FPGA Based Highly Efficient AES Implementation

Yong Zhang, Fang Zhou, Ning Wu, and Yasir

***Abstract*—This paper proposes a highly efficient 128-bit AES implementation based on FPGA. The S-box in AES is implemented in composite field, and the Common Sub-expression Elimination (CSE) algorithm is applied to reduce the redundant hardware overhead further more by 42.22% of XOR gates and 52.73% of AND gates. Then we analyze the delay of each arithmetic unit and use pipeline in the proper position to improve the throughput without bringing extra resource consumption. The experiment shows that our strategy can achieve the throughput of 93.54Gbps at the cost of 5081 slices on a Xilinx Virtex-6 XC6VLX240T device. The efficiency of our implementation is 18.41 Mbps/Slice which is much higher than the previous works.**

***Index Terms*—AES, High efficient, Composite field, CSE, Pipeline, FPGA**

I. Introction

IN 2001, National Institute of Standards and Technology (NIST) formally adopted the Rijndael algorithm as the Advanced Data Encryption Standard (AES) algorithm encryption standard. Since then, it has become a research hotspot to achieve high throughput and low hardware consumption of AES and it's very necessary for high-speed but resource-constrained applications.

Some methods have been proposed to realize the highly efficient AES. Chih-Peng Fan and Jun-Kui Hwang used the Content-Addressable Memory method to design a high-speed S-box and it achieved the throughput of 0.867Gbps and 32Gbps respectively in the loop structure and the unrolled structure [1]. Chi-Jeng Chang et al. designed a 32-bit AES circuit reaching the throughput of 0.867Gbps with the consumption of 156 slices [2], although it occupied little hardware resource, its throughput was too low. Shanxin Qu et al. used the external pipelining method to accelerate the AES circuit achieving the throughput of 73.737Gbps with 22994 LUTs, but this method consumed too much hardware resources causing the low efficiency [3]. In Ref.[4], there is no pipeline in S-box, and in Ref.[5] [6] and [7], the sub-pipelined method was applied reaching the throughput more than 80Gbps, however, their pipeline strategy was not optimal, and the efficiency still could be improved.

In order to improve the implementation efficiency, we use the composite field (CF) arithmetic to design the S-box, and the DACSE algorithm is applied to optimize the arithmetical units. We can get a compact AES circuit with this method. Based on the compact AES achieved, we analyze the delay of each unit. Then our pipeline strategy is carried out to boost the frequency which improves the throughput of the implementation.

The rest of the paper is organized as follows: Section II introduces the principle of AES algorithm and CSE algorithm briefly. Section III describes the composite field implementation of S-box, and applies the CSE algorithm to optimize the S-box circuit. The optimal pipelining method is represented in Section IV. Section V shows the results of our research and the comparison with previous works. Section VI concludes this paper.

Manuscript received July 1, 2017; revised July 27, 2017. This work was supported in part by the National Natural Science Foundation of China (61376025), the Fundamental Research Funds for the Central Universities (NS2017023) and the Natural Science Foundation of Jiangsu Province (BK20160806).

Y. Zhang is with College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China (e-mail: 2450069140@qq.com).

F. Zhou is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China.

N. Wu is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China (e-mail: wunee@nuaa.edu.cn).

Yasir is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China.

II. Review Of Aes And Cse Algorithm

A. Review of AES

AES is a symmetric block cipher algorithm with a data length of 128 bits, a key length of 128 bits, 192 bits or 256 bits, and the number of round transformation is 10, 12 and 14, respectively. Our work focuses on the 128-bit key AES. During the process, the 128-bit input data is mapped to the matrix of 4 by 4 called the State Matrix. Fig.1 illustrates the process of AES.

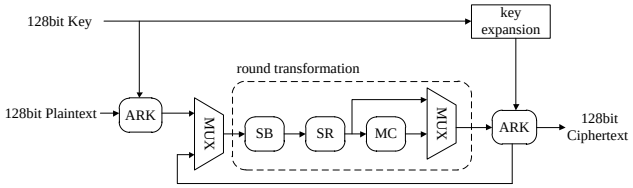


Fig. 1 Process of AES

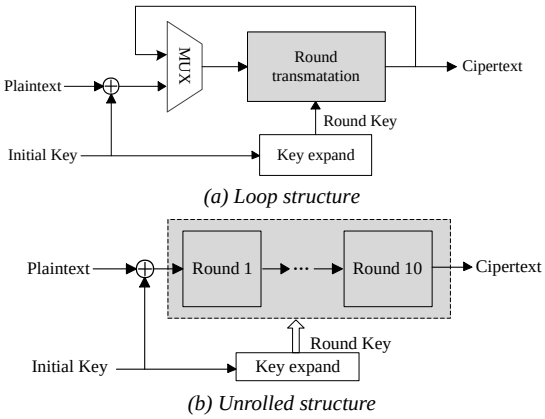
AES algorithm is mainly composed of key expansion and round transformation. Key expansion uses the initial key to provide round key for each round transformation. In this paper, we use the pre-calculation method to expand the round key. And there are four parts in round transformation, SubBytes (SB), ShiftRows (SR), MixColumns (MC) and AddRoundKey (ARK). The last round does not have MixColumns.

SB is the only nonlinear transformation in the round transformation, which requires two steps, the first step is computing the multiplication inverse (MI) of the input byte on the finite field $GF(2^8)$, the second step is computing the affine transformation (AF). SR shifts each row of the State Matrix cyclically left to get a new matrix, the first, second, third and the fourth row is shift left by zero, one, two and three bytes respectively. The MC takes each column of the

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

State Matrix as the target, the column is multiplied by the polynomial $3x^3+x^2+x+2$ in the $GF(2^8)$ field and modulo by x^4+1. Finally, ARK adds the result of the MC and the round key together to get the round transformation result.

Usually, there are two kinds of the AES overall structure as shown in Fig.2, Loop structure and Unrolled structure.



(a) Loop structure

(b) Unrolled structure

Fig.2 AES Overall Structure

Loop structure consumes few hardware resources but achieves low throughput, the unrolled structure can reach high throughput by applying the pipeline strategy, but will consume a lot of hardware resources. The AES implemented in this paper is based on the unrolled structure. The optimization method for the hardware consumption will be described in Section II. The optimization method of the throughput will be explained in the Section III.

B. Principle of CSE

Usually there exists redundancy in many expressions which consist of exclusive-or operation. In order to reduce the redundancy in the AES circuit, we choose CSE algorithm to make the circuit become compact. The idea of CSE is to identify common sub-expressions (CSs) that are present in expressions more than once and replace them with a single variable. CSE algorithm can be exploited to extract the common factors in all the bit-level equations, in order to reduce the hardware cost of combinational logic implementation.

Generally, CSE algorithm involves the following steps:

i) Identify CSs presenting in the transformation.

ii) Select a CS for elimination.

iii) Remove all the occurrences of the selected CS.

iv) The eliminated CS is computed only once.

v) Repeat Step i~iv until none of multiple CSs is present.

Several CSE algorithms have been proposed, and the CSE used in this paper is called Delay Aware CSE (DACSE) [10]. The DACSE algorithm considers both the hardware overhead and the delay of the operation. We take the matrix [0x3, 0x7, 0xf, 0xd] multiplication as an example.

$$\begin{bmatrix} y_3 \\ y_2 \\ y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} = \begin{bmatrix} x_1 + x_0 \\ x_2 + x_1 + x_0 \\ x_3 + x_2 + x_1 \\ x_3 + x_2 + x_1 + x_0 \end{bmatrix}$$

$$= \begin{bmatrix} (x_1 + x_0)_{b_1} \\ x_2 + b_1 \\ x_3 + x_2 + x_1 \\ x_3 + x_2 + b_1 \end{bmatrix} = \begin{bmatrix} (x_1 + x_0)_{b_1} \\ (x_2 + b_1)_{b_2} \\ x_3 + x_2 + x_1 \\ x_3 + b_2 \end{bmatrix}$$

First, we select x_1+x_0 into the new factor b_1 to get a new matrix, then we select the common expression x_2+b_1 into the new factor b_2. As we can see, before optimization, the circuit needs 8XORs, and delay is $4T_{XOR}$, after optimization, the circuit only needs 5XORs, and delay is $4T_{XOR}$.

III. The Proposed Unrolled AES Architecture

A. Reducing Hardware Resource Method

The hardware overhead of S-box is the largest in the four arithmetic units of the round transformation, occupying about 75% area of the round transformation. So, we focus on the method to reduce the hardware consumption of the S-box in this part.

Usually, there are two kinds of methods to implement the S-box, the LUT method and the CF method. In this paper, we design a 128-bit AES with full parallel S-boxes used in the unrolled structure. Each round transformation needs 16 S-boxes, there will be 160 S-boxes in ten round transformations, and if we add the 40 S-boxes used in the key expansion, the entire AES needs 200 S-boxes. If the S-box is designed by LUT, it will consume a lot of hardware resources like [8]. The method based on composite field can reduce the S-box hardware overhead to a great degree, so we choose the CF way to achieve a compact S-box. In this work, we decompose the finite field $GF(2^8)$ to the composite field $GF((2^4)^2)$ to realize the composite field S-box [9]. The method will be introduced step by step as follows.

i) Step1: S-box In $GF((2^4)^2)$

In CF technology, an isomorphic mapping matrix is demanded to map the input vector from the finite field $GF(2^8)$ to the composite field $GF((2^4)^2)$, and its inverse matrix is required to revert the computing results to $GF(2^8)$ at last. So the S-Box based on CF technique can be expressed as:

$$Z = M(\delta^{-1}(\delta X)^{-1}) + Y \tag{1}$$

where X is the input vector, Z is the output vector, M is the affine matrix and δ is the mapping matrix. The S-box in $GF((2^4)^2)$ is displayed in Fig.3.

First, we map the input vector from the finite field $GF(2^8)$ to the composite field $GF((2^4)^2)$, the expression is:

$$\begin{cases} y_7 = x_7 \oplus x_6 \oplus x_4 \oplus x_1 \\ y_6 = x_7 \oplus x_5 \oplus x_3 \oplus x_2 \\ y_5 = x_7 \oplus x_6 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \\ y_4 = x_6 \oplus x_5 \oplus x_4 \\ y_3 = x_2 \\ y_2 = x_7 \oplus x_6 \oplus x_5 \oplus x_4 \oplus x_3 \oplus x_2 \\ y_1 = x_7 \oplus x_6 \oplus x_5 \\ y_0 = x_7 \oplus x_5 \oplus x_0 \end{cases} \tag{2}$$

The mapping operation costs 5XOR gates. Then we make MI in composite field $GF((2^4)^2)$. In CF technology, the MI over $GF(2^8)$ is built iteratively from $GF(2)$ by using the following irreducible polynomials:

$$\begin{cases} f_1(z) = z^2 + z + \gamma \\ f_2(y) = y^2 + y + \lambda \\ f_3(w) = w^2 + w + 1 \end{cases} \tag{3}$$

In Eq.(3) $f_1(z)$, $f_2(y)$, $f_3(w)$ represent the irreducible polynomials of field $GF((2^4)^2)$, $GF((2^2)^2)$ and $GF((2)^2)$

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

respectively. In Ref.[9] the circuit occupies the least hardware resources when $\gamma=(0001)_2$ and $\lambda=(10)_2$.

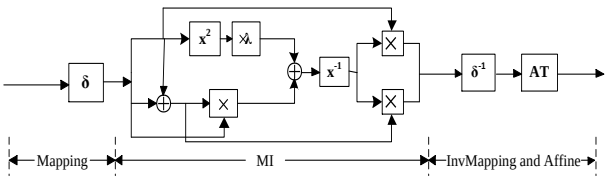


Fig.3 S-box Structure in $GF((2^4)^2)$

We use the same coefficients. According to Eq.(3), the MI over $GF(2^8)$ decomposed to the $GF((2^4)^2)$ is expressed as:

$$A^{-1}=\left(A_h^2\lambda+A_l\left(A_h+A_l\right)\right)^{-1}\left(A_h+\left(A_h+A_l\right)\right) \tag{4}$$

where $\{A_h, A_l\}\in GF(2^4)$. From Eq.(4), we can see that the MI needs three multiplication units, two addition units, one multiplying-constant unit, one inverse unit and one square arithmetic unit.

ii) Step2: Arithmetic Unit Analysis

We analyze all the arithmetic units in the composite field S-box, assuming that $\boldsymbol{a}=\{a_3, a_2, a_1, a_0\}$, $\boldsymbol{b}=\{b_3, b_2, b_1, b_0\}$ are two operations belonging to $GF(2^4)$. We assume that $\boldsymbol{c}=\{c_3, c_2, c_1, c_0\}$ represents the multiplication result of $\boldsymbol{a}$ and $\boldsymbol{b}$, so $\boldsymbol{c}=\boldsymbol{ab}$ mod$(x^4+x^3+x^2+x+1)$ can be expressed as:

$$\begin{cases} c_3=(a_3b_1\oplus a_1b_3)\oplus(a_3b_0\oplus a_0b_3)\oplus(a_2b_1\oplus a_1b_2)+a_2b_2 \\ c_2=(a_3b_1\oplus a_1b_3)\oplus(a_2b_0\oplus a_0b_2)\oplus a_2b_2\oplus a_1b_1 \\ c_1=(a_3b_1\oplus a_1b_3)\oplus(a_1b_0\oplus a_0b_1)\oplus a_3b_3\oplus a_2b_2 \\ c_0=(a_3b_2\oplus a_2b_3)\oplus(a_3b_1\oplus a_1b_3)\oplus a_2b_2\oplus a_0b_0 \end{cases} \tag{5}$$

We can see that the multiplication (Mul) needs 21XOR gates and 25AND gates. We let $\boldsymbol{d}=\{d_3, d_2, d_1, d_0\}$ represents the square result, $\boldsymbol{d}$ can be expressed as:

$$\begin{cases} d_3=a_2 \\ d_2=a_2\oplus a_1 \\ d_1=a_3\oplus a_2 \\ d_0=a_2\oplus a_0 \end{cases} \tag{6}$$

The square (Squ) needs 3XOR gates. Similarly, we let $\boldsymbol{e}=\{e_3, e_2, e_1, e_0\}$ represents the result of multiplying-constant (MulC) where the constant is $\{\lambda=(10)_2\}$.The result can be expressed as:

$$\begin{cases} e_3=a_3\oplus a_2 \\ e_2=a_3\oplus a_1 \\ e_1=a_3\oplus a_0 \\ e_0=a_3 \end{cases} \tag{7}$$

And the multiplying-constant needs 3XOR gates. If we merge the Squ and MulC together as square-multiplying-constant (SquMulC), we can get the result $\boldsymbol{f}=\{f_3, f_2, f_1, f_0\}$ as follows:

$$\begin{cases} f_3=a_1 \\ f_2=a_3 \\ f_1=a_0 \\ f_0=a_2 \end{cases} \tag{8}$$

From Eq.(8), we can see that the merged SquMulC does not consume hardware resources.

When we make inverse (Inv) of $\boldsymbol{a}$, we can get the result expression $\boldsymbol{g}=\{g_3, g_2, g_1, g_0\}$ as follows:

$$\begin{cases} g_3=a_2\oplus a_1\oplus a_3a_1\oplus a_2a_0\oplus a_3a_2a_0\oplus a_3a_1a_0 \\ g_2=a_3\oplus a_1\oplus a_2a_0\oplus a_1a_0\oplus a_3a_2a_0\oplus a_2a_1a_0 \\ g_1=a_2\oplus a_3a_0\oplus a_2a_1\oplus a_2a_0\oplus a_3a_2a_1\oplus a_3a_1a_0\oplus a_2a_1a_0 \\ g_0=a_1\oplus a_0\oplus a_3a_2\oplus a_2a_0\oplus a_3a_2a_1\oplus a_3a_2a_0 \end{cases} \tag{9}$$

So, the Inv of $\boldsymbol{a}$ needs 21XOR gates and 27AND gates.

At last, we revert the computing results to $GF(2^8)$ and perform the affine operation, we represent the operation using InvMapping, the result is expressed as:

$$\begin{cases} y_7=x_3\oplus x_2 \\ y_6=x_7\oplus x_5\oplus x_4 \\ y_5=x_5\oplus x_1 \\ y_4=x_7\oplus x_4\oplus x_2\oplus x_1\oplus x_0 \\ y_3=x_7\oplus x_2\oplus x_0 \\ y_2=x_7\oplus x_6\oplus x_3\oplus x_2\oplus x_1\oplus x_0 \\ y_1=x_7\oplus x_4\oplus x_1\oplus x_0 \\ y_0=x_6\oplus x_2\oplus x_0 \end{cases} \tag{10}$$

iii) Step3:Resource Optimization

In order to reduce the hardware overhead, we adopt the DACSE algorithm to optimize the operations appeared in the composite field S-box. The DACSE algorithm consider both the hardware overhead and the delay of the operation, which is very helpful for realizing our pipeline strategy. We take the Mul as an example to illustrate the optimization.

The optimization of the Mul operation is expressed as:

$$\begin{cases} R_1=a_3b_2\oplus a_2b_3 \\ R_2=a_3b_1\oplus a_1b_3 \\ R_3=a_3b_0\oplus a_0b_3 \\ R_4=a_2b_1\oplus a_1b_2 \\ R_5=a_2b_0\oplus a_0b_2 \\ R_6=a_1b_0\oplus a_0b_1 \end{cases}, \begin{cases} r_3=a_3b_3 \\ r_2=a_2b_2 \\ r_1=a_1b_1 \\ r_0=a_0b_0 \end{cases} \tag{11}$$

Then the product of $\boldsymbol{a}$ and $\boldsymbol{b}$ is:

$$ab=\begin{cases} c_3=R_2\oplus R_3\oplus R_4\oplus r_2 \\ c_2=R_2\oplus R_5\oplus r_2\oplus r_1 \\ c_1=R_2\oplus R_6\oplus r_3\oplus r_2 \\ c_0=R_1\oplus R_2\oplus r_2\oplus r_0 \end{cases} \tag{12}$$

Then we let

$$S=R_2\oplus r_2 \tag{13}$$

So, result is:

$$\begin{cases} c_3=S\oplus R_3\oplus R_4 \\ c_2=S\oplus R_5\oplus r_1 \\ c_1=S\oplus R_6\oplus r_3 \\ c_0=S\oplus R_1\oplus r_0 \end{cases} \tag{14}$$

The optimized Mul operation occupies 15XOR gates and 16AND gates, and the delay is $3T_{XOR}+1T_{AND}$.

The hardware consumption of each unit is shown in TABLE.I. It is obvious that the reduction of hardware overhead is quite significant with the optimized method.

B. High Throughput Implementation

In order to implement high efficiency AES circuit, we need not only to reduce hardware overhead, but also improve the circuit throughput. T7he method to reduce hardware overhead has been introduced and it is the basis of our pipeline strategy. The point is how to use the pipeline reasonably to boost the circuit at the low cost of hardware resources.

First, we need to analyze the delay of each arithmetic unit. The delay of every optimized arithmetic is shown in TABLE.II.

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE.I HARDWARE CONSUMPTION OF EACH UNIT

Arithmetic	Unoptimized Overhead		Optimized Overhead	
	XOR	AND	XOR	AND
Mapping	22	0	11	0
Mul	21	25	15	16
Squ	3	0	3	0
MulC	3	0	3	0
SquMulC	6	0	0	0
Inv	21	27	16	10
InvMapping	20	0	13	0

TABLE.II DELAY OF EACH ARITHMETIC UNIT IN S-BOX

Arithmetic	Mapping	XOR	Mul	SquMulC	Inv	InvMapping
Delay	$3T_{XOR}$	$1T_{XOR}$	$3T_{XOR}+1T_{AND}$	0	$3T_{XOR}+2T_{AND}$	$4T_{XOR}$

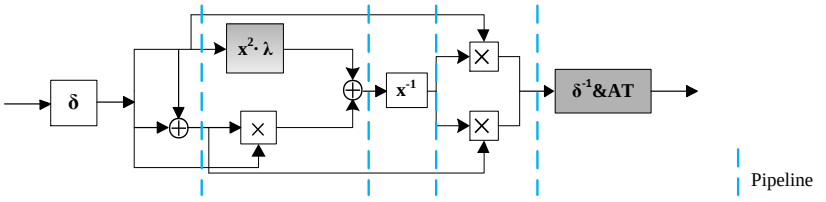


Fig.4 Proposed Pipeline S-box

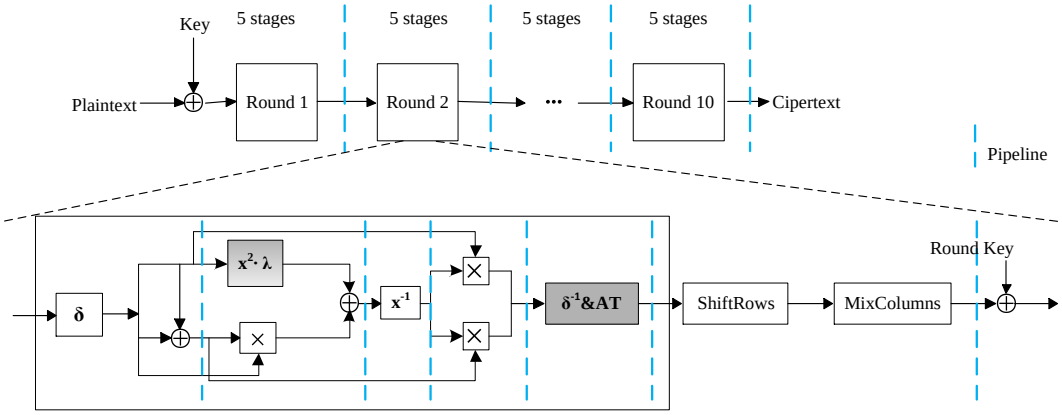


Fig.5 Proposed Pipeline AES Round Transformation

Based on the delay shown in TABLE.II, we can insert pipeline in S-box. The pipelined S-box is shown in Fig.4. The structure in Fig.2 guarantees that the delay of each pipeline stage is nearly equivalent, so, it will not bring the unnecessary hardware overhead. In Ref. [5] [6] and [11], the S-box is pipelined. However, they did not analyze the delay of each arithmetic unit in S-box, the pipeline strategy was not reasonably enough resulting in the extra hardware overhead.

Based on the pipelined S-box, we design the pipelined AES circuit, whose round transformation is demonstrated in Fig.5.

First, we insert pipeline among each round transformation forming 10 extern pipeline stages. Inside the round transformation, we insert a pipeline after the SubBytes forming 5 inner pipeline stages, that's because the delay of MixColumns is $3T_{XOR}$, the delay of AddRoundKey is $1T_{XOR}$, and the total delay is $4T_{XOR}$. The total pipeline stages in our implementation are 60.

Our pipeline strategy can make sure the delay of each pipelining stage nearly equivalent. This will improve the throughput meanwhile taking no extra hardware overhead.

V. COMPARISION AND RESULTS

The AES designed in this paper has been implemented using the Verilog HDL language. We use Modelsim-10.4 for simulation, ISE-14.4 for synthesis and Place and Route. We instantiate the design in Xilinx Virtex4 xc4vLX100, Xilinx Virtex5 xc5vLX85 and Xilinx Virtex6 xc6vLX240T platforms. The implementation results are shown in TABLE.III.

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE.III THE IMPLEMENTATION RESULTS AND COMPARISION WITH PREVIOUS WORKS

Works	Platform	Frequency(MHz)	Throughput(Gbps)	Slices	Efficiency(Mbps/Slice)
Ref.[1]	XC4VLX200	250	32	86806	0.37
Ref.[2]	XC2VP2	306	0.876	156	5.62
Ref.[3]	XC5VLX85	576.07	73.74	22994	3.21
Ref.[4]	XC5VLX110T	198.26	22.89	14522	2.11
Ref.[6]	XC6VLX240T	764.059	97.8	10760	9.08
This Work	XC4VLX200	544.959	69.75	9857	7.08
	XC5VLX85	552.425	70.71	5237	13.5
	XC6VLX240T	730.727	93.53	5081	18.41

Our implementation can reach the throughput of 79.98Gbps, 70.71Gbps and 93.53Gbps at the cost of 11722Slices, 5237Slices and 5081Slices respectively.

In Ref. [2], the AES is implemented in Loop structure, although it occupied little hardware resource, its throughput was too low. Compared to Ref.[1] and [4], our implementation improve a lot both in throughput and hardware overhead, that's because their S-boxs are not designed in composite field, and the pipeline stages are not enough. Compared to Ref.[3] [6], although our throughput is slightly lower, our hardware consumption is much less than they do, so our design is much more efficient. They don't optimize the arithmetic in the S-box, and their pipeline strategies are not readonable enough causing the extra hardware overhead.

VI. CONCLUSION

In this paper, we design a 128-bit unrolled AES based on FPGA, the S-box is implemented based on the composite field. The CSE algorithm is used to optimize the mapping, multiplication, inverse and inverse mapping units in the S-box to reduce the hardware overhead. Then we analyze the delay of the round transformation, to apply a reasonable pipeline strategy to ensure the best performance. The AES designed in this paper can reach frequency of 730.727MHz in the Virtex6 xc6vLX240T platform, the throughput is 93.53Gbps at the cost of 5081 Slices and the efficiency of the implementation is 18.41 Mbps/Slice.

REFERENCES

[1] C. P. Fan, and J. K. Hwang. "Implementations of high throughput sequential and fully pipelined AES processors on FPGA" International Symposium on Intelligent Signal Processing and Communication Systems IEEE, 2007: pp, 353-356.
[2] C. Chang and C. Hwang. "High throughput 32-bit AES implementation in FPGA" Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on IEEE, 2009: pp, 1806-1809.
[3] S. Qu, and G. Shou. "High Throughput, Pipelined Implementation of AES on FPGA" International Symposium on Information Engineering and Electronic Commerce IEEE, 2009: pp, 542-545.
[4] M. Vanitha., R. Sakthivel, and Subha. "Highly secured high throughput VLSI architecture for AES algorithm" International Conference on Devices, Circuits and Systems IEEE, 2012: pp, 403-407.
[5] D. Chen, and G. Shou et al. "Efficient architecture and implementations of AES" International Conference on Advanced Computer Theory and Engineering IEEE, 2010.
[6] Abolfazl Soltani, and S. Sharifian. "An ultra-high throughput and fully pipelined implementation of AES algorithm on FPGA" Microprocessors & Microsystems 39.7(2015): pp, 480-493.
[7] M. E. Mohamed, and S. F. Babiker. "An efficient implementation of a fully combinational pipelined S-Box on FPGA" Basic Sciences and Engineering Studies IEEE, 2016: pp, 57-63.
[8] H. Trang, and V. L. Nguyen. "An Efficient FPGA Implementation of the Advanced Encryption Standard Algorithm" IEEE Rivf International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future IEEE, 2012: pp, 1-4.
[9] D. Canright, "A Very Compact S-Box for AES" Lecture Notes in Computer Science 3659(2005): pp, 441-455.
[10] Xiaoqiang Zhang, et al. "An optimized delay-aware common subexpression elimination algorithm for hardware implementation of binary-field linear transform." Ieice Electronics Express 11.22(2014): pp ,20140934-20140934.
[11] M. M. Wong, M. L. D. Wong. "Construction of optimum composite field architecture for compact high-throughput AES S-boxes" IEEE Transactions on Very Large Scale Integration Systems 20.6(2012): pp, 1151-1155.
[12] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "A new algorithm for elimination of common subexpressions". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 18, No. 1, 1999, pp.58-68.

ISBN: 978-988-14047-5-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)