THE INTERNATIONAL JOURNAL OF
SCIENCE & TECHNOLEDGE

# Design of High Performance Pipelined Data Encryption Standard (DES) Using Xilinx Virtex-6 FPGA Technology

**S. B. Sridevi**
M.Tech., Faculty, Department of ECE, SPMVV, Tirupati, India
**B. Alekya Himabindu**
M.Tech., Faculty, Department of ECE, SDITW, Nandyala, India

***Abstract:***
*This paper describes the design of high performance Data encryption process is a predominant symmetric-key algorithm for the encryption of electronic data. It was highly influential in the advancement of modern cryptography and it is easily be quite complicated and usually requires significant computation time and power despite significant simplifications. This discusses about the pipelined implementation and it is most commonly used symmetric encryption algorithm, Data Encryption Standard (DES). The platform used for this matter is, Xilinx new high performance silicon foundation, Virtex-6 Field Programmable Gate Array technology. The testing of the implemented design shows that it is possible to generate data in 16 clock cycles when non-pipelined approach is employed and then the pipelined approach is employed on the other hand, 17 clock signals are required for the initial phase only, and one clock signal is sufficient afterwards for each data generation cycle. In this, the Very High Speed Integrated Circuit Hardware Description Language (VHDL) is used to program the design.*

***Key words**: Data Encryption Standard, Field Programmable Gate Arrays, Very High Speed Integrated Circuit Hardware Description Language, design methodology*

## 1. Introduction

The rapid growth of secure transmission is a critical point nowadays. We have to exchange data securely at very high data rates. Efficient solutions, with huge data rate constraints, have to be hardware implemented and exible in order to evolve with the permanent changes in norms. The most commonly used technique for producing confidentiality in data transmission is symmetric encryption [1]. Symmetric encryption scheme, also referred to as single key encryption method, has five main modules. The term plaintext is used to denote the original incoming data. The key is an essential part of the encryption process and it provides the secure data traffic among the sender and the recipient. Encryption algorithm performs various mathematical and logical functions on the plaintext by using the key. Cipher data is the encrypted message produced by encryption algorithm by using the key and the plain text. Decryption algorithm is employed on cipher data and performs reverse action to generate the plain text. The symmetric key encryption algorithm shares the same key between sender and receiver and a strong algorithm for encryption and decryption processes is essential to provide an efficient security mechanism [2].
The block cipher algorithms are commonly used symmetric encryption techniques which analyse the input data as fixed size blocks and produce the scrambled data as equal size as the original data. Due to heavily increased volume of sensitive information, recently the implementations of cryptographic algorithms have been increased as countermeasures for security threats. The diversity of the security applications has also been raised dramatically. This trend has introduced additional challenges not only in terms of highly secure algorithms but also in terms of fast solution approaches for high performance applications. Cryptographic designers have explored not only realizations on software platforms but at the same time on hardware platforms [1], [3]. In a single chip implementation of DES algorithm is presented for the first time by using Xilinx XC4000 series Field Programmable Gate Array (FPGA) under the XACT step design flow integration. In a Java-based application programming interface implementation of the DES algorithm in a Virtex FPGA is presented. It is shown that, Dynamic specialization of the DES circuit gives better performance where the overall cost is lower. The implementation also reduces the requirements in terms of device size, pin and power. Similarly in the implementation of high performance DES and Triple DES algorithms are considered by using Virtex™-II and Virtex-E devices. The authors claim that the encryption hardware proposed does not become a bottleneck even for the networks capable of transferring several gigabits per second. A compact and efficient reconfigurable hardware implementation of DES algorithm is presented on a VirtexE XCV400e device. The design provided, establishes a speedup almost 10 times better than the design presented in within less configurable logic block (CLB) slices. In [3] theoretical analysis of DES Algorithm is performed and a new algorithm is proposed to increase the performance. New design of encryption key and substitution boxes (S-box) which are used to obscure the relationship between the key and the cipher text is

provided together with the FPGA implementation as a prototype. Various approaches are presented in above mentioned studies to improve the performance of DES algorithm.
Some approaches improve the theoretical part of DES such as [3] and others improve the DES based on the different implementation approaches such as [4]. The last well known, fastest and fully pipelined implementation of DES was announced by Xilinx Company where the modification was on the implementation part and not on the theoretical algorithm of DES. The data rate in the fore mentioned design is 15.1 Gbps using VirtexII FPGA platform . In this study, we use the first definition of DES and in addition to comparison of non-pipelined and pipelined approaches, we propose a pipelined design with data rate up to 18.82 Gbps by using Virtex 6 FPGA technology. The paper is organised as follows: The next section presents the DES algorithm under study. The section on Xilinx Virtex- 6 FPGA provides the details of the hardware platform employed. The sections on non-pipelined and pipelined implementation give the details of two different approaches respectively.

## 2. DES Algorithm

DES algorithm uses complicated logical functions such as various types of permutations, XOR and SHIFT functions. Since the key employed is transformed to mentioned function, by following the algorithm provided, the only way to decrypt the plaintext is to apply the same key in decryption algorithm as well. DES takes 64 bits plaintext and 56 bits key as input and generates 64 bits cipher data as output [2]. The block diagram of algorithm is shown in Fig. 1. Sometimes the key is considered as 64 bits where 8 bits is used for parity check. The DES structure is first described by Horst Feistel in 1973 [2], as shown in Fig. 2.
In this method, after initial permutation (IP) of the plaintext, it is divided into two halves L(0), R(0). The two halves pass through 16 rounds. Then after the final permutation (FP), the cipher data is produced. IP and FP work exactly in opposite ways to each other [2]. Each Round i has three inputs: L (i-1), R (i-1) and K (i) where K (i) is generated from Key Scheduler program which is described in the following section. All rounds have the same structure. In Round i, L (i) is equal to R (i-1). But R (i) is derived from [L (i-1) XOR F(R (i-1), K (i))a]. Where XOR is an exclusive-OR operator and F is a round function. Thefunction diagram of F is shown in Fig. 3.

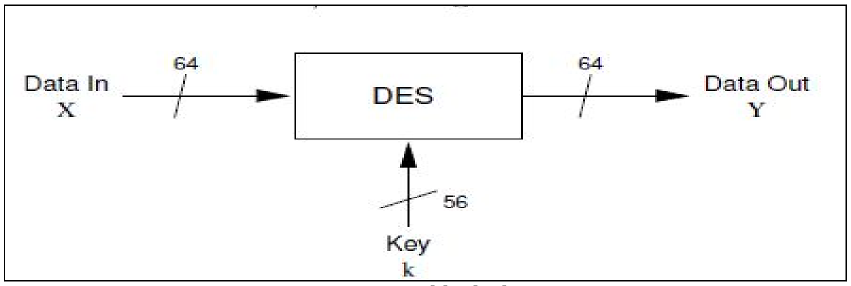


*Figure 1: DES block elements*

The F(R, K) or round function is the core of the DES. It substitutes the right half of data and generate 48 bits of data.After taking XOR with corresponding key K (i), it passes the data through S-box functions where each S-box function has its own look up table. After that a permutation of output of Sbox function is provided and, the result of F(R, K) is generated.
The key scheduler program generates a sub key K (i) and form the 56 bits key by taking different permutation and rotate the function to left (in encryption mode) or right (in decryption mode) on the original key. That means, for each round i, a different sub key, K (i), is generated. The decryption algorithm is similar to encryption algorithm. The only difference is the sequence of generated sub key. It assigns the K (16) to round 1 and K (15) to round 2 and so on [2]. An alternative to that would be to use rotate right instead of rotate left in key scheduling part.

## 3. XILINX VIRTEX-6 FPGA

An FPGA is an integrated circuit which contains array of programmable logic cell in rows and columns [5]. FPGAs' performance and features vary for different vendors. The Virtex-6 is a new FPGA from Xilinx built on 40nm process technology [5] which is one of the fastest FPGA in the world. The Virtex-6 family aims to provide up to 50% lower power and 20% lower cost than previous generation. The basic programmable part of FPGA is called slice. Each Virtex-6 FPGA slice contains four LUTs (Look up Table) and eight flip-flops. Some of the slices can use their LUTs as distributed RAM. The simplified diagram of the slice is shown in Fig. 5.

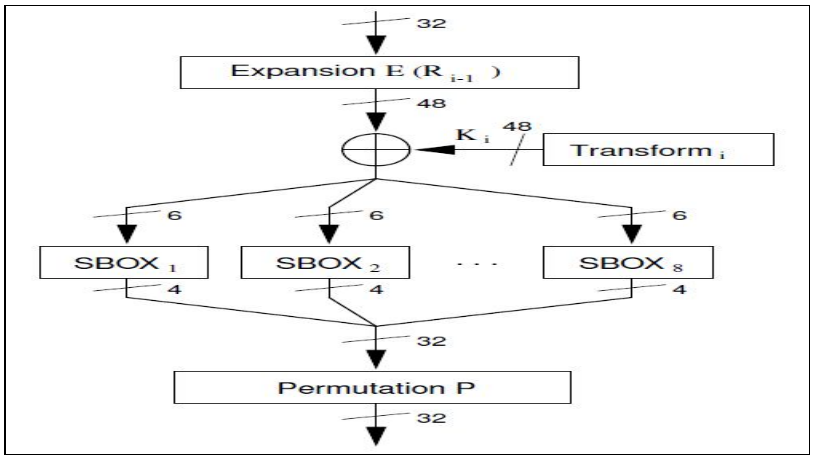


*Figure 2: DES Algorithm*

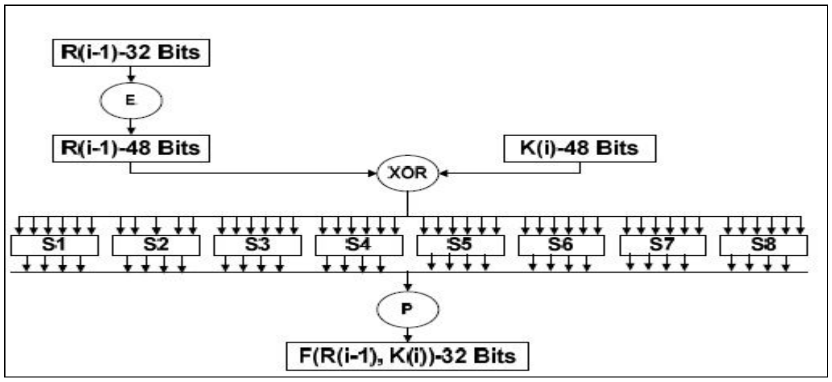


*Figure 3: F (R, K)*

**4. Programming Design**

The hardware description language used to build this design is VHDL. Different capabilities and features of VHDL lead to various implementation of the design in terms of performance and speed. Modelsim-SE [6] and ISim software are used as simulation tools and XST is used as a synthesis tool.

**5. Non-Pipelined Implementation**

The non-pipelined mode design is divided into three main function blocks. The block diagram of the design is shown in Fig. 6. In traditional implementation, counters are used inorder to control the round sequence. On the other hand, in prevalent designs, Finite State Machines (FSM) are used to control the round sequence.

In non-pipelined mode implementation, FSM is used to control the round sequence and handshaking between different function blocks. It generates the appropriate data for key scheduler and F(R, K) function blocks, and it receives the produced data from them. Xilinx XST uses processes [7] to describe FSM in VHDL language. Most of the FPGA synthesis tools use various templates for implementing the FSM. XST has 8 different
techniques to describe FSM:

- Auto-State Encoding
- One-Hot State Encoding
- Gray State Encoding
- Compact State Encoding
- Johnson State Encoding
- Sequential State Encoding
- Speed1 State Encoding
- User State Encoding

Auto-State Encoding is used in this study, which tries to select the best suited algorithm for a FSM. FSM modules are implemented in slice logic (LUT) by default. However it can also be implemented into the block RAM. For larger FSM, using block RAM makes FSM faster and leaves the slice logics to its targeted design which causes better utilization in device slice usage. Only Virtex FPGA family uses block RAM feature [7].

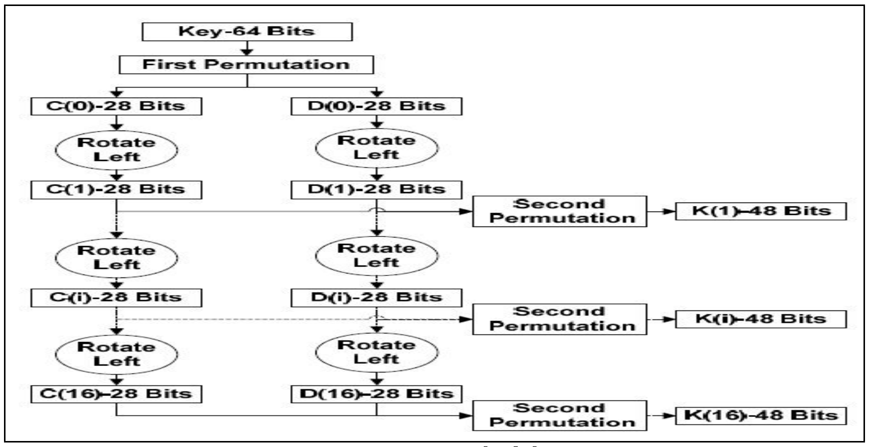


*Figure 4: Key Scheduler*

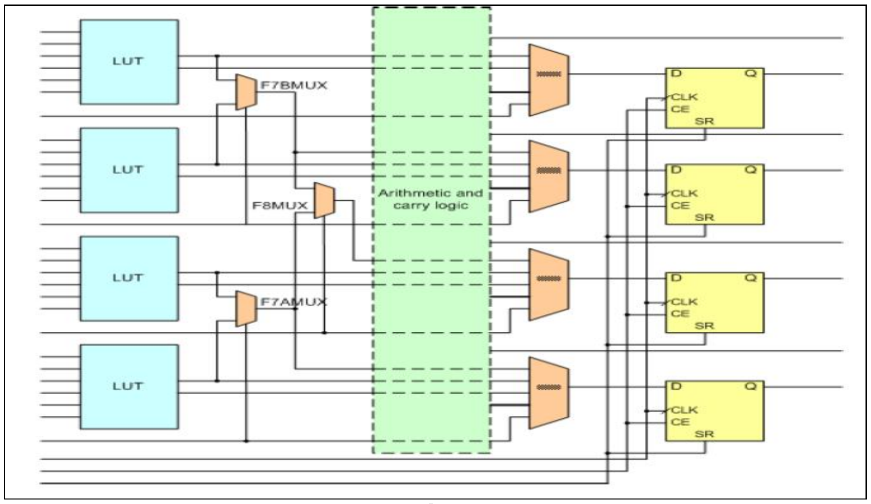


*Figure 5: Slice Diagram*

In the non-pipelined mode implementation, the cipher code will be generated in 16 clocks. The decryption process will generate the original data in 16 clocks as well. The simulation waveform created by Modelsim [7] is shown in Fig. 7. The waveform depicts the following information:
Encryption Process,input data (64 bits): x"AAAAAAAAAAAAAAAA"
key (64 bits) x"BBBBBBBBBBBBBBBB"
ciphered Data (64 bits) x"AC972FC04E884797"
Decryption Process input data (64 bits): x"AC972FC04E884797"
key (64 bits) x"BBBBBBBBBBBBBBBB"
deciphered Data (64 bits) x"AAAAAAAAAAAAAAAA".

**6. Pipelined Implementation**
Pipeline is an important technique to increase the performance of a system. The basic idea is to overlap the processing of several tasks so that more tasks can be completed in the same amount of time. If a combinational digital circuit can be divided into stages, we can insert buffers (registers) at proper places and convert the circuit into a pipelined design [8]. Two criteria are used to examine the performance of a system:
a. Delay: is the time required to complete one task.
b. Throughput: is the number of tasks that can be completed per unit time.
Adding pipeline into a combinational design can only increase a system's throughput. Such an approach does not reduce the delay in an individual task. Actually, because of the overhead introduced by the registers and non-ideal stage division, the delay will be worse than that of the nonpipelined design [8].Various implementations are presented in various platforms for DES algorithm in [9], [10]. However the design provided in this study is implemented in Virtex6 FPGA. In the current pipelined design, we put buffers in input and output stages of each round. The key scheduler part does quite a novel task in current implementation. It floods the sub keys to appropriate round.
The key scheduler implementation specifications are as follow:
a.In output part of sub key 1 K(1), there is one register which means in every one clock signal, K(1) will be transferred to core function (f) of program.

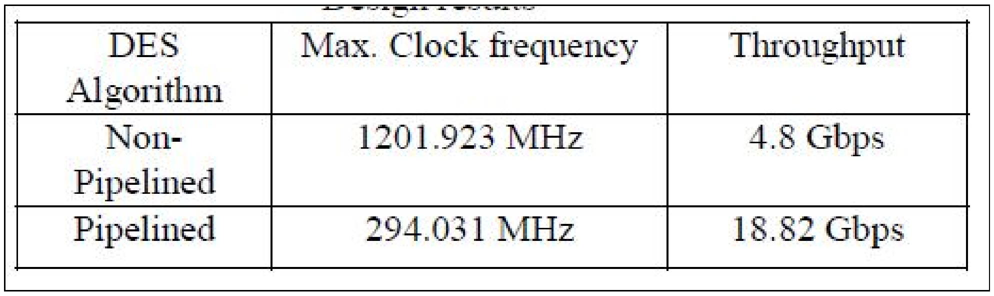
b. In output part of sub key 2 K(2), there are two registers which means in every two clock signal, K(2) will be transferred to core function (f) of the program. This sequence will continue until 16th sub key K(16) which will be generated and transferred to control part of the program in every 16 clock signal.
Fig. 8 shows the diagram of generating the sub keys and core function. As a result, for a specific plaintext and key, the appropriate K(1) to K(16) will be generated exactly at the needed clock cycle. For proper result, key scheduler must be synchronized properly to core function otherwise the result will be incorrect.
Fig. 8 illustrates the simulation wave form generated byISim [14], for the following test bench (encryption):

```
DATA<=X"ffffffffffffffff";
Key<=X"ffffffffffffffff";
Wait for 10 Ns;
Wait for 10 ns;
DATA<=X"0000000000000000";
Key<= X"0000000000000000";
Wait for 10 ns;
Key<=X"3b3898371520f75e";
Wait for 10 ns;
DATA<=X"1111111111111111";
```

The results clearly show that, in pipelined implementation, the ciphered or deciphered data will be generated in one clock signal.

## 7. Results

The timing report of both designs is briefly shown in Table I. It is obvious that there is a trade-off between clock or delayand throughput. While we increase the throughput, we face with some delays in the design. In the current design, while we increase the throughput from 4.8 Gbps to 18.82 Gbps, we lose clock frequency from 1201.923 MHz to 294.031 MHz. However, the throughput of current pipelined design is more than the Xilinx one [10] which was 15.1 Gbps.



| DES Algorithm | Max. Clock frequency | Throughput |
|---|---|---|
| Non-Pipelined | 1201.923 MHz | 4.8 Gbps |
| Pipelined | 294.031 MHz | 18.82 Gbps |

Table 1: Design results

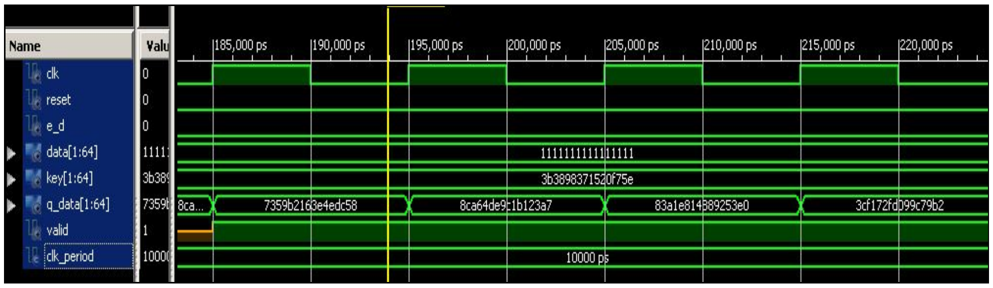


Figure 7: Waveform of DES simulation (Pipelined Mode)

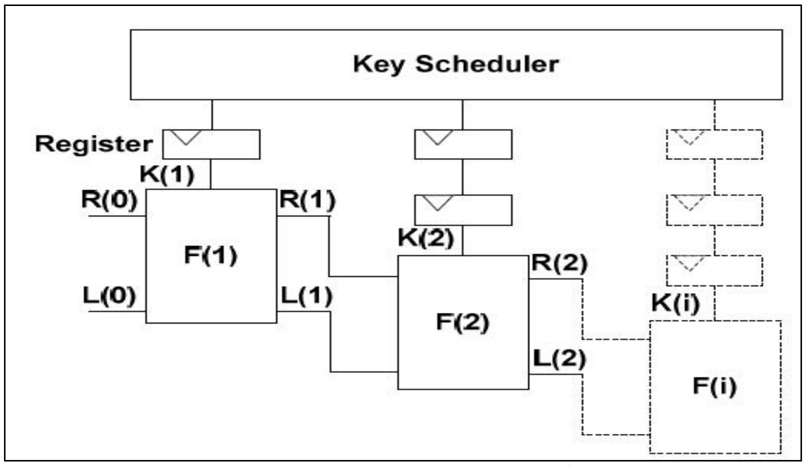


*Figure 8: Core Function and Key Scheduler Task for Pipelined Mode*

## 8. Conclusion

The pipelined DES algorithm implementation are presented in this paper. The results shows that it is possible to implement a design in order to operate at high system clock frequency (1.2 GHz) and the original implementation of the DES algorithm is considered and the theoretical part is not modified. The implementation presented by using Virtex-6 family FPGAs have better performance than the existing ones since it is possible to have throughput going up to18.82 Gbps by implementing a fully pipelined design including pipelined key schedulers.

It is desirable to implement the improved DES algorithm presented by using a similar approach in order to test the performance improvements for the future studies. Optimization and synthesis of design is carried out at latest and fastest FPGA Viretx-6 device that improves performance. The pipelined design represents better performance than the non pipelined implementation.

## 9. References

1. C. Patterson, “High performance DES encryption in Virtex FPGA’s using JBits,” Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on, pp.113-121.
2. Data Encryption Standard, FIPS PUB 46, 1977 Jan 15, available from NTIS; Springfield, VA 22151 USA.
3. Fu Li, Pan Ming, “A simplified FPGA implementation based on an Improved DES algorithm,” IEEE Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on, pp.227-230.
4. Ke Wang, “An encrypt and decrypt algorithm implementation on FPGA’s,” IEEE Semantics, Knowledge and Grid, 2009. SKG 2009. Fifth International Conference on, pp.298-301.
5. L. Floyd, “Digital Fundamental with VHDL,” pp.362-368, ISBN: 0-13-099527-4, Pearson Education, 2003
6. Mentor Graphics, “Modelsim Data sheet,” 2008.
7. N. A. Saqib, F. Rodrıguez-Henriquez, and A. Dıaz-Perez, “A compact and efficient fpga implementation of the DES algorithm,” 2004.
8. Pong P. Chu, “RTL Hardware Design Using VHDL,” pp.293-348, ISBN: 0-471-72092-5, Wiley-Interscience, 2006.
9. Teo Pock Chueng, Yusoff, Z.M., Sha’ameri, A.Z., “Implementation of Pipelined Data Encryption Standard (DES) Using Ultera CPLD,” IEEE TENCON 2000. Proceedings, publication year 2000, Page 17-21 vol. 3.
10. V. Pasham, S. Trimberger, “High-Speed DES and Triple DES Encryptor/Decryptor,” Aug. 2001,
11. W. Stallings, L. Brown “Computer Security Principle and Practice,” pp.593-600, ISBN: 978-0-13-600424-0, Pearson Education, 2008
12. Wong, K., Wark, M., Dawson, E.: A Single-Chip FPGA Implementation of the Data Encryption Standard (des) Algorithm. In: IEEE Globecom Communication Conf., Sydney, Australia (1998) 827–832