. Benoit, P.V. Bonzom and P. Le-Quere, 2015. A survey on security features in modern FPGAs. Proceedings of the 2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), June 29-July 1, 2015, IEEE, Bremen, Germany, ISBN:978-1-4673-7942-7, pp: 1-8.

Guillou, L.C., M. Ugon and J.J. Quisquater, 2001. Cryptographic authentication protocols for smart cards. Comput. Networks, 36: 437-451.

Intel, 2005. Stratix II vs Virtex 4 density comparison. Intel, Santa Clara, California, USA.

Intel, 2010. Introduction to the quartus II software. Intel, Santa Clara, California, USA.

Jyrwa, B. and R. Paily, 2009. An area-throughput efficient FPGA implementation of the block cipher AES algorithm. Proceedings of the International Conference on Advances in Computing, Control and Telecommunication Technologies, December 28-29, 2009, Trivandrum, India, pp: 328-332.

Markantonakis, K., M. Tunstall, G. Hancke, I. Askoxylakis and K. Mayes, 2009. Attacking smart card systems: Theory and practice. Inf. Secur. Tech. Report, 14: 46-56.

Messerges, T.S., E.A. Dabbish and R.H. Sloan, 2002. Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Comput., 51: 541-552.

Mohammed, L.A., A.R. Ramli, V. Prakash and M.B. Daud, 2004. Smart card technology: Past, present and future. Intl. J. Comput. Internet Manage., 12: 12-22.

Oksar, M., B. Ors and G. Saldamli, 2011. System level design of a secure healthcare smart card system. Proceedings of the 2011 IEEE International Conference on Systems and Information Engineering Design Symposium (SIEDS), April 29-29, 2011, IEEE, Charlottesville, Virginia, USA., ISBN: 978-1-4577-0446-8, pp: 170-175.

Ren, Y. and L. Wu, 2013. Power analysis attacks on wireless sensor nodes using CPU smart card. Proceedings of the 2013 22nd International Conference on Wireless and Optical Communication (WOCC), May 16-18, 2013, IEEE, Chongqing, China, ISBN:978-1-4673-5699-2, pp: 665-670.

Rossudowski, A.M., H.S. Venter, J.H. Eloff and D.G. Kourie, 2010. A security privacy aware architecture and protocol for a single smart card used for multiple services. Comput. Secur., 29: 393-409.

Yaakob, W.F.H., H.H. Manab and S.N.M. Adzmi, 2014. Smart card chip design implementation on ARM processor-based FPGA. Proceedings of the IEEE 3rd Global Conference on Onsumer Electronics, October 7-10, 2014, Tokyo, Japan, pp: 294-297.