International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 1, January- 2013

FPGA Implementation of Blowfish Cryptosystem Using VHDL

L. Kranthi Kiran
Student

J. E. N. Abhilash
Assoc. Professor

P. Suresh Kumar
Assistant Professor

Swarnandhra College of Engineering & Technology[1,2,3]

Abstract — *This paper gives a clear idea regarding VHDL implementation of Blowfish algorithm. The Blowfish cryptosystem is a very fast and useful scheme, even though it was introduced over a decade ago. Blowfish Cryptosystem hardware implementation is presented in this paper. In present scenario we are forced to find more and more secured encryption technique. Blowfish algorithm has no known cryptanalysis. If this technique is available in hardware, such a system may be the most powerful tool for any communication system where high security is needed. This cryptosystem is designed and is implemented using VHDL. Here is the approach for high speed embedded applications such as mobile phone networks. Wireless communication schemes greatly need highly secured encryption technique. In many of such applications it is difficult to use software cryptosystems. Hardware implementation of such a system is need for most of the wireless applications. Here is the approach with less hardware requirement and of high speed. This approach is of high speed and of less area. By implementing same algorithm in pipelining method speed can be drastically increased with a cost of area and power consumption.*

***Index Terms*— VHDL, Cryptosystem, Blowfish Algorithm, Cryptanalysis, Pipelining**

I. INTRODUCTION

Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message shared the decoding technique needed to recover the original information only with intended recipients, thereby precluding unwanted persons to do the same. Since World War the methods used to carry out cryptology have become increasingly complex and its application more widespread.

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system but it is infeasible to do so by any known practical means. These schemes are therefore termed computationally secure; theoretical advances (e.g., improvements in integer factorization algorithms) and faster computing technology require these solutions to be continually adapted. There exist information-theoretically secure schemes that provably cannot be broken even with unlimited computing power—an example is the one-time pad—but these schemes are more difficult to implement than the best theoretically breakable but computationally secure mechanisms.

Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key (or, less commonly, in which their keys are different, but related in an easily computable way). This was the only kind of encryption publicly known until June 1976[1]. Symmetric key ciphers are implemented as either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plaintext as opposed to individual characters, the input form used by a stream cipher.

The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block cipher designs which have been designated cryptography standards by the US government (though DES's designation was finally withdrawn after the AES was adopted).[2] Despite its deprecation as an official standard, DES (especially its still-approved and much more secure triple-DES variant) remains quite popular; it is used across a wide range of applications, from ATM encryption[3] to e-mail privacy[4] and secure remote access.[5] Many other block ciphers have been designed and released, with considerable variation in quality. Many have been thoroughly broken, such as FEAL.[7][6]

Stream ciphers, in contrast to the 'block' type, create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad. In a stream cipher, the output stream is created based on a hidden internal state which changes as the cipher operates. That internal state is initially set up using the secret key material. RC4 is a widely used stream cipher; see Category:Stream ciphers.[7] Block ciphers can be used as stream ciphers; see Block cipher modes of operation.

Symmetric-key cryptosystems use the same key for encryption and decryption of a message, though a message or group of messages may have a different key than others. A significant disadvantage of symmetric ciphers is the key management necessary to use them securely. Each distinct pair of communicating parties must, ideally, share a different key, and perhaps each ciphertext exchanged as well. The number of keys required increases as the square of the number of network members, which very quickly requires complex key management schemes to keep them all straight and secret. The difficulty of securely establishing a secret key between two communicating parties, when a secure channel does not already exist between them, also presents a chicken-and-egg problem which is a considerable practical obstacle for

cryptography users in the real world.

II. Data Encryption Standard

DES standard is originated go back to the early 1970s. In 1972, after concluding a study on the US government's computer security needs, the US standards body NBS (National Bureau of Standards) — now named NIST (National Institute of Standards and Technology) — identified a need for a government-wide standard for encrypting unclassified, sensitive information.[8] After consulting with the NSA, NBS solicited proposals for a cipher that would meet rigorous design criteria. None of the submissions, however, turned out to be upto the mark. A second request was issued on 27 August 1974. This time, IBM submitted a candidate which was deemed acceptable — a cipher developed during the period 1973–1974 based on an earlier algorithm, Horst Feistel's Lucifer cipher. The team at IBM involved in cipher design and analysis included Feistel, Walter Tuchman, Don Coppersmith, Alan Konheim, Carl Meyer, Mike Matyas, Roy Adler, Edna Grossman, Bill Notz, Lynn Smith, and Bryant Tuckerman.

Despite the criticisms, DES was approved as a federal standard in November 1976, and published on 15 January 1977 as FIPS PUB 46, authorized for use on all unclassified data. It was subsequently reaffirmed as the standard in 1983, 1988 (revised as FIPS-46-1), 1993 (FIPS-46-2), and again in 1999 (FIPS-46-3), the latter prescribing "Triple DES" (see below). On 26 May 2002, DES was finally superseded by the Advanced Encryption Standard (AES), following a public competition. On 19 May 2005, FIPS 46-3 was officially withdrawn, but NIST has approved Triple DES through the year 2030 for sensitive government information.[9]

The algorithm is also specified in ANSI X3.92,[10] NIST SP 800-67[9] and ISO/IEC 18033-3[11] (as a component of TDEA).

Another theoretical attack, linear cryptanalysis, was published in 1994, but it was a brute force attack in 1998 that demonstrated that DES could be attacked very practically, and highlighted the need for a replacement algorithm. These and other methods of cryptanalysis are discussed in more detail later in this article.

The introduction of DES is considered to have been a catalyst for the academic study of cryptography, particularly of methods to crack block ciphers. According to a NIST retrospective about DES,

The DES can be said to have "jump started" the nonmilitary study and development of encryption algorithms. In the 1970s there were very few cryptographers, except for those in military or intelligence organizations, and little academic study of cryptography. There are now many active academic cryptologists, mathematics departments with strong programs in cryptography, and commercial information security companies and consultants. A generation of cryptanalysts has cut its teeth analyzing (that is trying to "crack") the DES algorithm. In the words of cryptographer Bruce Schneier,[12] "DES did more to galvanize the field of cryptanalysis than anything else. Now there was an algorithm to study." An astonishing share of the open literature in cryptography in the 1970s and 1980s dealt with the DES, and the DES is the standard against which every symmetric key algorithm since has been compared.[13]

III. Blowfish Algorithm

Blowfish encrypts 64-bit blocks of plaintext into 64-bit blocks of ciphertext. Blowfish is implemented in numerous products and has received a fair amount of scrutiny. So far, the security of Blowfish is unchallenged. Blowfish is a keyed, symmetric block cipher, designed in 1993 by Bruce Schneier and included in a large number of 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 now receives more attention.

Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the ageing 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/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." Notable features of the design include key-dependent S-boxes and a highly complex key schedule.

Blowfish has a 64-bit block size and a variable key length from 32 bits up to 448 bits.[2] 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 diagram to the left shows the action of Blowfish. Each line represents 32 bits. The algorithm keeps two subkey arrays: the 18-entry P-array and four 256-entry S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. One entry of the P-array is used every round, and after the final round, each half of the data block is XORed with one of the two remaining unused P-entries.

The diagram to the upper right shows Blowfish's F-function. The function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 232 and XORed to produce the final 32-bit output.

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.

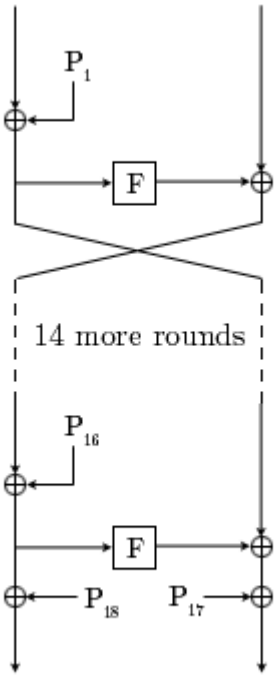

Fig 1: Structure of Blowfish algorithm (Encryption)

Because the P-array is 576 bits long, and the key bytes are XORed through all these 576 bits during the initialization, many implementations support key sizes up to 576 bits. While this is certainly possible, the 448 bits limit is here to ensure that every bit of every subkey depends on every bit of the key,[2] as the last four values of the P-array don't affect every bit of the ciphertext. This point should be taken in consideration for implementations with a different number of rounds, as even though it increases security against an exhaustive attack, it weakens the security guaranteed by the algorithm. And given the slow initialization of the cipher with each change of key, it is granted a natural protection against brute-force attacks, which doesn't really justify key sizes longer than 448 bits.

There is no effective cryptanalysis on the full-round version of Blowfish known to the public as of 2011. A sign extension bug in one publication of C code has been identified.[3]

In 1996, Serge Vaudenay found a known-plaintext attack requiring 28r + 1 known plaintexts to break, where r is the number of rounds. Moreover, he also found a class of weak keys that can be detected and broken by the same attack with only 24r + 1 known plaintexts. This attack cannot be used against the regular Blowfish; it assumes knowledge of the key-dependent S-boxes. Vincent Rijmen, in his Ph.D. thesis, introduced a second-order differential attack that can break four rounds and no more.[1] There remains no known way to break the full 16 rounds, apart from a brute-force search.[4]

Bruce Schneier notes that while Blowfish is still in use, he recommends using the more recent Twofish algorithm instead.[5]

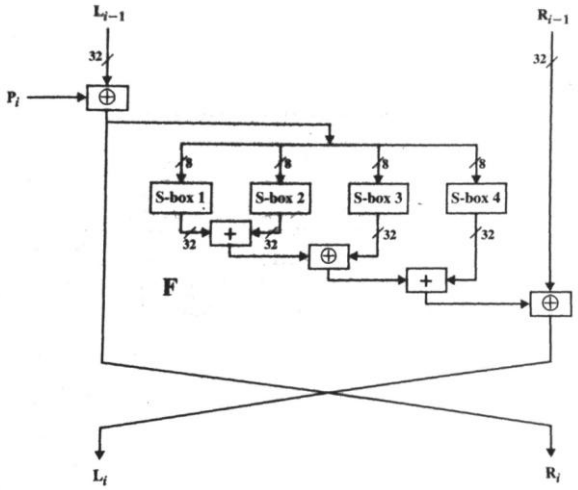

Fig 2: Implementation of F function

Blowfish is a fast block cipher, except when changing keys. Each new key requires pre-processing equivalent to encrypting about 4 kilobytes of text, which is very slow compared to other block ciphers. This prevents its use in certain applications, but is not a problem in others. In one application, it is actually a benefit: the password-hashing method used in Open BSD uses an algorithm derived from Blowfish that makes use of the slow key schedule; the idea is that the extra computational effort required gives protection against dictionary attacks. See key stretching.

Blowfish has a memory footprint of just over 4 kilobytes of RAM. This constraint is not a problem even for older desktop and laptop computers, though it does prevent use in the smallest embedded systems such as early smartcards.

Blowfish was one of the first secure block ciphers not subject to any patents and therefore freely available for anyone to use. This benefit has contributed to its popularity in cryptographic software.

Keys are stored in K array and the sub keys are stored in P array. There are four S-boxes. Each S-box has 256 entries each of width 32 bits. Here the Hexadecimal representation of π is used for P array and then continued with S-boxes. P array is XORed with K array for updating the P array. 64 bit plain text is encrypted now with P array and also using four S-boxes. Outputs are again encrypted using updated P array and S-boxes. After 521 iterations of the Blowfish encryption algorithm final P-array and S-array are generated. After these many iterations the encrypted data is obtained. That's why this encryption standard is of high secure and there is no software cryptanalysis up to now for this encryption standard. Decryption algorithm also same structure but the P array must be supplied in reverse order. Hardware implementation of such a high secured algorithm is well suited for high speed applications.

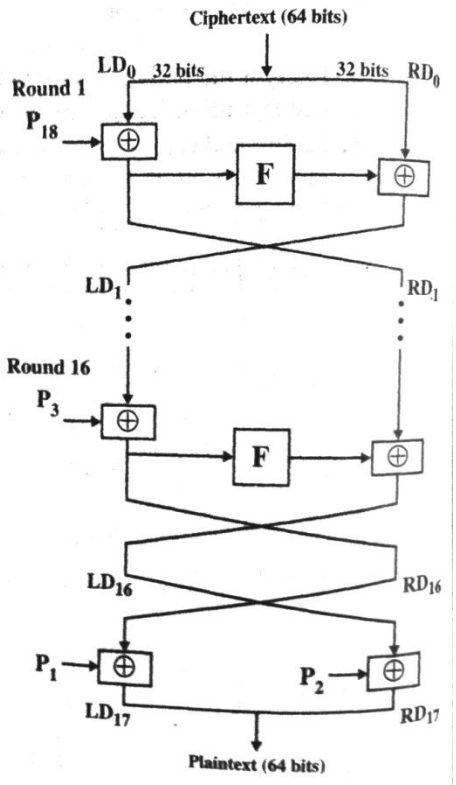

Fig 3: Structure of Blowfish algorithm (Decryption)

IV. Simulation Results

High secured Blowfish algorithm is implemented using VHDL language and is verified functionally using Xilinx ISE 9.1. The simulated results are presented in this chapter. Xilinx ISE simulation results shows that the encryption scheme is a complex encryption scheme and is highly reliable.

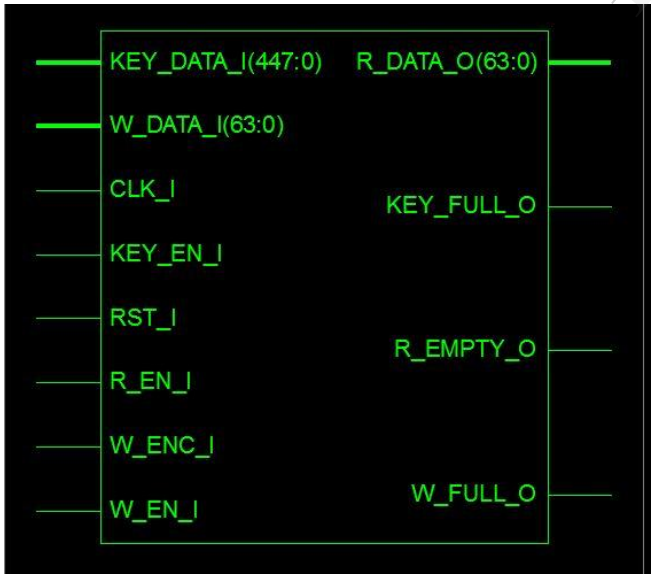

Fig 4: Blowfish Encryption module Black box

In order to implement this algorithm first of all 1MB Read only memory is designed with the Hexadecimal values of π. A dual port Read Ram is designed for storing P array. Same is used to store K array also. FIFO is used to read the plain text and store it temporary.

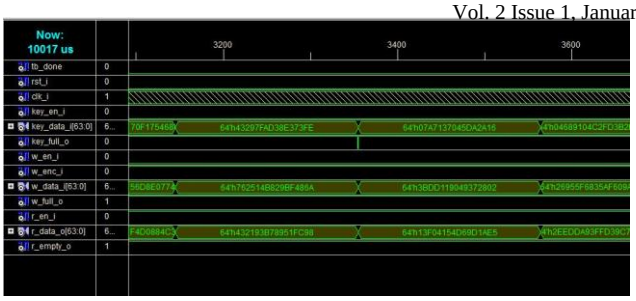

Fig 5: Simulation results showing Encryption Scheme

Figure 5 shows the simulation results for the Blowfish algorithm. Here in this figure plain text is supplied through W_Data_I input terminal. This plain text is a 64 bit plain thext and is encrypted using 32 bit keys. Figure 5 also shows the encrypted obtained through R_Data_O output signal at 7000ns. Again the encrypted data is supplied to the decryption module after 7000ns and the decrypted plain text is obtained through R_Data_O output signal as shown in figure 6.

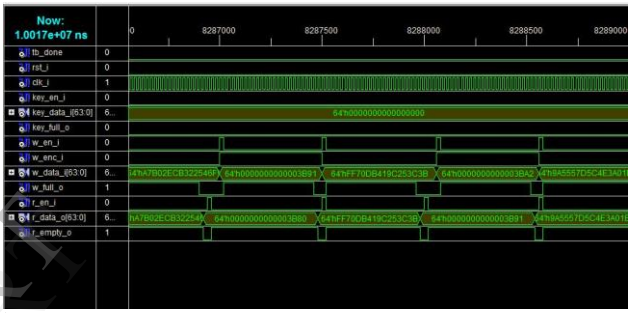

Fig 6: Simulation results showing Decryption Scheme

V. Conclusion

Blowfish Encryption scheme and Decryption is implemented using VHDL language. This is a high secured algorithm. Software cryptanalysis of such a complex algorithm is highly impossible. Hardware implementation of such a high secured algorithm is highly necessary for high speed applications. Because of the hardware implementation of the algorithm there is no need of any external platform to run the algorithm, and also the plain text is encrypted in negligible amount of time. Simulation results shows that this algorithm it can encrypt a plain text in less than 7000 ns of time. In future we can reduce this figure enormously by implementing this algorithm using pipelining technique.

VI. REFERENCES

[1] Whitfield Diffie and Martin Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644–654
[2] FIPS PUB 197: The official Advanced Encryption Standard.
[3] NCUA letter to credit unions, July 2004
[4] RFC 2440 - Open PGP Message Format
[5] SSH at windowsecurity.com by Pawel Golen, July 2004
[6] Bruce Schneier, Applied Cryptography, 2nd edition, Wiley, 1996, ISBN 0-471-11709-9.
[7] AJ Menezes, PC van Oorschot, and SA Vanstone, Handbook of Applied Cryptography ISBN 0-8493-8523-7.

[8] Walter Tuchman (1997). "A brief history of the data encryption standard". Internet besieged: countering cyberspace scofflaws. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA. pp. 275–280.
[9] National Institute of Standards and Technology, NIST Special Publication 800-67 Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Version 1.1
[10] American National Standards Institute, ANSI X3.92-1981 American National Standard, Data Encryption Algorithm
[11] "ISO/IEC 18033-3:2010 Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers". Iso.org. 2010-12-14.
[12] Bruce Schneier, Applied Cryptography, Protocols, Algorithms, and Source Code in C, Second edition, John Wiley and Sons, New York (1996) p. 267
[13] William E. Burr, "Data Encryption Standard", in NIST's anthology "A Century of Excellence in Measurements, Standards, and Technology: A Chronicle of Selected NBS/NIST Publications, 1901–2000