PSEUDO RANDOM KEY GENERATION IN DATA ENCRYPTION STANDARD ALGORITHM

Radha.K[1], Swathi.R[2], Shankari.K[3]

[1] *Student, Electronics and Communication, Prince Shri Venkateshwara Padmavathy Engineering College, Tamil Nadu, India*
[2] *Student, Electronics and Communication, Prince Shri Venkateshwara Padmavathy Engineering College, Tamil Nadu, India*
[3] *Assistant Professor, Electronics and Communication, Prince Shri Venkateshwara Padmavathy Engineering College, Tamil Nadu, India*

ABSTRACT

For the construction of cipher based applications, including health-monitoring and biometric data based recognition system, need short-term data security. Data encryption standard (DES) is well-fitted for the operation of low-cost lightweight cryptography applications. In this paper, an effective VLSI architecture for DES algorithm based encryption/decryption engine is explained. Depending upon the enciphering/deciphering needs, the same set of architectures performs both encipher and decipherment operations. In DES algorithm, in the generation of sub keys pseudo random key generators are utilized. The VLSI architecture is modeled in the VHDL or Verilog design language and synthesized in the Xilinx field-programmable gate array (FPGA) device.

Keyword: - *Cryptography, DES, Encryption/Decryption, Pseudo random key, VLSI architecture, FPGSs etc.….*

1. INTRODUCTION

In the last decade, a strong need for portable devices that can store confidential data (e.g., Smartcards) has led to the widespread adoption of cipher algorithms and circuits. However, when communicating with the outside environment the devices used for ciphering drip information. Cryptography is the process adopted to ensure the protected repository of data in an unprotected repository place by avoiding the eaves drops. Secured data exchange over public computer network is achieved by authentication, confidentiality and integrity. A considerable exploration has been focused on providing embedded systems with cipher functionality without overloading the already weak sources of embedded system. In modern verification-based applications, such as: bank transactions, electronic mail, audio/video conferencing etc., secure communication is very essential. Cryptography is a science that enables the classified nature of communication through an insecure channel. Nowadays cipher/decipher is widely used for computerized payment in prepaid telephone cards, e-cash cards etc... Some of the applications including health-monitoring and biometric data based recognition applications need short-term security. For the implementation of low-cost lightweight ciphering, the DES algorithm is very well appropriate. The implementation of DES in VLSI architecture follows an expensive and time consuming fabrication process. Computer program operation offer high adjustability but they are not fast enough for the applications where span factor is crucial. On the other hand, reconstruct able devices are alluring since the time and costs of VLSI conception and fabrication can

be cut down. Moreover, they offer high likeliness for reprogramming and experimenting on multiple architectures or several revisions of the same architecture. Among the different cipher algorithms, the most popular example in the field of proportional ciphers is the Data Encryption Standard (DES) algorithm which was developed by IBM in the mid-seventies. The DES algorithm is formulated in repetitive rounds composed of several bit-level operations such as logical operations, permutations, substitutions, shift operations, etc. Although those features are naturally fitted for efficient operations on reconfigurable devices FPGAs, DES operations can be found on all platforms: software, VLSI and reconfigurable hardware using FPGA devices. In this project, an adequate and compact DES architecture especially with pseudo random key generators outlined for reconfigurable hardware platforms is designed. The DES implementation presented in this project differs from other previous works by making use of a pseudo random generator for key generation. The leaving of this project is detailed and describe the DES algorithm. The projected DES architecture and its implementation on a reconfigurable hardware device. The carryover of this paper is methodized as follows. Section 1 describes the generalized introduction about the proposed system. Existent system explains about the use of Data encryption standard algorithm, Section 2 provides a detailed description about the projected system, which provides explanation about the DES. A brief description about the proposed system is being explained in this section. Section 3 provides the explanation about the key generation process. Section 4 deals with the experimental results. Section 5 deals with the conclusion.

2. DES ALGORITHM

In the data encryption, standard algorithm initially, a plain text of maximum 64-bit data is been given as the input as in Fig 1. This plain text is divided into two 32-bit left and right registers with the help of initial permutation process. In the initial permutation process the 64-bit input plaintext is divided into left and right 32-bit registers wherein permutation is achieved by a bit selection process the 32-bit right half bock of data contents are then been subjected to operation of expansion permutation wherein the 32-bit block of data is being expanded to 48-bit of data for the purpose performing XOR operation with the sub-keys generated as another input. The key generation process is as explained in section 3. The output of the XOR operation performed between two inputs, expansion permutation input and key is being fed into the substitution boxes. There are eight substitution boxes, the 48-bit output of the XOR operation is divided into eight substitution boxes such that each box is given an input of 6-bit and the outcome of each substitution box is given as 4-bit. Thus, the final output of all the eight substitution boxes is given as 32-bit. This output 32-bit of data is being given as one of the input to the XOR operation along with the left 32-bit block of register and the output is stored in the temporary register. In the DES algorithm, sixteen rounds or iteration is being followed up wherein after the completion of the first round and for every such iteration the temporary register is transferred to the right block and the data contents in right block are transferred to the left block. After the successful completion of the sixteen iterations, the left bock of register and the right block of register undergoes inverse initial permutation process and the cipher text is generated.

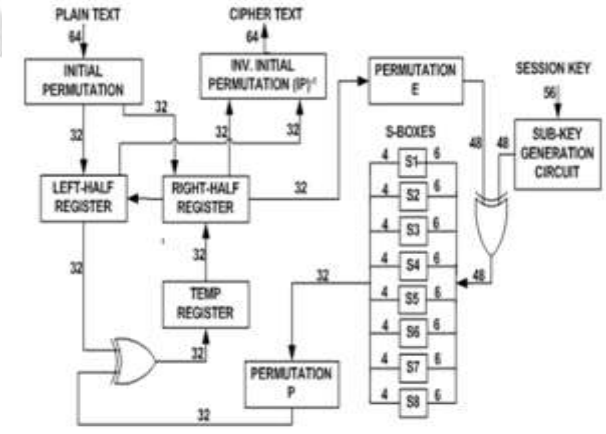

Fig -1: Basic block diagram of DES algorithm

3. GENERATION OF PSEUDO-RANDOM KEY

In the previous implementations of data encryption standard algorithm in VLSI the key or sub-key generation process begins with an input of 64-bit key unit by the user where it is subjected to division of two blocks of 32-bit size data blocks. One of the 32-bit bock of data will be processed to circular left shift operation and another 32-bit half block of data undergoes right circular shift operation. Finally, the outcome of both right and left 32-bit bock of data is been combined by using the compression permutation unit, which by using a certain bit allocation process converts the 64-bit block of contents formed using the input key into 48-bit block of data for processing it in the upcoming XOR operation involving input of 64-bit size. In the proposed architecture, the pseudo random key generators are being used to generate sub-keys using the initial input key given by the user. Even in this design the 64-bit block of data contents is sub-divided into 32-bit blocks of data into left 32-bit register and right 32-bit register using standard bit allocation process and performing XOR operation between the two 32-bit blocks of data and feeding this output as the input to an NOT operation, a 64-bit outcome is achieved as in Fig 2, which is then processed through compression permutation block to get a 48-bit block of data.

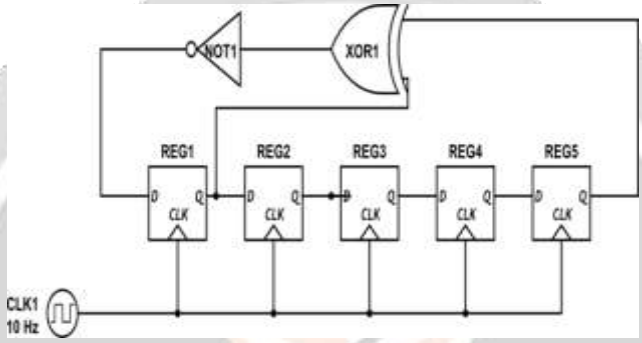

Fig -2: Pseudo random key generator

In the host device of Pseudo random number generators, they are limited by the entropy. These entropy sources determine the quality of the sequences resulting from the random number generation. If the initial values of a pseudo random number generator are known, every other value in the sequence is easily determined and even recalculated. This makes securing pseudo random generators against attackers of pivotal importance. The generator should be designed in such a way that determining the internal variables at any given time is impracticable. Going further, assume that an attacker determines the internal variables at any point of time, pseudo random generators will still be able to protect themselves due to their randomness. Forward security is the term used to describe a generator where knowing the internal state of a generator at a point in time will not help an attacker learn about previous outputs. If random number generators are being used for purposes like password creation, keeping up forward security becomes vital. Backward security denotes that an attacker who learns the state of the generator at a point in time will not be able to determine future numbers that will be produced. Backward security is only possible if the generator introduces some level of entropy into its equation. True random number generators always have forward and backward security, because they have no deterministic components. Outside attackers are not the only problem inherent in pseudo random generators. At times, honest or malicious mistakes can render an entire generator insecure. For example, the National Institute of Standards and Technology, or NIST, periodically publishes a list of pseudo random generators it deems secure enough for cryptography. In their publication, one of the four generators listed was championed by the National Security Agency, or NSA. Theoretically, there exists a set of constant numbers that, when known, would allow an attacker to predict every value of NSA's generator after collecting thirty-two bytes of random output. Although it is impossible to tell if the NSA possessed that set of constants, or even knew that such a thing existed, this served as a reminder that all pseudo random generators should be thoroughly examined by experts before they are trusted. Since random numbers are used in many important applications such as setting up secure Internet communications, there are many groups that desire to crack the generators involved. Pseudo random number generators generally lack entropy after initialization, so once they are broken the attacker requires no additional effort to monitor the system. Additional complexity and thorough security checks need to be included in all pseudo random number generators to make them safe for public use. Aside from vulnerability to attacks, all pseudo random generators share fundamental limitations. Without continual entropy, random generators can only create sequences based off a limited set of initial conditions. Sequences created this way can only last a limited amount of time before they reach their starting point and repeat themselves exactly. The

length a sequence can extend before it repeats itself is referred to as its period. A major consideration in the choice of a pseudo random number generator is the size of its period, because this directly affects the frequency that a generator can be used. Pseudo random generators are capable of producing sequences at rapid speeds, so in applications that use vast quantities of randomness, the threat of exceeding the period is not trivial. The field of cryptography also contributes to the creation of pseudo random numbers. Good encryption techniques that make messages undecipherable can also be used to encrypt a starting value into seemingly random numbers. Most encryption techniques have a poor production rate however, so using encryption techniques in generators is generally a bad idea. At times, it can be easier to describe pseudo random number generators in terms of hardware instead of their mathematical form. This is the case with feedback shift registers. These are best visualized as a string of n bits sitting inside a register in hardware. An even number of positions are selected: such as indexes five, seven, nine, and n. These generators operate by performing XOR on the bits at these positions, taking the result as the new leftmost bit, and shifting the rest of the string right by one. The bit that gets shifted out is the next random bit in the generators output

4. EXPERIMENTAL RESULTS

A Modelsim simulation result of the proposed design is shown in Fig 3 and 5. The key generation is depicted in Fig 4. The CLK is given a logic '1', the chip select and RST are given a logic '0'. The simulation of the design has been thoroughly verified by providing sample vectors in the form of uniform random numbers for plaintext and cipher keys. The plain text is converted into the cipher text in the encryption process and then in the decryption process the cipher text is given as the input text and the original input data is obtained as the cipher text.

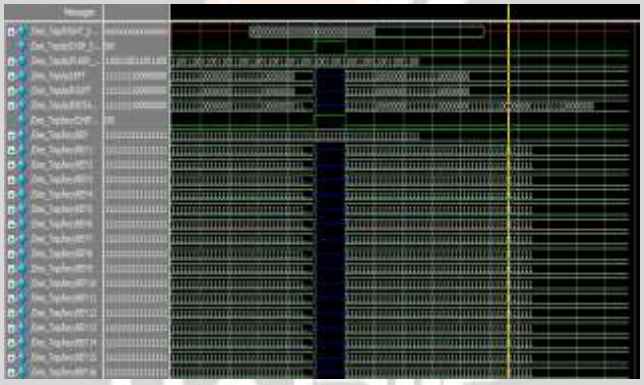

Fig -3: Encryption process

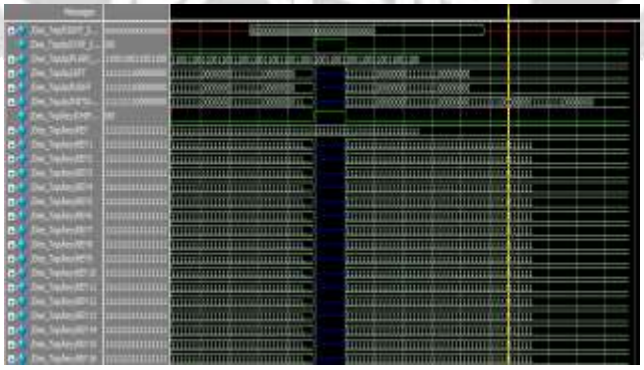

Fig -4: Key generation process

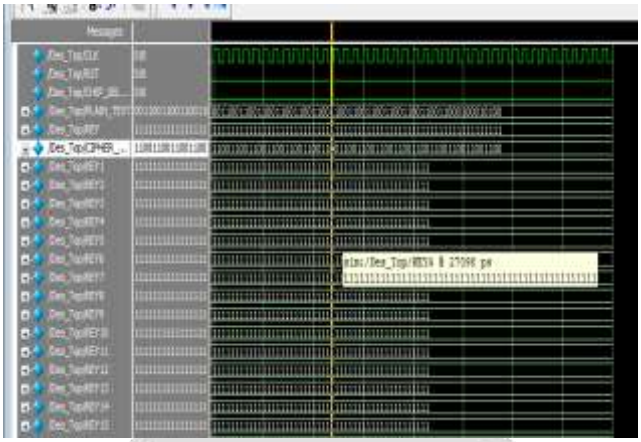

Fig -5: Decryption process

5. CONCLUSIONS

In the proposed system, a high-speed variable pseudorandom key is being generated. Encryption and decryption can also be implemented in high speed of operation up to 1 GHZ. It is also planned to design both the encryption and decryption in a mutual understanding manner so that the decryption always depends upon the operations and logics held in encrypted module. This mode of operation assures even more accuracy and secure data transfer. Implementation can be done in XILINX ISE XC9572XL low power cool runner kit can be used for hardware implementation.

6. REFERENCES

[1]. Abdel Wahab, M.M. (2015) 'High performance FPGA implementation of Data Encryption Standard', International Conference on Computing, Control, Networking, Electronics and Embedded System Engineering (ICCNEE), pp.37-40.

[2]. Arora, R and Sharma, S. (2012) 'Performance Analysis of Cryptography Algorithms', International Journal of Computer Applications, Vol 48, pp. 35-39.

[3]. Patterson, C. (2000) 'High Performance DES Encryption in Virtex™ FPGAs using JBits™', IEEE Symposium on Field-programmable custom computing machines, pp.113-121.

[4]. Rakeshkumar, S.K. (2013) 'Performance Analysis of Data Encryption Standard Algorithm & Proposed Data Encryption Standard Algorithm', International Journal of Engineering Research and Development, Vol.7, pp.11–20

[5]. W. Stallings, Cryptography and Network Security Principles and Practice, 5th ed. Prentice Hall, 2011