8-bit AES Implementation in FPGA by Multiplexing 32-bit AES Operation

Chi-Jeng Chang[1], Chi-Wu Huang[1], Hung-Yun Tai[2], Mao-Yuan Lin[1],
**1 Institute of Applied Electronics Technology *2 Department of Industrial Education*
National Taiwan Normal University

Abstract

8-bit AES implementation was first proposed by Tim Good[8] as Application-Specific-Instruction-Process(ASIP), featured in low area design based on the stored-program design concept, which the software programs runs in a hardware processor. This paper proposes a direct hardware implementation of AES algorithm. There are two kinds of implementation, one uses shift registers for KeyExpansion and Mixcolumn called Shift-type, the other called BRAM-type uses Block RAMs (BRAMs) instead of shift registers. Both Implementations gain much higher throughput than ASIP. However, BRAM-type uses only 130 slices and achieves a throughput of 27 Mega bit per second (Mbps). Comparing to ASIP's 122 slices and 2.18 Mbps throughput, it achieves 12 times increase in throughput, 8% increase in slice number and no software programming necessary.

1. Introduction

Since the National Institute of Standards and Technology(NIST) selected the Rijndael Algorithm as a new Advanced Encryption Standard[1] in 2000, many hardware implementation of AES for high-throughput design[2][3][4][5] have been proposed. They typically unroll the loops with AES algorithm followed by pipelining of 128-bit data-path to achieve the order of tens Gigabit per second (Gbps). These designs are used mostly for high end devices such as accelerator cards for e-commercial services and secure trunk communication.

The 32-bit AES implementation started presenting after 2003 for low area and lower throughput applications by Chodowiec[6] and Rouvroy [7].

Finally, 8-bit AES application started after 2005 for even lower area resource design. Tim Good proposed an 8-bit Application-Specific-Instruction-Processor (ASIP)[8] having 15 instructions for the programming of encipher, decipher and key expansion processes. The programs stored in 2 Block RAMs (BRAM) are executed in the processor that included control, data memory, and a multiplier-accumulator unit for performing GF(2^8) arithmetic operations. ASIP uses 122 slices achieving 2.2 Mbps and 17.8 Kbps per slice.

Two types for AES implementation are proposed in this paper Shift-type was shift register for ShiftRow operation, while BRAM-type uses BRAMs for ShiftRow operation. Shift-Type uses 200 slices, achieves 38 Mbps throughput , BRAMS-type ,uses 130 slices achieves 34 Mbps throughput , which has 12 times increase in throughput and 8% increase in slice number when comparing to ASIP's 2.2 Mbps throughput and 122 slices in CLB area[8].

The main contribution of this paper is to provide a high throughput choice for low area FPGA design without software development cost comparing to ASIP.

The structure of this paper is as follow. In Section 2, a brief AES algorithm is described. In Section 3, general hardware implementation of AES based on 8-bit are summarized. Shift-type implementation is described in Section 4. BRAM-type is described in Section 5. Performance and Comparison are described in Section 6. Finally, some concluding remarks are made in Section 7.

2. AES Algorithm

Due to space constraints, please refer to [1] for AES algorithm in detail.

3. Hardware implementation of AES

Fig. 1 shows an 8-bit data-path configuration. It operates 16 clocks per round and need 160 clocks to finish 10-round encryption/decryption.

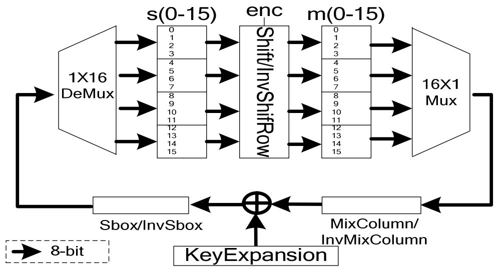

Fig. 1. 8-bit configuration for 16 clocks per round

4. 8-bit AES using Shift Register

Our design is based on the straight forward hardware design in Fig. 1. We use shift registers instead of multiplexer/de-multiplexer for saving some resource area. Because 16x1 Mux and 1X16 DeMux occupy more slice than shift registers, Fig. 2 shows the configuration.

1. KeyExpansion using 16-tyte and 4-byte registers

An 8-bit KeyExpansion is shown in Fig.2. Shift register M are 16-byte first-in-first-out(FIFO) shift registers. Registers r_0, r_1, r_2, r_3 are 4-byte parallel-in-serial-rotate shift registers with 4 inputs connected from M_{13}, M_{14}, M_{15}, M_{12}, respectively. When counter C counts to 15 (C_3 , C_2 , C_1 and C_0 all equal "1") the data in M_{12}, M_{13}, M_{14}, M_{15} are parallel shifted to r_3, r_0, r_1, r_2 to accomplish the shifting needed in AES KeyExpansion. Then r_0, r_1, r_2 and r_3 start to rotate 4 times at the next 4 clock cycles. 16 clocks are needed to complete a round for KeyExpansion. Therefore, 16-byte and 4-byte shift register are used for 8-bit KeyExpansion shown in Fig. 2.

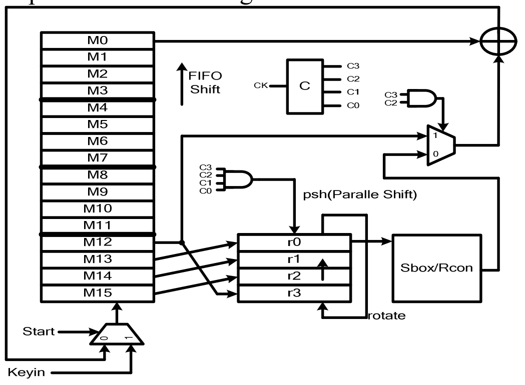

Fig. 2. 8-bit KeyExpansion using shift register

2. MixColumn using 16-byte and 4-byte registers

8-bit MixColumn needs two 16-byte registers S and M, ShiftRow is done by direct connection between S and M according to AES requirement. Registers (M0,M1,M2,M3),(M4,M5,M6,M7),(M8,M9,M10,M11) and (M12,M13,M14,M15) in Fig.3 are multiplexed to (b0,b1,b2,b3) at c=0,4,8,12. Registers (b0,b1,b2,b3) starts to rotate 4 times after receiving 4 bytes data from register M. Registers (b0,b1,b2,b3) are inputs to MixColumn/InvMixColumn for futher processing. Parallel-shift-16 signal psh16 is enabled when counter C=15 (psh16= $C3 \bullet C2 \bullet C1 \bullet C0$) to shift the data in parallel from S to M and accomplish the 128-bit ShiftRow processing.

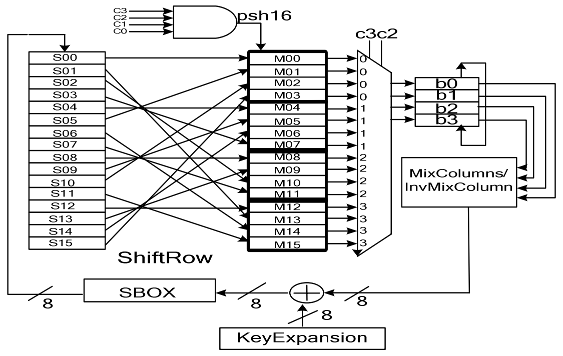

Fig. 3. 8-bit data-path using shift registers keeping 16 clocks per round

3. Multiplication and addition in MixColumn

MixColumn and InvMixColumn are multiplied a vector $B(x) = b_3x^3 + b_2x^2 + b_1x^1 + b_0$ by the constant vectors (3,1,1,2) and (b,d,e,9), respectively. If $(b0', b1', b2'b3')$ and $(d0, d1, d2.d3)$ are the results of mixcolumn and inverse mixcolumn, then they can be expressed as following, mentioned by Rouvroy[7].

$$\begin{bmatrix} b0' \\ b1' \\ b2' \\ b3' \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix} = \begin{bmatrix} 2\cdot b0 + & (2\cdot b1 + b1) + & b2 + & b3 \\ b0 + & 2\cdot b1 + & (b\cdot b2 + b2) + & b3 \\ b0 + & b1 + & 2\cdot b2 + & (2\cdot b3 + b3) \\ (2\cdot b0 + b0) + & b1 + & b2 + & 2\cdot b3 \end{bmatrix} --- (1)$$

$$\begin{bmatrix} d0 \\ d1 \\ d3 \\ d3 \end{bmatrix} = \begin{bmatrix} e & b & d & 9 \\ 9 & e & b & d \\ d & 9 & e & b \\ b & d & 9 & e \end{bmatrix} = \left(\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0 & 4 & 0 \\ 0 & 4 & 0 & 4 \\ 4 & 0 & 4 & 0 \\ 0 & 4 & 0 & 4 \end{bmatrix} \right) \otimes \begin{bmatrix} b0' \\ b1' \\ b2' \\ b3 \end{bmatrix}$$

$$= \begin{bmatrix} b0' + & 8\cdot(b0 + b1 + b2 + b3) & +4\cdot(b0 + b2) \\ b1' + & 8\cdot(b0 + b1 + b2 + b3) & +4\cdot(b1 + b3) \\ b2' + & 8\cdot(b0 + b1 + b2 + b3) & +4\cdot(b0 + b2) \\ b3' + & 8\cdot(b0 + b1 + b2 + b3) & +4\cdot(b1 + b3) \end{bmatrix} --- (2)$$

Expression (1) and (2) can be realized by logic circuit summarized in Fig. 4.

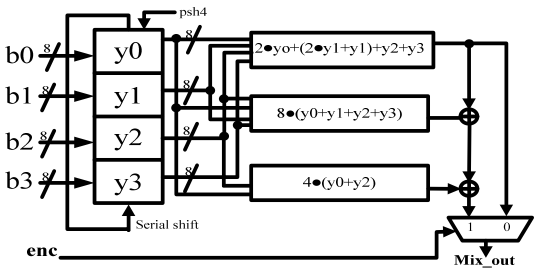

Fig. 4. MixColumn/InvMixColumn circuit realization

Shown in Fig. 4, register vector(y0,y1,y2,y3) = (b0,b1,b2,b3) when receiving a multiplexed data from M, and start rotate-shifting at next clock, then at next clock, (y0,y1,y2,y3) = (b1,b2,b3,b0), the MixColumn result signal Mix_out

$$= (2\cdot b1 + (2\cdot b2 + b2) + b3 + b0 + 8\cdot(b0 + b1 + b2 + b3) + 4\cdot(b1 + b3))$$
$$= d1$$

Thus, the circuit in Fig. 4 realizes the second row of expression (2), which is d1.

5. ShiftRow using BRAM

The Shift-type 8-bit AES in Section 4 needs 200 slices, which can be further reduced to 130 slices by moving shift register S and M into BRAMs A and B, BRAM-type is called for such implementation.
Observing the ShiftRow connect of S and M in fig. 3, it is found M(0,1,2,3,4,5,6,7..)=S(0,5,10,15,4,9,14,3..). Therefore, ShiftRow in BRAMs can use a 4-bit count-up-5 counter C5, as the output address addrA, addrB of Memory A and B, respectively. Another 4-bit counter C, which is a count-up-1 counter, is used for input address in Fig.5. The Round Switching Signal (rsw) switches the input/output of A and B at each round. Parallel load (pld) is performed when the value of C equals to 3,7,11,15, or (pld=C1 $\bullet$ C0).

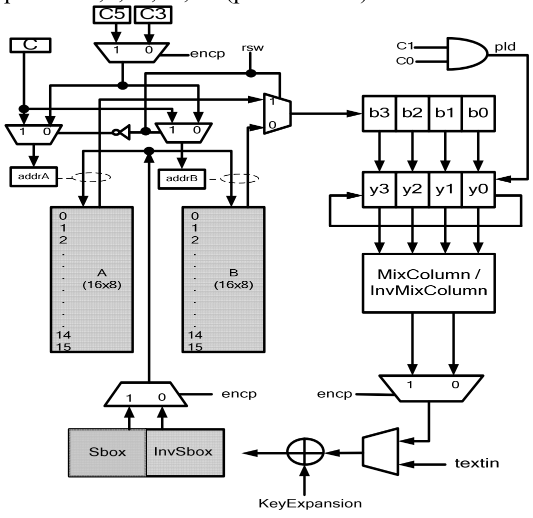

Fig. 5. ShiftRow, Mixcolumn and Sbox configuration

When A is inputting data from Sbox, then B must be outputting data as well as shifting row to MixColumn and vice versa. A count-down-3 binary counter C3, is used for InvShiftRow by the same observation. If encryption signal encp = "1" then C5 is selected for encryption else C3 is selected for decryption.

6. Performance Comparison

Table 1 lists our two types of 8-bit AES Implementation and ASIP, which run in 2 Xilinx FPGA Chip (XC2S30,XC2S15) and synthesized by ISE 8.2i.

The comparison basically shows that our designs have 66% (Shift-type) and 6.6% (BRAM-type) increases in slice number, respectively, both achieve more than 12 times (1200%) increase in throughput as well as over 8 times increase in throughput/slice.

Table. 1
Comparison our design with ASIP

Design & FPGA	ASIP Spartan-2 (XC2S515-6) [8]	Ours(Shift-type) Spartan-2 XC2S30	Ours(BRAMS-type) Spartan-2 XC2S15
Encipher cycles	3691	160	160
Max. Clock Freq.	72.3MHz	38.5MHz	34MHz
Datapath Bits	8	8	8
Slices	122	200	130
NO.of Block RAMs used	2	2	4
Block RAM Size(kbits)	4	4	4
Throughput(Mbps)	2.18	30.83	27

7. Concluding Remarks

Both types of our design provide high throughput choices for low area (under 200 slices) FPGA design without software development cost when comparing to ASIP.

BRAM-type achieves 130 slices but needs 4 BRAMs while Shift-type achieves 200 slices and only 2 BRAMs needed. There is a trade off between slice area and BRAMs for our two types implementations.

8 References

[1] NIST. Announcing the advanced encryption standard(AES), FIPS 197. Technical report, National Institute of Standards and Technology, November 2001.

[2] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," IEEE Trans. Very Large Scale Integr. (VLSI) Syst.

[3] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware ArchitectureWith S-Box Optimization," in Proc. LNCS ASIACRYPT'01.

[4] Alireza Hodjat, Ingrid Verbauwhede,"Minimum Area Cost for a 30 to 70 Gbits/s AES Processor", IEEE Computer society Annual Symposium on VLSI, 2004. Proceedings.

[5] Ricardo Chaves, Georgi Kuzmanov, Stamatis Vassiliadis, Leonel Sousa,"Reconfigurable Memory Based AES Co-Processor", IPDPS 2006. 20th International Parallel and Distributed Processing Symposium.

[6] Pawel Chodowiec, Kris Gaj,"Very Compact FPGA Implementation of the AES Algorithm", CHES 2003.

[7] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, J.-D. Legat,"Compact and efficient encryption/decryption module for FPGA implementation of the AES Rijndael very well suited for small embedded applications", Information Technology Coding and Computing, 2004. Proceedings. ITCC 2004,

[8] Tim Good, Mohammed Benaissa, "Very small FPGA application-specific instruction processor for AES", IEEE Trans. Circuit and System,vol. 53, no. 7, 2006.