Design and Evaluation of AES Encryption Circuits with Various S-Box Implementations

Taosong Zhao [1] Hiroki Nishikawa [2] Xiangbo Kong [3] Hiroyuki Tomiyama [1]

[1] *Graduate School of Science and Engineering, Ritsumeikan University, Kusatsu, Japan*
[2] *Graduate School of Information Science and Technology, Osaka University, Suita, Japan*
[3] *Faculty of Engineering, Toyama Prefectural University, Imizu, Japan*

Abstract

AES is an encryption standard approved by the National Institute of Standards and Technology (NIST) to replace the DES cipher. Since its approval in 2001, AES has been widely used, and various studies on its performance and security have been conducted so far. The S-Box, a fixed table used in the SubBytes process and the only non-linear transformation in the AES encryption algorithm, plays a crucial role. In this work, we design a number of AES circuits with different S-Box generation methods, and investigate their impacts on hardware cost, performance and power consumption.

Keywords

AES, S-Box, high-level synthesis, FPGA

1. Introduction

To ensure secure message interactions, it is necessary to encrypt messages when they are sent. AES (Advanced Encryption Standard) is a symmetric encryption algorithm that was certified by the National Institute of Standards and Technology in 2001 to replace the less secure DES encryption algorithm. AES was developed by Joan Daemen and Vincent Rijmen and designed to be efficient in both hardware and software [1][2][3][4]. AES supports a block length of 128 bits and key lengths of 128, 192, and 256 bits. The AES encryption flow is outlined in Figure 1. It consists of four main parts: Key Expansion, Initial Round Key Addition, 9, 11, or 13 rounds of encryption, and Final Round Encryption. For a 128-bit key length, the total encryption rounds would be 10. AES operates on a 4×4 column-major order array of 16 bytes. The key size used for an AES cipher specifies the number of transformation rounds that convert the input, called plaintext, into the final output, called ciphertext.

For each encryption round, AES includes four parts in order: SubBytes, ShiftRows, MixColumns, and AddRoundKeys. With a key length of 128 bits, the encryption process is the same for the first 9 rounds, and the MixColumns step is omitted for the last round of encryption. In the SubBytes step, each byte in the state array is replaced with a SubBytes value using an 8-bit substitution box (S-Box). The SubBytes step is the only non-linear process within AES encryption. The ShiftRows step cyclically shifts the bytes in each row of the state by a certain offset. In the MixColumns step, the four bytes of each column of the state are combined using a reversible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. In the AddRoundKeys step, the sub-key is combined with the state. For each round, a sub-key is derived from the main key using Rijndael's key schedule. Each sub-key is the same size as the state. The sub-key is added by combining each byte of the state with the corresponding byte of the sub-key using bitwise XOR.

ATAIT 2023: The 5th International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT 2023), August 28–29, 2023, Kusatsu, Japan

Email: gr0571he@ed.ritsumei.ac.jp (T.Zhao);kong@pu-toyama.ac.jp (X. Kong); nishikawa.hiroki@ist.osaka-u.ac.jp (H. Nishikawa); ht@fc.ritsumei.ac.jp (H. Tomiyama)

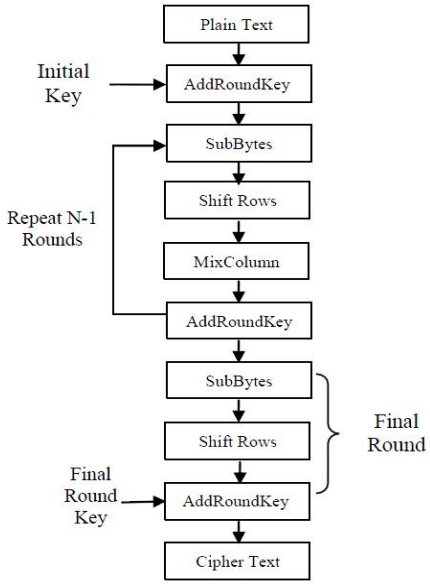

Figure 1: Flowchart of AES encryption

In this work, we design a number of AES circuits with different S-Box generation methods for FPGA and investigate their impact on hardware cost, performance and power consumption of the circuits. The contribution of this work is quantitative evaluation of existing S-Box generation methods in terms of hardware resources, power consumption and critical path delay. In addition, we propose a modified method to further reduce hardware resources.

The remainder of the paper is organized as follows. Section 2 outlines our flow for designing AES encryption circuits and describes the different methods used to produce the S-Box. Section 3 evaluates the designed AES circuits, and Section 4 concludes the paper.

2. AES Encryption Circuit Design

This section describes how to design AES encryption circuits with different S-Box methods. The AES circuits are designed based on two design flows: high-level synthesis and register-transfer level (RTL) design. We use Xilinx's Zynq-7000 SoC device as the target hardware, Vitis HLS for high-level synthesis to generate RTL code, and Vivado for logic synthesis and simulation to evaluate hardware resources, performance and power consumption.

2.1. Overall Design Flow

As a first step, we use the Vitis HLS tool to generate an AES encryption circuit from a C/C++ program, with the S-Box implemented as a simple look-up table (LUT). This initial design serves as a baseline for subsequent designs. We then design eight S-Box circuits, either using Vitis HLS or manually coding in Verilog, and replace the initial LUT-based S-Box with the new ones, while the remaining modules are unchanged.

2.2. S-Box Generation Methods

The S-Box, shown in Figure 2, is one of the most important components of AES encryption, which maps an 8-bit input to an 8-bit output, and is interpreted as polynomials over GF(2). The S-Box is used for byte substitution operations, which are the only nonlinear operations in AES encryption and have a

significant impact on the overall performance and security of the algorithm. Thus, the method used to generate the S-box affects the comprehensive performance and security of AES encryption.

Table 1: The AES encryption S-Box

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

We have tested two flows to generate S-Box circuits. One uses high-level synthesis, and the other is manual RTL coding in Verilog.

2.2.1. Look-up Table

First, we employ a byte substitution method using direct query of the S-Box table [5][6]. In this method, the 256 fixed values are stored in advance in a 16×16 look-up table, and the pre-stored values are directly called when byte substitution is performed. Using the table look-up method for byte substitution, the pre-stored values are directly used without the need to calculate the values in the S-Box.

2.2.2. Composite Field Arithmetic

Next, we design S-Box circuits based on composite field arithmetic (CFA). Specifically, we employ Galois fields GF(2^2) [7][8] and GF(2^4) [9][10]. Since the computation of S-Box in GF(2^8) is hardware intensive, CFA of order GF((2^4)^2) is used to reduce hardware requirement [8].

Since the computation of multiplicative inverse in composite fields cannot be directly applied to an element α, which is based on GF(2^8), it has to be mapped to its composite field representation via an isomorphic function given by:

$$\delta \times \alpha = \begin{bmatrix} 10100000 \\ 11011110 \\ 10101100 \\ 10101110 \\ 11000110 \\ 10011110 \\ 01010010 \\ 01000011 \end{bmatrix} \times \begin{bmatrix} \alpha_7 \\ \alpha_6 \\ \alpha_5 \\ \alpha_4 \\ \alpha_3 \\ \alpha_2 \\ \alpha_1 \\ \alpha_0 \end{bmatrix} \quad (1)$$

Similarly, after performing the multiplicative inversion, the result has to be mapped back from its composite field representation to its equivalent in GF(2^8) using the inverse isomorphic function δ^{-1}. The matrix δ^{-1} required to map an element β is shown as follows:

$$\delta^{-1} \times \beta = \begin{bmatrix} 11100010 \\ 01000100 \\ 01100010 \\ 01110110 \\ 00111110 \\ 10011110 \\ 00110000 \\ 01110101 \end{bmatrix} \times \begin{bmatrix} \beta_7 \\ \beta_6 \\ \beta_5 \\ \beta_4 \\ \beta_3 \\ \beta_2 \\ \beta_1 \\ \beta_0 \end{bmatrix} \quad (2)$$

2.2.3. Positive Polarity Reed-Muller

The main disadvantage of S-Box realization using CFA is the complex signal paths, which can result in long signal delays. To address this issue, Positive Polarity Reed-Muller (PPRM) methods have been proposed [11][12]. With PPRM, the S-Box is converted into two-level logic using only arrays of XOR and AND gates. The main advantage of this approach is that it reduces dynamic hazards that can occur in combination circuits.

The PPRM-based SubBytes transformation architecture is shown in Figure 2. These three stages of operation take place at the three different stages of the SubBytes transformation, which are called pre-inversion, inversion, and post-inversion stages. All three stages can be realized using an array of AND and XOR gates. In the second stage, only one of the first stage's outputs needs to be computed, and the result will be input into the third stage.

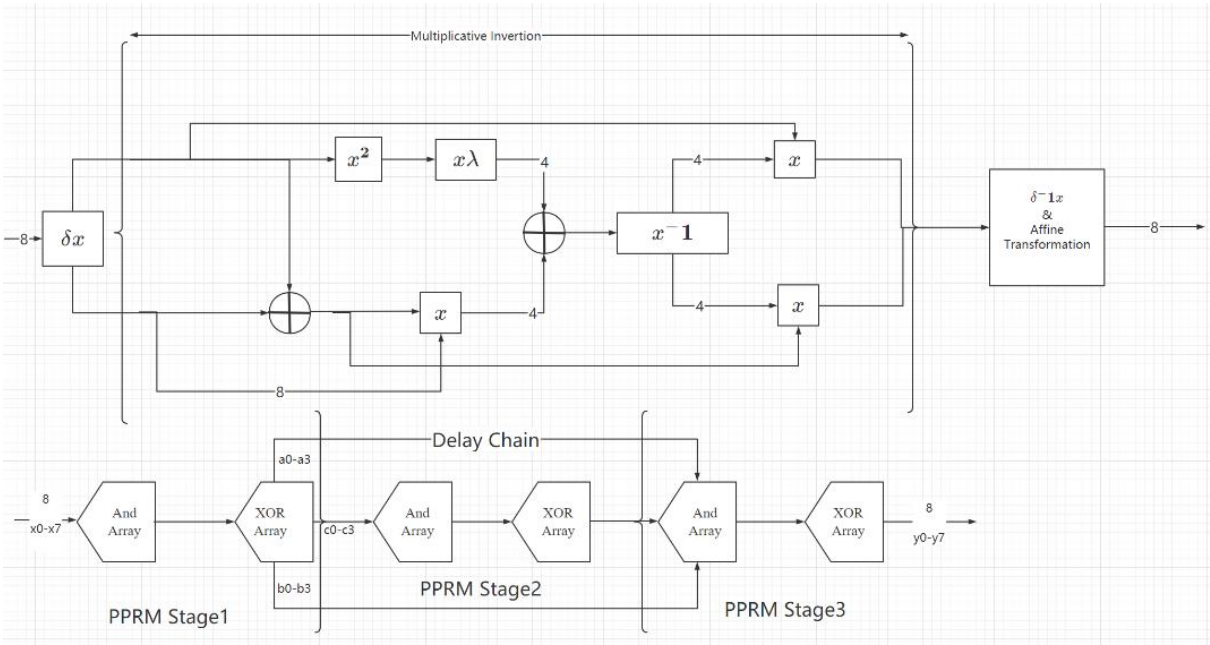

Figure 2: PPRM-based SubByte transformation [11]

In this work, we make further modification to the PPRM architecture. Fragment of Verilog code for the first stage of PPRM is shown in Listing 1, where we can identify some redundant computations.

Listing 1: Verilog code for PPRM stage 1

```
c3 = (x5&x1)^(x7&x1)^(x5&x2)^(x5&x6)^(x5&x7)^(x5&x4)^(x7&x4)^(x5&x0)^(x7&x0)
     ^(x3&x1)^ (x4&x1)^(x3&x2)^(x2&x4)^(x4&x6)^(x2&x1)^(x2&x6)^(x6&x1);
c2 = (x6&x1)^(x2&x6)^(x3&x6)^(x7&x6)^(x1&x0)^(x2&x0)^(x3&x0)^(x4&x0)^(x6&x0)
     ^(x7&x0)^ (x5&x2)^(x5&x3)^(x2&x4)^(x3&x4)^(x5&x7)^(x7&x2)^(x5&x6)^(x3&x2)
     ^(x7&x3);
c1 = (x2&x1)^(x2&x4)^(x5&x4)^(x3&x6)^(x5&x6)^(x2&x0)^(x3&x0)^(x5&x0)^(x7&x0)
     ^(x5&x2)^(x7&x2)^(x5&x3)^(x5&x7)^(x3&x2)^x1^x2^x4^x5^x7;
c0 = (x1&x0)^(x2&x0)^(x3&x0)^(x5&x0)^(x7&x0)^(x3&x1)^(x6&x1)^(x3&x6)^(x5&x6)
     ^(x7&x6)^(x3&x4)^(x7&x4)^(x5&x3)^(x4&x1)^x2^(x3&x2)^(x4&x6)^x6^x5^x3^x0;
```

We extract common computations which appear three times or more, and reuse the computation results for the second appearance and later as shown in Listing 2. In Listing 2,we take PPRM stage 1 as an example. We made the same optimizations for the remaining PPRM stage 2 and stage 3.

Listing 2: Modified Verilog code for PPRM stage 1

```
temp1 = (x5&x6)^(x7&x0)^(x3&x2);
temp2 = (x5&x7)^(x5&x2)^(x2&x4);
c3    = (x7&x1)^(x5&x1)^(x5&x4)^(x7&x4)^ (x5&x0)^(x3&x1)^(x4&x1)^(x4&x6)^(x2&x1)
        ^(x2&x6)^(x6&x1)^temp1^temp2;
c2    = (x6&x1)^(x2&x6)^(x3&x6)^(x7&x6)^ (x1&x0)^(x2&x0)^(x3&x0)^(x4&x0)^ (x6&x0)
        ^(x5&x3)^(x3&x4)^(x7&x2)^ (x7&x3)^temp1^temp2;
c1    = (x2&x1)^ (x5&x4)^(x3&x6)^(x2&x0)^(x3&x0)^(x5&x0)^(x7&x2)^(x5&x3)^ (x7^x2)
        ^(x4^x5)^x1^temp1^temp2;
c0    = (x1&x0)^(x2&x0)^(x3&x0)^(x5&x0)^(x3&x1)^(x6&x1)^(x3&x6)^(x7&x6)^(x3&x4)
        ^(x7&x4)^(x5&x3)^(x4&x1)^(x4&x6)^x2^x6^x5^x3^x0^temp1;
```

3. Evaluation

We have designed nine AES encryption circuits, synthesized them for FPGA, and run simulation using the Vivado toolkit. A clock period constraint was set to be 10 ns. The number of clock cycles executed was the same among the nine circuits, as only the S-Box implementation differs between them.

The synthesis results are summarized in Table 2. Among the four existing methods to implement the S-Box, PPRM-based implementation is the smallest in size. Our modified PPRM-based circuit outperforms the original PPRM-based one in terms of size. It is also observed that HLS-generated circuits are larger in size and consumes more power than manually-designed ones. On the reasons for this outcome, we hypothesize that it is possible that the circuits automatically generated by the high level synthesis method contain some redundant designs, resulting in consuming more hardware resource consumption. In VHDL design, we can have a good grasp of the use of hardware resources, which is more favorable for the design of the area constraints.

Table 2: Synthesis results

		Hardware resources		Power (mW)	Delay (ns)
		Slices	BRAMs		
Look-up Table	HLS	494	2	102.01	6.624
	RTL	369	0	94.11	6.358
CFA GF(2^2)	HLS	922	2	103.60	8.365
	RTL	422	0	96.38	9.102
CFA GF(2^4)	HLS	922	3	104.00	8.247
	RTL	632	0	96.40	7.025
PPRM	HLS	1,171	3	102.65	9.657
	RTL	359	0	95.83	6.073
Modified PPRM	RTL	353	0	95.84	6.028

4. Summary

In this work, we have designed and evaluated nine AES circuits, where design flows and S-Box generation methods are different. The synthesis results show that the PPRM-based circuit modified in this work is the best among the nine in terms of hardware cost. However, at this stage we have not characterized the possible negative effects of these methods tested on the whole circuit.

In future, we plan to further optimize the circuit and adjust the clock cycles. Also, we plan to evaluate hardware security in addition to hardware cost, performance and power consumption.

Acknowledgements

This work is supported partly by KAKENHI 20H00590, 21K19776 and 22K21276.

References

[1] Yulin Zhang, Xinggang Wang, "Pipelined Implementation of AES Encryption Based on FPGA," *International Conference on Information Theory and Information Security*, 2010.

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

[3] Mohammad A. Musa, Edward F. Schaefer, Stephen Wedig, "A Simplified AES Algorithm and Its Linear and Differential Cryptanalyses," *Cryptologia*, vol. 27, no. 2, 2003.

[4] Chen Yicheng, Zou Xuecheng, Liu Zhenglin, Han Yu, Zheng Zhaoxia, "Energy-efficient and Security-optimized AES Hardware Design for Ubiquitous Computing," *Journal of Systems Engineering and Electronics*, vol. 19, no. 4, 2008.

[5] Yuko Hara, Hiroyuki Tomiyama, Shinya Honda, Hiroaki Takada, "Proposal and Quantitative Analysis of the CHStone Benchmark Program Suite for Practical C-based High-level Synthesis," *Journal of Information Processing*, vol. 17, pp. 242-254, 2009.

[6] Pawel Chodowiec, Kris Gaj, "Very Compact FPGA Implementation of the AES Algorithm," *International Workshop on Cryptographic Hardware and Embedded Systems*, 2003.

[7] Aditya Pradeep, Vishal Mohanty, Adarsh Muthuveeru Subramaniam, Chester Rebeiro, "Revisiting AES SBox Composite Field Implementations for FPGAs," *IEEE Embeded Systems Letters*, vol. 11, no. 3, 2019.

[8] Rashmi Ramesh Rachh, B. S. Anami, P. V. Ananda Mohan, "Efficient Implementations of S-Box and Inverse S-Box for AES Algorithm," *IEEE Region 10 Conference*, 2010.

[9] Ahmet Dogan, S. Berna Ors, Gokay Saldamli, "Analyzing and Comparing the AES Architectures for their Power Consumption," *Journal of Intelligent Manufacturing*, vol. 25, 2014.

[10] Thanikodi Manoj Kumar, Kasarla Satish Reddy, Stefano Rinaldi, Bidare Divakarachari Parameshachari, Kavitha Arunachalam, "A Low Area High Speed FPGA Implementation of AES Architecture for Cryptography Application," *Electronics*, vol. 10, no. 16, 2023.

[11] T. Manojkumar, P. Karthigaikumar, Varatharajan Ramachandran, "An Optimized S-Box Circuit for High Speed AES Design with Enhanced PPRM Architecture to Secure Mammography Images," *Journal of Medical Systems*, vol. 43, 2019.