International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 1, February 2016, pp. 298~306
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i1.8388

High Throughput FPGA Implementation of Data Encryption Standard with Time Variable Sub-Keys

Soufiane Oukili, Seddik Bri
Department of Electrical Engineering, High School of Technology, Moulay Ismail University, Morocco

Article Info

Article history:

Received Jun 18, 2015
Revised Nov 28, 2015
Accepted Dec 14, 2015

Keyword:

Data encryption standard
FPGA implementation
Pipelining
Security
Sub-keys
Time variable

ABSTRACT

The Data Encryption Standard (DES) was the first modern and the most popular symmetric key algorithm used for encryption and decryption of digital data. Even though it is nowadays not considered secure against a determined attacker, it is still used in legacy applications. This paper presents a secure and high-throughput Field Programming Gate Arrays (FPGA) implementation of the Data Encryption Standard algorithm. This is achieved by combining 16 pipelining concept with time variable sub-keys and compared with previous illustrated encryption algorithms. The sub-keys vary over time by changing the key schedule permutation choice 1. Therefore, every time the plaintexts are encrypted by different sub-keys. The proposed algorithm is implemented on Xilinx Spartan-3e (XC3s500e) FPGA. Our DES design achieved a data encryption rate of 10305.95 Mbps and 2625 number of occupied CLB slices. These results showed that the proposed implementation is one of the fastest hardware implementations with much greater security.

Corresponding Author:

Soufiane Oukili,
Department of Electrical Engineering,
High School of Technology, Moulay Ismail University, Morocco
B.P 3103, High School of Technology, Meknès, Morocco
Email: soufiane.oukili@gmail.com

1. INTRODUCTION

Cryptography is the science of using mathematics to transform intelligible information to unintelligible data. Cryptography enables to store sensitive information or transmit it across insecure networks, so that it cannot be read by anyone except the intended recipient. This can be done by two techniques, symmetric key and asymmetric key. Symmetric key cryptography involves the usage of the same key for encryption and decryption. On the other hand the asymmetric key involves the usage of one key for encryption and another, different key for decryption. Secret key cryptography includes DES, AES, 3DES, IDEA, Blowfish algorithms etc. and public key cryptography includes RSA, Digital Signature and Message Digest algorithms [1-2].

The Data Encryption Standard (DES) is an encryption standard for protecting confidential information. It has been developed in the 1970s at IBM and adopted as a Federal Information Processing Standard since 1977 by the National Institute of Standards and Technology [3-4]. DES has been used pervasively by many applications that require data confidentiality. However, from year 2001, DES has been superseded by the Advanced Encryption Standard AES [5]. But in practice, a lot of hardware or software applications still resort to DES.

The DES is a block cipher that operates on 64-bit blocks of data and uses 56-bit private effective key. Because of its small key size, several attacks against DES algorithm were published [6-8]. To increase the security of the algorithm, we proposed the Data Encryption Standard based on variable sub-keys with time.

The proposed scheme uses permutation choice 1(PC^{-1}) box in the key schedule, which contains several permutations to be selected by the sender. Whenever the permutation changes, the sub-keys change also, so there are different cipheretexts for the same key and plaintext. Therefore, the security is increased.

Implementation of DES is usually divided into software and hardware approaches. While the software method has security problems, the hardware encryption can be a better choice. FPGA implementation of DES encryption algorithm performs at much faster data-rates and provides better security than equivalent software implantations [9].

In this paper, we present an efficient and a secure hardware implementation of 16-stage pipelined DES, based on the variation of the key schedule permutation Choice 1 (PC^{-1}) with time. Data blocks can be loaded at each clock cycle and after an initial delay of 17 clock cycles, the cipheretexts will appear on consecutive clock cycles. The design is implemented on Xilinx Spartan FPGA technology. The FPGAs offer the advantage of hardware speed and software flexibility and programmability.

The rest of this paper is organized as follows: Section 2 describes the DES algorithm. Pipelining DES and pipelining DES based on time variable sub-keys are presented in Section 3 and Section 4. Section 5 gives implementation summary. Section 6 compares the achieved results with the previous DES implementations. Conclusion and references are given in Section 7 and 8 respectively.

2. DATA ENCRYPTION STANDARD ALGORITHM

DES algorithm encrypts 64-bit plaintext blocks with 64-bit key and generates 64-bit ciphertext blocks as shown in Figure 1. This algorithm uses complicated logical functions such as various types of permutations, XOR and shift functions. One bit in each 8 bits of the key may be utilized for error detection in key generation. Bits 8, 16, 24, 32, 40, 48, 56 and 64 are used in ensuring that each byte of the key is of odd parity and otherwise ignored. Consequently, the effective key length is 56 bits.

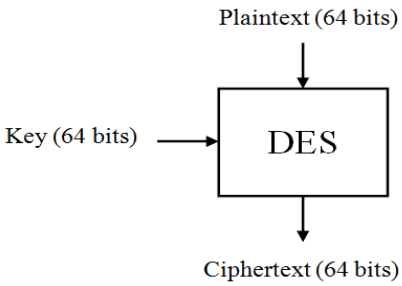

Figure 1. DES block view

DES is an iterative algorithm as shown in Figure 2. For each block of plaintext, encryption is handled in 16 rounds which all perform the identical operation. In every round a different sub-key is used and all sub-keys are derived from the main key.

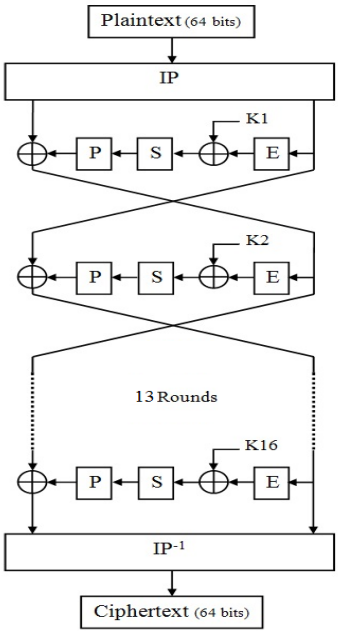

Figure 2. DES algorithm block diagram

The incoming block of plaintext (64 bits) firstly passes through an initial permutation (IP) and then will be divided into two 32-bit halves, 32 right bits and 32 left bits.

In every round, the right 32 bits are expanded to 48 bits using the expansion permutation (E), by duplicating half of the bits. Then, the result is combined with a sub-key using an XOR operation. The XOR output is divided into eight 6-bit and fed into eight substitution boxes (S). Each of these boxes replaces its six input bits with four output bits, according to a non-linear transformation. The outputs are concatenated and pass through a straight permutation (P). The result is processed through XOR function with the left 32 bits and the output is the right bits of the next round. The left bits of the next round are the right bits of the previous round as shown in Figure 3.

After the 16th iteration, the right and left bits are concatenated and finally pass through a final permutation (IP-1), which is the inverse of the initial permutation (IP). The output is the ciphertext block (64 bits).

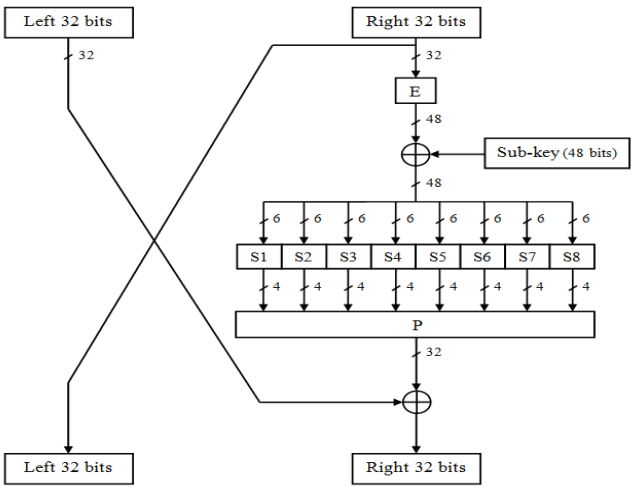

Figure 3. One round of DES

DES is a private key algorithm, in which the same key is used for both encryption and decryption. Figure 4 shows the key schedule generation. It is also an iterative process comprising 16 rounds and generates 16 sub-keys from the main key. The 56-bit effective key gets firstly permutated (PC^{-1}) and splits up into two 28-bit halves; each half is thereafter treated separately. Then, for every round, both halves are shifted left by either one or two bits, depending on the round number. After that, the two outputs go through another permutation (PC^{-2}). The result is the sub-key and it is coded on 48 bits, 24 bits from the left half, and 24 from the right.

The decryption algorithm is exactly the same as the encryption one, but the only difference is that the round keys are used in the reverse order. The output of each round during decryption is the input to the corresponding round during encryption. This greatly simplifies implementation, particularly in hardware, as there is no need for separate encryption and decryption algorithms.

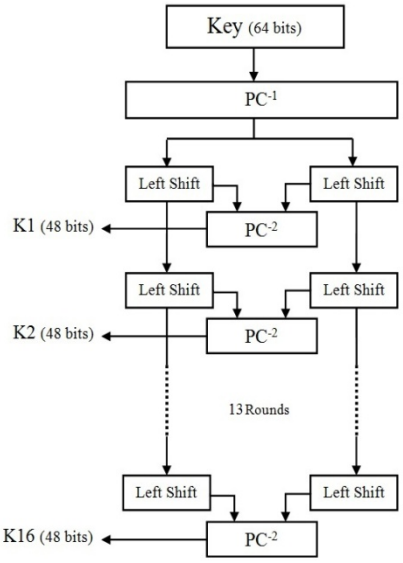

Figure 4. Key schedule generation

3. PIPELINED DES

Pipeline is an important technique to increase the performance of a system [10]. The iterative nature of the DES algorithm makes it ideally suited to pipelining and it can be 4, 6, 8 or 16 stages [11]. The pipelining strategy consists in parallelizing the data inputs and outputs with the processing. Basically, it means to process the data that is given as input in a continuous manner without having to wait for the current process to get over. This pipelining concept is seen in many processors. Registers are used to store the current output of the round that is being executed. In this case instead of passing the output of each round to the next round directly we use a register which would act as a bypass or an internal register. Since the current rounds value is stored in the register the next input to the current round is given as soon as the current output is obtained. In this way the input to the next round is given from the register avoiding a direct contact between the two rounds.

DES implementation presented in this paper is based on 16 stages pipelining. In order to pipeline the algorithm, registers R and L (32-bit) are placed at the left and right of the outputs of each round of the algorithm to allow the sequentiality of the data, as shown in Figure 5.

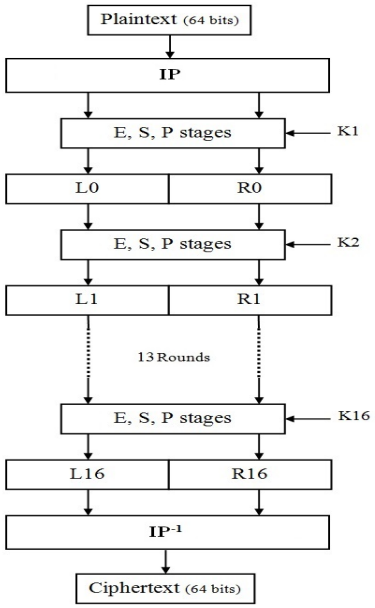

Figure 5. Pipelined DES

4. PIPELINING DES BASED ON TIME VARIABLE SUB-KEYS

There have been several approaches to attack DES algorithm. The most popular is the linear cryptanalysis [6], differential cryptanalysis [7] and exhaustive key search [8]. In order to make DES more secure, we developed a new algorithm shown in Figure 6. It has a key schedule permutation choice 1(PC^{-1}) box that contains four different permutations in order to be used periodically. As a result of this, the sub-keys change. Therefore, every time the plaintexts are encrypted by different sub-keys. Detecting the algorithm will be difficult for the attackers because of the time variant behavior.

We can use several permutations in the box. In our design, we have four. Sender specifies how many clock cycles (N1, N2, N3 and N4) that he will use each of these four permutations. Constantly, the program checks the Time value. If it is less than sender clock cycle value, the permutation is kept, otherwise, the next permutation is selected from the permutation box and time value is set to zero. Flow chart shown in Figure 7 introduces the steps of changing the key schedule permutation choice 1(PC^{-1}).

The sender and receiver have the same permutation box. They are connected to have the same permutation at any specific time. In order to avoid the disadvantage of the synchronization between them, sender transmits additional data with the ciphertexts to receiver, to indicate the correct choice of permutation from the permutation box.

The key schedule is pipelined, registers are placed to store the current output of the round that is being executed, so this part will be performed very fast and the algorithm supports the use of different keys every clock cycle, thus improving overall security since users are not restricted to using the same key during any one session of data transfer. It is noticeable that the design of these registers is the same as registers used in round blocks. Our proposed design in presented in Figure 8.

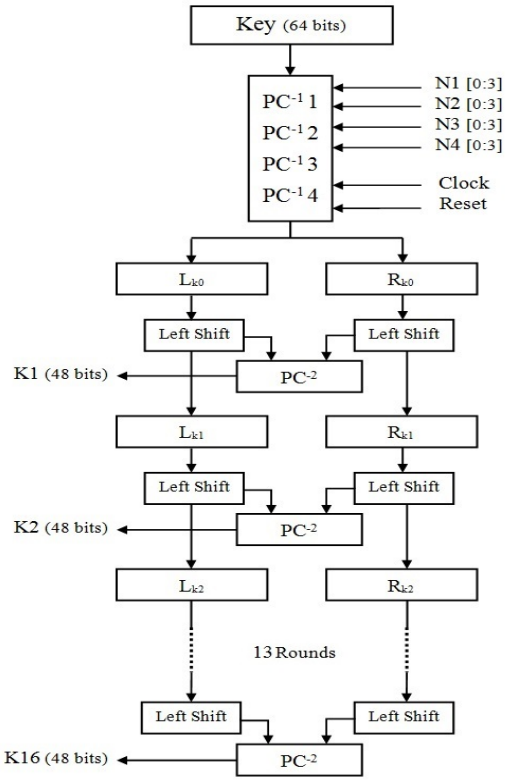

Figure 6. Key schedule based on time variable permutation (PC^{-1})

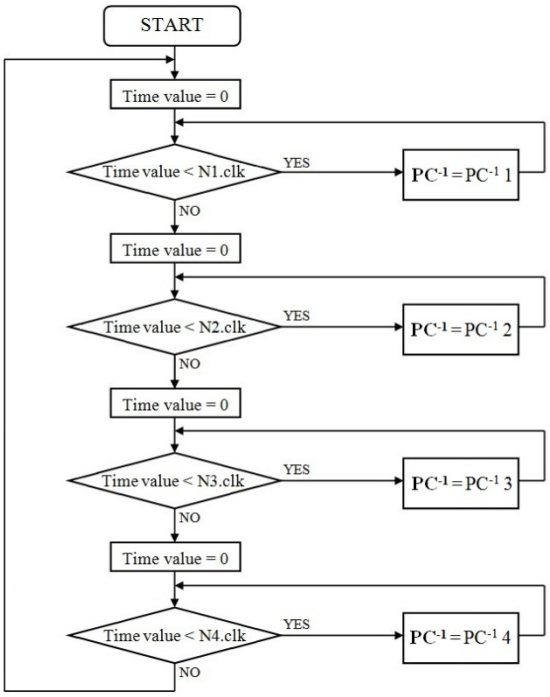

Figure 7. Flow chart of permutation

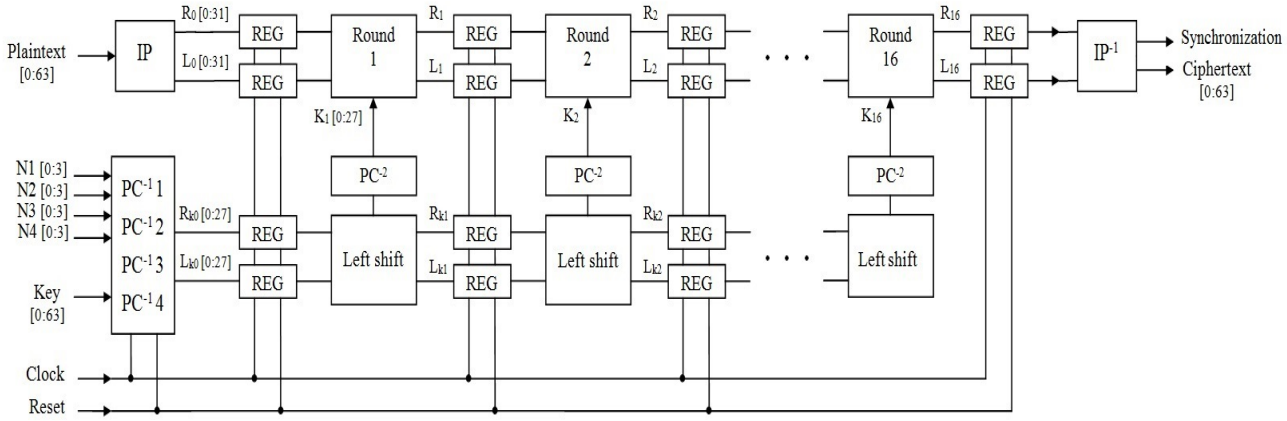

Figure 8. Our proposed design

5. IMPLEMENTATION SUMMARY

Several cryptographic mechanisms have been established in order to combat security threats. Security applications vary in their requirements, which add an extra challenge since a highly secure mechanism may not be the only requirement but rather a more efficient implementation in terms of performance and area.

FPGA implementation of our modified DES algorithm was accomplished on a Spartan-3e device XC3s500e-4fg320 using Xilinx ISE Design Suite 14.7 as synthesis and Modelsim 6.1f as simulation tool. The design was coded using VHDL language. It occupied 2625 (56%) CLB slices, 1989 (21%) slice Flip Flops and 203 (87%) I/Os. The design achieves a frequency of 161.03 MHz. It takes 17 clock cycles latency first time only then encrypts one data block (64 bits) per clock cycle. Therefore, the achieved throughput is (161.03×64) = 10305.95 Mbps and the throughput per CLB slice is (10305.95/2625) = 3.92 Mbps/slice. Simulation window is shown in Figure 9.

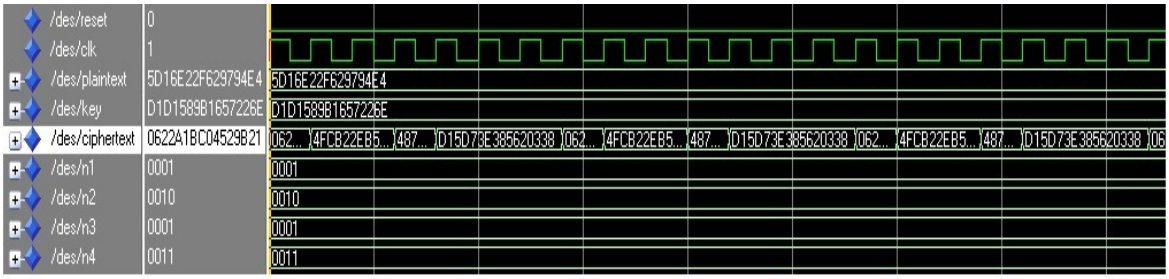

Figure 9. Simulation Window of our DES design

6. PERFORMANCE COMPARISONS

There are several hardware implementations for the DES algorithm that aim to achieve the most efficient architecture, by improving high-throughput and area-efficient. Table 1 shows the performance figures for some representative hardware implementations of the DES.

DES implementation at [11] uses a pipelined design with skew core key-scheduling to load different keys every clock cycle, allowing the possibility of using multiple keys in any one session of data transfer. A Java-base DES implementation achieves the fastest encryption rate of 10752 Mbps [12]. It utilizes Jbits on a Virtex XCV150-6 device. Jbits provides a Java-based Application Programming Interface (API) for the run-time creation and modification of the configuration bitstream. In this design the key schedule is computed in software. Also, it can only accommodate one key per data transfer session. In [13], the implementation uses a non-standard representation and view the processor same as a SIMD (Single Instruction Multiple Data) computer, as 64 parallel one-bit processors computing the same instruction. A VLSI DES implementation

uses 0.6 micron CMOS technology [14]. DES implementation at [15] uses a novel method for implementing the key schedule. This method supports the use of different keys every clock cycle and utilizes permutations to create the sub-keys from the input key. The sub-keys are delayed by the required amount using the necessary array of latches. The Implementation of DES at [16] is based on time variable data permutation. The design uses an initial permutation box that contains several permutations to be selected periodically. A single-chip implementation of an iterative DES algorithm is presented in [17].

The design implemented all DES primitives in one-round scheme. Implementation in [18] used pipelining techniques to enhance the throughput of their iterative design.

The designs in [12-14, 17-18] use the implementation of the original version of DES. In [16], DES is implemented with time-varying behavior. DES with the possibility to support the use of different keys per data transfer session is presented in [11, 15]. Comparing our proposed design with these implementations, we conclude that it is more secure due to the time-varying behavior and the possibility to use different keys every clock cycle.

Table 1. Performance comparison

Authors	Device used	CLB slices	System clock (MHz)	Throughput (Mbps)	Throughput per slices (Mbps/slice)	
Patel, Joshi, Saxena [11]	XC3S500E	2814	111.882	7160	2.54	16-stage pipelined designs
Patterson [12]	XCV150	1584	168	10752	6.78	
Biham [13]	Alpha 8400	---	300	127	---	
Wilcox, Pierson, Robertson, Witzke, Gass [14]	ASIC	---	---	9280	---	
McLoone, McCanny [15]	XCV1000	6446	59.5	3808	0.59	
Abd El-Latif, Hamed, Hasaneen [16]	XC3S500E	2062	124.734	7983	3.87	
Kaps, Paar [18]	XC4028EX	741	25.18	402.7	0.543	
Wong, Wark, Dawson [17]	XC4020E	438	10	26.7	0.061	One round design

From the results in the table, we find that our 16-stage pipelined design gives 1.439, 81.149, 1.11 and 2.706 times more throughput than the designs in [11, 13-15], respectively. Looking at the CLB slices area count, our design needs only 0.932 times the CLB slices used in [11] and 0.407 times in [15]. CLB slices in [13] and [14] are not reported.

Also from the table, we note that our proposed design gives 1.29, 25.592 and 385.99 times more throughput than the pipelined designs [16, 18] and the iterative one [17]. But, it needs 1.273 times the CLB slices used in [16], 5.993 times in [17] and 3.542 times in [18]. However, our design achieves higher throughput per CLB slice than the three designs [16-18].

The proposed design achieves only 0.958 times throughput of the design in [12] and needs 1.657 times the CLB slices used. However, this design is not a single-chip implementation of the full DES algorithm since the key schedule is computed in software and it can only support one key per data transfer session [12].

From the comparison, we notice that our implementation is competitive with the reported implementations. It is more secure and one of the fastest single-chip FPGA designs with area efficient.

7. CONCLUSION

This paper presents an efficient implementation for the design of a 16-stage pipelined DES algorithm with time variable sub-keys. The algorithm uses different key schedule permutation choice 1 (PC^{-1}) periodically with time. Therefore the ciphertext changes by time for the same key and plaintext. As a result of this, the security of the algorithm is increased. The plaintext blocks can be loaded every clock cycle and after an initial delay of 17 clock cycles, the ciphertext blocks will appear on consecutive clock cycles. The implementations of the DES algorithm based on hardware are low cost, flexible and efficient encryption solution. The implementation of our design is presented by using Spartan-3E (XC3S500E) family FPGAs and is one of the fastest hardware implementations with much greater security. At a clock frequency of 161.03MHz, it can encrypt or decrypt data blocks at a rate of 10305.95 Mbps.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of Moulay Ismail University in the realization of this work.

REFERENCES

[1] W. Stallings, "*Cryptography and Network Security Principles and Practices*", Prentice Hall, November 16, 2005.
[2] A. Kahte, "*Cryptography and Network Security*", Tata Mcgraw Hill, 2007.
[3] National Institute of Standards and Technology (NIST), "Data Encryption Standard", *Federal Information Processing Standards Publication*, 46-3, 1999.
[4] National Institute of Standards and Technology (NIST), "DES modes of operation", *Federal Information Processing Standards Publication*, 81, 1980.
[5] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard", *Federal Information Processing Standards Publication*, 197, 2001.
[6] M. Matsui, "The first experimental cryptanalysis of the data encryption standard", *Advances in Cryptology*, 14th Annual International Cryptology Conference, California, USA, pp. 1-11, 1994.
[7] E. Biham, A. Shamir, "Differential cryptanalysis of DES-like cryptosystem", *Journal of cryptology*, vol. 4, no. 1, pp. 3-72, 1991.
[8] W. Diffie, M.E. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10, no. 6, pp. 74-84, 1977.
[9] J. Kaps, "High Speed FPGA Architectures for the Data Encryption Standard", Master's thesis, ECE Dept., Worcester Polytechnic Institute, Massachusetts, USA, 1998.
[10] S. Taherkhani, E. Ever, G. Orhan, "Implementation of Non-Pipelined and Pipelined Data Encryption Standard (DES) Using Xilinx Virtex-6 FPGA Technology," 10th *IEEE International Conference on Computer and Information Technology*, Bradford, UK, pp. 1257-1262, 2010.
[11] V. Patel, R.C. Joshi, A.K. Saxena, "FPGA Implementation of DES Using Pipelining Concept With Skew Core Key-Scheduling", *Journal of Theoretical and Applied Information Technology*, vol.5, no.3, pp. 295-300, 2009.
[12] Patterson, "High Performance DES Encryption in Virtex FPGAs Using Jbits", In *Field-Programmable Custom Computing Machines, IEEE Comput. Soc*., Napa Valley, California, USA, pp. 113-121, 2000.
[13] E. Biham, "A Fast New DES Implementation in Software", 4th *International Workshop on Fast Software Encryption*, Israel, pp. 260-271, 1997.
[14] D.C. Wilcox, L. Pierson, P. Robertson, E. Witzke, K. Gass, "A DES ASIC Suitable for Network Encryption at 10 Gbps and Beyond", First *International Workshop on Cryptographic Hardware and Embedded Systems*, Massachusetts, USA, pp. 37-48, 1999.
[15] M. McLoone, J.V. McCanny, "*High-performance FPGA implementation of DES using a novel method for implementing the key schedule*", IEE proc: Circuits, Devices and Systems, vol. 150, no. 5, pp. 373-378, 2003.
[16] K.M.A. Abd El-Latif, H.F.A. Hamed, E.A.M. Hasaneen, "FPGA Implementation of the Pipelined Data Encryption Standard (DES) Based on Variable Time Data Permutation", *The Online Journal on Electronics and Electrical Engineering*, vol. 2, no. 3, pp. 298-302, 2011.
[17] K. Wong, M. Wark, E. Dawson, "A single-chip FPGA implementation of the data encryption standard (DES) algorithm", *IEEE Globecom Communication on the Bridge to Global Integration*, Sydney, Australia, pp. 827-832, 1998.
[18] J. Kaps, C. Paar, "Fast DES Implementations for FPGAs and Its Application to a Universal Key-Search Machine", 5th *Annual Workshop on selected areas in cryptography*, Ontario, Canada, pp. 234-247, 1998.