높은 처리량을 가지는 AES를 위한 효율적인 파이프라인을 적용한 하드웨어 설계

Alexander O.A Antwi · 류광기

한밭대학교 정보통신전문대학원

Hardware Design with Efficient Pipelining for High-throughput AES

Alexander O.A Antwi · Kwangki Ryoo

Graduate School of Information and Communication, Hanbat National University

E-mail : alex.oa.antwi@gmail.com, kkryoo@hanbat.ac.kr

요 약

IoT 기술의 발전으로 IoT 기기들 사이의 통신에 보안이 중요해지고 있으며, 다양한 보안 알고리즘을 사용하고 있다. 많은 대칭 키 알고리즘 중에 AES (Advanced Encryption Standard) 알고리즘은 높은 보안성으로 지금까지 사용하고 있다. 본 논문에서는 효율적인 AES 알고리즘의 하드웨어 구조를 제안한다. 제안하는 하드웨어 구조는 암호화 모듈과 키 생성 모듈에 4단 파이프라인 구조를 적용하여, 높은 처리량과 낮은 지연시간을 가진다. 총 512비트의 일반 텍스트를 46 사이클에 처리가 가능하다. 제안하는 하드웨어 디자인은 65nm 공정에서 1.18GHz의 최대 주파수와 13Gbps의 처리량을 가지며, 180nm 공정에서 800MHz의 최대 주파수와 8.9Gbps의 처리량을 가진다.

ABSTRACT

IoT technology poses a lot of security threats. Various algorithms are thus employed in ensuring security of transactions between IoT devices. Advanced Encryption Standard (AES) has gained huge popularity among many other symmetric key algorithms due to its robustness till date. This paper presents a hardware based implementation of the AES algorithm. We present a four-stage pipelined architecture of the encryption and key generation. This method allowed a total plain text size of 512 bits to be encrypted in 46 cycles. The proposed hardware design achieved a maximum frequency of 1.18GHz yielding a throughput of 13Gbps and 800MHz yielding a throughput of 8.9Gbps on the 65nm and 180nm processes respectively.

키워드

Symmetric key cryptography, Advanced Encryption Standard, AES, Pipeline Structure

Ⅰ. INTRODUCTION

Security has become of utmost importance in our everyday transactions. In order to prevent an attacker from hijacking city traffic lights, cars or even water and gas supply, IoT devices that control them must be securely encrypted. AES is one such powerful algorithm for ensuring secured transactions between devices. It is one of the most secure symmetric block cipher algorithms in use currently.

IoT devices are embedded devices with limited resources and low power usage. As such, we need an AES implementation that can utilize this small area while increasing speed without increasing power and resource usage.

Previous works have focused on increasing throughput and reducing area, The techniques used include; fully pipelined architecture, implementation on an ASIP-based crypto-

processor as well as a one-time key expansion [1-6]. In this paper, we present a high throughput low area implementation of the AES algorithm. We present a four-stage pipeline architecture of the encryption and key generation which causes a reduction in the critical path, hence achieving a high frequency.

Section II describes the traditional AES algorithm. Section III describes our optimized implementation. Section IV compares our results with other papers. Section V is a conclusion of our results.

II. The AES Algorithm

Fig. 1 shows the AES encryption structure. AES algorithm has four main operations which it performs on a plain text (encryption) or cipher text (decryption). In order to encrypt or decrypt we need a key. The size of the plain text or key can be 128, 192 or 256 bits and the output is correspondingly 128, 192 or 256 bits.

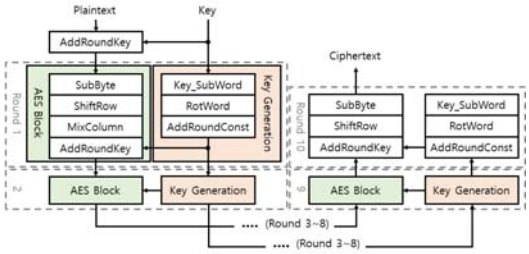

Fig. 1. AES encryption structure

The 128 plain text is sub divided into 16 bytes. This plain text as well as all the intermediate results and the final output are called states. In order to easily imagine an AES operation, we need to visualize the state as a 4x4 matrix that is filled column-wise from the first to the last column of the matrix. AES performs four operations in a total of 10 iterative rounds for each block size of 128. Each stage in AES depends on the previous stage and cannot proceed until the previous stage is done[3]. The four basic operations are explained below. Sub-byte is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). The shift rows operation performs a left cyclic shift on the state matrix depending on the row in question. Mix column operates on the columns. Each 4-byte column is considered as a vector and multiplied with a fixed (4 x 4) matrix. The operations in mix column are done in GF(2^8) field. Mix Column operation is not performed for the last round of AES. AddRoundKey operation is a bitwise XOR of the current state matrix with the round key. At the initial stage, we add the plain text to the original key.

III. Proposed AES hardware architecture

In this paper, we present a round pipelined architecture of the AES algorithm: Instead of using ten modules of each sub-round for our encryption process, we iterate over these modules for the total ten rounds using pipelining. This efficiently reduces the critical path of the design and explains our high throughput and frequencies. The mix column step takes a column of the state matrix and multiplies with a fixed matrix. Each column is multiplied by a either 0x02, 0x03 or 0x01. As a result, we decided to group these operations as shown in Fig. 2.

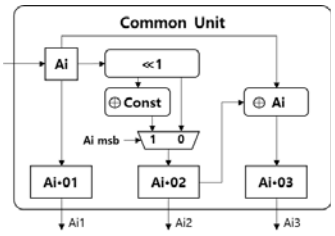

Fig. 2. MixColumn Common Unit

From this idea, we made a common unit to perform these operations on each byte, Ai. We then route the outputs of each common unit, (1 for each byte) as shown in Fig. 3, to give Bi. We find that this method of implementing mixed column helps us reduce the size of the design.

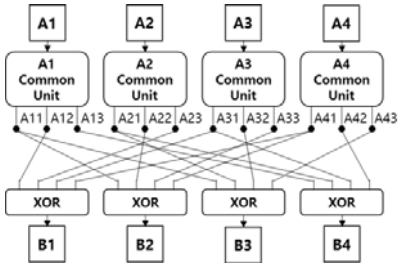

Fig. 3. Proposed MixColumn

Fig. 4 shows proposed hardware architecture.

The init_ARK module adds the plain text to the original key. The S_BOX_16 module consists of 16 instances of the S-Box look up table. Shift_Row module cyclically shifts the rows of the state matrix to the left based on the row in question. Mix_Column module operates on the columns of the state matrix for nine rounds and skips on the tenth round. The ARK module adds a round key to the Mix_Column result for nine rounds and to the Shift_Row result in the tenth round. The Key_Gen module generates a key for each round to be added to the plain text or the round cipher result.

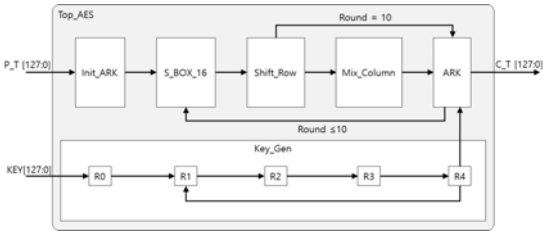

Fig. 4. Proposed hardware architecture

Ⅳ. Results and comparison

Our proposed design was implemented by Verilog HDL, synthesized with Synopsis Design Compiler and simulated with Model Sim. By using a four-stage pipeline structure, we were able to achieve an output block size of 512 bits. The use of a BRAM for the S-box as well as pipelining at the inner round ensured that the critical path was reduced. In Table 1, we do a comparison with the other works, [1-3] for the 65nm and 180nm processes. It can be seen that We present two technologies for the sake of comparison with other implementations.

Table 1. Comparison of results

Ref.	Freq. (MHz)	Gates (k)	Throughput (Gbps)	Cycles	Technology
[1]	125	58.4	1.6	10	180nm
[2]	1000	135	11.6	11	65nm
[3]	300	N/A	3.84	-	180nm
Ours1	1180	29.4	13	46	65nm
Ours2	800	27.1	8.96	46	180nm

Ⅴ. Conclusion

In this paper, proposed hardware design with efficient pipelining for high-throughput AES. The proposed hardware architecture uses a four-stage pipeline for both encryption and key generation. This enables us to encrypt four 128 bit plain texts at once. This was achieved in 46 cycles. We used a common unit for the Mix_Column module which caused a reduction in area. The proposed hardware was synthesized on two processes. The 180nm process yielded a frequency of 800MHz and a throughput of 8.9Gbps. The 65nm process yielded a frequency of 1.18GHz and a throughput of 13Gbps.

Acknowledgments

This research was supported by the MSI(Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program(IITP-2017-0-01681) and Human Resource Development Project for Brain scouting program(IITP-2016-0-00352) supervised by the IITP(Institute for Information and Communication Technology Promotion)

REFERENCES

[1] P.V. S. Shastry, A. Kulkarni, and M. S. Sutaone, “ASIC implementation of AES,” 2012 Annual IEEE India Conference (INDICON), pp. 1255-1259, Dec. 2012.

[2] H. Yuanhong and L. Dake, “High throughput area-efficient processor for cryptography,” Chinese Journal of Electronics, Vol. 26, No. 3, May 2017.

[3] H. Li, “Efficient and flexible architecture for AES,” IEEE Proceedings - Circuits, Devices and Systems, Vol. 153, No. 6, pp. 533-538, Dec. 2006.

[4] D. Smekal, J. Frolka, and J. Hajny, “Acceleration of AES Encryption Algorithm Using Field Programmable Gate Arrays,” 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016 Brno, Vol. 49, No. 25, pp.384-389, Oct. 2016.

[5] M. Mali, F. Novak, and A. Biasizzo, “Hardware Implementation of AES Algorithm,” Journal of Electrical Engineering, Vol. 56, No. 9-10, pp. 265-269, 2005.

[6] A. Soltani and S. Sharifian “An Ultra-high throughput and fully pipelined implementation of AES algorithm on FPGA,” Vol. 39, No. 7, pp. 480-493, Oct. 2015.