JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

# Implementation of 128 bit Pipelined Blowfish Algorithm and Performance Improvement

[1]Satibha Sahu, [2]Prof. Santosh Onker

[1]Research Scholar, [2]Assistant Professor

Department of Electronics and Communication Engineering,
SAM College of Engineering & Technology, Bhopal, India

***Abstract :*** Cryptography is best known as a method for keeping the substance of a message mystery. The designed Blowfish as a general-purpose algorithm intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. The focus of proposed research is to implement built in self test using verilog coding on xilinx 14.7software. The main motivation behind for proposed algorithm is to extend the existing blowfish and improve performance. Previously it is designed for the 64 bit encryption and decryption process. Presently it is designed for the 128 bit processing. Calculate parameters using standard formula and approach and compare from existing work.

***IndexTerms* – Blowfish, Cryptography, DES, Xilinx, Encryption, Decryption, Delay, Power.**

## I. INTRODUCTION

The Blowfish is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in many cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date. However, the Advanced Encryption Standard (AES) now receives more attention, and Schneier recommends Twofish for modern applications.

Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. At the time Blowfish was released, many other designs were proprietary, encumbered by patents or were commercial or government secrets. Schneier has stated that, "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone.

Blowfish has a 64-bit block size and a variable key length from 32 bits up to 448 bits.[3] It is a 16-round Feistel cipher and uses large key-dependent S-boxes. In structure it resembles CAST-128, which uses fixed S-boxes.

### The Feistel structure of Blowfish

The adjacent diagram shows Blowfish's encryption routine. Each line represents 32 bits. There are five subkey-arrays: one 18-entry P-array (denoted as K in the diagram, to avoid confusion with the Plaintext) and four 256-entry S-boxes (S0, S1, S2 and S3).

Every round are consists of 4 actions:

| | |
|---|---|
| Action 1 | XOR the left half (L) of the data with the r th P-array entry |
| Action 2 | Use the XORed data as input for Blowfish's F-function |
| Action 3 | XOR the F-function's output with the right half (R) of the data |
| Action 4 | Swap L and R |

The F-function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. The outputs are added modulo 232 and XORed to produce the final 32-bit output (see image in the upper right corner).[4] After the 16th round, undo the last swap, and XOR L with K18 and R with K17 (output whitening).

Decryption is exactly the same as encryption, except that P1, P2, ..., P18 are used in the reverse order. This is not so obvious because xor is commutative and associative. A common misconception is to use inverse order of encryption as decryption algorithm (i.e. first XORing P17 and P18 to the ciphertext block, then using the P-entries in reverse order).

Blowfish's key schedule starts by initializing the P-array and S-boxes with values derived from the hexadecimal digits of pi, which contain no obvious pattern (see nothing up my sleeve number).

The secret key is then, byte by byte, cycling the key if necessary, XORed with all the P-entries in order. A 64-bit all-zero block is then encrypted with the algorithm as it stands. The resultant ciphertext replaces P1 and P2. The same ciphertext is then encrypted again with the new subkeys, and the new ciphertext replaces P3 and P4. This continues, replacing the entire P-array and all the S-box entries. In all, the Blowfish encryption algorithm will run 521 times to generate all the subkeys - about 4KB of data is processed.

## II. PROPOSED METHODOLOGY

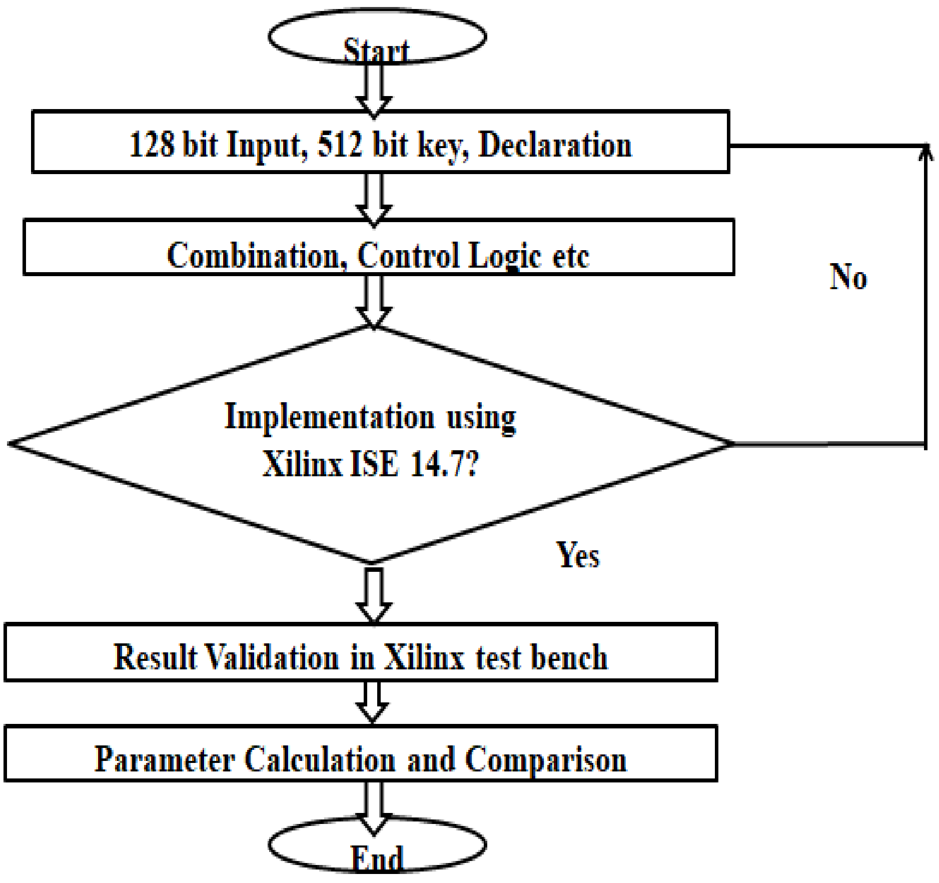

Start
128 bit Input, 512 bit key, Declaration
Combination, Control Logic etc
No
Implementation using Xilinx ISE 14.7?
Yes
Result Validation in Xilinx test bench
Parameter Calculation and Comparison
End

Figure 3: Flow Chart

**Encryption:** Generation of subkeys:

18 subkeys{P[0]…P[17]} are needed in both encryption aswell as decryption process and the same subkeys are used for both the processes.

These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.

It is initialised with the digits of pi(?).

The hexadecimal representation of each of the subkeys is given by:

The resultant P-array holds 18 subkeys that is used during the entire encryption process

**Decryption:**

The Decryption function also consists of two parts:

Rounds: The decryption also consists of 16 rounds with each round(Ri)(as explained above) taking inputs the cipherText(C.T.) from previous round and corresponding subkey(P[17-i])(i.e for decryption the subkeys are used in reverse).

## III. RESULT AND ANALYSIS

The implementation and simulation of the proposed algorithm is done over Xilinx 14.7. The behavioral modeling style and Isim simulator is adopted for simulation. RTL and synthesis results are also generated.

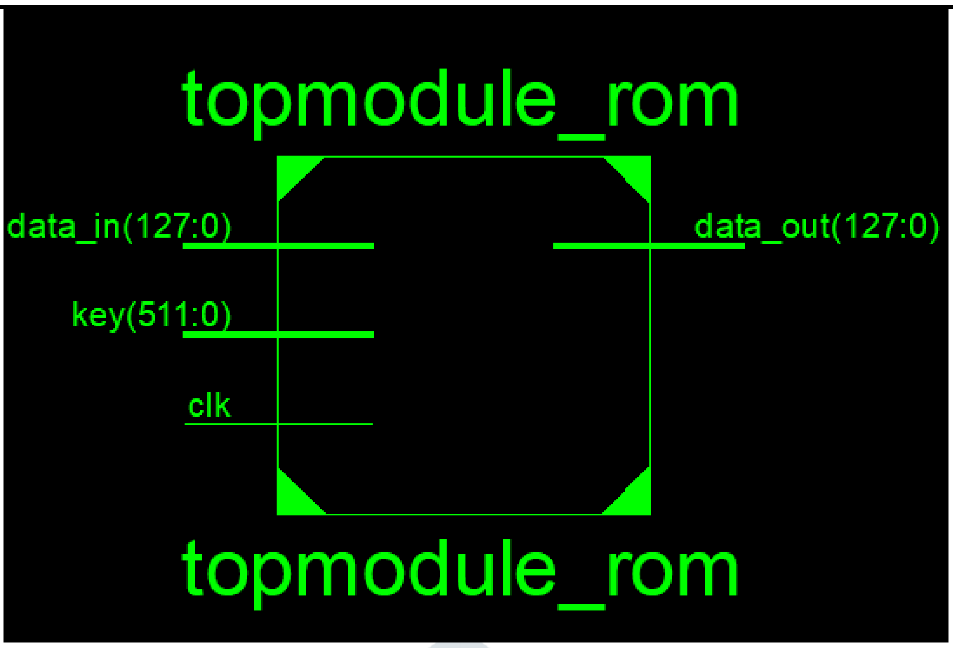
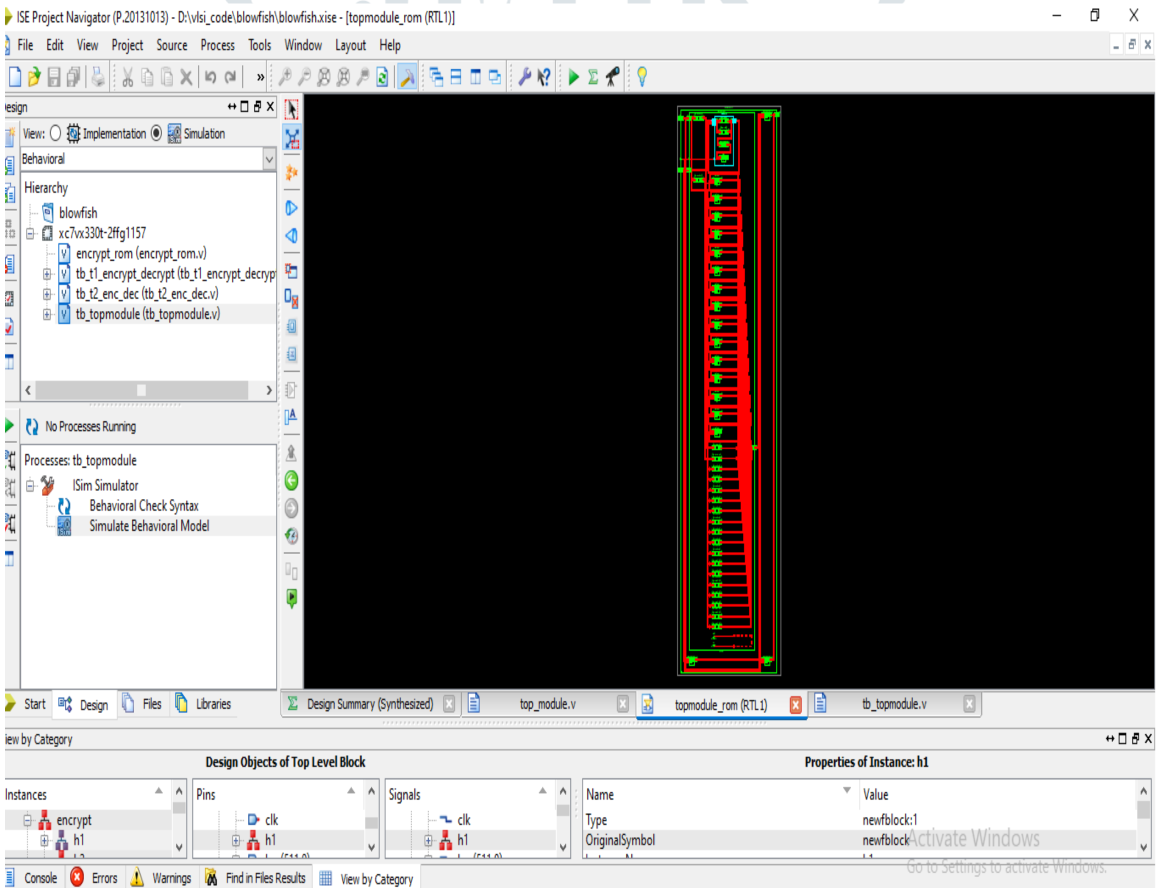


Figure 2: Top module of proposed 128 bit blowfish

Figure 2 is showing the top module of the proposed 128 bit blowfish algorithm. The input bit is 0:127 means 128 bit, key size is 0:511 means 512 and one clock pulse. The encryption and decryption process done and the 128 bit output are generated.


ISE Project Navigator (P.20131013) - D:\vlsi_code\blowfish\blowfish.xise - [topmodule_rom (RTL1)]
File Edit View Project Source Process Tools Window Layout Help
View: Implementation Simulation
Behavioral
Hierarchy
blowfish
xc7vx330t-2ffg1157
encrypt_rom (encrypt_rom.v)
tb_t1_encrypt_decrypt (tb_t1_encrypt_decryp
tb_t2_enc_dec (tb_t2_enc_dec.v)
tb_topmodule (tb_topmodule.v)
No Processes Running
Processes: tb_topmodule
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
Design Summary (Synthesized)
top_module.v
topmodule_rom (RTL1)
tb_topmodule.v
View by Category
Design Objects of Top Level Block
Properties of Instance: h1
Instances
encrypt
h1
Pins
clk
h1
Signals
clk
h1
Console Errors Warnings Find in Files Results View by Category
Activate Windows
Go to Settings to activate Windows.

| Name | Value |
|---|---|
| Type | newfblock:1 |
| OriginalSymbol | newfblock |

Figure 3: Complete RTL view

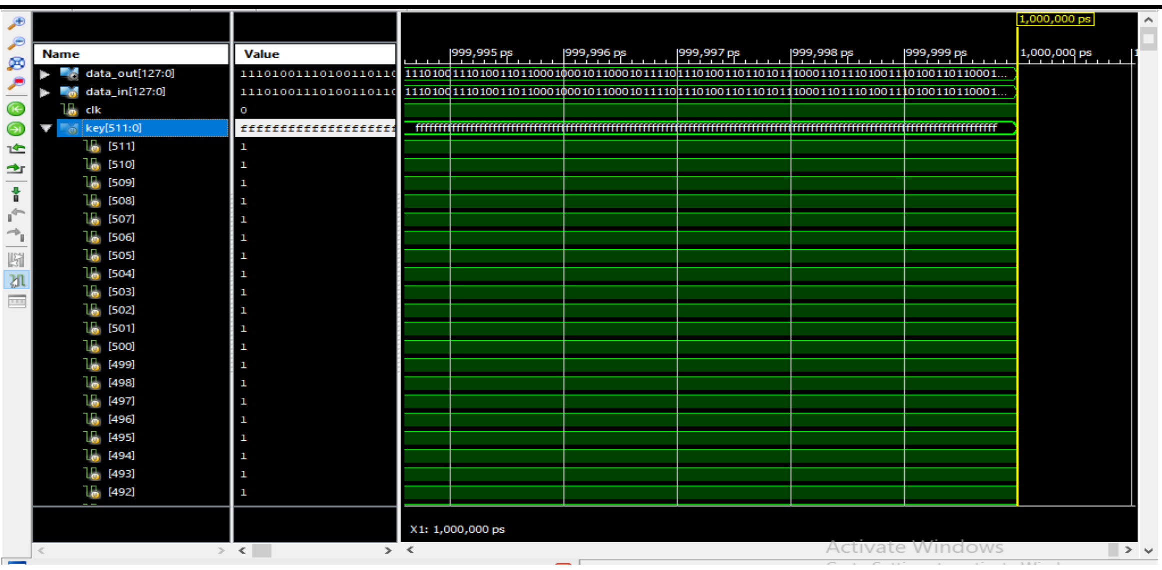


Figure 4: Results in test bench-4

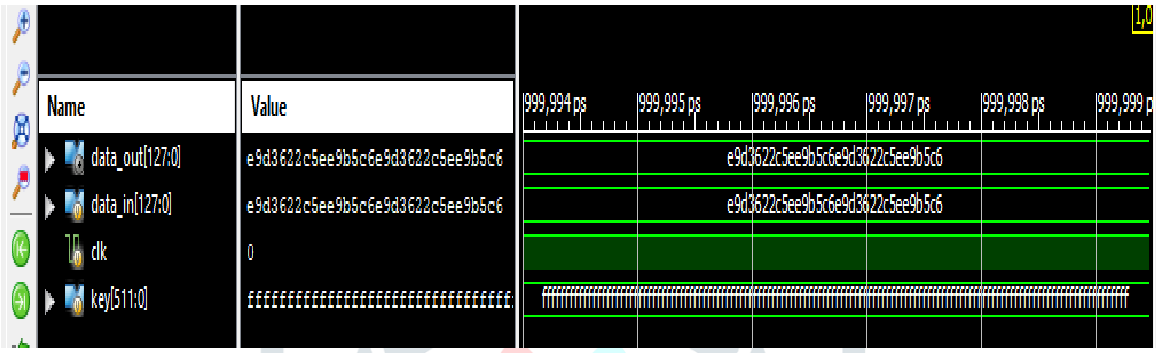


Figure 5: Results in test bench-5

Table 1: Simulation Parameters

| Sr No | Parameter | Value |
|---|---|---|
| 1 | Area | 12% |
| 2 | Power | 0.45 mW |
| 3 | Latency | 2.993 |

Table 2: Result comparison

| Sr No. | Parameters | Existing work result | Proposed work result |
|---|---|---|---|
| 1 | Method | 64 bit | 128 bit |
| 2 | Area | 12.5% | 12% |
| 3 | Power | 0.46 mW | 0.40 mW |
| 4 | Latency | 8 ns | 2.993 ns |

Therefore proposed work result is better than previous work so 128 bit blowfish approach is considerable and significant results is achieved.

## IV. CONCLUSION

This paper proposed the scheme to extend the Blowfish block cipher security. In proposed scheme, 128 bit Blowfish is implemented. So total no of rounds are altered by skipping few Blowfish rounds using round key. As a result, proposed scheme increase additional Blowfish cipher security against attack apart from minimum to maximum size of Blowfish key. In addition to that the proposed scheme also decreases encryption and decryption execution time of Blowfish cipher.

## REFERENCES

1. B. M. B. Beron, V. T. Duhaylungsod, K. G. Jimenez, J. Hora, R. C. O. Calimpusan and O. Joy Gerasta, "ASIC Implementation of Pipelined Blowfish Cryptographic Core in 0.13 µm CMOS Process Technology," 2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management, Laoag, Philippines, 2019, pp. 1-6, doi: 10.1109/HNICEM48295.2019.9073385.
2. H. Setiawan and K. Rey Citra, "Design of Secure Electronic Disposition Applications by Applying Blowfish, SHA-512, and RSA Digital Signature Algorithms to Government Institution," 2018 International Seminar on Research of Information

Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2018, pp. 168-173, doi: 10.1109/ISRITI.2018.8864280.

3. M. A. Muin, M. A. Muin, A. Setyanto, Sudarmawan and K. I. Santoso, "Performance Comparison Between AES256-Blowfish and Blowfish-AES256 Combinations," 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, 2018, pp. 137-141, doi: 10.1109/ICITACEE.2018.8576929.
4. S. Vyakaranal and S. Kengond, "Performance Analysis of Symmetric Key Cryptographic Algorithms," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, 2018, pp. 0411-0415, doi: 10.1109/ICCSP.2018.8524373.
5. S. Varshney, T. Sudarshan and S. Khare, "Efficient Hardware Architecture for Amalgam of Blowfish and Rc6," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, 2017, pp. 1126-1130, doi: 10.1109/CTCEEC.2017.8455189.
6. I. A. Landge and B. K. Mishra, "VHDL based BLOWFISH implementation for secured Embedded System design," 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, 2017, pp. 497-501, doi: 10.1109/AEEICB.2017.7972363.
7. T. K. Hazra, A. Mahato, A. Mandal and A. K. Chakraborty, "A hybrid cryptosystem of image and text files using blowfish and Diffie-Hellman techniques," 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Bangkok, 2017, pp. 137-141, doi: 10.1109/IEMECON.2017.8079577.
8. A. Chauhan and J. Gupta, "A novel technique of cloud security based on hybrid encryption by Blowfish and MD5," 2017 4th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, 2017, pp. 349-355, doi: 10.1109/ISPCC.2017.8269702.
9. A. Gaur, A. Jain and A. Verma, "Analyzing storage and time delay by hybrid Blowfish-Md5 technique," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, 2017, pp. 2985-2990, doi: 10.1109/ICECDS.2017.8390003.
10. R. Ahmad[1], A. A. Manaf and W. Ismail, "Development of an improved power-throughput Blowfish algorithm on FPGA," 2016 IEEE 12th International Colloquium on Signal Processing & Its Applications (CSPA), Malacca City, 2016, pp. 237-241, doi: 10.1109/CSPA.2016.7515838.
11. V. C. Dongre and S. G. Shikalpure, "Ensuring privacy preservation in wireless networks against traffic analysis by employing network coding and Blowfish encryption," 2016 International Conference on Signal and Information Processing (IConSIP), Vishnupuri, 2016, pp. 1-5, doi: 10.1109/ICONSIP.2016.7857442.
12. S. S. Kondawar and D. H. Gawali, "Blowfish algorithm for patient health monitoring," 2016 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, 2016, pp. 1-6, doi: 10.1109/INVENTIVE.2016.7830230.
13. N. Jayapandian, A. M. J. M. Zubair Rahman, R. B. Sangavee and R. Divya, "Improved cloud security trust on client side data encryption using HASBE and Blowfish," 2016 Online International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016, pp. 1-6, doi: 10.1109/GET.2016.7916767.
14. P. V. Maitri and A. Verma, "Secure file storage in cloud computing using hybrid cryptography algorithm," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2016, pp. 1635-1638, doi: 10.1109/WiSPNET.2016.7566416.
15. V. P. Bansal and S. Singh, "A hybrid data encryption technique using RSA and Blowfish for cloud computing on FPGAs," 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), Chandigarh, 2015, pp. 1-5, doi: 10.1109/RAECS.2015.7453367.