An Implementation on the High Speed Blowfish

Jong Tae PARK and Kang Hyeon RHEE

School of Electronics, Information & Communications Eng., Chosun Univ.,
501-759, Seoseok-dong 375, Dong-gu, Gwangju city, Korea

{pjt, khrhee}@vlsi.chosun.ac.kr
http://multimedia.chosun.ac.kr
Tel : +82-62-230-230-7066
Fax : +82-62-233-1120

Abstract

Blowfish is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32bit to 448bit, making it ideal for both domestic and exportable use. This paper is somewhere middle-of-the-line, where this paper made significant tradeoffs between speed, size and ease of implementation. The main focus was to make an implementation that was usable, moderately compact, and would still run at an acceptable clock speed. For the real time process of blowfish, it is required that high-speed operation and small size hardware.

So, A structure of new adders constructed in this study has all advantages abstracted from other adders. As for this new adder, area cost increases by 1.06 times and operation speed increases by 1.42 times.

1. Introduction

Cryptosystems are increasingly important in electronic communication, seeking to ensure that information is neither meaningful nor useful to an unauthorized receiver.
Cryptography is widely applied to protect digital data.

Nowadays, there are many kinds of cryptography and most of them require a secret key to encode digital data. After applying a cryptography algorithm to our digital data, others can't regain the original data easily without the secret key.
Then, the private data are under protection.

Blowfish is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use. Bruce Schneier designed blowfish in 1993.

As a fast, free alternative to existing encryption algorithms. Since then it has been analyzed considerably, and it is slowly gaining acceptance as a strong encryption algorithm.

The Blowfish algorithm has many advantages. It is suitable and efficient for hardware implementation. Besides, it is unpatented and no license is required.
The elementary operators of Blowfish algorithm include table-lookup, addition and XOR. The table includes four S-boxes (256×32bits) and a P-array (18×32bits).

Blowfish is a cipher based on Feistel rounds, and the design of the f-function used amounts to a simplification of the principles used in DES to provide the same security with greater speed and efficiency in software. The block ciphers Khafre and CAST have somewhat similar rounds.[1][2]

In this paper, the scheme of Blowfish has been designed to completely and correctly implements the algorithm with a focus on ease-of-design and ease-of-use without sacrificing too much speed or size. Certainly better implementations could be used and my existing circuit could definitely be optimized.

For the real time process of blowfish, it is required that high-speed operation and small size hardware.
So, A structure of new adders constructed in this study has all advantages abstracted from other adders.

2. A Theoretical Background of Blowfish

Unlike DES, Blowfish applies the f-function to the left half of the block, obtaining a result XORed to the right half of the block. Originally, I had said that this departure from convention might cause confusion in reading the description of Blowfish.

However, upon further reflection, I think that it is really DES that is creating confusion; the time sequence of events should move from left to right (particularly in a design that is otherwise big-endian); this is generally what happens in more recent designs, such as the AES candidates, and particularly in ciphers with unbalanced Feistel rounds.

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts: a key-expansion part and a data-encryption part. Key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes.

Data encryption occurs via a 16-round Feistel network. Each

round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.[3][4]

The Data flow graph of blowfish block cipher is shown in Fig. 1.

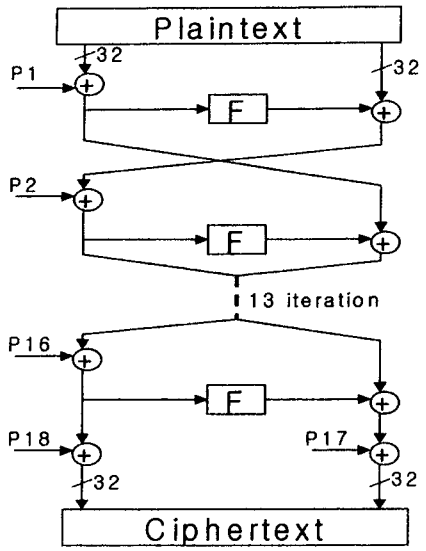

Fig. 1. Data flow graph of Blowfish block cipher

Blowfish consists of sixteen rounds. For each round, first XOR the left half of the block with the subkey for that round. Then apply the f-function to the left half of the block, and XOR the right half of the block with the result. Finally, after all but the last round, swap the halves of the block. There is only one subkey for each round; the f-function consumes no subkeys, but uses S-boxes that are key dependent.

After the last round, XOR the right half with subkey 17, and the left half with subkey 18.

First dividing XL into Four 8bit quarters calculates F(XL). And then

$F(X_L) = ((S_1[box1] + S_2[box2]) + XOR\ S_3[box3]) + S_4[box4]$

The Function F is shown in Fig. 2.

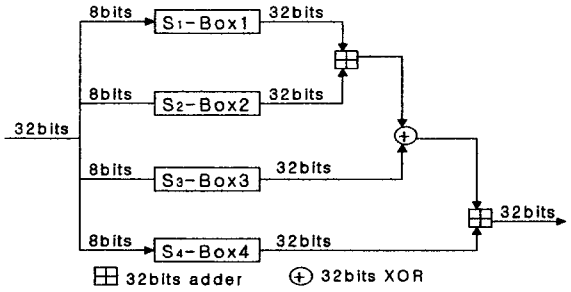

Fig. 2. Function F

3. Realization of Blowfish

In any implementation, there are several separate circuit pieces that can be identified. First is the encryption core that implements the actual Fiestel network. Second is the function F(xL) that crypt relies on for each round of the Fiestel network. Third is a generated array of sub-keys, called the p-array, which is also used by crypt each round. Fourth are the four key-dependent sboxes that are read by the F(xL) function also each round. Fifth would be and control logic necessary to initialize the p-array and sboxes.

3.1 Encryption Core

Blowfish is a Feistel network consisting of 16 rounds. The input is a 64-bit data element.

Encryption can be expressed as

Input : X(64bit data block plaintext)

Divide X into two 32bit halves X_L and X_R

For I = 1 to 16 $X_L = X_L \oplus P_I$

$X_R = F(X_L) \oplus X_R$ $X_L \Leftrightarrow X_R$

End for

$X_L \Leftrightarrow X_R$ $X_R = X_R \oplus P_{17}$

$X_L = X_L \oplus P_{18}$ Recombine X_L and X_R

Output : X(64bit data block ciphertext)

The Block diagram of encryption blowfish is shown in Fig. 3.

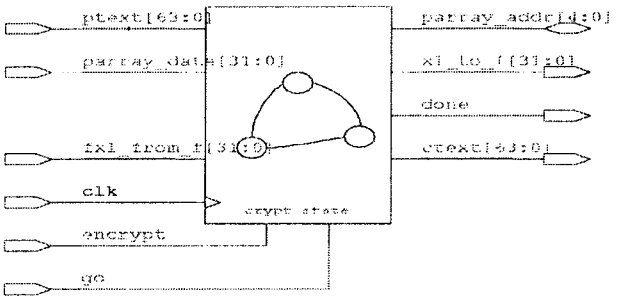

Fig. 3. Block diagram of encryption blowfish

3.2 The f-function

Blowfish uses four S-boxes. Each one has 256 entries, and each of the entries is 32 bits long.

To calculate the f-function: use the first byte of the 32 bits of input to find an entry in the first S-box, the second byte to find an entry in the second S-box, and so on. The Block diagram of f-function is shown in Fig. 4.

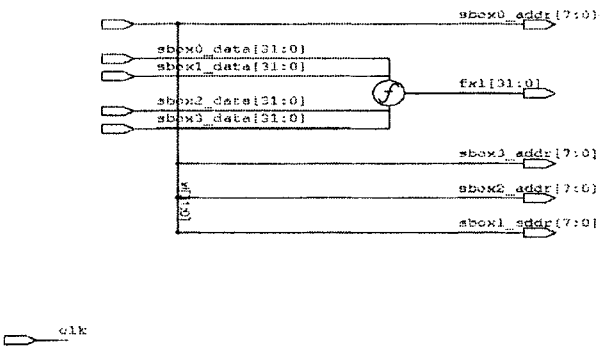

Fig. 4. Block diagram of f-function

3.3 Subkey generation

Begin by initializing subkeys 1 through 18, followed by elements zero through 255 of the first S box, then elements zero through 255 of the second S box, all the way to element 255 of the fourth S box, with the fractional part of pi. The most significant bit of the fractional part of pi becomes the most significant bit of the first subkey.

Then, take the key, which may be of any length up to 72 bytes, and, repeating it as often as necessary to span the entire array of 18 subkeys, XOR it with the subkey array contents.

Then execute the Blowfish algorithm repeatedly, with an initial input of a 64-byte block of all zeroes as plaintext input. After each execution, replace part of the subkeys or S boxes with the successive outputs of Blowfish, in the same order as the digits of pi in binary (or hexadecimal) form were placed in them; after the first iteration, replace subkeys 1 and 2; after the tenth iteration, replace the first two entries (0 and 1) in S-box 1; and so on.

The Block diagram of P-array is shown in Fig. 5.

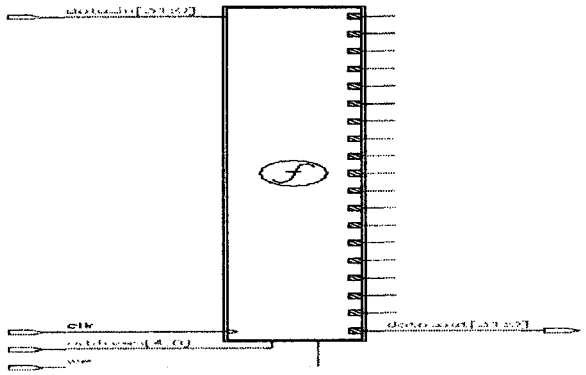

Fig. 5. Block diagram of P-array

3.4 S-box

This paper used four different S-boxes instead of one S-box primarily to avoid symmetries when different bytes of the input are equal, or when the 32-bit input to function F is a bytewise permutation of another 32-bit input. I could have used one S-box and made each of the four different outputs a non-trivial permutation of the single output, but the four S-box designs is faster, easier to program, and seems more secure. The block diagram of S-box is shown in Fig. 6.

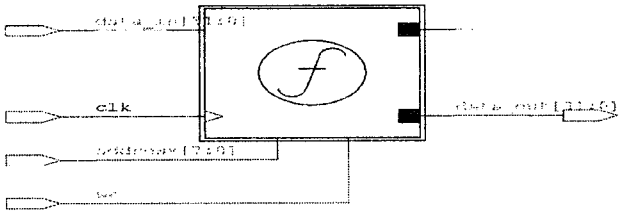

Fig. 6. Block diagram of S-box

4. The Proposed 32bit Adder

As a way to realize adders, there are Ripple Carry Adder, Conditional Sum Adder, Carry Look Adder, and Carry Select Adder. A structure of new adders constructed in this study has all advantages abstracted from other adders. Its size is a little bigger than Carry Select Adder, but it processes faster. The 32bit adder used in this study consists of three ripple carry adders, one 4-bit mux, and one 1-bit mux to make 8bit adders. The block diagram of 8bit adders is shown in Fig. 7.

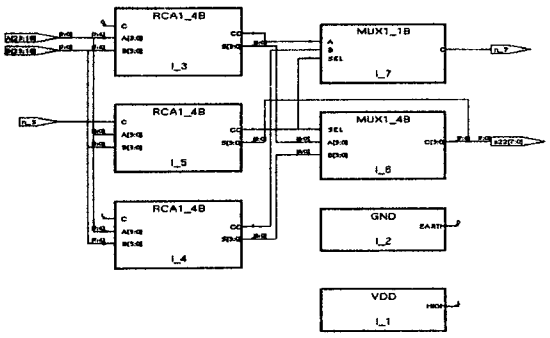

Fig. 7. Block diagram of 8bit adder

32bit adders were constructed with seven 8bit adders and three 8bit MUX. As of movement, one adder was for input = 0 and the other was for input = 1. One of them was selected in terms of the previous value of carry. To construct 32bit, each addition of 8bit was completed and the output carry was connected to input 8bit carry. The output of sum was made through 8bit MUX. The block diagram of 32bit adders proposed in this paper is shown in Fig. 8.

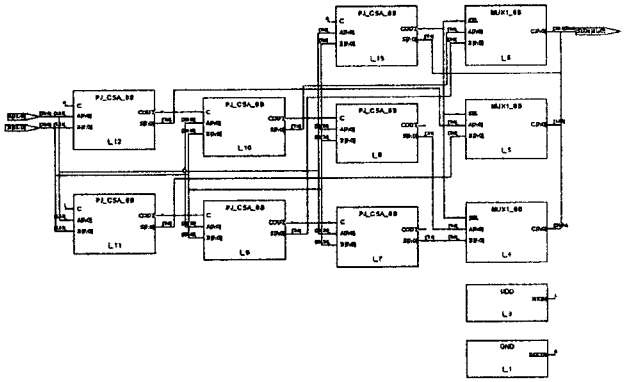

Fig. 8. Block diagram of 32bit adder

5. Resulting and Examination

As for this new adder, area cost increases by 1.06 times and operation speed increases by 1.42 times and operation frequency increases by 1.15 times.

This study examined the adders constructed in many ways as shown in Table 1.

Table 1. Compared results with the conventional and proposed adder.

	Ripple Carry Adder	Carry Select Adder	New Designed Adder
Frequency	28.5	50.2	57.7
Timing	64.08	32.55	22.88
Gate count	64	111	118

The synthesis result of proposed 32bits adder is shown in Fig. 9.

The test bench of blowfish and The Final Block of blowfish are shown in Fig. 10, Fig. 11.

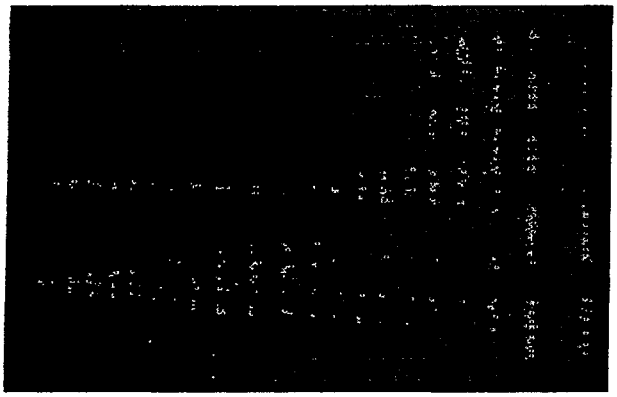

Fig. 9. Synthesis result of proposed 32bits adder

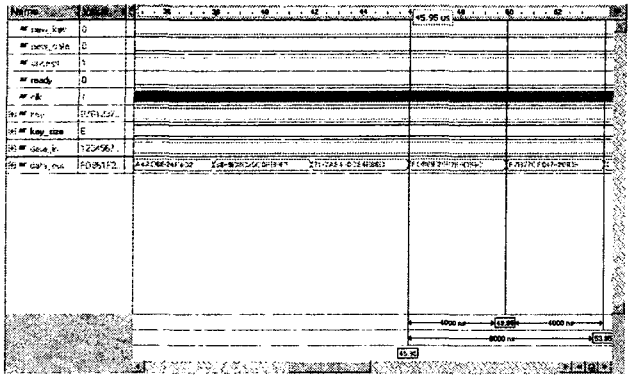

Fig. 10. Test-bench of blowfish

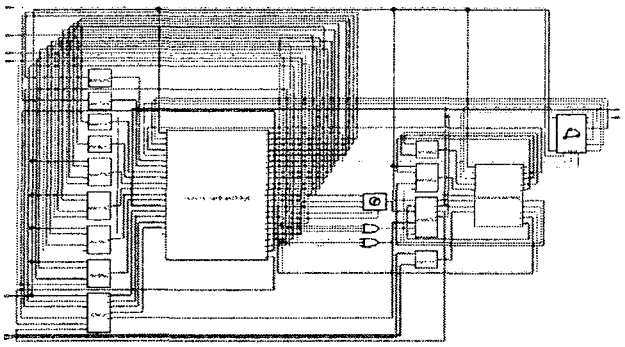

Fig. 11. Final Block of blowfish

In this paper, Simulation and synthesis of all circuits were performed by synopsys design tool.

Synopsys VSS was used for timing of the system. The maximum moving speed was 51.232Mhz and moving speed of stable state was about 50.341Mhz.

6. Conclusions

My implementation of Blowfish has been designed to completely and correctly implement the algorithm with a focus on ease-of-design and ease-of –use without sacrificing too much speed or size. Certainly better implementations could be used and my existing circuit could definitely be optimized.

To increase speed of the system, a new adder method is designed. To solve this problem, a new adder was designed, in which Ripple carry adder is combined with Carry Selector adder to consider efficiency of area and high speed of multiplication respectively. As for this new adder, area cost increases by 1.06 times and operation speed increases by 1.42 times and operation frequency increases by 1.15 times.

Reference

[1] E. Biham and A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.

[2] J. Deamen, R. Govaerts, and J. Vandewalle, "Block Ciphers Based on Modular Arithmetic," Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography, Rome, Italy, 15-16 Feb 1993, pp. 80-89.

[3] GOST 28147-89, "Cryptographic Protection for Data Processing Systems," "Cryptographic Transformation Algorithm," Government Standard of the U.S.S.R., Inv. No. 3583, UDC 681.325.6:006.354. (in Russian)

[4] R.C. Merkle, "Method and Apparatus for Data Encryption," U.S. Patent 5,003,597, 26 Mar 1991.

Jong Tae PARK was born in Haenam, Korea, on March 22, 1970. He received the B.S degree in the Department of Electronic Engineering from Chosun University, Kwangju city, Korea, in 1998. He is currently pursuing the Ph.D. degree in the Department of Electronic Engineering at Chosun University. His interests include VLSI architecture design.

Kang Hyeon RHEE received the Ph.D. degree in the Department of Electronic Engineering at Ajou university, Soowon, Korea, in 1991. He is presently professor at School of Elec. And Info-Comm. Eng., Chosun University, Kwangju city, Korea since 1997. Also now, he is a chairman of Multimedia session in Institute of Electronic Engineering Korea and a vice president of Institute of Webcasting Internet TV of Korea. His interests include multimedia ASIC/VLSI system design, Internet Broadcasting/TV systems and Bio-matrix.