Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 3, Number 1, pages 63–66, January 2014

High-Speed Implementation of Encryption Circuit using a High-Level Synthesis Tool

Masashi Watanabe, Keisuke Iwai, Hidema Tanaka, and Takakazu Kurokawa
National Defense Academy of Japan
Kanagawa, Japan
Email:{em52037,iwai,hidema,kuro}@nda.ac.jp

***Abstract*—It is mainstream to describe the design of a circuit in hardware description languages such as VHDL or Verilog HDL. In contrast, many high-level synthesis tools which provides the way to implement a circuit using a software programming language, such as C language, are released in late years. Therefore, in this paper, we discussed the implementation methods of high-speed encryption circuit using a high-level synthesis tool. Then we implemented two kinds of encryption circuit (MISTY1 and AES) by 4 different types of substitution table on an FPGA using Vivado HLS which is a high-level synthesis tool provided by Xilinx. As a result, the fastest methods of implementation are 1.9 and 37.6 times faster than the slowest methods, in cases of MISTY1 and AES respectively. In addition, the smallest methods of implementation require 10% and 38% smaller circuit's area than the biggest methods, in cases of MISTY1 and AES respectively.**

Keywords*—*High-Level Synthesis, FPGA, SoC, AES, MISTY1

I. Introduction

In late years, the need of the high-speed encryption processing for the memorized data and data on the channel keeps increasing because of the growth of the cloud computing, and so on. Simultaneously, the power consumption becomes the big problem because the power supply such as the smartphone is limited. Therefore, the study on high speed calculation having good energy efficiency is performed by various approaches. A many-core processor represented by GPGPU which used GPU for general-purpose processing [1][2][3], the distributed processing using general-purpose processors and reconfigurable computing using FPGA are paid attention. Although reconfigurable computing achieves high power efficiency, the design of a circuit on FPGA takes time and effort than software. As a mean to solve these problems, high-level synthesis is paid attention. Although it is mainstream to describe the design of a circuit to implement on FPGA using hardware description languages (HDL) such as VHDL or Verilog HDL in register-transfer-level (RTL), high-level synthesis tool provides easier design environment than HDLs and various automatic optimization. In this paper, we implemented two kinds of encryption circuits (MISTY1 and AES) by 4 different types of substitution table on an FPGA accelerator using Vivado HLS which is a high-level synthesis tool provided by Xilinx, and their implementation results are compared on the standpoint of speed, area and speed per area.

II. Algorithms of encryption circuits

In this paper, two symmetric block ciphers MISTY1 and AES are implemented. MISTY1 is a 64-bit symmetric block cipher and is used in a car, mobile phone and so on. It is designed for high security and high speed, small size purposes, and has been adopted in E-Government recommended ciphers list of CRYPTREC[4]. AES is a 128-bit symmetric block cipher, and is used by the U.S.Government and worldwide.

A. MISTY1

MISTY1 is a 64-bit symmetric block cipher which was introduced in 1996 by Matsui et.al. with 128-bit key[5]. Its algorithm is shown in Fig.1. It has a Feistel network with n rounds which should be multiple of 4. MISTY1 consisted of eight rounds is discussed in this paper.

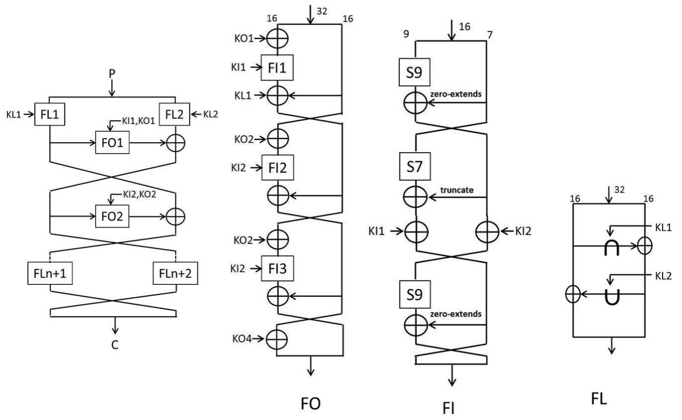

Fig. 1. Algorithm of MISTY1.

B. AES

AES is a 128-bit symmetric block cipher which was introduced in 2001 by NIST[6]. 128-bit, 192-bit and 256-bit key size can be available. We discuss only 128-bit key size in this paper. Its algorithm defines 10-round processes. Each round includes four transformations : SubBytes, ShiftRows, MixColumns and AddRoundKey. The final round slightly differs from the other rounds ; it does not include MixColumns. SubBytes, ShiftRows, MixColumns and AddRoundKey are shown in Fig. 2, 3, 4 and 5 respectively.

III. Types of implementation

The loop architecture was adopted for each circuit. In MISTY1, we implement two rounds of FL functions and FO functions in a loop-body. The loop-body is repeated four times

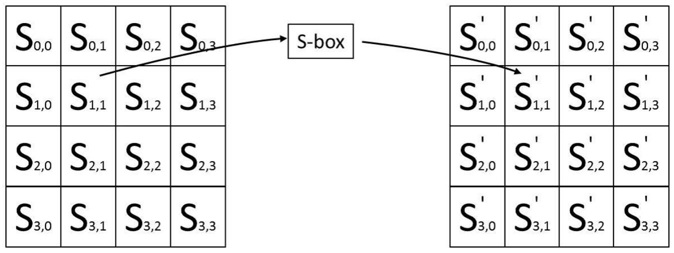

Fig. 2. AES SubByte.

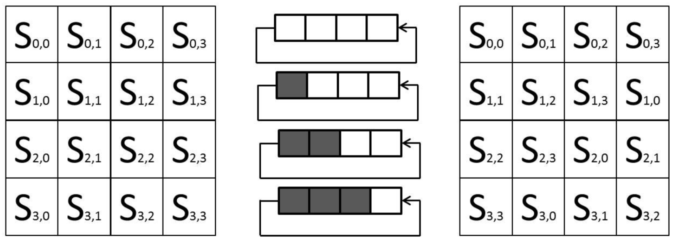

Fig. 3. AES ShiftRows.

and a ciphertext block is generated. In AES, we implement two rounds of SubByte, ShiftRows, MixColumns and AddRound-Key in a loop-body. A cipher text block is generated through four times loop-body, round 9 and round 10.

In this paper, four different types of substitution table are implemented in each encryption circuit. Details of each type are shown below.

A. *Type 1*

Each substitution table is implemented as a look-up table one by one, and optimized by the high-level synthesis tool.

B. *Type 2*

Each substitution table is implemented as a look-up table. Each table is duplicated for multiple accesses. Optimizations are applied by the high-level synthesis tool.

C. *Type 3*

Substitution tables are implemented by the calculation. Table I and II show the calculation of S7 and S9 of MISTY1 respectively[7]. In AES, we calculate its substitution table using inverse element in a Galois field ($GF(2^{2^2})$) and an affine transformation.

D. *Type 4*

The expanded substitution table which combines a substitution table and a part of other calculations are implemented in look-up tables and optimized by the high-level synthesis tool.

IV. Implementation results

A. *MISTY1*

Implementation details of each type are shown as follows:

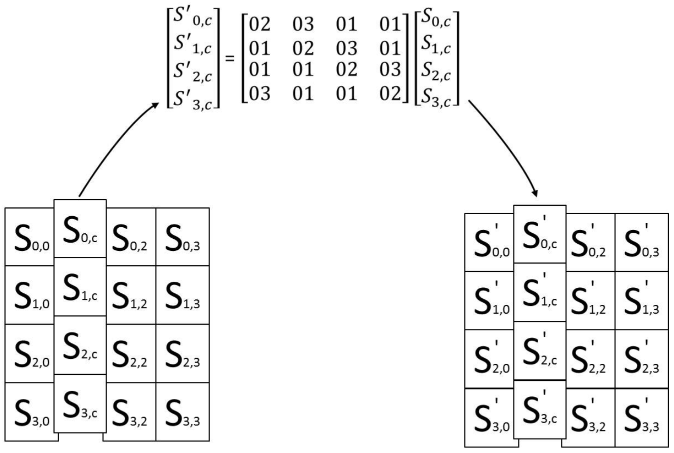
$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$

Fig. 4. AES MixColumns.

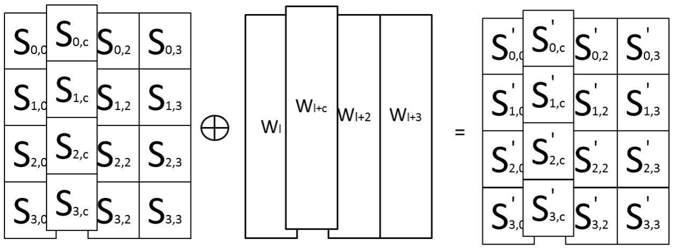

Fig. 5. AES AddRoundKey.

1) *Type 1:* Each substitution table S7 and S9 are implemented one by one for FI function.

2) *Type 2:* Six S7 and twelve S9 substitution tables for multiple access are implemented.

3) *Type 3:* We calculate substitution tables S7 and S9 in expressions as shown in Table I and II [7].

4) *Type 4:* Fig.6 shows an example of the speedup. We can transform the FI function like the same speed up method presented in [5]. S7A (the 7-bit input 16-bit output) and S9A (the 9-bit input 16-bit output) in Fig.6 are calculated as $S7A(x) = S7(x) \oplus (S7(x) << 9)$ and $S9A(x) = S9(x) \oplus (x\&0x7f) \oplus ((x\&0x7f) << 9)$ respectively.

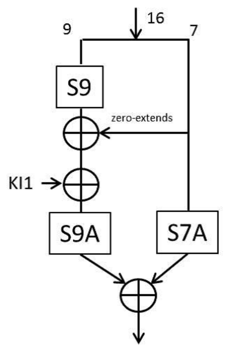

Fig. 6. MISTY1's FI'.

TABLE I. THE CALCULATION OF S7 OF MISTY1

Bit of input	Bit of output
y_0	$x_0 \oplus x_1x_3 \oplus x_0x_3x_4 \oplus x_1x_5 \oplus x_0x_2x_5 \oplus x_4x_5 \oplus x_0x_1x_6$ $\oplus x_2x_6 \oplus x_0x_5x_6 \oplus x_3x_5x_6 \oplus 1$
y_1	$x_0x_2 \oplus x_0x_4 \oplus x_3x_4 \oplus x_1x_5 \oplus x_2x_4x_5 \oplus x_6 \oplus x_0x_6$ $\oplus x_3x_6 \oplus x_2x_3x_6 \oplus x_1x_4x_6 \oplus x_0x_5x_6 \oplus 1$
y_2	$x_1x_2 \oplus x_0x_2x_3 \oplus x_4 \oplus x_1x_4 \oplus x_0x_1x_4 \oplus x_0x_5 \oplus x_0x_4x_5$ $\oplus x_3x_4x_5 \oplus x_1x_6 \oplus x_3x_6 \oplus x_0x_3x_{6+}x_4x_6 \oplus x_2x_4x_6$
y_3	$x_0 \oplus x_1 \oplus x_0x_1x_2 \oplus x_0x_3 \oplus x_2x_4 \oplus x_1x_4x_5 \oplus x_2x_6$ $\oplus x_1x_3x_6 \oplus x_0x_4x_6 \oplus x_5x_6 + 1$
y_4	$x_2x_3 \oplus x_0x_4 \oplus x_1x_3x_4 \oplus x_5 \oplus x_2x_5 \oplus x_1x_2x_5$ $\oplus x_0x_3x_5 \oplus x_1x_6 \oplus x_1x_5x_6 \oplus x_4x_5x_6 + 1$
y_5	$x_0 \oplus x_1 \oplus x_2 \oplus x_0x_1x_2 \oplus x_0x_3 \oplus x_1x_2x_3 \oplus x_1x_4$ $\oplus x_0x_2x_4 \oplus x_0x_5 \oplus x_0x_1x_5 \oplus x_3x_5 \oplus x_0x_6 \oplus x_2x_5x_6$
y_6	$x_0x_1 \oplus x_3 \oplus x_0x_3 \oplus x_2x_3x_4 \oplus x_0x_5 \oplus x_2x_5 \oplus x_3x_5$ $\oplus x_1x_3x_5 \oplus x_1x_6 \oplus x_1x_2x_6 \oplus x_0x_3x_6 \oplus x_4x_6 \oplus x_2x_5x_6$

TABLE II. THE CALCULATION OF S9 OF MISTY1

Bit of input	Bit of output
y_0	$x_0x_4 \oplus x_0x_5 \oplus x_1x_5 \oplus x_1x_6 \oplus x_2x_6 \oplus x_2x_7 \oplus x_3x_7$ $\oplus x_3x_8 \oplus x_4x_8 \oplus 1$
y_1	$x_0x_2 \oplus x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_4x_5 \oplus x_0x_6$ $\oplus x_2x_6 \oplus x_7 \oplus x_0x_8 \oplus x_3x_8 \oplus x_5x_8 \oplus 1$
y_2	$x_0x_1 \oplus x_1x_3 \oplus x_4 \oplus x_0x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_4x_5$ $\oplus x_0x_6 \oplus x_5x_6 \oplus x_1x_7 \oplus x_3x_7 \oplus x_8$
y_3	$x_0 \oplus x_1x_2 \oplus x_2x_4 \oplus x_5 \oplus x_1x_5 \oplus x_3x_5 \oplus x_4x_5 \oplus x_5x_6$ $\oplus x_1x_7 \oplus x_6x_7 \oplus x_2x_8 \oplus x_4x_8$
y_4	$x_1 \oplus x_0x_3 \oplus x_2x_3 \oplus x_0x_5 \oplus x_3x_5 \oplus x_6 \oplus x_2x_6 \oplus x_4x_6$ $\oplus x_5x_6 \oplus x_6x_7 \oplus x_2x_8 \oplus x_7x_8$
y_5	$x_2 \oplus x_0x_3 \oplus x_1x_4 \oplus x_3x_4 \oplus x_1x_6 \oplus x_4x_6 \oplus x_7 \oplus x_3x_7$ $\oplus x_5x_7 \oplus x_6x_7 \oplus x_0x_8 \oplus x_7x_8$
y_6	$x_0x_1 \oplus x_3 \oplus x_1x_4 \oplus x_2x_5 \oplus x_4x_5 \oplus x_2x_7 \oplus x_5x_7 \oplus x_8$ $\oplus x_0x_8 \oplus x_4x_8 \oplus x_6x_8 \oplus x_7x_8 \oplus 1$
y_6	$x_1 \oplus x_0x_1 \oplus x_1x_2 \oplus x_2x_3 \oplus x_0x_4 \oplus x_5 \oplus x_1x_6 \oplus x_3x_6$ $\oplus x_0x_7 \oplus x_4x_7 \oplus x_6x_7 \oplus x_1x_8 \oplus 1$
y_6	$x_0 \oplus x_0x_1 \oplus x_1x_2 \oplus x_4 \oplus x_0x_5 \oplus x_2x_5 \oplus x_3x_6 \oplus x_5x_6$ $\oplus x_0x_7 \oplus x_0x_8 \oplus x_3x_8 \oplus x_6x_8 \oplus 1$

Table III summarizes our implementation results. Types 2 and 4 became the fastest implementations, and are 1.9 times faster than the slowest implementation($Type1$). In addition, the smallest implementation ($Type4$) became 10% smaller than the biggest implementation ($Type1$).

TABLE III. MISTY1.

Type	Throughput [Mbps]	Area [slices]	Throughput/Area [Kbps/ slices]
Type1	108.4	518	214.3
Type2	202.3	513	403.8
Type3	168.1	468	367.8
Type4	202.3	462	448.4

B. AES

Each type of implementation is shown as follows.

1) Type 1: A substitution table (S-box) is implemented as a look-up table.

2) Type 2: Sixteen S-boxes are implemented as a look-up table.

3) Type 3: S-box is implemented by calculation using inverse element in a Galois field and an affine transformation.

4) Type 4: A substitution table (T-box) which combines a calculation result of MixColumns and S-box is implemented.

Table IV summarizes our implementation results. The fastest implementation ($Type4$) became 37.6 times faster than the slowest implementation ($Type1$). In addition, the smallest implementation ($Type4$) became 38% smaller than the biggest implementation ($Type1$).

TABLE IV. AES.

Type	Throughput [Mbps]	Area [slices]	Throughput/Area [Kbps/ slices]
Type1	96.7	455	217.6
Type2	448.1	403	1138.6
Type3	37.1	563	66.3
Type4	1393.5	646	2208.9

As a result, the fastest types of implementation became 1.9 and 37.6 times faster than the slowest types, in cases of MISTY1 and AES respectively. Moreover, the smallest types of implementations required 10% and 8% smaller area than the biggest types, in cases of MISTY1 and AES respectively. These implementation results show that the transaction speed per area became $Type4 > Type2 > Type3 > Type1$ in case of MISTY1, and $Type4 > Type2 > Type1 > Type3$ in case of AES. However, $Type3$ implementation of AES has a different structure from other types. Its loop-body contains only one round. Because $GF(2^{2^2})$ requires too much area size, so that its speed became slower.

Finally, the encryption speed of the implementation using the look-up tables (Type2 and Type4) could achieve high-speed in both MISTY1 and AES. The implementation by the calculation ($Type3$) became slower than the implementation by look-up tables ($Type2$ and $Type4$), and the area did not become so small. If the circuit designed by HDL, the area of the implementation by the calculation ($Type3$) would become smaller than the implementations by look-up table ($Type1$, $Type2$ and $Type4$). There may be a limit by a high-level synthesis tool.

V. CONCLUSION

In this paper, we discussed the implementation methods of the high-speed encryption circuit using a high-level synthesis tool. Then we implemented two kinds of encryption circuit (MISTY1 and AES) by four different implementation types of substitution table on an FPGA using a high-level synthesis tool. As a result, the fastest types of implementation became 1.9 and 37.6 times faster than the slowest types, in cases of MISTY1 and AES respectively. Moreover, the smallest types of implementations required 10% and 8% smaller area than the biggest types, in cases of MISTY1 and AES respectively. The transaction speed per area became $Type4 > Type2 > Type3 > Type1$ in case of MISTY1, and $Type4 > Type2 > Type1 > Type3$ in case of AES. These results show that the implementation using the expanded substitution table(S7A, S9A and T-box) ($Type4$) should be the fastest method of implementation in both MISTY1 and AES.

ACKNOWLEDGMENT

This work was supported by Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research (KAKENHI) Grant Number 25871223. (Grant-in-Aid for Young Scientists B)

REFERENCES

[1] N. Nishikawa, K. Iwai, and T. Kurokawa, "High-performance symmetric block ciphers on cuda," in *Proceedings of the 2011 Second International Conference on Networking and Computing*, ser. ICNC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 221–227. [Online]. Available: http://dx.doi.org/10.1109/ICNC.2011.40

[2] D. A. Osvik, J. W. Bos, D. Stefan, and D. Canright, "Fast software aes encryption," in *Proceedings of the 17th international conference on Fast software encryption*, ser. FSE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 75–93. [Online]. Available: http://dl.acm.org/citation.cfm?id=1876089.1876096

[3] K. Iwai, T. Kurokawa, and N. Nishikawa, "Aes encryption implementation on cuda gpu and its analysis." in *ICNC*. IEEE Computer Society, 2010, pp. 209–214. [Online]. Available: http://dblp.uni-trier.de/db/conf/ic-nc/ic-nc2010.html#IwaiKN10

[4] T. National Institute of Standerd adn Technology(MIC), Ministry of Economy and Industry(METI), "E-government recommended ciphers list," Tech. Rep., 2003. [Online]. Available: http://www.cryptrec.go.jp/images/cryptrec_ciphers_list_fy2005.pdf

[5] M. E. Croporation, "Specifications in 2001 misty1," Tech. Rep., 2001.

[6] N. I. of Standerd adn Technology(NIST), "Fips-197 advanced encryption standard(aes)," Tech. Rep., 2001.

[7] M. Matsui, "New block encryption algorithm misty," in *FSE*, ser. Lecture Notes in Computer Science, E. Biham, Ed., vol. 1267. Springer, 1997, pp. 54–68. [Online]. Available: http://dblp.uni-trier.de/db/conf/fse/fse97.html#Matsui97