Implementation of Substitution Cipher on Field Programmable Gate Arrays

Ali F. Kaeib
University of Sabratah
Alikaeib@gmail.com

Osama A. S. Abourodes
Engineering Technology College
Zawia.
Abourodes.osa@gmail.com

Abstract

This paper presents algorithms implementation of substitution cipher on a FPGA “Field Programming Gate Arrays”, FPGA provides faster data rate, more flexibility to make changes to the programme and better physical security than other hardware.
The design was coded, simulated and tested by MATLAB /Simulink and Xilinx system generator and implemented on Xilinx Spartan 3A DSP XC3SD3400A -4CSG84C hardware implanted and test by using Xilinx ISE 12.4.

Index Terms Decryption, Encryption, FPGA, MATLAB, Substitution Cipher, Xilinx.

1. INTRODUCTION

Communications in the past were dependent mainly on traditional letters, payments were mainly made in cash or by checks, and confidential documents were sealed and stored in boxes. Traditional paper systems have been subject to development for a long time in parallel with suitable regulations that maintain their security and reliability.

Nowadays, all this has been changed dramatically. Large numbers of people have switched to new and modern methods of communications such as emails; this is due to their speed and cost efficiency in the communication process. “More people prefer to make their payments in electronically via the internet” [1].
Such trends save time and effort for users; however, they tend to cause security risks to their users at the same time. Therefore, a secure infrastructure is essential to deal with changes in the rapid development of technology of electronic communication systems [2].
The cryptography technology is as old as written language itself. The term “cryptography” originated from Greek roots, meaning "hidden word"; it is used in describing the earliest science of secret communications.
Until recently, such technology was mostly used by governments to protect diplomatic and military communications. Today, cryptography plays an important role in ensuring information and communications systems are secure [3].
Cryptography is a mathematical tool which security engineers use to protect data from manipulation or illegal access. Cryptography provides security personnel with the necessary utility to cover data, control access to it, authenticate its integrity, and estimate the time and cost of breaching security. Like any other services, security comes with a cost; hence it requires time, money and effort to implement cryptography tasks.
Key (asymmetric) and private (symmetric) key protocols are the most popular types of cryptographic protocols. Both communication partners use a common key in private key protocols, the encryption and decryption purposes both using this same key. Among them are the AES “Advanced Encryption Standard”, IDEA “International Data Encryption Algorithm”, and DES “Data Encryption Standard”.
The systems mentioned provide a high speed, which have a big disadvantage; this being that, for each participant, a common key has to be created. Public key protocols contain two keys; the first is left

confidential (private key) and is used for either decryption (confidentiality) or encryption (signature) of messages. The reverse operation done by the other key, (public key), is published.

RSA (which stands for Rivest, Shamir and Adleman, who were the first to describe it publicly), Elliptic curve cryptography (ECC) are examples of public key and ElGamal, DSS (Digital Signature Standard) systems. Although these systems are not as fast as symmetric systems, they offer very high security levels and do not require the presence of the initial private key exchange. The two are used in real life applications. The algorithm of the public key establishes first a common private key over an insecure channel. Then the system of symmetric formed can be used for a more secure communication with high throughput [2].

2. Introduction to Cryptography

Encryption is a technique used to transform data, referred to as clear-text or plaintext, into a structure which appears unreadable and unsystematic, with this form known by the name of cipher-text. Plaintext comes in an understandable form for either a computer (executable code) or a person (a document). As soon as it is not changed into cipher-text, neither the machine nor the person can correctly process it unless it is decrypted. This provides the ability to transmit classified information through insecure channels with no unauthorized exposure.

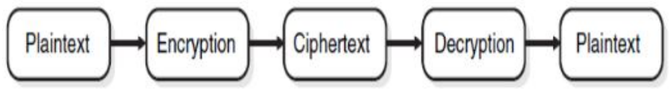


Figure 1: Encryption and Decryption

Fig 1 illustrates the process of transforming plaintext into cipher-text by encryption and vice versa, (cipher-text into plaintext) by the decryption process.

A cryptosystem is a system that can provide decryption; it can be created either by program codes or via hardware. Encryption algorithms are used in a cryptosystem and control the complexity of the process. The majority of algorithms come as complex mathematical formulas which are functional in a particular cycle and applied to the plaintext. In most encryption methods a key is used which is a secret value (commonly a long series of bits), which operates alongside the algorithm in encrypting and decrypting the text. Confidentiality, authenticity and integrity services can be provided by cryptosystems. They do not provide the availableness of systems or data. Confidentiality denotes that only authorised parties have access to information. Authenticity refers to certifying the message's source to maintain proper identification of the sender. Integrity ensures that no modification has been done to the message at any point of transmission, deliberately or by accident. [5].
Symmetric and Asymmetric Cryptography (asymmetric) Key and private (symmetric) key protocols are the most popular types of cryptographic protocols. Both communication partners use a common key in private key protocols, this key being used for both decryption and encryption purposes. Among them are DES "Data Encryption Standard" and IDEA "International Data Encryption Algorithm".

3. Symmetric Cryptography

In a cryptosystem that employs symmetric cryptography, the same key of encryption will be used for decryption and encryption by both parties, as shown in Figure 2. This provides a functionality mode which is dual. Symmetric keys can also be referred to as secret keys because this type of encryption depends on both users keeping the key a secret and also protected. Once an intruder lays hands on this key, he will have the ability to decrypt all messages that have been encrypted with this key which he intercepted. Any two users who intend to use symmetric key encryption for exchanging data must have a restricted set of keys for only both of them. For example if persons (A) and (B) want to communicate using a symmetric encryption, both of them must obtain of a copy of the similar keys. Also assuming person A wants to communicate with another person (person C) using symmetric, a second key which is different from the first key has to be obtained.

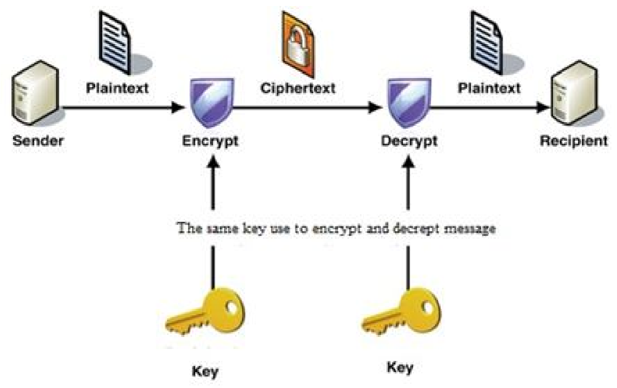


Figure خطأ! لا يوجد نص من النمط المعين في المستند.2: Symmetric cryptography system

This may not seem a complicated matter until person (A) needs to communicate for a longer period with a larger number of people: the task of keeping track of each correspondent and linking him with the correct key will become a complicated task [1].
The symmetric method's security depends completely on the user's ability in protecting the key. If a key is compromised, the intruder can easily decrypt and read all messages that have been encrypted by this key.
Symmetric systems can only provide confidentiality; they cannot provide authentication. If two people are using the same key, it is not possible to prove which one of them sent the message [5].
One may ask, are symmetric cryptosystems used if they have all these disadvantages?
It can be argued that symmetric algorithms are much faster than

asymmetric algorithms. Large amounts of data can be encrypted and decrypted in a short time, while if an asymmetric algorithm is used to encrypt or decrypt the same amount of data, it will take an unacceptable amount of time. Additionally, uncovering data encrypted by a large key size in a symmetric algorithm is a very difficult task. [1]. The strengths and weaknesses of symmetric key systems can be summarized as follows:

Strengths:

1. Symmetric systems are faster than asymmetric systems
2. They are usually difficult to work out when using a large key size

Weaknesses:

1. Key distribution: in order to properly deliver keys, it requires a secure mechanism.
2. Scalability: a unique pair of keys is required by each pair of users, which allows for an exponential growth in the number of keys.
3. Limited security: confidentiality is provided, but not authenticity or non-repudiation.

The following are some examples of symmetric key cryptography algorithms:

- IDEA "International Data Encryption Algorithm"
- RC4, RC5 and RC6
- Blowfish
- DES "Data Encryption Standard"
- 3 DES "Triple DES"

4. Asymmetric Cryptography

One single key is used between entities in symmetric key cryptography, while a different key is used for each entity in public key systems: these are called asymmetric keys. Both keys are related mathematically. Assuming one key is used in a message to encrypt it, the other key is necessary for decrypting. In public key systems a pair

of keys consists of a private key and a public key. Only the owner can know the private key, whereas the public key can be known by everybody.
In many cases the public keys can be listed in email databases and directories in order to be available for anyone who wishes to use them for encrypting or decrypting messages while communicating with another person. Figure 3 gives a clear illustration of an asymmetric cryptosystem.

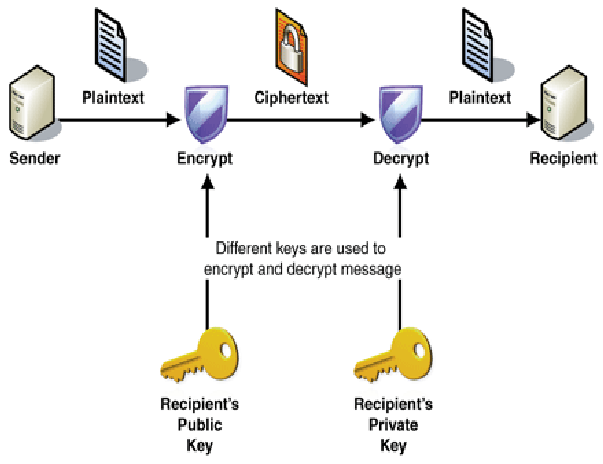


Figure 3 Asymmetric cryptosystem

In the figure above two people are involved in the asymmetric cryptosystem (person A is female and person B is male). In such a system both keys are mathematically related, but neither of them can be derived from the other. This means that once an intruder gains a copy of person B's public key he cannot know B's private key to read messages encrypted by using B' private key: the intruder can only decrypt this message if he has a copy of the private key [5].
The receiver can decrypt B's message with an intention to reply to B in a form which is encrypted. All that needs to be done is for her to encrypt the reply with B's public key, to which B can decrypt the message with his private key. In the case where asymmetric key card

encryption technology is used, encrypting and decrypting are not possible when the exact same key is used. A message can be encrypted by B using his private key while the receiver can then use B's public key to decrypt it. In order for the receiver to be sure that the message was sent from B, the message has to be decrypted with B's public key. Encrypting a message with the corresponding private key is the only way it can be decrypted on the other end with a public key. This also helps in providing authentication, the reason being that B is the only person who should have his private key. Encrypting her response with B's public key is probably the only way in which she can guarantee that B would be the only person reading the reply from her. B would be the only person who can decrypt the message because the necessary private key which is required to decrypt the message is known only by him. On the other hand, the receiver using her own private key can encrypt her response instead of using B's public key. What is the point of her doing this? This makes it known to B that the message being sent was from her. Encrypting her response using B's public key does not guarantee the authenticity, because anyone could have gotten hold of B's public key.
Using her private key to encrypt the message allows for B to be sure that no one else but her sent the message. There is no authenticity produced by symmetric keys because at both ends, the same key is normally used. In the case where one of the secret keys is used, this does not specify that the message came from a specific entity. Encrypting the file with the receiver's public key is the only means to guarantee confidentiality if this is the most important factor for the sender [1].
When a message can only be decrypted by the person who has the matching private key, this is often called a secure message format. An open message format is when the message is encrypted with the sender's private key for which confidentiality is not ensured.
Encrypting a message with a sender's private key and also encrypting it once more with the receiver's public key would guarantee a

message to be secure and in a signed format. In this case, the receiver would have to use his own private key to decrypt the message and also use the sender's public key to decrypt it again. This ensures that the message delivered is confidential and authentic.

It should be noted here that the public key is not used solely for encryption purposes while the private key's role is data decryption. Both keys have the capability of either encrypting or decrypting data. After all, data encrypted with a private key cannot be decrypted with the same private key. Any data encrypted with a private key has to be in all cases decrypted with a public key corresponding to it.

Also vice versa, data encrypted with a public key requires a corresponding private key to decrypt the data. A cryptosystem which is asymmetric normally works slower than a system which is symmetric, but this provides for authentication, confidentiality and non-repudiation depending on how the configuration is used. Asymmetric systems do not have scalability problems that symmetric systems have and they provide easier and more manageable key distribution [5].

The following are the strengths and weaknesses of asymmetric key systems:

Strengths:

1. Its key distribution is better than the symmetric system.
2. The scalability is much better than the symmetric system.
3. There is provision of non-repudiation, authentication and confidentiality.

Weaknesses:

The asymmetric key systems are not as fast as the symmetric system. Systems listed below are some examples of asymmetric key algorithms:

1. El Gamma
2. Digital Signature Standard (DSS)
3. Diffie-Hellman
4. RSA

5. Elliptic Curve Cryptosystem (ECC)

5. FPGA

Field programmable gate arrays (FPGAs) are valuable tools and can help in a number of stages of design. A prototyping model, on the other hand can be developed using a FPGA module. Developing an Application-specific integrated circuit (ASIC) chip is quite expensive due to the fact that in order to change its structure, a completely new chip is required once a design is completed. Opportunities are given to FPGAs designers in order to test the complete hardware (limitations to timing are inherent) for possible bugs and problems. There are also large and inexpensive FPGAs which have been developed recently: the design of a complete system is now possible in a single chip (SoC, or a system on chip) [6]. The figure 4 below shows the structure of the FPGA which is a structure of logic elements (LEs), which through neighboring LEs can be linked. This link is possible through the programmable interconnect which the grid of rows and columns would form throughout the chip [8].

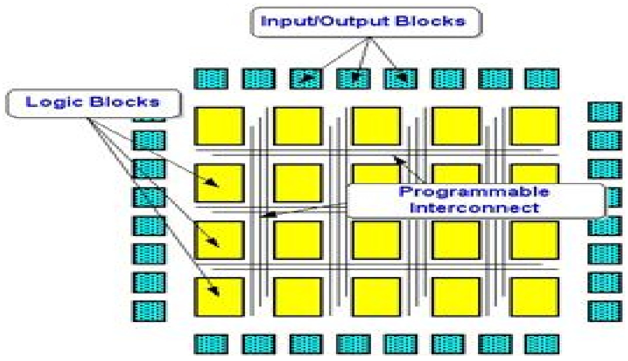


Figure 4: FPGA structure

In addition, the FPGA contains several embedded memory blocks collected in columns with each block ("MK4 block") holding 4 kbits. A number of columns of the M4K blocks might be available for each FPGA. A 4-bit look up table (LUT) is contained in each LE having a single output as shown in Fig 5, and this store 16 values which are programmable. This is similar to storing a 16 row truth table and can be used as a 4 bit input/1 bit output SRAM memory. Bypassing the flip flop ("FF"), the LUT output can give rise to an 'unregistered' value, or give a 'registered' value by entering the FF. This routing can be achieved by the multiplexers (MUXs) [7].

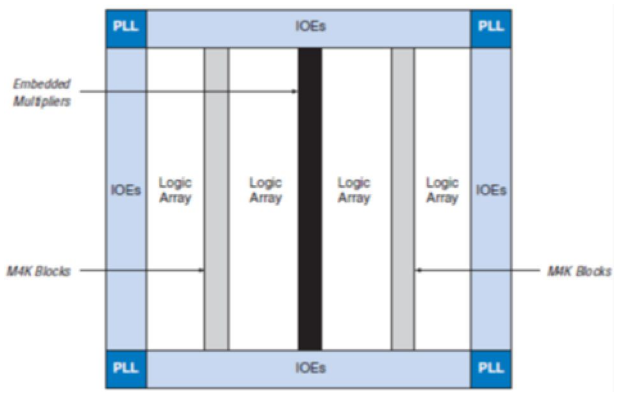


Figure 5: Embedded memory blocks on FPGA

6. Benefits of FPGA

1. Performance. FPGAs in attractive advantage of hardware parallel with exceed the computing power of DSPs "digital signal processors" by breaking the concept of sequential execution and accomplishes more per clock cycle.
2. Input and output (I/O) that are controlled at the hardware level, closely match the applications requirements by providing a faster response time and specialized functionality [8].
3. Time to market – The technology offered by FPGAs gives flexible and rapid prototyping capabilities when faced with increased time-to-market concerns. Without necessarily going through the custom ASIC designs, which is a long fabrication process, an idea or concept can be tested and also verified in hardware. This makes it possible to implement incremental

changes within hours on an FPGA design which normally would take weeks. Commercial off-the-shelf (COTS) hardware with different types of I/O is available which are already connected to a user programmable FPGA chip. The decrease in the learning curve is as a result of the growing high level software availability tools that have layers of abstraction, and do often include valuable IP cores (pre-built functions) for the control of advanced signal processing [7].

4. Reliability – Program execution of the FPGA circuitry is said to be 'hard' implementing since software tools provide for the programming environment. Processor based systems in order to share resources and schedule tasks frequently involve several layers of concept among multiple processes. The operating system manages the processors bandwidth and memory, while the hardware resource is controlled by the driver layer. At any particular time, only one instruction can be executed for any given processor core. The processor based systems by pre-empting one another are at risk of time critical tasks continually. There is minimal concern on reliability of FPGAs that do not use operating systems causing true parallel execution and deterministic hardware dedicated to every task [6].
5. Long-term maintenance – FPGA chips which are field upgradeable as mentioned earlier, compared to ASIC redesigns, are not time consuming nor expensive. For example, digital communication protocols can change overtime due to some specifications, causing maintenance and forward compatibility problems to an ASIC based interface. FPGA chips, being reconfigurable, can keep up with necessary future modifications that require to be performed. Functional enhancements can be made as a system or product matures, without time being spent in redesigning the hardware or the board layout being modified [7].

7. Substitution Cipher

This chapter explains substitution cipher and implementation on FPGA. Substitution cipher is a very old and simple cipher, and was used by Julius Caesar; the so-called “Caesar Cipher”. The algorithm used a simple shift of letters by three letters in plain text message to produce unintelligible messages. To obtain the original message simply needing to put back each letter in the unreadable message “ coded” by the letter three places to the left as shown table below [2].

Original alphabet:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Substitution cipher alphabet:

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

If the characters of the plain text alphabet and cipher-text alphabet are numbered and denoted by i and j respectively, then in the above example, for all i = 1,...,26: j = i +3 (mod 26). Mod 26 implies that the left part and right part of the equation may only differ by a multiple of 26. In a more general form, j = i + t (mod 26), in which t represents the number of characters the two alphabets are shifted.

An important characteristic of the Caesar substitution is the fact that the order of the characters of the substitution alphabet remains unchanged. The total number of keys is no more than 26, so this cipher can very easily be cracked; once a single letter of the cipher text can be related to a letter of the plaintext, the system breaks down. If the message is sufficiently large, it is all the more straightforward to find such a relation; simply note the most frequently occurring letter and the chances are that this is equal to the letter of the original plain text [9].

8. Methodology

In this paper MATLAB Simulink (containing Xilinx blocks) was used for the implementation.

8.1 Xilinx ISE

The final step is to place and route the synthesized code with Xilinx ISE. ISE means Integrated Software Environment. It provides many tools to accomplish each step of the design process from design entry to downloading the design to the FPGA. A synthesizer (XST) is part of these tools. One of the other ISE's tools is Impact, used to download the bit generated file directly to the FPGA.

8.2 Hardware

The FPGA used in this project is a Spartan (3A DSP XC3SD3400A - 4CSG84C) as shown in Figure 5 below.

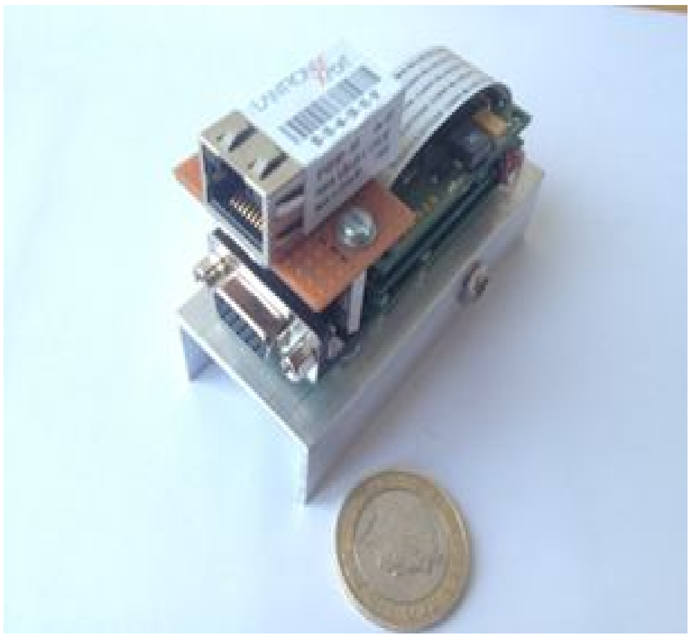


Figure 5: FPGA board

TABLE 1: Main characteristics of FPGA

Characteristic	Value
System Gates	3400.000
Logic Cells	53712
Slices	23842
Block RAM	2268 Kbit
DCM (Digital Clock Manager)	8

8.3 Design Flow

The next steps are summarized in the design flow chart below.

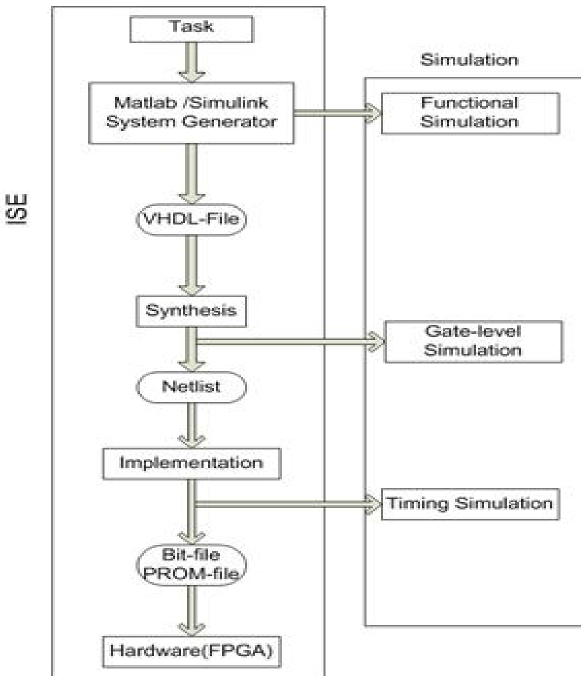


Figure 6: Design flow chart

9. Implementation of Substitution Cipher

In Substitution Cipher the design was coded and tested by MATLAB/Simulink and Xilinx system generator. For implementation on FPGA Xilinx ISE was used. MATLAB /Simulink provides a high-level view over the test environment using the system generator for (DSP) toolbox from Xilinx. In order to test the substitution cipher in Simulink text was converted to ASCII code because one cannot enter text as input in MATLAB Simulink at the end we get the result every single character shifted lift by three letters.

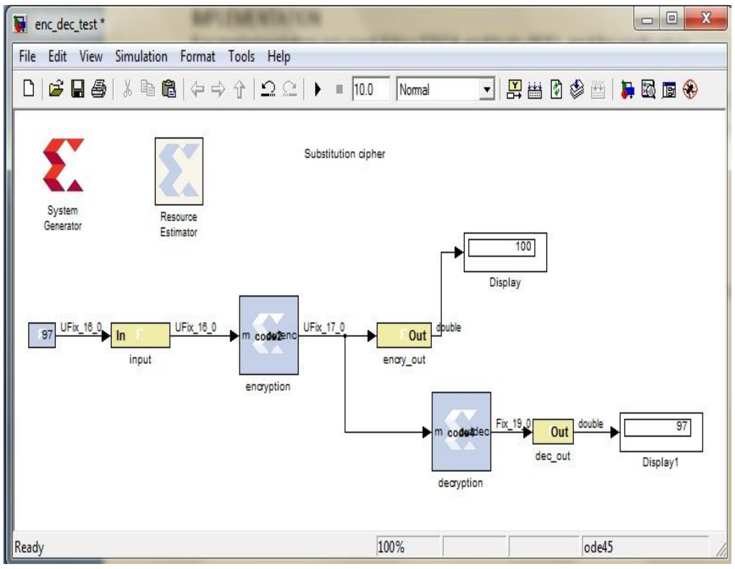
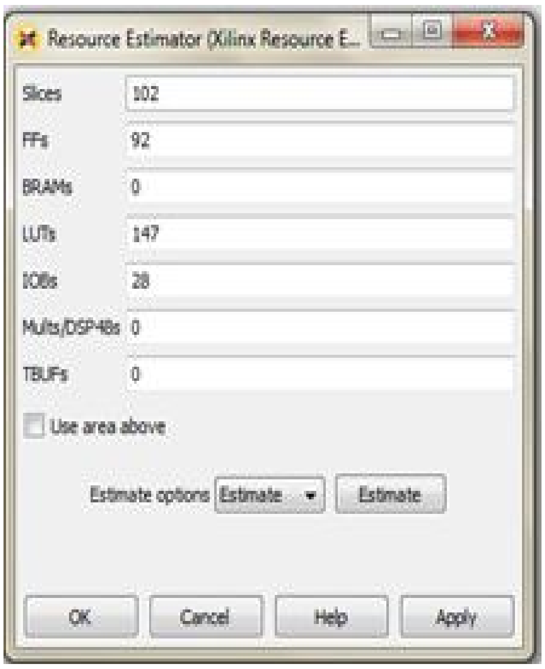


Figure 7 : MATLAB Simulink Substitution cipher

9.1 Resources Estimator

Xilinx tools provided resource estimator which is used to know how much hardware is needed for implementation, the number of hardware used depending on FPGA type.

The figure below shows us the requirement of hardware that is needed for implementing the design on FPGA.


Resource Estimator (Xilinx Resource E...

Slices	102
FFs	92
BRAMs	0
LUTs	147
IOBs	28
Mults/DSP48s	0
TBUFs	0

☐ Use area above

Estimate options: Estimate

Figure 8: Resource Estimator

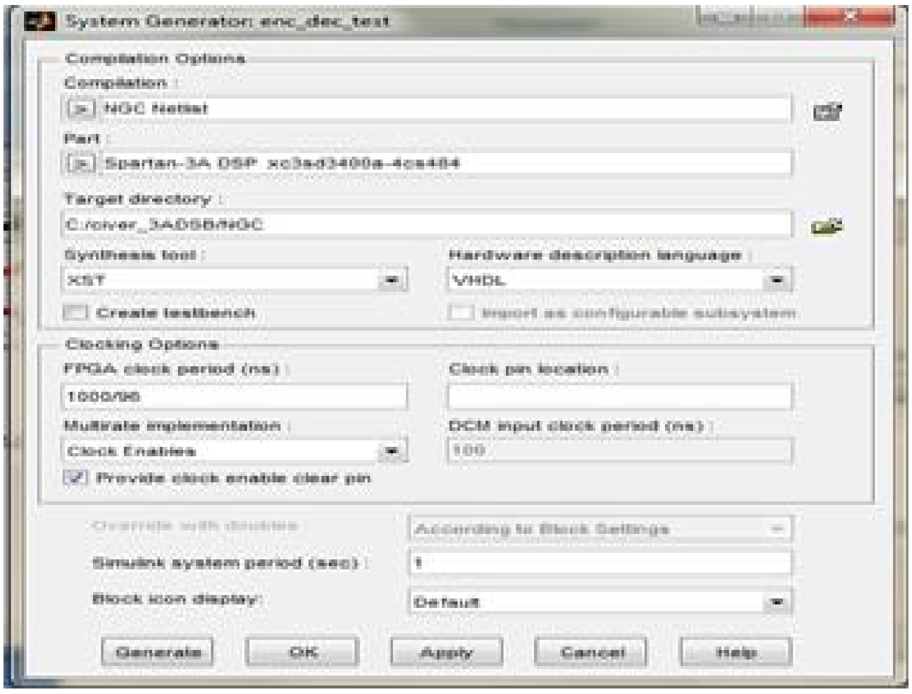
9.2 System Generator

The System Generator block provides control of the system and simulation parameters, and is used to invoke the code generator. The System Generator block is also referred to as the System Generator "token" because of its unique role in the design. Every Simulink model containing any element from the Xilinx Blockset must contain at least one System Generator block (token). "Once a System Generator block is added to a model, it is possible to specify how code generation and simulation should be handled" [10].

9.3 Xilinx ISE Process

After testing in MATLAB Simulink the system generator was run to generate NGCNitlist which converts MATLAB /Simulink design to HDL -"Hardware Description Language"- also known as synthesis presses.

Synthesis is one of the most essential steps in our design methodology because it takes the conceptual Hardware Description Language (HDL) design definition and generates the physical or logical symbol for the targeted FPGA. Synthesis is the optimisation process of adapting a logic design to the logic resources available on the chip. Then the schematic symbol shown in figure 9 was created. In this step one has to define the inputs and outputs.



System Generator: enc_dec_test

Compilation Options

Compilation : NGC Netlist

Part : Spartan-3A DSP xc3sd3400a-4cs484

Target directory : C:/civer_3ADSB/NGC

Synthesis tool : XST

Hardware description language : VHDL

☐ Create testbench

☐ Import as configurable subsystem

Clocking Options

FPGA clock period (ns) : 1000/96

Clock pin location :

Multirate implementation : Clock Enables

DCM input clock period (ns) : 100

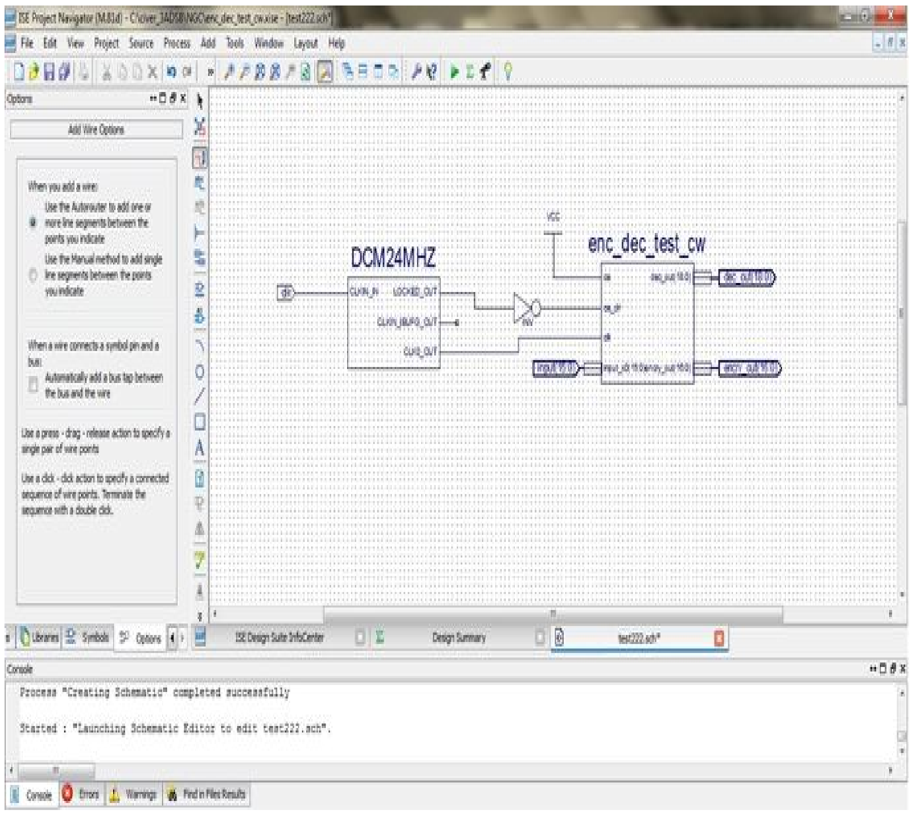
☑ Provide clock enable clear pin

Override with doubles : According to Block Settings

Simulink system period (sec) : 1

Block icon display : Default

Figure 9: System Generator (A)



When you add a wire:

(●) Use the Autorouter to add one or more line segments between the points you indicate

( ) Use the Manual method to add single line segments between the points you indicate

When a wire connects a symbol pin and a bus:

☐ Automatically add a bus tap between the bus and the wire

Use a press - drag - release action to specify a single pair of wire points

Use a click - click action to specify a connected sequence of wire points. Terminate the sequence with a double click.

```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit test222.sch".
```

Figure 9: Schematic (B)

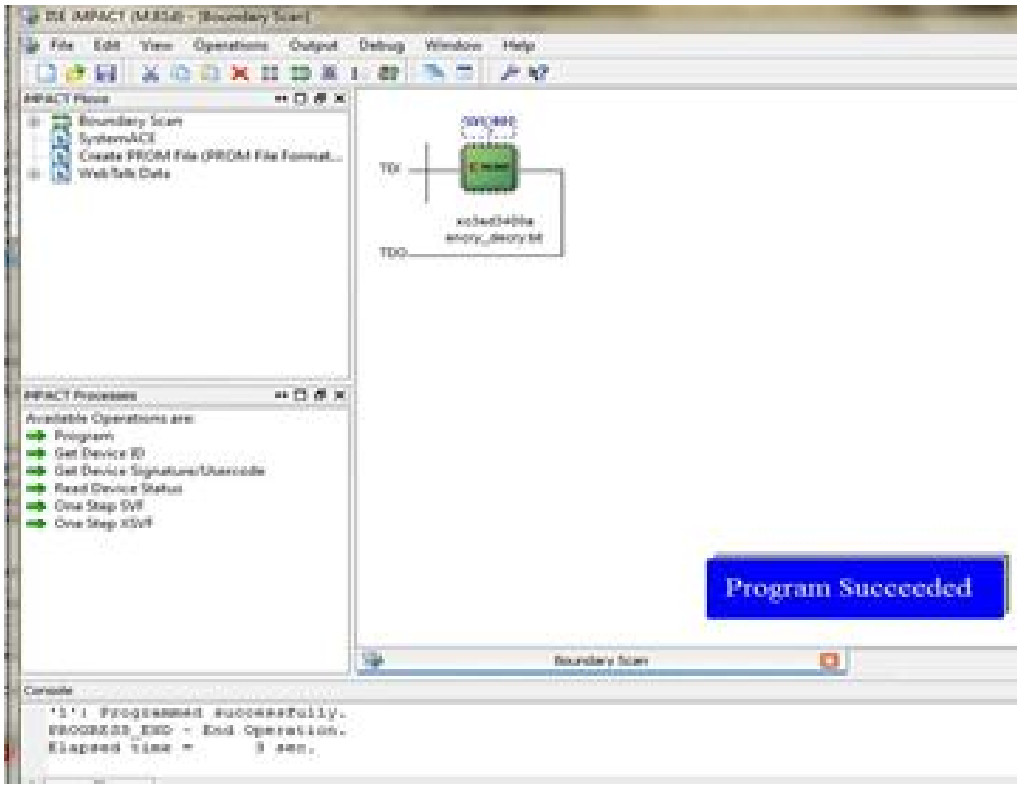
9.4 Configuration Process

This is the implementation step summarized in the steps below. These steps are the same in all implementation of FPGA:

1) Create the program file/bit file, conversion of our design in loadable bit file
2) A bitstream is generated from the physical place and route information and is transferred through cables to the target device
3) Under the start up options tab, the start-up clock can adjust in JTAG clock or CCLK.

The last step to configuration is Impact process:

1) Programming FPGA respective PROM
2) ISE IMPACT is a robust configuration tool that automatically manages everything from bitstream generation to the device download as shown in the figure below.



```
'1': Programmed successfully.
PROGRESS_END - End Operation.
Elapsed time =       3 sec.
```

Figure 10: Shows download program on FPGA succeeded (Impact process)

9.5 Results

This table below shows the results from ISE tools after implementation on FPGA.

Table 2 shows the results of implementation of the substitution cipher

enc_dec_test_cw Project Status			
Project File:	enc_dec_test_cw.xise	**Parser Errors:**	No Errors
Module Name:	encry_decry	**Implementation State:**	Programming File Generated
Target Device:	xc3sd3400a-4cs484	**Errors:**	
Product Version:	ISE 12.4	**Warnings:**	
Design Goal:	Balanced	**Routing Results:**	All Signals Completely Routed

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of4 input LUTs	**79**	**47,744**	**1%**	
Number of occupied Slices	**51**	**23,872**	**1%**	
Number of Slices containing only related logic	**51**	**51**	**100%**	
Number of Slices containing unrelated logic	**0**	**51**	**0%**	
Total Number of 4 input LUTs	**93**	**47,744**	**1%**	
Number used as logic	**79**			
Number used as a route-thru	**14**			
Number of bonded IOBs	**52**	**309**	**16%**	
Average Fanout of Non-Clock Nets	**2.24**			

The table above has been taken from ISE tools that show the type of FPGA used errors and warning in our design: no error found. The second part in the table shows the device utilization summary that means the number of hardware used in the design, and which are available in FPGA.

10. Conclusion

In this paper, we present an algorithm implementation of Substitution Cipheron on FPGAs "Field Programming Gate Arrays", because FPGA provides faster data rate, more flexibility to make changes to the program and better physical security than other hardware. Therefore the implemented design used a total number of 79 LUTs on a product version of Xilinx ISE 12.4.

References

1. Shokrollahi, J., Efficient Implementation of Elliptic Curve Cryptography 2006.
2. Buchmann, J. Introduction to cryptography. New York: Springer 2004.
3. Salomaa, A., Public Key Cryptography. Berlin: Springer 1996..
4. Schneir, B.,. Applied cryptography: protocols, algorithms, and source code in C. New York: Wiley 1996 .
5. Meyers, M. & Harris, S.. CISSP All-in-One Certification Exam2001
6. El-Maleh, A., n.d. Introduction to field Programmable Gate Array
7. National Instruments, 2012. Introduction to FPGA Technology
8. OptiMagic™, Inc, Frequently-Asked Questions (FAQ) About Programmable Logic, 2000.
9. Lubbe, J. C. A.. Basic method of cryptography. Cambridge: Cambridge University Press 1994, pp 62-84
10. Xilinx Inc. Design Tools, 2012 URL: http://www.xilinx.com/products/design-tools/ise-design-suite/index.htm