Using a Pipelined S-Box in Compact AES Hardware Implementations

Cheng Wang and Howard M. Heys
Faculty of Engineering and Applied Science
Memorial University
St. John's, Canada
{cwang, hheys}@mun.ca

***Abstract*—Pipelined S-boxes are usually used in high speed hardware implementations of the Advanced Encryption Standard (AES), and not typically found in compact implementations because of the extra complexity added by the pipeline registers. In this paper, the area and speed performance of applying a pipelined S-box to compact AES hardware implementations is examined. A new compact AES encryption hardware core with 128-bit keys is proposed. The proposed design employs a single 4-stage pipelined S-box that is shared by the data path operation and the key expansion operation. Compared with the previous smallest encryption-only ASIC implementation of AES, it achieves an increase in throughput of 2.12 times while maintaining a similar gate count. This result indicates that it is reasonable to consider using pipelined S-boxes in AES hardware implementations targeted at applications requiring low area and moderate speed.**

I. INTRODUCTION

AES is a block cipher algorithm standardized by the US government [1], and it is regarded as the most reliable block cipher currently because there are no serious security flaws reported since it was released in 1999. Due to the wide recognition and adoption of AES, there has been a lot of interest in developing a compact AES hardware implementation for low cost security applications. Generally, a compact implementation refers to the low gate count of the implementation, and low gate count would result in low manufacture cost and contribute to low power consumption.

There have been many research works dedicated to the design of compact AES implementation. Typical works based on ASIC technology include [2], [3], [4], [5] and [6]. According to the comparison in [2], the design proposed in [2] achieves the smallest gate count while having a significant improvement on the throughput compared with [3], [4] and [5]. The work presented in [6] is a recent proposal for compact AES implementation with the focus on low power consumption, and it has poorer area and speed performance than the design in [2]. The design of [2] is an AES encryption core with an 8-bit data path where two S-boxes are implemented, one used by round operations and the other used by the key expansion. Even though the throughput of this design is higher than other compact designs, the critical path, which determines the maximum clock frequency and consequently the throughput, is quite long because it comprises the entire critical path of the S-boxes. S-boxes are the most complex component in an AES implementation and it generally involves a large number of gates on its critical path. Commonly in an AES implementation for high speed applications, the S-boxes are pipelined to several stages in order to reduce the critical path of the overall design. However this method is seldom applied to compact implementations for throughput improvement because it is assumed that pipeline registers would incur large hardware overhead, which is not affordable for the compact implementations targeted at low cost applications. In this paper, the applicability of using a pipelined S-box in compact AES hardware implementations is examined. A new VLSI architecture design for AES implementation is proposed to accommodate a 4-stage pipelined S-box and the implementation results show that the new design can achieve more than double the throughput of [2] while keeping the same gate count. The performance of using an S-box with other number of pipelined stages is also investigated and the results are compared and discussed. In the following, the design from [2] is referred to as the reference design.

II. AES OVERVIEW

AES is a block cipher algorithm with a block size of 128

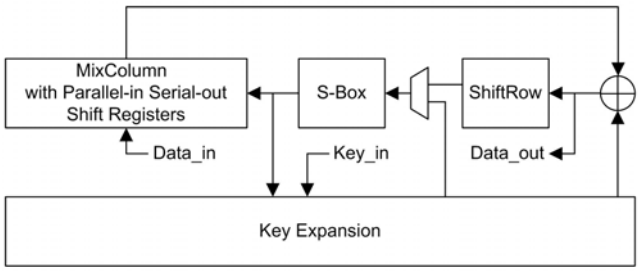

Figure 1. Block diagram of the proposed AES encryption core architecture

Support for this research was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and CMC Microsystems.

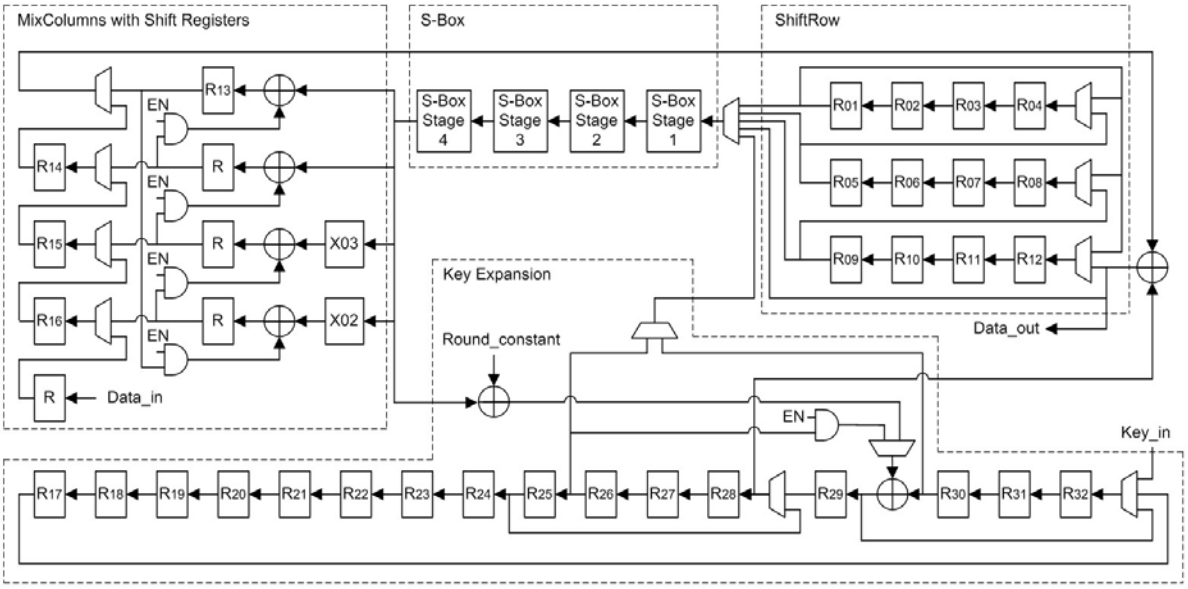

Figure 2. Architecture of the AES encryption core with a 4-stage pipelined S-box

bits. The key size of AES can be independently specified to 128, 192 or 256 bits, and accordingly there are 10, 12 or 14 iteration rounds to be performed for the encryption or decryption of a block. Each iteration round consists of *SubByte*, *ShiftRow*, *MixColumn* and *AddRoundKey* operations except the final round where the *MixColumn* operation is skipped. The intermediate results produced by these operations are denoted as *State*. The *SubByte* operation performs the non-linear transformation of each byte in the *State* according to the S-box mapping of AES. The *ShiftRow* operation cyclically shifts left the bytes in the rows of the *State* with offsets from 0 to 3 bytes. In the *MixColumn* operation, the columns of the *State* are considered as polynomials with coefficients in GF(2^8) and multiplied modulo $m(x) = x^4+1$ with a fixed polynomial $c(x) = 03x^3 + 01x^2+01x+02$. The *AddRoundKey* operation performs the bitwise exclusive-or (XOR) of the *State* and the round keys. The round keys are produced by the key expansion operation that involves substitution, word rotation, and XOR operations. Refer to [1] for a detailed description of the AES algorithm.

III. ARCHITECTURE DESIGN

The block diagram of the proposed architecture design is shown in Fig 1. In the architecture, the round operations have an 8-bit data path, and on the path, the *ShiftRow*, *SubByte*, *MixColumn* and *AddRoundKey* operations are performed byte by byte in sequence by the corresponding components. To complete the operation of one round of AES encryption, all the bytes of the *State* need to traverse the round operation data path once, so totally 10 traversals are required to encrypt an 128-bit plaintext after the data path loads it. The key expansion component also has an 8-bit data path and generates round keys on-the-fly using 128-bit keys. One S-box is used alternately by round operations and the key expansion. During the period the S-box is occupied by the key expansion component, the round operations are frozen by clock gating. The proposed architecture adopts the same *ShiftRow*, *MixColumn* and S-box structures as the reference design. The interconnection between components is modified and the key expansion component is newly designed in order to fit the interleaving use of the S-box and the influence of clock gating. The detailed architecture of the proposed design is shown in Fig. 2. All the paths in Fig. 2 have a width of 8 bits, and the blocks marked with "R" are 8-bit registers. The operation of each component and their interaction will be described in the following separately.

A. *ShiftRow Component*

The ShiftRow component consists of 12 8-bit registers connected in series and there are shortcuts from the input to the output and every fourth register. The component takes bytes arriving in the order of *State* columns and reorders the bytes while they are passing through. The detailed operation of the component is described in [2].

B. *S-Box*

The S-box adopted in the proposed design is developed in [7], and is considered to be the most compact AES S-box hardware structure [8]. Since the computation of multiplicative inverse over GF(2^8) can be converted to the computations in subfields, in [7] the S-box structure is examined for a number of representations of subfields, including both polynomial bases and normal bases, and the one leading to the implementation with the smallest gate count is identified. In the proposed architecture, the S-box is pipelined to 4 stages, and the places to insert pipeline registers are carefully studied and selected at the gate level so that the critical path of the circuit is minimized by balancing the delays of each stage while the number of required pipelined registers is minimized. The pipeline registers are placed between two consecutive stages but not shown in Fig. 2.

C. *MixColumn Component*

The *MixColumn* component is a serial-in, parallel-out

Table 1. REGISTER STATES OF THE ROUND OPERATION DATA PATH

Cycle	R_1	R_2	R_3 …… R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}
2	X	X	X	X	X	X	X	X	X	X	0_1
15	X	0_1	0_2 …… 0_7	0_8	0_9	0_{10}	0_{11}	X	0_{12}	0_{13}	0_{14}
*16-20	0_1	0_2	0_3 …… 0_8	0_9	0_{10}	0_{11}	0_{12}	X	0_{13}	0_{14}	0_{15}
27	0_8	0_9	0_{10} …… 0_{15}	0_{16}	X	X	X	1_1	X	X	X
28	0_9	0_{10}	0_{11} …… 0_{16}	X	X	X	1_1	X	1_2	1_3	1_4
*39-43	1_1	1_2	1_3 …… 1_8	1_9	1_{10}	1_{11}	1_{12}	1_{13}	X	X	X
*223-227	9_1	9_2	9_3 …… 9_8	9_9	9_{10}	9_{11}	9_{12}	8_{13}	X	X	X
230	9_4	9_5	9_6 …… 9_{11}	9_{12}	9_{13}	9_{14}	9_{15}	X	8_{16}	X	X
231	9_5	9_6	9_7 …… 9_{12}	9_{13}	9_{14}	9_{15}	9_{16}	10_1	X	X	X
232	9_6	9_7	9_8 …… 9_{13}	9_{14}	9_{15}	9_{16}	X	10_2	X	X	X
244	X	X	X	X	X	X	X	10_{14}	X	X	0_1
246	X	X	X	X	X	X	X	10_{16}	0_1	0_2	0_3

Table 2. REGISTER STATES OF THE KEY EXPANSION COMPONENT

Cycle	R_{17}	R_{18} …… R_{24}	R_{25}	R_{26}	R_{27}	R_{28}	R_{29}	R_{30}	R_{31}	R_{32}
1	X	X	X	X	X	X	X	X	X	0_1
17	0_2	0_3 …… 0_9	0_{10}	0_{11}	0_{12}	0_{13}	0_{14}	0_{15}	0_{16}	0_1
20	0_5	0_6 …… 0_{12}	0_{13}	0_{14}	0_{15}	0_{16}	1_1	0_2	0_3	0_4
*21	0_5	0_6 …… 0_{12}	0_{14}	0_{15}	0_{16}	0_{13}	1_1	0_3	0_4	1_2
*23	0_5	0_6 …… 0_{12}	0_{16}	0_{13}	0_{14}	0_{15}	1_1	1_2	1_3	1_4
*24-27	0_5	0_6 …… 0_{12}	0_{13}	0_{14}	0_{15}	0_{16}	1_1	1_2	1_3	1_4
28	0_6	0_7 …… 0_{13}	0_{14}	0_{15}	0_{16}	1_1	1_2	1_3	1_4	0_5
30	0_8	0_9 …… 0_{15}	0_{16}	1_1	1_2	1_3	1_4	0_5	0_6	0_7
39	1_1	1_2 …… 1_8	1_9	1_{10}	1_{11}	1_{12}	1_{13}	0_{14}	0_{15}	0_{16}
43	1_5	1_6 …… 1_{12}	1_{13}	1_{14}	1_{15}	1_{16}	2_1	1_2	1_3	1_4
227	9_5	9_6 …… 9_{12}	9_{13}	9_{14}	9_{15}	9_{16}	10_1	9_2	9_3	9_4
*231	9_5	9_6 …… 9_{12}	9_{13}	9_{14}	9_{15}	9_{16}	10_1	10_2	10_3	10_4
232	9_6	9_7 …… 9_{13}	9_{14}	9_{15}	9_{16}	10_1	10_2	10_3	10_4	9_5
234	9_8	9_9 …… 9_{15}	9_{16}	10_1	10_2	10_3	10_4	10_5	9_6	9_7
246	10_4	10_5 …… 10_{11}	10_{12}	10_{13}	10_{14}	10_{15}	10_{16}	0_1	0_2	0_3

matrix multiplier. It takes one byte input per clock cycle continuously for 4 cycles to receive a column of the *State*. At every fourth clock cycle, the computation of the *MixColumn* operation on the current column of the *State* is completed and the first byte of the result is output while the remaining three bytes are fed to the input of the parallel-in, serial-out shift registers incorporated in the *MixColumn* component. Subsequently, the three bytes are shifted out in the following three cycles. The blocks "X02" and "X03" in Fig. 2 generate the products of the current input byte and 02H and 03H, accordingly. The AND gates are used to bypass the XOR gates. This is done by setting EN to 0 and thus ensuring that the XOR operation does not change the data. During the loading of a 128-bit plaintext, only the shift registers at the right side of the component are working to shift in and shift out the plaintext bytes in serial. Refer to [2] for a detailed explanation of the component.

D. Key Expansion Component

The key expansion component has an 8-bit data path, which is implemented mainly by circularly connected shift registers R_{17} to R_{32}. The bytes of a round key are generated while the key state circulates through the shift registers and the generation of a round key is completed every time all of the key state has circulated along the path once. The computation of the next round key involves the substitution of the last four bytes of the current round key. This is realized by an 8-bit multiplexer switching the input of the S-box between the round operation data path and the key expansion data path. During the load period of key bits, the AND gate has EN set to 0 to bypass the XOR gate on the shift register path.

E. Overall Design

In order to clarify the operation of the architecture, the states of the numbered registers in Fig. 2 in certain selected clock cycles are shown in Tab. 1 and Tab. 2, dedicated to the round operation component and the key expansion component, respectively. For both tables, the output of the register during a clock period is regarded as the state of the register. In Tab. 1, for each state N_m ($0 \le N \le 10$, $1 \le m \le 16$), N represents the N-th round within which the byte of the *State* is processed (with the exception that N=0 indicates the *State* prior to the first round) and m represents the m-th byte of the *State* in the order of columns. Similarly, in Tab. 2 the state of a register N_m indicates the m-th byte of the N-th round key with the original key bits represented with N=0. The operation of the multiplexers and the AND gates can be easily determined from Tab. 1 and Tab. 2. Clock gating is applied regularly to both round operation and key expansion components. The selected cycles that demonstrate the happening of clock gating are marked with "*" in Tab. 1 and Tab. 2. The registers that require clock gating and the cycles when clock gating is active can be deduced from Tab. 1 and Tab. 2. It should be mentioned that, as is shown in Tab. 1 and Tab. 2, the

Table 3. IMPLEMENTATION RESULTS

Implementation	Area (gates)	Max. Freq. (MHz)	Clocks per block	Throughput (Mbps)
Proposed	2749	233	243	117.6
Reference design [2]	2815	69	160	55.6

Table 4. NORMALIZED PERFORMACE COMPARISON OF THE ARCHITECTURE USING A SINGLE S-BOX WITH DIFFERENT NUMBER OF STAGES

# Pipeline Stages	1	2	3	4	5
Area	0.93	0.96	0.99	1	1.05
Throughput	0.37	0.65	0.82	1	1.15
Ratio (Throughput/Area)	0.40	0.65	0.83	1	1.1

architecture works in a way for the final round operations slightly different from that for other rounds because the *MixColumn* operation is skipped in the final round. It takes 246 clock cycles to complete the encryption of a 128-bit plaintext including loading and unloading, and since there is overlapping of three clock cycles during loading and unloading, the effective clock count of the architecture is 243 for the encryption of a block.

IV. IMLEMENATION RESULTS, COMPARISON AND DISCUSSION

The proposed AES architecture design with a 4-stage pipelined S-box is synthesized using Synopsys Design Compiler version X-2005.09 under 0.18-μm CMOS standard cell technology from TSMC through CMC Microsystems [9]. The synthesis results of the proposed design with the constraint of minimum area are reported in Tab. 3. In order to have fair comparison, the synthesis results of the reference design shown in Tab. 3 is achieved by implementing and synthesizing the data path and key expansion part with the same tool, technology and constraint as the proposed design. It can be seen that the design with the pipelined S-box uses slightly fewer gates than the reference design and achieves an increase in throughput by a factor of 2.12. Although the overhead of control logic is not included in the comparison, the slight increase in gates used for the controller of the proposed design would be cancelled out by the slight decrease in gates on the data path. The implementation results and comparison show that, even though the pipelined S-box would introduce the latency of several clock cycles per round operation compared with the reference design, the reduction of the critical path delay by using the pipelined S-box compensates for the increased latency and brings significant boost to the throughput. Therefore, when throughput is a concern for a low gate count AES hardware implementation, the proposed design with a pipelined S-box is a much better choice than the reference design in [2] with two S-boxes. Only the performance comparison with the reference design of [2] is presented here because the reference design uses the lowest hardware cost among published works based on an ASIC platform.

In order to determine the influence of the number of pipeline stages on the overall performance of a compact design, the scenarios for varying number of pipeline stages are investigated. The area and throughput performance of the architecture using a single S-box with a variety of pipeline stages is normalized to the 4-stage pipeline scenario and shown in Tab. 4. It should be noted that the architecture in Fig. 2 only works with a 4-stage pipelined S-box and for other stage numbers up to 5 the architecture requires minor changes to fit. The architecture will not fit an S-box with more than 5 stages of pipeline without a major modification. The differences in area between pipeline stage numbers come from the different amount of pipeline registers required in each case. The data under one pipeline stage in Tab. 4 indicates the scenario of using an un-pipelined S-box. From Tab. 4, it can be seen that the ratio of throughput/area is gradually improved as the number of the pipeline stages increases. The architecture with a 4-stage pipelined S-box is selected to be specified in this paper because it has the best performance for an architecture with an area smaller than the reference design of [2].

V. CONCLUSION

In this paper, a new architecture design for compact hardware implementation of an AES encryption core is presented. The new design is featured with a 4-stage pipelined S-box. The implementation results show that, compared with the previous smallest encryption-only AES hardware implementation, the new design uses the same amount of gates to achieve an increase of 2.12 times in throughput. The implementation results indicate that pipelined S-boxes are applicable to compact implementations of AES for the purpose of speed improvement.

REFERENCES

[1] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," Federal Information Processing Standard Publication 197, Nov. 2001. Available at: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[2] P. Hämäläinen, T. Alho, M. Hännikäinen and T. D. Hämäläinen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core," 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006), pp. 577-583, Eindhoven, the Netherlands, Aug 2006.

[3] M. Feldhofe and J. Wolkerstorfer, "Strong authentication for RFID systems using the AES algorithm," Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), pp. 357–370, Boston, MA, USA, Aug. 2004.

[4] M. Feldhofer, J. Wolkerstorfer and V. Rijmen, "AES implementation on a grain of sand," IEE Proc. Inf. Secur., vol. 152(1), pp. 13–20, 2005.

[5] A. Satoh, S. Morioka, K. Takano and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization," 7th Int. Conf. on Theory and Application of Cryptology and Inf. Secur., Advances in Cryptology (ASIACRYPT 2001), pp. 239–254, Gold Coast, Australia, Dec. 2001.

[6] T. Good and M. Benaissa, "692-nW Advanced Encryption Standard (AES) on a 0.13-μm CMOS," to appear in IEEE Trans. VLSI Syst.

[7] D. Canright, "A very compact S-box for AES," 7th Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2005), pp. 441–455, Edinburgh, UK, Aug. 2005.

[8] S. Tillich, M. Feldhofer and J. Großschädl, "Area, delay, and power characteristics of standard-cell implementations of the AES s-box," J. Signal Process. Syst., vol. 50, pp. 251–261, 2008.

[9] CMC Microsystems, www.cmc.ca