Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/100611

High Throughput Parallel Implementation of Blowfish Algorithm

Soufiane Oukili[*] *and Seddik Bri*

Materials and Instrumentation (MIN), High School of Technology, Moulay Ismail University, Meknes, Morocco

Received: 13 Apr. 2016, Revised: 15 Jul. 2016, Accepted: 19 Jul. 2016
Published online: 1 Nov. 2016

Abstract: Each day, millions of users generate and interchange large volumes of information in various fields. Cryptography plays an important role in preserving the confidentiality of data transmitted over public networks especially with rapidly growth in communication techniques. In the recent years, there is an increasing requirement to implement cryptographic algorithms in fast rising high-speed network applications. In this article, we present high throughput efficient hardware architecture of Blowfish cryptographic algorithm. We have adopted pipeline technique in order to increase the speed and the maximum operating frequency. Therefore, registers are inserted in optimal placements. In addition, the S-box tables of each round of the algorithm have been implemented in block RAMs to allow parallel data encryption. The implementation has been successfully done by virtex-5 (xc5vlx220t) FPGA device using Xilinx ISE 14.7. Our proposed architecture is very fast, it achieves a throughput of 12 Gbps and occupied 1280 slices, whereas the highest reported throughput in the literature as our knowledge is 6.3 Gbps.

Keywords: Cryptography, Blowfish, pipeline, high-throughput, implementation, FPGA

1 Introduction

Cryptography plays an important role in data security against known attacks and decreases the risk of hacking information. The expanding use of digital communications, electronic financial transactions and digital signature applications has put more and more attention on security issues. In this context, cryptographic development has been a high priority and challenging research area in both fields of mathematics and engineering. Blowfish is a symmetric block cipher algorithm, where a same key is used for both encryption and decryption processes. It has been designed by Bruce Schneier in 1993. It takes 64-bit plaintext and variable-length key, from 32 bits to 448 bits, as an inputs and 64-bit ciphertext as an output [1,2].

Blowfish has been identified as a powerful cryptographic algorithm since it can satisfy two basic requirements: high immunity to attacks and relative low algorithm complexity [3]. It is unpatented and no license is required, available free for all uses. Besides, it is suitable and efficient for hardware implementation [4]. The hardware implementation provides greater physical security and higher speed as compared to software implementation [5]. Because of the increasing requirements for high-speed and high-volume secure communications combined with physical security, hardware implementation becomes essential.

Several different methods have been presented in the literature to implement Blowfish algorithm. Lin and Lin [4] proposed hardware architecture of the Blowfish algorithm that can achieve high-speed data transfer up to 4 bits per clock cycle and by applying operator-rescheduling method, the critical path delay is improved by 21.7%. Salomao et al. [6] presented SCOB, a Soft-Core implementation of the Blowfish cryptographic algorithm. This Soft-Core is oriented towards applications demanding a high throughput and exploits both the spatial and the temporal parallelism available in the Blowfish algorithm. Lai and Shu [7] presented a novel VLSI architecture of the Blowfish block cipher. It integrates loop-folding technique combined with four secure modes (ECB, CBC2, CFB2 and OFB2) of operation. The architecture can make data encryption/decryption more efficient and secure. Cody et al. [8] presented robust implementation of Blowfish in hardware. The design utilizes the simplicity of the algorithm to create a relatively straightforward implementation and uses the core-slow library for

[*] Corresponding author e-mail: soufiane.oukili@gmail.com

worst-case scenario analysis. Kumara and Benakop [9] proposed four different implementations of Blowfish algorithm and analyzed the performance of it with and without Wave Dynamic Differential Logic (WDDL) style to provide security against Differential Power Analysis (DPA) attack. Guerrero and Noras [10] presented fast Blowfish encryption in hardware with an throughput of 1032 Mbps. Sudarshan et al. [11] presented flexible architecture for Blowfish algorithm called Dynamic reconfiguration, Replication, Inner loop pipeline, Loop folding architecture abbreviated as DRIL. DRIL Architecture aims at efficient utilization of hardware through replication and loop folding, higher throughput through replication and inner loop pipeline, flexibility through dynamic reconfiguration and replication. Kumar and Baskaran [12] proposed low power, area and high throughput 4-stage pipelined implementation of the Blowfish cryptographic algorithm. Joshi et al. [13] proposed implementation of Blowfish algorithm with modifying its function. Swagata et al. [14] presented pipelined implementation of Blowfish.

In this article, we present high throughput efficient hardware architecture and implementation of Blowfish algorithm. Pipeline technique is introduced in order to increase the speed and the maximum operating frequency. The pipelining strategy consists in parallelizing the data inputs and outputs with the processing. Consequently, the algorithm is divided into stages and registers are placed. By incrementing the number of these stages, the critical path is decreased and as a result the throughput is increased. Furthermore, in each encryption round of the algorithm, S-box tables have been implemented in block RAMs to allow parallel data encryption. Our proposed design is implemented on Xilinx Virtex-5 FPGA device. The FPGAs offer the advantage of hardware speed and software flexibility and programmability.

This article is structured as follows. Section 2 and 3; present a background and cryptanalysis of the Blowfish algorithm, respectively. Our proposed Blowfish architecture is presented in section 4. Section 5 provides implementation summary and comparison between our implementation and different reported implementations. Finally, conclusion is given in Section 6.

2 Background of Blowfish algorithm

Blowfish is a symmetric block cipher. It has a fixed 64-bit data block size and a variable secret key range from 32 bits to 448 bits. The algorithm consists of two parts: key expansion and data encryption. The key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes. The 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 [1].

2.1 Key expansion

Blowfish uses a large number of subkeys (eighteen 32-bit P-array and four 32-bit S-boxes with 256 entries each). These subkeys must be calculated before any data encryption or decryption using the Blowfish algorithm. The method is as follows:

1: Initialize first the P-array and then the four S-boxes with hexadecimal digits pi.

2: XOR P1 with the first 32-bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (up to P18).

3: Encrypt the all zero string with the blowfish algorithm, using the keys described in steps (1) and (2).

4: Replace P1 and P2 with the output of step (3).

5: Encrypt the output of step (3) using the Blowfish algorithm with the modified keys.

6: Replace P3 and P4 with the output of step (5).

7: Continue the process, replacing all elements of the P-array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

2.2 Data encryption

As mentioned previously, Blowfish is a Feistel network consisting of 16 rounds, as shown in figure 1. The inputs are 64-bit plaintext and 18 P-array sybkeys (32 bits). The output is a ciphertext (64 bits). The Blowfish algorithm is described in algorithm 1.

```
Data: Plaintext (64 bits) and P1, P2, , P18
Result: Ciphertext (64 bits)
Divide plaintext into two 32-bit halves: XL , XR;
for i=1 to 16 do
    Calculate XL = XL XOR Pi;
    Calculate XR = F(XL) XOR XR;
    Swap XL and XR (undo the last swap);
end
Calculate XR = XR XOR P17;
Calculate XL = XL XOR P18;
Recombine XL and XR (ciphertext);
```

Algorithm 1: Blowfish

Function F is calculated by equation (1). It divides XL into four eight-bit quarters: a, b, c, and d. These quarters are used as input to the S-boxes. The outputs are added $(modulo\ 2^{32})$ and XORed to produce the final 32bit output. This is shown in figure 2.

$$F(X_L) = ((S_{1,a} + S_{2,b}\ mod\ 2^{32}) XOR\ S_{3,c}) + S_{4,d}\ mod 2^{32} \quad (1)$$

Decryption is exactly same as encryption, except that P1, P2 ... P18 are used in the reverse order.

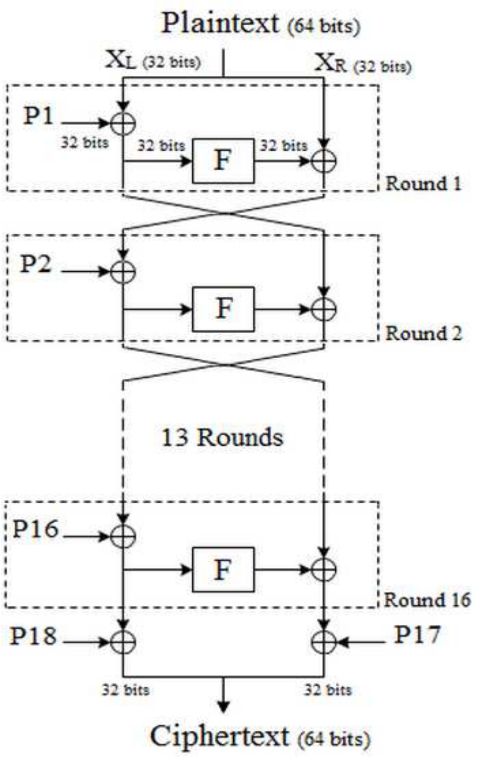


Fig. 1: Block diagram of Blowfish

3 Cryptanalysis

Cryptanalysis refers to the process of illegally attempting to recover the plaintext (or the key) that corresponds to a particular ciphertext. Full-round version of Blowfish algorithm is invulnerable against cryptanalysis, to date. Numerous attack schemes have been proposed to break the cryptographic system and extract secret information, but none succeeded. John Kesley could only break 3-round of Blowfish and his cryptanalysis cannot be extended beyond 3 rounds. Serge Vaudenay examined a simplified variant of Blowfish, with the S-boxes known and not key-dependent. For this variant, a differential attack can recover the P-array with $2^{(8r+1)}$ chosen plaintexts, where r is the number of rounds. This attack is impractical in reality and does not work against 8-round Blowfish and higher, since more plaintext is required than can possibly be generated with a 64-bit block cipher [3]. In 1996, Vincent Rijmen proposed a promising attack in his doctoral dissertation, but it can only break 4 rounds of Blowfish and no more [15].

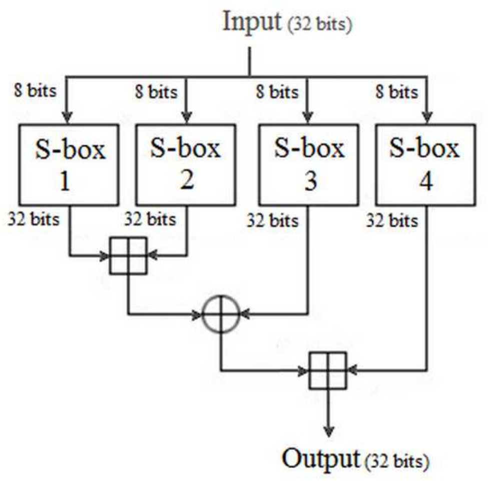


Fig. 2: Function F

4 Proposed Blowfish architecture

The main objective of our proposed architecture is to design high throughput efficient Blowfish algorithm for hardware implementation. Therefore, pipeline and parallel encryption techniques are introduced.

The pipeline technique modifies the critical path by increasing the possible frequency of clock cycle. It consists in parallelizing the data inputs and outputs with the processing. Consequently, the algorithm is divided into stages and registers are placed. As a result of this, the throughput can be increased. As mentioned before, Blowfish is an16-round Feistel network cipher. Thus, we have inserted 16 64-bit registers after each round. Moreover, two registers (64-bit) after the plaintext and before the ciphertext are placed. This is shown in figure 3. To reduce more the critical path, we have added registers in every round, as shown in figure 4. This will further increase the throughput of the algorithm. Note that the increase of throughput requires an increase in area, as registers are required to store intermediate results. The elements of the P-array (18x32 bits) and the four S-boxes (256x32 bits) subkeys are stored in block RAMs. To make a parallel encryption, these S-boxes are duplicated in all 16 rounds. To our knowledge, this is the first article that proposes this manner for implementing Blowfish S-boxes. This will tremendously increase the performance of the architecture. The ciphertext takes 34 clock cycles latency, first time only. Then we recover it at each clock cycle.

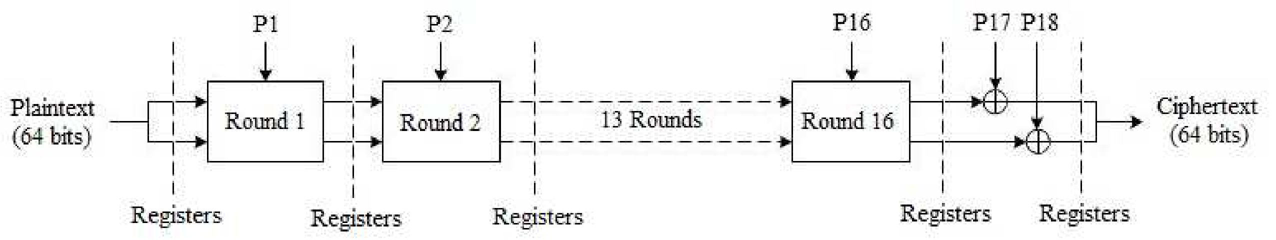


Fig. 3: Proposed pipelined Blowfish

In our proposed architecture, we used pipelining and parallelism in order to break the critical path delay and to obtain high encryption throughput at the expense of area as compared with the normal non-pipelined Blowfish algorithm.

5 Implementation summary and comparison

FPGA implementation of our proposed Blowfish architecture was established on Virtex-5 device (xc5vlx220t-2ff1738) using Xilinx ISE Design Suite 14.7 as synthesis tool. The device utilization summary is given in table (1). The design was described using VHDL language and simulated with Modelsim 6.1f. The first encrypted data takes 34 clock cycles latency. Then, we recover the forthcoming encrypted data at each clock cycle. Simulation window is shown in figure (5). The design achieves a maximum clock frequency of 187.63 MHz (5.33 ns), a throughput of 12 Gbps and an efficiency of 9.38 Mbps/slice. We employ well-known equation (2) and equation (3) to calculate the throughput and the efficiency, respectively.

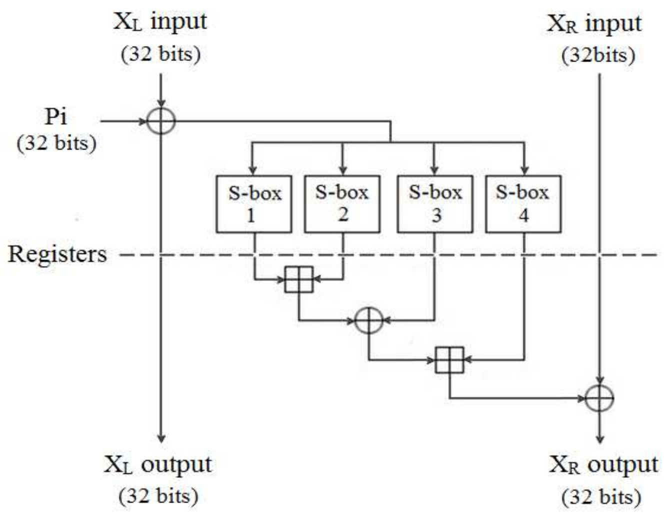


Fig. 4: Proposed pipelined round i

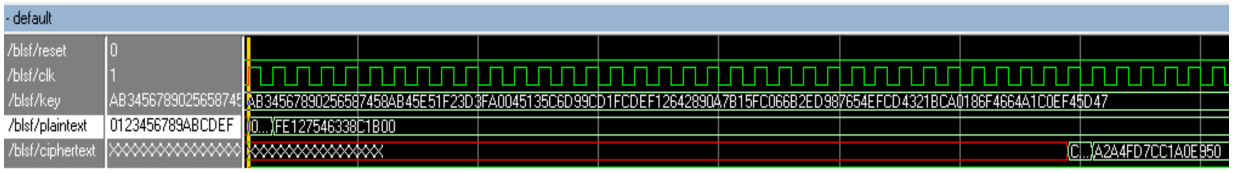
Table 1: Device utilization summary (xc5vlx220t-2ff1738)

Resources	Utilization
Number of slices	1280/ 34560 3%
Number of slice LUTs	3263/138240 2%
Number of slice registers	3002/138240 2%
Number of bonded IOBs	579/680 85%
Number of block RAMs	79/212 37%
Total equivalent cells	9525
Minimum period	5.33 ns
Maximum frequency	187.63 Mhz

$$Throughput = \frac{Number\ of\ outputed\ bits}{Delay\ of\ the\ critical\ path} \qquad (2)$$

$$Efficiency = \frac{Throughput}{Utilised\ slices} \qquad (3)$$

There are several implementations for the Blowfish algorithm that aim to achieve the most efficient architecture, by improving high throughput and area-efficient. Table (2) shows the performance figures for some reported architectures up to our best knowledge. It provides values of hardware utilization, maximum frequency, throughput and the increase in throughput of the proposed architecture compared to the existing techniques, by a factor of.



```
- default
/blsf/reset       0
/blsf/clk         1
/blsf/key         AB3456789025658745  AB34567890256587458AB45E51F23D3FA0045135C6D99CD1FCDEF12642890A7B15FC066B2ED987654EFCD4321BCA0186F4664A1C0EF45D47
/blsf/plaintext   0123456789ABCDEF    0... FE127546338C1B00
/blsf/ciphertext  XXXXXXXXXXXXXX      XXXXXXXXXXXXXX                                  C... A2A4FD7CC1A0E950
```

Fig. 5: Simulation Window of our proposed Blowfish design

Table 2: Hardware utilization, maximum frequency and throughput results

Architectures	Size (Standard Cells)	Maximum frequency (Mhz)	Throughput (Mbps)	Increase in Throughput [a]
Lin and Lin [4]	16000	50	200	60
Salomao et al. [6]	4620	66	266	45.14
Lai and Shu [7]	13000	72	288	41.7
Cody et al. [8]	4996	167	590	20.35
Kumara and Benakop [9]	-	13.09	840	14.3
Guerrero and Noras [10]	-	-	1032	11.63
Sudarshan et al. [11]	-	146.515	1545	7.77
Kumar and Baskaran [12]	5986	167	2670	4.5
Joshi et al. [13]	4608	-	3680	3.26
Chatterjee et al. [14]	-	295.63	6300	1.9
Proposed design	9525	187.633	12008	-

[a] Increase in throughput of our proposed design compared to existing techniques (by a factor of)

As can be observed from table (2), the highest throughput reported to our knowledge is 6.3 Gbps with an efficiency of 1.955 Mbps/slice [14]. By comparing these results with our proposed implementation, we see that ours gives 1.9 times more throughput. Furthermore, it is 5.1 times more efficient. Therefore, we notice that our implementation reaches the highest throughput with efficiency in compare to the reported implementations.

6 Conclusion

In this paper, we present high throughput efficient Blowfish architecture. It integrates pipeline technique to break the critical path delay and increase speed. Moreover, S-boxes stored in block RAMs at each round of the algorithm are introduced to perform a parallel encryption. The input can be loaded every clock cycle and after an initial delay of 34 clock cycles, the encrypted data will appear consecutively. The implementation is done by virtex-5 FPGA device and achieves an encryption rate of 12 Gbps. The results show that our proposed architecture of Blowfish algorithm provides better performance in terms of throughput than the previous implementations at the cost of increasing the area little more.

References

[1] B. Schneier, Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). Cambridge Security Workshop Proceedings, 191204 (1993).

[2] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley and Sons, New York, 1996.

[3] B. Schneier, The Blowfish Encryption Algorithm-One Year Later. Dr. Dobb's Journal, 1995. Available online at: https://www.schneier.com/cryptography/archives/1995/09/the_blowfish_encrypt.html, (accessed 04.14.2016).

[4] M.C.J Lin and Y.L Lin, A VLSI Implementation of the Blowfish Encryption/Decryption Algorithm. Asia and South Pacific Design Automation Conference, 1-2 (2000).

[5] S.M Yoo, D. Kotturi, D.W Pan and J. Blizzard, An AES crypto chip using a high-speed parallel pipelined architecture. Microprocessor and Microsystem 29, 317-326 (2005).

[6] S. L. C Salomao, J. M. S de Alcantara, V. C Alves and A. C. C Vieira, SCOB, a Soft-Core for the Blowfish Cryptographic Algorithm. XII Symposium on Integrated Circuits and Systems Design, 220-223 (1999).

[7] Y. K Lai and Y. C Shu, VLSI ARCHITECTURE DESIGN AND IMPLEMENTATION FOR BLOWFISH BLOCK CIPHER WITH SECURE MODES OF OPERATION. The 2001 IEEE International Symposium on Circuits and Systems 4, 57-60 (2001).

[8] B. Cody, J. Madigan, S. MacDonald and K. W Hsu, High Speed SOC Design for Blowfish Cryptographic Algorithm .IFIP International Conference on Very Large Scale Integration, 284-287 (2007).

[9] S.V Kumara and P. Benakop, HIGH THROUGHPUT AND HIGH SPEED BLOWFISH ALGORITHM FOR SECURE INTEGRATED CIRCUITS. Anale. Seria Informatic 12, 24-29 (2014).

[10] F. Guerrero and J.M Noras, Implementing block ciphering algorithms in hardware. International Journal of Electronics 83, 581-598 (1997).

[11] T.S.B Sudarshan, R.A Mir and S. Vijayalakshmi, DRIL A Flexible Architecture for Blowfish Encryption Using Dynamic Reconfiguration, Replication, Inner-Loop Pipelining, Loop Folding Techniques. Advances in Computer Systems Architecture 3740, Springer Berlin Heidelberg, 625-639 (2005).

[12] P.K Kumar and K. Baskaran, An ASIC implementation of low power and high throughput blowfish crypto algorithm. Microelectronics Journal 41, 347-355 (2010).

[13] T. Joshi, R. Yadav and U. Malviya, Design of enhanced speed Blowfish Algorithm for cryptography with merged encryption & decryption in VHDL. International Journal of Engineering Research and Applications, Special issue ICIAC, 68-71 (2014).

[14] S. R. Chatterjee, S. Majumder, B. Pramanik and M. Chakraborty, FPGA Implementation of Pipelined Blowfish Algorithm. Fifth International Symposium on Electronic System Design, 208-209 (2014).

[15] V. Rijmen, Cryptanalysis and design of iterated block ciphers. Doctoral thesis, Katholieke Universiteit Leuven (1997).

Soufiane Oukili received his engineering degree in Electrical Engineering from the National School of Applied Sciences, Mohammed First University, Morocco and he is currently a PhD Student at the Faculty of Sciences, Moulay Ismail University, Morocco. His fields of interest are FPGA design, Communications Security and parallel and distributed Computing.

Seddik Bri is a Professor at the Electrical Engineering Department in High School of Technology (ESTM), Moulay Ismail University, Meknes -Morocco. His scientific research interests are the microwaves applications and the security in communications systems.