Design of a 128-bit AES Block Cipher Core

Jung Hwan Oh, Sang Muk Lee, Young Hyun Yoon, and Seung-Eun Lee[a]
Department of Electronic Engineering, Seoul National University of Science and Technology
E-mail : ohjunghwan@seoultech.ac.kr

***Abstract* - Advanced encryption standard (AES), most widely used to encrypt and decrypt data, is computationally intensive algorithm. In order to implement the AES encryption algorithm, the data must go through complicated operations composed of several steps. The AES encryption algorithm is generally used for internet-based applications. In this paper, we propose the AES core, which selects the operation mode for the block cipher. Also, we implement the AES core by using an field programmable gate array (FPGA) and verify the functionality of the AES core with prototype board. The implementation results are obtained using the AES block cipher processors.**

I. INTRODUCTION

As the Internet of Things (IoTs) and data transmissions based on wireless communications develop rapidly, the IoT networks and devices are interchanging data more explosively. However, the threats on network or devices such as password attacks, spying and hacking have been increased dramatically [1]. As a result, the data security has become one of the essential and major factor for the protection and stability of system. For the protection of data from threatens, the cryptography is deployed to the various techniques for safe and secure communication [2].

In 2000, Rijndael algorithm was selected for a new Advanced Encryption Standard (AES) by the National Institute of Standard and Technology (NIST) [3]. AES, the most widely encryption algorithm used to encrypt or decrypt data, is computationally intensive applications. AES is a symmetric block cipher which means it uses a same key for encrypting and decrypting [4]. The secret keys are exploited to encrypt and decrypt plaintext with mathematical transformations. Data encryption standard (DES) [5], triple-DES algorithm [6], AES [7], and SEED [8] are widely used in standard symmetric cryptography algorithms. With the symmetric block cipher, AES has higher security level and is lower complexity than other encryption techniques using asymmetric method [9]. Both hardware and software implementation is available for AES design. However, adopting hardware implement for AES algorithm can make higher speed, throughput and consume lower power than software implementation [10].

In this paper, we propose a 128-bit AES encryption core with three different operation modes for secure data communication. The AES encryption core is able to select the operation modes, which are consists of Electronic Code Book (ECB), Cipher-Block Chaining (CBC), CounTeR (CTR) modes. In our previous research [11], we exploited the AES core to implement the hardware-based multicore block cipher processor. The simulation and field programmable gate array (FPGA) emulation results show the feasibility of our encryption core. Moreover, the verification with prototype board shows the feasibility of our proposal.

The rest of the paper is organized as follows: In Section 2, we describe on AES algorithm and the three encryption modes. We present the system architecture in Section 3. Section 4 describes the simulation results and FPGA prototyping for the functional verification. The conclusion on AES encryption core and future work are presented in the Section 5.

II. AES ENCRYPTION

AES block cipher core is SPN (Substitution Permutation Network) architecture. Substitution Permutation is the changed architecture that does not divide blocks concurrently. Substitution is a process, which converts the plaintext into the look-up table (LUT). The converted value is called 'state'. Permutation is operation, which multiplies state and fixed matrix. These feature enables the parallel design of the AES core for the performance efficiency.

Figure 1 shows the AES algorithm system flow. AES algorithm consists of three steps: Initial Round, Round, and Final Round. Each part has four or less steps, AddRoundKey, SubBytes, ShiftRows, and MixedColumns. In the Initial Round, AddRoundKey step is processed. The

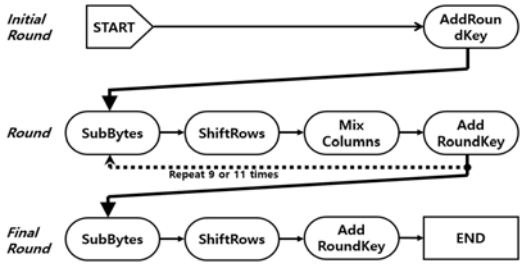


Fig. 1. AES algorithm system flow

a. Corresponding author; seung.lee@seoultech.ac.kr

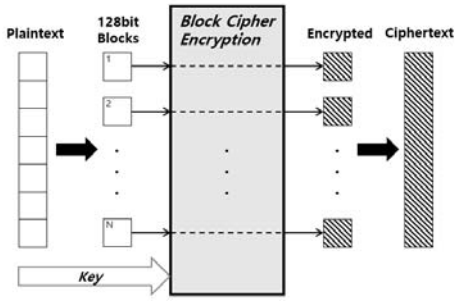


Fig. 2. ECB mode encryption

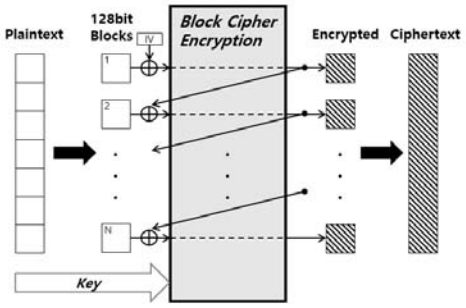


Fig. 3. CBC mode encryption

AddRoundKey features the bitwise exclusive-OR (XOR) of each plaintext blocks with the key value. The result of the previous step (Round part) is repeated in SubBytes, ShiftRows, MixedColumns, and AddRoundKey process, 9 times or 11 times. In the SubBytes step, input block value is replaced into the LUT value (S-Box). The upper 4 bits are used to row index of S-Box and the lower 4 bits are used to column index of S-Box. Therefore, we find and change each value from the S-Box. In the ShiftRows step, the value is rotated by the order of the column. For example, the zero order of column is rotated 0 time (do not rotate), the first order of column is rotated 1 time, the second order of column is rotated 2 times and the third is 3 times, i.e., if the first order of column is (01, 23, 45, 67), it is switched into (23, 45, 67, 01) in the ShiftRows step. The MixedColumns step has the complex product calculation and replacement process while each column working seperately. In the Final Round, the three steps except the MixedColumns are repeated one time.

AES encryption modes is generally classified into feedback and non-feedback modes. In the feedback mode, the encryption of the next block must be implemented after the previous block encryption is completed. In contrast, each block of data can be encrypted respectively in non-feedback modes. We consider one feedback mode and two non-feedback modes as the AES operation modes. The detailed descriptions on operation modes are as follows.

A. ECB Mode

ECB mode is the simplest block cipher mode. Figure 2 shows the ECB encryption process. The plaintext is divided into blocks, and each block is encrypted using block cipher encryption. Then, encrypted blocks are combined into the complete ciphertext. Since there is no dependency on the previous encryption result, the ECB mode can be processed simultaneously in multiple cores. In the ECB mode, the encryption of same data blocks always achieves the same encrypted results. Inversely, when decrypting in the ECB mode, it undergoes similar process as encrypting process. Separating ciphertext into blocks, decrypting each block and Combining block are the procedure of decryption.

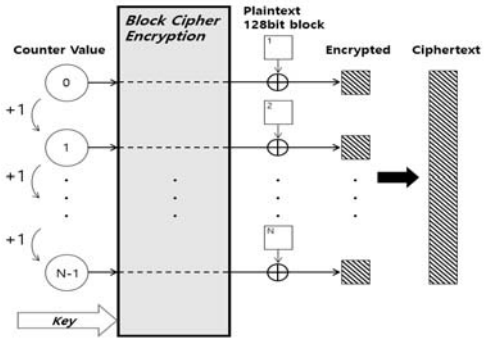


Fig. 4. CTR mode encryption

B. CBC Mode

CBC mode has the highest security among the operating modes of AES. This encryption process features the bitwise XOR of the plaintext blocks with previous ciphertext calculated in the previous encryption. When the first block of the plaintext is entered, initialization vector (IV) is required for encrypting in the CBC mode. An IV is generated newly whenever the encryption operation is executed. The first input block is formed by an XOR between the first block of the plaintext and the IV. After the first encryption, a first ciphertext comes out and is using for the XOR operation with the second block of plaintext. Similarly, from now on, the results of each XOR operating between the previous ciphertext and the next plaintext block make the encrypted ciphertext by passing through Block Cipher Encryption. As this mode asks the previous ciphertext for the next encryption, it is difficult to implement in parallel processing. On the other hand, decrypting in the CBC mode can process in parallel. It doesn't need the previous result for next decrypting, because the next decrypted plaintext is resulted by XOR operation between the previous ciphertext and the resulted block from Block Cipher Decryption.

C. CTR Mode

CTR mode exploits the AES block cipher into a stream cipher [12]. CTR mode uses the encrypted counter value and avoids the data dependency of the CBC mode [13]. Whenever a block cipher encryption is performed, the counter value (starting from zero) is increased by one, and the counter value is encrypted. So, the ciphertext is obtained as a bitwise XOR between each plaintext block and the encrypted counter value. Because there is no dependency of each encryption processing, it can be computed in parallel simultaneously, in several cores along with the ECB mode. Also, partial encryption is available in this mode, since each block has been indexing. Likewise, decryption could be

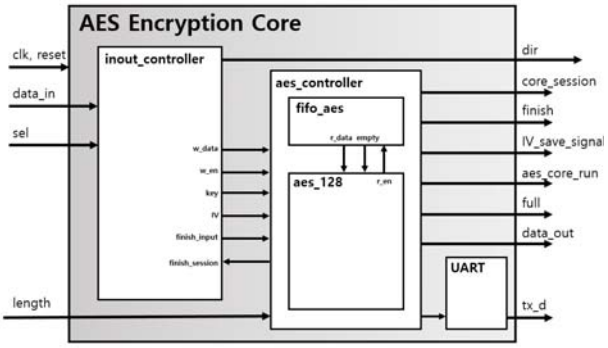


Fig. 5. AES encryption core architecture

done similarly by the series of encrypting process in the ECB mode.

III. SYSTEM ARCHITECTURE

The AES core, based on the open core crypto processor [14], is used as the computing unit of the block cipher processor. Figure 5 shows our AES core architecture supporting three operation modes. The AES core consists of the inout controller, AES controller and universal asynchronous receiver and transmitter (UART) module. The AES core executes a part of block cipher functions on the hardware scheduler platform. Our AES core exploits 128-bit encryption, we also designed the interface controller in order to use the inout pads due to the lack of chip level pins.

A. Inout controller

The inout controller is exploited to fulfill the 128-bit plaintext, key, IV input requirements by controlling with 3-bit input. The inout controller controls 168 inout pads. In addition, the inout controller is able to start the AES encryption with previous inputs. Table I shows the inout controller operation with following input.

TABLE I.
Inout controller operation

Port value	Direction	Description
000	Input	IDLE
001	Input	Lower data
010	Input	Upper data
011	Input	Key
100	Input	IV
101	Output	Ciphertext
110	Input	Encrypt with previous input
111	Output	UART mode

First, input ‘001’ and ‘010’ is used to set the plaintext, operation mode, and counter pointer. The ‘011’ and ‘100’ is the input value to set the key and IV, respectively. When the encryption is finished, the port value is set as ‘101’ to notify the ciphertext and encrypted data is achieved with the flag signal. The encryption is restarted with the previous plaintext and operation mode with input ‘110’. With input ‘111’ the encrypted data is transmitted through the UART module to verify the encrypted results.

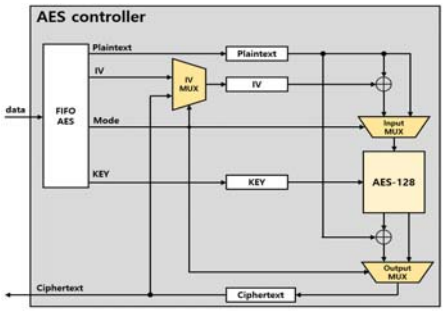


Fig. 6. AES controller architecture

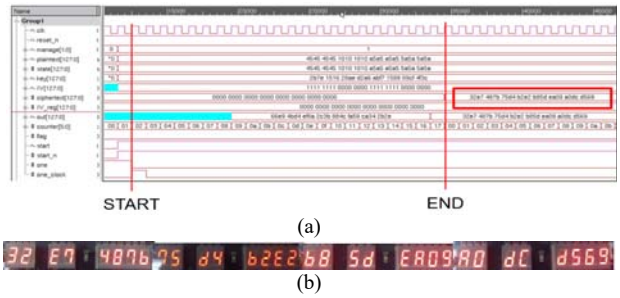
B. AES controller

The AES controller consists of the buffer and AES-128 module. The FIFO buffer receives the plaintext, encryption mode, key and IV from inout controller. The AES-128 implement the Rjindael algorithm, which is described in Section 2 and includes the S-Box as a LUT for the encryption.

We implement three operation mode (ECB, CBC, CTR) to support feedback and non-feedback through multiplexors and bitwise XOR. When the AES core operates in the CBC mode, the input multiplexor selects one of input signals that bitwise XORs between plaintext and IV. In the CTR mode, the output multiplexor selects one of results, which is bitwise XORs between plaintext and ciphertext.

C. UART module

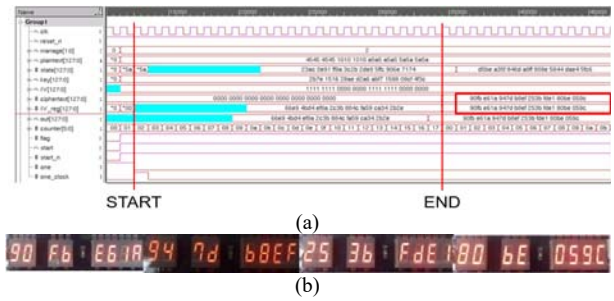
The UART module receives the ciphertext from the AES controller after every encryption is finished. The UART module buffers the ciphertext in the buffer. When the inout controller controls the UART module to transmit the ciphertext using the UART module, the UART module transmits the encrypted data for checking the ciphertext.



(a)

(b)

Fig. 7. ECB Mode (a) simulation result (b) FPGA emulation result



(a)

(b)

Fig. 8. CBC Mode (a) simulation result (b) FPGA emulation result

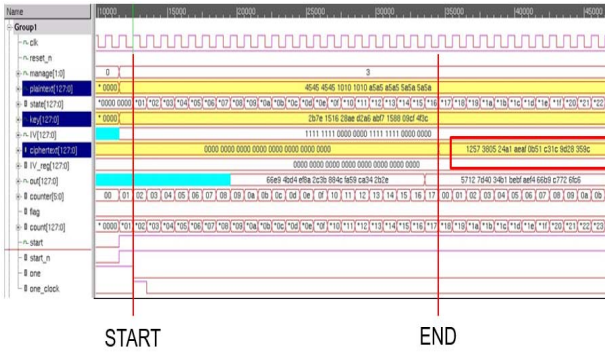


Fig. 9. CTR Mode Encryption Simulation Result

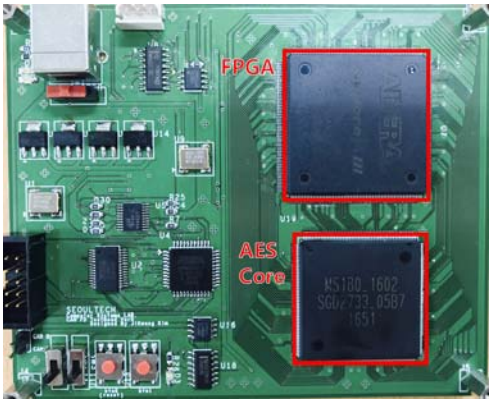


Fig. 10. A prototype board for verification

IV. IMPLEMENTATION

We verify the operations of AES core by comparing with software encryption results, simulation results and FPGA emulations. Figure 7 shows ECB mode encryption simulation and emulation results. We set the input and key data. When we start the encryption, we verify the result in simulation after 24 cycles. We check the emulation result using on FPGA. Figure 8 shows CBC mode encryption simulation results. This encryption process features the bitwise XOR of the plaintext blocks with previous ciphertext calculated in the previous encryption. We set the input, key and initial vector data. We verified the correctly results using the simulation and emulation. Figure 9 shows CTR mode encryption simulation results. CTR mode is combined ECB and CBC mode. To simulate the CTR mode, we set the input, key, and counter data.

The AES core was fabricated by using Magnachip/Hynix 0.18 μm CMOS technology. Core size is 3.5×3.5mm^2 and operation frequency is 50 MHz with 3.3 V supply voltage as shown in Table II.

TABLE II.
Characteristics of the AES core

Chip Specifications	
Technology	180nm CMOS
Supply Voltage	3.3 V
Chip Size	3.5 mm^2
Max. Frequency	100 MHz
Gate Counts	437 K @ 50 MHz

For the verification of AES encryption core, we designed the prototype board. The prototype board includes AES core chip, an UART module and general purpose I/O (GPIOs). We also exploit an FPGA in order to set key, IV and plaintext and verify the encrypted ciphertext. After finishing the encryption, the AES core transmits the ciphertext through UART communication. Furthermore, the encrypted data is compared with software-based AES encrypted results. With these procedures, we conducted the verification of the AES core. Figure 10 illustrates the prototype board of the AES core for verification. As a result, Fig. 11 shows the

Fig. 11. A verification result of AES core

encryption result of the AES core. We compared with software encrypted results with same key, IV and plaintext. We successfully demonstrated the feasibility of our AES core.

V. CONCLUSIONS

In this paper, we presented the AES core with three operating modes. Research environment is implemented by fabricating the AES core, which is integrated into ASIC, and producing the prototype board including the external memory. The performance analysis of the AES core is conducted by comparing the software-based AES encryption motion, along with making progress of parallel processing and scheduling of hardware.

The AES core, which is designed, can be operated into three operation modes. Using the AES core which is integrated into ASIC, process three different types of encryption, and store the encrypted result into the external memory. If the system needs the decryption process to use encrypted data, the process can be decrypted by the software method. The AES core with three operations mode can enhance the security of the system by protecting the data effectively, and improve the performance by designing in hardware. We expect that the encryption core for data security will take an important role in IoT devices. For the future work, we plan to design the AES decryption core and merge with existing AES encryption core to build full-encryption and decryption system.

ACKNOWLEDGMENT

This work was supported by the IDEC (IC Design Education Center). This work was also supported by a grant from the IT R&D program of MOTIE/KEIT [10076314, Development of lightweight SW-SoC solution for respiratory medical device]

REFERENCES

[1] R. Jinnai, A. Inomata, I. Arai and K. Fujikawa, "Proposal of hardware device model for IoT endpoint security and its implementation," *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, HI, 2017, pp. 91-93.

[2] N. S. S. Srinivas and M. Akramuddin, "FPGA based hardware implementation of AES Rijndael algorithm for Encryption and Decryption," *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, 2016, pp. 1769-1776.

[3] H. Kuo, I. Verbauwhede and P. Schaumont, "A 2.29 Gbits/sec, 56 mW non-pipelined Rijndael AES encryption IC in a 1.8 V, 0.18 μm CMOS technology," *Proceedings of the IEEE 2002 Custom Integrated Circuits Conference*, 2002, pp. 147-150.

[4] P. U. Deshpande and S. A. Bhosale, "AES encryption engines of many core processor arrays on FPGA by using parallel, pipeline and sequential technique," *2015 International Conference on Energy Systems and Applications*, Pune, 2015, pp. 75-80.

[5] *Data encryption standard (DES).* Technical Report Federal Information Processing Standard (FIPS) 46, National Bureau of Standards, 1977.

[6] Triple data encryption algorithm (TDEA, a.k.a. “Triple DES”). Technical Report Federal Information Processing Standard Publication 46-3, the standard ANSI X9.52-1998. NIST, 1998.

[7] *Advanced encryption standard (AES).* Technical Report Federal Information Processing Standard Publication (FIPS PUBS) 197, [Online] Available: http://csrc.nist.gov/CryptoTollkit/aes/rijndael, 2000.

[8] *SEED*. National Industrial Association Standard (TTAS KO-12.0004, 1999), KISA, 1998

[9] S. Mewada, P. Sharma and S. S. Gautam, "Exploration of efficient symmetric AES algorithm," *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, Indore, 2016, pp. 1-5.

[10] Pritamkumar N. Khose, Prof. Vrushali G. Raut, "Implementation of AES Algorithm on FPGA for Low Area Consumption",2015 International Conference on Pervasive Computing (JCPC).

[11] S. M. Lee, E. N. R. Ko and S. E. Lee, "A Hardware Scheduler for Multicore Block Cipher Processor," *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, Heraklion, 2016, pp. 750-754.

[12] D. Jayasinghe, R. Ragel, J. A. Ambrose, A. Ignjatovic and S. Parameswaran, "Advanced modes in AES: Are they safe from power analysis based side channel attacks?," *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, Seoul, 2014, pp. 173-180.

[13] S. Y. Lin and C. T. Huang, "A High-Throughput Low-Power AES Cipher for Network Applications," *2007 Asia and South Pacific Design Automation Conference*, Yokohama, 2007, pp. 595-600.

[14] Open cores, Crypto core. [Online]. Available: http://opencores.org/project,tiny_aes

Jung-Hwan Oh received the B.S. degree in the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea, in 2017. He is M.S. student in the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea. His research interests include computer architecture, System-on-Chip Design and hardware multi-core scheduler design.

Sang-Muk Lee received the M.S degree in the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea, in 2017. received the B.S. degree in Electronic Engineering from the Seoul National University of Science and Technology, Seoul, Korea, in 2014. He is M.S. student. His research interests include computer architecture, System-on-Chip, and hardware multi-core scheduler design.

Young-Hyun Yoon is B.S. student in the Department of Electronic Engineering at the Seoul National University of Science and Technology, Seoul, Korea. His research interests include computer architecture and hardware design.

Seung-Eun Lee received the Ph.D. degree in electrical and computer engineering from the University of California, Irvine (UC Irvine) in 2008 and the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon in 1998 and 2000, respectively. After graduating, he had been with Intel Labs., Hillsboro, OR, where he worked as Platform Architect. In 2010, he joined the faculty of the Seoul National University of Science and Technology, Seoul. His current research interests include computer architecture, multi-processor system-on-chip, low-power and resilient VLSI, and hardware acceleration for emerging applications.