The Design of GF $(2^4)^2$ AES Encryption based on FPGA

W. Suntiamorntut, S. Charoenpanyasak, W. Wittayapanpracha, J.Nopparat, S. Julrat, M.Manopirol
Centre of Excellence in Wireless Sensor Networks,
Department of Computer Engineering, Faculty of Engineering,
Prince of Songkla University, Hatyai Songkhla Thailand, 90112

{wannarat@coe.psu.ac.th}

Abstract—This paper presents the results of our study in the energy efficient AES encryption using S-box based on Galoisfield (GF) aiming for a wireless FPGA node. The mapped composite-field GF(2_4)$_2$ has been applied to compute S-box. In order to implement the energy efficient AES on FPGA for IoT Devices, the several architectures such as pipelining and parallelism AES have been compared and analyzed. The low power design technique named gating is also applied to enable the necessary logics. By using this technique, we can gain the power saving about 19 percentages. In addition, the AES based on GF(2_4)$_2$ gives the throughput per milli-Watts better than the AES based on LUT about 33 percentages. From the results, the AES based on GF (2_4)$_2$ has been therefore chosen to build on a wireless reconfigurable node instead of using a basic structure; LUT in FPGA.

Keywords— AES; Galois field (GF); FPGA;

I. Introduction

Wireless sensor networks technology [1] has been introduced and widely used since the early 2000. Many applications such as precision agriculture, military, warning disaster and health care monitoring systems have deployed wireless sensor networks. The wireless sensor network consists of several *nodes* or motes which comprise a microcontroller, a radio transceiver, sensors and energy source (usually a battery). According to the limitation of battery, the energy usage has to be paid attention to seriously. Using the reconfigurable devices in wireless sensor networks [2] has great potential to enhance the processing power and reduce the overall energy consumption, especially in the respects of security and routing. This is a consistency of the use of reconfigurable device such as Field Programmable Gate Array (FPGA) as a sensor node for health care application to perform both signal processing and encryption/decryption.

The Advanced Encryption Standard (AES) is one of the top data encryptions in wireless sensor networks [3]. AES has been already appeared in the IEEE 802.15.4 and ZigBee standard [4]. The minimum performance of AES is set to the maximum data rate (250 kbps) of the IEEE 802.15.4 specification. To meet the lower limit requirement of AES in wireless sensor networks, the operation has to be completed as fast as possible. Meanwhile the energy consumption has to be considered carefully during the design process.

The traditional AES consists of four operations, ShiftRow, SubByte, MixColumn and AddRoundKey, whereas SubBytes is the bottleneck of AES operation. The 16-byte data (or 128 bits) as shown in Fig.1 are taken to four processes. In AddRoundKey process, the key can be varied into 128, 196 and 256 bits and be added to the plaintext. All processes have been repeated 10, 12 or 14 rounds depending on the security requirements. In the beginning, Cipher Key is taken to Expand Key performing rotate word, sub-word, round constant and word column. The expanded key operation will be described in more details in Section IV.

SubByte or S-box function is able to be represented using a multiplicative inverse of GF(2^8) and affine transformation. A multiplicative inverse of GF(2^8) can be implemented by mapping GF(2^4)^2 in order to achieve the performance and energy efficiency [5]. Even, the optimized GF(2^4)^2 AES was proposed in [6]. Recently, the circuit was aiming for an ASIC implementation. According to the advantage of reconfigurable architecture, AES based on Look-Up Table (LUT) can be a good candidate. Thus, both AES encryption circuits based on GF(2^4)^2 [5] and LUT have been studied and considered in this paper. We are going to compare and analyse both circuits in terms of throughput per mW to find out the suitable AES circuits for wireless sensor network applications. Moreover, we also compare and analyse the pipelining and parallel AES. The rest of this paper is organized as follows: the conventional AES algorithm is described in Section II; the S-box operation, multiplicative inverse based on GF(2^4)^2, pipelining,parallelism and the low power design technique are discussed in Section III; the performance and energy efficiency of the experiment results simulated based on virtex-5 (XC5VLX50) in Xilinx ISE and XPower Analyzer tool are explained in Section IV; and the paper is concluded in SectionV.

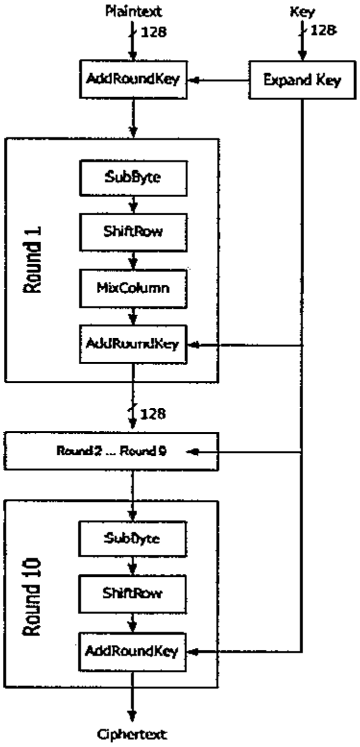
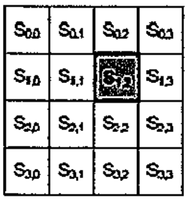


Figure 1. AES Operations

II. AES Algorithms

As shown in Fig.1, each round of AES algorithm has been divided into four operations, the details of each operation are described as follows:



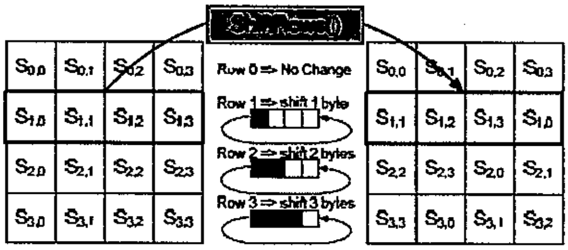
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Figure 2. ShiftRow Operations

A. Substitute Bytes

AES uses an array of byte sized 4x4 named states. Each state is denoted with $S_{r,c}$, where "r" is row and "c" is column as shown in Fig. 2. This function block is called SubByte. All bytes of the state are computed using look-up table called Sbox. The substitute data shown in Table I are the hexadecimal numbers. The value of $S_{r,c}$ can be found by looking up the data from a table from X cross Y. For example, $S_{1,2}= \{5_x3_y\} = \{ed\}$. Normally, the table will be implemented using memory structure. However, all entries in table can be calculated using an inversion in the finite field $GF(2^8)$.

B. Shift Row



$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

Figure 3. ShiftRow operations

This function performs a simple shift-left as shown in Fig.3. The first row (Row0) does nothing while the second byte of the second row (Row1) is moved to the left. In the third row (Row2) and forth row (Row3), the data have been rotated 2 and 3 bytes, respectively.

C. MixColumn

The Mixcloumn is a multiplication operation; the product of a column of state and a finite field constant as shown in (1):

$$a(x) = \{03\}(x)^3 + \{01\}(x)^2 + \{01\}x + \{02\} \qquad (1)$$

The result of this operation is $s'_{r,c}$ as following:

$$\begin{aligned} s'_{0,c} &= (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &= s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}) \end{aligned}$$

D. AddRoundKey

AddRoundKey performs bitwise XOR between the data state and the key from key expansion.

III. S-box based on $GF(2^4)^2$

In AES, the throughput and power are mainly determined by S-box. The implementation of S-box is very important because it can influence the speed and power dissipation of AES circuits. Normally, S-box can be easily built by look-up table which is suited to the reconfigurable device. However, there is another argument to replace the look-up table by the combinational logics of finite filed $GF(2^8)$. Unfortunately, the circuits of $GF(2^8)$ is complex. Thus, S-box based on $GF(2^4)^2$ is widely used instead.

The ordinary S-box operation is shown in Fig.4. The encrypt signal has been used to switch between encryption and decryption mode. If the encrypt signal is set to '1', the encryption process will begin from bypassing the inverse affine transformation (Aff_tran-1) to multiplicative inverse directly. The output of multiplicative inverse will perform in the affine transformation. The gating technique has been applied here by inserting 'AND' gate in front of Aff_tran-1 and Aff_trans-blocks in order to stop the unnecessary signals propagate to those two blocks. This will get rid of the waste energy according to the logic switching inside the unwanted block. Affine transformation and inverse affine transformation are computed using the equations in [5] as shown in (2) and (3).

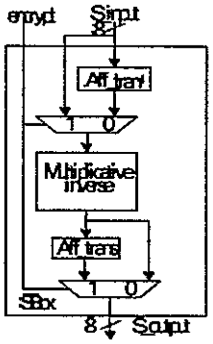


Figure 4. S-box operationg with gating technique

$$
\begin{aligned}
&q = aff\_trans(a) &(2)\\
&a_A = a_0 \oplus a_1,\ a_B = a_2 \oplus a_3,\\
&a_C = a_4 \oplus a_5,\ a_D = a_6 \oplus a_7\\
&q_0 = \overline{a_0} \oplus a_C \oplus a_D\\
&q_1 = \overline{a_5} \oplus a_A \oplus a_D\\
&q_2 = a_2 \oplus a_A \oplus a_D\\
&q_3 = a_7 \oplus a_A \oplus a_B\\
&q_4 = a_4 \oplus a_A \oplus a_B\\
&q_5 = \overline{a_1} \oplus a_B \oplus a_C\\
&q_6 = \overline{a_6} \oplus a_B \oplus a_C\\
&q_7 = a_3 \oplus a_C \oplus a_D
\end{aligned}
$$

$$
\begin{aligned}
&q = aff\_trans^{-1}(a) &(3)\\
&a_A = a_0 \oplus a_5,\ a_B = a_1 \oplus a_4,\\
&a_C = a_2 \oplus a_7,\ a_D = a_3 \oplus a_6\\
&q_0 = \overline{a_5} \oplus a_C\\
&q_1 = a_0 \oplus a_D\\
&q_2 = \overline{a_7} \oplus a_B\\
&q_3 = a_2 \oplus a_A\\
&q_4 = a_1 \oplus a_D\\
&q_5 = a_4 \oplus a_C\\
&q_6 = a_3 \oplus a_A\\
&q_7 = a_6 \oplus a_B
\end{aligned}
$$

S-box has been designed based on pipelining architecture in order to increase the throughput by dividing the operation into several small steps corresponding to the available hardware resources. Normally, the pipelining architecture can achieve a high performance by applying a faster clock frequency. This method allows us to increase the throughput of the circuits with a small amount of logics rather than the parallel architecture. Only the latches are the overhead of pipelining that required to store the data between each state.

This paper proposes five states pipelining S-box as shown in Fig.5. We have implemented both 8-bit and 32-bit pipeline to explore which one can give the best energy efficiency of Sbox.

In 8-bit pipelining, each 8-bit input or one byte (state $S_{0,0}$ or $S_{0,1}$, ... or $S_{3,3}$) is fetched to the state in pipelining. Therefore, we need 21 clock cycles to finish one state as shown in Fig.6. In 32-bit pipelining, each 32-bit input or one row is fetched to the pipelining and takes 9 clock cycles to finish one state as shown in Fig.7.

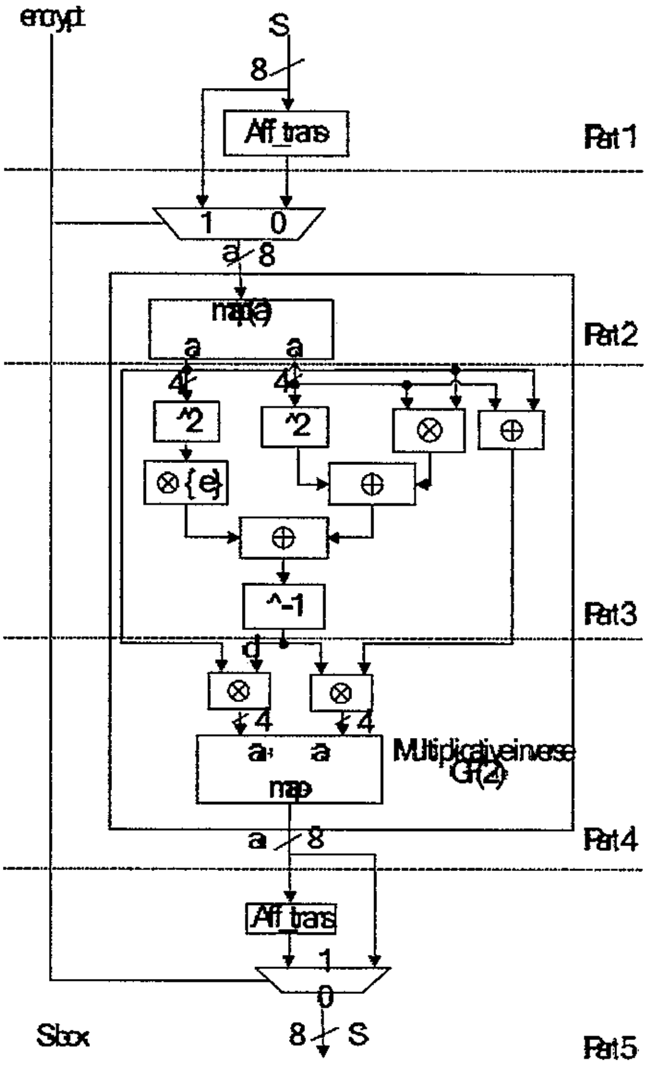


Figure 6. Five state pipeline of S-box

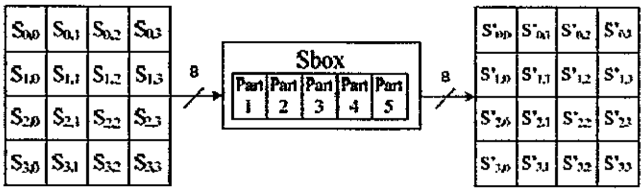


Figure 7. 8-bit S-box pipelining

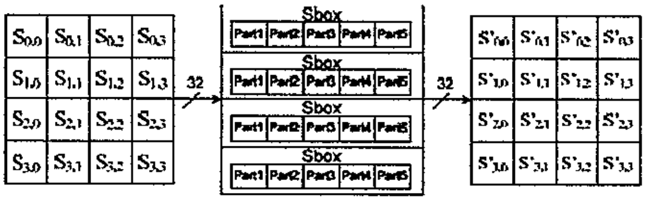

$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
32
Sbox
Part1 Part2 Part3 Part4 Part5
Sbox
Part1 Part2 Part3 Part4 Part5
Sbox
Part1 Part2 Part3 Part4 Part5
Sbox
Part1 Part2 Part3 Part4 Part5
32
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{1,1}$ $S'_{1,2}$ $S'_{1,3}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$

Figure 8. 32-bit S-box pipelining

In FPGA or reconfigurable device, the design can take an advantage of parallel structure for performing the arithmetic algorithms. Thus, we implemented the parallel S-box to compare with another structures. The results of our study will inform us the good energy efficient architecture. By using the parallel S-box, the operation is completed within one clock cycle. In this case, the clock frequency is not necessarily set to be fast in order to achieve a high throughput. Therefore, we can reduce the power dissipation by avoiding the computing with a high clock frequency. However, we have to pay for the overhead due to the large amount of logics. A trade-off between power and performance can be achieved by adjusting the numbers of parallel logics.

IV. Measurement results

After S-box has been implemented using many different types, the AES system has been designed and built. Not only the AES encryption/decryption block, the key schedule, but also the RAM (stored the key) and the clock divider have been implemented. RAM is created to keep the keys that generated in every round of AES operation. Thus, the RAM size should be 128x11 bits.

Each round key is generated by the function of expanded key. The number of key can be added up to 4x (10+1) = 44. In words (state: $w_0, w_1, \ldots, w_{33}$) as well as W_i, i is in the range 0 <= $i < N_b$*(Nr+1), $N_b = 4$ and $N_r = 10$.

The initial key is defined in term of state key ($K_{0,0}, K_{0,1}, K_{0,2} \ldots K_{3,2}, K_{3,3}$). Then, each column will be read out to form in the word format (W_0, W_1, W_2, W_3). After that, the rotate word function or cyclic permutation is invoked. The next step is sub-word function whereas S-box is used here. The output of sub-word will be XORed with the constants of round constant function. Both S-box and AES algorithm have been implemented using VHDL Language. The synthesized HDLs based on Virtex-5 device XC5VLX50 are simulated on Xilinx ISE 10.1. The power consumption of the circuits has been analysed by XPower tool. The random 128 bits plaintexts and keys are used to evaluate the system.

Table I Power and resources comparison between original $GF(2^4)^2$. S-box and gating $GF(2^4)^2$ S-box.

Parameter	original $GF(2^4)^2$ S-box	Gating $GF(2^4)^2$ S-box
Number of slice logic	976	992
Latency (ns)	10.849	10.936
Power consumption (mW)	814	661
- Dynamic power (mW)	432	278
- Quiescent power (mW)	382	383

The original S-box based on $GF(2^4)^2$ in [5] implemented to compare the power and resource usages with the gating $GF(2_4)_2$ S-box. The same random plaintexts in the previous scenario have been taken to test these two types of S-box. The results in Table IV show that we can reduce the dynamic power according to the logic switching in the unnecessary block by about 19% when the logic gating technique has been applied. Thus the gating S-box is selected to apply in our AES.

V. Conclusion

The evaluations of resource and power efficiency indicate that AES using S-box based on Galois Field gives the best power efficiency even LUT is a basic architecture in FPGA. Therefore, AES based on $GF(2^4)^2$ S-box is well fit to a wireless FPGA node.

Acknowledgment

The author would like to thank to BTFP, Office of The National Broadcasting and Telecommunications Commission (NBTC) for research grant supportive.

References

[1] Hu and X. Cao, Wireless Sensor Networks: Principles and Practice, CRC Press: Taylor & Francis Group, 2010.

[2] G.G-Mplemenos, K.Papadopoulos and I. Papaefstathiou, Using Reconfigurable Hardware Devices in WSNs for Reducing the Energy Consumption of Routing and Security Tasks," in Proc. IEEE GLOBECOM, 2010, paper 10.1109, pp. 1-5.

[3] T. Kavitha, and D. Sridharan, Security Vulnerabilities In Wireless Sensor Networks: A Survey," in Journal of Information Assurance and Security, vol.5, pp.31-44, 2010.

[4] IEEE Standard for Information technology- Telecommunications and information exchange between systems- Local and metropolitan area networks- Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), ser. IEEE Standard 802.15.4-2006, September 2006.

[5] J. Wolkerstorfer et al., "An ASIC implementation of the AES SBoxes," in Proc. Cryptography Track at RSA Conf.,2002, pp.67-78.