High Speed Architecture Implementation of AES using FPGA

Nilima D. Parmar
P.G. Student
Department of EXTC
DJ Sanghvi College of Engineering

Poonam Kadam
Assistant Professor,
Department of EXTC,
DJ Sanghvi College of Engineering

ABSTRACT
FPGA implementation of Advanced Encryption Algorithm for 128 bits is presented in this paper for high speed applications. It explores pipelining and sub-pipelining to gain speed optimization without increasing area considerably. It concentrates on placement of the pipelining registers rather than just increasing its number to gain speed. An encryptor with 8 stages of sub-pipelining for each round unit using the proposed architecture gives a throughput of 24.33 Gbps on Xilinx XCV1000 e-8 bg560 device and that of 29.99 Gbps on XC3S4000-5fg676 device.

Keywords
Rijndael, AES, pipelining, sub-pipelining, S-box.

1. INTRODUCTION
Rijndael was selected as the Advanced Encryption Standard (AES) algorithm by The National Institute of Standards and Technology (NIST) in 2001 [1]. AES is used extensively in cryptography from security point of view. It finds its application in various fields ranging from wireless phones, internet servers, smart cards etc.The AES uses a single key for both encryption and decryption, i.e it is a symmetric-key cipher. There are three variations in AES depending on the length of the cryptographic key, viz "AES-128", "AES-192" and "AES-256, where 128, 192 and 256 represents the key length. The AES encrypt/decrypt 128 bits of data accepted as input in the form of 4x4 array also known as State. The size of each array element of State is 8 bits. The general process for AES encryption and decryption is as shown in figure 1. There are three transformations namely, SubByte (SBox)/InvSubByte, ShiftRow/InvShiftRow, MixColumn/InvMixcolumn and a Key Expansion and Addition unit. FIPS defines these standard transformations for AES. These transformations are standard and are defined by FIPS for AES. The three transformations and key addition forms one AES round unit, the rounds units are repeated Nr number of times depending on the key length [1].In the works reported previously on AES, a traditional approach of using look-up tables (LUT) is used to implement the SubByte and InvSubByte transformation, as done in [2]-[6]. These LUTs occupy more area and also introduces unbreakable delay. To overcome this drawback, [7] and [8] introduces combinational logic only implementation of SubByte and InvSubByte using composite field arithmetic which was further exploited in [9] to obtain high-speed optimization. Compared to software implementations, hardware implementations of the AES algorithm provide more physical security as well as higher speed [9]. Hence pipelining and multiple stage sub-pipelining were introduced in AES implementation as in [9]-[12].

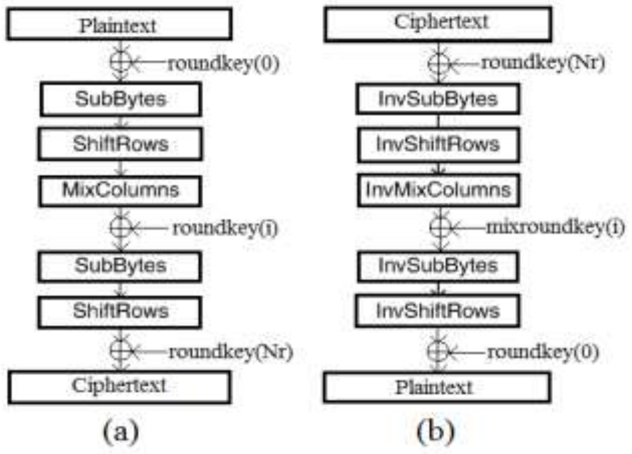

Fig 1: (a) Encryption; (b) Decryption.

The work proposed in this paper exploits sub-pipelining further to gain speed without significant increase in area. The previous fastest FPGA implementation by [9] employed sub-pipelining of 7 sub-stages in each round unit for the encryptor. The reported throughput achieved for the same on Xilinx XCV1000 e-8 bg560 device is 21.56 Gbps. In the work presented in this paper, the design for sub-pipelined encryptor consist of 8 sub-stages in each round unit and can achieve a throughput of 24.33 Gbps on the same FPGA architecture. Hence it can be deduced that the architecture proposed in this paper is more efficient than those proposed previously. This gain in speed and throughput is the result of proper placement of sub-pipelining registers rather than just increasing its number.

2. AES ARCHITECTURE
AES consists of encryption as well as decryption algorithm as shown in figure 1. In this paper only the encryptor design is implemented. Figure 2 below shows the architecture in detail along with the placement of sub-pipelining registers for one round unit.

2.1 SubByte
The SubByte transformation consist of isomorphic mapping (δ) followed by multiplicative inversion module, inverse isomorphic mapping (δ^{-1}) and affine transformation (AT). The

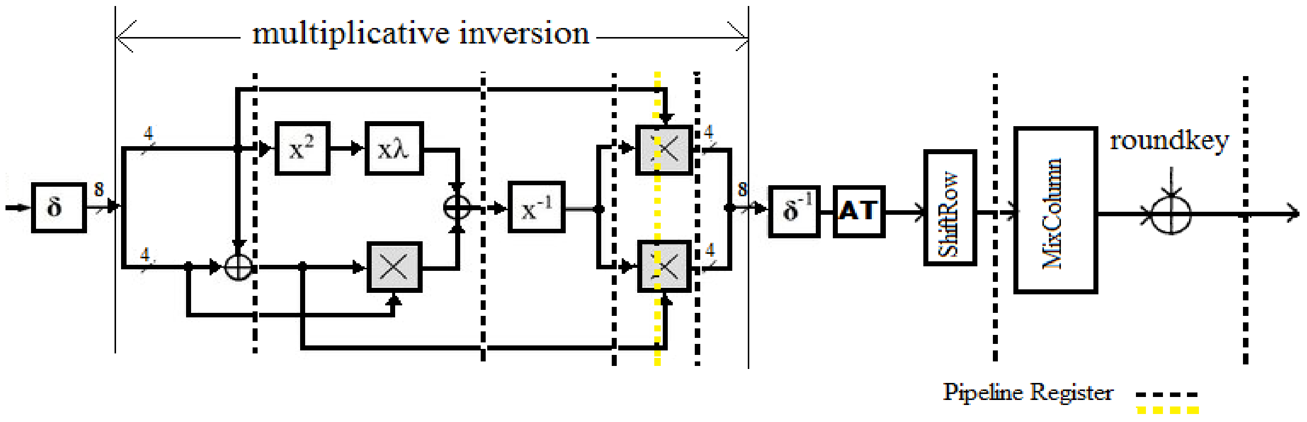

Fig 2: Hardware implementation of sub-pipelined Encryptor.

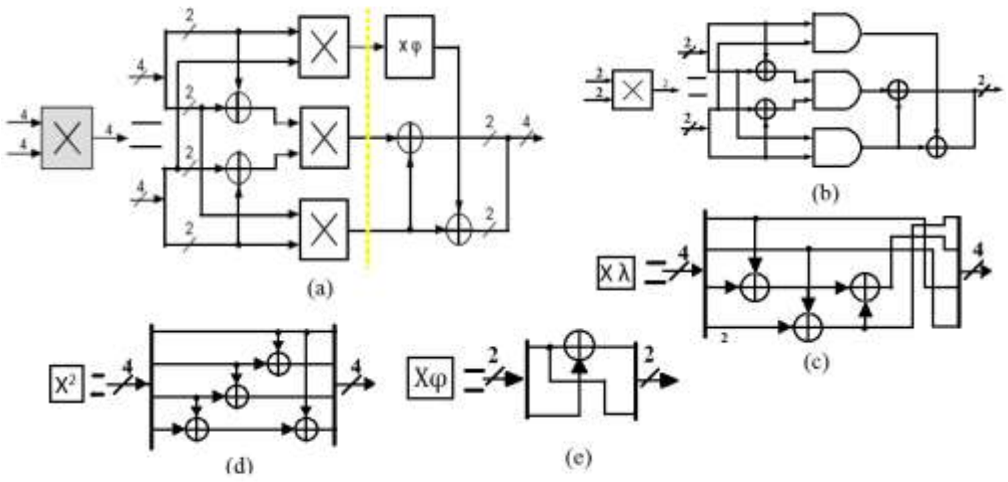

Fig 3: Hardware implementation of individual blocks: (a) multiplier in GF(2^4)[9]; (b) multiplier in GF(2^2)[9]; (c) constant multiplier (×λ)[9]; (d) squarer in GF(2^4)[9]; (e) constant multiplier (×φ)[9]

detailed hardware implementation about it as shown in figure 3. This transformation was traditionally implemented as LUTs. However it introduced unbreakable latency in the design and occupied ROM space. Hence they were replaced by combinational logic design which produced the values for SubByte and its inverse dynamically in real time.

2.2 ShiftRow and MixColumn

In ShiftRow, each row within State array is shifted cyclically by a specific value. The first row remains same without any shift. The second row shifts to left by one byte, whereas the third and fourth rows are shifted two and three bytes to the left respectively.

In MixColumn, each byte of a column is replaced with a value that is a function of all four bytes in the given column. Each column of the State is considered as polynomial over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial such that it has an inverse[1]. The equation for each element of MixColumn transformations are as given below;

$$s'_{0,c} = (\{02\}_{16} \bullet s_{0.c}) \oplus (\{03\}_{16} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = (\{02\}_{16} \bullet s_{1.c}) \oplus (\{03\}_{16} \bullet s_{2,c}) \oplus s_{0,c} \oplus s_{3,c}$$
$$s'_{2,c} = (\{02\}_{16} \bullet s_{2.c}) \oplus (\{03\}_{16} \bullet s_{3,c}) \oplus s_{0,c} \oplus s_{1,c}$$
$$s'_{3,c} = (\{02\}_{16} \bullet s_{0.c}) \oplus (\{03\}_{16} \bullet s_{0,c}) \oplus s_{2,c} \oplus s_{1,c}$$

(1)[1]

The above equation can be rearranged to simply the complexity and hence reduce the constant multiplication value to only {02} in hexadecimal.

$$s'_{0,c} = \{02\}_{16}(s_{0.c} \oplus s_{1,c}) \oplus (s_{2,c} \oplus s_{3,c}) \oplus s_{1,c}$$
$$s'_{1,c} = \{02\}_{16}(s_{1.c} \oplus s_{2,c}) \oplus (s_{3,c} \oplus s_{0,c}) \oplus s_{2,c}$$
$$s'_{2,c} = \{02\}_{16}(s_{2.c} \oplus s_{3,c}) \oplus (s_{0,c} \oplus s_{1,c}) \oplus s_{3,c}$$
$$s'_{3,c} = \{02\}_{16}(s_{3.c} \oplus s_{0,c}) \oplus (s_{1,c} \oplus s_{2,c}) \oplus s_{0,c}$$

(2)[9]

2.3 Key Expansion

Roundkeys are used to encrypt the State array. These roundkeys are generated by Key Expansion unit. They can be either produced beforehand and stored as LUTs to use for later purposes or can be generated on fly. The former approach can only be used for fixed key input whereas the latter can be used for variable key value. Also it enables pipelining to take place unlike that in case of LUTs.

The design for the same is as given in figure 4.

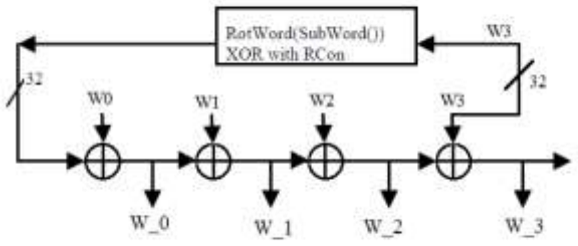

Fig 4: Key Expansion [14]

*S*ubWord is SubByte transformation on 32 bits word. The function. RotWord rotates the 32 bits word left by 8 bits. RCon is a round constant for every round [1]. It has a unique value for every round. It can be either implemented as LUT or on fly.The 128 bits input key is divided into four 32 bits words. They are used to generate roundkey words for all the ten rounds. W0, W1, W2 and W3 represent previous round roundkey words which are used to generate the current round roundkey words W_0, W_1, W_2 and W_3. These roundkeys are used for encryption.

3. IMPLEMENTATION AND RESULTS

The sub-pipelined encryptor for AES-128 is implemented by repeating the hardware for each round unit 10 times i.e creating 10 copies of the round unit and one Key Expansion unit. The design is implemented on two FPGA platforms namely VirtexE and Spartan 3. The tool used for synthesis and post implementation timing result is Xilinx ISE 9.2i. The sub-pipelined architecture takes *m x Nr +1* clock cycles to produce a valid output initially, where m is the number of sub-stages. After that it accepts and generates output at every clock cycle. In this design m = 8. This architecture when implemented on Xilinx XCV1000 e-8 bg560 device gives a maximum operating frequency of 190.11 MHz and a throughput of 24.33 Gbps. The same architecture when implemented on Xilinx XC3S4000-5fg676 device operates at a maximum frequency of 234.36 MHz and a throughput of 29.99 Gbps. The table 1 below shows the comparison between the previously suggested FPGA implementations on the same FPGA platform.From the below table 1 it can be seen that the design proposed in this paper is the most efficient one as per speed and also throughput per slice. The key contributor to this increase in speed is the SubByte transformation unit and its careful placement of sub-pipelining registers. The minimum clock period i.e maximum operating frequency is determined by the indivisible component with the longest delay [9]. Hence dividing a code into arbitrary number of sub-stages does not always increase the operating speed. Therefore dividing the rest of the code into more sub-stages with shorter delay does not reduce the minimum clock period [9].Therefore, the overall speed does not improve despite increased area caused by the additional registers [9]. It was observed that the Galois multiplication block in SubByte transformation produced the largest delay within the component. Hence it was further divided by inserting a pipeline register (shown in yellow colour figure 2 and 3) within the Galois multiplication block. Thus this delay was reduced considerably and led to speedup. Furthermore resource sharing was done in Key Expansion unit by employing the same SubByte transformation unit for the SubWord unit.

Table 1. Comparison of the FPGA implementation of AES algorithm

Design	Slices	Frequency (MHz)	Throughput (Gbps)	Mbps/slice
Jarvinen [10] XCV1000e-8	11719	129.2	16.500	1.408
Saggese [11] XCV2000e-8	5810 + 100 BRAM	158	20.300	1.091
Standnert [12] XCV3200e-8	15112	145	18.560	1.228
Xinmiao [9] XCV1000e-8	11022	168.4	21.556	1.956
This work XCV1000e-8	10964	190.11	24.22	2.219
This work XCS4000-5	11057	234.36	29.99	2.2712

4. CONCLUSION

This paper presents the FPGA implementation of fully sub-pipelined AES encryptor. The work proposed succeeds in increasing the speed of operation without highly affecting area consumption. This speedup is the result of combinational logic implementation of SubByte and Key Expansion along with resource sharing between them. Along with this, sub-pipelining and its placement has played a key role in achieving greater performance as compared to previous work Compared to the previous fastest architecture[9], his design has approximately 13% more throughput/slice and has an increased clock frequency of 12.89%. On similar lines, a decryptor can be constructed combined with encryptor. As both encryptor ans decryptor have some common modules such as SubByte and the Key Expansion unit with a little modification for creating mixroundkeys, this can be addressed in future work.

5. REFERENCES

[1] FIPS 197, "Advanced Encryption Standard (AES)", November26,2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] K. Gaj and P. Chodowiec, "Comparison of the hardware performance of the AES candidates using reconfigurable hardware". Presented at *Proc.3rd AES Conf. (AES3).*

[3] J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalist", presented at *Proc. 3rd AES Conf. (AES3).*

[4] H. Kuo and I. Verbauwhede, "Architectural optimization for a 1.82 Gbits/sec VLSI implementation of the AES Rijndael algorithm," in *Proc. CHES 2001*, Paris, France, May 2001, pp. 51–64.

[5] M. McLoone and J. V. McCanny, "Rijndael FPGA implementation utilizing look-up tables," in *IEEEWorkshop on Signal Processing Systems*,Sept. 2001, pp. 349–360.

[6] V. Fischer and M. Drutarovsky, "Two methods of Rijndael implementation in reconfigurable hardware," in *Proc. CHES 2001,* Paris, France, May 2001, pp. 77–92.

[7] Akashi Satoh, Sumio Morioka, Kohji Takano and Seiji Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization.", Springer-Verlag Berlin Heidelberg, 2001

[8] A.Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi,"Efficient implementation of Rijndael encryption with composite field arithmetic," in *Proc. CHES 2001*, Paris, France, May 2001, pp. 171–184.

[9] Xinmiao Zhang and Keshab K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm," IEEE Transactions on Very Large Scale Integration(VLSI) Systems, Vol.12, No. 9, Septemper 2004.

[10] K. U. Jarvinen, M. T. Tommiska, and J. O. Skytta, "A fully pipelined memoryless 17.8 Gbps AES-128 encryptor," in Proc. Int. Symp. Field-Programmable Gate Arrays (FPGA 2003), Monterey, CA, Feb. 2003,pp. 207–215.

[11] G. P. Saggese, A. Mazzeo, N. Mazocca, and A. G. M. Strollo, "An FPGA based performance analysis of the unrolling, tiling and pipelining of the AES algorithm," in Proc. FPL 2003, Portugal, Sept. 2003.

[12] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat, "Efficient implementation of Rijndael encryption in reconfigurable hardware: Improvements & design tradeoffs," in Proc. CHES 2003, Cologne, Germany,Sept. 2003.

[13] Naga M. Kosaraju, Murali Varanasi and Saraju P. Mohanty "A High-Performance VLSI Architecture for Advanced Encryption Standard (AES) Algorithm," IEEE Proceedings of the 19th International Conference on VLSI Design (VLSID'06).

[14] Amruta Page, P. V. Sriniwas Shastry, "AES-128 Key Expansion with LUT and OTF S-Box," International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 4, Issue 3, June 2014, An ISO 9001: 2008 Certified Journal

[15] Edwin NC Mui, "Practical Implementation of Rijndael S-Box Using Combinational Logic."

[16] Marian Cretu1and Cristian-Gabriel Apostol "A Modified Version of Rijndael Algorithm Implemented to Analyze the Cyphertexts Correlation for Switched S-Boxes"IEEE conference on Communication (COMM), Bucharest,2012

[17] K. U. Jarvinen, M. T. Tommiska, and J. O. Skytta, "A fully pipelined memoryless 17.8 Gbps AES-128 encryptor," in Proc. Int. Symp. Field-Programmable Gate Arrays (FPGA 2003), Monterey, CA, Feb. 2003,pp. 207–215.

[18] Ion Sima, Adrian- viorel Diaconu and Marian Cretu. "Analysis of Modified ShiftRows and MixColumns Transformations in Rijndael Algorithm" IEEE conference on Electronics, Computers and Artificial Intelligence (ECAI), 2013.

[19] Poonam Kadam, Nilima Parmar, "Pipelined Implementation of Dynamic Rijndael S-Box" in International Journal of Computer Applications, Vol. 111 (19578-1384), Feb 2015.