ENCRYPTION KEY GENERATION FOR DIGITAL CIRCUITS USING ANALOG CIRCUITS

A Project

Presented to the faculty of the Department of Electrical and Electronic Engineering
California State University, Sacramento
Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical and Electronic Engineering

by
Hafiza Ammara Abbas

SPRING
2016
ENCRIPTION KEY GENERATION FOR DIGITAL CIRCUITS USING ANALOG CIRCUITS

A Project

by

Hafiza Ammara Abbas

Approved by:

Dr. Fethi Belkhouche

Second Reader

Dr. Preetham Kumar

Date

05/02/16
Student: Hafiza Ammara Abbas

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

Dr. Preetham Kumar

Department of Electrical and Electronic Engineering
Abstract

of

ENCRYPTION KEY GENERATION FOR DIGITAL CIRCUITS USING ANALOG CIRCUITS

by

Hafiza Ammara Abbas

Traditionally, encryption key generation for digital systems is done using two techniques: software algorithms and hardware. The problem with both approaches is that they can be replicated because of their pseudo random nature. Hackers can use software to regenerate the encryption key by understanding and replicating the algorithm used to generate the key.

This issue can be addressed by generating a random key that does not follow any pattern or algorithm. This project is proposing a highly effective solution to generate a random encryption key for digital systems, where a hybrid approach is used. The proposed circuit combines both analog and digital parts. The analog domain consists of Chua’s circuit with a random noise source. Chua’s circuit is an assembly of two capacitors, a resistor and an inductor capable of producing an oscillating waveform and the output is always unique. The addition of noise to the circuit assures that Chua’s circuit is getting
excited by a randomized input, which creates a much more random output. Once the key is generated, the data is stored in a digital storage device.

The analog part is simulated in PSpice and the digital domain is modeled in Verilog language. The output of the circuit consists of 8-bit digital signals that can be shown in the form of waveforms. The system consists of analog circuit diagram, Verilog code, digital circuit diagram and waveforms to represent the key stored in a memory block. The final outcome of the project is a randomly generated 8-bit encryption key for digital systems.

Dr. Fethi Belkhouche

05/02/2016

Date
Acknowledgements

The support of my advisor Prof. Fethi Belkhouche cannot go unmentioned, as he has proved to be a consistent source of support and encouragement, and has inspired me to go beyond and work harder. Without his encouragement, I would never have undertaken this task. I am thankful to him for providing me an opportunity to undertake such a project and enhance my engineering skills. I express my gratitude to him for his help and patience, and for guiding me and taking out time to explain and answer my queries.

I am obliged to my caring, loving, and supportive husband, Khurram Akbar. Thank you for inspiring me and pushing me beyond my comfort zone. You are my rock and I know you will be by my side whenever I need you. Finally, to my mother Shazia Abbas I am indebted to you for showing faith in me when I doubted myself.
TABLE OF CONTENTS

| Acknowledgements                                                                 | vii |
| List of figures                                                                 | x   |
| Chapter                                                                       |
| 1. INTRODUCTION                                                               | 1   |
| 1.1 Overview                                                                  | 1   |
| 2. HISTORY                                                                   | 4   |
| 2.1 Overview                                                                  | 4   |
| 3. ENCRYPTION KEY GENERATOR                                                  | 9   |
| 3.1 Overview                                                                  | 9   |
| 3.2 Analog circuit                                                            | 10  |
| 3.2.1 Chua’s circuit                                                          | 11  |
| 3.2.2 Noise generator                                                         | 13  |
| 3.3 Digital circuit                                                           | 16  |
| 3.3.1 Analog to digital converter                                             | 16  |
| 3.3.2 Digital to binary converter                                             | 17  |
| 4. MODELING OF ENCRYPTION KEY GENERATOR AND RESULTS                           | 20  |
| 4.1 Overview                                                                  | 20  |
| 4.1.1 Analog modeling                                                         | 20  |
| 4.1.2 Analog output                                                           | 23  |
4.1.3 Digital modeling ................................................................. 24
4.1.4 Digital output ................................................................. 27
4.2 Data analysis ................................................................. 28
5. ENHANCEMENTS AND FUTURE WORK ................................ 32
  5.1 Overview .................................................................. 32
    5.1.1 Varying the key length ........................................ 32
    5.1.2 Varying sampling time ........................................ 32
6. CONCLUSION ................................................................. 33
Appendix .................................................................................. 35
References .................................................................................. 42
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Ancient rock used to encrypt secret message</td>
<td>5</td>
</tr>
<tr>
<td>2. Caesar cipher disk</td>
<td>6</td>
</tr>
<tr>
<td>3. Bi-literal alphabet</td>
<td>6</td>
</tr>
<tr>
<td>4. The Enigma machine</td>
<td>7</td>
</tr>
<tr>
<td>5. Analog circuit block diagram</td>
<td>9</td>
</tr>
<tr>
<td>6. Digital circuit block diagram</td>
<td>10</td>
</tr>
<tr>
<td>7. Chua's circuit</td>
<td>11</td>
</tr>
<tr>
<td>8. Graph of burst noise</td>
<td>14</td>
</tr>
<tr>
<td>9. Thermal noise</td>
<td>15</td>
</tr>
<tr>
<td>10. The input to A/D converter</td>
<td>17</td>
</tr>
<tr>
<td>11. Block diagram of the whole circuit</td>
<td>19</td>
</tr>
<tr>
<td>12. Schematic diagram used to generate first bit of random key</td>
<td>21</td>
</tr>
<tr>
<td>13. Schematic diagram used to generate 8-bit random key</td>
<td>22</td>
</tr>
<tr>
<td>14. Output of the circuit for one bit (bottom to top)</td>
<td>23</td>
</tr>
<tr>
<td>15. Top module of digital domain</td>
<td>24</td>
</tr>
<tr>
<td>16. Internal connection of digital circuit</td>
<td>26</td>
</tr>
<tr>
<td>17. Internal connection</td>
<td>27</td>
</tr>
<tr>
<td>18. Digital simulation output</td>
<td>28</td>
</tr>
<tr>
<td>19. Digital simulation output converted radix</td>
<td>28</td>
</tr>
</tbody>
</table>
20. Bar chart for random keys .................................................................29
21. Scatter chart for random keys ..........................................................30
22. Pie chart for random keys ...............................................................31
Chapter 1

INTRODUCTION

1.1 Overview

Humans have devised many ways to protect confidential information from unauthorized users. To send a secret letter in ancient Greece, people wrote messages on wood that was covered with wax and bore an innocent message [3]. Similarly, today in our digital world, we need to hide confidential information. In order to achieve this, we also try to design a "covering" for our secret data. This process is known as encryption. Generally encryption in digital systems is done by using hardware or software techniques which generate an encryption key to encrypt the data. These techniques usually involve algorithms that generate a random key. Unfortunately, any algorithm can have randomness up to some extent [2]. After a number of iterations, the algorithm will be repeated, this gives an opportunity for a hacker to steal secret information and regenerate the encryption key.

Most techniques used to generate an encryption key are pseudo random. They run on a particular algorithm, therefore it is possible to regenerate the key by understanding the repeated pattern in the encryption key or by using reverse engineering techniques. The solution to this problem is to generate an encryption key that does not follow any algorithm and cannot be reproduced using reverse engineering techniques. This project details a
technique to achieve this goal. The encryption key generation methodology proposed in this project will not use any algorithm, therefore no pattern repetition can be seen in the encryption key.

This project will present a unique way to generate the encryption key for digital systems. The key generation will be done using a hybrid approach which will involve an analog circuit attached to a digital circuit. The analog circuit will have Chua’s circuit in addition to a small chain of inverters. Chua’s circuit is a simple assembly of electrical components that will produce oscillations such that the output is never repeated. The Chua’s circuit has some requirements. It should have one resistor, one nonlinear component and three or more energy storage elements.

In order to introduce more randomness in the encryption key, a noise signal will be added to the circuit. The modeling of the noise signal is done using a voltage device that will mimic the noise signals present in digital systems. This technique will create randomness in the encryption key so that tracing it back will be almost impossible, as there is no perfect algorithm to generate the encryption key. The noise signal will keep changing with the atmospheric changes such as temperature, cross signals etc. It is almost impossible to regenerate the encryption key even by the same manufacturer. Assuming the key generator is designed again in the same lab using the same circuit design, the designer has no control over the heat noise and other factors in the environment used to generate that particular key.
Once the encryption key is generated, it is important to translate it to a proper digital format. The encryption key will be generated in the analog domain, however it will be used in digital circuits. That is why the analog signal is translated into a digital signal. An analog to digital converter will do this job. This digital convertor is designed to comply with the digital and analog circuit requirements in terms of speed and voltage levels.

Once the formatting is completed, a memory array is used to store the encryption keys. Now the encryption key is stored in a digital component and ready to be used by any digital system to encrypt the digital data. Signal conversion, formatting, and storage are all done in the digital domain. This hybrid approach will help produce a very random encryption key that cannot be regenerated using any algorithm or reverse engineering.
Chapter 2

HISTORY

2.1 Overview

The word encryption comes from the Greek word “kryptos”, meaning hidden or secret. The use of encryption is nearly as old as the art of communication itself. In ancient age, people used different techniques to keep their secret message encrypted, such as invisible inks, embedding the message in a wooden box, animal skin etc. In the middle age, encryption became more advanced where polyalphabetic substitution came into picture. This technique uses multiple swaps of alphabets to limit the use of frequency analysis to crack a cipher text. The Enigma electro-mechanical rotor cipher machine used by the Germans during World War II is the perfect example of polyalphabetic encryption. More complex techniques like public or private key exchange methods and RSA (Rivest-Shamir-Adleman) algorithms are being used today. In the past governments were the only real users of encryption. However, with time, encryption became a norm to protect commercial and even personal data. All banks, flash devices and networks use encryption techniques to make their system secure and robust.
The first known evidence of the use of cryptography was attributed to the Egyptians. Their technique consisted of using a carved rock [3] with hieroglyphic symbols carved for simple words. There was a key rock that had all the corresponding letters to decrypt the messages as shown in figure 1.

![Figure 1 Ancient rock used to encrypt secret message [1a]](image)

This is the earliest known form of encryption. At that time pottery was considered an art and encryption was used to encrypt methods to make glazed pottery [1]. Encryption became more advanced with time. The Romans came up with the Caesar cipher in 100-44.
BC [3]. Julius Caesar used a simple method to shuffle the letters to write a secret message.

It was used for government communication. The Caesar cipher is shown in figure 2.

![Figure 2: Caesar cipher disk](image)

Figure 2 Caesar cipher disk [2a]

In 1623 a bi-literal cipher that consisted of a 5-bit binary encoding device was used to encrypt data. This concept was used to generate tri-literal cipher encoding [3].

Figure 3 shows the bi-literal alphabet.

![Figure 3: Bi-literal alphabet](image)

**Example 3. Of a Bi-literal Alphabet.**

| Aaaa, aaab, aaab, aabaa, aabab, aabba, ababa, ababb, abbaa, abbba, abbb, abba, baba, babaa, babab, babba, babbb |
| A, B, C, D, E, F, g, h, i, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z |

Figure 3 bi-literal alphabet [3a]
There were significant improvements in cryptography techniques in eightieth century. In 1933, the Enigma machine shown in figure 4 was used for secret communication.

Figure 4 The Enigma machine [4a]

With time cryptography became stronger and more complex. But the concept was the same: random number generation. In 1977 there was another improvement in the field. Cryptography was divided into two types: Symmetric and Asymmetric [3]. In symmetric cryptography, the key used to encrypt and decrypt the data is the same however in asymmetric cryptography the encryption and decryption keys are different but linked mathematically. This was a huge turning point in the field of cryptography. Now
with the new technology, longer cipher keys can be generated and complex algorithms are used to achieve high level of randomness to generate cipher key. Cryptography is still an evolving field which needs more and more ideas to generate more robustness and security in the system.
Chapter 3

ENCRYPTION KEY GENERATOR

3.1 Overview

This chapter details the design and implementation of the encryption key generator. The encryption key generator has the ability to generate a random key that does not follow a fixed algorithm. The randomness of the key depends on environmental factors like temperature, noise and crosstalk signals etc. The encryption key is generated to encrypt signals in digital systems. In this project, the generator has two parts: an analog and a digital circuit. The analog circuit consists of Chua’s oscillator and a noise generator. The addition of the output of these two circuits will produce a random signal. Figure 5 presents the block level implementation of the analog part of the encryption key generator.

Figure 5 Analog circuit block diagram

The output of the circuit above is a random analog signal that cannot be directly fed to a digital circuit. An analog to digital converter is used to convert the signal so it can be
used as an input to the digital system. Sampling is done in the analog to digital converter which produces a digital output. This signal is fed to the digital circuit which will convert the digital signal to a binary signal. Once the binary signal is generated, it will be stored in a register. Figure 6 shows the higher level implementation of the digital part of the encryption key generator.

![Digital circuit block diagram](image)

Figure 6 Digital circuit block diagram

The next part of this chapter will describe each block of the analog and digital circuit in more detail.

3.2 Analog circuit:
The analog part of the circuit has two main parts:

- Chua's circuit
- Noise generator

Adding the output of both will generate a highly random signal that does not follow any symmetric pattern or algorithm that can be traced back by a hacker.
3.2.1 Chua’s circuit:

Chua’s circuit was invented in the fall of 1983 (Chua, 1992) [9]. It has been built and used in many laboratories as a physical source of pseudo random signals [9]. It is a special type of oscillator that generates oscillations such that the output is almost never repeated. Chua’s circuit has three standard components that are assembled to generate non-repeating output. The components are as follows:

- One active resistors
- Three energy storage elements.
- One non-linear element like diode.

![Figure 7 Chua's circuit. [5a]](image)
Component $N_R$ is a nonlinear element known as Chua's diode. Chua's circuit is the simplest electronic circuit meeting these criteria [4]. As shown figure 7, the energy storage elements are two capacitors (labeled $C_1$ and $C_2$) and an inductor (labeled L)[4]. An active resistor is a device that has a negative resistance and the ability to amplify, providing the power to generate the oscillating current. The locally active resistor and nonlinearity are combined in the nonlinear element $N_R$. The circuit shown in figure 7 shows one common implementation of Chua's circuit.

Using Kirchhoff's circuit laws, we get the following equations [4]:

$$RC_2 \frac{dy}{dt} = x - y + Rz$$

$$\frac{dz}{dt} = -\beta y$$

$$\frac{dx}{dt} = \alpha[y - x - f(x)]$$

where $C_1$ and $C_2$ are capacitors, and the electric current in the inductor L. Function $f(x)$ describes the electrical response of the nonlinear resistor, its shape depends on the particular configuration of the components in the circuit. The parameters $\alpha$ and $\beta$ are determined by the particular values of the circuit components [4].
A very specific Chua’s circuit is designed for this project to meet the requirements for encryption key generator design. The output voltage was achieved by using a specific type of diode and certain values of the capacitor, inductor and resistors. The next chapter will discuss the values and their significance.

3.2.2 Noise generator

A very specific noise generator is designed for the project. The noise signal characteristics such as the amplitude are designed to mimic the real noise present in digital systems. There are two major types of digital noise signals that are considered to design the noise generator: burst noise and thermal noise [5].

- Burst noise

Burst noise is a type of electronic noise that occurs in digital systems. It is also called popcorn noise because it is a sudden bursting sound like popcorn [6]. It is also called non-Gaussian noise. It is a random signal with steps that may be large or small. It is highly unpredictable in nature. This type of noise may have multiple sources in one circuit, however the most common source is the diode point of contact with the circuit. It impacts the performance of the digital systems, but for our circuit, we will be taking advantage of its randomness by adding it to the output of Chua’s circuit. Hence producing huge amount of randomness in the generated encryption key [6]. Figure 8 shows the density graph of a burst noise.
Thermal noise is another major type of noise present in the digital systems. It is also called Johnson–Nyquist noise. This noise is caused by the kinetic energy and thermal agitation of electrons in the digital systems [7]. This type of noise is always present in the system regardless of the input voltage [7]. This is a random signal that usually looks like small bell curves. The zoomed graph represents small Gaussian curves. Figure 9 is a graphical presentation of this type of noise.
To mimic the multiple types of noise present in the digital system, a voltage source is designed in PSpice to generate the output which is the combination of burst and thermal noise. We are able to generate the input of the voltage source as a text file. This file is generated using software techniques to mimic random noise in the digital system and manually induce burst signal. Gaussian noise signal is then added to burst noise. Ten Gaussian signals are generated in one Nano second with new amplitude and random deviation. Once we have these two signals, they are added and injected in the output of Chua’s circuit designed in the previous stage. At this point we have a very random output ready to be injected to the next stage of the circuit. The next chapter will show the simulation results and the output of the noise generator used to simulate the circuit. It will
also show how the random output is generated by adding two signals one coming from the Chua's circuit and other from the noise generator. This random signal is an analog signal that will be converted to digital signal and fed to the digital circuit.

3.3 Digital circuit

The digital domain has two major circuits, one is the analog to digital converter and the other is digital to binary converter. The input to the digital circuit will be an analog signal that will be sampled in order to be converted to a digital signal. This digital random value is stored in a memory unit. This will be fed to the binary converter. The binary converter will have a threshold value to convert the output into zeros and ones and store them in an output register.

3.3.1 Analog to digital converter

An analog to digital converter is designed with a fixed sampling rate to sample the input analog signal such that adjacent samples have distinct values. The input analog signal has a varying time period and the sampling rate is chosen to make sure that enough time is given to the analog signal to propagate. Hence, the adjacent values are distinct after sampling. Figure 10 illustrates the input and output of the analog to digital converter. However in our circuit the input is not a smooth sinusoidal signal but a highly randomized signal and the output consists of random samples.
Figure 10 Input to A/D converter looks like analog signal and output looks like the discrete time signal. [8a]

3.3.2 Digital to binary converter

The input to the digital to binary converter is the output from the analog to digital converter. The input is a set of discrete random numbers stored in a register. This converter has a built-in threshold value stored in a register. A comparator is designed to compare with the incoming stream of values, and generate the output as zeros and ones. Hence generating random binary bit stream.

This whole process mentioned from the start of chapter 3 constitutes one bit of random values. This project is designed to generate 8-bit encryption keys and hence eight
parallel circuits will be running to generate eight random bits that constitute one 8-bit random number to encrypt digital data.

Figure 11 shows the complete diagram for the 8-bit random encryption key generation chip.
Figure 11 Block diagram of the whole circuit.
Chapter 4

MODELING OF ENCRYPTION KEY GENERATOR AND RESULTS

4.1 Overview

This chapter covers the schematics and diagrams, code in Verilog and test bench used to show a working encryption key generator model. Modeling refers to representing the functionality in schematics for analog circuits and Verilog presentation of the digital part of the circuit.

4.1.1 Analog modeling

The analog design methodology is done in two steps: designing the circuit for the first bit of the random key and then creating similar instances running in parallel. The schematic diagram in figure 12 shows the analog circuit used to generate the first bit of the random key.
Figure 12 Schematic diagram used to generate first bit of random key

The schematic diagram has two circuits: Chua’s circuit and a noise generator. Chua’s circuit is designed using two capacitors, an inductor, a resistor and two diodes. One diode is operating in reverse bias mode while the other is operating in forward bias using a 9V battery. Chua’s circuit has a particular operating point called the Chua’s point. There is a limited range of resistor values that can be used to produce Chua’s effect. In the schematic of figure 2, the range for $R_2$ is 1100 ohms to 1200 ohms. Chua’s circuit is attached to an op-amp from the LM700 family. This op-amp was selected because it is the closest to industry standards. The input required to activate the op-amp is matching the voltage source used for the rest of the circuit. This op-amp is used in a non-inverting amplifier configuration. It helps to amplify the oscillations generated by the Chua’s oscillators.
The noise generator mimics the burst and thermal noise in an analog system. It takes a text file as an input. The input file is made using excel. The excel file has time and voltage values generated as a sum of Gaussian and non-Gaussian noise. The appendix has all input files used to generate the noise. There will be eight text files in total. Each file has unique contact and is used to simulate a different circuit.

The project intent is to generate 8-bit random key so eight parallel circuits were designed to simulate the results. Figure 13 presents the eight circuits running in parallel.

Figure 13 Schematic diagram used to generate 8-bit random key
4.1.2 Analog output

The outputs of the schematic shown in figure 14 are eight Chua's signal and eight noise generators. The noise generator output is added to the Chua's output and a highly random signal is obtained. Figure 14 shows the output for the first bit of the key. The output diagram has three plots. The bottom plot represents the output of the noise generator. The middle plot is the output from Chua's circuit. The top plot is the sum of the noise generator and the Chua's circuit output. The appendix has outputs for all eight circuits shown in figure 14 and the output files are fed to the digital circuits.

Figure 14 Output of noise generator, Chua's circuit and final output of the circuit for one bit (bottom to top)
4.1.3 Digital modeling

The digital model of the circuit consists of analog to digital converters, digital to binary converters and a digital storage unit. The design methodology used in Verilog is a bottom-up methodology. In this methodology, the leaf component is designed and instantiated multiple times in the design hierarchy. The blocks are then inter-connected in the top module. Figure 15 shows the inputs and outputs of the top module.

Figure 15. Top module of digital domain.
The input consists of reset and clock signal named as “clk” in figure 15. The output is a digital signal that consists of 8-bit encryption key. Figure 15 shows the internal diagram of the digital circuit. There are eight text files read internally which were generated by the analog circuit. In reality the two circuits will be connected together. The analog signal is fed to eight parallel analog to digital converters and then fed to eight digital to binary converters. The outputs of the eight digital to binary converters are fed to eight bit array which constitutes the random encryption key.
Figure 16 Internal connection of digital circuit
Figure 17 shows the zoomed version of one of the analog to digital converters attached to digital to binary converter.

Figure 17 Internal connection of the analog to digital converter attached to digital to binary converter.

4.1.4 Digital output

A Verilog test bench was written to simulate the circuit. Clock and reset signals were tested in the test bench. The digital output was seen as expected and consisted of 8-bit array of random numbers in binary format. This simulation was generated for one hundred encryption keys for a digital system. The results can be seen in figure 18. To show the results clearly the altering internal signals are also pulled out.
Figure 18 Digital simulation output

Figure 19 shows output signal in signed decimal format. The encryption keys can be seen in figure 19.

Figure 19 digital simulation output converted radix

4.2 Data analysis

Data analysis was performed to verify the results. The randomness in encryption was the focus of this project. The results plotted in different form of charts clearly show that the key generated was highly randomized. The test bench was designed to generate
hundreds of encryption keys. The keys were used to make the bar chart shown in figure 20. There were one hundred inputs, and the output consisted of distinct values with a huge variation. Figure 20 is the bar chart representation of the random keys generated by the designed model.

![Bar Chart for the random key values](image)

**Figure 20** Bar chart for random keys

The scatter chart represents the values at different points and shows how much the values are random. Figure 21 is the scatter chart for the random keys generated.
Figure 21 Scatter chart for random keys

The data analysis was also done using a pie chart. The pie chart shows very clearly the variation and the distribution of one hundred distinct values proving that the data is highly randomized and does not follow a fixed pattern or signature that can be replicated.
It is clear from data analysis that the proposed key generation methodology can be used to generate highly randomized encryption keys and protect any system that requires high level of security.
Chapter 5

ENHANCEMENTS AND FUTURE WORK

5.1 Overview

Cryptography is a growing field and enhancements of the existing methods play a vital role to deal with emerging challenges. There are couple of possible enhancements for this particular project. These enhancements can be done according to the requirements of the end user which can induce more randomness in the output and make it more secure. Below are the two major enhancements that can be performed:

5.1.1 Varying the key length:

Varying the length of the key will create a huge difference. The length of the encryption key will become another factor for the hacker to explore. The length of the key can have a possible range defined by the length of the data that needs to be encrypted. This feature will add a new dimension of robustness to the encryption key.

5.1.2 Varying sampling time:

Varying sampling time is another factor that can contribute towards making an encryption key very unpredictable. The implementation can be done by picking a range of random sampling rates with respect to the oscillation frequency of the Chua's circuit. The varying sampling rate will make the time at which the key is generated very unpredictable. This feature can make the encryption key more random and secure.
Chapter 6

CONCLUSION

This project presented a new vision for encryption key generation for digital systems using a mixed signal approach. The project also highlights the importance of the noise and oscillating signals in digital systems.

Mixed signal approach was used to implement the model circuit for the encryption key generator. Phase I includes the design and simulation of Chua’s circuit. Finding the correct values of the parameters of the circuit to generate the Chua’s effect was a challenge. To overcome this hurdle, multiple equations were used by applying Kirchhoff’s laws. The components of the circuit were used so that the overall circuit is power efficient. This property makes the design very desirable for applications with low power requirements. In addition to power advantage, the circuit has a parallel bit generation mechanism, which makes the circuit very efficient.

Phase II involves designing a noise generator that can mimic the noise in a real system. Multiple types of noise equations, random number generators and software were used to achieve this target.

Phase III consists of digital circuit modeling in a hardware description language. Verilog was used to model a pipeline structure. Pipeline structure is more efficient as compared to non-pipeline models. This makes the implementation very effective for time
critical applications. Finally a test bench was written in Verilog to simulate the circuit and the expected output 8-bit encryption key was generated. The LSI 10K library was used to simulate the digital circuit at a maximum frequency of 200MHz. However using the latest libraries available in the market will improve the efficiency up to 400 MHz frequency.
Appendix

Verilog code for top module:

module top_module(clk, reset,
                  dig_out // output
                 );
//--------------Input Ports-----------------
input clk, reset;
//--------------Output Ports----------------
output [7:0] dig_out;
//--------------Internal Variables--------
wire [7:0] dig_out;
//--------------Internal Variables--------
wire [9:0] connect[0:7];
//wire d[0:7]; //--------------Code Starts Here---------------

reg_file uu0(.out_int0(connect[0]), .clk(clk), .reset(reset));
reg_file uu1(.out_int1(connect[1]), .clk(clk), .reset(reset));
reg_file uu2(.out_int2(connect[2]), .clk(clk), .reset(reset));
reg_file uu3(.out_int3(connect[3]), .clk(clk), .reset(reset));
reg_file uu4(.out_int4(connect[4]), .clk(clk), .reset(reset));
reg_file uu5(.out_int5(connect[5]), .clk(clk), .reset(reset));
reg_file uu6(.out_int6(connect[6]), .clk(clk), .reset(reset));
reg_file uu7(.out_int7(connect[7]), .clk(clk), .reset(reset));
int_in_dig_out uu8(.clk(clk), .reset(reset), .int_in(connect[0])
                  , .dig_out(dig_out[0]));
int_in_dig_out uu9(.clk(clk), .reset(reset), .int_in(connect[1])
                  , .dig_out(dig_out[1]));
int_in_dig_out uu10(.clk(clk), .reset(reset), .int_in(connect[2])
                    , .dig_out(dig_out[2]));
int_in_dig_out uu11(.clk(clk), .reset(reset), .int_in(connect[3])
                    , .dig_out(dig_out[3]));
int_in_dig_out uu12(.clk(clk), .reset(reset), .int_in(connect[4])
                    , .dig_out(dig_out[4]));
int_in_dig_out uu13(.clk(clk), .reset(reset), .int_in(connect[5])
                    , .dig_out(dig_out[5]));
int_in_dig_out uu14 (.clk(clk), .reset(reset), .int_in(connect[6])
, .dig_out(dig_out[6]) );
int_in_dig_out uu15 (.clk(clk), .reset(reset), .int_in(connect[7])
, .dig_out(dig_out[7]) );

endmodule //End Of Module

Register file for digital domain

module reg_file(out_int0, out_int1, out_int2,
out_int3, out_int4, out_int5, out_int6, out_int7, clk, reset);

//output ports
output signed [9:0] out_int0;
output signed [9:0] out_int1;
output signed [9:0] out_int2;
output signed [9:0] out_int3;
output signed [9:0] out_int4;
output signed [9:0] out_int5;
output signed [9:0] out_int6;
output signed [9:0] out_int7;

//input ports
input reset, clk;

//internal registers
reg signed [9:0] regfile0;
reg signed [9:0] regfile1;
reg signed [9:0] regfile2;
reg signed [9:0] regfile3;
reg signed [9:0] regfile4;
reg signed [9:0] regfile5;
reg signed [9:0] regfile6;
reg signed [9:0] regfile7;

//internal registers
reg signed [9:0] regfile11[0:99];
reg signed [9:0] regfile12[0:99];
reg signed [9:0] regfile13[0:99];
reg signed [9:0] regfile14[0:99];
reg signed [9:0] regfile15[0:99];
reg signed [9:0] regfile16[0:99];
```verilog
reg signed [9:0] regfile17[0:99];
reg signed [9:0] regfile18[0:99];

reg signed [9:0] out_int0;
reg signed [9:0] out_int1;
reg signed [9:0] out_int2;
reg signed [9:0] out_int3;
reg signed [9:0] out_int4;
reg signed [9:0] out_int5;
reg signed [9:0] out_int6;
reg signed [9:0] out_int7;

integer i;
always @(posedge clk or negedge reset )
begin
  if (reset == 0 )
begin
    out_int0 <= 0;
    out_int1 <= 0;
    out_int2 <= 0;
    out_int3 <= 0;
    out_int4 <= 0;
    out_int5 <= 0;
    out_int6 <= 0;
    out_int7 <= 0;
  end
  else 
    begin
      out_int0 <= regfile0;
      out_int1 <= regfile1;
      out_int2 <= regfile2;
      out_int3 <= regfile3;
      out_int4 <= regfile4;
      out_int5 <= regfile5;
      out_int6 <= regfile6;
      out_int7 <= regfile7;
    end
end
```

```
end

end

initial
begin#
$readmemb("read0.txt",regfile11);
#1 $readmemb("read1.txt",regfile12);
#1 $readmemb("read2.txt",regfile13);
#1 $readmemb("read3.txt",regfile14);
#1 $readmemb("read4.txt",regfile15);
#1 $readmemb("read5.txt",regfile16);
#1 $readmemb("read6.txt",regfile17);
#1 $readmemb("read7.txt",regfile18);
end

initial
begin
for(i=0;i<99;i=i+1)
begin
$display("Data at %0d is %d",i,regfile11[i]);
$display("Data at %0d is %d",i,regfile12[i]);
$display("Data at %0d is %d",i,regfile13[i]);
$display("Data at %0d is %d",i,regfile14[i]);
$display("Data at %0d is %d",i,regfile15[i]);
$display("Data at %0d is %d",i,regfile16[i]);
$display("Data at %0d is %d",i,regfile17[i]);
$display("Data at %0d is %d",i,regfile18[i]);
#5 regfile0 = regfile11[i];
#5 regfile1 = regfile12[i];
#5 regfile2 = regfile13[i];
#5 regfile3 = regfile14[i];
#5 regfile4 = regfile15[i];
#5 regfile5 = regfile16[i];
#5 regfile6 = regfile17[i];
#5 regfile7 = regfile18[i];
end
Digital to binary converter

module int_in_dig_out(clk, reset,
        int_in , // input
        dig_out   // output
        );
//-------Input Ports-------
input signed [9:0] int_in;
input clk, reset;

//-------Output Ports-------
output dig_out;
//-------Internal Variables-------
reg dig_out;
//-------Code Starts Here-------
always @ (posedge clk or negedge reset)
if (!reset)
    dig_out = 0;
else
    begin : convert
        if (int_in <= 0) begin
            dig_out = 0;
            end
        else
            begin
                dig_out = 1;
            end
    end
endmodule //End Of Module
Test bench to generate output

```verilog
module top_module_tb;

// Inputs
reg clk;
reg reset;

// Outputs
wire [7:0] dig_out;

// Instantiate the Unit Under Test (UUT)
top_module uut (.
  .clk(clk),
  .reset(reset),
  .dig_out(dig_out))
);

initial
  clk = 0;
```
always #2.5 clk = ~clk;

initial begin
// Initialize Inputs
reset = 0;
#10 reset = 1;

// Wait 100 ns for global reset to finish
#1000;

// Add stimulus here
end

endmodule
References


