International Journal on Future Revolution in Computer Science & Communication Engineering ISSN: 2454-4248
Volume: 3 Issue: 11 548 – 552

# VHDL Implementation of 128 bit Pipelined Blowfish Algorithm

Dakey Rahul Khanna[1]
Student, M.Tech , VLSISD,Dept. of ECE,
Swarnandhra College of Engg. &Tech.
Narsapur , A.P,India.
*e-mail: rahulkhanna556@gmail.com*

N.Srikanth[2]
Associate Professor, Dept. of ECE,
Swarnandhra College of Engg. &Tech.
Narsapur , A.P,India.
*e-mail: srikanth648@yahoo.co.in*

Dr. B.Subrahmaneswara Rao[3]
Professor, Dept. of ECE
Swarnandhra College of Engg. &Tech.
Narsapur , A.P,India.
*e-mail: boyina_a@yahoo.com*

***Abstract***— Communication through public networks imposes threat to our sensitive data. Information security plays an important role in public networking and wireless communication. In order to achieve the protection of information or private data in networking, Cryptography can be used. It is the automated method in which security goals are accomplished. Cryptographic algorithm is a mathematical function used in Encryption and Decryption process. Blowfish is a keyed symmetric Cryptographic algorithm. It is a very fast and useful scheme for Encryption and Decryption. A key is the strongest point of any algorithm but it can become the weakest point if it is not secured. Our information can be secured if it is encrypted by using multiple keys. Hence implementation of Blowfish algorithm is very much use. Usually blowfish in existing method is 64 bit block cipher and Throughput depends on the size of the blocks applied. In this proposed paper, the blowfish algorithm is designed for 128 bit block size and pipelining operation is carried to improve the speed and reduce the delay accordingly, and so that this architecture improves the throughput of an encoder. The implementation results indicate that the proposed pipelined architecture shows 10% of improvement in Throughput. VHDL Implementation of proposed architecture has done by using XILINX ISE 9.1.

***Key words****: Cryptography, blowfish, pipelined architectures.*

*****

## I. INTRODUCTION

Embedded devices are becoming more popular and used in network communications. These devices are now used commonly for storing data and exchange of data through public networks. Wireless networking imposes threat to the sensitive data. To protect sensitive data against threat embedded devices are designed with inbuilt security features. The sensitive data is encrypted before transmission so that only authorized user can have access to such information. Hardware implementation of encryption algorithm is helpful in designing secured Embedded System.

Modern embedded systems need data security more than ever before. It is now common to see portable electronic devices of all sorts, ranging from cellular phones to military radio systems and from inventory tracking systems to medical record tablets. Devices like PDAs and cell phones store personal e-mail and contact lists [1]. GPS receivers keep logs of our movements and our automobiles record our driving habits. User-friendly products are in demand which can be reprogrammed during normal use and add new features as firmware upgrades become available. The user expects that the system should be protected from unauthorized access.

Embedded device manufacturers expect protection of their systems from unauthorized duplication or reverse engineering. Cryptography can be used to secure private data and keep it private. Encrypted data transmission will protect contact lists and personal e-mail from unauthorized or unintended persons. Firmware upgrades can be encrypted so that only intended user can use it and other devices are prevented from using it without proper authentication. In certain applications data security from tampering is mandatory by law. The applications like electronic engine controllers are required to be protected from tampering emission and performance data. Devices used to store medical data and utility meter like taxi, gas, electricity must be tamperproof. Devices can be secured with passwords, identification numbers, or tokens. The devices can be designed with better security using cryptography.

## II. RELATED WORK

It was concluded in [8] that AES is faster and more efficient than other encryption algorithms. When the trans-mission of data is considered there is insignificant difference in performance of different symmetric key schemes (most of the resources are consumed for data transmission rather than computation). Even under the scenario of data transfer it would be advisable to use AES scheme in case the encrypted data is stored at the other end and decrypted multiple times.

A study in [9] is conducted for different popular secret key algorithms such as DES, 3DES, AES, and Blowfish. They were implemented, and their performance was compared by encrypting input files of varying contents and sizes. The algorithms were tested on two different hardware platforms, to compare their performance. They had conducted it on two different machines: P-II 266 MHz and P-4 2.4 GHz. The results showed that Blowfish had a very good performance compared to other algorithms. Also it showed that AES had a better performance than 3DES and DES. It also shows that 3DES has almost 1/3 throughput of DES, or in other words it needs 3 times than DES to process the same amount of data [10].

A study in [11] provides evaluation of six of the most common encryption algorithms namely: AES, DES, 3DES, RC2, Blowfish, and RC6. A comparison has been conducted for those encryption algorithms at different settings for each algorithm such as different sizes of data blocks, different data types, battery power consumption, different key size and finally encryption/decryption speed. Several points can be concluded from the Experimental results. Firstly; there is no significant difference when the results are displayed either in hexadecimal base encoding or in base 64 encoding. Secondly; in the case of changing packet size, it was concluded that Blowfish has better performance than other common encryption algorithms used, followed by RC6. Thirdly; we find that 3DES still has low performance compared to algorithm DES. Fourthly; we find RC2, has disadvantage over all other algorithms in terms of time consumption. Fifthly; we find AES has better performance than RC2, DES, and 3DES. In the case of audio and video files we found the result as the same as in text and document. Finally, in the case of changing key size, it can be seen that higher key size leads to clear change in the battery and time consumption.

## III. ALGORITHM

There are two parts to this algorithm; a part that handles the expansion of the key and a part that handles the encryption of the data.The first step in the algorithm is to break the original key into a set of subkeys. Specifically, a key of no more than 448 bits is separated into 4168 bytes. There is a P-array and four 32-bit S-boxes. The P-array contains 18 32-bit subkeys, while each S-box contains 256 entries.

The following steps are used to calculate the subkeys:

- Initialize the P-array and S-boxes
- XOR P-array with the key bits. For example, P1 XOR (first 32 bits of key), P2 XOR (second 32 bits of key),
- Use the above method to encrypt the all-zero string
- This new output is now P1 and P2
- Encrypt the new P1 and P2 with the modified subkeys.
- This new output is now P3 and P4
- Repeat 521 times in order to calculate new subkeys for the P-array and the four S-boxes

Blowfish has a 64-bit block size and a key length of anywhere from 32 bits to 448 bits. It is a 16-round Feistel cipher and uses large key-dependent S-boxes. It is similar in structure to CAST-128, which uses fixed S-boxes. Here is the visual representation of this encryption algorithm.

## IV. PROPOSED DESIGN

There are software and hardware approaches to implement cryptographic Blowfish algorithm. Hardware implementation provides greater physical security and higher speed as compared to software implementation. Because of the increasing requirement to implement cryptographic algorithms in fast rising high-speed network applications combined with physical security, hardware implementation becomes essential.

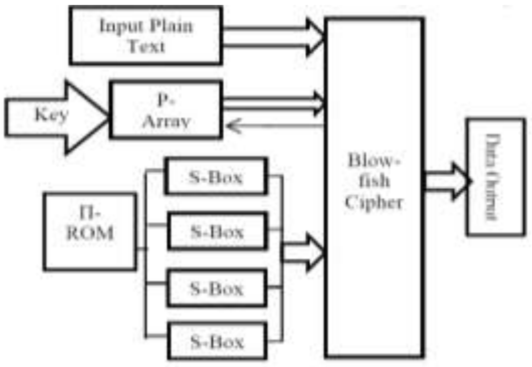

Figure 1: Blowfish Algorithm Design Blocks

The encrypter is required to be designed as integrated part of the system in order to attain superior performance. Embedded system designs are constraint by space, memory and time. Lookup tables are considered for S box and P array implementations which will generate keys fast and precomputed values can be retained for encryption and decryption. The symmetric nature of Blowfish helps in reutilizing hardware for encryption and decryption which further reduces area requirement or space. Blowfish is feistel network. This simplifies encryption and decryption realization in hardware.

Input plain text is applied to the core Blowfish cipher block. Input will be processed as block of 64 bits. Each 64 bit data further bifurcated into two 32bit data, in order to apply to the cipher block. While applying data to the feistel block Padding will be done for framing data as multiple of 8 bits. Π ROM will give initial key values to S Box. 32bit are divided into four groups of 8 bit each. It acts as address for selecting precomputed substitution values of S box. Permutation values are computed using lookup table. Data obtained from four S-boxes further underwent to Arithmetic operations to get 32bit random data. The sub key of length 32 bit obtained from P array elements which are XORed and given to F function.

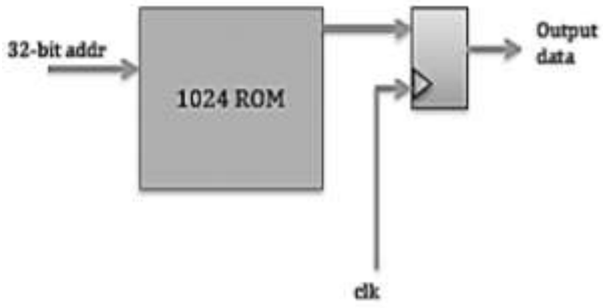

Figure 2: Proposed Memory-based Method

The architecture of the proposed Blowfish consists of a 128-bit block size and 64bit key size, whereby it comprises two parallel blocks of 64-bit Blowfish algorithm that are simultaneously executed. This design technique enables the throughput of the Blowfish algorithm to be maximized. The parallel blocks share the same *S-box* that is used for the F function. As the implementation of the Blowfish design is targeted to reduce the core size and timing delay, The implementation for the blow fish using S-boxes can be replaced with the proposed method uses a read-only memory (ROM) that contains 1024 × 32-bit input data of *addr*. The *addr* represents the data of four 32-bit *S-boxes*. The 32-bit output data are read from the ROM. The proposed method can also lessen the total of slices used by the Blowfish design. A slice contains a set number of look-up tables (LUTs), FFs, and multiplexers. Thus, less logic resources are used to perform logic, arithmetic, and ROM functions that can lead to a faster encryption/decryption process.

This implementation of Hardware block can be reused two times to perform 128bit operation by means of selector and de-selector blocks aided to the above design. That is the proposed implementation of the Blowfish algorithm. The design can be further modified to a pipelined implementation of blowfish algorithm for improved performance as shown in the figure.

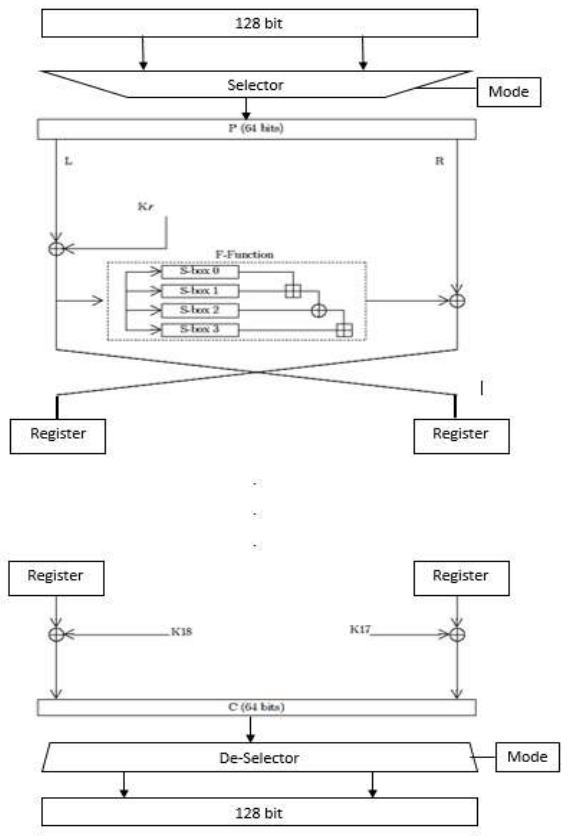

Figure 3: Block Diagram of improved throughput pipelined blowfish Algorithm

If we observe the feistel block implementation, which is the building block, in hardware using VHDL the critical path delay will be of the order of 21.618ns. The building block comprised of logic for performing XOR operation between key and data and feistel function using S-boxes.

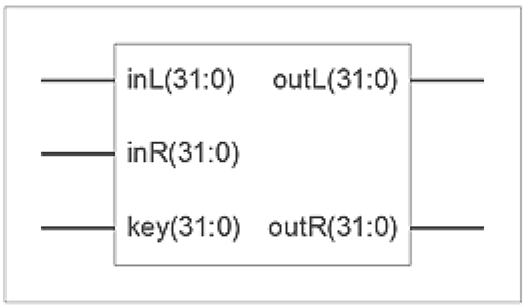

Figure 4: Black box of building block of the Design implementation.

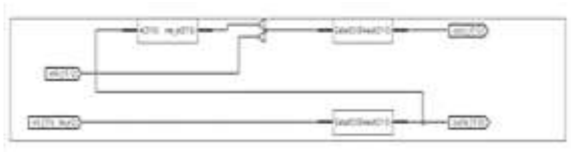

Figure 5: internal implementation of building block.

As the single block to perform 64bit operation requires 21.618ns, to perform 128bit operation by reusing the block for two times requires 43.236ns. Hence pipelining can be introduced by inserting registers between every two building blocks and make sure that the time lag between two clock ticks must be at least 43.236ns. Hence the proposed improved throughput pipelined blowfish Algorithm can work at the application of maximum clock frequency of 23.128MHz.

## V. RESULTS AND COMPARISONS

64bit fixed block cipher implemented in standard blowfish hardware architecture by make use of 4 S-boxes whose entries are calculated from the key and are independent of both data and key while executing. The RTL (Register Transfer Level) diagram and Design summary of implemented Blowfish Algorithm are as shown below. The hardware description done using VHDL and the synthesis of the design done for Vertex E FPGA.

Figure 6: Simulated Waveform Diagram of 64bit blowfish Algorithm

The black box of improved throughput blowfish Algorithm is as shown below. It generates 128bit cipher output from 64bit design by time sharing using select input.

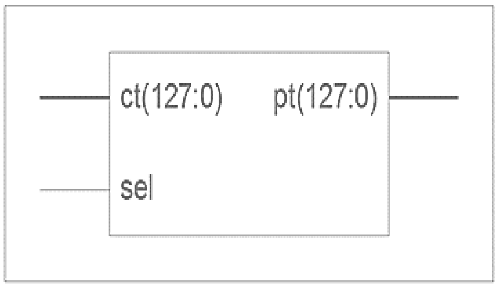

Figure 7: Black Box of area efficient blowfish Algorithm.

The black box of implemented improved throughput pipelined blowfish Algorithm is as shown below. Here the select input makes the 64bit design to operate twice in order produce 128bi output. The clock signal drives the registers separating the combinational building block of blowfish hardware architecture and triggers with approximate minimum lapse of 50ns. The select signal trigger once between two clock triggers. The clear signal clears all the registers whenever asserted.

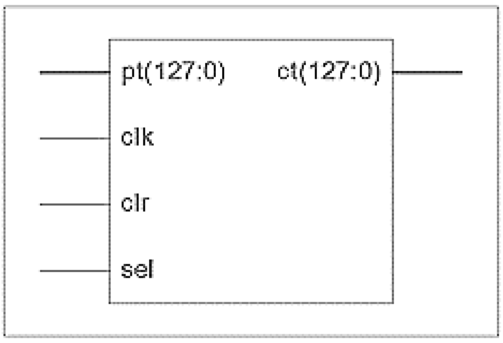

Figure 8: Black Box of Improved throughput pipelined blowfish Algorithm.

The cipher result obtained from the plain data applied to the implemented design for two inputs is as given below, showing minimum devotion of 56 of128.

Plaintext:
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
0000000000000000000000000000

Ciphertext:
00010011110000010111011100110110001001000001011000
00101101010011000100111100000101110111001101100010
0100000101100000101101010011

Plaintext:
11111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
1111111111111111111111111111

Ciphertext:
01101111011000001101101001111110001110111011000001
00101111111011011011110110000011011010011111100011
1011101100000100101111111011

Here by comparing result obtained from 64bit fixed block size blowfish algorithms implemented using S-box method and ROM for feistel network, we can conclude that the ROM based design is optimised to produce less delay and lesser equivalent gate count. The logic utilization for the both implementations are almost same.

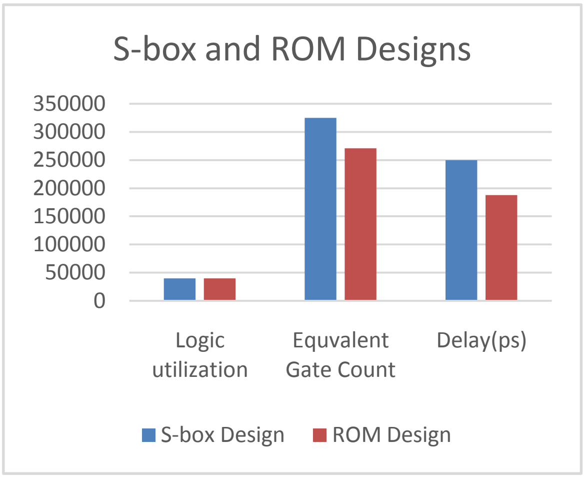

Figure 9: Comparison between 64bit blowfish algorithms using S-box and ROM

Throughput is defined as the average rate of successful message delivery over a communication channel. Throughput is directed toward evaluating each architecture's characteristic and performance. Throughput is calculated as Eq.1.

$$\text{Throughput (Gbps)} = \frac{128\ \text{bits} * \text{Clock Frequency (MHz)}}{\text{Latency}}$$

Latency is the encryption/decryption time that is calculated in clock cycles. Latency should be as small as possible to achieve a power-saving system. Furthermore, a long battery life is necessary, particularly for mobile devices.

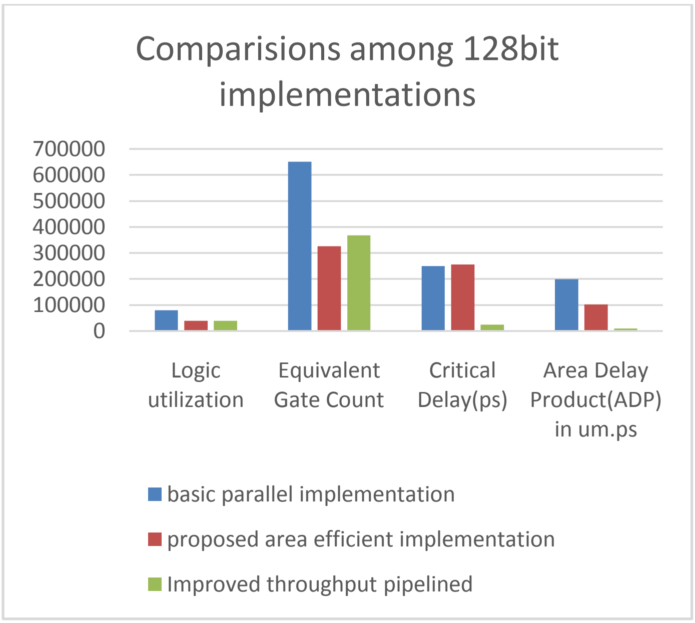

Figure 10: Comparison among implemented 128bit blowfish algorithm implementations.

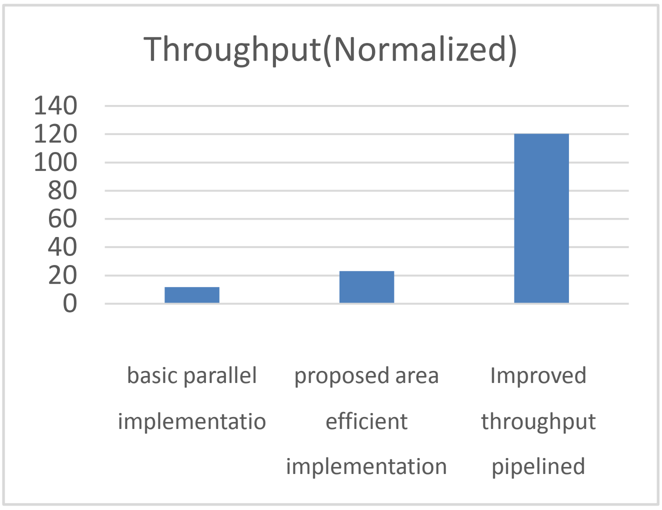

Figure 11: Throughput Comparison among implemented 128bit blowfish algorithms

From the comparison results shown above, the proposed improved throughput method of implementation shows off less requirement of LUTs (Look up Tables), equivalent gates when compared to parallel implementation method. The evaluation delay may be little higher due to inclusion of selector and de-selector, but overall performance metric ADP is lower for proposed design which is the index for improved performance. The proposed design is further modified to get improved throughput pipelined blowfish architecture, which shows off extremely optimised performance. Results in less logic utilisation, less delay and lesser Area Delay Product for superior performance. Still it seems to occupy more equivalent gate count than the proposed design due to inclusion of Registers.

The performance metric throughput is seems to be doubled for proposed implementation, due to parallel implementation of hardware, whereas for pipelined architecture throughput is n times, where n is the number of pipeline stages. Overall performance of the proposed improved throughput pipelined architecture is better compared to others.

## VI. CONCLUSION

Implemented 64bit fixed block size blowfish algorithm using VHDL. Simulated for satisfactory result in MODELSIM and synthesized in XILINX ISE simulator, also implemented feistel network of blowfish algorithm with ROM and observed the performance improvement. The implementation of 128bit blowfish algorithm using proposed improved throughput method, proposed area efficient method and proposed improved throughput pipelined method is done and obtained results shows that the proposed improved throughput pipelined method poses superior performance. The implementation results indicate that the proposed pipelined architecture shows 10% of improvement in Throughput.

## REFERENCES

[1] Ravi, Srivaths, et al. "Security in embedded systems: Design challenges." ACM Transactions on Embedded Computing Systems (TECS) 3.3 (2004): 461-491.

[2] Arya, S. "An Implementation of Blowfish Algorithm Using FPGA." International Journal of Engineering Research and Technology. Vol. 2. No. 8 (August-2013). IJERT, 2013.

[3] Patel C.R., Gohil N.B., Shah V.“FPGA - Hardware Based DES & Blowfish Symmetric Cipher Algorithms For Encryption & Decryption Of Secured Wireless Data Communication”, Journal of Information, Knowledge and Research in Electronics and Communication Engineering, 2(2), 739–744. (2013).

[4] Dakate, Deepak Kumar, and Pawan Dubey. "Blowfish encryption: A comparative analysis using VHDL." International of Engineering and Advanced Technology (IJEAT) 1.5 (2012): 177-179.

[5] W. Stalling, “Cryptography and Network Security Principles and Practices”, Prentice Hall, 4th ed., 2005.

[6] A. Kahate, “Cryptography and Network Security”, Tata McGraw Hill, 2nd ed., 2007.

[7] National Institute of Standards and Technology, Federal Information Processing Standards Publication 46-3: Data Encryption Standard, 1999.

[8] S. Hirani, Energy Consumption of Encryption Schemes in Wireless Devices Thesis, University ofPittsburgh, Apr. 9,2003, Retrieved Oct. 1, 2008.

[9] A. Nadeem, \A performance comparison of data encryption algorithms," IEEE Information and Communication Technologies, pp. 84-89, 2006.

[10] Results of Comparing Tens of Encryption Algorithms Using Di®erent Settings- Crypto++ Benchmark,RetrievedOct. 1, 2008. (http://www.eskimo.com/ weidai/benchmarks.html)

[11] DiaaSalamaAbdElminaam, et,al. “Evaluating The Performance of SymmetricEncryption Algorithms” International Journal of Network Security, Vol.10, No.3, PP.213{219, May 2010.