A High-Performance ASIC Implementation of the 64-bit Block Cipher CAST-128

Takeshi Sugawara, Naofumi Homma, Takafumi Aoki
Graduate School of Information Sciences, Tohoku University
Sendai, Miyagi, Japan
{sugawara, homma}@aoki.ecei.tohoku.ac.jp
aoki@ecei.tohoku.ac.jp

Akashi Satoh
IBM Research, Tokyo Research Laboratory
Yamato, Kanagawa, Japan
akashi@jp.ibm.com

***Abstract*—We propose a compact hardware architecture for the 64-bit block cipher CAST-128, which is one of the ISO/IEC 18033-3 standard algorithms. Part of the complexity of CAST-128 is its use of various S-boxes in various sequences, and three types of f-function are switched depending on the round numbers. Therefore a large amount of hardware resources are required for a straight-forward implementation. In order to create compact CAST-128 hardware, we minimized the number of S-box components, and merged the three f-functions into one arithmetic component. The CAST-128 hardware based on the proposed architecture was synthesized using 0.13-μm and 0.18-μm CMOS standard cell libraries and small, practical circuits of 26.4~39.5 Kgates and 189.9~614.7 Mbps were obtained.**

I. INTRODUCTION

CAST-128 [1] is a 64-bit block cipher developed by Carlisle Adams, and its specification was published as RFC 2144 [2]. CAST-128 was approved by the CSE (Communications Security Establishment) for use by the Government of Canada [3], and was also adopted as one of the ISO/IEC standard block ciphers [4]. The popular e-mail ciphering tool PGP (Pretty Good Privacy) [5][6] uses CAST-128 as the default algorithm.

CAST-128 has eight different types of 8-bit input and 32-bit output S-boxes defined as lookup tables, and they are used a number of times in the key scheduling and data randomization processes. It also has three different types of 32-bit f-functions. The algorithm can be efficiently implemented on 32-bit processors, but the large S-boxes and the three f-functions are problematic in the development of compact hardware. Therefore, only one hardware implementation on a FPGA platform with a sufficiently large RAM exists [7], and no evaluation on an ASIC platform was done, as far as the authors know.

In this paper, we propose a small CAST-128 hardware architecture, where a minimum set of S-boxes is used and the three f-functions are merged by using a unified arithmetic unit. The ASIC performances of our CAST-128 circuit are evaluated in comparison with the standard block ciphers AES and DES in 0.13-μm and 0.18-μm CMOS standard cell libraries.

II. CAST-128 ALGORITHM

The 64-bit block cipher CAST-128 has a Feistel-type data randomization block as shown in Fig. 1, and three f-functions of Types 1~3 are switched in accordance with Table I. The 32-bit additions and subtractions are performed on modulus 2^{32}, and the four S-boxes S1~S4 are 8-bit input and 32-bit output lookup tables defined by using a bent function.

The key length is variable in the range of 40~128 bits but divisible by 8. The number of iteration rounds is 12 or 16 for 40~80-bit and 88~128-bit keys, respectively. When an input key is shorter than 128 bits, zero bytes are padded on the right end. Then 32-bit round keys K_1~K_4 are generated according to Equations (1)~(8), where $x_0x_1x_2x_3x_4x_5x_6x_7x_8x_9x_Ax_Bx_Cx_Dx_Ex_F$ and $z_0z_1z_2z_3z_4z_5z_6z_7z_8z_9z_Az_Bz_Cz_Dz_Ez_F$ represent the input key and the intermediate key with 8-bit granularity, x_0 is the most significant byte, and x_F is the least significant byte.

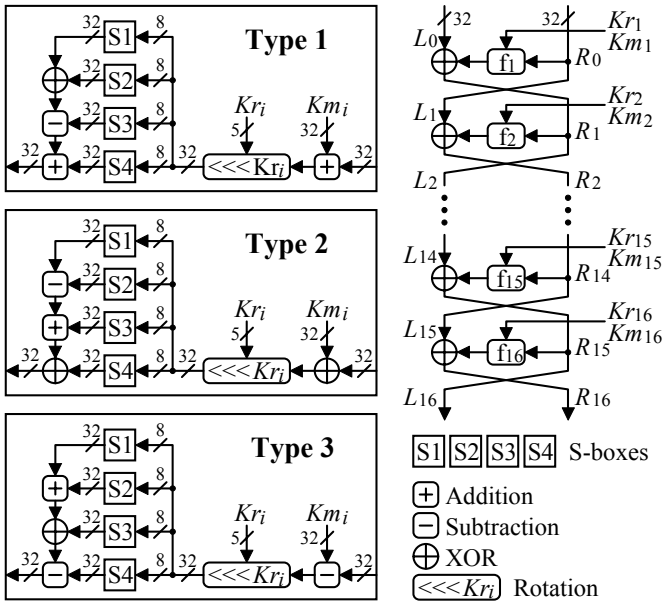

Fig. 1. Data randomization block of CAST-128

TABLE I. Type of f-function used in each round

Type	Rounds
1	1,4,7,10,13,16
2	2,5,8,11,14
3	3,6,9,12,15

$$z_0 z_1 z_2 z_3 = x_0 x_1 x_2 x_3 \oplus S5[x_D] \oplus S6[x_F] \oplus S7[x_C] \oplus S8[x_D] \oplus S7[x_8] \quad (1)$$

$$z_4 z_5 z_6 z_7 = x_8 x_9 x_A x_B \oplus S5[z_0] \oplus S6[z_2] \oplus S7[z_1] \oplus S8[z_3] \oplus S8[z_A] \quad (2)$$

$$z_8 z_9 z_A z_B = x_C x_D x_E x_F \oplus S5[z_7] \oplus S6[z_6] \oplus S7[z_5] \oplus S8[z_4] \oplus S5[z_9] \quad (3)$$

$$z_C z_D z_E z_F = x_4 x_5 x_6 x_7 \oplus S5[z_A] \oplus S6[z_9] \oplus S7[z_B] \oplus S8[z_8] \oplus S6[z_B] \quad (4)$$

$$K_1 = S5[z_8] \oplus S6[z_9] \oplus S7[z_7] \oplus S8[z_6] \oplus S5[z_2] \quad (5)$$

$$K_2 = S5[z_A] \oplus S6[z_B] \oplus S7[z_5] \oplus S8[z_4] \oplus S6[z_6] \quad (6)$$

$$K_3 = S5[z_C] \oplus S6[z_D] \oplus S7[z_3] \oplus S8[z_2] \oplus S7[z_9] \quad (7)$$

$$K_4 = S5[z_E] \oplus S6[z_F] \oplus S7[z_1] \oplus S8[z_0] \oplus S8[z_C] \quad (8)$$

The key scheduling procedure uses four 8-bit input and 32-bit output S-boxes S5~S8 that are different from the S-boxes used in the data randomization. Similar procedures are repeated seven times (as the order of the input bytes and S-boxes changes) to generate the rest of the round keys K_5~ K_{32}. The round keys K_1~K_{16} are used as the 32-bit mask keys Km_1~Km_{16} in Fig. 1 without any modification, and the rotate keys Kr_1~Kr_{16} are the lower 5 bits of each of the sixteen 32-bit round keys K_{17}~K_{32}. When the number of iteration rounds is 12 for a short key, Km_1~Km_{12} and Kr_1~ Kr_{12} are used.

III. Proposed Hardware Architecture

A. Data Randomization Block

Fig. 2 shows the data randomization block of our compact CAST-128 hardware. This datapath can be used for both encryption and decryption because of the Feistel-network feature of CAST-128. Four S-boxes S1~S4 and the barrel shifter in Fig. 1 can be shared between the three f-functions of Types 1~3. A 32-bit adder can easily support subtraction with minor additional circuitry, and XOR gates are already included in the adder. Therefore, we designed a unified arithmetic unit ASX (Add-Sub-OR) shown in Fig. 3, which switches three arithmetic operations, and merged the three f-functions into one functional block.

A carry look-ahead scheme is used for addition and subtraction, considering the balance between speed and gate count. The signals *Sel*0 and *Sel*1 are used to control the operations in ASX. When *Sel*0=*Sel*1=0 in Fig. 3, all of the carry signals C and $C_{1\sim31}$ fed to the ASX units are disabled, and then the 32-bit XOR result between $A_{0\sim31}$ and $B_{0\sim31}$ is output to $S_{0\sim31}$. To perform subtraction, the signal *Sel*0 is set to 1, and then the carry signals are enabled. *Sel*1 is also set to 1 so that all $A_{0\sim31}$ bits are inverted and 1 is added at the LSB ASX cell through the carry signal C, and then the two's complement form of the operand $A_{0\sim31}$ is added to $B_{0\sim31}$. The carry generation unit CG0 that does not generate the MSB carry is used for mod 2^{32} operation. Addition between $A_{0\sim31}$ and $B_{0\sim31}$ is performed by setting *Sel*0=1 and *Sel*1=0. When *Sel*0=0 and *Sel*1=1, the XNOR result is output to $S_{0\sim31}$, though this operation is not used in the CAST-128 hardware.

B. Key Scheduler

The same S-box is used twice in each of the Equations (1)~(8) for the key scheduling. Therefore, two sets of S-boxes S5~S8 are required if each equation is executed in one clock. To minimize the number of S-boxes, we transformed Equations (1)~(4) and (5)~(6) into Equations (9)~(12) and (13)~(16), respectively, by executing each equation in two clocks.

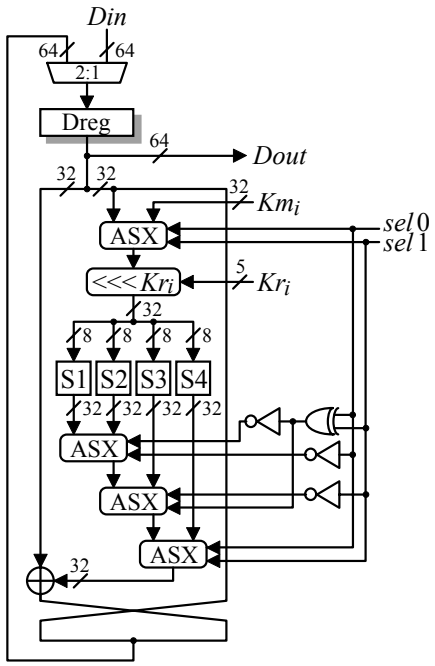

Fig. 2. Datapath of data randomization block

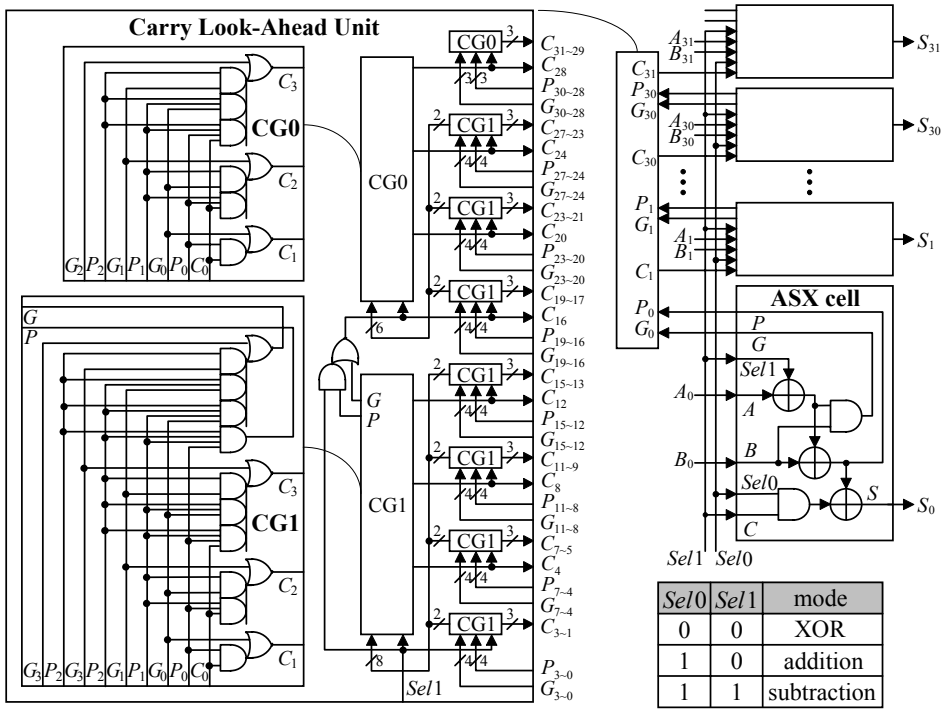

Fig. 3. Unified arithmetic unit ASX

$$\begin{cases} z_0z_1z_2z_3 = x_0x_1x_2x_3 \oplus S7[x_8] \\ z_0z_1z_2z_3 = z_0z_1z_2z_3 \oplus S5[x_D] \oplus S6[x_F] \oplus S7[x_C] \oplus S8[x_D] \end{cases} \quad (9)$$

$$\begin{cases} z_4z_5z_6z_7 = x_8x_9x_Ax_B \oplus S8[x_A] \\ z_4z_5z_6z_7 = z_4z_5z_6z_7 \oplus S5[z_0] \oplus S6[z_2] \oplus S7[z_1] \oplus S8[z_3] \end{cases} \quad (10)$$

$$\begin{cases} z_8z_9z_Az_B = x_8x_9x_Ax_B \oplus S5[x_9] \\ z_8z_9z_Az_B = z_8z_9z_Az_B \oplus S5[z_7] \oplus S6[z_6] \oplus S7[z_5] \oplus S8[z_4] \end{cases} \quad (11)$$

$$\begin{cases} z_Cz_Dz_Ez_F = x_4x_5x_6x_7 \oplus S6[x_B] \\ z_Cz_Dz_Ez_F = z_Cz_Dz_Ez_F \oplus S5[z_A] \oplus S6[z_9] \oplus S7[z_B] \oplus S8[z_8] \end{cases} \quad (12)$$

$$\begin{cases} K_1 = S5[z_8] \oplus S6[z_9] \oplus S7[z_7] \oplus S8[z_6] \\ K_1 = K_1 \oplus S5[z_2] \end{cases} \quad (13)$$

$$\begin{cases} K_2 = S5[z_A] \oplus S6[z_B] \oplus S7[z_5] \oplus S8[z_4] \\ K_2 = K_2 \oplus S6[z_6] \end{cases} \quad (14)$$

$$\begin{cases} K_3 = S5[z_C] \oplus S6[z_D] \oplus S7[z_3] \oplus S8[z_2] \\ K_3 = K_3 \oplus S7[z_9] \end{cases} \quad (15)$$

$$\begin{cases} K_4 = S5[z_E] \oplus S6[z_F] \oplus S7[z_1] \oplus S8[z_0] \\ K_4 = K_4 \oplus S8[z_C] \end{cases} \quad (16)$$

The round keys of CAST-128 cannot be executed in reverse order for decryption, and thus all the keys are generated and stored in key registers in advance. Therefore, even though the number of clocks for key generation is doubled, it has no affect on a number of clocks required for data randomization.

Fig. 4 shows the datapath architecture of our key scheduler. There are two paths for output from the two 128-bit registers $x_0x_1x_2x_3x_4x_5x_6x_7x_8x_9x_Ax_Bx_Cx_Dx_Ex_F$ and $z_0z_1z_2z_3z_4z_5z_6z_7z_8z_9z_Az_Bz_Cz_Dz_Ez_F$. One path goes to four S-boxes after four bytes are selected by the "Switching Box," and the other path is XORed with the S-box output after one 32-bit data block is selected by a 10:1 multiplexer. Then the results are fed back to the registers as shown in Equations (9)~(12). In order to generate four 32-bit round keys K_1~K_4 (identical to the four mask keys Km_1~Km_4) by XORing the five S-box outputs according to Equations (13)~(16), the four 32-bit registers Km_1~Km_4 in Fig. 4, and the feedback path of the 10:1 multiplexer is used. The round keys K_{17}~K_{32} are generated similarly, but only the lower 5 bits of each key are used as the rotate key. Therefore, the registers Kr_1~Kr_{16} are 5 bits each. During the rotate key generation processes the outputs from the 5-bit registers Kr_1~Kr_{16} are fed back to the XOR-tree through the 10:1 multiplexer. At that time, the upper 27 bits of the multiplexer output are not used.

IV. Performance Evaluation in ASIC

The proposed compact CAST-128 hardware architecture described above was designed and synthesized with two optimizations, size and speed, by using 0.13-μm and 0.18-μm CMOS standard cell libraries under the worst case conditions. Tables II and III show the synthesis results. AES [8] and DES [9] circuits were also synthesized under the same conditions for performance comparisons.

Our CAST-128 hardware achieves 26.9 Kgates for size optimization and 39.5 Kgate for speed optimization using the 0.13-μm library. The gate counts are 26.4K and 32.8K with the 0.18-μm library. These numbers are comparable with 26.7 Kgates and 43.6 Kgates of AES hardware with lookup table S-boxes. When we implement CAST-128 S-boxes using memory, the eight different 8-bit input (2^8=256 addresses) and 32-bit output (4 Bytes) lookup tables requires 256 % 4 Bytes % 8 = 8 Kbytes. In contrast, AES uses sixteen 256-Byte S-boxes for data randomization, and four of them for key scheduling, and thus the total capacity is 256 Bytes % 20 = 5.1 Kbytes. AES can generate round keys on-the-fly, but CAST-128 needs many clocks to generate round keys for encryption, and cannot generate the keys on-the-fly for decryption. Therefore, all the CAST-128 round keys should be pre-computed and stored into large registers (registers $Kr1$~ $Kr16$ and $Km1$~ $Km16$ in Fig. 4). Considering these facts, it was unexpected that our compact CAST-128 hardware architecture would be comparable in size to AES.

However, the highest throughput is 614.7 Mbps by using the 0.13-μm library with speed optimization, which is only 1/5 of AES. This depends largely on the structure of cipher algorithms. The Feistel-type 64-bit block cipher CAST-128 can process only 32 bits at a time, while the SPN-type 128-bit block cipher AES does 128 bits. This disadvantage in speed is the same as for the 64-bit block cipher DES with the Feistel network. The lower frequency of CAST-128, which is 2/3~1/2 of AES, is another reason for the lower throughput. This is mainly due to the four 32-bit arithmetic operations in the CAST-128 f-functions, while AES uses operations on binary fields

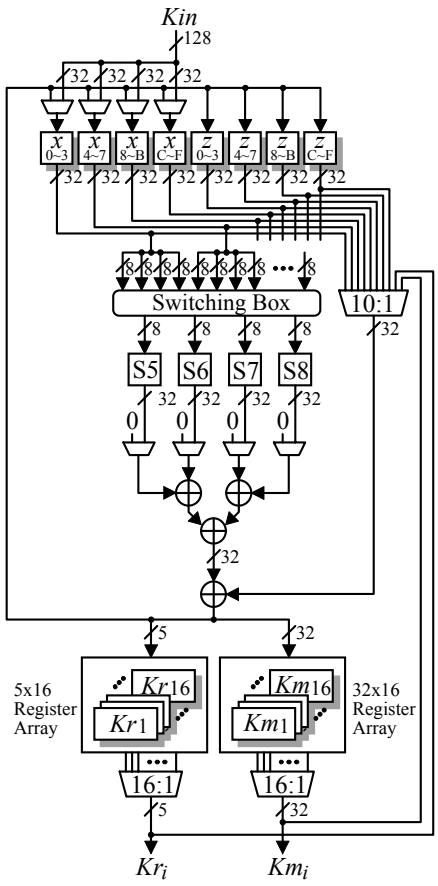

Fig. 4. Datapath of key scheduler

where no carry propagation occurs. Additional circuitry in the critical path is also required to switch the f-functions. By utilizing the advantage of binary field arithmetic, AES can also achieve a very compact S-box on the composite field $GF(((2^2)^2)^2)$ S-box [10] as shown in Tables II and III.

The gate counts of DES is much lower than CAST-128, but this is obviously because DES has only 256 Bytes for eight 6-bit input and 4-bit output S-boxes in total while CAST-128 needs 8 Kbytes. The throughputs of CAST-128 are only 1/4~1/5 of DES, but triple-DES that repeats DES three times is recommended because of security issues. In comparison with triple-DES, the throughput of CAST-128 is from the same level to half, which is good enough for practical use. As far as the authors know, only one FPGA implementation was reported for CAST-128 hardware in [7], and a throughput of 220 Mbps was obtained for a loop architecture version. Even though the platforms are different (FPGA and ASIC), our design achieved the three-times-higher throughput of 614.7 Mbps.

CAST-128 that uses large S-boxes has usually been implemented as software, but our results show that small and fast ASIC hardware implementations can be achieved by using our proposed architecture.

V. CONCLUSION

In this paper, we proposed a compact hardware architecture for the ISO/IEC 18033-3 standard 64-bit block cipher CAST-128. Its performances were evaluated using 0.13-μm and 0.18-μm CMOS standard cell libraries and gate counts of 26.4~39.5 Kgates with throughputs of 189.9~614.7 Mbps were obtained. These gate counts are almost the same as AES with lookup table S-boxes. The throughputs are rather low, but good enough for actual use.

We are developing ASIC hardware for all of the other ISO/IEC standard ciphers such as Camellia, SEED, and MISTY1, and will report performance comparisons in the near future.

REFERENCES

[1] "CAST Encryption Algorithm Related Publications" http://adonis.ee.queensu.ca/cast/

[2] C. Adams, "The CAST-128 Encryption Algorithm," RFC 2114, May 1997. http://www.ietf.org/rfc/rfc2144.txt

[3] CSE, "IT Security Program - Cryptographic Algorithms," http://www.cse-cst.gc.ca/services/crypto-services/crypto-algorithms-e.html

[4] ISO/IEC 18033-3 "Information technology -- Security techniques -- Encryption algorithms -- Part 3: Block ciphers," Jul. 2005. http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=37972

[5] "The International PGP Home Page," http://www.pgpi.org/

[6] J. Callas, et. al, "Open PGP Message Format," RFC2440, Nov. 1998. http://www.rfc.net/rfc2440.html

[7] P. Kitsos, N. Sklavos, M. D. Galanis, and O. Koufopavlou, "64-bit Block ciphers: hardware implementations and comparison analysis," *Computers and Electrical Engineering*, Vol. 30, Mar. 2005. http://www.vlsi.ee.upatras.gr/~mgalanis/pubs/caee.pdf

[8] A. Satoh, et. al, "Hardware- Focused Performance Comparison for the Standard Block Ciphers AES, Camellia, and Triple-DES," *ISC 2003,* LNCS 2851, pp.252-266, Oct. 2003.

[9] http://www.aoki.ecei.tohoku.ac.jp/crypto/index.html

[10] A. Satoh, et. al, "A Compact Rijndael Hardware Architecture with S-Box Optimization," *ASIACRYPT 2001*, LNCS 2248, pp.239-254, Dec. 2001.

TABLE II. Hardware Performance Comparison of Block Ciphers in a 0.13μm-CMOS ASIC (gate = 2-way NAND)

Algorithm	Data Size (bits)	Key Size (bits)	Cycle	S-box	Gate Counts	Critical Path (ns)	Oper. Freq. (MHz)	Throughput (Mbps)	(Kbps /gate)	Optimize
CAST-128	64	128	17	Table	26,853	8.50	117.7	442.9	16.49	Size
					39,497	6.20	161.3	614.7	15.56	Speed
AES [8]	128	128	10	Table	26,691	5.69	175.8	2,249.6	84.28	Size
					36,923	3.70	270.3	3,459.5	93.69	Speed
				$GF(((2^2)^2)^2)$	15,512	6.90	144.9	1,855.0	119.59	Size
					20,328	4.60	217.4	2,782.6	136.89	Speed
DES [9]	64	56	16	Table	2,768	2.50	400.0	1,600.0	578.03	Size
					5,534	1.80	555.6	2,222.2	401.56	Speed

TABLE III. Hardware Performance Comparison of Block Ciphers in a 0.18μm-CMOS ASIC (gate = 2-way NAND)

Algorithm	Data Size (bits)	Key Size (bits)	Cycle	S-box	Gate Counts	Critical Path (ns)	Oper. Freq. (MHz)	Throughput (Mbps)	(Kbps /gate)	Optimize
CAST-128	64	128	17	Table	26,428	19.82	50.5	189.9	7.19	Size
					32,786	12.89	77.6	292.1	8.91	Speed
AES [8]	128	128	10	Table	24,206	11.80	84.7	1,084.7	44.81	Size
					43,590	6.32	158.2	2,025.3	46.46	Speed
				$GF(((2^2)^2)^2)$	15,990	11.86	84.3	1,079.3	67.50	Size
					27,787	7.86	127.2	1,628.5	58.61	Speed
DES [9]	64	56	16	Table	3,037	3.81	262.5	1,049.9	345.61	Size
					6,526	2.81	355.9	1,423.5	218.11	Speed