ISSN 2250-3153

Design of Efficient XTEA Using Verilog

Shweta Gaba, Iti Aggarwal, Dr. Sujata Pandey

Electronics and Communication Engineering, Amity School of Engineering and Technology,
Amity University, India

Abstract- In this age of viruses and hackers of electronic eavesdropping and electronic fraud, security is paramount. A cryptographic system (or a cipher system) is a method of hiding data so that only certain people can view it. A cryptographic system typically consists of algorithms, keys, and key management facilities. There are several algorithms to choose from that vary in the security they provide, their size, the time it takes to encrypt or decrypt a block of data. In this paper, we analyze and evaluate the development of a cheap and relatively fast hardware implementation of the extended tiny encryption algorithm (XTEA). Originally the research was split into separate encipher/decipher units, but these have now been combined into a single unit. The design will start by using finite state machine (FSMs) and will use Verilog hardware description language to describe the design. Minimizing the chip area and security transmission of data will be our main goal. The targeted hardware systems are the reconfigurable Spartan III and Xilinx Virtex IV modern field programmable gate arrays (FPGAs).

Index Terms- Encipher, Decipher, Finite State Machines, cryptographic system.

I. INTRODUCTION

The security of symmetric cryptosystem [9] is a function of two parameters: the strength of the algorithm and the length of the key. The algorithm must be so secure that there is no better way to break it than with a brute-force attack. The security of the algorithm must be resided in the key and delta value we choose. So therefore, there is balance between choosing long key, delta and the time required to complete the enciphering operation. Many encryption algorithms [8] are available in the market and the selection of a specific one is dependent on the relatively tight constraints. The selected algorithm should be small, relatively secure, with a proven history of overcoming possible well known attacks on it. The Tiny Encryption Algorithm (TEA) (Wheeler and Needham 1994), [6] and hence its successor the Extended-TEAs (XTEAs) (Needham and Wheeler 1997; Russell 2004; Kelsey et al. 1997; Moon et al. 2002) [7] are among the best choices available for security purpose. The name of block cipher came from the fact that block cipher encrypts plaintext as blocks [8]. These blocks differ in size between block cipher algorithms, for example, in Data Encryption Standard DES the plaintext is divided into blocks of length 64, but it is 32 in International Data Encryption Algorithm (IDEA) [11], if the length of block cipher equal one them, it will become stream cipher.

This paper uses the Verilog description language to implement the core function of XTEA and integrate them into a FPGA [10] chip. XTEA consist of two parts namely encipher and decipher units but we are developing in a single module. The unit accepts data in data_in1 and data_in2, a key in key, a delta in delta and the mode in block (‘00’ for encipher, ‘11’ for decipher). The all_done wire is raised when the results of the operation are ready to be read from out_1 and out_2. It needs to be reset before each use. The paper is arranged as follows:

1) Abstract
2) Introduction
3) Verilog designing of XTEA
4) Results of simulation and synthesis
5) Conclusions

II. IDENTIFY, RESEARCH AND COLLECT IDEA

After attending the conference on VLSI MEMS and NEMS held in Amity University we got an idea to do some research work on XTEA. We have studied some research papers and Google it thoroughly. Till now a key of different 128 bit is required for both encipher and decipher but we have design it in such a way that we require same key for both encipher an decipher module.

III. EXPERIMENTAL DETAILS

XTEA [1] is a symmetric block cipher designed to correct weakness in TEA [6]. Like TEA, XTEA is a 64-bit block Feistel network with a 128-bit key and a suggested 64 rounds. Several differences from TEA are apparent, including a somewhat more complex key-schedule and a rearrangement of the shifts, XORs and additions (Hong et al. 2003; Ko et al.2004). Fig.1 shows the block diagram of an XTEA.

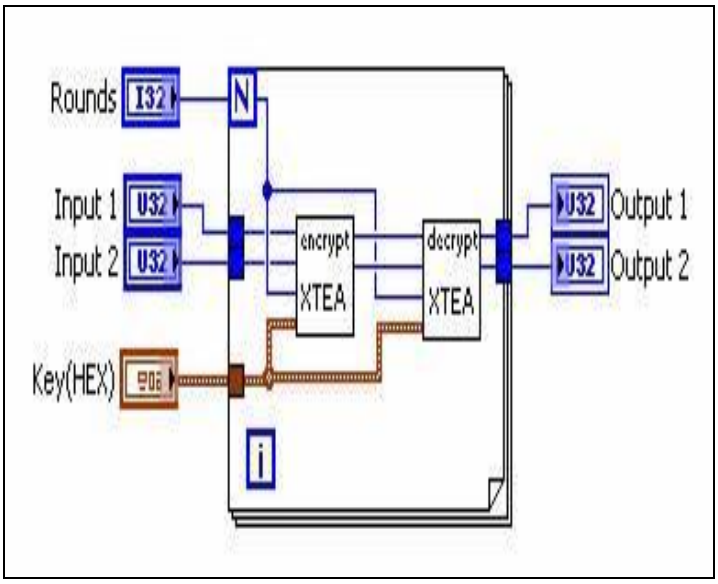


Fig.1 Block Diagram of XTEA

In this proposed XTEA we are using different value of DELTA and KEY to increase the security. By doing this only the sender and receiver knows the value of both delta and key. And we also use 32 cycles for encrypting and decrypting the data in Feistel function to increase the security. Fig.2 shows the block diagram of a single XTEA round.

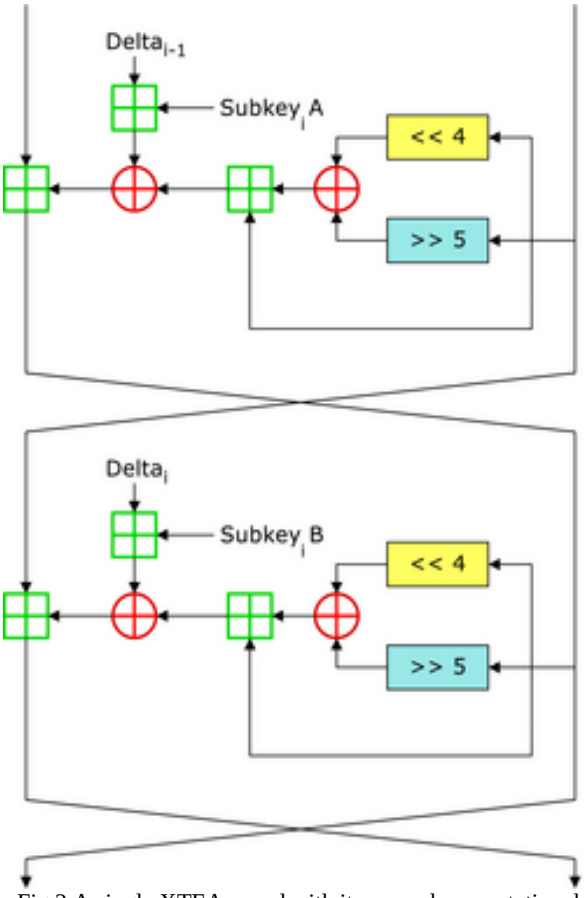
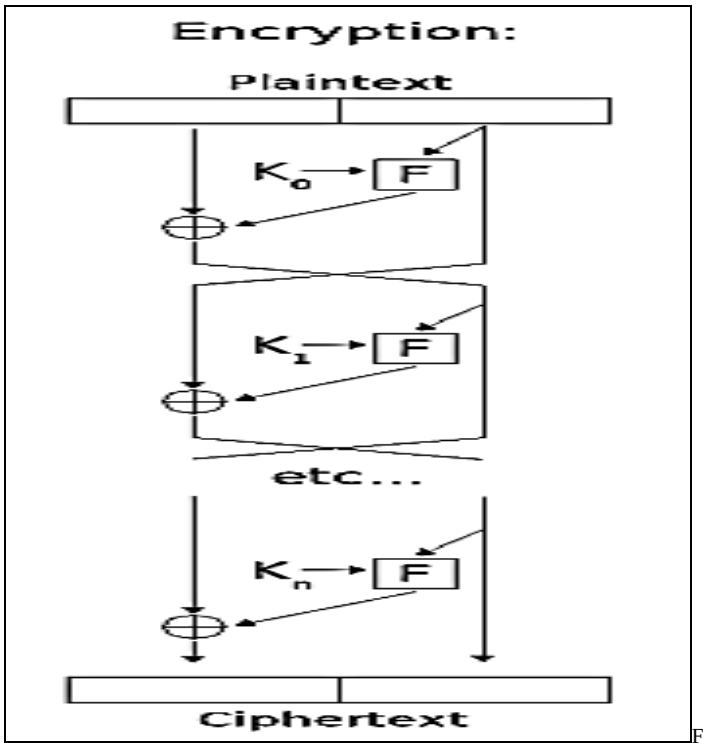


Fig.2 A single XTEA round with its normal computational constructs. The crossed square for the sum, crossed circle for an XOR, >> for a right shift, << for a left shift.

The two main components of the XTEA such as the encipher and decipher are described below:

A. Encipher Module

To use the encipher.v, the inputs are two 32-bit U32 data values, the number of iterations and the 128-bit key value. The key is represented as a four 32-bit hexadecimal values. Key values can be created using a 128-bit key generator. The output from the system is two 32-bit values that are the encrypted version of the input. The Feistel function is used for encrypting the data and which is shown below in Fig.3.



F

Fig.3 Flow diagram for data encryption using Feistel function

B. Decipher Module

To use the decipher.v the inputs are two 32-bit U32 data values, the number of iterations, and the 128-bit key value. The key is represented as four 32-bit hexadecimal values. The input data should be the encrypted data from the encipher.v and having the same 128-bit key value which is used in the encryption process. In this module an additional delta value i.e. also called magical value is included in the project and is used to implement the logic of the algorithms but in this programming we can use any hexadecimal value of 32-bit.The Feistel function is used to decrypt the data and shown below in Fig.4:

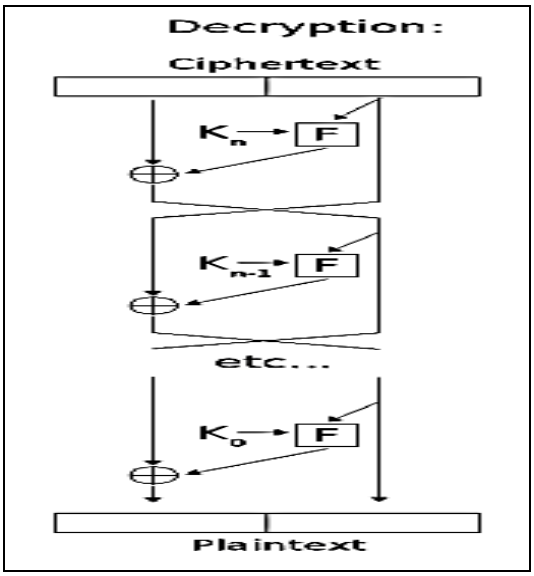


Fig.4 Flow diagram for data decryption using Fiestel function

IV. STATE DIAGRAM OF XTEA

The state diagram of XTEA shows that how our XTEA works and tells about the actual flow of data during encryption and decryption and shown below in Fig.5:

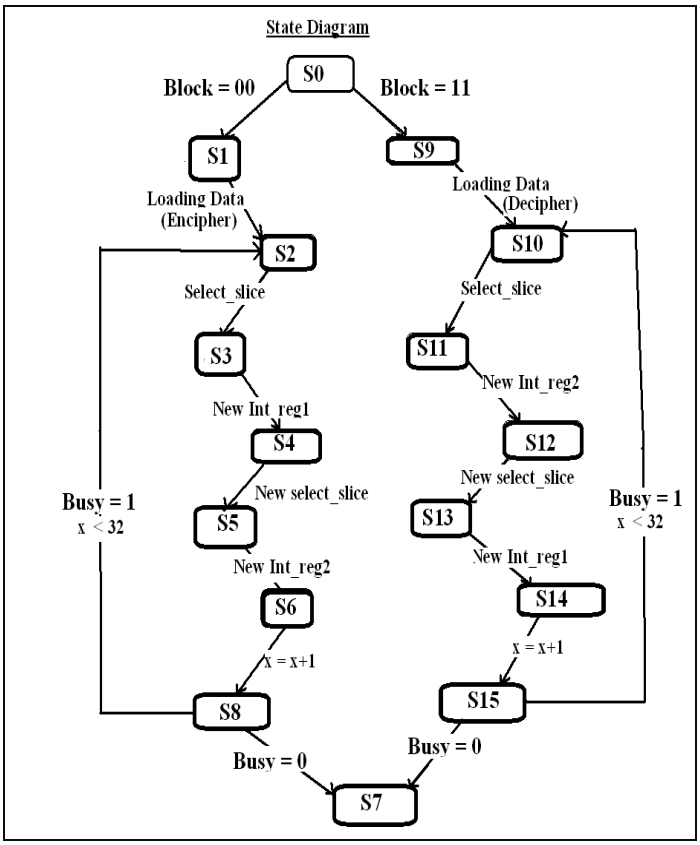


Fig.5 State diagram of XTEA

While coding for XTEA, we want our algorithm which utilize minimum chip area and provide more security transmission of data. So, we use Mealy Machine because mealy Machines use less number of states than Moore machine. In this state diagram, after resetting all the data at S0 state decides whether data has to be encrypted or decrypted. This is done by Block: If "Block=00" data goes for encryption i.e. state S1 and If "Block=11" data goes for decryption i.e. State S9. After engaging the Encrypt signal, the system will enter the state s1, and will finish the encryption after busy signal goes low (x>32) and we get data at state S7.

In the same manner, data which will be decrypted goes for decryption i.e. state S9.After engaging the Decrypt signal ,the system will enter the state S9,and will finish the decryption after busy signal goes low(x>32) and will get the decrypted data at S7. It must be noted that the encrypted data must be sent as input for decryption to get the same input data (as in encryption process) as output.

V. RESULTS AND DISCUSSIONS

The proposed XTEA has been simulated on the ModelSim SE 10.0a and has been synthesized on the Xilinx ISE 10.1. The figure 1 shows the generalized block diagram of XTEA with the maximum frequency of 129.099 MHz in case of Virtex4 and 71.114MHz in case of Spartan3.

In Fig.6, we show the waveforms of testing the encryption process. The reset signal was activated at first to insure that all the registers are cleared before starting any operation, note that the asynchronous reset is active high once. After that low the reset and select the block to set the module. Now, the module is ready to either encrypt or decrypt. One can distinguish between these two by the control signal i.e. block provided as an input to the system. After engaging the Encrypt signal, the system will enter the state s1, and will finish the encryption after busy signal goes Low (x>32). You can notice that the output is available at that time when x is equal to 32 which can easily shown in Fig.7.

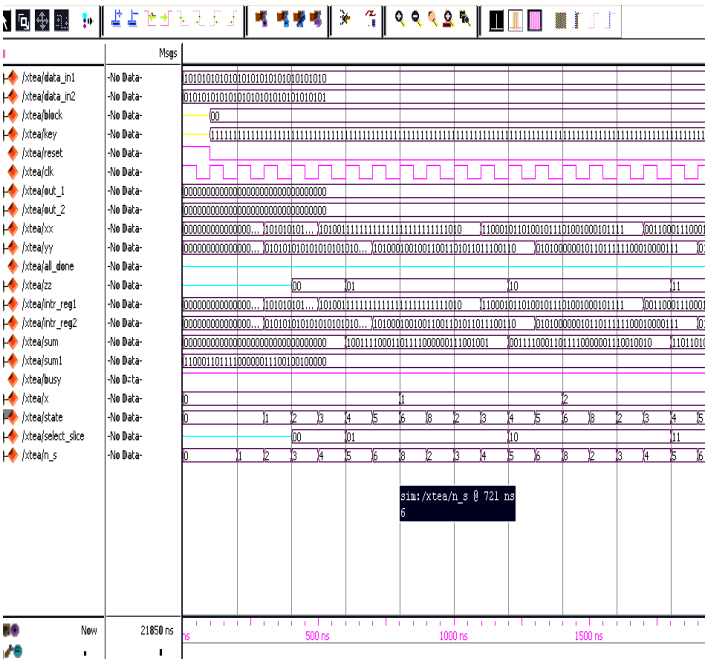


Fig.6 Testing the Encryption Process (shows the starting Portion of the waveform).

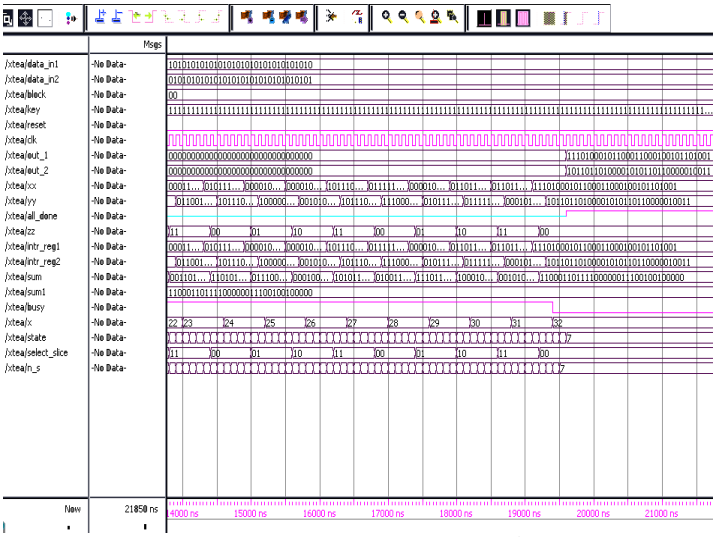


FIG.7 Testing the Encryption Process (shows the Output after x is equal to 32)

In Figure 8, we depict the waveforms of decryption state transition testing. The encryption and the Decryption processes are quite the same in architecture, but the decryption operates in a reverse manner on the data, and uses a subtract or rather than an adder. This test shows the transition of the state from the S0 to S9, because block is selected for decryption and here the input given is same as of output of encryption module, and will finish the decryption after busy signal goes high. You can notice that the output is available at that time when x is equal to 32 which can easily shown in figure 9.

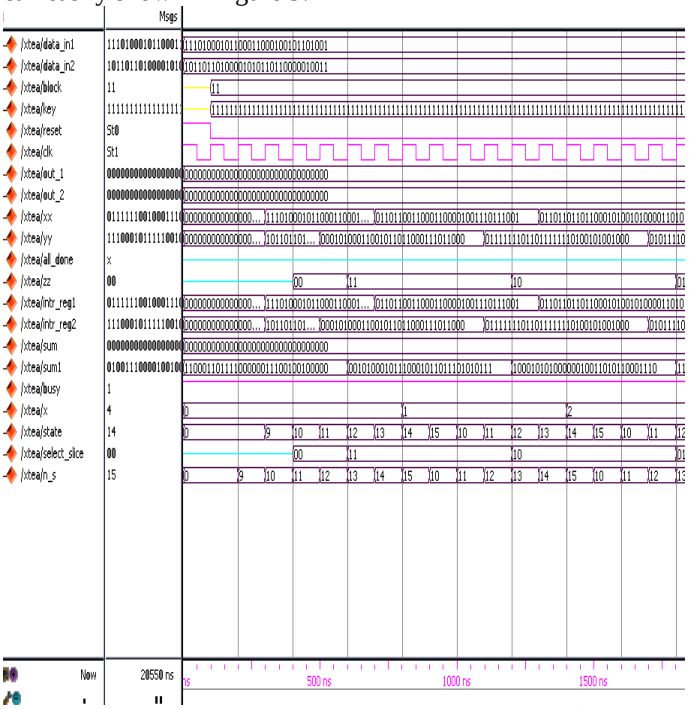


Fig. 8 Testing the Decryption Process (shows the starting Portion of the waveform).

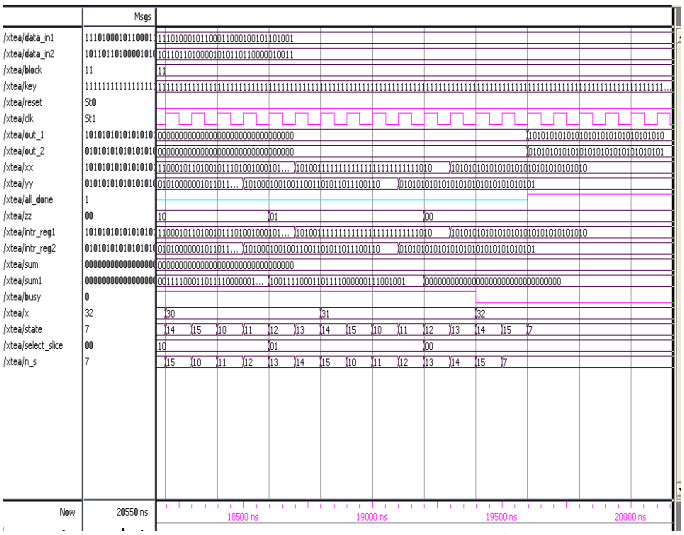


FIG.9 Testing the Decryption Process (shows the Output after x is equal to 32).

VI. RTL SCHEMATIC

The top module of the XTEA is shown below which we will get after synthesize our main module on Xilinx ISE 10.1.

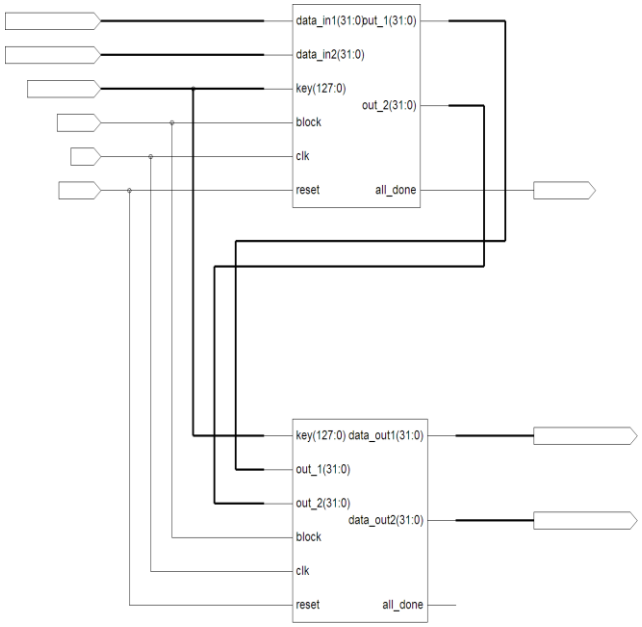


Fig.10 INTERNAL STRUCTURE OF XTEA WITH TWO BLOCKS - ENCIPHER AND DECIPHER

VII. SYNTHESIS RESULT

The synthesis result contains a table which shows the comparison between the old research and the proposed XTEA. While doing comparison we will find that our research shows great significance as our all the parameters are consuming less area and consumes less power to operate.

TABLE.1Comparison between old research and proposed XTEA

Parameters	Virtex4SX25 FF668	Virtex4SX25 FF668 [3]	Spartan3s200tq 144-4
LUTs	1252	4,608	1182

Clock Period(ns)	7.746	6.403	14.062
Slice flip flops	244	1,081	259
Maximum frequency	129.099MHz	156.177MHz	71.114MHz

VIII. CONCLUSIONS

This uses Verilog description language to get the modules of XTEA. After studying the comparative analysis with [3] we conclude that there is a difference in between the number of slices, LUTs, GCLKs and the maximum frequency. While synthesizing our optimization goal is speed. The results are quiet stable and reliable and has great flexibility with high integration.

REFERENCES

[1] Extended TEA Algorithm proposed by Tom St Denis, April 20th 1999.
[2] Julio C. Hernandez, Pedro Isasi "New results on the Genetic Cryptanalysis of TEA Reduced Round versions of XTEA" 2000 IEEE.
[3] Steven M. Aumack, Michael D. Koontz Jr. "Hardware Implementation of XTEA".
[4] Derek Williams CPSC 6128- Network Security Columbus State University "The Tiny Encryption Algorithm (TEA)" April 26, 2008.
[5] Gaidaa Saeed Mahdi, "A Modification of TEA Block Cipher Algorithm for Data Security (MTEA)", Eng. & Tech., Journal, vol.29, No.5, 2011.
[6] Wheeler, David J. and Needham, Roger M. TEA, "a Tiny Encryption Algorithm" Computer Laboratory, Cambridge University, England. November, 1994.
[7] Wheeler, David J. and Needham, Roger M. TEA Extensions. Computer Laboratory, Cambridge University, England. October, 1997.
[8] Scheier, Bruce." A Self-Study Course in Block-Cipher Cryptanalysis", Crypto logia, Vol.24 (1).January 2000.
[9] Feistel, Horst." Cryptography and Computer Privacy", Scientific American. Vol. 228(5). May 1973.
[10] Ke Wang," An Encrypt and Decrypt Algorithm Implementation on FPGAs", Semantics, Knowledge and Grid, 2009. SKG 2009. Fifth International Conference, Page(s): 298 – 301. B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.
[11] Biham, Eli and Shamir, Adi," Differential Cryptanalysis of the Data Encryption Standard", Springer Verlag, 1993. ISBN 0-387-97930-1, ISBN 3-540-97930-1.

AUTHORS

First Author – Shweta Gaba, Pursuing M.Tech in VLSI, Amity School of Engineering and Technology, Amity University and e-mail – gabashweta18@yahoo.co.in.

Second Author – Iti Aggarwal, Pursuing M.Tech in VLSI, Amity School of Engineering and Technology, Amity University and e-mail – agarwal.iti7@gmail.com

Third Author – Dr. Sujata Pandey, HOI of Electronics and Communication Dept. Amity School of Engineering and Technology, Amity University and e-mail – spandey@amity.edu.