A VLSI Implementation of the Blowfish Encryption/Decryption Algorithm†

Michael C.-J. Lin, Youn-Long Lin

Department of Computer Science
National Tsing Hua University
Hsin-Chu, Taiwan 30043, R.O.C.

Abstract We propose an efficient hardware architecture for the Blowfish algorithm [1]. The speed is up to 4 bit/clock, which is 9 times faster than a Pentium. By applying operator-rescheduling method, the critical path delay is improved by 21.7%. We have successfully implemented it using Compass cell library targeted at a 0.6 μm TSMC SPTM CMOS process. The die size is 5.7x6.1 mm^2 and the maximum frequency is up to 50MHz.

I. Introduction

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.

The Blowfish algorithm was designed by Bruce Schneier in 1993. It is a symmetric block cipher and each block is 64 bits. The secret key of Blowfish cryptography ranges from 32 bits to 448 bits.

Blowfish has been examined for five years. Serge Vaudenay has examined weak keys in Blowfish. Vincent Rijmen's Ph.D. paper includes a second-order differential attack on 4-round Blowfish [2]. The key of the Blowfish algorithm is 448 bits, so it re-quires 2^{448} combinations to examine all keys.

The Blowfish algorithm has many advantages. It is suitable and efficient for hardware implementation. Besides, it is unpatented and no license is required.

The proposed architecture can produce 4-bit data per clock. The scan chains are also included in this architecture for testing. The die size of the chip is 5.7x6.1 mm^2, and the maximum frequency is up to 50MHz.

```
Divide X into two 32-bit halves: XL, XR
For i = 1 to 16:
    XL = XL⊕ Pi
    XR = F(XL) ⊕ XR
    Swap XL and XR
Swap XL and XR (Undo the last swap.)
XR = XR ⊕ P17
XL = XL ⊕ P18
Concatenate XL and XR
```

```
Divide XL into four eight-bit quarters: a, b, c, and d
F(XL) = ( ( S1[a] + S2[b] ) ⊕ S3[c] ) + S4[d]
```

Fig. 1 Blowfish algorithm

II. Blowfish Algorithm

The elementary operators of Blowfish algorithm include table-lookup, addition and XOR. The table includes four S-boxes (256x32bits) and a P-array (18x32bits).

The Blowfish algorithm consists of four steps including table initialization, key initialization, data encryption and data decryption. Fig. 1 shows the Blowfish encryption algorithm

III. The Proposed Architecture

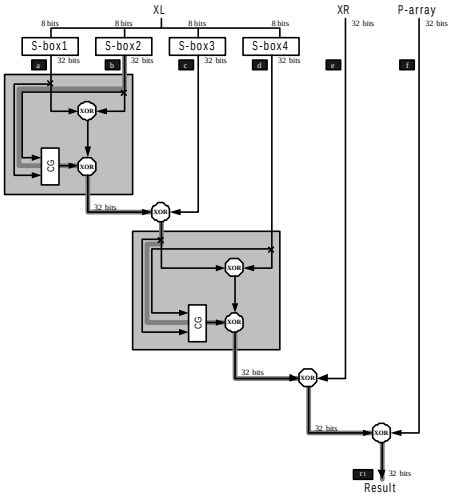

Fig 2. DFG of the loop body

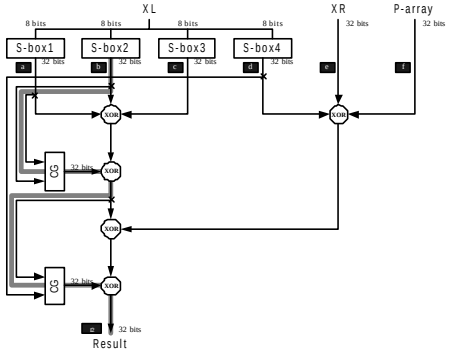

Fig. 3 DFG after rescheduling

A. Operator Rescheduling

When calculating "s = a + b", the i-th bit of s is equal to $a_i \oplus b_i \oplus c_i$, where c_i is the carry-in of i-th bit.

Fig. 2 shows the original DFG of the loop body after replacing the add operation with CG and XOR function.

† Supported in part by the National Science Council, R.O.C, under contract no. NSC 88-2215-E-007-025

The operators include only carry generators and XOR, so we can use operator-rescheduling method to reduce the critical path delay. Fig. 3 shows the result of operator rescheduling.

The gray line in these figures shows the critical path. The original critical path delay is two CG delay plus five XOR delay. After rescheduling, the critical path delay is reduced to two CG delay plus two XOR delay. Three 2-input XOR delays are hidden. According to a synthesizer's report, the improvement of critical path delay is about 21.7%.

B. Fast Carry Generator

The fast carry generator is based on a carry-lookahead adder [3]. We construct the carry generator using hierarchical 4-bit carry generators.

C. The System Configuration

Controller

The controller is implemented as a finite state machine and described in a behavioral Verilog model. See Fig. 4.

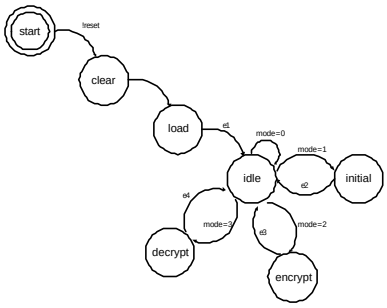

Fig. 4 FSM of the controller

Datapath

It includes ROM modules, SRAM modules, and the main arithmetic units of Blowfish. Fig. 5 shows the datapath architecture.

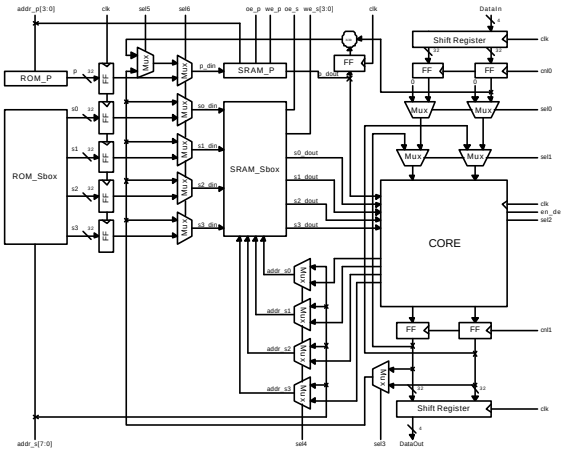

Fig. 5 The architecture of the datapath

Because the size of SRAM module is 2^n words, P1 and P18 are implemented as registers, and the others are mapped to 16x32 bits SRAM. We use a shift register under DataIn to expand 4-bit input to 64-bit input and a shift register over DataOut to reduce 64-bit output to 4-bit output.

CORE implements the loop of the 16-round iteration. A pipeline stage is added to the output of the SRAM modules. The pipeline stage will double the performance of the Blowfish hardware but lead to the overhead of area.

D. DFT Consideration

The testing circuit of the controller is done by adding scan registers to store the signals of the controller and scan out the contents of the registers in test mode.

The datapath is described by Verilog RTL model. All of the flip-flops of the datapath are replaced by scan flip-flops.

IV. Experimental Results

Table 1 shows the feature of this chip. The maximum frequency of this Blowfish cipher chip is 50MHz. Fig. 6 shows the photomicrograph.

Table 1 The chip feature

Die size		5.7 x 6.1 mm^2
Pad		
	Ext. Power	5 vdd 7 gnd
	Int. Power	4 pairs
	Input	12
	Output	6
	Clock buffer	PC5C03
Macro		
	SRAM	256x32(x4), 16x32(x1)
	ROM	256x32(x4)
Random logic		16K gates

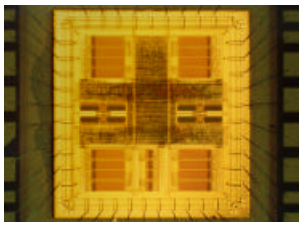

Fig. 6 Photomicrograph

V. Conclusion

The proposed hardware architecture of the Blowfish algorithm can achieve high-speed data transfer up to 4 bits per clock, which is 9 times faster than a Pentium. By applying operator-rescheduling method, the critical path delay is improved about 21.7%. Besides, DFT is also taken into consideration. Specially, the chip is cascadable that means if two chips are used, the performance is double. The test results show that the maximum frequency of this Blowfish cipher chip is 50MHz. The proposed architecture has satisfied the need of high-speed data transfer and can be applied to security device of a system.

Reference

[1] Bruce Schneier, "Applied Cryptography", John Wiley & Sons, Inc. 1996

[2] The homepage of description of a new variable-length key, 64-bit block cipher http://www.counterpane.com/bfsverlag.html

[3] Patterson and Hennessy, "Computer Organization & Design: The Hardware/ Software Interface", Morgan Kaufmann, Inc. 1994