Pipelined Implementation of Dynamic Rijndael S-Box

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

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

ABSTRACT
Pipelined architecture for S-Box is proposed in this paper. ROM based look-up table implementation of S-Box requires more memory and introduce unbreakable delay for its access. Pipelined S-Box of combinational logic based implementation gives higher throughput and less delay as compared to that of no pipelined S-Box. 5, 6 and 7 stages of pipelined architecture has been simulated using Xilinx 9.2i for SPARTAN-3 FPGA. The result from Place and Route reports shows increase in maximum clock frequency at the cost of increased number of used slices. However the total delay calculated for the SubByte substitution for large amount of data is reduced considerably.

Keywords
Rijndael, AES, S-Box.

1. INTRODUCTION
The National Institute of Standards and Technology (NIST) selected and named Rijndael as the Advanced Encryption Standard (AES) algorithm in 2001 [1].The AES algorithm operates on a fixed block size of 128 bits taken as a 4x4 array of bytes called as state. The length of key used to encrypt this block can be 128, 192 or 256 bits. The state undergoes four transformations which are namely the AddRoundKey, SubByte, ShiftRow and MixColumn transformation [1]. Of all the transformation above, the SubByte transformation is the most computationally heavy transformation [2].

In most of the works related to AES, SubByte transformation is implemented as a ROM based look-up table which costs highly in terms of delay and memory. [2], proposed the dynamic design of S-Box. Dynamic design implies S-Box based on combinational logic. [5], has implemented a 2 stage pipelined combinational logic based S-Box. The reported results from the Place and Route report for the said architecture indicate that area occupied by this architecture is 43 slices with a maximum clock frequency of 72.155 MHz on a Spartan II XCS200-5 FPGA [5]. In this paper, S-Box as combinational logic is implemented with further increase in the number of pipeline stages and its effect on the architecture is studied. In this paper S-Box with 5, 6 and 7 stages of pipeline has been simulated on Xilinx 9.2i ISE FPGA family of XCs200 device. From the simulated results it can be deduced that increasing the number of pipelining stages, increases the maximum clock frequency, however it increases the number of clock cycles required for computation and the area occupied also increases. However the time required to compute large amount of data decreases considerably due to parallel processing.

2. SUBBYTE SUBSTITUION
The SubByte process involves Multiplicative Inversion in GF(2^8) followed by Affine Transformation (AT) [5]. The inverse SubByte process involves Inverse Affine Transformation followed by Multiplicative Inversion in GF(2^8) [5]. The multiplicative inversion module is used both in SubByte and inverse SubByte and hence both the processes can be combined together in a single architecture as shown in figure 1 below.

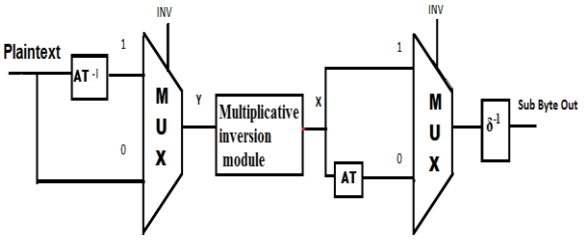

Fig 1: Combined SubByte and InvSubByte sharing a common multiplicative inversion module. [3]

The AT and AT^{-1} are the Affine Transformation and its inverse [5]. Multiplexer is used to decide which operation will take place, i.e SubByte during encryption and inverse SubByte during decryption. INV is the select line used for the purpose of switching. INV is set to 0 for SubByte while 1 is set when InvSubByte operation is desired.

The detailed architecture of multiplicative inversion module is as shown below, where $\varphi = \{10\}_2$ and $\lambda = \{1100\}_2$.

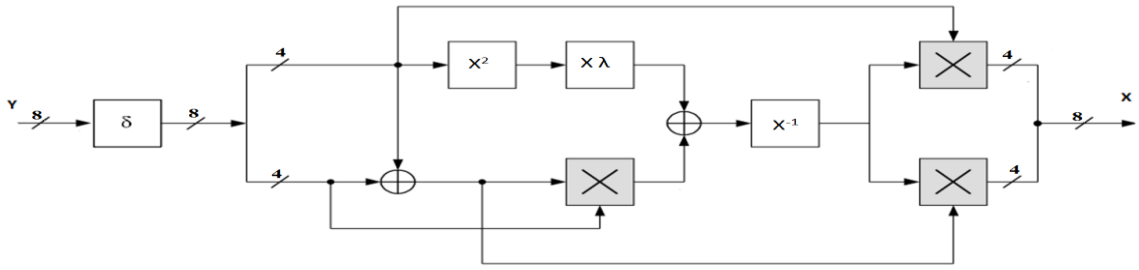

Fig 2 (a): Multiplicative inversion module [2]

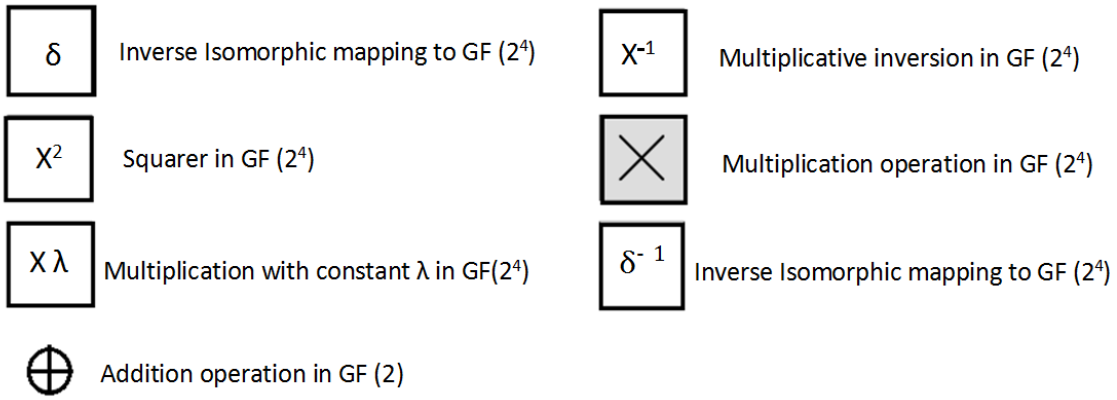

Fig 2 (b): Legends for the building blocks within the multiplicative inversion module [2]

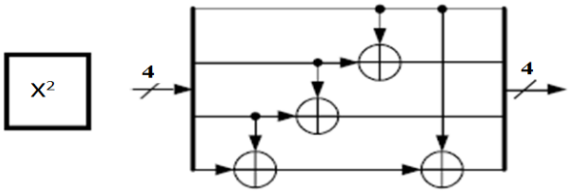

Fig 2 (c): Hardware diagram for Squarer in GF(2^4) [2]

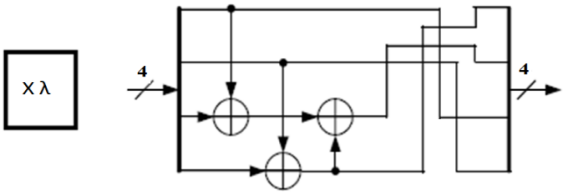

Fig 2 (d): Hardware diagram for multiplication with constant λ [2]

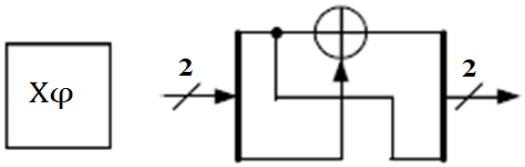

Fig 2 (e): Hardware implementation of multiplication with φ [2]

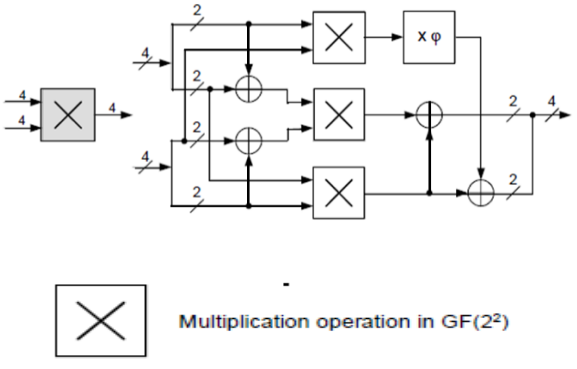

Fig 2 (f): Hardware implementation of multiplication in GF(2) [2]

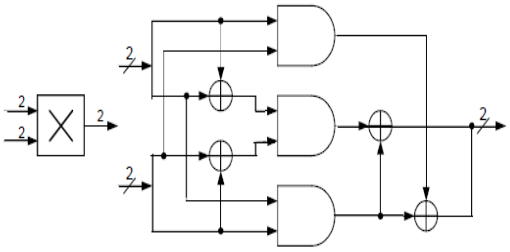

Fig 2 (g): Hardware implementation of multiplication in GF(2) [2]

3. PIPELINING ARCHTIECTURES

3.1 Stage Pipeline:

The short dash in figure 3 indicates the pipelined registers. Simulation results shows a maximum clock frequency of 284.819 MHz from the Place and Route report and the number of slices occupied is 62.

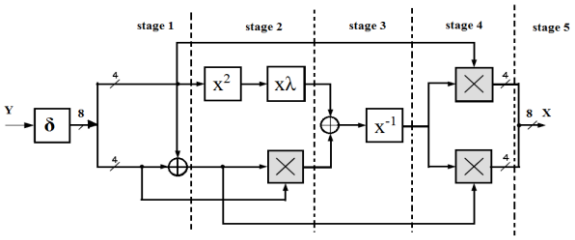

Fig 3: Multiplicative inversion module with 5 stage pipeline

3.2 Stage Pipeline:

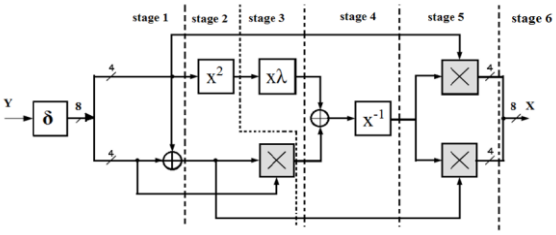

Fig 4: Multiplicative inversion module with 6 stage pipeline

The maximum clock frequency obtained for this architecture is 288.6 MHz. The number of slices used is 71.

3.3 Stage Pipeline:

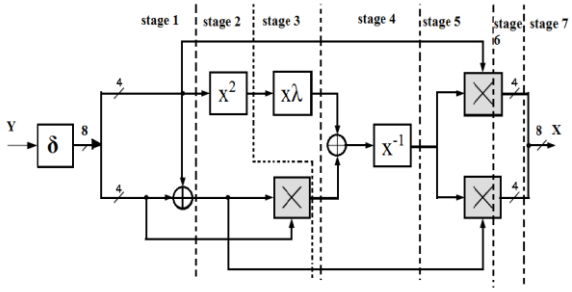

Fig 5: Multiplicative inversion module with 7 stage pipeline

In this architecture, the number of pipelining stages is 7. The simulation results from place and Route reports state the maximum clock frequency of 327.22 MHz and the number of slices used is 106.

Table 1. Performance Analysis for 8 Bit SubByte Transformation

Parameters	2 pipe	5 pipe	6 pipe	7 pipe
No of slices	43	62	71	69
Maximum clock frequency (MHz)	72.155	284.819	288.6	327.22
Delay for computing 48 bits (ns)	N/A	117.618	119.542	108.488

$$Delay = \frac{number\ of\ clock\ cycles}{maximum\ clock\ frequency}$$

From table 1, it can be deduced that as the number of pipeline stages increases, the area occupied will also increase. However the maximum clock frequency obtained also increases implying faster processing. From the above table, the performance of 6 pipe degrades as compared to 5 pipe in terms of delay and area. From this result it can be implied that performance not only depends on the number of pipeline stages but also the placement of the pipeline registers.

4. CONCLUSION

Pipelining helps for parallel processing hence a large number of data can be processed simultaneously within the constricted hardware hence increasing the effective area efficiency and also the throughput. The speed and performance also depends on the way pipelining is done rather than just the number of stages. The proposed pipeline architectures simulated results shows approximately 4. 4 and 4.5 times increase in maximum clock frequency as compared to 2 pipeline architecture.

5. FUTURE WORK

The performance of SubByte transformation can be further increased by placing the pipelining registers at a regular interval within the combinational logic. Time constraints can be added to make the code perform faster with the same resources. This can happen with pipelining or careful construction of combinational logic code.

6. REFERENCES

[1] FIPS 197, "Advanced Encryption Standard (AES)", November 26, 2001.

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] 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, September 2004.

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

[4] Vincent Rijmen, "Efficient Implementation of the Rijndael S-Box.", Katholieke Universiteit Leuven, Dept. ESAT. Belgium.

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

[6] Dr.R.V.Kshirsagar1 and M.V. Vyawahare "FPGA Implementation of High speed VLSI Architectures for AES Algorithm" IEEE conference on Emerging Trends in Engineering and Technology, 2012.

[7] Peter Ashenden, "The User Guide to VHDL, Third Edition" Morgan Kaufmann Publication.

[8] Joan Daemen and Vincent Rijmen, "The Design of RijndaeL: AES - The Advanced Encryption Standard (Information Security and Cryptography)", Springer.

[9] 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).

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

[11] 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.

[12] X. Zhang and K. K. Parhi, "Implementation approaches for the advanced encryption standard algorithm," IEEE Circuits Syst. Mag., vol. 2, no. 4, pp. 24–46, 2002.