

**IMPLEMENTATION OF SYNCHRONIZED CHAOTIC
SYSTEMS BY FIELD PROGRAMMABLE GATE
ARRAY**

**A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of**

MASTER OF SCIENCE

in Electrical and Electronics Engineering

**by
Can EROĞLU**

**October 2007
İZMİR**

We approve the thesis of **Can EROĞLU**

Prof. Dr. F. Acar SAVACI
Supervisor

Assoc. Prof. Dr. Uğur ÇAM
Committee Member

Assist. Prof. Dr. Orhan COŞKUN
Committee Member

17 October 2007
Date

Prof. Dr. F. Acar SAVACI
Head of the Department of Electrical
and Electronics Engineering

Prof. Dr. Hasan BÖKE
Dean of the Graduate School of
Engineering and Science

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Dr. F. Acar Savacı for his valuable guidance and support. I am very proud that i had the chance to work with him.

I would like to thank to the members of my Thesis Committee Assoc. Prof. Dr. Uğur Çam and Asst. Prof. Dr. Orhan Coşkun for their useful comments. I also would like to thank to Dr. Tolga Yalçın for his unfailing support.

Finally, I am deeply thankful to my parents Dudu and Zafer, and my sister Irmak for their support and endless love through my life.

ABSTRACT

IMPLEMENTATION OF SYNCHRONIZED CHAOTIC SYSTEMS BY FIELD PROGRAMMABLE GATE ARRAY

In this thesis, the geometric properties of chaotic systems are used to determine their synchronization. First, each system is constructed with MATLAB Simulink blocks. Afterwards, feedback control system for synchronization of chaotic systems is proposed by using complete synchronization approach. In the next stage, Simulink designs are translated into System Generator design so that *bitstream* file which is used to program FPGA is obtained. Finally, the design is implemented into FPGA by downloading *bitstream* file into FPGA. As an application of FPGAs the synchronization of chaotic systems have been achieved.

ÖZET

SENKRONİZE EDİLMİŞ KAOTİK SİSTEMLERİN ALAN PROGRAMLANABİLİR KAPI DİZİLERİ UYGULAMALARI

Bu tezde, kaotik sistemlerin senkronizasyonunu sağlamak amacıyla bu sistemlerin geometrik özellikleri kullanılmıştır. İlk olarak, her sistem MATLAB Simulink blokları kullanılarak oluşturulmuştur. Daha sonra, tam senkronizasyon yaklaşımı kullanılarak, kaotik sistemlerin senkronizasyonunu sağlamak amacıyla geri beslemeli kontrol sistemi önerilmiştir. Bir sonraki aşamada, yapmış olduğumuz Simulink tasarımları, FPGA' i programlamada kullanacağımız *bitstream* dosyasını elde etmek için, System Generator tasarımına çevirilmiştir. Son olarak, tasarım, *bitstream* dosyasının FPGA' e yüklenmesi ile FPGA uygulaması tamamlanmıştır. FPGA uygulaması olarak kaotik sistemlerin senkronizasyonu başarı ile gerçekleştirilmiştir.

TABLE OF CONTENTS

LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER 1 . INTRODUCTION	1
CHAPTER 2 . THE ORIGIN OF FIELD PROGRAMMABLE GATE ARRAY	3
2.1. Basic Process Technology Types	3
2.1.1. A Simple Programmable Function	3
2.1.2. Fusible Link Technologies	4
2.1.3. Antifuse Technologies	5
2.1.4. Mask-Programmed Devices	7
2.1.5. Programmable Read-Only Memories	8
2.1.6. EPROM Technologies	9
2.1.7. EEPROM Technologies	10
2.1.8. Static Random-Access Memory Based Technologies	11
2.2. Programmable Devices and FPGAs	12
2.2.1. SPLDs and CPLDs	13
2.2.1.1. PROMs	13
2.2.1.2. Programmable Logic Arrays	15
2.2.1.3. PALs and GALs	17
2.2.2. Complex Programmable Logic Devices	18
2.2.3. Application-Specific Integrated Circuits	19
2.2.3.1. Full Custom	19
2.2.3.2. Gate Arrays	19
2.2.3.3. Standart Cell Devices	21
2.2.3.4. Structured ASICs	21
2.2.4. Field Programmable Gate Arrays	23
CHAPTER 3 . CHAOTIC GENERATORS	27
3.1. Chaotic Systems	27

3.1.1. Lorenz System	27
3.1.2. Rössler System	29
3.1.3. Linz and Sprott System	32
3.1.4. Chua System	35
CHAPTER 4 . SYNCHRONIZATION OF CHAOTIC SYSTEMS	41
4.1. Chaos Synchronization	41
4.2. Synchronizability from Control of Chaotic Systems	42
4.2.1. Local Controllability for Complete Synchronizability	43
4.2.2. Local Observability for Complete Synchronizability	44
4.3. Complete Synchronizability	45
4.4. Synchronization of Lorenz System	46
4.5. Synchronization of Rössler System	57
4.6. Synchronization of Linz and Sprott System	65
4.7. Synchronization of Chua System	72
CHAPTER 5 . FPGA IMPLEMENTATION	80
5.1. Implementation Process	80
5.1.1. Implementation of Synchronized Lorenz System	84
5.1.2. Implementation of Synchronized Rössler System	92
5.1.3. Implementation of Synchronized Linz and Sprott System	100
5.1.4. Implementation of Synchronized Chua System	107
CHAPTER 6 . CONCLUSION	116
REFERENCES	119

LIST OF TABLES

<u>Table</u>	<u>Page</u>
Table 3.1. Parameters used to plot the Chua's attractors	38
Table 5.1. MATLAB function "module.m"	81
Table 5.2. The source used by synchronized Lorenz system.	88
Table 5.3. The source used by synchronized Rössler system.	95
Table 5.4. The source used by synchronized Linz and Sprott system	105
Table 5.5. The source used by synchronized Chua system.	111

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 2.1. A simple programmable function.	4
Figure 2.2. The device with unprogrammed fusible links.	4
Figure 2.3. Programmed fusible links.	5
Figure 2.4. Unprogrammed antifuse links.	6
Figure 2.5. Programmed antifuse links.	6
Figure 2.6. Growing an antifuse.	7
Figure 2.7. A transistor-based mask programmed ROM cell.	8
Figure 2.8. A transistor-and-fusible-link-based PROM cell.	8
Figure 2.9. Standart MOS versus EPROM transistors.	9
Figure 2.10. An EPROM transistor-based memory cell.	10
Figure 2.11. An EEPROM - cell.	10
Figure 2.12. An SRAM-based programmable cell.	11
Figure 2.13. Summary of programming technologies.	12
Figure 2.14. Technology timeline.	12
Figure 2.15. Classes of PLDs.	13
Figure 2.16. Unprogrammed PROMs.	14
Figure 2.17. A small block of combinational logic.	14
Figure 2.18. Programmed PROM.	15
Figure 2.19. Unprogrammed PLA.	16
Figure 2.20. Programmed PLA.	16
Figure 2.21. Unprogrammed PAL.	17
Figure 2.22. A generic CPLD structure.	18
Figure 2.23. Different types of ASIC.	19
Figure 2.24. Examples of simple gate array basic cells.	20
Figure 2.25. Channeled gate array architectures.	20
Figure 2.26. Example of structured ASIC tiles.	22
Figure 2.27. Generic structured ASIC.	22
Figure 2.28. The gap between PLD and ASICs.	23

Figure 2.29. A simplified view of a LC.	24
Figure 2.30. Configuring a LUT.	24
Figure 2.31. A <i>slice</i> containing two LCs.	25
Figure 2.32. A CLB containing four <i>slices</i>	25
Figure 2.33. Top-down view of simple FPGA architecture.	26
Figure 3.1. Simulink structure of Lorenz's equations.	28
Figure 3.2. <i>X1outs</i> , <i>X2outs</i> and <i>X3outs</i> outputs of Lorenz's equations.	28
Figure 3.3. The attractor of Lorenz's Simulink block diagram.	29
Figure 3.4. Simulink structure of Rössler equations.	30
Figure 3.5. Four different Rössler's attractors.	30
Figure 3.6. <i>X1outs</i> , <i>X2outs</i> and <i>X3outs</i> outputs of Rössler equations.	31
Figure 3.7. Three dimensional Rössler's attractor for $\gamma = 5.7$	31
Figure 3.8. Simulink structure of Linz and Sprott equations.	33
Figure 3.9. The attractor of the Linz and Sprott's Simulink block.	33
Figure 3.10. <i>X1outs</i> , <i>X2outs</i> and <i>X3outs</i> outputs of Linz and Sprott Equation.	34
Figure 3.11. Three dimensional Linz and Sprott's attractor for $\alpha = 0.6$	34
Figure 3.12. Chua's circuit.	35
Figure 3.13. Chua's circuit non-linear characteristic.	35
Figure 3.14. Simulink structure of Chua equations.	36
Figure 3.15. Non-linear function $g(x_1)$	36
Figure 3.16. <i>X1outs</i> , <i>X2outs</i> and <i>X3outs</i> outputs of Chua Equation for parameter set A.	39
Figure 3.17. Chua's attractor with A, B and C parameters.	40
Figure 4.1. Simulink block diagram of the control command u for Lorenz system.	54
Figure 4.2. Simulink block diagram of synchronized Lorenz system.	55
Figure 4.3. Lorenz master and slave system with different initial conditions.	55
Figure 4.4. The control command u and error signals e_1 , e_2 and e_3	56
Figure 4.5. Simulink block diagram of the control command u for Rössler system.	63
Figure 4.6. Simulink block diagram of the synchronized Rössler system.	63

Figure 4.7. Rössler master and slave system with different initial conditions.	64
Figure 4.8. The control command u and error signals e_1 , e_2 and e_3	64
Figure 4.9. Simulink block diagram of the control command u for Linz and Sprott system	69
Figure 4.10. Simulink block diagram of the synchronized Linz and Sprott system.	70
Figure 4.11. Linz and Sprott master and slave system with different initial conditions.	70
Figure 4.12. The control command u and error signals e_1 , e_2 and e_3	71
Figure 4.13. Simulink block diagram of the control command u for Chua systems.	77
Figure 4.14. Simulink block diagram of the synchronized Chua systems.	78
Figure 4.15. Chua master and slave system with different initial conditions.	78
Figure 4.16. The control command u and error signals e_1 , e_2 and e_3	79
Figure 5.1. <i>MCode</i> Xilinx Block.	81
Figure 5.2. <i>MCode</i> Xilinx block parameters.	81
Figure 5.3. Integrator model using Xilinx blockset library.	82
Figure 5.4. System Generator block in Xilinx blokset library.	83
Figure 5.5. Source window of ISE	83
Figure 5.6. Reduction of resolution and inverting of most significant bit.	84
Figure 5.7. Lorenz system by System Generator.	85
Figure 5.8. Lorenz system attractor.	85
Figure 5.9. The control command u for Lorenz system by System Generator.	86
Figure 5.10. Synchronized Lorenz system by System Generator.	87
Figure 5.11. Synchronized Lorenz system attractors.	87
Figure 5.12. The control command u and error signals e_1 , e_2 and e_3 respectively.	88
Figure 5.13. Lorenz master system state variables and Lorenz slave system state variables.	89
Figure 5.14. The control command u and error signals e_1 , e_2 and e_3 for Lorenz system.	90

Figure 5.15. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by System Generator.	91
Figure 5.16. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.	91
Figure 5.17. Rössler system by System Generator.	92
Figure 5.18. Rössler system attractor.	93
Figure 5.19. The control command u for Rössler system by System Generator. . .	93
Figure 5.20. Synchronized Rössler system by System Generator.	94
Figure 5.21. Synchronized Rössler system attractors.	94
Figure 5.22. The control command u and error signals e_1 , e_2 and e_3 for Rössler system.	96
Figure 5.23. Rössler master system and Rössler slave system state variables. . . .	97
Figure 5.24. The control command u and error signals e_1 , e_2 and e_3 for Rössler system.	98
Figure 5.25. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by System Generator.	99
Figure 5.26. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.	100
Figure 5.27. Linz and Sprott system by System Generator.	101
Figure 5.28. Linz and Sprott system attractor.	101
Figure 5.29. The control command u for Linz and Sprott system by System Generator.	102
Figure 5.30. Synchronized Linz and Sprott by System Generator.	103
Figure 5.31. Synchronized Linz and Sprott system attractors.	103
Figure 5.32. The control command u and error signals e_1 , e_2 and e_3	104
Figure 5.33. Linz and Sprott master and Linz and Sprott slave system state variables.	105
Figure 5.34. The control command u and error signals e_1 , e_2 and e_3	106
Figure 5.35. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by System Generator.	107
Figure 5.36. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.	108

Figure 5.37. Chua system by System Generator.	108
Figure 5.38. Non-linear function.	109
Figure 5.39. Chua system attractor.	109
Figure 5.40. System Generator block diagram of the control command u for Chua systems.	110
Figure 5.41. Synchronized Chua system by System Generator.	110
Figure 5.42. Chua master and slave system with different initial conditions.	111
Figure 5.43. The control command u and error signals e_1 , e_2 and e_3	112
Figure 5.44. Chua master system state variables and Chua slave system state variables.	113
Figure 5.45. The control command u and error signals e_1 , e_2 and e_3	114
Figure 5.46. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by System Generator.	115
Figure 5.47. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.	116

CHAPTER 1

INTRODUCTION

In this thesis an implementation of a geometric nonlinear controller for chaos synchronization in a Field Programmable Gate Array (FPGA) is presented. Chaotic systems are used to show the implementation of chaos synchronization via nonlinear controller implemented in a Xilinx FPGA Virtex - IV XC4VSX35-10ff668. The main idea is to design a nonlinear geometric controller which synchronizes a slave chaotic system to a master chaotic system and then embed them the FPGA. The aim is to show that a reconfigurable device can perform a complicated operation such as the chaos synchronization. The verification of each chaotic system was simulated with MATLAB Simulink and Xilinx System Generator. After verification process is completed, FPGA is configured to perform designed systems so that the results can be obtained by experimentally.

In chapter 2, basic process technology types will be presented. After giving information basic process technology types, programmable devices are introduced. They are programmable logic devices, application-specific integrated circuits and field programmable gate arrays, respectively. Because synchronization of chaotic generators are chosen as an application of FPGAs which are more attracted than other programmable devices. FPGAs offer wide variety of applications such as : Digital signal processor (DSP) (Lund, et al. 2004), software-defined radio (Cummings and Haruyama 1999), aerospace and defense systems (Li, et al. 2000), medical imaging (Leeser, et al. 2005), computer vision (Sen, et al. 2005), speech recognition (Marcus and Nolzco-Flores 2005), cryptography (Mentens, et al. 2006), bioinformatics (Luethy and Hoover 2004) can be counted as recent studies.

Lorenz, Rössler, Linz and Sprott and Chua generators as an application of synchronization problem will be presented and simulated by Matlab Simulink program, in chapter 3.

Synchronization of chaotic generators will be examined in chapter 4. Synchronization problem in chaotic system can be achieved by geometric approach (Solis-Perales, et al. 2003).

Implementation of synchronized chaotic systems can be seen in chapter 5. By

using Xilinx System Generator program, VHDL code can be generated. After VHDL code is obtained by using ISE program, FPGA can be configured to perform properly. In this chapter simulation and experimental results can be seen.

CHAPTER 2

THE ORIGIN OF FIELD PROGRAMMABLE GATE ARRAY

Field Programmable gate arrays (FPGAs) are digital integrated circuits (ICs). FPGAs contain configurable (programmable) logic blocks and configurable interconnects between these blocks. The logic blocks also include memory elements which may be simple flip-flops or more complete blocks of memories. FPGAs are programmed to perform a variety of tasks.

There are two types of FPGAs. One of them is programmed one time. A device which can be programmed one time is referred to as one-time programmable (OTP). The other one is reprogrammed several times.

2.1. Basic Process Technology Types

In this section, some basic technology types will be mentioned. They are fusible link technologies, antifuse technologies, mask-programmed devices, programmable read-only memories, erasable programmable read-only memories, electrically erasable programmable read-only memories and SRAM-based technologies, respectively.

2.1.1. A Simple Programmable Function

As a basis for basic process technology, it is better to start with a simple programmable function so that we can understand the basis of programmable devices. A simple programmable function can be constructed by using two NAND gates and a AND gate as in the Figure 2.1.

NOT gates are used for having inverting form of inputs so there are both unmodified inputs and inverting inputs. As can be seen in the Figure 2.1 there is no connection between inputs and output so the output is always a logic 1 for this case since all of the inputs to the AND gate are connected by pull-up resistors to a logic 1 value. In order to obtain different outputs depending on inputs, The potential links can be established via

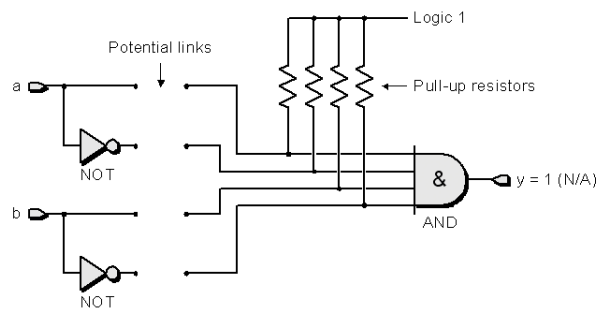


Figure 2.1. A simple programmable function.

some mechanism. These mechanisms will be seen next sections.

2.1.2. Fusible Link Technologies

The fusible-link technology is one of the first methods to program the devices by users and is still used for programming devices. In this case, the device is fabricated with fuses as can be seen in Figure 2.2.

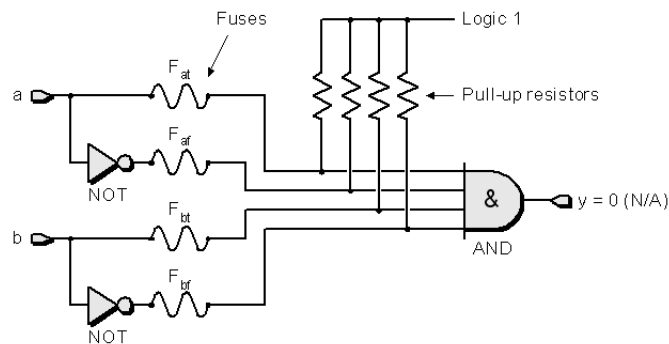


Figure 2.2. The device with unprogrammed fusible links.

These fuses work principle is the same as household products. For example, when something that consumes too much power from the television then its fuse will burn out. This cause an open circuit which protects the rest of unit from harm. Of course, the fuses which are used in programmable devices are microscopically small.

At first, all of the fuses of the programmable device are initially intact. This means that, the output of programmable function is always logic 0 since there are all combination

of inputs. For example, if input a is 0, then the output will be 0 because of the AND gate. Alternatively, if input a is 1 then complementary of a is 0 then again the output is 0.

The thing is that undesired fuses are removed by applying relatively high voltage or current to the device's inputs in order to achieve desired function. For instance, what happens if we remove fuses F_{af} and F_{bt} as in Figure 2.3.

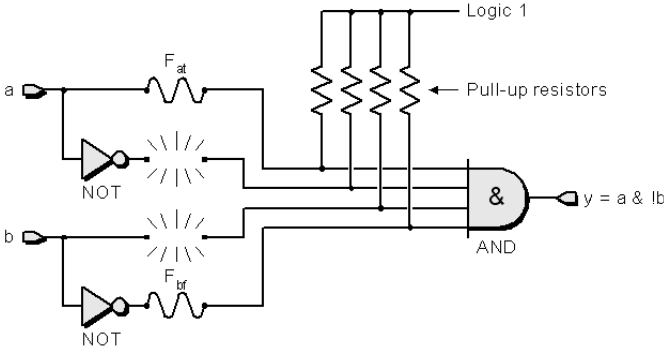


Figure 2.3. Programmed fusible links.

The complementary of input a and the input b are removed from the circuit so we obtain new function which is $y = a \& b$. (AND operation is represented by $\&$ character and NOT operation is represented by $!$ character. The process of removing fuses is called programming the device or burning the device. Fusible-link based devices are said to be one-time programmable (OTP), since after blowing fuse there is no way to replace blown fuse.

2.1.3. Antifuse Technologies

As an alternative to fusible-link technologies, there are antifuse technologies. In this time each configurable path is called antifuse. At first, in unprogrammed state of antifuses, an antifuse has such a high resistance that it acts as a insulator so there is no current flow through unprogrammed antifuse. This time the output is always logic 1 because of pull-up resistors as in Figure 2.4.

In order to achieve desired function by applying relatively high voltage or current to the device's inputs so that antifuses can be programmed or grown. For example, the complementary version of input a and input b are grown by applying proper voltage or current then the device will perform the function $y = !a \& b$ as in Figure 2.5.

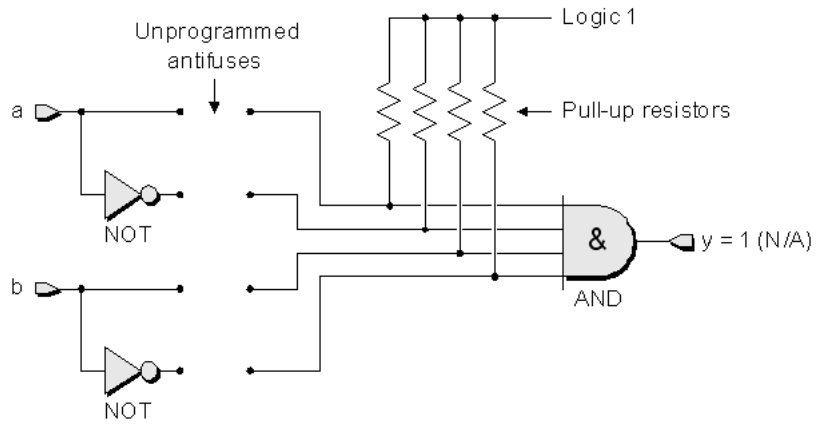


Figure 2.4. Unprogrammed antifuse links.

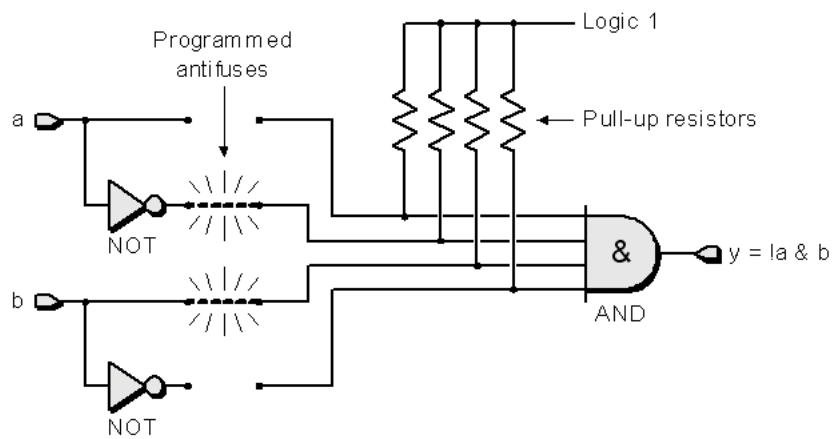


Figure 2.5. Programmed antifuse links.

In unprogrammed state there is amorphous silicon (Figure 2.6.a) link between nodes. it act as an insulator so there is no current flow. By converting the insulating amorphous silicon to conducting polysilicon (Figure 2.6.b) by applying relatively high voltage or current, the programmed link between nodes will be constructed.

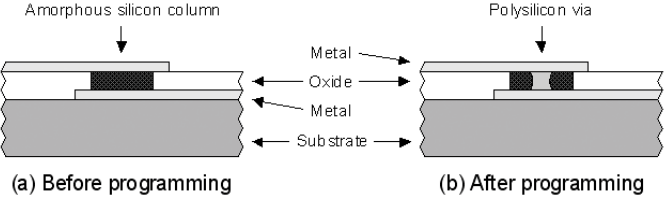


Figure 2.6. Growing an antifuse.

Like fusible-link technologies, antifuse technologies are also said to be one time programmable because after growing an antifuse there is no way to replace grown antifuse.

2.1.4. Mask-Programmed Devices

There are two types of memory in electronicsystems. One of them is read-only memory (ROM) and the other one is random-access memory (RAM).

ROMs can be programmed one time after that the data is read from ROMs but new data cannot be written after they are programmed once. The written data remains even power is removed from the system. This kind of system is called nonvolatile system. On the contrary, data can be both written into and read from RAM devices. The written data does not remain when power removed from the system. This kind of system is called volatile system.

Basic ROMs are mask-programmed devices since process of constructing ROMs are done by means of photo-mask which are used to create the transistors and the metal tracks. For instance, a transistor-based ROM cell can be seen in Figure 2.7.

In order to understand how to program mask-programmed ROM cell two options will be examined. One of them is that there is a mask-programmed connection and the other one is that there is no mask-programmed connections. Now consider a row line is active state. In the first case, there is a connection between transistor and the column line so transistor acts as a resistor then the output will be logic 0. The second case because

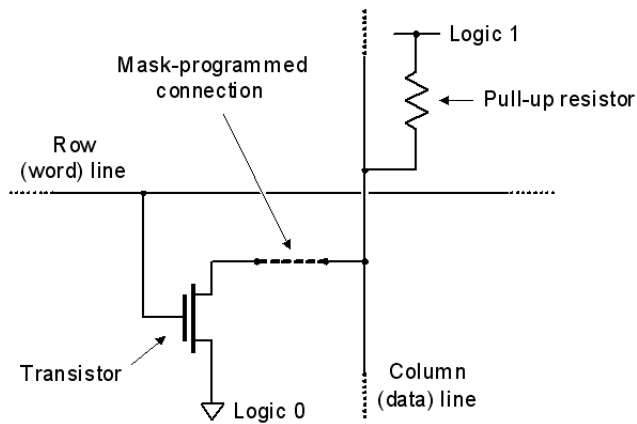


Figure 2.7. A transistor-based mask programmed ROM cell.

there is no connection because of pull-up resistor the output will be logic 1.

2.1.5. Programmable Read-Only Memories

The programmable read-only memory devices were developed since mask-programmed devices have problem with their cost and producing time. PROM devices were created using a nichrome-based fusible-link technology (Maxfield 2004). As an example, simplified representation of a transistor-and-fusible-link based PROM cell as can be seen in Figure 2.8.

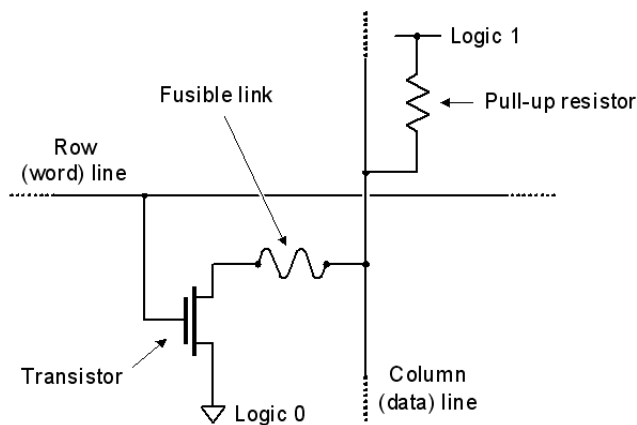


Figure 2.8. A transistor-and-fusible-link-based PROM cell.

At first, all of the fusible links in the device exist. Now consider a row line is active state so transistors act as a resistor then the output will be logic 0. In order to obtain desired function some of fusible links are burnt by applying relatively high voltage

and current. Because of burning some of the fusible links the output will be logic 1. As can be understood these devices are also OTP because they use fusible link technologies as well.

2.1.6. EPROM Technologies

As was previously seen previous section PROM devices are one time programmable devices. For this reason, new devices which could be programmed, erased and reprogrammed were developed. This was erasable programmable read-only memory (EPROM).

An EPROM transistor has the same basic structure as a standart MOS transistor. The difference is that there is extra polysilicon floating gate isolated by layers of oxide (Maxfield 2004) as can be seen in Figure 2.9.

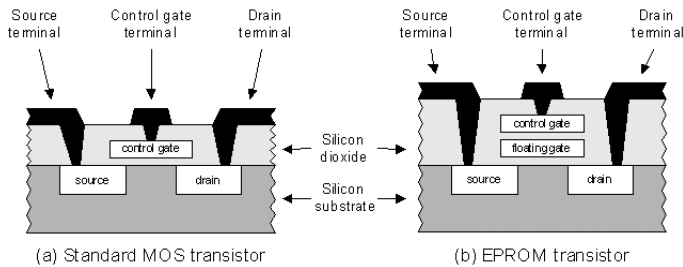


Figure 2.9. Standart MOS versus EPROM transistors.

Before programmed the device, EPROM transistors work as standart MOS transistors. When the relatively high voltage is applied between the control gate and drain terminals EPROM transistors are programmed. Afer programming process even though programming signal is removed from the system, negative charge remains on the floating gate.

EPROM transistor-based devices are established as all of the floating gates transistors are uncharged as in Figure 2.10. Consider, a row line in its active state so EPROM transistors act as a resistor then output will be pulled down to logic 0. In order to program device, the floating gates are charged by relatively high voltage so there is no current flow through transistors then the output will be logic 1.

The main difference between PROM and EPROM is that an EPROM cell can be erased by discharging the electrons on Floating gate. The discharge operation needs

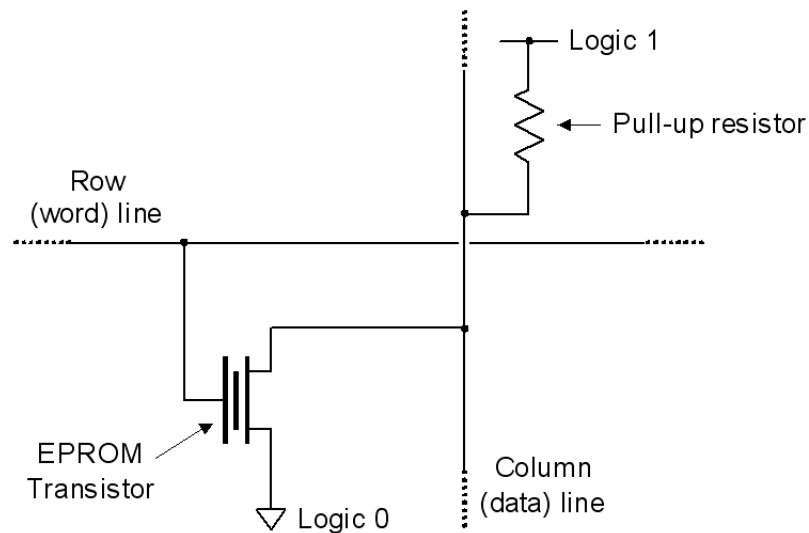


Figure 2.10. An EPROM transistor-based memory cell.

energy which is provided by ultraviolet (UV) radiation. In order to erase the device, the device is removed from its host circuit board and then it is placed in an enclosed container with an intense UV source.

2.1.7. EEPROM Technologies

Electrically erasable programmable read-only memories were developed so that erasing process could be done by electrically. EEPROM cell is approximately 2.5 times larger than EPROM cell since EEPROM cell include two transistor (as can be seen in Figure 2.11), on the other hand EPROM cell include only one.

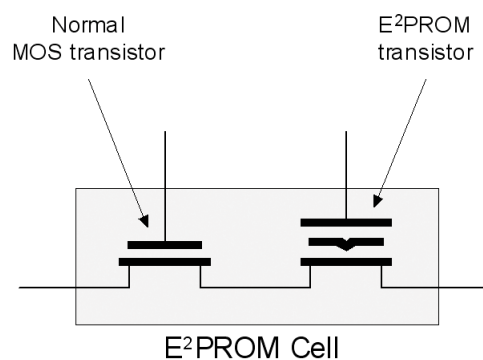


Figure 2.11. An EEPROM - cell.

The EEPROM transistor has similar form of EPROM transistor. EEPROM tran-

sistor has also contain floating gate but it is thinner than the one in EPROM cell. The normal MOS transistor in EEPROM cell is used for erasing the cell electrically.

2.1.8. Static Random-Access Memory Based Technologies

There are two types of RAM devices. One of them is dynamic RAM (DRAM) and the other one is static RAM (SRAM). In the case of DRAMs, each cell is formed from transistor-capacitor pair. In order to keep the data unchanged, that has been loaded into a DRAM cell each cell must be periodically recharged. In order for that there must be a control circuitry. It makes design complex and cost effective.

SRAM-based programmable cell do not need to be refreshed. When the data has been loaded into an SRAM cell it will remain unchanged until power is removed from the system. SRAM-based programmable cell can be seen in Figure 2.12.

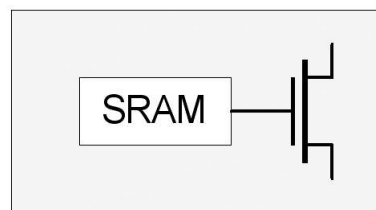


Figure 2.12. An SRAM-based programmable cell.

The SRAM-based programmable cell is composed of transistors. One of them is used for driving the output whether it is logic 0 or logic 1 depending on the contents of the storage element. The number of transistors used for a latch is either 4 or 6 so SRAM cells consume a significant amount of silicon area estate. Another disadvantage is with respect to fuse and antifuse technologies that when power is removed from the system the data will be lost. On the other hand, SRAM based programmable devices can be reprogrammed quickly and repeatedly.

Figure 2.13 shows which technologies are used to program SPLDs, CPLDs and FPGAs.





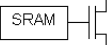
Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E ² PROM/ FLASH		SPLDs and CPLDs (some FPGAs)
SRAM		FPGAs (some CPLDs)

Figure 2.13. Summary of programming technologies.

2.2. Programmable Devices and FPGAs

In order to understand the development process of FPGAs we will examine related programmable technologies. As can be seen in Figure 2.14 which technologies appeared on the scene in order.

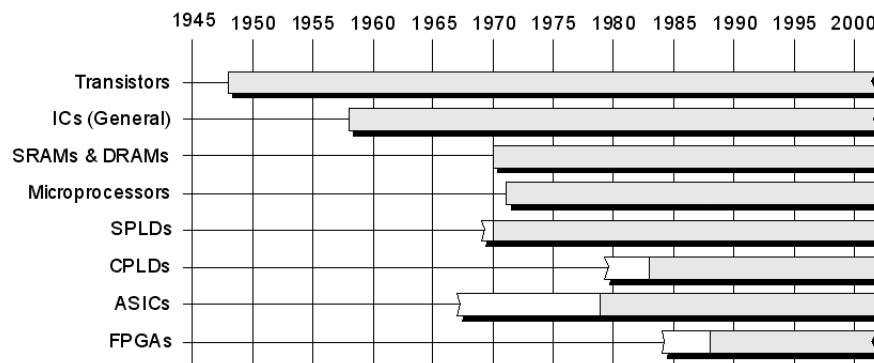


Figure 2.14. Technology timeline.

The white part of the bars indicate that although those technologies existed, they were not paid attention effectively by system designers because of some reasons. For example, the first FPGA was introduced in 1984 by Xilinx but design engineers did not realized how important FPGA is until 1990s.

In Figure 2.14, there are eight technologies but we will examine four of them which are more closer to FPGAs.

2.2.1. SPLDs and CPLDs

The programmable logic devices (PLDs) are the first programmable integrated circuits (ICs). PROMs can be counted as the first example of PLDs. They were on the scene in 1970 but they were rather simple so more complex versions became available at the end of the 1970s. So as to separate complex ones and simple ones they were called as complex PLDs (CPLDs) and simple PLDs (SPLDs). SPLDs can be divided into four main classes as PROMs, PLAs, PALs and GALs as can be seen in Figure 2.15. CPLDs will be examined as one class.

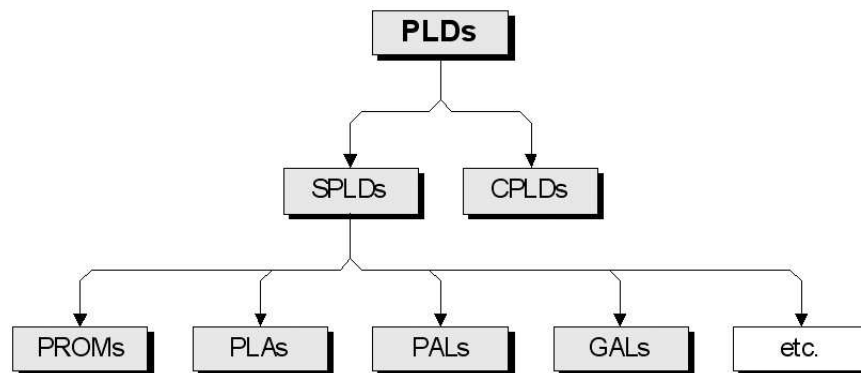


Figure 2.15. Classes of PLDs.

2.2.1.1. PROMs

PROMs were the first simple PLDs. They were on the scene in 1970. In order to see how PROM performs, simple example of PROM will be examined. PROM consists of a fixed array of AND gates driving a programmable array of OR gates. 3-input, 3 output PROM (as in Figure 2.16) can be considered as an example of basic PROM cell.

Fusible links, EPROM transistors or EEPROM cells can be used to construct programmable links in the OR array. In fact Figure 2.16 is an illustration. It does not represent an actual circuit diagram since in reality, each AND gate in the AND array has 3 inputs and similarly each OR gate in the OR array has 8 inputs provided by the outputs from the AND array.

The PROM can be used to implement any block of combinational logic provided that it does not have too many inputs or outputs. In Figure 2.16, the simple 3-input, 3-output PROM can be seen. It can be used to implement any combinational logic function

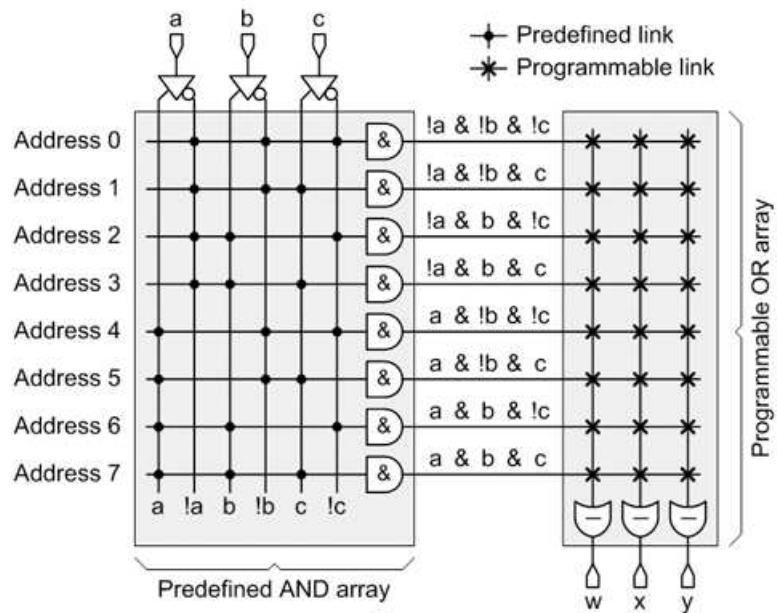


Figure 2.16. Unprogrammed PROMs.

with up to 3 inputs and 3 outputs. Example of 3-input and 3-output combinational logic function can be seen in Figure 2.17.

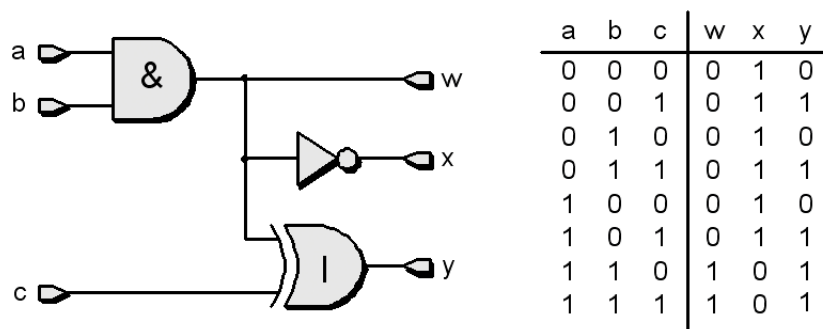


Figure 2.17. A small block of combinational logic.

In order to construct above combinational function the PROM can be programmed such a way that the same results can be obtained. It can be done by programming the appropriate links in the OR array. By using that truth table the outputs can be obtained as in Equation 2.1 below:

$$\begin{aligned}
 w &= (a \& b) \\
 x &= (!a \& !b \& !c) | (!a \& !b \& c) | (!a \& b \& !c) | (!a \& b \& c) | (a \& !b \& !c) | (a \& !b \& c) \\
 y &= (!a \& !b \& c) | (!a \& b \& c) | (a \& !b \& c) | (a \& b \& !c) | (a \& b \& c)
 \end{aligned}
 \tag{2.1}$$

Figure 2.18 shows programmed PROM. As can be seen in Figure 2.18, unnecessary links were removed so that desired outputs w , x , and y are obtained.

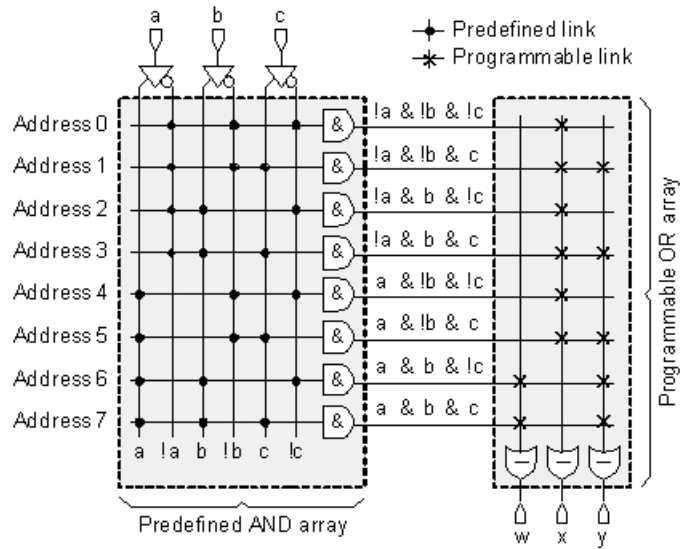


Figure 2.18. Programmed PROM.

The example in Figure 2.17 is very simple. It is just for illustration of real PROMs. Indeed, real PROMs have more inputs and outputs so they are capable of implementing larger blocks of combinational logic. Before PROMs, combinational logic was usually implemented by means of integrated circuits such as the TI 74xx series devices (Maxfield 2004).

2.2.1.2. Programmable Logic Arrays

Programmable logic arrays (PLAs) became first available circa 1975. Because both the AND array and OR array were programmable, PLAs were the most user configurable of the SPLDs. For example, 3-input, 3-output unprogrammed PLA can be seen in Figure 2.19 .

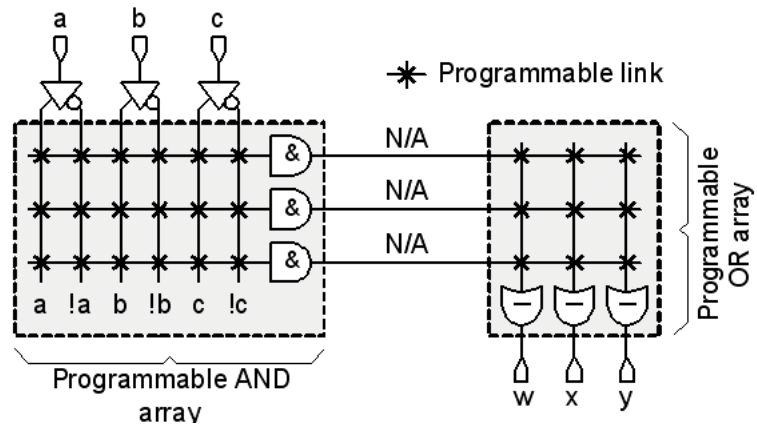


Figure 2.19. Unprogrammed PLA.

The number of AND gates in the AND array is independent of inputs to the device. Similarly, the number of OR gates in the OR array is independent of inputs to device and the number of AND gates in the AND array.

Assume that three equations (Equation 2.2) are implemented by PLA. After PLA is programmed, the programmed PLA diagram can be seen in Figure 2.20.

$$\begin{aligned}
 w &= (a \& c) | (!b \& !c) \\
 x &= (a \& b \& c) | (!b \& !c) \\
 y &= (a \& b \& c)
 \end{aligned}
 \tag{2.2}$$

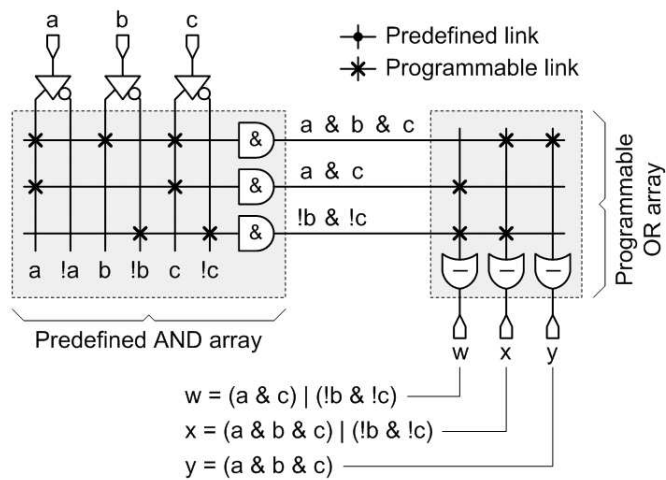


Figure 2.20. Programmed PLA.

PLAs were useful for large designs whose logical equations has a lot of common product terms. Common product terms can be used multiple outputs, for example, the term $(!b \& !c)$ is used in both the w output and x output in Figure 2.20. This feature might be called as product-term sharing.

On the other hand, because both their AND and OR arrays were programmable, signals take long time to pass through programmable links compared to their predefined counterparts. Therefore, PLAs were significantly slower than PROM.

2.2.1.3. PALs and GALs

A next class of device called programmable array logic (PAL) was introduced in the late 1970s. PAL is almost the exact opposite of a PROM since it has a programmable AND array and predefined OR array. As an example, 3-input, 3-output PAL can be seen in Figure 2.21.

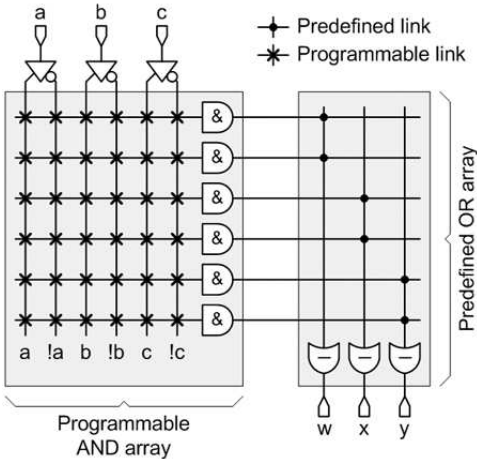


Figure 2.21. Unprogrammed PAL.

The advantage of PALs is that they are faster than PLAs since the only one of PALs arrays is programmable. On the other hand, they are more limited as compared to PLAs because they are not capable of being programmed by OR gates in OR array.

The Generic Array Logic (GAL) device was an innovation of the PAL and was invented by Lattice Semiconductor. The GAL was an improvement on the PAL because one device was able to take the place of many PAL devices or could even have functionality not covered by the original range. In addition, it was erasable and reprogrammable making prototyping and design changes easier for engineers (Wikipedia 2007a).

2.2.2. Complex Programmable Logic Devices

Because bigger (in terms of functional capability), smaller (in terms of physical size), faster, more powerful and cheaper devices were needed, complex programmable devices (CPLDs) were developed at the end of the 1970s.

Altera introduced CMOS-based EPROM technologies which make CPLDs popular in 1984. Because CMOS-based transistors allowed CPLD to achieve functional density and complexity while consuming little power. Because of EPROM technologies, CPLDs were used to develop prototyping environments.

A generic CPLD consists of a number of SPLD blocks (usually PALs). There is a programmable interconnection matrix which make SPLD-like blocks connect each other as can be seen in Figure 2.22.

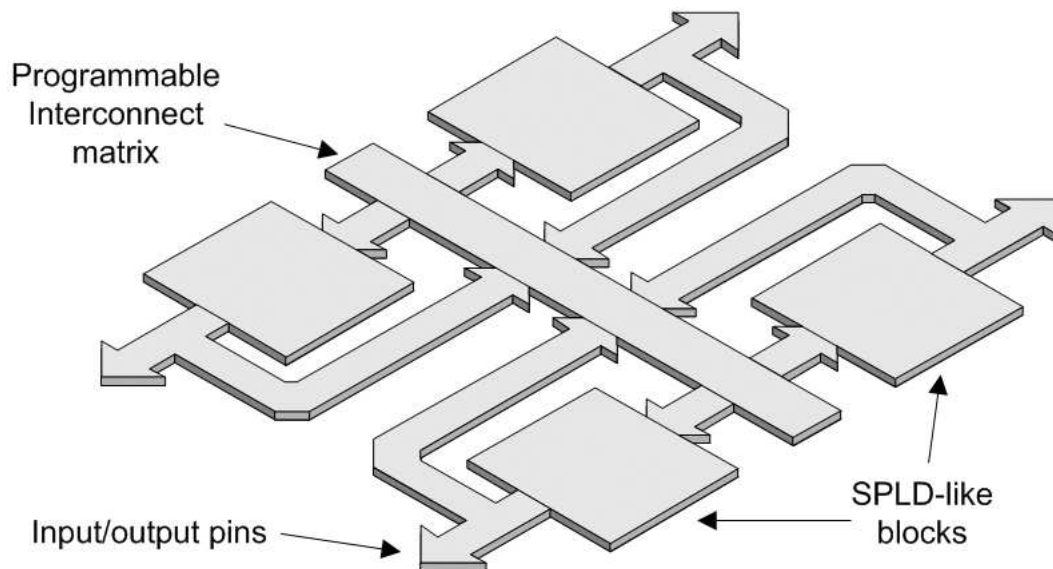


Figure 2.22. A generic CPLD structure.

The programmable interconnection matrix is used to program links between SPLD-like blocks. SPLD-like blocks are also programmable so in order to achieve desired function, both the SPLD-like blocks and programmable interconnection matrix are programmed properly.

2.2.3. Application-Specific Integrated Circuits

There are four types of application-specific integrated circuits (ASICs) which are full custom, gate arrays, standart cell devices, sructured ASICs in the order of time. ASIC types can be seen in Figure 2.23 in the order of the complexcity. ASIC types will be described in the order of time in next subsections.

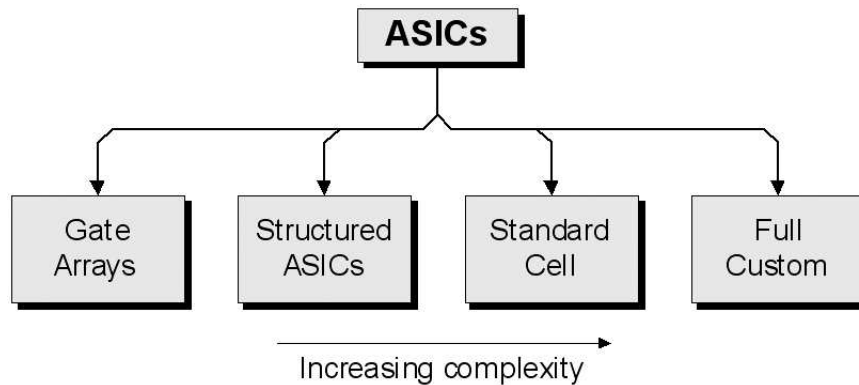


Figure 2.23. Different types of ASIC.

2.2.3.1. Full Custom

The full custom ASICs are like microprocessors. They were designed in order to be used by a specific company. In the case of full custom devices, every mask layer used to fabricate the silicon chip is controlled completely by design engineers. No components are prefabricated in full custom ASICs and they do not have any libraries of predefined logic gates and functions.

The design of full custom devices is highly complex and time-consuming, but the resulting chips contain the maximum amount of logic with minimal waste of silicon real estate (Maxfield 2004).

2.2.3.2. Gate Arrays

CMOS-based gate array technology became available in the mid-1970s. The idea of gate arrays is that a collection of unconnected transistors and resistors were implemented on basic cell. Each ASIC vendor decides the optimum mix of components provided in its particular basic cell. As an example of basic cell can be seen in Figure 2.24.

The basic cells are typically presented as either single-column or dual-column arrays. The free areas between the arrays are called as the channels (Figure 2.25).

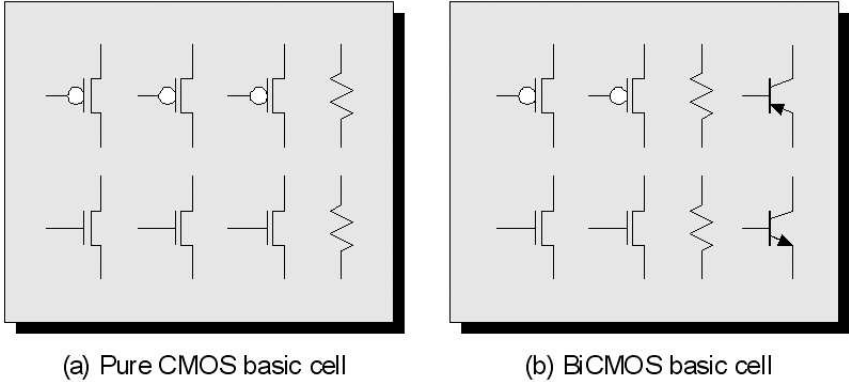


Figure 2.24. Examples of simple gate array basic cells.

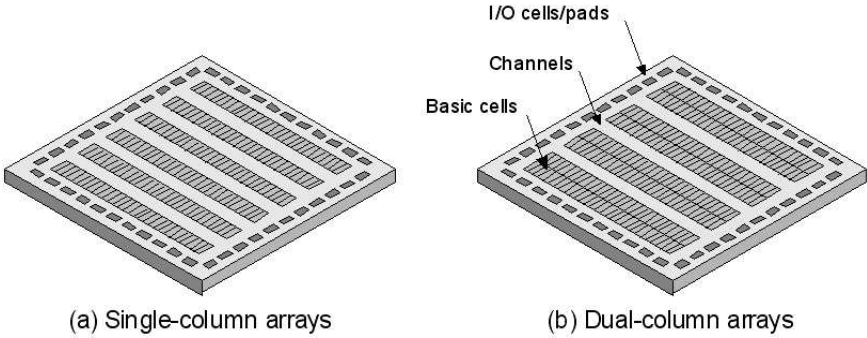


Figure 2.25. Channeled gate array architectures.

A gate level netlist is used to describe the logic gates and connections between them. Special mapping, placement, and routing software tools are used to assign the logic gates to specific basic cells and define how the cells will be connected together (Maxfield 2004). The component inside the basic cells and the basic cells itself are linked together by the metalization layer which is created by the photo-mask.

An advantage of gate arrays is that they have considerable cost advantages because the components in basic cell are prefabricated like transistors so the only thing whcih needs to be customized is the metalization layers. On the other hand, the power consumption and performance of design are not effective since the most designs leave significant amounts of internal resources unutilized.

2.2.3.3. Standart Cell Devices

Standart cell devices became available in the early of 1980s. In the case of standart cell devices, cell library is defined by ASIC vendor so as to be used by design engineers. The vendor also offers hard-macro and soft-macro libraries, which include elements such as processors, communication functions, RAM and ROM functions.

As compared to gate arrays, no components are prefabricated on the chip. Unlike gate arrays, the concept of a basic cell are not used in standart cell devices. Special tools are used to place each logic gate in the netlist. They are also used to determine the optimum way which the gates are to be linked.

The standart cell devices provide each logic function to be constructed by using the minimum number of transistors. Standart cell device, hence, offer more optimal solution for the silicon real estate than gate arrays do.

2.2.3.4. Structured ASICs

Structured application-specific integrated circuits were seen at first at the beginning of the 1990s but they were not used effectively at all. ASIC manufacturers started to investigate reducing ASIC design cost and development times circa 2001. In 2003, structured ASIC became available.

Structured ASIC devices have fundamental element called a module by some and a tile by others. Module or tile element might contain prefabricated generic logic (either as gates, mutiplexers, or a lookup table), registers and possibly a little local RAM as can be seen in Figure 2.26 (Maxfield 2004).

An array (sea) of tiles is prefabricated. If the tile contains only generic logic in the form of prefabricated gates, multiplexers etc. then it can be called base tile. By combaning base tiles and adding registers, small memory elements and other logic to base tile, the master tile can be obtained. Master tiles are also prefabricated on the chip. RAM blocks, clock generators, boundary scan logic are also prefabricated on chip as can be seen in Figure 2.27.

Structured ASIC can be configured usnig only the metalization layers like a standart gate array. The difference is that, the tile of structured ASIC is more sophisticated rather than the standart gate array so most of the metalization layers are predefined as

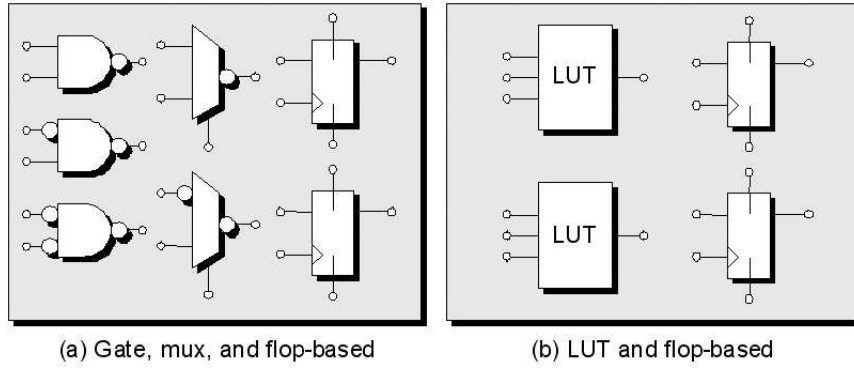


Figure 2.26. Example of structured ASIC tiles.

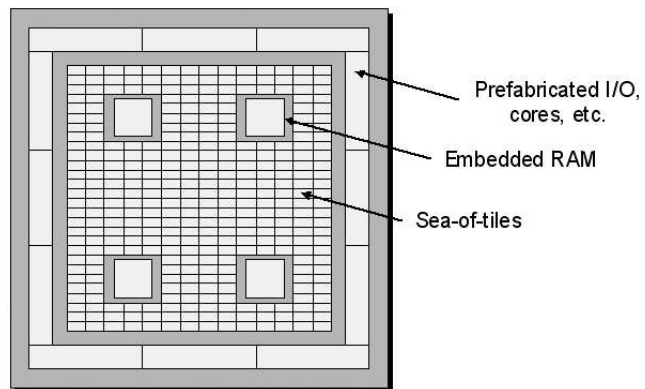


Figure 2.27. Generic structured ASIC.

well.

On the other hand, structured ASICs require three times the real estate and consume two to three times the power of a standard cell device to perform the same function because predefined functions which structured ASIC use is more sophisticated than the standard cell device so these functions consume much power and much real estate.

2.2.4. Field Programmable Gate Arrays

A field programmable gate array (FPGA) is a general-purpose integrated circuit that is programmed by the designer rather than the device manufacturer (Xilinx 2007a). FPGAs were available at the beginning of the 1980s. There was a gap between PLDs and ASICs (Figure 2.28). PLDs were highly configurable and had fast design and modification times, but they could not support large or complex functions. On the other hand, ASICs could support large and complex functions, but they were time-consuming to design and, once a design was implemented, it can not be changed. FPGAs filled the gap by being placed between PLDs and ASICs since FPGAs were programmable like PLDs and they could support large and complex functions to be performed like ASICs.

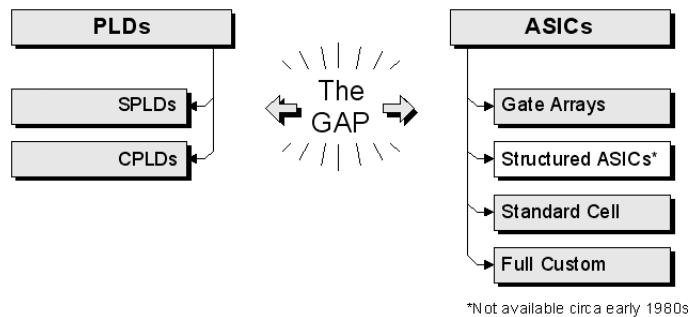


Figure 2.28. The gap between PLD and ASICs.

The first FPGAs were based on CMOS and used SRAM cells for configuration purposes (Maxfield 2004). Early FPGAs were simple and contained few gates as compared to recent FPGAs. The early devices were based on a programmable logic block (PLB). PLBs contain a 3-input lookup table (LUT), a register that could act as a flip-flop or a latch and a multiplexer. Recent FPGAs contain a 4-input LUT which can be used as 16×1 RAM or a 16-bit shift register, a multiplexer and flip-flop as can be seen in Figure 2.29. The new form of PLB is called Logic Cell (LC) which will be used next sections.

The board we used to implement synchronized chaotic systems has 34560 LCs.

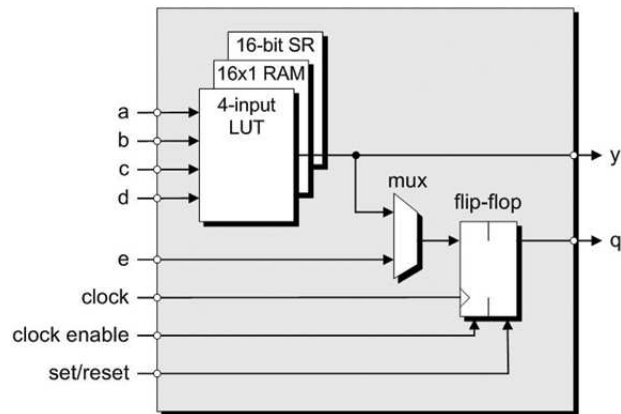


Figure 2.29. A simplified view of a LC.

Every logic block in the device could be configured to perform a different function by means of appropriate SRAM programming cells. A flip-flop can be configured to keep the data in it until being triggered by a positive or negative-going-clock. The multiplexer can be configured to select the output from the LUT or an independent input from LC input.

The example is given to understand how LC works. In order to perform $y = (a \& b) | !c$ function the LUT cell has to be loaded appropriate output values as can be seen in Figure 2.30. By means of the multiplexer in the LUT, the output of function can be selected among the inputs in the SRAM cell.

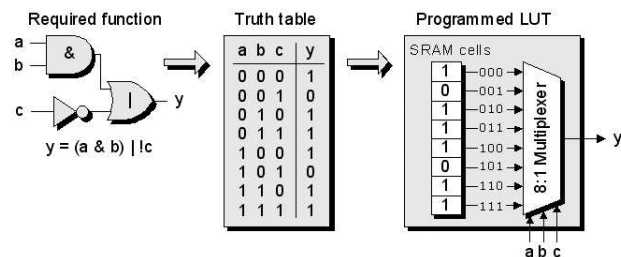


Figure 2.30. Configuring a LUT.

The next step up the hierarchy is called as *slice*. Each slice contains two LCs as can be seen in Figure 2.31. The board we used to implement synchronized chaotic

systems has 15360 *slices*. There are internal wires between LCs they are not drawn for simplicity.

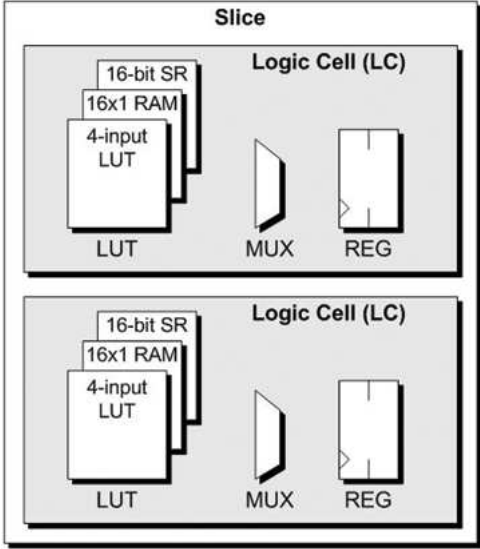


Figure 2.31. A *slice* containing two LCs.

The next step up the hierarchy is called as configurable logic block (CLB). Each CLB contains usually four *slices* as can be seen in Figure 2.32. The CLB of the board we used has four *slices* as well. There is fast programmable interconnect within the CLB as well. This interconnect is used to connect slices which are placed at the same CLB.

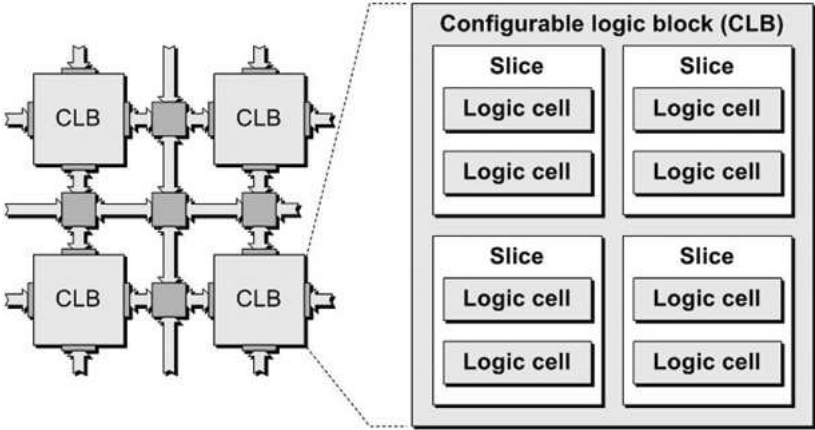


Figure 2.32. A CLB containing four *slices*.

As can be understood previous paragraphs, the hierarchy of FPGA is like this LC \rightarrow *Slice* (with two LCs) \rightarrow CLB (with four slices). The hierarchy is complemented by

an equivalent hierarchy in the interconnect. Therefore, there is the fastest interconnect between the LCs in a *slice*, then slightly slower interconnect between *slice* in a CLB, there is the slowest interconnect between CLBs. These interconnects hierarchy were constructed to avoid interconnect-related delays.

The complete FPGA contains a large number of programmable CLBs surrounded by programmable interconnects as can be seen in Figure 2.33. Besides programmable CLBs and interconnects, FPGA include primary input/output (I/O) pins. By means of its own SRAM cells, the interconnect can be programmed such that the primary inputs to the device are connected to the inputs of one or more CLBs, and the outputs from any CLB can be used to drive the inputs to any other logic block, the primary outputs from the device, or both (Maxfield 2004).

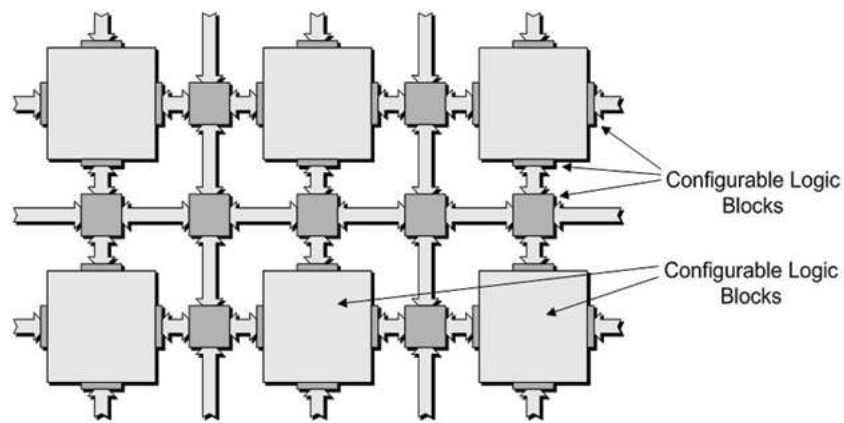


Figure 2.33. Top-down view of simple FPGA architecture.

At the end, FPGAs successfully fill the gap between PLDs and ASICs. They are highly configurable like PLDs and capable of implementing large and complex functions like ASICs. In addition, they have fast design and modification times as compared to PLDs and they are able to be reprogrammed unlike ASICs.

CHAPTER 3

CHAOTIC GENERATORS

3.1. Chaotic Systems

In the next subsections, Lorenz system, Rössler system, Linz and Sprott system and Chua system will be studied. The differential nonlinear equations of these systems will be presented and the chaotic behaviour will be studied. Simulink model will be constructed for each chaotic system, so the results of mathematical simulations of these chaotic differential equations can be seen.

3.1.1. Lorenz System

$$\begin{aligned}\frac{dx_1}{dt} &= \alpha(x_2 - x_1) \\ \frac{dx_2}{dt} &= \beta x_1 - x_2 - x_1 x_3 \\ \frac{dx_3}{dt} &= x_1 x_2 - \gamma x_3\end{aligned}\tag{3.1}$$

where the parameters $\alpha = 10$, $\beta = 28$ and $\gamma = 8/3$. Lorenz's equations (Lorenz 1963) have two nonlinearities responsible for the chaotic behaviour: the products $x_1 x_3$ and $x_1 x_2$ that are performed by two multipliers. The equations are simulated using the Simulink block diagram presented by Figure 3.1. At least, one of the integrators presented in this diagram must have a non-zero initial condition in order to produce non-zero outputs. In this diagram the block *Integrator* has internal initial condition set to 0.001. *Integrator1* and *Integrator2* blocks have zero initial condition.

Figures 3.2 shows the outputs *X1outs*, *X2outs*, *X3outs* produced by the Simulink block diagram of Figure 3.1. The Lorenz's attractor (Figure 3.3) is also produced by the Simulink block diagram of Figure 3.1. The Figure 3.3 is obtained by plotting the output variable *X1outs* versus *X2outs*.

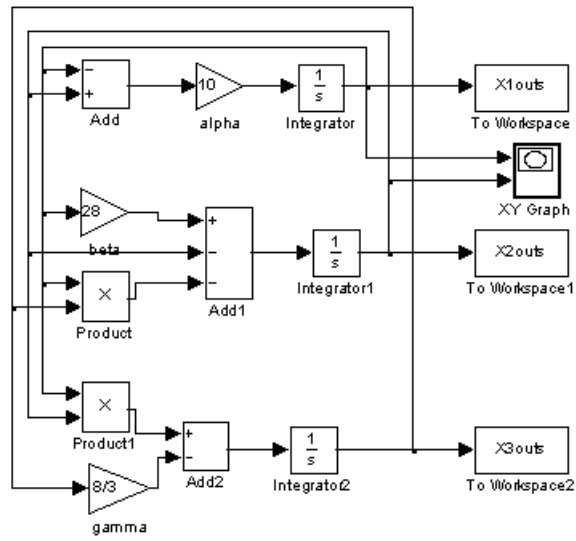


Figure 3.1. Simulink structure of Lorenz's equations.

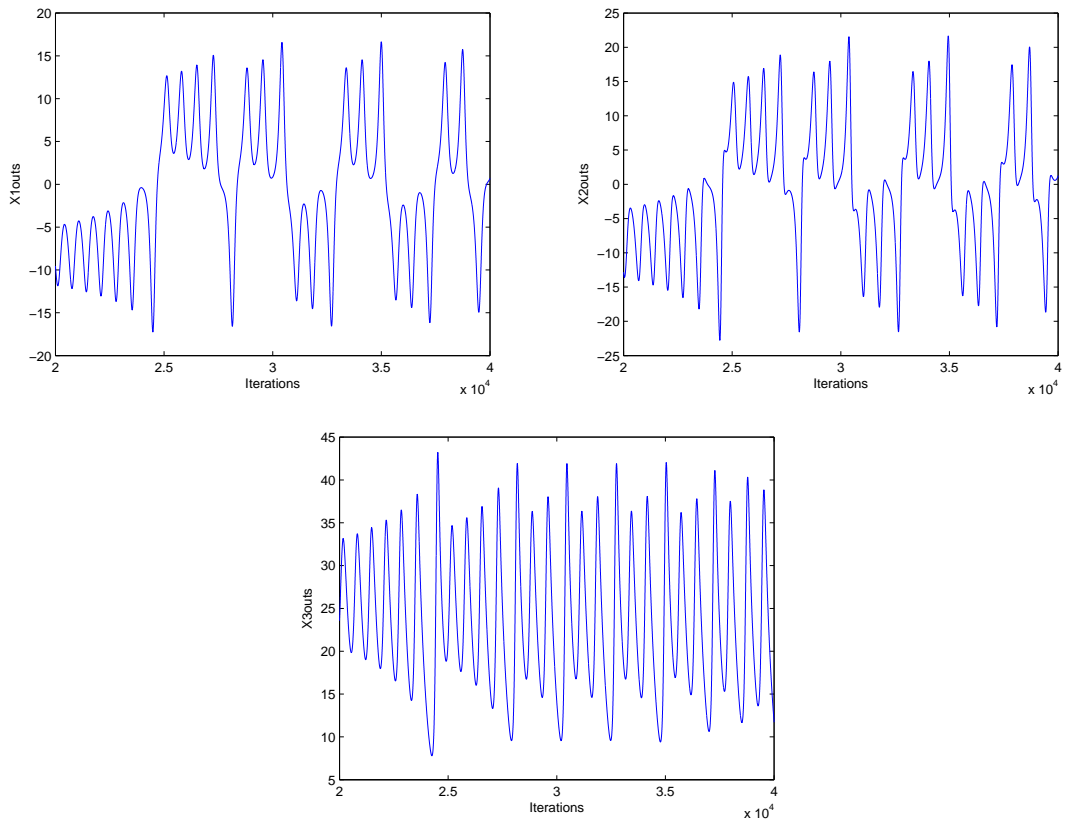


Figure 3.2. $X1outs$, $X2outs$ and $X3outs$ outputs of Lorenz's equations.

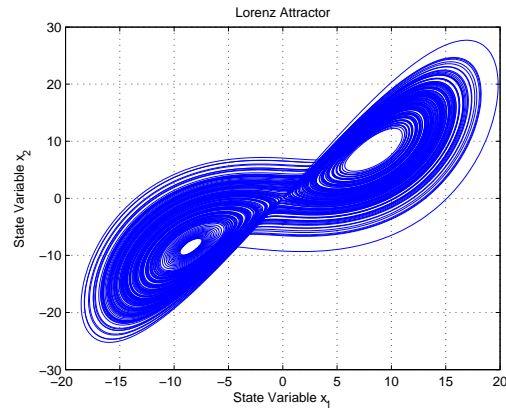


Figure 3.3. The attractor of Lorenz's Simulink block diagram.

3.1.2. Rössler System

Rössler equations were created by Otto Rössler (Rössler 1976). Rössler system was described with three non-linear differential Equations 3.2. Rössler's system is simpler than Lorenz's, in which there is only one nonlinear term x_1x_3 in the third equation.

$$\begin{aligned}
 \frac{dx_1}{dt} &= -(x_2 + x_3) \\
 \frac{dx_2}{dt} &= x_1 + \alpha x_2 \\
 \frac{dx_3}{dt} &= \beta + x_3(x_1 - \gamma)
 \end{aligned}
 \tag{3.2}$$

Figure 3.4 below presents Simulink block diagram of the Rössler system described with three non-linear differential equations as can be seen in Equations 3.2. In this Simulink design, the parameters are set to: $\alpha = 0.2$, $\beta = 0.2$ and $\gamma = 5.7$. Integrators initial conditions are set to zero, except the *integrator1* block which is set to 3.0474. The simulation parameters use a *Fixed - Step* solver type with *fixed step* size equal to 0.1 in Simulink configuration box.

By setting the parameters α and β equal to 2 and varying the parameter γ , the chaotic behaviour of Rössler system can be seen. Period one ($\gamma = 2.5$), period two ($\gamma = 3.5$) and period four trajectories ($\gamma = 4.0$) can be easily seen in Figure 3.5. These period doubling lead eventually a chaotic state as shown by the last attractor plotted for $\gamma = 5.7$.

The behaviour of the x_3 state variable in the chaotic state ($\gamma = 5.7$) is understood by looking at time series in Figure 3.6. From Figure 3.6, x_3 state variable is small most of

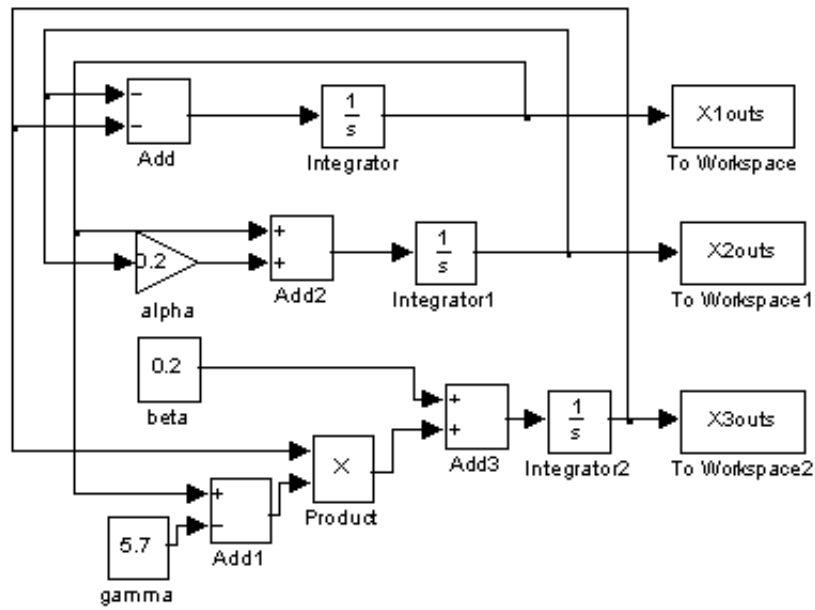


Figure 3.4. Simulink structure of Rössler equations.

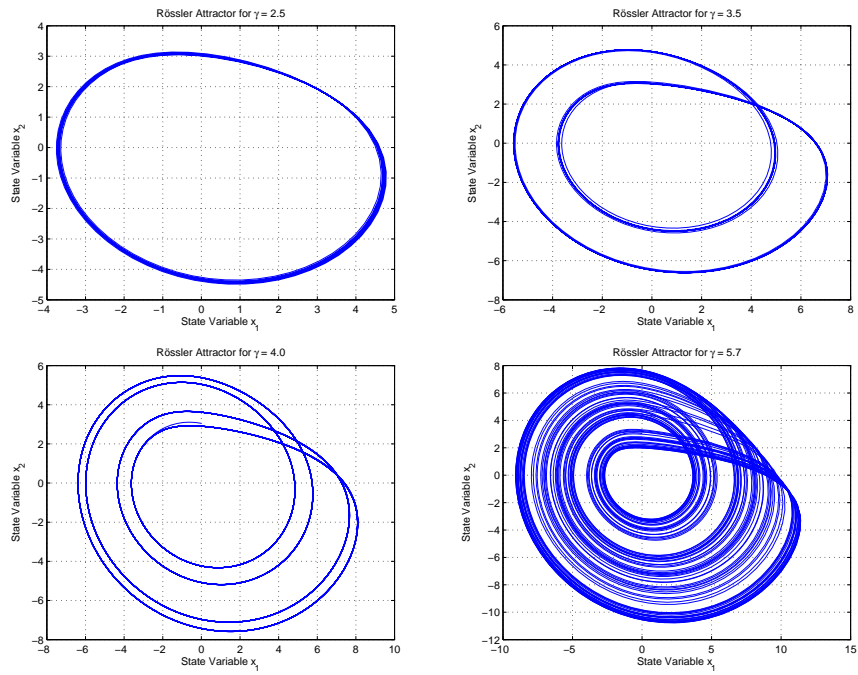


Figure 3.5. Four different Rössler's attractors.

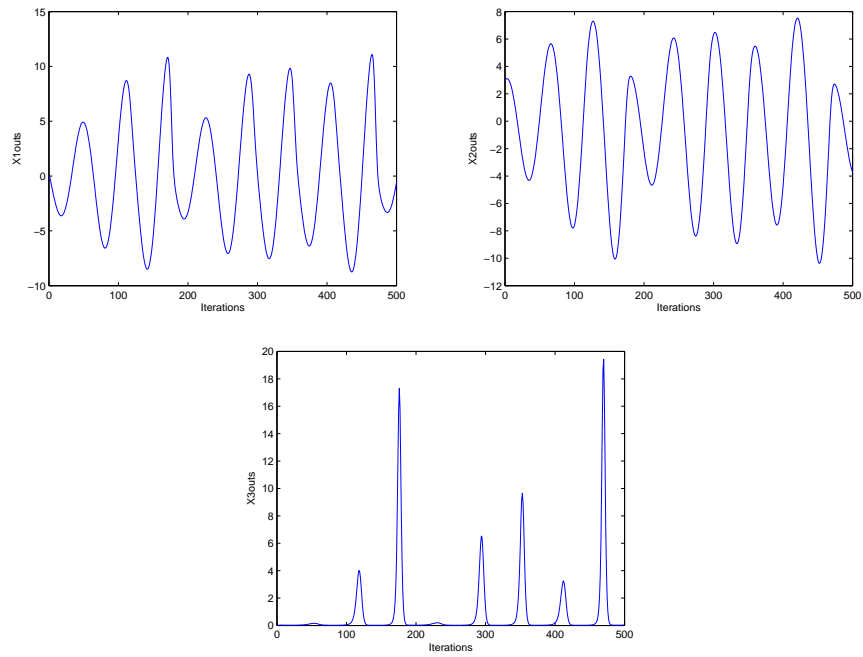


Figure 3.6. $X1_{outs}$, $X2_{outs}$ and $X3_{outs}$ outputs of Rössler equations.

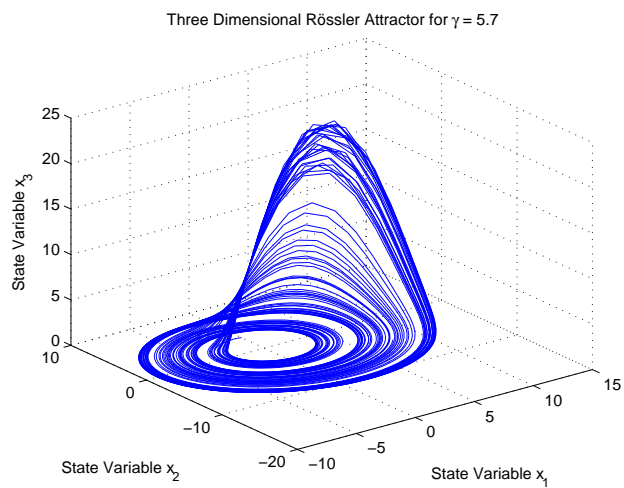


Figure 3.7. Three dimensional Rössler's attractor for $\gamma = 5.7$

the time which means that trajectories mostly are close to the $x - y$ plane. But sometimes there are spikes in the x_3 time series. For reference, the time series for the x_1 and x_2 variables can be also seen in Figure 3.6.

The full three - dimensional phase portrait of Rössler system is shown in Figure 3.7. This is the attractor for the system. Note that the attractor lies in a bounded region. However, it does not show random behaviour at all so it is clear that deterministic chaos is different from noise or randomness.

3.1.3. Linz and Sprott System

Linz and Sprott propose the chaotic system presented in Equation 3.3 below, that exhibits chaos for $\alpha = 0.6$ and $\beta = 1$. The constant $\beta = 1$ affects only the size of the attractor. Chaos also occurs in Equation 3.3 with the nonlinear term $|x_1|$ replaced by $|x|^n$, for nonzero value of the exponent n (Linz and Sprott 1999).

$$\begin{aligned}\frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= x_3 \\ \frac{dx_3}{dt} &= -\alpha x_3 - \beta x_2 \pm (|x_1| - 1)\end{aligned}\tag{3.3}$$

This system also exhibits a period-doubling for $\alpha = 0.675, \beta = 1$. Linz has recently proved that chaos can not exist in this system if any of the terms of the equation above are set to zero (Linz and Sprott 1999).

Simulink block diagram of Linz and Sprott's chaotic system is presented in Figure 3.8. All integrator blocks are set to a zero initial condition, except block *integrator2*, which has initial condition equals to 0.000001. In the third part of Equation 3.3 above there is a \pm sign; only the negative sign is used in this Simulink realization, as can be seen in block *Add*.

Figure 3.9 presents the attractors of Linz and Sprott's chaotic system for $\beta = 1$ and for 4 different values of α equal to 0.8, 0.675, 0.644 and 0.6. The system oscillates but does not present chaotic behaviour for $\alpha = 0.8, \alpha = 0.675$ and $\alpha = 0.644$. When α equals to 0.6 then the chaotic behaviour can be seen (the values of both outputs *X1outs* and *X2outs* never cross in the phase portrait of the system). *X1outs, X2outs* and *X3outs* of Rössler equations can be seen in Figure 3.10 for $\alpha = 0.6$.

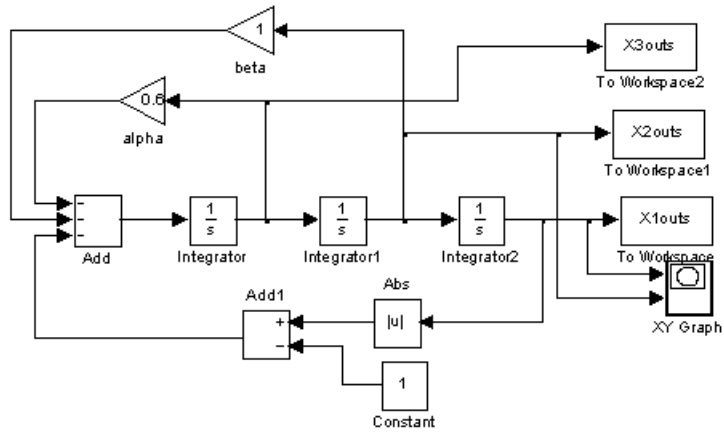


Figure 3.8. Simulink structure of Linz and Sprott equations.

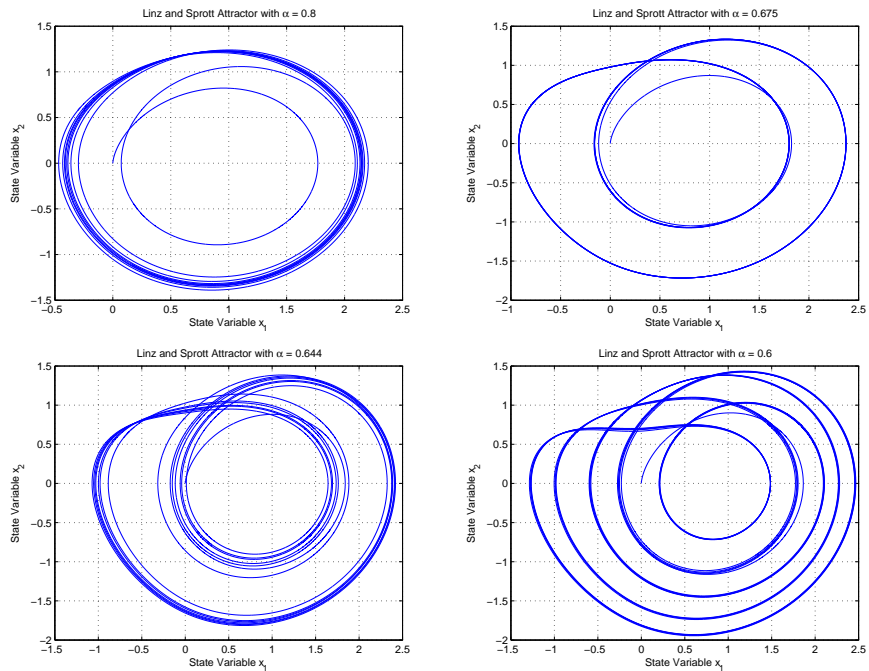


Figure 3.9. The attractor of the Linz and Sprott's Simulink block.

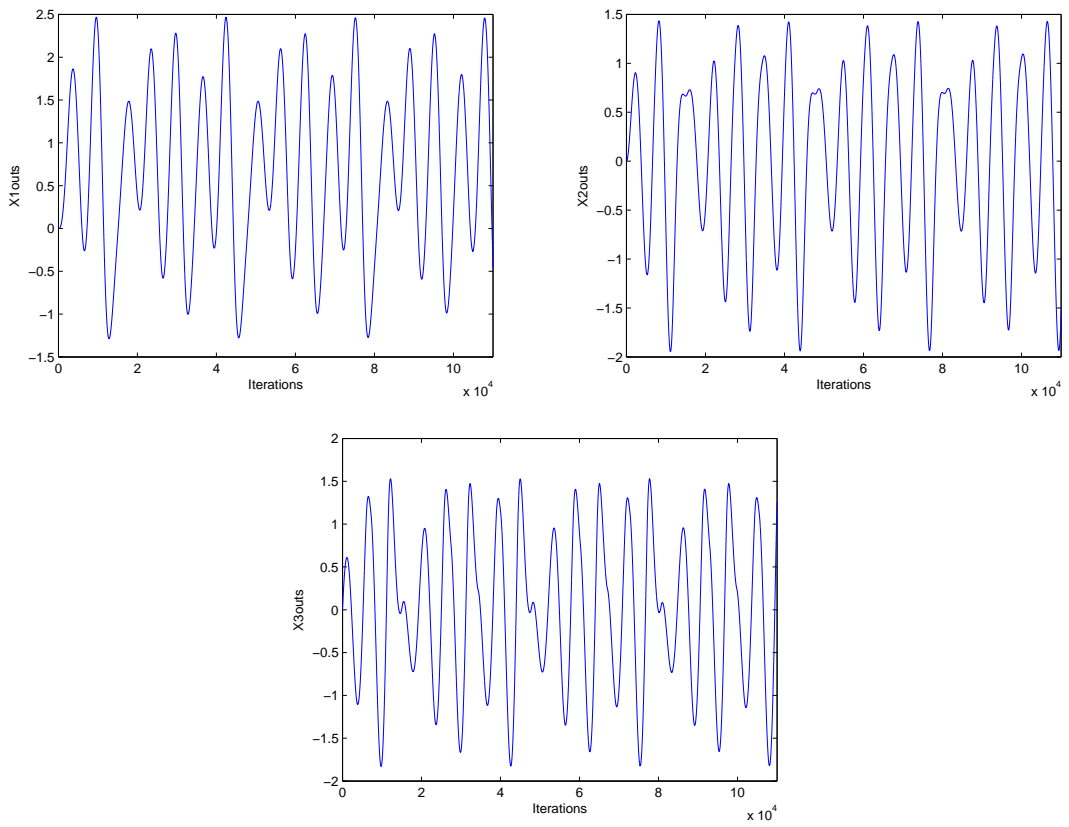


Figure 3.10. $X1outs$, $X2outs$ and $X3outs$ outputs of Linz and Sprott Equation.

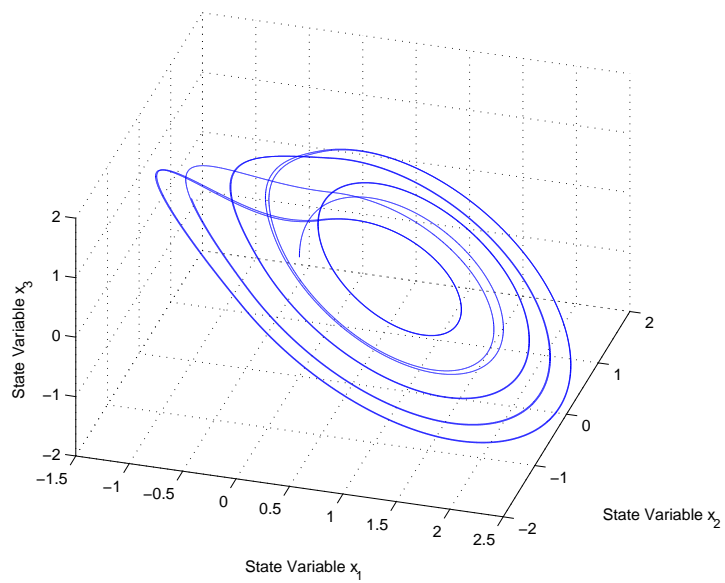


Figure 3.11. Three dimensional Linz and Sprott's attractor for $\alpha = 0.6$

3.1.4. Chua System

Chua's system is the third order circuit. It is simple since it can be constructed by 4 linear circuit elements (1 resistor, 1 inductor and 2 capacitor) and 1 nonlinear element. The circuit can be seen in Figure 3.12 (Chua, et al. 1986). The dynamic of Chua's system is described by Equation 3.4, where $g(V_R)$ is the non-linear function described by Equation 3.5. Three differential Chua's equations can be easily obtained by analysing the circuit in Figure 3.12 by Kirchhoff's law then Equations 3.4 can be obtained.

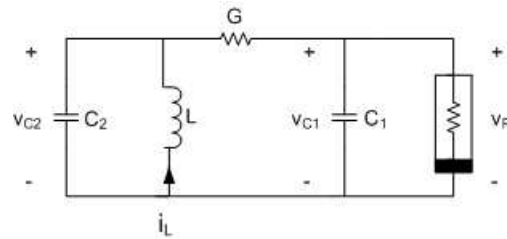


Figure 3.12. Chua's circuit.

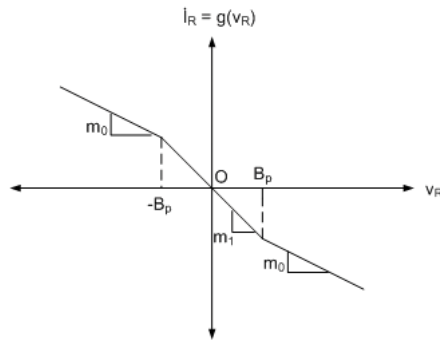


Figure 3.13. Chua's circuit non-linear characteristic.

$$\begin{aligned}
 C_1 \frac{dv_{c1}}{dt} &= G(v_{c2} - v_{c1}) - g(v_R) \\
 C_2 \frac{dv_{c2}}{dt} &= G(v_{c1} - v_{c2}) + i_L \\
 L \frac{di_L}{dt} &= -v_{c2} - R_L i_L
 \end{aligned} \tag{3.4}$$

$$g(v_R) = m_0 v_R + \frac{1}{2} (m_1 - m_0) (|v_R + B_P| - |v_R - B_P|) \tag{3.5}$$

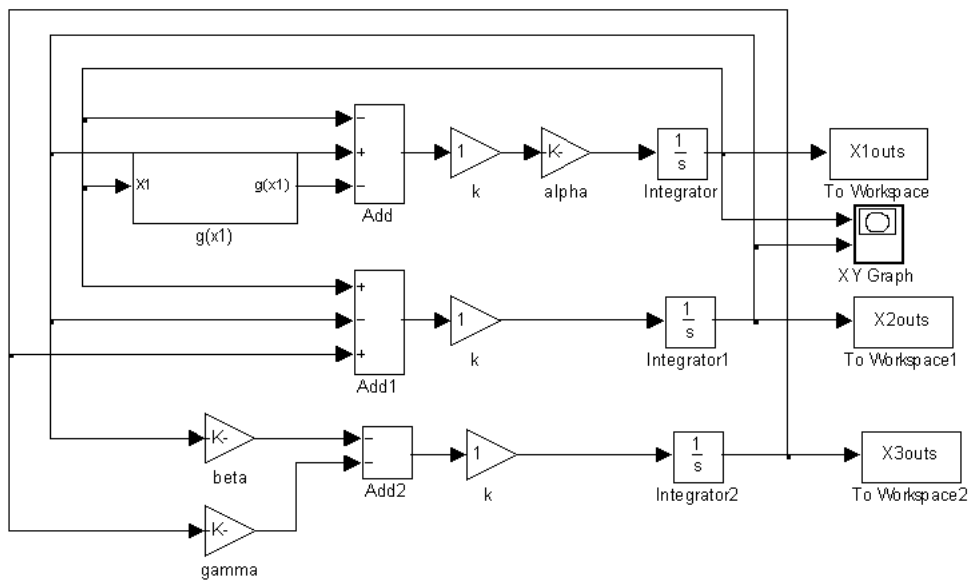


Figure 3.14. Simulink structure of Chua equations.

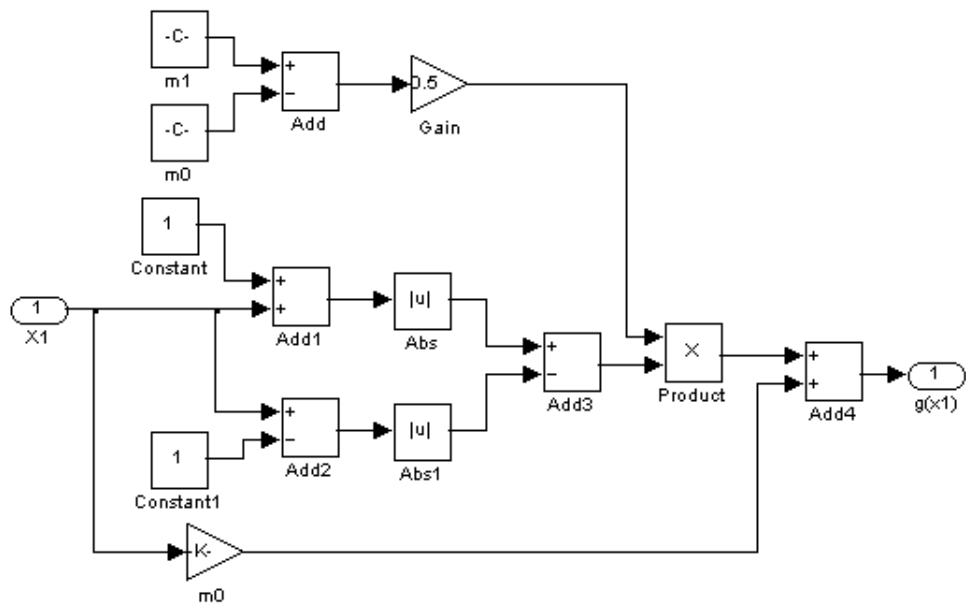


Figure 3.15. Non-linear function $g(x_1)$.

By using a change of variables as in Equation 3.6, the differential equation presented in Equation 3.4 and the nonlinear function in Equation 3.5. can be turned into the state Equations 3.7 and 3.8 respectively.

$$\begin{aligned}
 G &= 1 \\
 k &= \frac{1}{C_2} \\
 \alpha &= \frac{C_2}{C_1} \\
 \beta &= \frac{C_2}{L} \\
 \gamma &= \frac{R_L C_2}{L} \\
 v_{c_1} &= x_1 \\
 v_{c_2} &= x_2 \\
 i_L &= x_3
 \end{aligned} \tag{3.6}$$

$$\begin{aligned}
 \frac{dx_1}{d\tau} &= k\alpha(x_2 - x_1 - g(x_1)) \\
 \frac{dx_2}{d\tau} &= k(x_1 - x_2 + x_3) \\
 \frac{dx_3}{d\tau} &= k(-\beta x_2 - \gamma x_3)
 \end{aligned} \tag{3.7}$$

$$g(x_1) = m_0 x_1 + \frac{1}{2}(m_1 - m_0)(|x_1 + B_P| - |x_1 - B_P|) \tag{3.8}$$

Equations 3.7 and Equations 3.8 are simulated by means of Simulink block diagram presented in Figure 3.14 below. Note that the non-linear Equation 3.7 is indicated by the one-input / one-output block $g(x_1)$ in Figure 3.15. The block $g(x_1)$ is detailed in Figure 3.15 below.

FPGA implementation of Equations 3.7 and 3.8 have an advantage over the analog implementation of Equations 3.4 and 3.5. In FPGA implementation, it is easier to deal with negative resistances, capacitances or inductances than analog domain because it is easier to adjust the system parameters by means of System Generator blocks. 3-different parameters set were used in Figures 3.17 below that it can be seen how easy parameters

Table 3.1. Parameters used to plot the Chua's attractors

	A Parameters	B Parameters	C Parameters
α	6.579229467	-1.458906000	-1.301814000
β	10.897662619	-0.093071920	-0.013607300
γ	-0.044744029	-0.321434600	-0.02969968
m_0	-0.652335418	-0.512843600	-0.476782200
m_1	-1.811973075	1.218416000	0.169081700
k	1	-1	1

can be adjusted. Using Simulink structure of Figure 3.14, different chaotic behaviours can be simulated. Figures 3.17, shows respectively 3 different chaotic attractors of the Chua's system. These attractors have the parameters given above in Table 3.1.

$X1outs$, $X2outs$ and $X3outs$ present respectively, x_1 , x_2 and x_3 outputs of Chua's circuit in time domain.

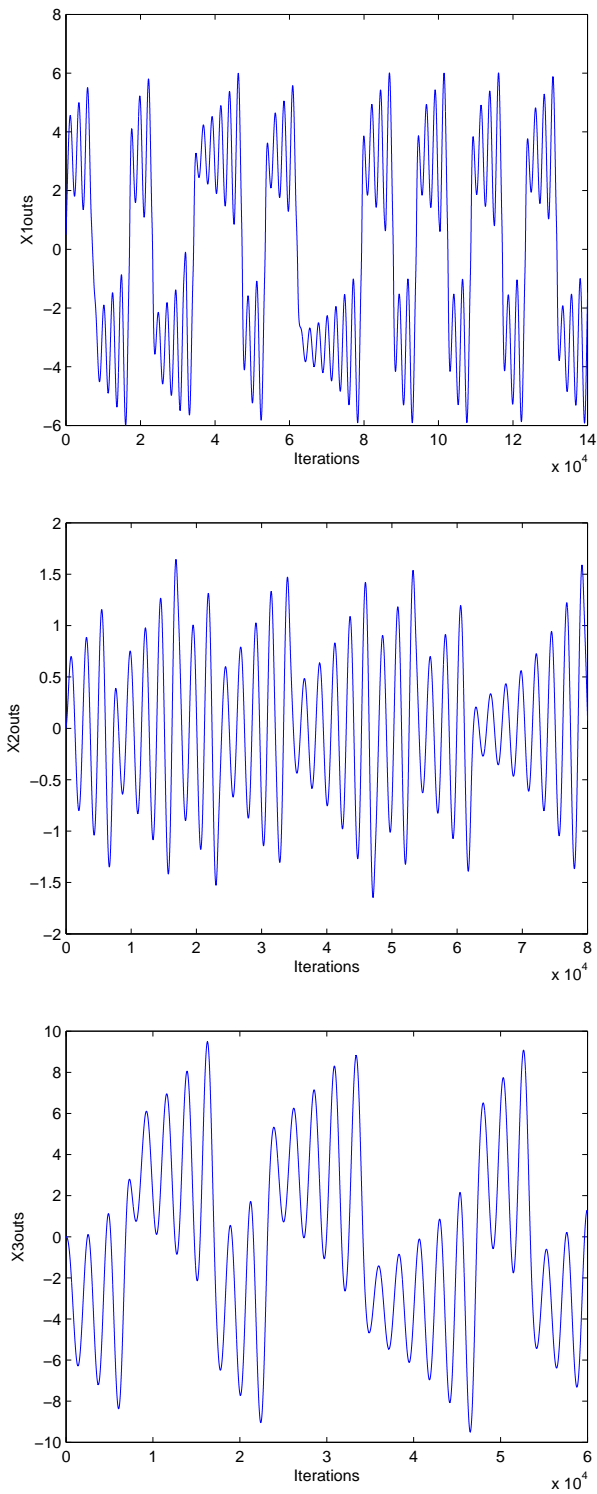


Figure 3.16. $X1outs$, $X2outs$ and $X3outs$ outputs of Chua Equation for parameter set A.

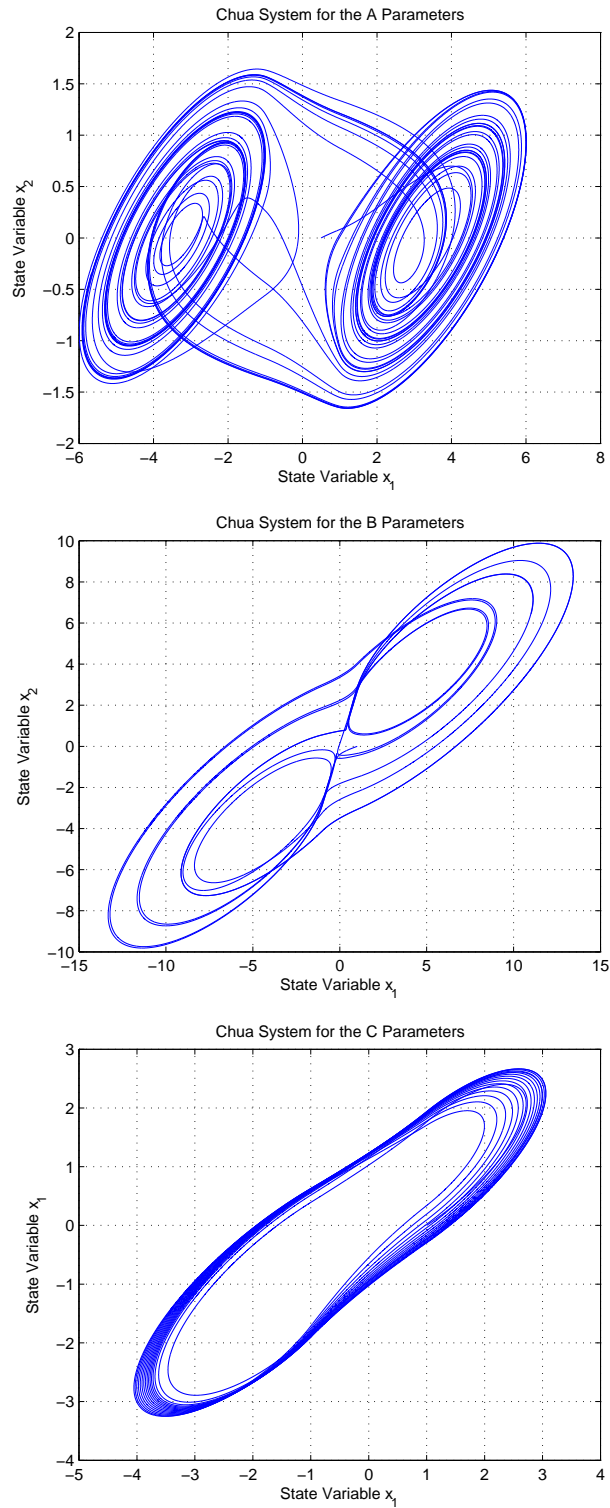


Figure 3.17. Chua's attractor with A, B and C parameters.

CHAPTER 4

SYNCHRONIZATION OF CHAOTIC SYSTEMS

Synchronization of chaos is a phenomenon that might occur when two, or more, chaotic oscillators are coupled. Due to the butterfly effect, which causes the exponential divergence of the trajectories of two identical chaotic system started with nearly the same initial conditions, having two chaotic system evolving in synchrony might appear quite surprising. However, synchronization of coupled chaotic oscillators is a phenomenon well established experimentally and reasonably understood theoretically (Wikipedia 2007b). There are various notions of chaos synchronizations such as generalized synchronization (Afraimovich, et al. 1987), complete synchronization (Pecora and Carrol 1990) and (Solis-Perales, et al. 2003), partial synchronization (Maistrenko and Popovych 2000) and phase synchronization (Rosenblum, et al. 1997) have been developed. The pioneering work (Pecora and Carrol 1990), has increased the interest in synchronization after having recently found many applications particularly in telecommunications (Abel and Schwarz 2002), in mechanical systems (Blekhman, et al. 1995) and in control theory (Nijmeijer 2001). Some different forms of synchronization between third-order chaotic systems has been studied by Femat and Solis Perales (Femat and Solis-Perales 1999). In this study, chaotic generators will be synchronized by using complete synchronizability of chaotic systems: a geometric approach (Solis-Perales, et al. 2003).

4.1. Chaos Synchronization

In this study, the chaos synchronization problem will considered as the tracking of the master system trajectories by the slave system. The difference between master and slave system is called as error system which can be constructed as in the following definition.

Definition 4.1 $\dot{\mathbf{x}} = \mathbf{F}_M(\mathbf{x})$ and $\dot{\mathbf{y}} = \mathbf{F}_S(\mathbf{y}) + \mathbf{g}(\mathbf{y})u$ be two chaotic systems in a manifold $M \subset \mathbb{R}^n$. $\mathbf{F}_M, \mathbf{F}_S$ smooth vector fields with output functions $s_M = h(\mathbf{x})$, $s_S = h(\mathbf{y})$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\mathbf{g}(\mathbf{y}) \in \mathbb{R}^n$ is a smooth input vector. The dynamical error system can be

found as $\dot{\mathbf{e}} = \mathbf{F}_M(x) - \mathbf{F}_S(y) - \mathbf{g}(y)u$. Because $\mathbf{e} = \mathbf{x} - \mathbf{y}$ then $\mathbf{y} = \mathbf{x} - \mathbf{e}$ so the dynamical error system can be written as $\dot{\mathbf{e}} = \mathbf{F}_M(x) - \mathbf{F}_S(x - e) - \mathbf{g}(x - e)u$, where $u \in \mathbb{R}$ is the control command (Solis-Perales, et al. 2003).

From Definition 4.1, synchronization error system can be written in the form as $\dot{\mathbf{e}} = \mathbf{F}_S(e) - \mathbf{g}(e)u + \Psi(x, e)$. Because $\Psi(x, e)$ depends on the solution of master system, the synchronization error system is extended so that $\dot{\mathbf{x}} = \mathbf{F}_M(x)$ can be considered. Extended synchronization error system can be written in the form as follows:

$$\dot{\mathbf{x}} = \mathbf{F}_M(x), \quad (4.1)$$

$$\dot{\mathbf{e}} = \mathbf{F}_M(x) - \mathbf{F}_S(x - e) - \mathbf{g}(x - e)u, \quad (4.2)$$

$$Y_e = h(x, e),$$

Y_e is the output of the synchronization error system. Extended synchronization error system can be written in affine form as:

$$\dot{\mathbf{X}} = \mathbf{F}(X) + \mathbf{G}(X)u, \quad (4.3)$$

where $\mathbf{X} = [\mathbf{x}, \mathbf{e}]^T$, $\mathbf{F}(X) = [\mathbf{F}_M, \mathbf{F}_M - \mathbf{F}_S]^T$ and $\mathbf{G}(X) = [\mathbf{0}, -\mathbf{g}]$. In this study, complete synchronization is considered (Solis-Perales, et al. 2003), which means that the slave and the master system display the same pattern at the same time. In order to achieve complete synchronization, synchronization error system in Equation 4.2 should be stabilized around the point $e^* = 0$.

Synchronization error system in Equation 4.2 can be stabilized at the origin by means of the control command u . If such control command is found, complete synchronization of chaotic system can be achieved. The proper control command can be found by using the geometrical tools. These tools are the properties of controllability and observability of non-linear affine systems (Nijmeijer, van der Schaft 1990). The controllability and observability of chaotic systems will be explained in sections 4.2 and 4.3 respectively.

4.2. Synchronizability from Control of Chaotic Systems

A definition for synchronizability can be considered for complete synchronization between chaotic systems.

Definition 4.2 $\exists u = u(x_M - x_S) \in \mathbb{R}$ such that $|x(t) - y(t)| \approx 0$, with $x, y \in \mathbb{R}^n$ for all $t > t^* < \infty$ and any initial conditions $x_0 = x(t = 0)$ and $y_0 = y(t = 0)$ belonging to that manifold $\mathbf{M} \subset \mathbb{R}^{2n}$ (Solis-Perales, et al. 2003).

The controllability and observability conditions can be used in order to find a tangent space at stabilization point $e = 0$. The vector \mathbf{G} can be calculated such that it generates a tangent space with constant dimension at $(x, 0)^T$. As previously defined $Y_e = h(x, e)$ which is an output function of synchronization error system. The computation of Y_e involves conditions for synchronizability of chaotic systems which will be shown in following sections.

4.2.1. Local Controllability for Complete Synchronizability

We start this section by giving Lemma 4.1 as following below:

Lemma 4.1 $C(x, e)$ be the accessibility distribution with constant dimension d at $e = 0$ then system (Equation 4.3) is locally accessible (locally controllable) and the controllable space has dimension d (Nijmeijer, van der Schaft 1990).

The local coordinate transformation can be found such that system in Equation 4.3 can be partially or completely transformed into a linear controllable system around $(x, 0)$ by taking into account 4.1. In order for local coordinate transformation, accessibility distribution needs to be found. In order to find accessibility distribution function, Lie brackets and Lie derivative are needed to be known. Definition 4.3 given below explains Lie brackets and Lie derivative respectively.

Definition 4.3 Let \mathbf{F} and \mathbf{G} be two vector fields in a manifold \mathbf{M} , the vector field $[\mathbf{F}, \mathbf{G}](\mathbf{X}) = \frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G}$ is called the Lie bracket of \mathbf{F} and \mathbf{G} . The Lie derivative of a real-valued function $h(x, e)$ along the vector field \mathbf{F} is defined as $L_{\mathbf{F}}h(x, e) = \langle \partial h, \mathbf{F} \rangle$ (Vidyasagar 1993).

The accessibility distribution $\mathbf{C}(x, e)$ can be expressed as $\mathbf{C}_d = \text{span}\{ad_{\mathbf{F}}^{d-1}\mathbf{G}\}$ where $ad_{\mathbf{F}} = [\mathbf{F}, \mathbf{G}]$ and $ad_{\mathbf{F}}^{d-1} = [\mathbf{F}, [\mathbf{F}, [\dots, [\mathbf{F}, \mathbf{G}], \dots,]]]$ for $d = 1, \dots, n$.

Remark 4.2 Equation 4.3 is composed of two subsystems (Equations 4.1 and Equations 4.2) the tangent space locally generated at $(x, 0)^T$ has dimension $d \leq n$. \mathbf{C}_d is generated

by d linearly independent vector fields that span of dimension $d \leq n$ because Equations 4.1 are unidirectionally coupled with Equations 4.2.

There are two restrictions concerning the controllability for synchronizability of master/slave systems:

- (i) The constant dimension $Dim(\mathbf{C}_d(x, 0)) = d$
- (ii) The distribution \mathbf{C}_{d-1} should be involutive.

At the end there is a Proposition 4.1 given by (Solis-Perales, et al. 2003) as follows

Proposition 4.1 *Suppose that the internal dynamics is stable and assume that accessibility distribution \mathbf{C} has constant dimension d (i.e., \mathbf{C} is involutive) in a neighborhood U^0 of $(x, 0)$, then system is locally controllable.*

4.2.2. Local Observability for Complete Synchronizability

The output function $Y_e = h(x, e)$, where h is real-valued function of the system in Equation 4.3. It is related to the observability of the synchronization system. Let us consider the following Definition 4.4 (Nijmeijer, van der Schaft 1990).

Definition 4.4 *Consider an affine system $\dot{X} = F(X) + G(X)u$, with an output function $Y_e = h(x)$. It is said that the system has relative degree ρ at x^0 if*

- (i) $L_G L_F^k h(x) = 0$ for all x in a neighborhood of x^0 and $k \leq \rho - 1$,
- (ii) $L_G L_F^{\rho-1} h(x^0) \neq 0$.

If the system in Equation 4.3 has relative degree ρ at $(x, 0)^T$, the Lie derivatives of the output function are given by $h(x), L_F h(x), L_F^2 h(x), \dots, L_F^{\rho-1} h(x)$, and the covector fields $\partial h(x), \partial L_F h(x), \partial L_F^2 h(x), \dots, \partial L_F^{\rho-1} h(x)$, which are independent in the neighborhood U of $(x, 0)^T$ (Nijmeijer, van der Schaft 1990). In this way, by using the Lie derivatives of $h(x, e)$, a local coordinate transformation can be defined at $(x, 0)^T$.

Such a transformation $z = \Phi(x, e)$ is defined as $\Phi_1 = h(x), \Phi_2 = L_F h(x), \dots, \Phi_\rho = L_F^{\rho-1} h(x)$ and the $2n - \rho$ complementary functions are obtained by finding the Jacobian matrix of $z = [\Phi_1, \Phi_2, \dots, \Phi_n, \dots, \Phi_{2n}]^T$ be nonsingular at $(x, 0)$ and $L_G \Phi_i(x, e) = 0$, with $i = \rho + 1, \dots, 2n$ and (x, e) in a neighborhood of $(x, 0)$ (Solis-Perales, et al. 2003).

Proposition 4.2 *Suppose that the system in Equation 4.3 has relative degree ρ at $(x, 0)$ and the codistribution $\ker(\text{span}\{\partial h(x, 0), \partial L_F h(x, 0), \partial L_F^2 h(x, 0), \dots, \partial L_F^{\rho-1} h(x, 0)\})$ has dimension ρ , then the system in Equation 4.3 is locally observable at $(x, 0)$ (Solis-Perales, et al. 2003).*

4.3. Complete Synchronizability

If Propositions 4.1 and 4.2 hold for complete synchronizability for Equation 4.3 then that system is linearizable by feedback. By using this property, an invertible coordinates transformation and a control command can be calculated. The Theorem 4.3 which was proposed by (Solis-Perales, et al. 2003) comprises stabilizing conditions mentioned before.

Theorem 4.3 *Consider the system in Equation 4.3. Suppose that there exist $2n - \rho$ functions $\Phi_i(x, e)$ such that $L_G \Phi_i(x, e) = 0$, $i = \rho + 1, \dots, 2n$. This system is feedback linearizable at $(x, 0)$ if and only if there exists a function $h(x, e)$ such that*

(i) $\langle \partial h, \text{ad}_F^{k-1} \mathbf{G} \rangle(x, e) = 0$ for $k = 1, \dots, \rho - 1$; $\rho > 1$ and (x, e) in a neighborhood U of $(x, 0)$,

(ii) $\langle \partial h, \text{ad}_F^i \mathbf{G} \rangle(x, 0) \neq 0$ for $i = \rho, \dots, n$ at $(x, 0)$,

where $\rho = d$ stands for the dimension of the tangent space.

Theorem 4.3 satisfies sufficient and necessary conditions for complete synchronizability of the same order systems. In addition to that, conditions (i) and (ii) in Theorem 4.3 can be used to calculate proper output function in order to make the system in Equation 4.3 locally observable at $(x, 0)$. Then there is a Corollary 4.4 as given below by (Solis-Perales, et al. 2003).

Corollary 4.4 *Two chaotic systems with the same order are completely synchronizable if and only if the dynamical error system is feedback linearizable at $(x, 0)$.*

The system Equation 4.3 is called feedback linearizable if there exist a smooth reversible change of coordinates $z = \Phi(x, e)$ and smooth transformation of the feedback (Andrievskii and Fradkov 2003)

$$u = \lambda(x, e) + \mu(x, e)v, \quad (4.4)$$

where $v \in \mathbb{R}^m$ is the new control if the closed-loop is linear. If the linearizability criterion is satisfied, then by means of transformations

$$\begin{aligned} z &= \Phi(x, e) = \text{col}(h(x, e), L_F h(x, e), \dots, L_F^{\rho-1} h(x, e)) \\ u &= \frac{1}{L_G L_F^{\rho-1} h(x, e)} (-L_F^\rho h(x, e) + v) \end{aligned} \quad (4.5)$$

$$\lambda(x, e) = \frac{-L_F^\rho h(x, e)}{L_G L_F^{\rho-1} h(x, e)} \quad (4.6)$$

$$\mu(x, e) = \frac{1}{L_G L_F^{\rho-1} h(x, e)} \quad (4.7)$$

$$\nu = K_i(z_i - z_i^*) \quad (4.8)$$

where K_i with $i = 1, \dots, \rho$ are the control gains and chosen in such a way that the closed-loop subsystem \dot{z} converges to origin and z_i^* 's are the coordinates of the stabilization point. In order to achieve complete synchronization z_i^* 's are set to zero.

Complete synchronization approach will be applied to chaotic generators and then by using Simulink in MATLAB, results will be seen in following sections.

4.4. Synchronization of Lorenz System

Firstly, Lorenz system will be examined in order to see whether complete synchronization approach is valid for this system. For simplicity, the master and slave systems are considered as they have the same parameter values,

$$\dot{x} = \mathbf{F}_M(x)$$

$$\begin{aligned} \dot{x}_1 &= \alpha(x_2 - x_1) \\ \dot{x}_2 &= \beta x_1 - x_2 - x_1 x_3 \\ \dot{x}_3 &= x_1 x_2 - \gamma x_3 \end{aligned} \quad (4.9)$$

and $\dot{y} = \mathbf{F}_S(y) + \mathbf{g}(y)u$

$$\begin{aligned} \dot{y}_1 &= \alpha(y_2 - y_1) + g_1(y)u \\ \dot{y}_2 &= \beta y_1 - y_2 - y_1 y_3 + g_2(y)u \\ \dot{y}_3 &= y_1 y_2 - \gamma y_3 + g_3(y)u \end{aligned} \quad (4.10)$$

where the parameters $\alpha = 10$, $\beta = 28$ and $\gamma = 8/3$. $g_1(y)$, $g_2(y)$ and $g_3(y)$ are the corresponding elements in the input vector of the slave system. The difference between master and slave system are found by calculating $e = x - y$ then it is easily seen that $\dot{e} = \dot{x} - \dot{y}$.

$$\begin{aligned}
\dot{e}_1 &= \alpha[(x_2 - y_2) - (x_1 - y_1)] - g_1(x - e)u \\
\dot{e}_2 &= \beta(x_1 - y_1) - (x_2 - y_2) + [x_1x_3 - (x_1 - e_1)(x_3 - e_3)] - g_2(x - e)u \\
\dot{e}_3 &= x_1x_2 - [(x_1 - e_1)(x_2 - e_2)] - \gamma(x_3 - y_3) - g_3(x - e)u
\end{aligned} \tag{4.11}$$

From Definition 4.1 extended synchronization error system can be calculated as follows:

$$\begin{aligned}
\dot{x}_1 &= \alpha(x_2 - x_1) \\
\dot{x}_2 &= \beta x_1 - x_2 - x_1x_3 \\
\dot{x}_3 &= x_1x_2 - \gamma x_3 \\
\dot{e}_1 &= \alpha(e_2 - e_1) - g_1(x - e)u \\
\dot{e}_2 &= \beta e_1 - e_2 + e_1e_3 - e_1x_3 - x_1e_3 - g_2(x - e)u \\
\dot{e}_3 &= -e_1e_2 - \gamma e_3 + e_1x_2 + x_1e_2 - g_3(x - e)u
\end{aligned} \tag{4.12}$$

This is $\dot{e} = \mathbf{F}_M(x) - \mathbf{F}_S(y) - \mathbf{g}(x - e)u$. Equation 4.12 can be written in affine form as: $\dot{\mathbf{X}} = \mathbf{F}(X) + \mathbf{G}(X)u$, where $\mathbf{X} = [x, e]^T$, $\mathbf{F}(X) = [\mathbf{F}_M, \mathbf{F}_M - \mathbf{F}_S]^T$ and $\mathbf{G}(X) = [\mathbf{0}, -\mathbf{g}(x - e)]^T$.

$$\mathbf{F} = \begin{bmatrix} \alpha(x_2 - x_1) \\ \beta x_1 - x_2 - x_1x_3 \\ x_1x_2 - \gamma x_3 \\ \alpha(e_2 - e_1) \\ \beta e_1 - e_2 + e_1e_3 - e_1x_3 - x_1e_3 \\ -e_1e_2 - \gamma e_3 + e_1x_2 + x_1e_2 \end{bmatrix} \tag{4.13}$$

$$\mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1(x-e) \\ -g_2(x-e) \\ -g_3(x-e) \end{bmatrix} \quad (4.14)$$

In order to achieve complete synchronization, conditions of Theorem 4.3 will be considered. Because subsystem Equations 4.11 has three state variable so $k = 1, \dots, 3$ then, $\mathbf{C}_1(x, e)$, $\mathbf{C}_2(x, e)$ and $\mathbf{C}_3(x, 0)$ has to be calculated. The distributions are $\mathbf{C}_1(x, e) = \text{span}\{\mathbf{G}\}$, $\mathbf{C}_2(x, e) = \text{span}\{\mathbf{G}, \text{ad}_F \mathbf{G}\}$ and $\mathbf{C}_3(x, e) = \text{span}\{\mathbf{G}, \text{ad}_F \mathbf{G}, \text{ad}_F^2 \mathbf{G}\}$. By choosing $\mathbf{G} = [0 \ 0 \ 0 \ -g_1 \ -g_2 \ -g_3]^T$ with g_1, g_2 and g_3 as constants. In order to obtain $\mathbf{C}_2(x, e)$ and $\mathbf{C}_3(x, e)$, at first $\mathbf{C}_1(x, e)$ has to be written down as follows:

$$\mathbf{C}_1(x, e) = \text{span} \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix} \right\} \quad (4.15)$$

In order to find $\mathbf{C}_2(x, e)$, $\text{ad}_F \mathbf{G} = [\mathbf{F}, \mathbf{G}]$ should be calculated. From Definition 4.3, $\text{ad}_F \mathbf{G}$ can be written as follows:

$$\text{ad}_F \mathbf{G} = [\mathbf{F}, \mathbf{G}] = \frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G} \quad (4.16)$$

The term $\frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} = 0$ because \mathbf{G} is constant vector. So Equation 4.16 turns into simpler form such that

$$\text{ad}_F \mathbf{G} = [\mathbf{F}, \mathbf{G}] = -\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G} \quad (4.17)$$

So as to find $\text{ad}_F \mathbf{G}$, $\frac{\partial \mathbf{F}}{\partial \mathbf{X}}$ can be calculated as in Equation 4.18

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} = \begin{bmatrix} -\alpha & \alpha & 0 & 0 & 0 & 0 \\ (\beta - x_3) & -1 & -x_1 & 0 & 0 & 0 \\ x_2 & x_1 & -\gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & -\alpha & \alpha & 0 \\ -e_3 & 0 & -e_1 & (\beta + e_3 - x_3) & -1 & (e_1 - x_1) \\ e_2 & e_1 & 0 & (x_2 - e_2) & (x_1 - e_1) & -\gamma \end{bmatrix} \quad (4.18)$$

Multiplying Equation 4.18 by $-\mathbf{G}$ then $ad_F \mathbf{G}$ can be obtained easily as in Equation 4.19

$$-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G} = - \begin{bmatrix} -\alpha & \alpha & 0 & 0 & 0 & 0 \\ (\beta - x_3) & -1 & -x_1 & 0 & 0 & 0 \\ x_2 & x_1 & -\gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & -\alpha & \alpha & 0 \\ -e_3 & 0 & -e_1 & (\beta + e_3 - x_3) & -1 & (e_1 - x_1) \\ e_2 & e_1 & 0 & (x_2 - e_2) & (x_1 - e_1) & -\gamma \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}.$$

$$ad_F \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \alpha(g_2 - g_1) \\ g_1(\beta + e_3 - x_3) - g_2 + g_3(e_1 - x_1) \\ g_1(x_2 - e_2) + g_2(x_1 - e_1) - \gamma g_3 \end{bmatrix} \quad (4.19)$$

$\mathbf{C}_2(x, e)$ can be written as in Equation 4.20 such that

$$\mathbf{C}_2(x, e) = span \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \alpha(g_2 - g_1) \\ g_1(\beta + e_3 - x_3) - g_2 + g_3(e_1 - x_1) \\ g_1(x_2 - e_2) + g_2(x_1 - e_1) - \gamma g_3 \end{bmatrix} \right\} \quad (4.20)$$

So as to find $\mathbf{C}_3(x, e)$ is that the only equation left to be needed to calculate is $ad_F^2 \mathbf{G}$ because $span\{\mathbf{G}\}$ and $span\{ad_F \mathbf{G}\}$ have already been found.

$$ad_F^2 \mathbf{G} = [\mathbf{F}, ad_F \mathbf{G}] \quad (4.21)$$

For simplicity, $ad_F \mathbf{G} = \Psi$ is chosen so Equation 4.21 can be written in form:

$$ad_F^2 \mathbf{G} = [\mathbf{F}, ad_F \mathbf{G}] = [\mathbf{F}, \Psi] = \frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi \quad (4.22)$$

$\frac{\partial \Psi}{\partial \mathbf{X}}$ is can be found as in Equation 4.23

$$\frac{\partial \Psi}{\partial \mathbf{X}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -g_3 & 0 & -g_1 & g_3 & 0 & g_1 \\ g_2 & g_1 & 0 & -g_2 & -g_1 & 0 \end{bmatrix} \quad (4.23)$$

Multiplying Equation 4.23 by \mathbf{F} , the first term of the $ad_F^2 \mathbf{G}$ can be found as follows:

$$\begin{aligned} \frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -g_3 & 0 & -g_1 & g_3 & 0 & g_1 \\ g_2 & g_1 & 0 & -g_2 & -g_1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha(x_2 - x_1) \\ \beta x_1 - x_2 - x_1 x_3 \\ x_1 x_2 - \gamma x_3 \\ \alpha(e_2 - e_1) \\ \beta e_1 - e_2 + e_1 e_3 - e_1 x_3 - x_1 e_3 \\ -e_1 e_2 - \gamma e_3 + e_1 x_2 + x_1 e_2 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ g_3 \alpha(x_1 - x_2) + g_1(\gamma x_3 - x_1 x_2) + g_3 \alpha(e_2 - e_1) + g_1(-e_1 e_2 - \gamma e_3 + e_1 x_2 + x_1 e_2) \\ g_2 \alpha(x_2 - x_1) + g_1(\beta x_1 - x_2 - x_1 x_3) + g_2 \alpha(e_1 - e_2) + g_1(e_1 x_3 + x_1 e_3 + e_2 - \beta e_1 - e_1 e_3) \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ g_1[\gamma x_3 - x_1 x_2 - e_1 e_2 - \gamma e_3 + e_1 x_2 + x_1 e_2] + g_3 \alpha(x_1 - x_2 + e_2 - e_1) \\ g_1[\beta x_1 - x_2 - x_1 x_3 - \beta e_1 + e_2 - e_1 e_3 + e_1 x_3 + x_1 e_3] + g_2 \alpha(x_2 - x_1 + e_1 - e_2) \end{bmatrix} \end{aligned} \quad (4.24)$$

In order to obtain the second term of $ad_F^2 \mathbf{G}$, $\frac{\partial \mathbf{F}}{\partial \mathbf{X}}$ can be written as in Equation 4.25.

$$\frac{\partial \mathbf{F}}{\partial \mathbf{X}} = \begin{bmatrix} -\alpha & \alpha & 0 & 0 & 0 & 0 \\ (\beta - x_3) & -1 & -x_1 & 0 & 0 & 0 \\ x_2 & x_1 & -\gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & -\alpha & \alpha & 0 \\ -e_3 & 0 & -e_1 & (\beta + e_3 - x_3) & -1 & (e_1 - x_1) \\ e_2 & e_1 & 0 & (x_2 - e_2) & (x_1 - e_1) & -\gamma \end{bmatrix} \quad (4.25)$$

Multiplying Equation 4.25 by $-\Psi$ we obtain the second term of $ad_F^2 \mathbf{G}$ such that

$$\begin{aligned} & -\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi = \\ & - \begin{bmatrix} -\alpha & \alpha & 0 & 0 & 0 & 0 \\ (\beta - x_3) & -1 & -x_1 & 0 & 0 & 0 \\ x_2 & x_1 & -\gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & -\alpha & \alpha & 0 \\ -e_3 & 0 & -e_1 & (\beta + e_3 - x_3) & -1 & (e_1 - x_1) \\ e_2 & e_1 & 0 & (x_2 - e_2) & (x_1 - e_1) & -\gamma \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ \alpha(g_2 - g_1) \\ g_1(\beta + e_3 - x_3) - g_2 + g_3(e_1 - x_1) \\ g_1(x_2 - e_2) + g_2(x_1 - e_1) - \gamma g_3 \end{bmatrix} \\ & = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \alpha^2(g_2 - g_1) - \alpha[g_1(\beta + e_3 - x_3) - g_2 + g_3(e_1 - x_1)] \\ (x_3 - e_3 - \beta)\alpha(g_2 - g_1) + g_1(\beta + e_3 - x_3) - g_2 + g_3(e_1 - x_1) + (x_1 - e_1)[g_1(x_2 - e_2) + g_2((x_1 - e_1) - \gamma g_3)] \\ (e_2 - x_2)\alpha(g_2 - g_1) + (e_1 - x_1)[g_1(\beta + e_3 - x_3) - g_2 + g_3(e_1 - x_1)] + \gamma[g_1(x_2 - e_2) + g_2(x_1 - e_1) - \gamma g_3] \end{bmatrix} \\ & -\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_1[-\alpha^2 - \alpha(\beta + e_3 - x_3)] + g_2(\alpha^2 + \alpha) + g_3\alpha(x_1 - e_1) \\ g_1[(\alpha + 1)(\beta + e_3 - x_3) + (x_1 - e_1)(x_2 - e_2)] + g_2[-\alpha(\beta + e_3 - x_3) - 1 + (x_1 - e_1)^2] + g_3[(e_1 - x_1)(\gamma + 1)] \\ g_1[(\alpha + \gamma)(x_2 - e_2) + (e_1 - x_1)(\beta + e_3 - x_3)] + g_2[(\alpha e_2 - x_2) + (\gamma + 1)(x_1 - e_1)] + g_3[(e_1 - x_1)^2 - \gamma^2] \end{bmatrix} \quad (4.26) \end{aligned}$$

Combining Equations 4.24 and 4.26 we have an $ad_F^2 \mathbf{G}$ such that

$$ad_F^2 \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_1[-\alpha^2 - \alpha(\beta + e_3 - x_3)] + g_2(\alpha^2 + \alpha) + g_3\alpha(x_1 - e_1) \\ g_1[\gamma(x_3 - e_3) + (\alpha + 1)(\beta + e_3 - x_3)] + g_2[(x_1 - e_1)^2 - \alpha(\beta + e_3 - x_3) - 1] + g_3[(e_1 - x_1)(1 + \gamma - \alpha) + \alpha(e_2 - x_2)] \\ g_1[(x_2 - e_2)(\alpha + \gamma - 1)] + g_2[(x_1 - e_1)(1 + \gamma - \alpha)] + g_3[(e_1 - x_1)^2 - \gamma^2] \end{bmatrix} \quad (4.27)$$

Note that the dimension of $\mathbf{C}_3(x, 0)$ is $d \leq 3$. Without lost of generality, \mathbf{G} is considered as $\mathbf{G} = [0 \ 0 \ 0 \ 0 \ -1 \ 0]^T$ by chosing $g_1 = g_3 = 0$ and $g_2 = 1$ so $\mathbf{C}_3(x, 0) = span\{\mathbf{G}, ad_F \mathbf{G}, ad_F^2 \mathbf{G}\}$ is written as follows:

$$\mathbf{C}_3(x, 0) = \text{span} \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \alpha & \alpha^2 + \alpha \\ -1 & -1 & [x_1^2 + \alpha(x_3 - \beta) - 1] \\ 0 & x_1 & x_1(1 + \gamma - \alpha) \end{bmatrix} \right\} \quad (4.28)$$

In order to achieve complete synchronization, the accessibility distribution \mathbf{C} has constant dimension which can be satisfied if below conditions from Theorem 4.3 hold.

$$\langle dh, \mathbf{G} \rangle(x, e) = 0 \quad (4.29)$$

$$\langle dh, \text{ad}_F \mathbf{G} \rangle(x, e) = 0 \quad (4.30)$$

$$\langle dh, \text{ad}_F^2 \mathbf{G} \rangle(x, 0) \neq 0 \quad (4.31)$$

In order to satisfy Equation 4.29 that the output function $h(x, e)$ can not depend on e_2 . In order to satisfy Equation 4.30 the output function can not depend on e_3 since if it depends on e_3 there is state variable x_1 and it can not be controlled. There is one alternative left which is $h(x, e) = e_2$ but if we chose output function depends on e_2 then Equation 4.31 is not hold since it also includes state variables x_1 and x_3 . As a consequence there is no output function which satisfies Equations 4.29, 4.30 and 4.31 at the same time. As a result $\mathbf{C}_3(x, 0)$ has variable dimension. From this result, we have found that two Lorenz systems can not generate a tangent space of dimension $d = 3$ with an input vector of constant elements. So we try to determine a tangent space of constant dimension $d = 2$. Now from Theorem 4.3 again, proper output function $h(x, e)$ can be found such that

$$-\frac{\partial h}{\partial e_2} = 0, \quad (4.32)$$

$$\alpha \frac{\partial h}{\partial e_1} - \frac{\partial h}{\partial e_2} + x_1 \frac{\partial h}{\partial e_3} \neq 0. \quad (4.33)$$

A possible function which satisfies Equations 4.32 and 4.33 is $h(x, e) = e_1$. Once we have an output function, the relative degree is calculated and for this case is $d = \rho = \text{Dim}(\mathbf{C}_3(x, 0)) = 2$ for all $x \in \mathbb{R}^3$. So we look for an invertible transformation for this output function. In order to have an relative degree $\rho = 2$ from Definition 4.4 so that the

system can be observable as long as below conditions hold

$$L_G h(x, e) = 0 \quad (4.34)$$

$$L_G L_F h(x, e) \neq 0 \quad (4.35)$$

These are calculated as below:

$$L_G h(x, e) = \langle \partial h, \mathbf{G} \rangle (x, w) = 0 \quad (4.36)$$

$$L_F h(x, e) = \langle \partial h, \mathbf{F} \rangle (x, w) = \alpha(e_2 - e_1) \frac{\partial e_1}{\partial e_1} = \alpha(e_2 - e_1) \quad (4.37)$$

$$L_G L_F h(x, e) = L_G [\alpha(e_2 - e_1)] = \alpha(-1 \frac{\partial e_2}{\partial e_2}) = -\alpha \quad (4.38)$$

$L_G h(x, e) = 0, L_G L_F h(x, e) = -\alpha$ so for $\rho = 2$ all requirements hold so there is a control command which can be written from Equation 4.5 for $\rho = 2$ as in Equation 4.57

$$u = \frac{1}{L_G L_F h(x, e)} (-L_F^2 h(x, e) + \mathbf{K}_1(z_1 - z_1^*) + \mathbf{K}_2(z_2 - z_2^*)) \quad (4.39)$$

From Equation 4.38, $L_G L_F h(x, e) = -\alpha$

From Equation 4.37, $L_F h(x, e) = \alpha(e_2 - e_1)$

Then, $L_F^2 h(x, e) = L_F [\alpha(e_2 - e_1)] = \alpha[(\beta e_1 - e_2 + e_1 e_3 - e_1 x_3 - x_1 e_3) \frac{\partial e_2}{\partial e_2} - \alpha(e_2 - e_1) \frac{\partial e_1}{\partial e_1}]$

So the control command u can be written down as follows:

$$u = \frac{1}{-\alpha} \{-\alpha[\beta e_1 - e_2 + e_1 e_3 - e_1 x_3 - x_1 e_3 - \alpha(e_2 - e_1)] + \mathbf{K}_1(z_1 - z_1^*) + \mathbf{K}_2(z_2 - z_2^*)\}$$

In this system $\rho = 2$ then $i = 1, 2$. So we have \mathbf{K}_1 and \mathbf{K}_2 which are the control gains and are chosen so that the system \dot{z} converges to origin. And z_1^* and z_2^* are the desired stabilization point and by choosing $z_1^* = z_2^* = 0$ we make the system converges to origin.

By choosing $\mathbf{K}_1 = -9$ and $\mathbf{K}_2 = -6$ and setting $z_1^* = z_2^* = 0$. We know that $z_1 = e_1$ and $z_2 = \alpha(e_2 - e_1)$ so we have the control command u such that

$$u = \frac{1}{-\alpha} \{-\alpha[\beta e_1 - e_2 + e_1 e_3 - e_1 x_3 - x_1 e_3 - \alpha(e_2 - e_1)] - 9e_1 - 6(e_2 - e_1)\} \quad (4.40)$$

In order to understand whether our control signal u is proper for Lorenz system to synchronize, the results from Simulink will be seen. As we know from section 3.3.1 initial

condition of master system's integrator block equals to 0.001. Since chaotic systems are so sensitive to initial condition by setting initial condition of slave system's integrator block equals to 5 the difference between master and slave system can be created. At first, the control command u can be built by Simulink blocks as in Figure 4.1.

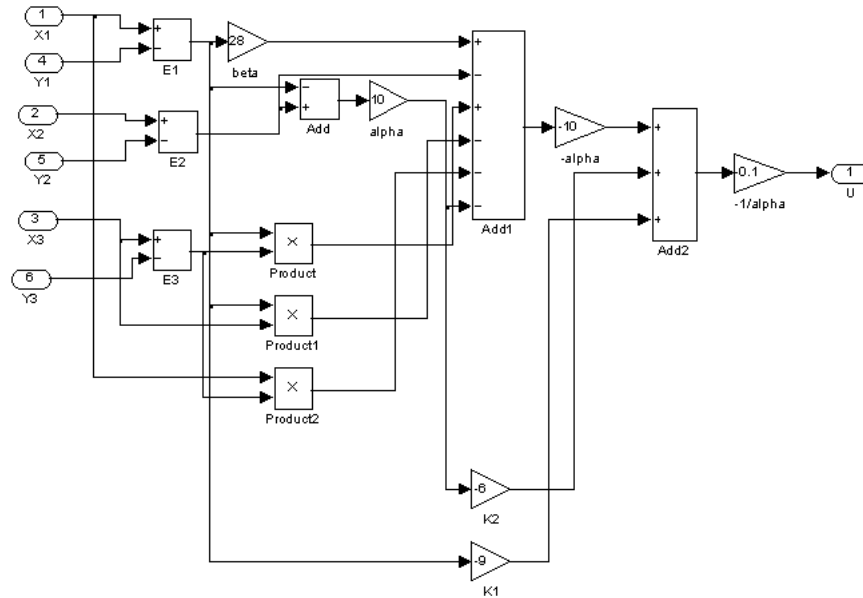


Figure 4.1. Simulink block diagram of the control command u for Lorenz system.

After constructing the control command u by adding it to Lorenz slave system, the synchronized Lorenz system will be obtained as in Figure 4.2. Lorenz system block diagram can be seen in section 3.3.1 so slave system can be constructed by using that block diagram. The only difference between master system and slave system is that there is a control command in slave system as opposed to master system.

As can be seen in Figure 4.3 master and slave system attractors draw the same pattern even if they have different initial conditions. In Figure 4.4, the control command u , error signals e_1 , e_2 and e_3 can be seen respectively. As can be seen in Figure 4.4, error signals reach to zero immediately which implies that Lorenz slave system exactly tracks Lorenz master system. Because control command depends on e_1 , e_2 and e_3 then it also reaches to zero. After it reaches to zero the slave system tracks master system

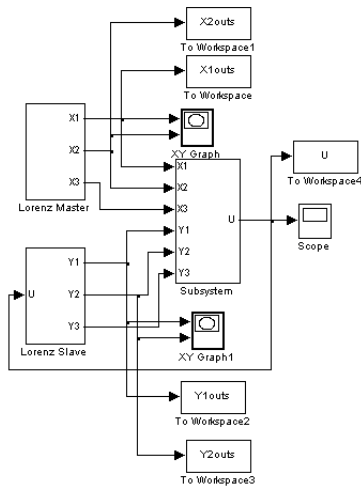


Figure 4.2. Simulink block diagram of synchronized Lorenz system.

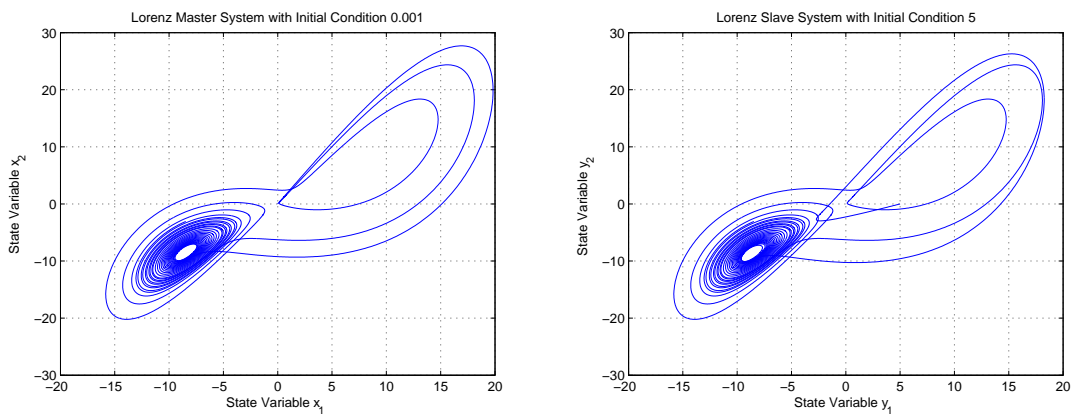


Figure 4.3. Lorenz master and slave system with different initial conditions.

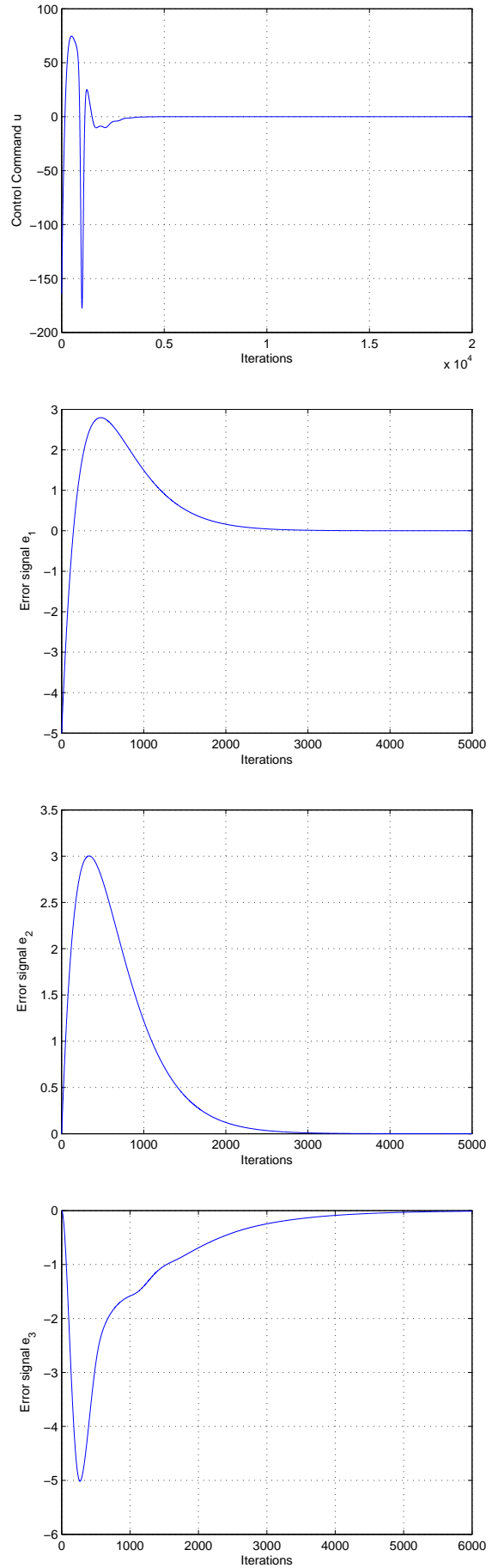


Figure 4.4. The control command u and error signals e_1 , e_2 and e_3 .

exactly in every state. So synchronization of Lorenz master system and slave system can be achieved by using complete synchronization approach. This Simulink design will be used later in our System Generator design.

4.5. Synchronization of Rössler System

To show the complete synchronization valids for different attractors we will apply it to Rössler system, too. For simplicity again we chose master and slave systems have the same parameter values.

$$\dot{x} = \mathbf{F}_M(x)$$

$$\begin{aligned} \dot{x}_1 &= -(x_2 + x_3) \\ \dot{x}_2 &= x_1 + \alpha x_2 \\ \dot{x}_3 &= \beta + x_3(x_1 - \gamma) \end{aligned} \quad (4.41)$$

$$\dot{y} = \mathbf{F}_S(y)$$

$$\begin{aligned} \dot{y}_1 &= -(y_2 + y_3) + g_1(y)u \\ \dot{y}_2 &= y_1 + \alpha y_2 + g_2(y)u \\ \dot{y}_3 &= \beta + y_3(y_1 - \gamma) + g_3(y)u \end{aligned} \quad (4.42)$$

where the parameters $\alpha = 0.2$, $\beta = 0.2$ and $\gamma = 5.7$. $g_1(y), g_2(y), g_3(y)$ are chosen again as a constant for simplicity. The extended synchronization error system can be calculated and written down as follows:

$$\begin{aligned} \dot{x}_1 &= -(x_2 + x_3) \\ \dot{x}_2 &= x_1 + \alpha x_2 \\ \dot{x}_3 &= \beta + x_3(x_1 - \gamma) \\ \dot{e}_1 &= -(e_2 + e_3) - g_1(x - e)u \\ \dot{e}_2 &= e_1 + \alpha e_2 - g_2(x - e)u \\ \dot{e}_3 &= e_3 e_1 + x_3 e_1 + e_3 x_1 - \gamma e_3 - g_3(x - e)u \end{aligned} \quad (4.43)$$

$\mathbf{F} = [\mathbf{F}_M, \mathbf{F}_M - \mathbf{F}_S]^T$, $\mathbf{G} = [\mathbf{0}, -\mathbf{g}(x - e)]^T$ and after $\mathbf{X} = [x_1, x_2, x_3, e_1, e_2, e_3]^T$ is defined then \mathbf{F} and \mathbf{G} vectors can be written in the form such that

$$\mathbf{F}(X) = \begin{bmatrix} -(x_2 + x_3) \\ x_1 + \alpha x_2 \\ \beta + x_3(x_1 - \gamma) \\ -(e_2 + e_3) \\ e_1 + \alpha e_2 \\ e_3 e_1 + x_3 e_1 + e_3 x_1 - \gamma e_3 \end{bmatrix}, \mathbf{G}(X) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}$$

The same procedure will be followed in Rössler system like Lorenz system. Recall that we should first calculate $\mathbf{C}_1(x, e)$. We know that $\mathbf{C}_1(x, e) = \text{span}\{\mathbf{G}\}$ so $\mathbf{C}_1(x, e)$ can be written such that

$$\mathbf{C}_1(x, e) = \text{span} \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix} \right\} \quad (4.44)$$

So as to find $\mathbf{C}_2(x, e)$, $ad_F \mathbf{G}$ should be calculated as follows:

$$ad_F \mathbf{G} = [\mathbf{F}, \mathbf{G}] = \frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G}$$

Because \mathbf{G} vector has only composed of constant elements the term $\frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} = 0$, so $ad_F \mathbf{G}$ can be obtained by finding second term of it. It can be found as follows:

$$-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G} = - \begin{bmatrix} 0 & -1 & -1 & 0 & 0 & 0 \\ 1 & \alpha & 0 & 0 & 0 & 0 \\ x_3 & 0 & x_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 & \alpha & 0 \\ e_3 & 0 & e_1 & (x_3 + e_3) & 0 & (e_1 + x_1 - \gamma) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}$$

$$ad_F \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -(g_1 + g_2) \\ g_1 + \alpha g_2 \\ g_1(x_3 + e_3) + g_3(e_1 + x_1 - \gamma) \end{bmatrix} \quad (4.45)$$

In order to find $\mathbf{C}_3(x, e)$ one step has to be taken which is to calculate $ad_F^2 \mathbf{G}$. $ad_F^2 \mathbf{G}$ is calculated as in Equation 4.22.

$$ad_F^2 \mathbf{G} = [\mathbf{F}, \Psi] = \frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi$$

So let us start with by finding the first term of $ad_F^2 \mathbf{G}$ which is $\frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F}$:

$$\frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ g_3 & 0 & g_1 & g_3 & 0 & g_1 \end{bmatrix} \cdot \begin{bmatrix} -(x_2 + x_3) \\ x_1 + \alpha x_2 \\ B + x_3(x_1 - \gamma) \\ -(e_2 + e_3) \\ e_1 + \alpha e_2 \\ e_3 e_1 + x_3 e_1 + e_3 x_1 - \gamma e_3 \end{bmatrix}$$

$$\frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ g_1[\beta + x_3(x_1 - \gamma) + e_3 e_1 + x_3 e_1 + e_3 x_1 - \gamma e_3] + g_3[-(x_2 + x_3) - (e_2 + e_3)] \end{bmatrix} \quad (4.46)$$

The second part of $ad_F^2 \mathbf{G}$ can be found such that

$$\begin{aligned}
-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi = & \begin{bmatrix} 0 & -1 & -1 & 0 & 0 & 0 \\ 1 & \alpha & 0 & 0 & 0 & 0 \\ x_3 & 0 & x_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 & \alpha & 0 \\ e_3 & 0 & e_1 & (x_3 + e_3) & 0 & (e_1 + x_1 - \gamma) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ -(g_1 + g_2) \\ g_1 + \alpha g_2 \\ g_1(x_3 + e_3) + g_3(e_1 + x_1 - \gamma) \end{bmatrix} \\
-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi = & \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_1 + \alpha g_2 + g_1(x_3 + e_3) + g_3(e_1 x_1 - \gamma) \\ g_1 + g_2 - \alpha(g_1 + \alpha g_2) \\ (x_3 + e_3)(g_1 + g_2) + (\gamma - e_1 - x_1)[g_1(x_3 + e_3) + g_3(e_1 + x_1 - \gamma)] \end{bmatrix} \\
& (4.47)
\end{aligned}$$

By combining Equations 4.46 and Equation 4.47, $ad_F^2 \mathbf{G}$ can be obtained as follows:

$$ad_F^2 \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_1(x_3 + e_3 + 1) + g_2 \alpha + g_3(e_1 + x_1 - \gamma) \\ g_1(1 - \alpha) + g_2(1 - \alpha^2) \\ g_1(\beta + x_3 + e_3) + g_2(x_3 + e_3) - g_3[x_2 + x_3 + e_2 + e_3 + (\gamma - e_1 - x_1)^2] \end{bmatrix}$$

Without loss of generality, \mathbf{G} vector can be considered as $\mathbf{G} = [0 \ 0 \ 0 \ 0 \ -1 \ 0]^T$ by choosing

$g_1 = g_3 = 0$ and $g_2 = 1$ so $\mathbf{C}_3(x, 0) = \text{span}\{\mathbf{G}, \text{ad}_F\mathbf{G}, \text{ad}_F^2\mathbf{G}\}$ is written as follows:

$$C_3(x, 0) = \text{span} \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & \alpha \\ -1 & \alpha & 1 - \alpha^2 \\ 0 & 0 & x_3 \end{bmatrix} \right\} \quad (4.48)$$

In order to achieve complete synchronization, accessibility distribution function has to be constant dimension. If below conditions from Theorem 4.3 hold then this system has constant degree with $d = 3$.

$$\langle dh, \mathbf{G} \rangle(x, 0) = 0 \quad (4.49)$$

$$\langle dh, \text{ad}_F\mathbf{G} \rangle(x, 0) = 0 \quad (4.50)$$

$$\langle dh, \text{ad}_F^2\mathbf{G} \rangle(x, 0) \neq 0 \quad (4.51)$$

In order to satisfy Equation 4.49, the output function $h(x, e)$ can not include the term e_2 . So as to satisfy Equation 4.50, the output function can not include e_1 . There is one alternative left which is $h(x, e) = e_3$. If we chose $h(x, e) = e_3$ then Equation 4.51 is violated since if we chose $h(x, e) = e_3$ then $\langle dh, \text{ad}_F^2\mathbf{G} \rangle(x, 0) = x_3$. However, this condition depends on state x_3 which is chaotic and can not be modified. As a result $\mathbf{C}_3(x, 0)$ has variable dimension. So, it is not possible to satisfy conditions Theorem 4.3. From this result we find that two Rössler system can not generate a tangent space of dimension $d = 3$ with an input of constant elements. With this in mind we try to determine a tangent space of constant dimension $d = 2$. From Theorem 4.3 we can calculate the output function $h(x, e)$ such that

$$-\frac{\partial h}{\partial e_2} = 0, \quad (4.52)$$

$$-\frac{\partial h}{\partial e_1} + \alpha \frac{\partial h}{\partial e_2} \neq 0. \quad (4.53)$$

A possible function which satisfies Equations 4.52 and 4.53 is $h(x, e) = e_1$. We have an output function the relative degree is calculated and for this case is $d = \rho = 2 = \text{Dim}(\mathbf{C}_3(x, 0))$ for all for all $x \in \mathbb{R}^3$. In order to have an invertible transformation with

$\rho = 2$, from Definition 4.4 (i) $L_G h(x, e) = 0$ and (ii) $L_G L_F h(x, e) \neq 0$ then they can be found as follows:

$$L_G h(x, e) = \langle \partial h, \mathbf{G} \rangle (x, w) = 0 \quad (4.54)$$

$$L_F h(x, e) = \langle \partial h, \mathbf{F} \rangle (x, w) = -(e_2 + e_3) \frac{\partial e_1}{\partial e_1} = -(e_2 + e_3) \quad (4.55)$$

$$L_G L_F h(x, e) = L_G[-(e_2 + e_3)] = -(-1 \frac{\partial e_2}{\partial e_2}) = 1 \quad (4.56)$$

$L_G h(x, e) = 0$, $L_G L_F h(x, e) = 1$ after all requirements hold the control command u for $\rho = 2$ can be written as follows:

$$u = \frac{1}{L_G L_F h(x, e)} (-L_F^2 h(x, e) + \mathbf{K}_1(z_1 - z_1^*) + \mathbf{K}_2(z_2 - z_2^*)) \quad (4.57)$$

The term $L_G L_F h(x, e)$ has been found previously in Equation 4.56. $L_F^2 h(x, e)$ needs to be found.

$$L_F^2 h(x, e) = L_F[-(e_2 + e_3)] = -[e_1 + \alpha e_2 + e_3 e_1 + x_3 e_1 + e_3 x_1 - \gamma e_3] \quad (4.58)$$

We know that $z_1 = h(x, e) = e_1$ and $z_2 = L_F h(x, e) = -(e_2 + e_3)$ then we can find the control command by choosing proper control gains $\mathbf{K}_1 = -9$ and $\mathbf{K}_2 = -6$ as follows

$$u = [e_1 + \alpha e_2 + e_3 e_1 + x_3 e_1 + e_3 x_1 - \gamma e_3 - 9e_1 + 6(e_2 + e_3)] \quad (4.59)$$

After obtaining control command u , the Simulink diagram will be constructed. Figure 4.5 represents feedback control signal u and by adding u to Rössler slave system the design can be completed. We know from section 3.3.2 that Rössler master system *Integrator1* block has initial condition equals to 3.0474. On the other hand, we set *Integrator1* block of Rössler system to -5 so that the difference between master and slave system can be created with respect to their initial conditions. The synchronized Rössler system can be constructed as in Figure 4.6.

As can be seen below Figure 4.7 trajectory of slave system follows trajectory of master system. We also examine Figure 4.8 which represents the behaviour of control command u and error signals e_1 , e_2 and e_3 respectively. We can understand from Figure

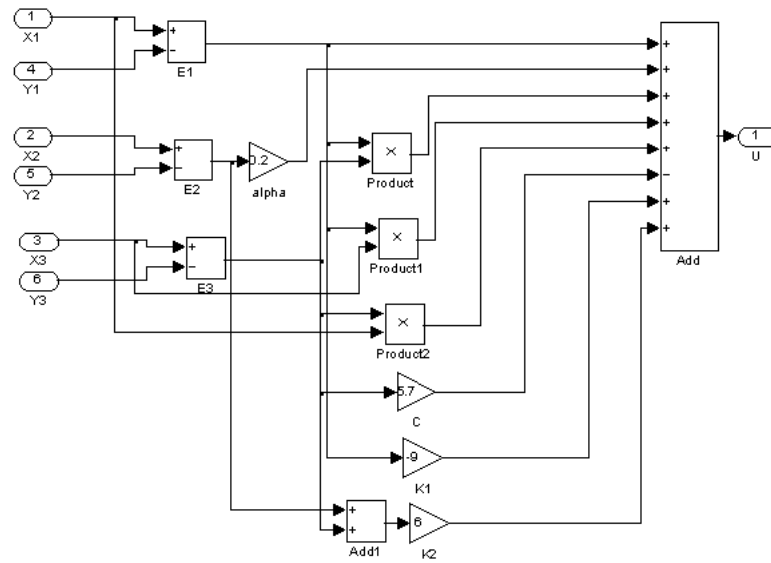


Figure 4.5. Simulink block diagram of the control command u for Rössler system.

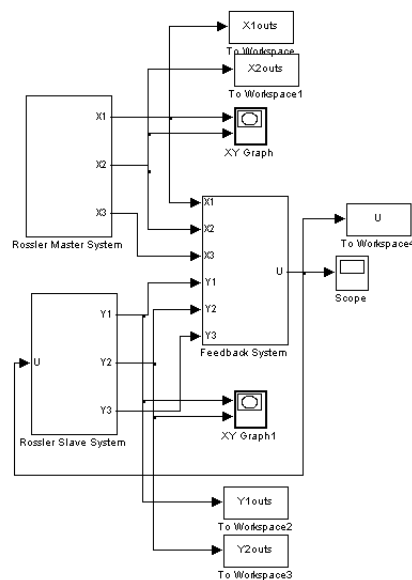


Figure 4.6. Simulink block diagram of the synchronized Rössler system.

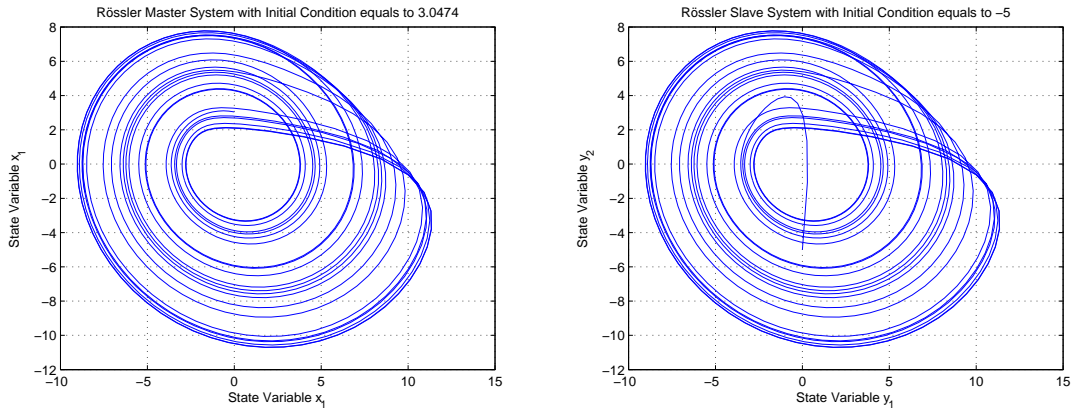


Figure 4.7. Rössler master and slave system with different initial conditions.

4.8 that when error signals stabilize at zero in amplitude then the control command u reaches to zero since it depends on error signals which means that the synchronization of Rössler system is completed successfully.

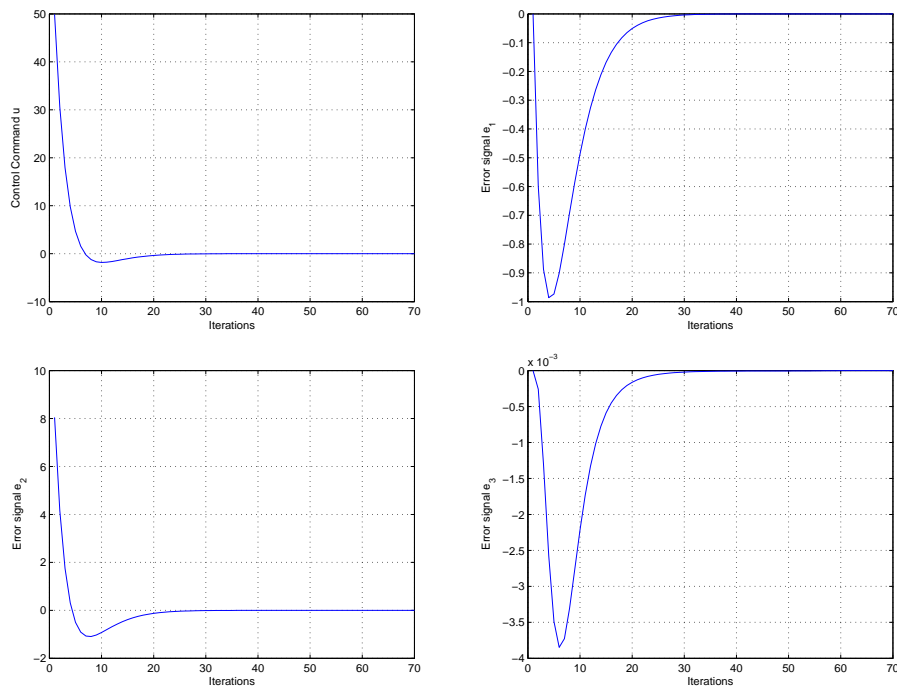


Figure 4.8. The control command u and error signals e_1 , e_2 and e_3 .

4.6. Synchronization of Linz and Sprott System

Linz and Sprott system is the third chaotic system which we will study on. Linz Sprott master and slave systems are given respectively:

$$\dot{x} = \mathbf{F}_M(x)$$

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -\alpha x_3 - \beta x_2 - (|x_1| - 1) \end{aligned} \quad (4.60)$$

and $\dot{y} = \mathbf{F}_S(y) + \mathbf{g}(y)u$

$$\begin{aligned} \dot{y}_1 &= y_2 + g_1(y)u \\ \dot{y}_2 &= y_3 + g_2(y)u \\ \dot{y}_3 &= -\alpha y_3 - \beta y_2 - (|y_1| - 1) + g_3(y)u \end{aligned} \quad (4.61)$$

where the parameters are $\alpha = 0.6$ and $\beta = 1$. g_i with $i = 1, 2, 3$ functions are chosen as constant for simplicity. The extended synchronization error system can again be constructed by using 4.1 as follows:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ \dot{x}_3 &= -\alpha x_3 - \beta x_2 - (|x_1| - 1) \\ \dot{e}_1 &= e_2 - g_1(x - e)u \\ \dot{e}_2 &= e_3 - g_2(x - e)u \\ \dot{e}_3 &= -\alpha e_3 - \beta e_2 - (|e_1| - 1) - g_3(x - e)u \end{aligned} \quad (4.62)$$

Then we can express above system such that $\mathbf{F} = [\mathbf{F}_M, \mathbf{F}_M - \mathbf{F}_S]^T$, $\mathbf{G} = [\mathbf{0}, -\mathbf{g}(x - e)]^T$

and $\mathbf{X} = [x_1, x_2, x_3, e_1, e_2, e_3]^T$ write them down as follows

$$\mathbf{F} = \begin{bmatrix} x_2 \\ x_3 \\ -\alpha x_3 - \beta x_2 - (|x_1| - 1) \\ e_2 \\ e_3 \\ -\alpha e_3 - \beta e_2 - (|e_1| - 1) \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}$$

The same procedure will be applied to Linz and Sprott system. At first, there is whether control command u which make two chaotic systems are sychronized. So let us start with writing \mathbf{C}_1 and then calculating \mathbf{C}_2 and \mathbf{C}_3 in order to see accessibility function has constant constant dimension $d = 3$. $\mathbf{C}_1(x, e)$ can be written as follows:

$$\mathbf{C}_1(x, e) = \text{span} \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix} \right\} \quad (4.63)$$

In order to find $\mathbf{C}_2(x, e)$ we should first calculate $ad_F \mathbf{G}$

$$ad_F \mathbf{G} = [\mathbf{F}, \mathbf{G}] = \frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G}$$

$\frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} = 0$ as \mathbf{G} vector only contains constant terms. We can find $ad_F \mathbf{G}$ by calculating the second term of it such that

$$-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G} = - \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -\text{sgn}(x_1) & -\beta & -\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\text{sgn}(e_1) & -\beta & -\alpha \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}$$

$$ad_F \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_2 \\ g_3 \\ -g_1 \operatorname{sgn}(e_1) - \beta g_2 - \alpha g_3 \end{bmatrix} \quad (4.64)$$

There is one calculation has to be done to obtain $\mathbf{C}_3(x, 0)$. By calculating ad_F^2 , the accessibility distribution of Linz and Sprott system will be obtained. ad_F^2 is wirtten as follows

$$ad_F^2 \mathbf{G} = [\mathbf{F}, \Psi] = \frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi \quad (4.65)$$

$\frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} = 0$ since Ψ does not contain any state variables. So in order to find ad_F^2 we should calculate the second part of Equation 4.65 as follows:

$$-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi = - \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -\operatorname{sgn}(x_1) & -\beta & -\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\operatorname{sgn}(e_1) & -\beta & -\alpha \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_2 \\ g_3 \\ -g_1 \operatorname{sgn}(e_1) - \beta g_2 - \alpha g_3 \end{bmatrix}$$

$$ad_F^2 \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_3 \\ g_1 \operatorname{sgn}(e_1) + \beta g_2 + \alpha g_3 \\ -g_1 \alpha \operatorname{sgn}(e_1) + g_2 [\operatorname{sgn}(e_1) - \alpha \beta] + g_3 (\beta - \alpha^2) \end{bmatrix} \quad (4.66)$$

$\mathbf{C}_3(x, 0)$ can be constructed by setting $g_1 = g_2 = 0$ and $g_3 = 1$ for simplicity then $\mathbf{C}_3(x, 0)$ is written as follows

$$C_3(x, 0) = \text{span} \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & \alpha \\ -1 & -\alpha & \beta - \alpha^2 \end{bmatrix} \right\} \quad (4.67)$$

If following Equations 4.68, 4.69 and 4.70 hold then Theorem 4.3 conditions are satisfied so we can chose proper output function then calculate proper control command u such that two Linz and Sprott systems can be synchronized each other.

$$-\frac{\partial h}{\partial e_3} = 0, \quad (4.68)$$

$$\frac{\partial h}{\partial e_2} - \alpha \frac{\partial h}{\partial e_3} = 0, \quad (4.69)$$

$$-\frac{\partial h}{\partial e_1} + \alpha \frac{\partial h}{\partial e_2} + (\beta - \alpha^2) \frac{\partial h}{\partial e_3} \neq 0. \quad (4.70)$$

A possible function which satisfies above Equations is $h(x, e) = e_1$. So we have an output function which satisfies Equation 4.68, 4.69 and 4.70 then the constant dimension is calculated and for this case is $d = 3 = \rho = \text{Dim}(\mathbf{C}_3(x, 0))$ for all $x \in \mathbb{R}^3$. In order to have relative degree $\rho = 3$, there three conditions which must be satisfied. These are (i) $L_G h(x, e) = 0$ (ii) $L_G L_F h(x, e) = 0$ and (iii) $L_G L_F^2 \neq 0$ from 4.4 then let us find them as follows:

$$L_G h(x, e) = \langle \partial h, \mathbf{G} \rangle = 0,$$

$$L_F h(x, e) = \langle \partial h, \mathbf{F} \rangle = e_2,$$

$$L_G L_F h(x, e) = L_G e_2 = 0, \quad (4.71)$$

$$L_F^2 h(x, e) = L_F L_F h(x, w) = L_F e_2 = e_3,$$

$$L_G L_F^2 = L_G e_3 = -1 \neq 0 \quad (4.72)$$

$L_G h(x, e) = 0$, $L_G L_F = 0$ and $L_G L_F^2 = -1$ have been calculated so the control command u can be written as follows:

$$u = \frac{1}{L_G L_F^2 h(x, e)} [-L_F^3 h(x, e) + K_1(z_1 - z_1^*) + K_2(z_2 - z_2^*) + K_3(z_3 - z_3^*)] \quad (4.73)$$

$L_G L_F^2 h(x, e) = -1$ is known so we need to find the term $L_F^3 h(x, e)$ such that

$$L_F^3 h(x, e) = L_F[L_F^2 h(x, e)] = L_F(e_3) = -\alpha e_3 - \beta e_2 - (|e_1| - 1) \quad (4.74)$$

Then we can write the control command u by choosing $K_1 = 50$, $K_2 = 50$ and $K_3 = 50$ properly so that synchronization error system reach stabilization point which we choose for Linz and Sprott system as zero.

$$u = -\alpha e_3 - \beta e_2 - (|e_1| - 1) - 50e_1 - 50e_2 - 50e_3 \quad (4.75)$$

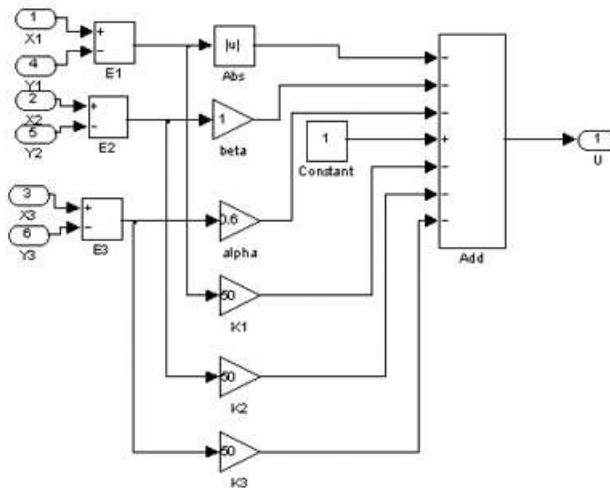


Figure 4.9. Simulink block diagram of the control command u for Linz and Sprott system

We completed our design by finding control command u . It is time to see the results in Simulink. Using Simulink blocks we construct a control command u as in Figure 4.9 so that two Linz and Sprott systems are synchronized. In order to see this we chose different initial conditions for master and slave system. Master system *Integrator2* block has initial condition equals to 0.000001 and slave system *Integrator2* block has initial condition equals to 1 in Figure 4.10.

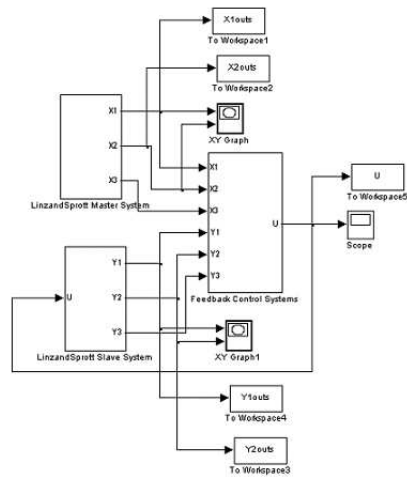


Figure 4.10. Simulink block diagram of the synchronized Linz and Sprott system.

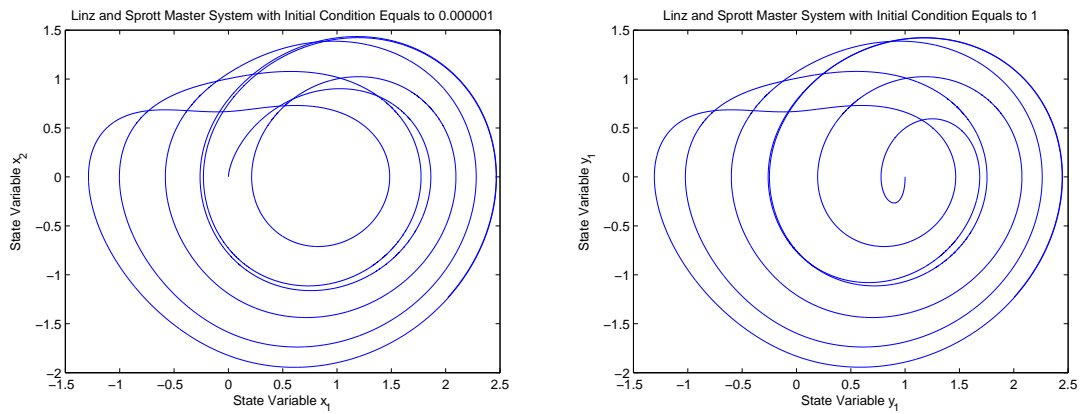


Figure 4.11. Linz and Sprott master and slave system with different initial conditions.

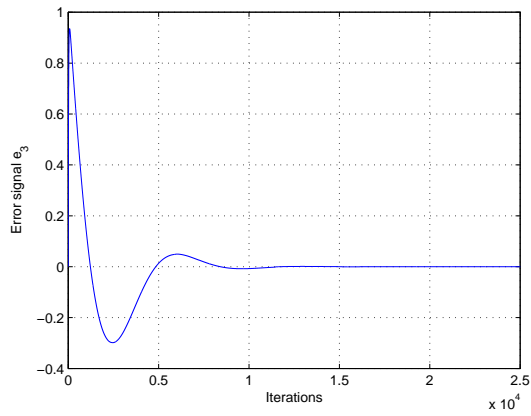
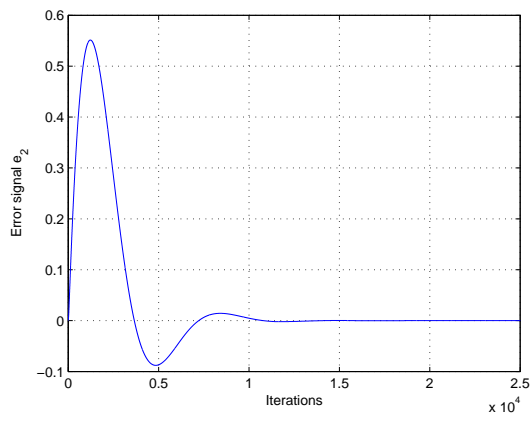
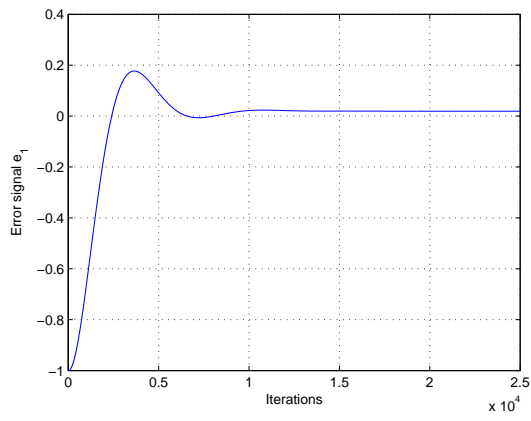
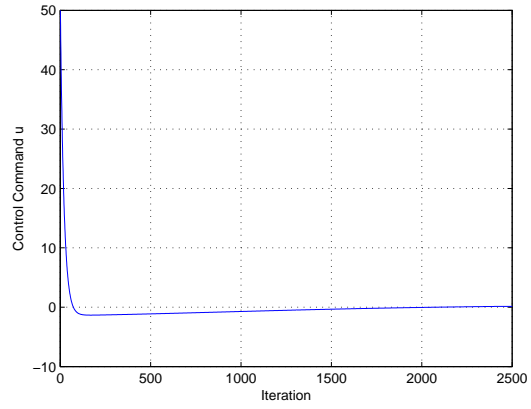


Figure 4.12. The control command u and error signals e_1 , e_2 and e_3 .

As can be seen in Figures 4.11 Linz and Sprott slave system exactly tracks Linz and Sprott master system. We can also see the synchronization of Linz and Sprott system by examining Figure 4.12. As we can see in the Figure 4.12 as the time passes error signals reaches to zero and remain on it. Because the control command u composed of error signals, it also reaches to zero which means there is no input vector for Linz and Sprott slave system which means that synchronization is completed successfully.

4.7. Synchronization of Chua System

In this section synchronization of two identical Chua's circuits will be examined. Master and slave system parameter values are chosen equal to each other for simplicity again. $g(x_1)$ function is replaced by $f(x_1)$ in order not to confused with the input vector of the slave system. Chua master and slave systems can be represented as follows:

$$\dot{x} = \mathbf{F}_M$$

$$\begin{aligned} \dot{x}_1 &= k\alpha[x_2 - x_1 - f(x_1)] \\ \dot{x}_2 &= k(x_1 - x_2 + x_3) \\ \dot{x}_3 &= k(-\beta x_2 - \gamma x_3) \end{aligned} \quad (4.76)$$

$$\dot{y} = \mathbf{F}_S + \mathbf{g}(y)u$$

$$\begin{aligned} \dot{y}_1 &= k\alpha[y_2 - y_1 - f(y_1)] + g_1(y)u \\ \dot{y}_2 &= k(y_1 - y_2 + y_3) + g_2(y)u \\ \dot{y}_3 &= k(-\beta y_2 - \gamma y_3) + g_3(y)u \end{aligned} \quad (4.77)$$

Functions $g_1(y)$, $g_2(y)$ and $g_3(y)$ are the corresponding elements in the input vector of the slave system again. $g_1(y)$, $g_2(y)$ and $g_3(y)$ are chosen as constant functions so that our design process will be easy to calculate. The extended synchronization error

system is written as follows:

$$\begin{aligned}
\dot{x}_1 &= k\alpha[x_2 - x_1 - f(x_1)] \\
\dot{x}_2 &= k(x_1 - x_2 + x_3) \\
\dot{x}_3 &= k(-\beta x_2 - \gamma x_3) \\
\dot{e}_1 &= k\alpha[e_2 - e_1 - f(e_1)] - g_1(x - e)u \\
\dot{e}_2 &= k(e_1 - e_2 + e_3) - g_2(x - e)u \\
\dot{e}_3 &= k(-\beta e_2 - \gamma e_3) - g_3(x - e)u
\end{aligned} \tag{4.78}$$

Equation 4.78 can be written in affine form as $\dot{\mathbf{X}} = \mathbf{F}(X) + \mathbf{G}(X)u$ where $\mathbf{F}(X) = [\mathbf{F}_M, \mathbf{F}_M - \mathbf{F}_S]^T$ and $\mathbf{G}(X) = [\mathbf{0}, -\mathbf{g}(x, e)]^T$ and $\mathbf{X} = [x_1, x_2, x_3, e_1, e_2, e_3]^T$. Then \mathbf{F} and \mathbf{G} vectors can be written such that

$$\mathbf{F} = \begin{bmatrix} k\alpha[x_2 - x_1 - f(x_1)] \\ k(x_1 - x_2 + x_3) \\ k(-\beta x_2 - \gamma x_3) \\ k\alpha[e_2 - e_1 - f(e_1)] \\ k(e_1 - e_2 + e_3) \\ k(-\beta e_2 - \gamma e_3) \end{bmatrix}, \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}$$

After defining \mathbf{F} and \mathbf{G} vectors. In order to obtain $\mathbf{C}_3(x, 0)$, $\mathbf{C}_1(x, e)$ and $\mathbf{C}_2(x, e)$ have to be calculated. $\mathbf{C}_1(x, e) = \text{span}\{\mathbf{G}\}$ is known. In order to obtain $\mathbf{C}_2(x, e)$, $ad_F \mathbf{G}$ has to be calculated such that

$$ad_F \mathbf{G} = [\mathbf{F}, \mathbf{G}] = \frac{\partial \mathbf{G}}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G}$$

Because the vector \mathbf{G} is only composed of constant elements then derivative of \mathbf{G} with respect to \mathbf{X} equals to zero then we only need to calculate $-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G}$ to find $ad_F \mathbf{G}$.

$$-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \mathbf{G} = - \begin{bmatrix} a_{11} & k\alpha & 0 & 0 & 0 & 0 \\ k & -k & k & 0 & 0 & 0 \\ 0 & -k\beta & -k\gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{44} & k\alpha & 0 \\ 0 & 0 & 0 & k & -k & k \\ 0 & 0 & 0 & 0 & -k\beta & -k\gamma \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g_1 \\ -g_2 \\ -g_3 \end{bmatrix}$$

$$a_{11} = -k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(x_1 + 1) - \text{sgn}(x_1 - 1)]\}$$

$$a_{44} = -k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\}$$

$$ad_F \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ g_1[-k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\}] + g_2k\alpha \\ g_1k - g_2k + g_3k \\ -g_2k\beta - g_3k\gamma \end{bmatrix} \quad (4.79)$$

After finding $ad_F G$, $\mathbf{C}_2(x, e)$ can be written such that

$$\mathbf{C}_2(x, e) = \text{span} \left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -g_1 & b_{42} \\ -g_2 & g_1k - g_2k + g_3k \\ -g_3 & -g_2k\beta - g_3k\gamma \end{bmatrix} \right\} \quad (4.80)$$

$$b_{42} = g_1[-k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\}] + g_2k\alpha.$$

In order to obtain $\mathbf{C}_3(x, 0)$, $ad_F^2 \mathbf{G}$ needs to be found. $ad_F^2 \mathbf{G}$ can be calculated as follows

$$ad_F^2 \mathbf{G} = [\mathbf{F}, ad_F \mathbf{G}] = [\mathbf{F}, \Psi] = \frac{\partial \Psi}{\partial \mathbf{X}} \cdot \mathbf{F} - \frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi$$

$\frac{\partial \Psi}{\partial \mathbf{X}} = 0$ because derivative of the Ψ with respect to \mathbf{X} equals to zero. So $ad_F^2 \mathbf{G} = -\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi$

$$-\frac{\partial \mathbf{F}}{\partial \mathbf{X}} \cdot \Psi = - \begin{bmatrix} a_{11} & k\alpha & 0 & 0 & 0 & 0 \\ k & -k & k & 0 & 0 & 0 \\ 0 & -k\beta & -k\gamma & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{44} & k\alpha & 0 \\ 0 & 0 & 0 & k & -k & k \\ 0 & 0 & 0 & 0 & -k\beta & -k\gamma \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ c_4 \\ g_1k - g_2k - g_3k \\ -g_2k\beta + g_3k\gamma \end{bmatrix}$$

$$\begin{aligned}
a_{11} &= -k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(x_1 + 1) - \text{sgn}(x_1 - 1)]\} \\
a_{44} &= -k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\} \\
c_4 &= g_1[-k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\}] + g_2k\alpha
\end{aligned}$$

$$ad_F^2 \mathbf{G} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ d_4 \\ d_5 \\ k\beta(g_1k - g_2k - g_3k) + k\gamma(-g_2k\beta - g_3k\gamma) \end{bmatrix}$$

$$d_4 = \{k\alpha + k\alpha\{m_0 + 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\}\}\{g_1[-k\alpha - k\alpha\{m_0 - 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\}] + g_2k\alpha\} + k\alpha(g_2k - g_3k - g_1k)$$

$$d_5 = k[g_1\{k\alpha + k\alpha\{m_0 + 1/2(m_1 - m_0)[\text{sgn}(e_1 + 1) - \text{sgn}(e_1 - 1)]\}\} + g_2k\alpha] + k(g_1k - g_2k + g_3k) + k(g_2k\beta + g_3k\gamma)$$

After setting $g_1 = g_2 = 0$ and $g_3 = 1$ for simplicity, $C_3(x, 0)$ can be written as follows:

$$C_3(x, 0) = \text{span} \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -k^2\alpha \\ 0 & k & k^2(1 + \gamma) \\ -1 & -k\gamma & -k^2(\beta + \gamma^2) \end{bmatrix} \right\} \quad (4.81)$$

First Theorem 4.3 conditions are stated as below in order to test whether Chua system accessibility distribution function has constant dimension for $\rho = 3$.

$$-\frac{\partial h}{\partial e_3} = 0 \quad (4.82)$$

$$k\frac{\partial h}{\partial e_2} - k\gamma\frac{\partial h}{\partial e_3} = 0 \quad (4.83)$$

$$-k^2\alpha\frac{\partial h}{\partial e_1} + k^2(1 + \gamma)\frac{\partial h}{\partial e_2} - k^2(\beta + \gamma^2)\frac{\partial h}{\partial e_3} \neq 0 \quad (4.84)$$

In order to satisfy Equations 4.82, 4.83 and 4.84, the output function can be chosen as $h(x, e) = e_1$ then conditions of Theorem 4.3 are satisfied. There is one more thing which has to be considered is that in order to have relative degree $\rho = 3$, there are three conditions below from Definition 4.4 as follows:

$$L_G h(x, e) = 0 \quad (4.85)$$

$$L_G L_F h(x, e) = 0 \quad (4.86)$$

$$L_G L_F^2 h(x, e) \neq 0 \quad (4.87)$$

So let us examine that above conditions hold for Chua system.

$$L_G h(x, e) = \langle \partial h, \mathbf{G} \rangle = 0 \quad (4.88)$$

$$L_F h(x, e) = \langle \partial h, \mathbf{F} \rangle = k\alpha[e_2 - e_1 - f(e_1)] \quad (4.89)$$

$$L_G L_F h(x, e) = 0 \quad (4.90)$$

$$L_F^2 h(x, e) = k\alpha\{k[e_1 - e_2 + e_3] - k\alpha[e_2 - e_1 - f(e_1)] - f(\dot{e}_1)\} \quad (4.91)$$

$$L_G L_F^2 h(x, e) = -k^2\alpha \quad (4.92)$$

$L_G h(x, e) = 0$, $L_G L_F h(x, e) = 0$ and $L_G L_F^2 h(x, e) = -k^2\alpha$. As it can be seen, all requirements hold then the control command can be written in the form for $\rho = 3$ as follows:

$$u = \frac{1}{L_G L_F^2 h(x, e)} [-L_F^3 h(x, e) + \mathbf{K}_1(z_1 - z_1^*) + \mathbf{K}_2(z_2 - z_2^*) + \mathbf{K}_3(z_3 - z_3^*)] \quad (4.93)$$

In order to calculate the control command u , $L_F^3 h(x, e)$ needs to be found, because other functions have been calculated.

$$\begin{aligned} L_F^3 h(x, e) = & k\alpha\{k[k\alpha(e_2 - e_1 - f(e_1)) - k(e_1 - e_2 + e_3) + k(-\beta e_2 - \gamma e_3)] \\ & - k\alpha[k(e_1 - e_2 + e_3) - k\alpha(e_2 - e_1 - f(e_1)) - f(\dot{e}_1)] - f(\ddot{e}_1)\} \end{aligned}$$

In order to synchronize master and slave system proper control gains have to be chosen such that $\mathbf{K}_1 = 9$, $\mathbf{K}_2 = -9$ and $\mathbf{K}_3 = -9$. $z_1 = h(x, e) = e_1$, $z_2 = L_F h(x, e) = k\alpha(e_2 - e_1 - f(e_1))$ and $z_3 = L_F^2 h(x, e) = k\alpha[k(e_1 - e_2 + e_3) - k\alpha(e_2 - e_1 - f(e_1)) - f(\dot{e}_1)]$.

And the stabilization points of the system coordinates are $z_1^* = z_2^* = z_3^* = 0$. Then the control command can be written as follows:

$$\begin{aligned}
 u = \frac{1}{k} \{ & k[k\alpha(e_2 - e_1 - f(e_1)) - k(e_1 - e_2 + e_3) + k(-\beta e_2 - \gamma e_3)] \\
 & - k\alpha[k(e_1 - e_2 + e_3) - k\alpha(e_2 - e_1 - f(e_1)) - f(\dot{e}_1)] - f(\ddot{e}_1) \} \\
 & - \frac{1}{k^2\alpha} \{ K_1 e_1 + K_2 k\alpha(e_2 - e_1 - f(e_1)) \\
 & + K_3 k\alpha[k(e_1 - e_2 + e_3) - k\alpha(e_2 - e_1 - f(e_1)) - f(\dot{e}_1)] \} \quad (4.94)
 \end{aligned}$$

then we can construct control command u by using Simulink block as in Figure 4.13.

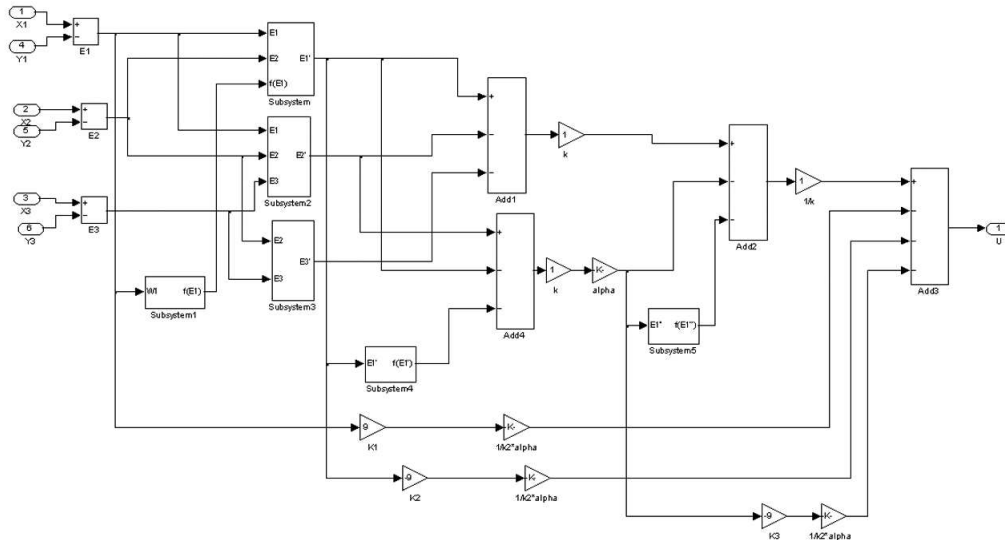


Figure 4.13. Simulink block diagram of the control command u for Chua systems.

Chua master and slave system trajectories draw the same pattern in Figures 4.15. In Figures 4.16, error signals remain at zero after synchronization is achieved then the control command also remains at zero since it depends on error signals. It means that there is no difference between master and slave systems after control command remains at zero which means that complete synchronization is achieved.

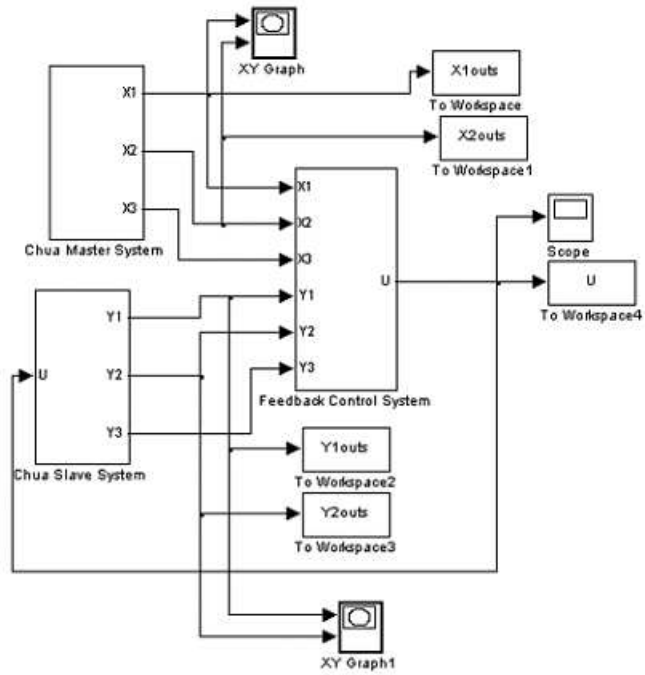


Figure 4.14. Simulink block diagram of the synchronized Chua systems.

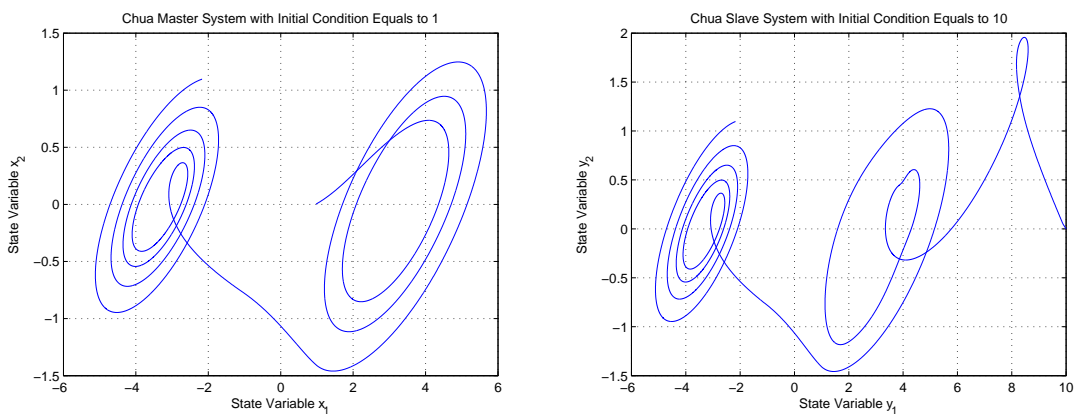


Figure 4.15. Chua master and slave system with different initial conditions.

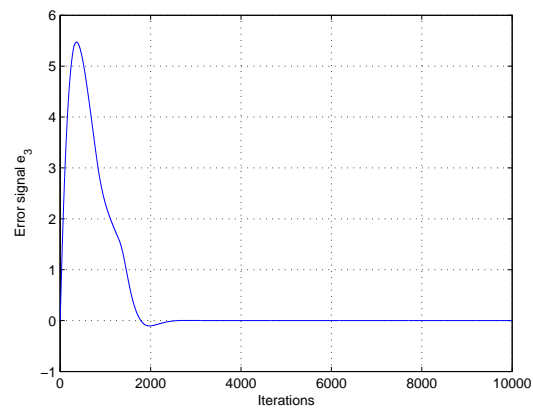
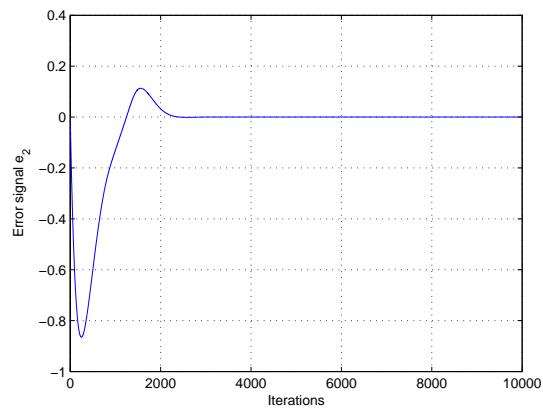
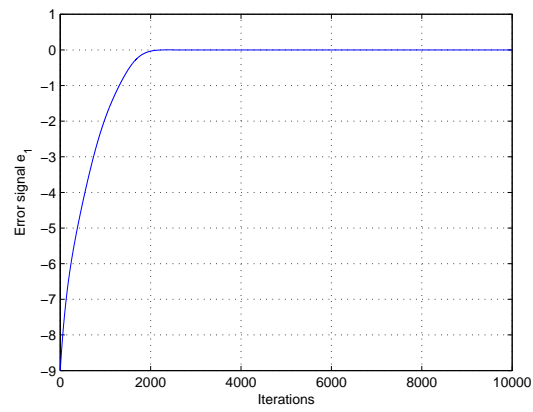
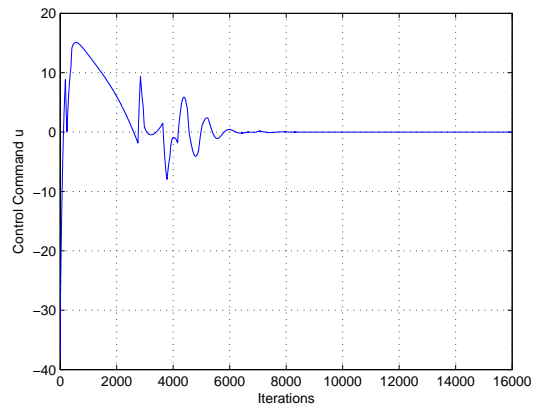


Figure 4.16. The control command u and error signals e_1 , e_2 and e_3 .

CHAPTER 5

FPGA IMPLEMENTATION

In this chapter, System Generator implementation is described. Lorenz system, Rössler system, Linz and Sprott system and Chua system (Leonardo, et al. 2005) are simulated by Matlab. Then, Integrated Software Environment (ISE) program is introduced. Next, Digital-to-Analog (DA) converter is explained. The last part of this chapter composed of synchronized chaotic generators which are implemented to FPGA and the results are presented.

5.1. Implementation Process

Since System Generator (Xilinx 2007a) uses the same Simulink program interface, basically it will be necessary only to translate Simulink structure presented in chapter 3 and chapter 4 and make some adjustments. The goal at this stage is to obtain the same results by using System Generator software. The first step in order to construct synchronized chaotic systems is to translate four Simulink structures designed in chapters 3 and 4. Simulink main library's block elements which were used previously are: *Gain*, *Sum*, *Constant*, *Products*, *Abs*, *Integrator*. So as to make the translation, we need to replace the Simulink main library's blocks by the corresponding Xilinx Blockset library's elements and correctly adjust their parameters.

The only two exceptions are Simulink main library's blocks *Abs* and *Integrator*, which are not available on Xilinx blockset library. Therefore they will be implemented using other blocks elements as will be seen below.

The implementation of Simulink main library's block *Abs* is very simple. By using the Xilinx block *MCode* block we can obtain proper block which acts as *Abs* in Simulink main library. This block provides a convenient and flexible way to calculate arithmetic functions.

The code used to implement the *Abs* function is presented on Table 5.1 below. As can be seen, the function *module* just compares the input u to zero and then multiplies it by -1 when the u is negative. There is only one point to take care of this MATLAB



Figure 5.1. *MCode* Xilinx Block.

Table 5.1. MATLAB function "module.m"

```

Function [mu] = module(u)
if u >= fix({xlSigned, 32, 16, xlTruncate, xlWrap}, 0)
mu = u;
else
mu = -u;
end

```

function: how to convert the double size number zero to a fixed number. We need to specify correctly how the number zero will be represented as can be seen on Table 5.1. In this example, the number zero is converted to a 32 bits long signed number, which has the binary point placed at position 16, the quantization method is *Truncate* and the overflow procedure is *Wrap*.

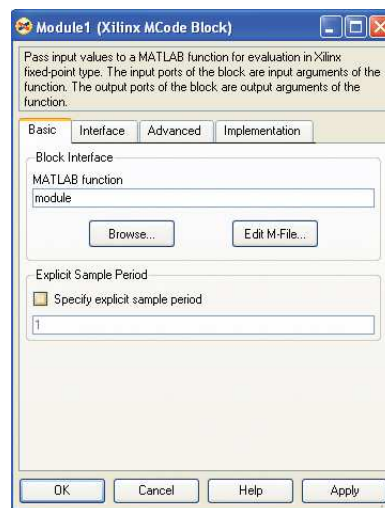


Figure 5.2. *MCode* Xilinx block parameters.

Figure 5.1 and 5.2 represent the *MCode* block and its configuration parameter window, respectively. As can be seen, the block has only one input *u* and one output *mu* that is the module on the value presented on the input.

The second block which Xilinx block library's does not have is *Integrator* block in Simulink library. The integrator model is translated using Xilinx blocks as can be seen in Figure 5.3. The mathematical expression of it can be written: $X'(t - dt) = \frac{X(t) - X(t - dt)}{dt}$. The parameter dt presented in the block dt represents the integration step. The initial state of the integrator is stored inside the block *Register*, in the field *Initial Value*.

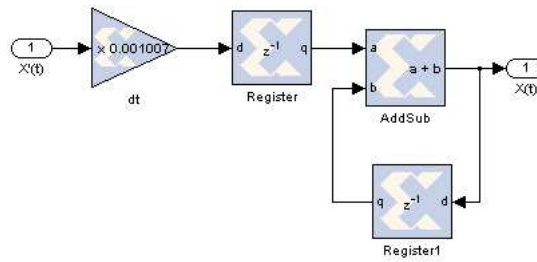


Figure 5.3. Integrator model using Xilinx blockset library.

After missing blocks in Xilinx blockset library's are constructed, the design can be completed by arranging System Generator block in Xilinx blockset library as in Figure 5.4. There are three options in *Synthesis tool section*: Synplify, Synplify Pro and XST (Xilinx Synthesis Tool). XST will be used to synthesize synchronized chaotic systems which was designed in the previous chapter. There are two options in *Hardware description language section*: Verilog and VHDL (Very-high-speed integrated circuits Hardware Description Language). VHDL will be used to program FPGA. FPGA clock period can be arranged so that the desired outputs will be obtained. 100 ns is used as a clock period in all designs. The last thing which has to be arranged is *Simulink system period section*. It can be set as the same in Simulink fundamental sample time. After all requirements are done by pressing *Generate* button in System Generator block, ISE Project file will be obtained.

ISE is Xilinx design software suite (Xilinx 2007b). As can be seen in Figure 5.5, there are three steps in order to obtain bitstream file. These steps are: *Synthesize*, *Implement Design* and *Generate Programming File*. By clicking *Configure Device (IMPACT)* in *Generate Programming File*, *bitstream* file can be obtained. By loading *bitstream* file to FPGA, programming process of FPGA can be completed.

The model of FPGA is Xilinx ML 402 (Xilinx 2006) which does not support analog output. In order to have analog outputs, DA converters are used. In this study

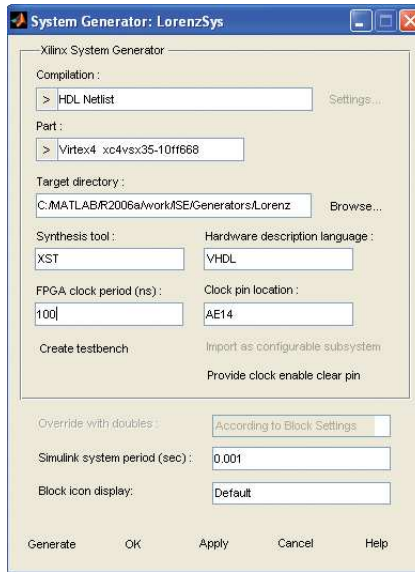


Figure 5.4. System Generator block in Xilinx blokset library.

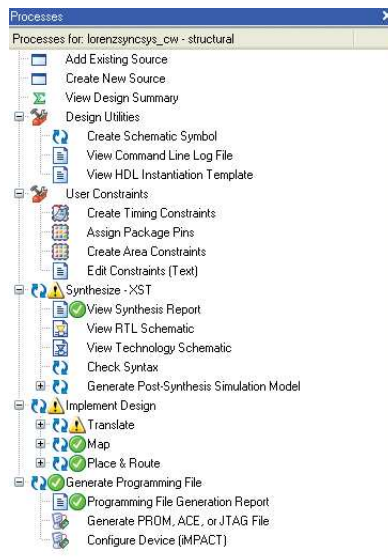


Figure 5.5. Source window of ISE

AD7846 (Analog Devices 2007) ICs from Analog Devices are used to convert digital signal to analog signal. AD7846 provides 16-bit resolution with parallel input. Synchronized chaotic generator systems are designed with 32-bit resolution so the size of output functions have to be converted as 16 bits so that AD7846 can work. At the same time AD7846 does not recognize signed bit in data set come from FPGA. In order to solve this problem the most significant bit of the data set can be inverted by inverter. These two operations can be done by using System Generator blocks as in Figure 5.6.

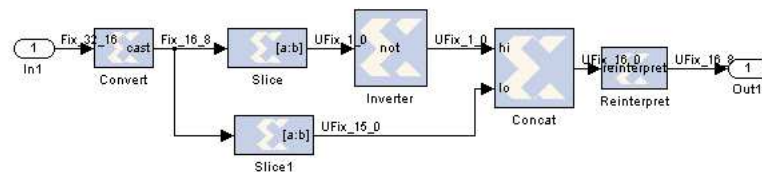


Figure 5.6. Reduction of resolution and inverting of most significant bit.

5.1.1. Implementation of Synchronized Lorenz System

The Lorenz system presented previously on Equation 3.1 is now implemented using Xilinx blockset library's elements as can be seen in Figure 5.7 below. Three integrators blocks that can be seen on this structure are the same as already presented in Figure 5.3. The *Integrator F1* block has initial condition equal to 0.001. The other two integrators have initial condition equal to zero. The integration step dt used in this design is equal to 0.001. The block *AddSub* is composed of one adder and one subtractor, and it performs the operation $a - (b + c) = a - b - c$. This circuit uses 32-bits words with the binary point position after the bit number 16.

The chaotic behaviour of this circuit can be seen on its attractor in Figure 5.8 (compare it with the Lorenz attractor presented in Figure 3.3).

In order to obtain synchronized Lorenz system, the control command u has to be constructed by using Xilinx blockset library's elements. As we know the expression of u from Equation 4.40 then the control command u can be constructed by Xilinx blockset elements as in Figure 5.9 below.

After constructing the control command u then we apply it to Lorenz slave system so that synchronized Lorenz system can be completed successfully. In Figure 5.10,

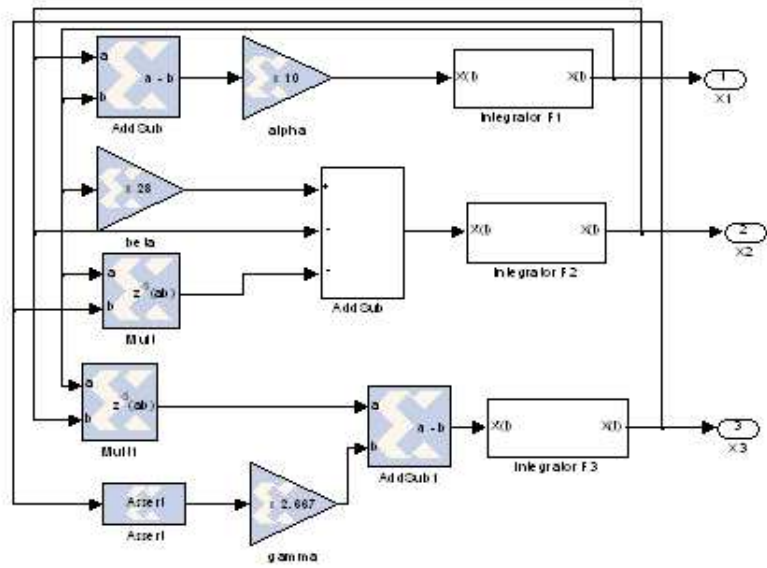


Figure 5.7. Lorenz system by System Generator.

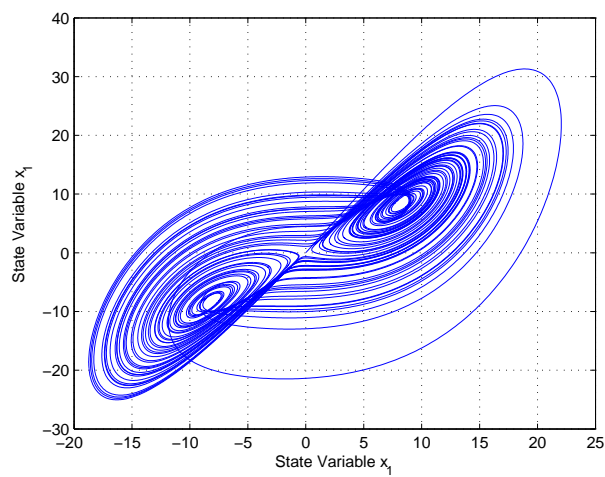


Figure 5.8. Lorenz system attractor.

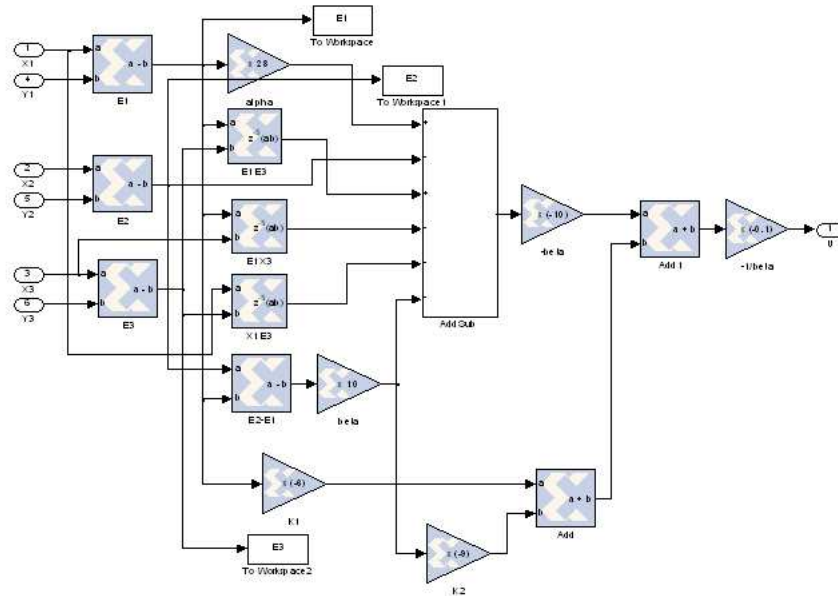


Figure 5.9. The control command u for Lorenz system by System Generator.

synchronized Lorenz system can be seen.

Lorenz master system's *Integrator F1* block initial condition is set to 0.001 and Lorenz slave system's *Integrator F1* is set to 5. By setting master and slave Lorenz system with different initial conditions, as in chapter 4 section 4, synchronization process can be observed. Figure 5.11 represents Lorenz master system attractors and Lorenz Slave system attractors, respectively. As can be seen in Figure 5.11, Although they have different starting points because of initial conditions, they draw the same pattern after synchronization is completed.

Figure 5.12 represents the behaviour of control command u and difference between Lorenz master system state variables and Lorenz slave system state variables, respectively. As chapter 4, It means that synchronization of Lorenz system can be achieved successfully. If Figure 5.12 and Figure 4.4 in chapter 4 draw the same pattern then this design can be implemented to FPGA by using ISE program.

After implementing synchronized Lorenz system to FPGA , by using ISE program, how much source is consumed by synchronized Lorenz system can be seen in Table 5.2.

Figure 5.13 represents Lorenz master system state variables x_1 , x_2 and x_3 and Lorenz slave system state variables y_1 , y_2 and y_3 , respectively. This Figure was captured

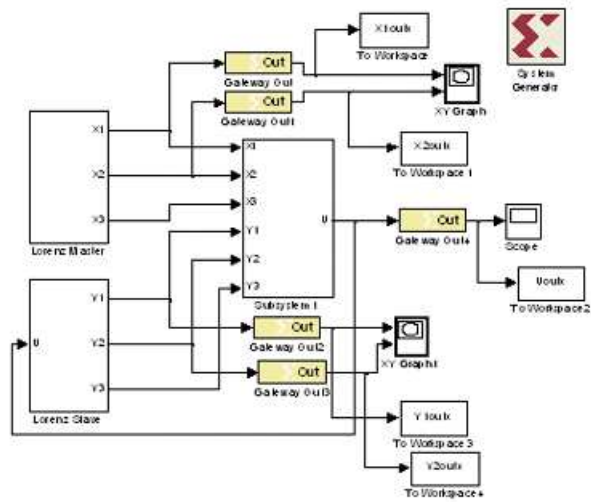


Figure 5.10. Synchronized Lorenz system by System Generator.

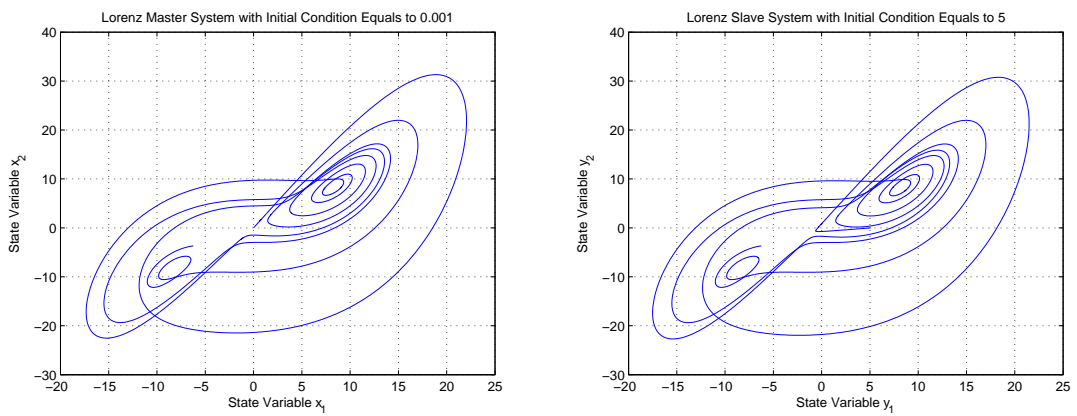


Figure 5.11. Synchronized Lorenz system attractors.

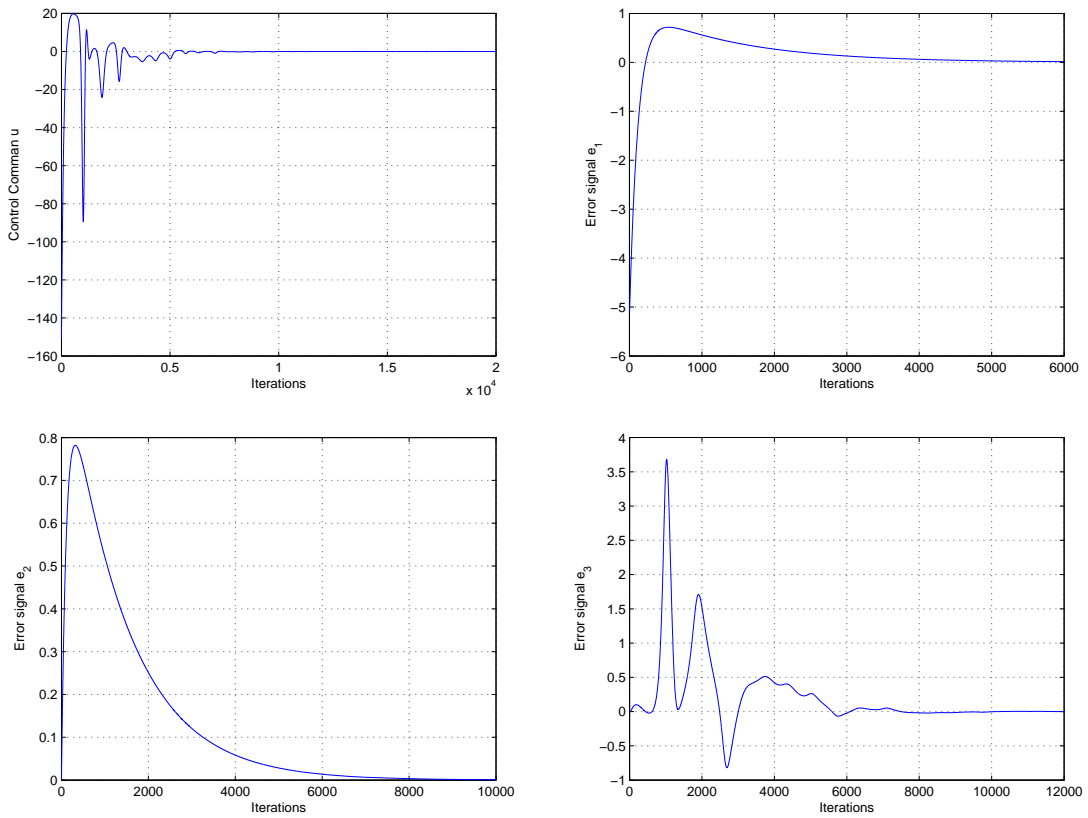


Figure 5.12. The control command u and error signals e_1 , e_2 and e_3 respectively.

Table 5.2. The source used by synchronized Lorenz system.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	650	30,720	2%
Number of 4 input LUTs	3,776	30,720	12%
Logic Distribution			
Number of occupied Slices	2,355	15,360	15%
Number of Slices containing only related logic	2,355	2,355	100%
Number of Slices containing unrelated logic	0	2,355	0%
Total Number of 4 input LUTs	4,208	30,720	13%
Number of bonded IOBs	161	448	35%
Number of DSP48s	28	192	14%
Total equivalent gate count for design	42,289		

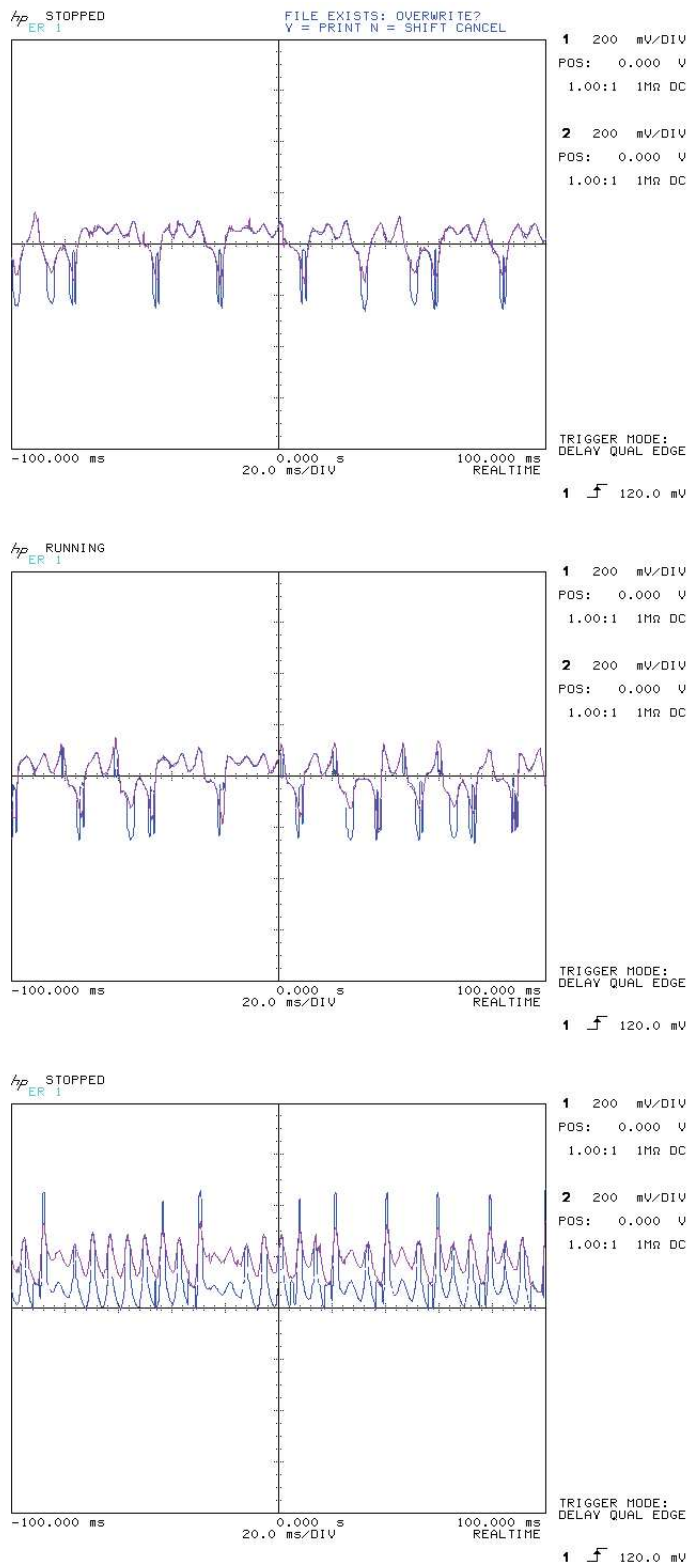


Figure 5.13. Lorenz master system state variables and Lorenz slave system state variables.

from scope screen and the blue line represents Lorenz master system state variables and the purple line represents Lorenz slave system state variables, respectively. As can be seen in Figure 5.13, the synchronization of two chaotic systems can be achieved.

Figure 5.14 represents control command u and synchronization error signals e_1 , e_2 and e_3 , respectively. As can also be seen in Figure 5.14, control command and error signals remain around zero which means that synchronization of two systems can be completed.

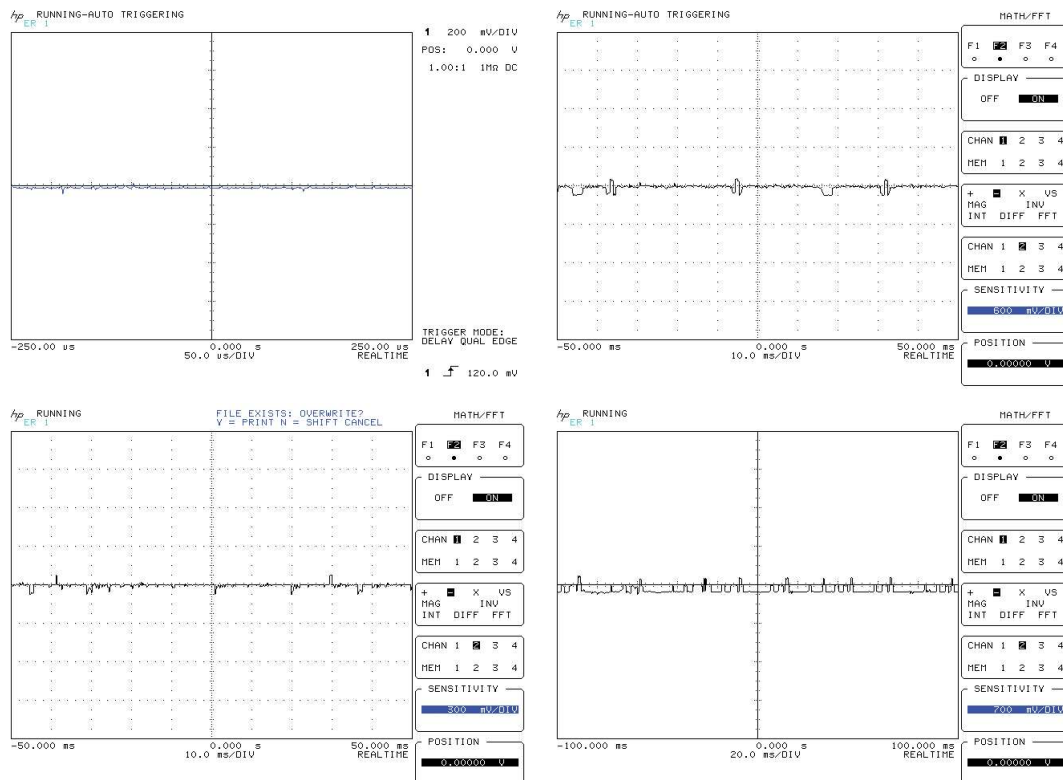


Figure 5.14. The control command u and error signals e_1 , e_2 and e_3 for Lorenz system.

The last Figures are 5.15 and 5.16. Figure 5.15 from Matlab which is drawn by using master state variables versus corresponding slave state variables. Figure 5.16 is captured from scope screen, it is drawn by using the same technique as Figure 5.15. They show similar pattern which again means that the synchronization of two chaotic systems are done.

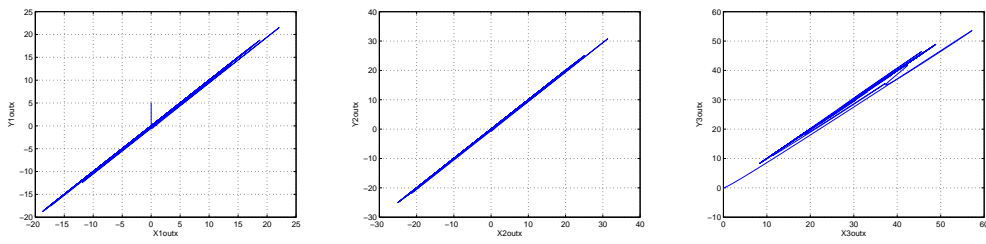


Figure 5.15. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by using System Generator.

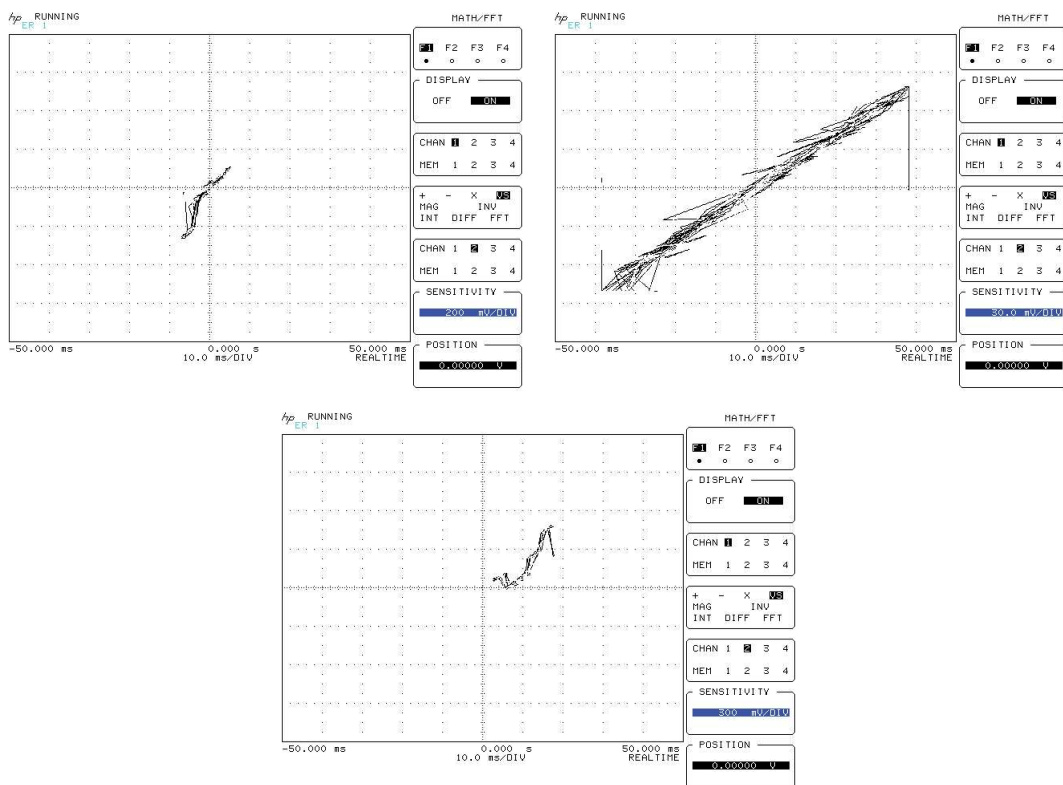


Figure 5.16. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.

5.1.2. Implementation of Synchronized Rössler System

Rössler system presented previously on Equation 3.2 is now implemented using Xilinx library's elements as can be seen in Figure 5.17 below. All integrators blocks have initial condition equal to zero, except the block *Integrator F2*, which has initial condition equal to 3.0474. The integration step dt used in this circuit equal to 0.01.

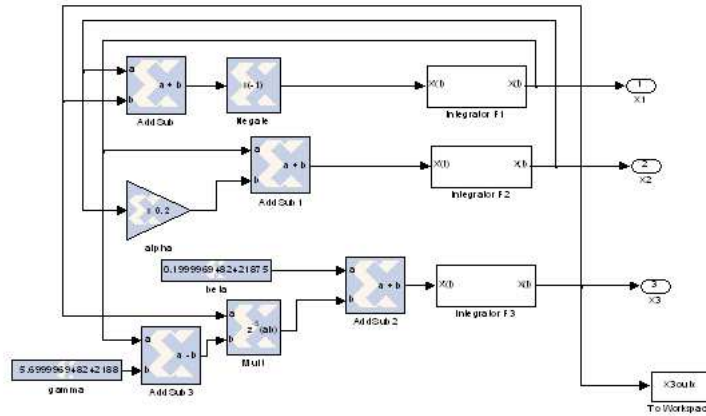


Figure 5.17. Rössler system by System Generator.

Figure 5.18 represents the attractor of Rössler system. After comparing Figure 5.18 with Figure 3.5 which was implemented by Simulink blocks, the control command u can be constructed by Xilinx blockset library's elements.

After constructing Rössler system as in Simulink design, the control command u can be created by using Xilinx blockset library's elements. Then, it is used to synchronize Rössler systems. The control command u expression is known in Equation 4.59 then, it is obtained as in Figure 5.19. .

Synchronized Rössler sytem can be completed by adding control command u to Rössler slave system and setting *Integrator F2* block in Rössler slave system equals to -5 . This system is built by using Xilinx blockset library's elements as in Figure 5.20.

Figure 5.21 represents Rössler master and Rössler slave systems trajectories, respectively. As can be seen in Figure 5.21, the starting point of two systems are different, however, they draw the same pattern after synchronization is completed.

Figure 5.22 represents control command u and synchronornization error signals e_1, e_2

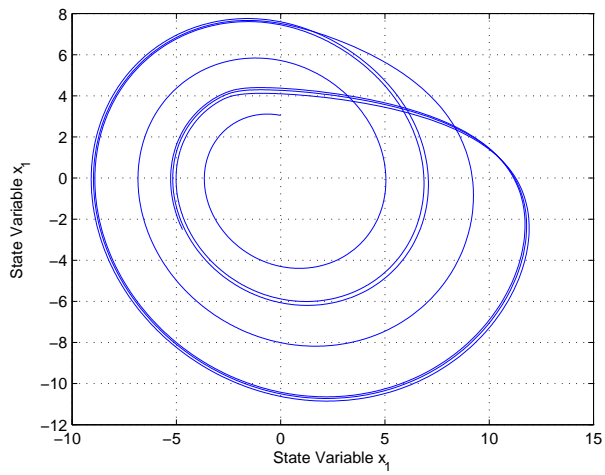


Figure 5.18. Rössler system attractor.

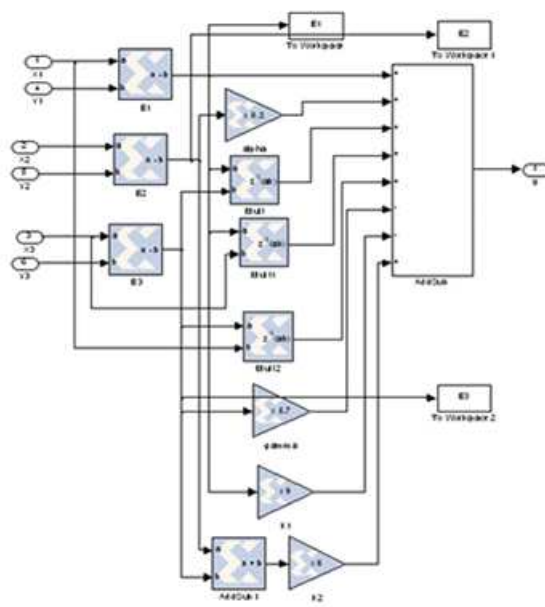


Figure 5.19. The control command u for Rössler system by System Generator.

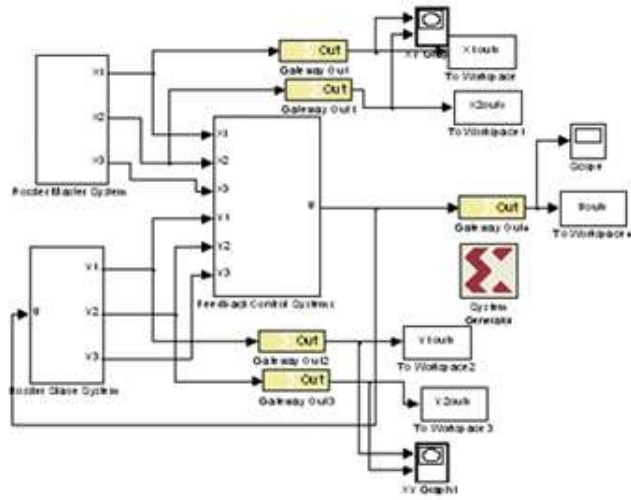


Figure 5.20. Synchronized Rössler system by System Generator.

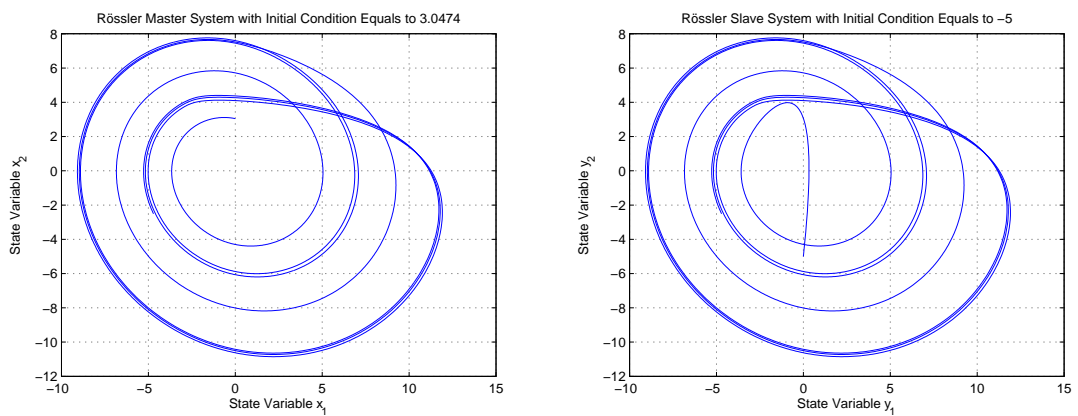


Figure 5.21. Synchronized Rössler system attractors.

Table 5.3. The source used by synchronized Rössler system.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	574	30,720	1%
Number of 4 input LUTs	1,859	30,720	6%
Logic Distribution			
Number of occupied Slices	1,153	15,360	7%
Number of Slices containing only related logic	1,153	1,153	100%
Number of Slices containing unrelated logic	0	1,153	0%
Total Number of 4 input LUTs	1,859	30,720	6%
Number of bonded IOBs	161	448	35%
Number of DSP48s	50	192	26%
Total equivalent gate count for design	21,635		

and e_3 . The synchronization of Rössler is completed when there is no difference between master and slave systems. If the error signals equal to zero then it means the synchronization of the system is achieved as in Figure 5.22.

After implementing synchronized Rössler system to FPGA, by using ISE program, how much source is consumed by synchronized Rössler system can be seen in Table 5.3.

Figure 5.23 represents Rössler master system state variables x_1 , x_2 and x_3 and Rössler slave system state variables y_1 , y_2 and y_3 , respectively. This Figure was captured from scope screen and the blue line represents Rössler master system state variables and the purple line represents Rössler slave system state variables, respectively. As can be seen in Figure 5.23, the synchronization of two chaotic systems can be achieved since master and slave system state variables have the same graphs.

As can be seen in Figure 5.24, the control command and error signals remain around zero which means that synchronization of two systems can be completed.

Figure 5.25 from Matlab which was drawn by using master state variables versus corresponding slave state variables. Figure 5.26 was captured from scope screen, it was drawn by using the same technique as Figure 5.25. They show similar pattern which again means that the synchronization of two chaotic systems are achieved.

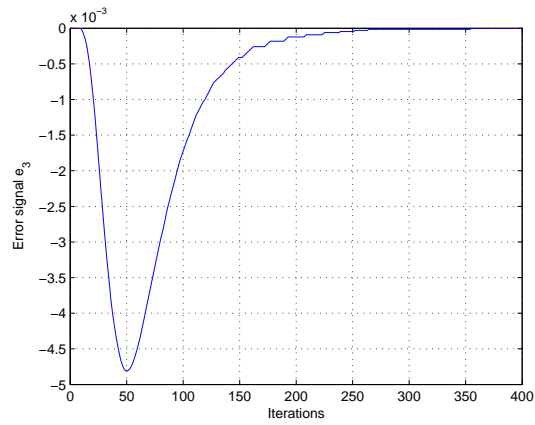
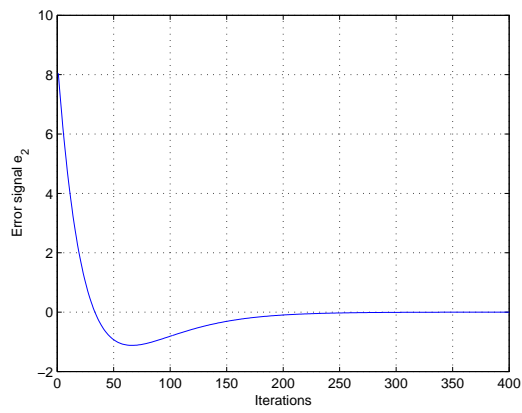
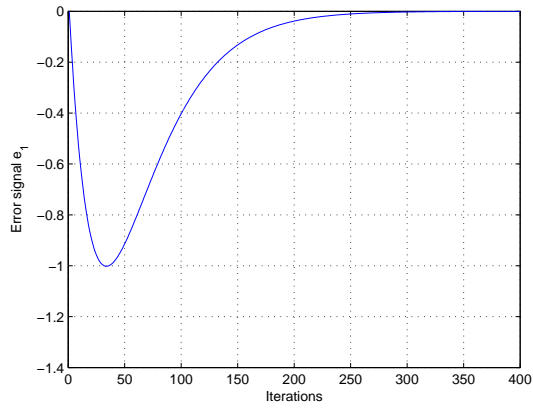
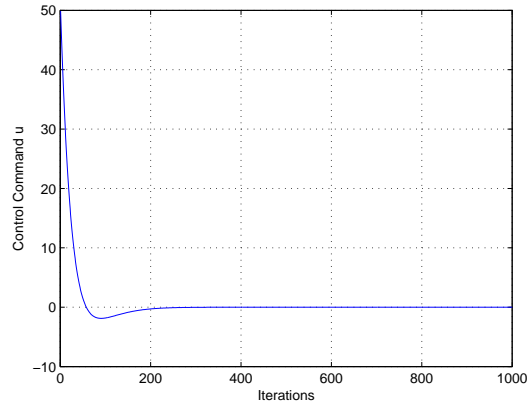


Figure 5.22. The control command u and error signals e_1 , e_2 and e_3 for Rössler system.

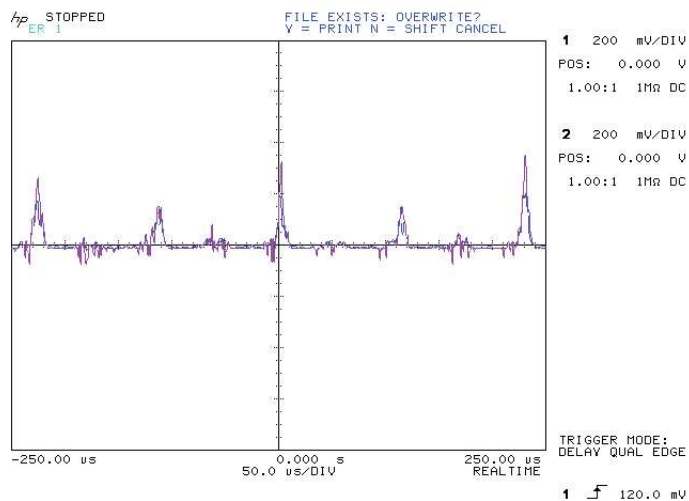
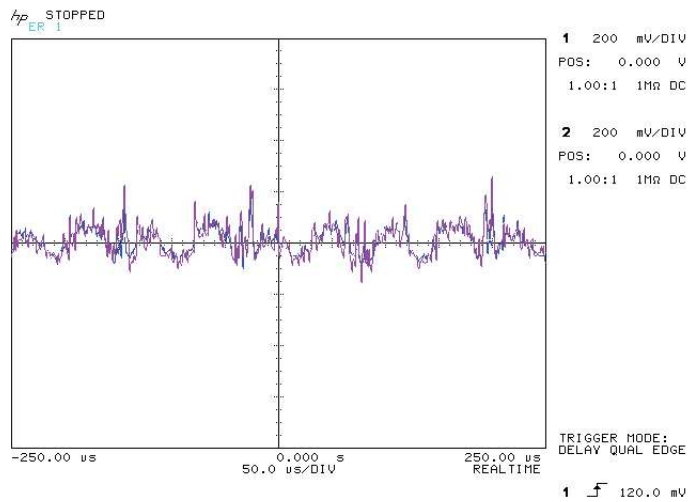
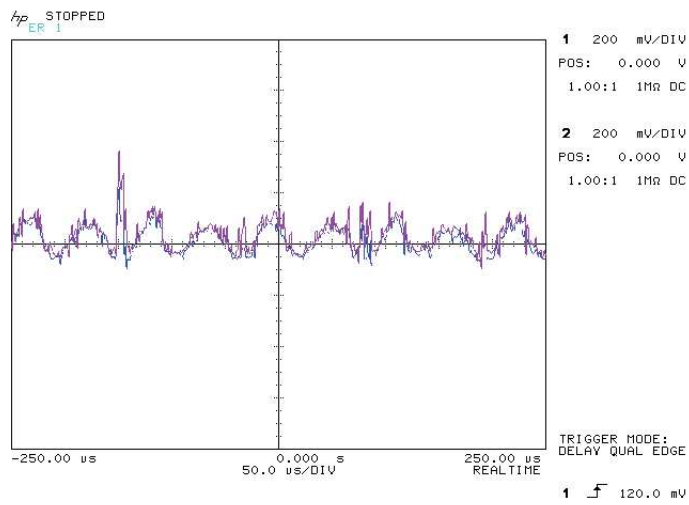


Figure 5.23. Rössler master system and Rössler slave system state variables.

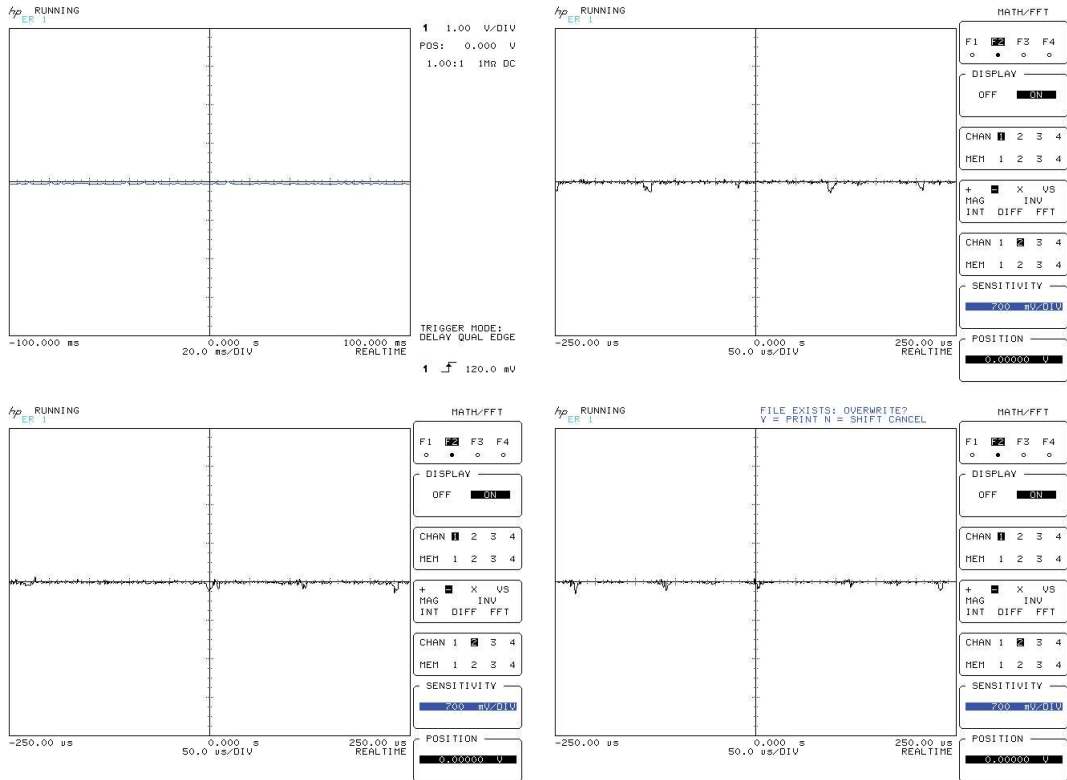


Figure 5.24. The control command u and error signals e_1 , e_2 and e_3 for Rössler system.

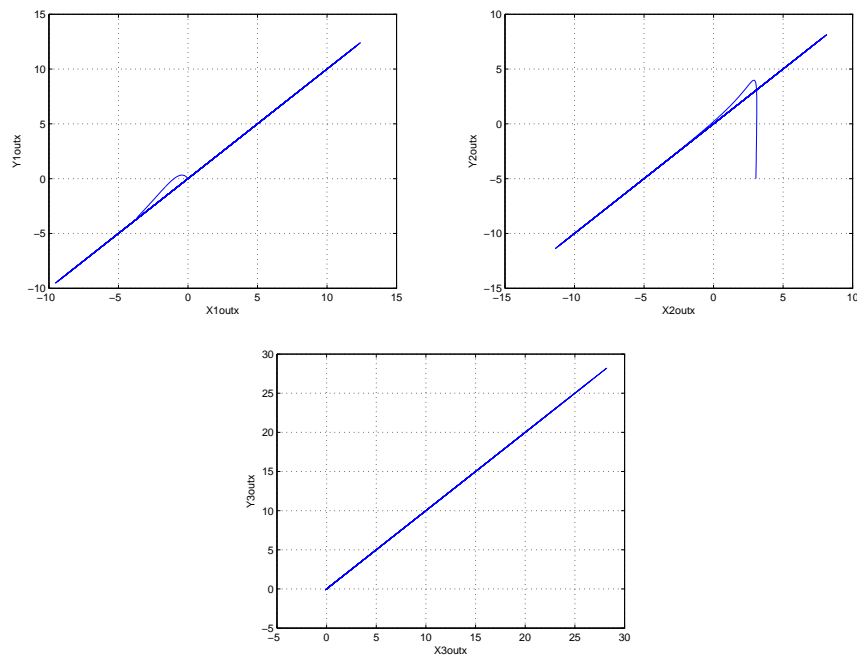


Figure 5.25. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by System Generator.

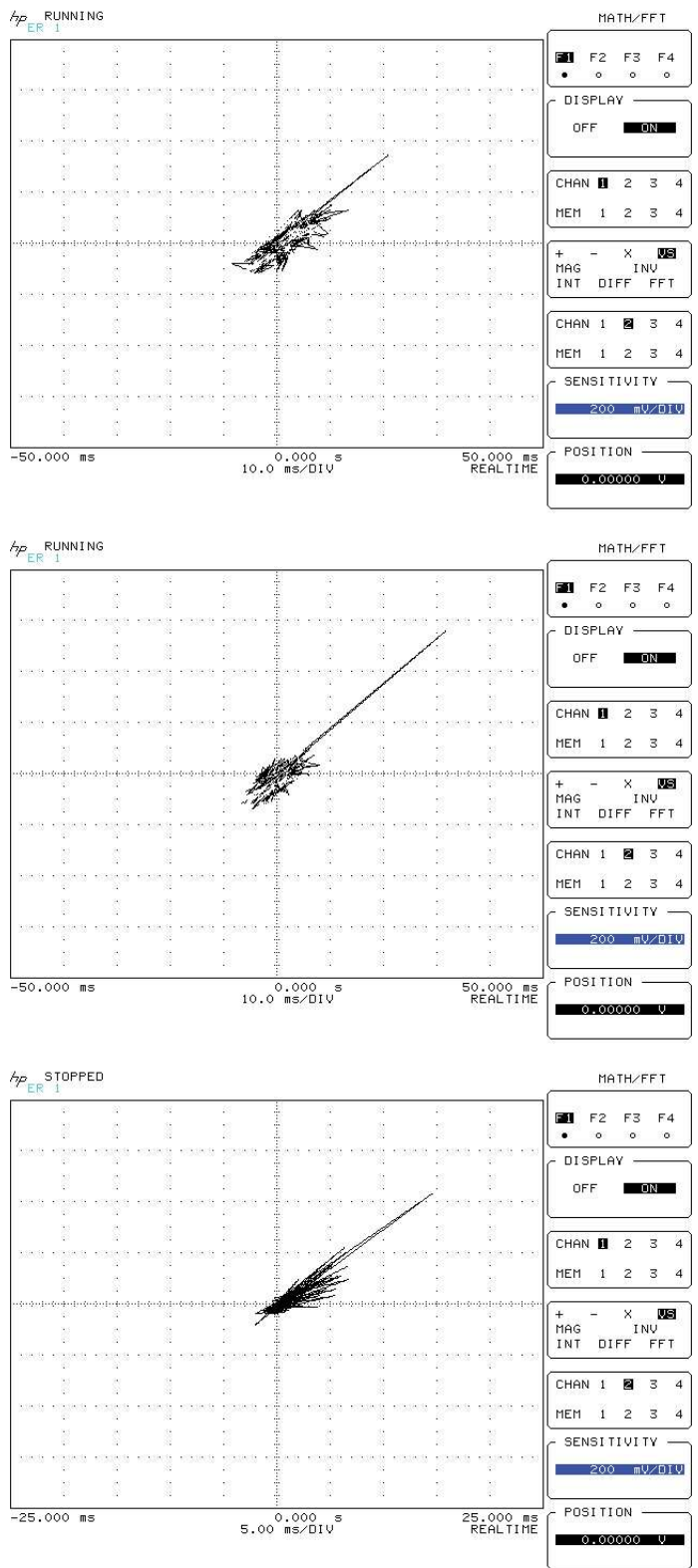


Figure 5.26. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.

5.1.3. Implementation of Synchronized Linz and Sprott System

Linz and Sprott system presented by Equations 3.3 is implemented using Xilinx blockset library's elements as can be seen in Figure 5.27 below. The system was implemented using a integration step dt equal to 0.01, the *integrator F3* has initial condition equal to 0.001 while the other two integrators has initial condition equal to zero.

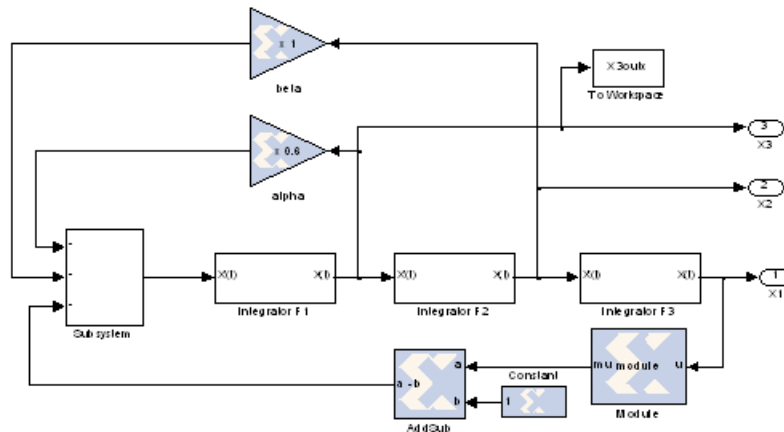


Figure 5.27. Linz and Sprott system by System Generator.

The chaotic behaviour of this system can be seen by means of its attractor in Figure 5.28.

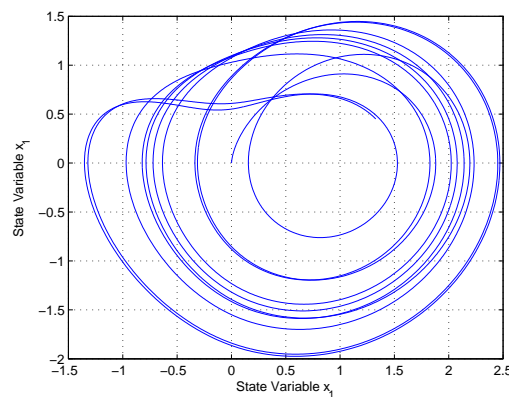


Figure 5.28. Linz and Sprott system attractor.

By using Equation 4.75 and Figure 4.9 from previous chapter, the control com-

mand u is easily constructed by using Xilinx blockset library's elements as in Figure 5.29. In order to simplify complexity of design, the subsystem was added to design and its name is *AddSub* block which is composed of 5 adders and one subtracter.

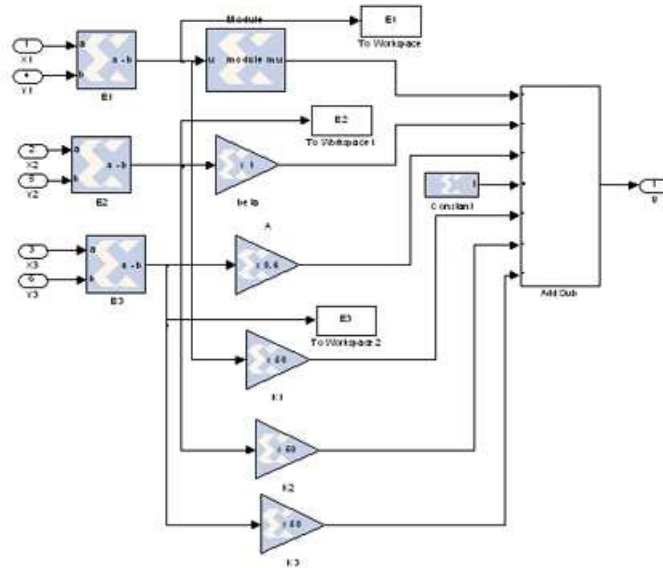


Figure 5.29. The control command u for Linz and Sprott system by System Generator.

After constructing proper control command u for Linz and Sprott system by applying it to Linz and Sprott slave system (Figure 5.30) in order to obtain slave system which tracks master system in every state. Although Linz and Sprott master system *integrator F3* block has initial condition equal to 0.001, Linz and Sprott slave system *integrator F3* block has initial condition equals to 1 as can be seen in Figure 5.31. After synchronization is completed, two attractors draw the same trajectories as in Figure 5.31.

Figure 5.32 represents the behaviour of control command u and synchronization error signals of synchronized Linz and Sprott system. When the errors of system equal to zero then slave system tracks master system in every state which means that the synchronization is completed successfully. After synchronization was done, the design can be implemented by ISE then Figure 5.33 represents Linz and Sprott master and slave state variables. The blue line represents Linz and Sprott master system state variables and the purple line represents Linz and Sprott slave system state variables. As can be seen in Figure 5.33, they are synchronized.

After implementing synchronized Linz and Sprott system to FPGA, by using ISE

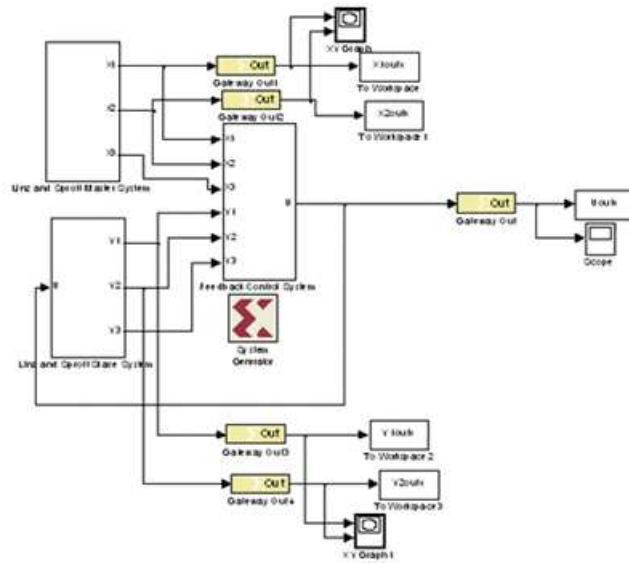


Figure 5.30. Synchronized Linz and Sprott by System Generator.

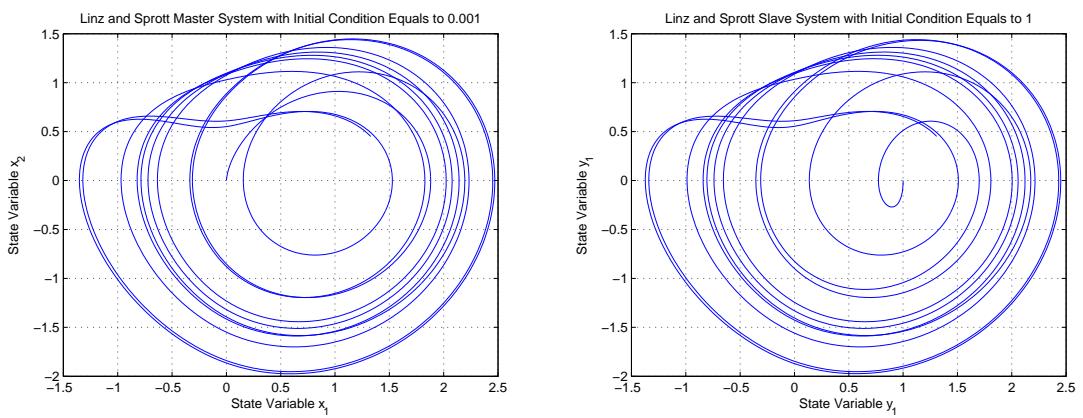


Figure 5.31. Synchronized Linz and Sprott system attractors.

program, how much source is consumed by synchronized Linz and Sprott system can be seen in Table 5.4. Because Linz and Sprott system has the simplest state space equation with respect to Lorenz, Rössler and Chua systems so it consumes minimum resource of FPGA.

Figure 5.34 represents control command u and synchronization error signals e_1 , e_2 and e_3 , respectively. As can also be seen in Figure 5.34, control command and error signals remain around zero which means that synchronization of two systems can be completed.

The last Figures are 5.35 and 5.36. Figure 5.35 from Matlab which is drawn by using master state variables versus corresponding slave state variables. Figure 5.36 is captured from scope screen, it is drawn by using the same technique as Figure 5.35. They show similar pattern which again means that the synchronization of two chaotic systems are successfully done.

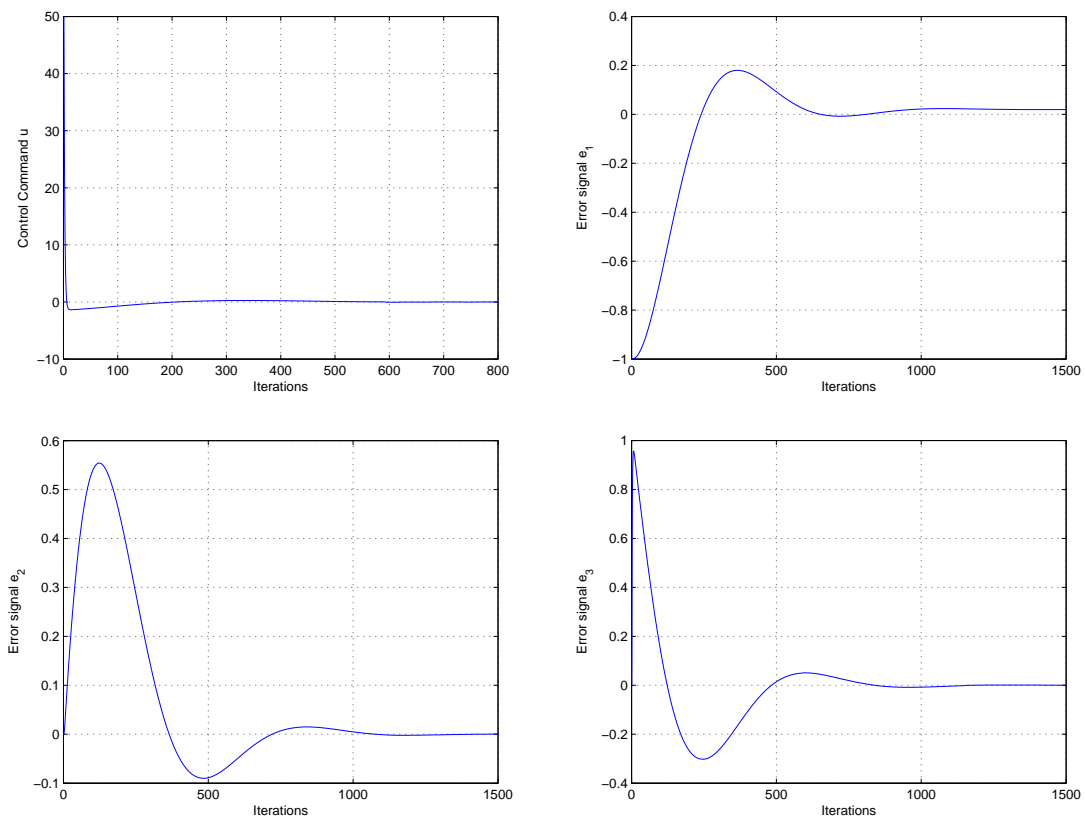


Figure 5.32. The control command u and error signals e_1 , e_2 and e_3 .

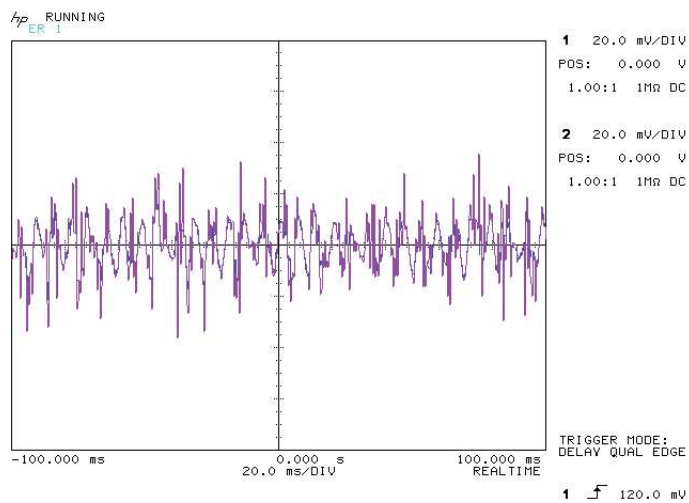
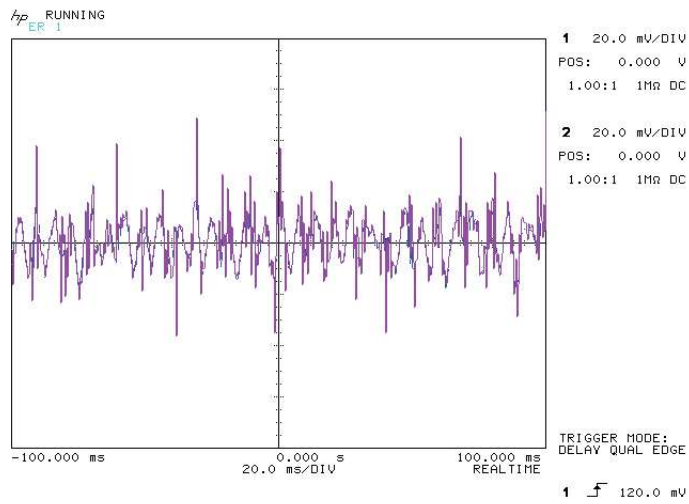
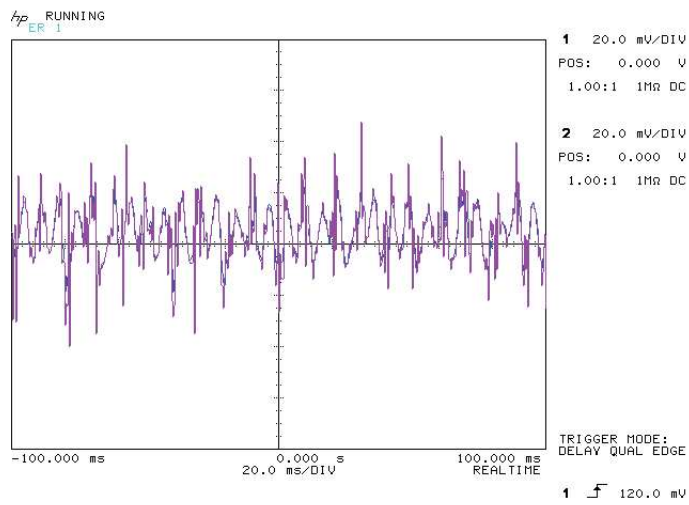


Figure 5.33. Linz and Sprott master and Linz and Sprott slave system state variables.

Table 5.4. The source used by synchronized Linz and Sprott system

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	384	30,720	1%
Number of 4 input LUTs	1,718	30,720	5%
Logic Distribution			
Number of occupied Slices	1,001	15,360	6%
Number of Slices containing only related logic	1,001	1,001	100%
Number of Slices containing unrelated logic	0	1,001	0%
Total Number of 4 input LUTs	1,718	30,720	5%
Number of bonded IOBs	161	448	35%
Number of DSP48s	30	192	15%
Total equivalent gate count for design	18,648		

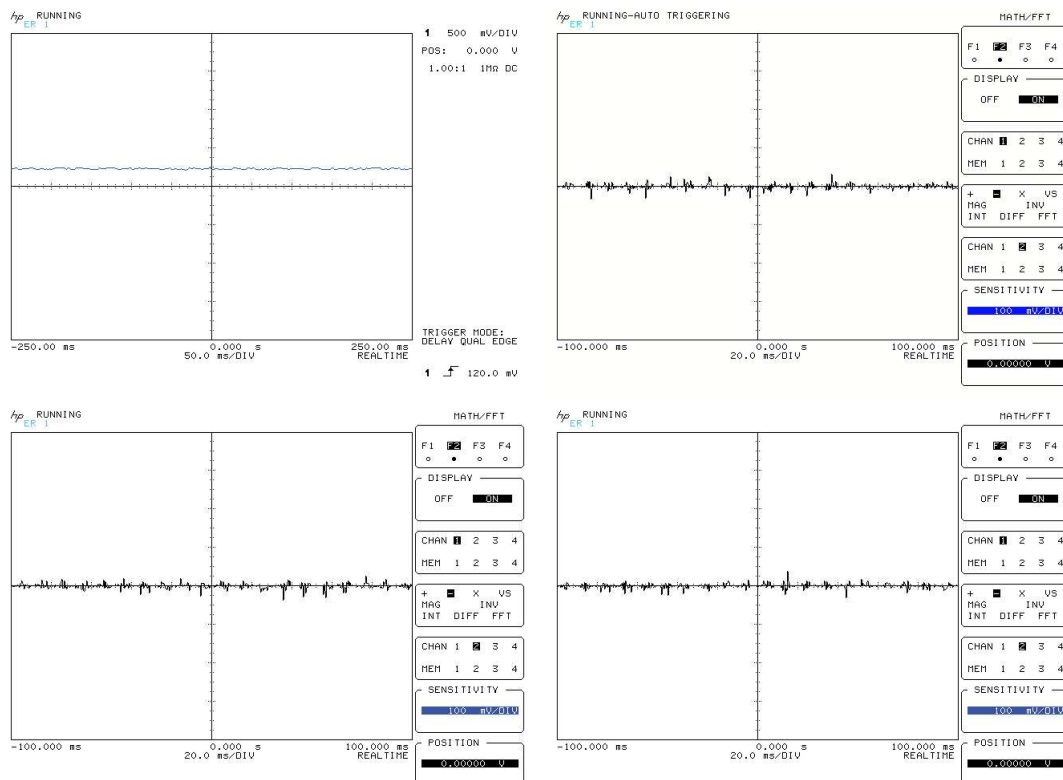


Figure 5.34. The control command u and error signals e_1 , e_2 and e_3 .

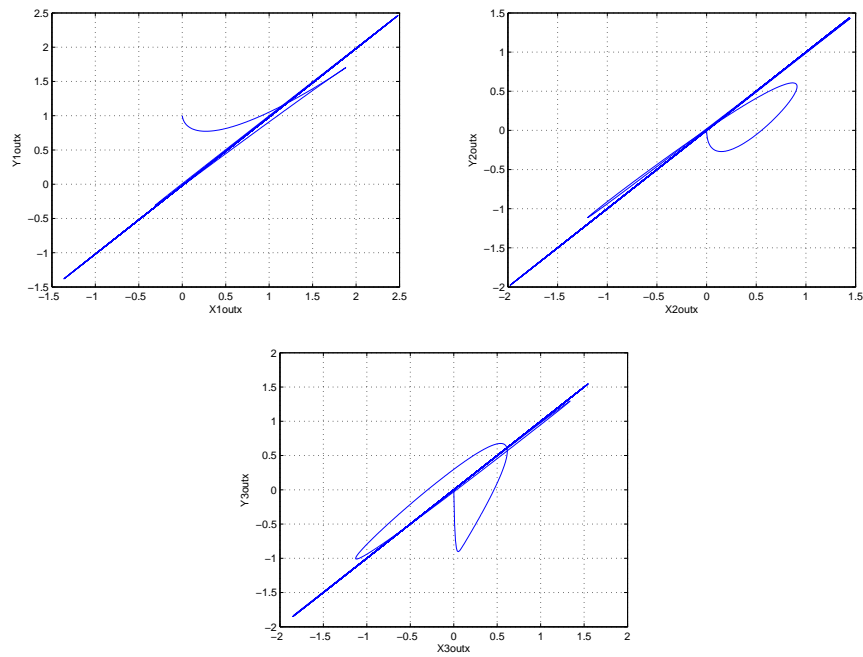


Figure 5.35. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by System Generator.

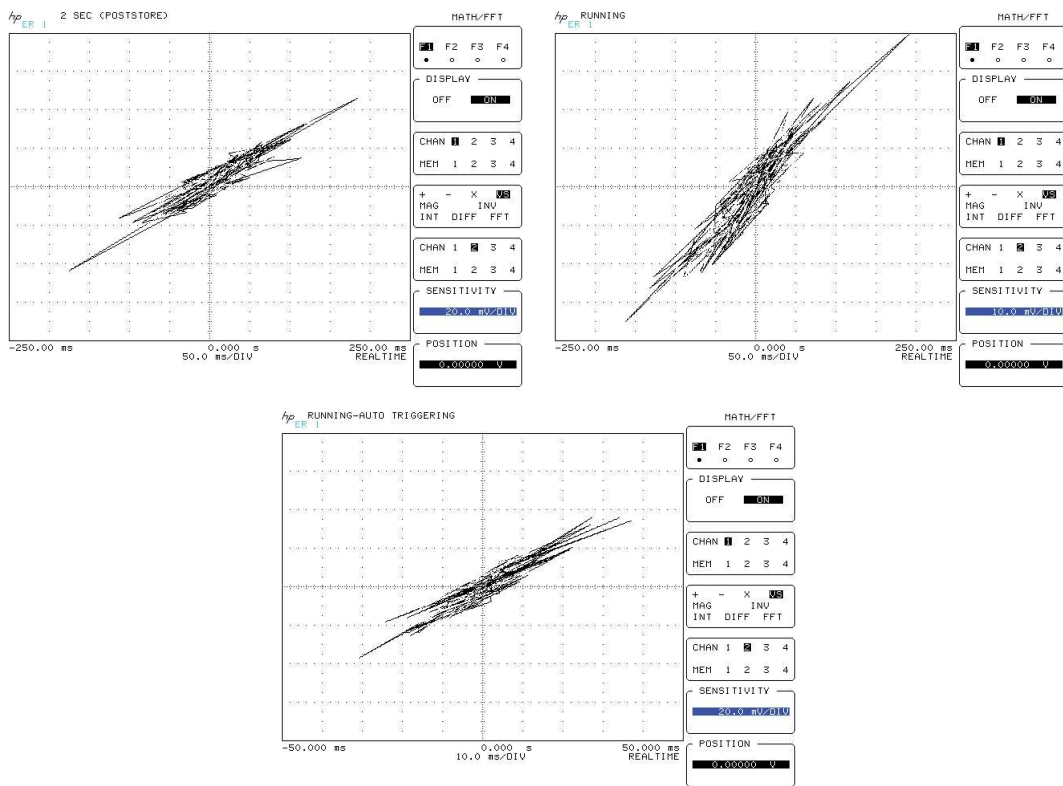


Figure 5.36. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.

5.1.4. Implementation of Synchronized Chua System

Chua system presented previously in Equations 3.7 is now implemented using Xilinx blockset library's elements as can be seen in Figure 5.37 below. The block *integrator F1* has initial condition equal to 1. The other two integrators have initial condition equal to zero. The integration step used here is $dt = 0.001$. The nonlinear element represented by function (Equation 3.8) is implemented as can be seen in Figure 5.38. In this design we used parameters which are stated below in Equations 5.1.

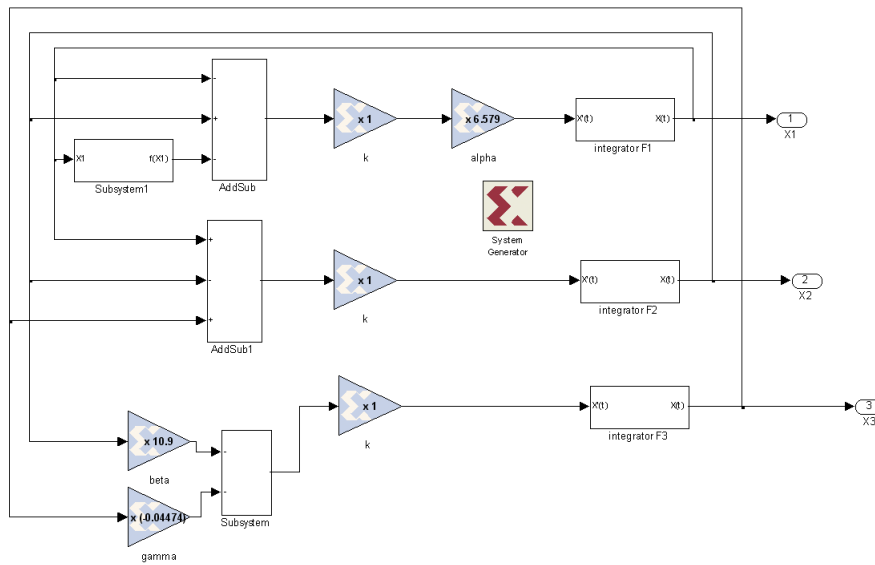


Figure 5.37. Chua system by System Generator.

$$\begin{aligned}
 \alpha &= 6.579229467 \\
 \beta &= 10.897662619 \\
 \gamma &= -0.044744029 \\
 m_0 &= -0.652335418 \\
 m_1 &= -1.811973075 \\
 k &= 1
 \end{aligned}
 \tag{5.1}$$

The same result is obtained in Figure 5.39 if we compare it with Figure 3.17. So

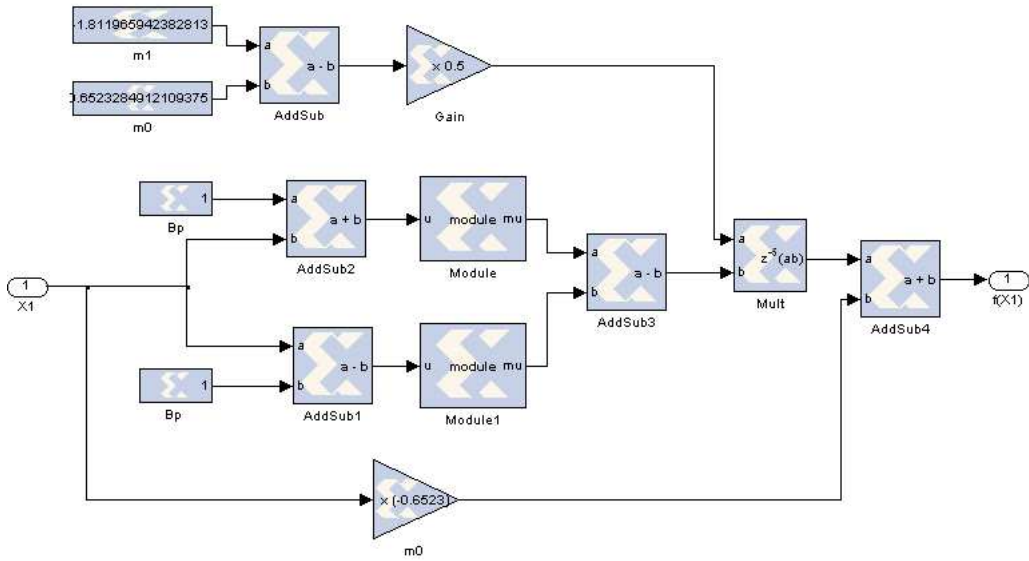


Figure 5.38. Non-linear function.

we can accomplish the same attractor behaviour of Chua system by using Xilinx blockset library's elements.

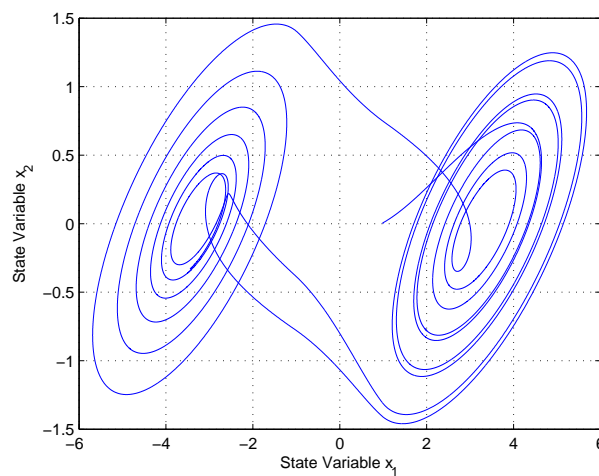


Figure 5.39. Chua system attractor.

The control command u can be constructed by using Xilinx blockset library's elements. As we know the expression of u from Equation 4.94 then we can build the control command u as in Figure 5.40 below.

After constructing the control command u then by applying it to Chua slave system, the synchronized Chua system can be completed successfully. Synchronized Chua

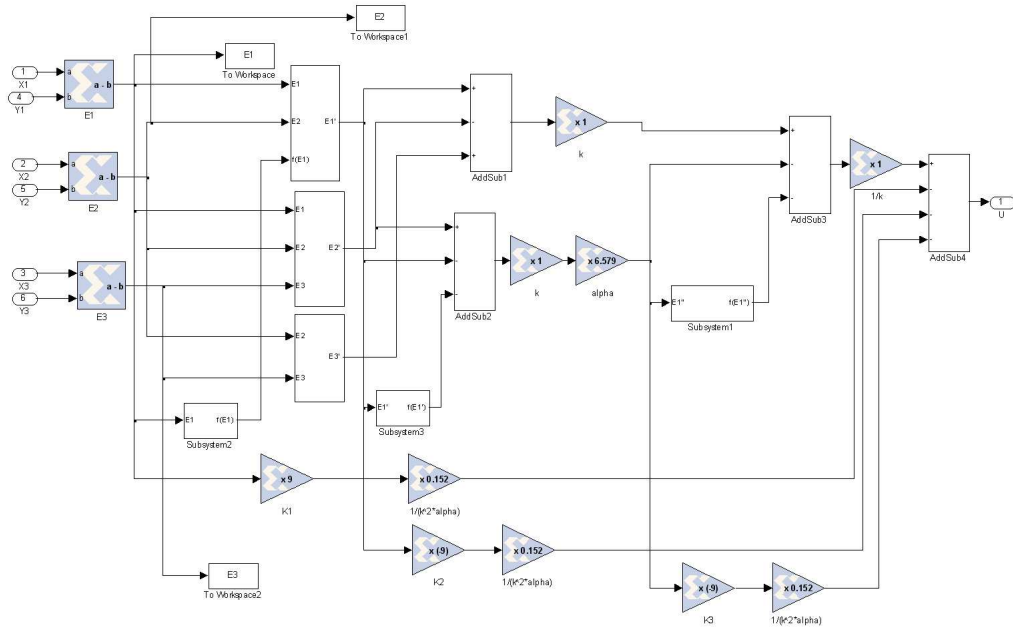


Figure 5.40. System Generator block diagram of the control command u for Chua systems.

system can be built by Xilinx blockset library's elements as in Figure 5.41.

Chua master system *Integrator F1* block initial condition is set to 1 and Chua slave system *Integrator F1* is set to 10. Although master and slave system totally different initial conditions they draw the same pattern as can be seen in Figure 5.42.

Figure 5.43 represents the behaviour of control command u and synchronization error signals of synchronized Chua system. When the errors of system equal to zero then slave system tracks master system in every state which means that the synchronization is completed successfully. After synchronization is done, the design can be implemented by ISE then Figure 5.44 represents Chua master and slave state variables. The blue line represents Chua master system state variables and the purple line represents Chua slave system state variables. As can be seen in Figure 5.44, they are synchronized.

After implementing synchronized Chua system to FPGA, by using ISE program, how much source is consumed by synchronized Chua system can be seen in Table 5.5.

As can be seen in Figure 5.45, the control command u and error signals e_1 , e_2 and e_3 remain around zero which means that synchronization of two systems can be completed.

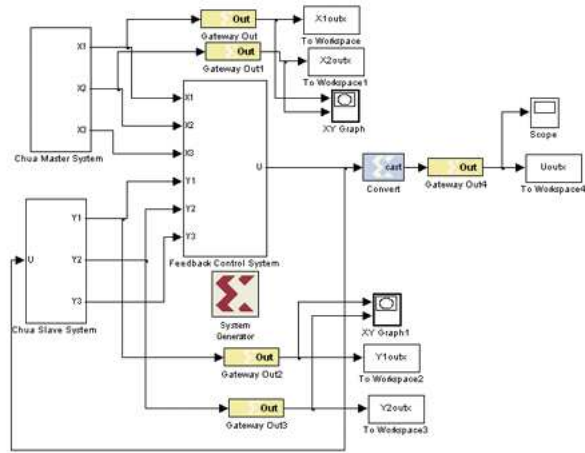


Figure 5.41. Synchronized Chua system by System Generator.

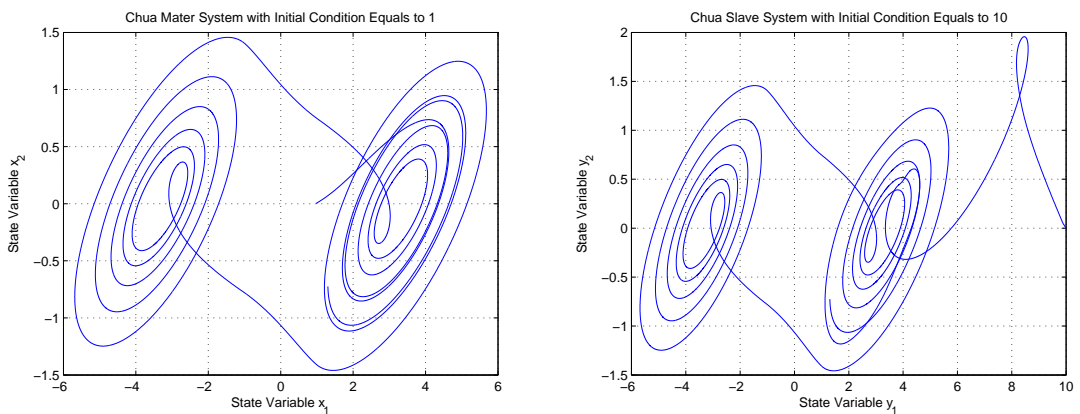


Figure 5.42. Chua master and slave system with different initial conditions.

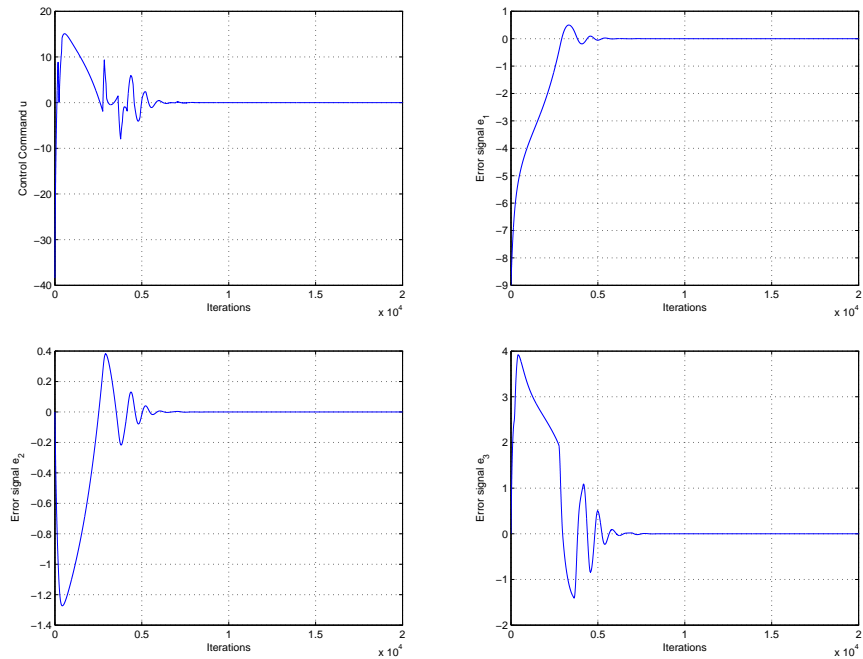


Figure 5.43. The control command u and error signals e_1 , e_2 and e_3 .

Table 5.5. The source used by synchronized Chua system.

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	590	30,720	1%
Number of 4 input LUTs	4,571	30,720	14%
Logic Distribution			
Number of occupied Slices	2,605	15,360	16%
Number of Slices containing only related logic	2,605	2,605	100%
Number of Slices containing unrelated logic	0	2,605	0%
Total Number of 4 input LUTs	4,580	30,720	14%
Number of bonded IOBs	161	448	35%
Number of DSP48s	94	192	48%
Total equivalent gate count for design	47,269		

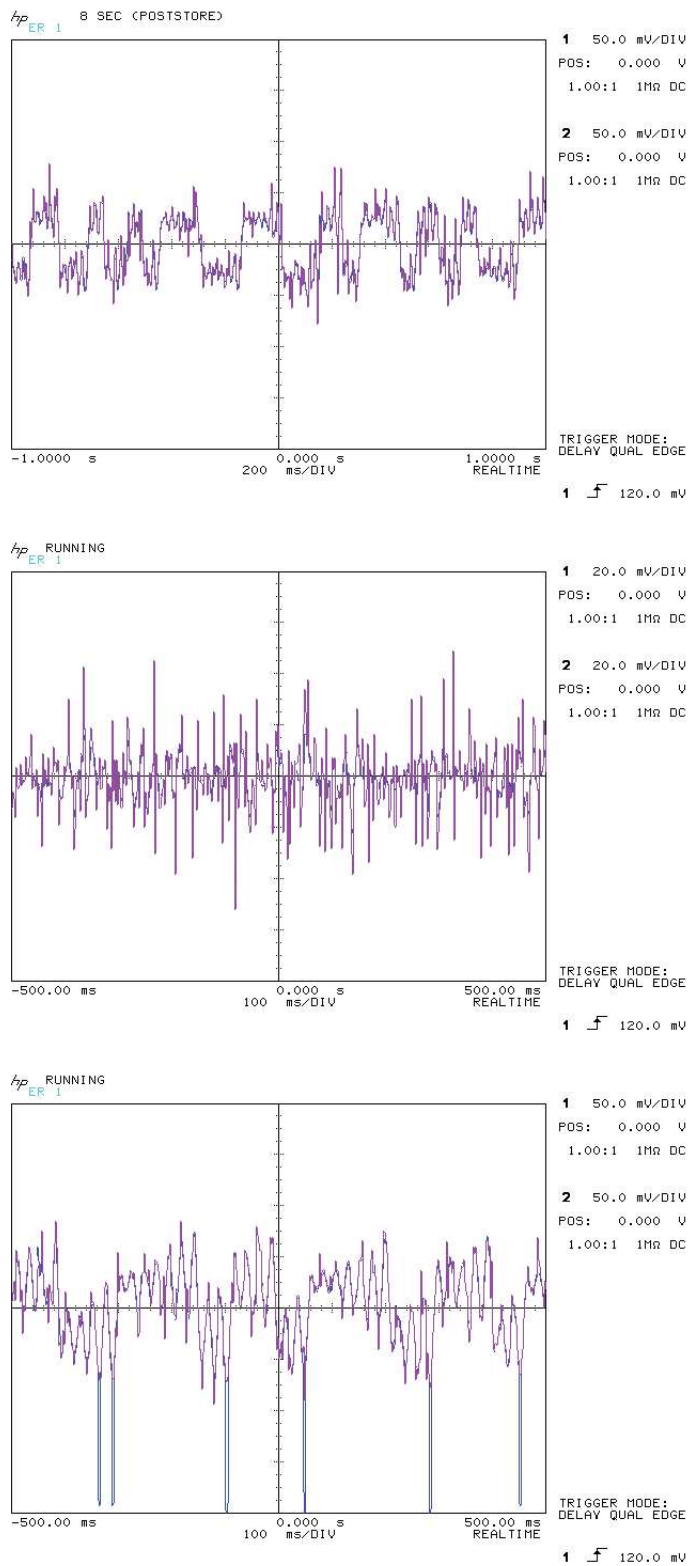


Figure 5.44. Chua master system state variables and Chua slave system state variables.

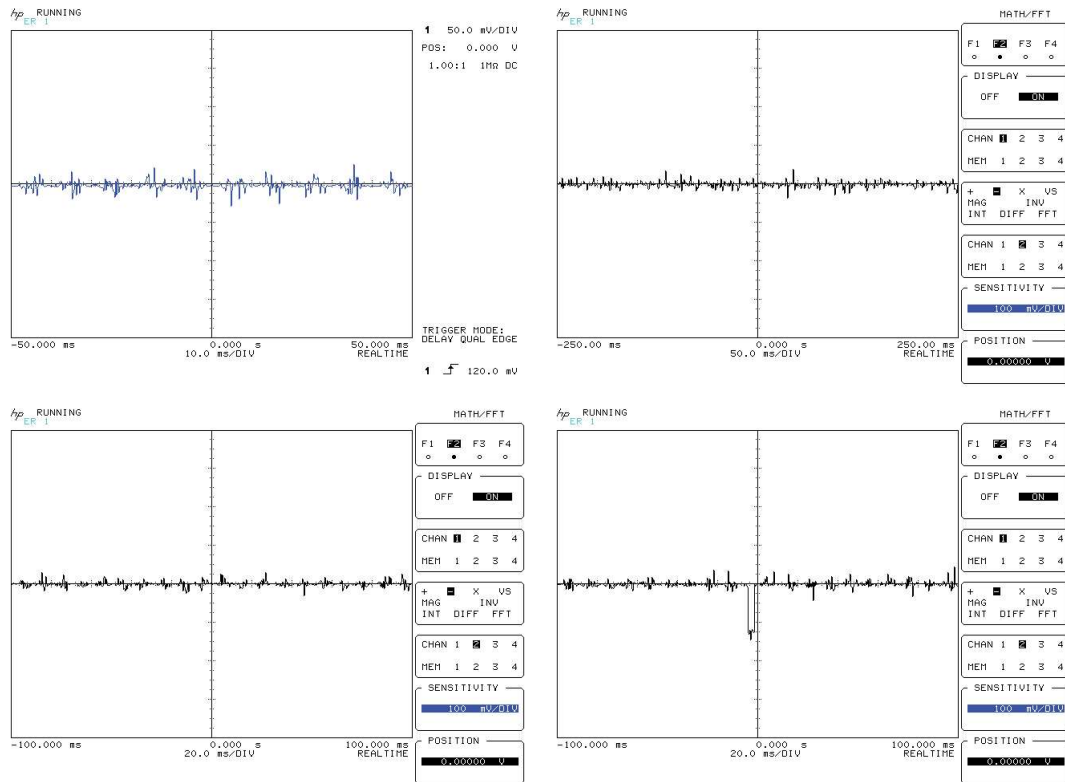


Figure 5.45. The control command u and error signals e_1 , e_2 and e_3 .

The last Figures are 5.46 and 5.47. Figure 5.46 from Matlab which is drawn by using master state variables versus corresponding slave state variables. Figure 5.47 is captured from scope screen, it is drawn by using the same technique as Figure 5.46. They show similar pattern which again means that the synchronization of two chaotic systems are done.

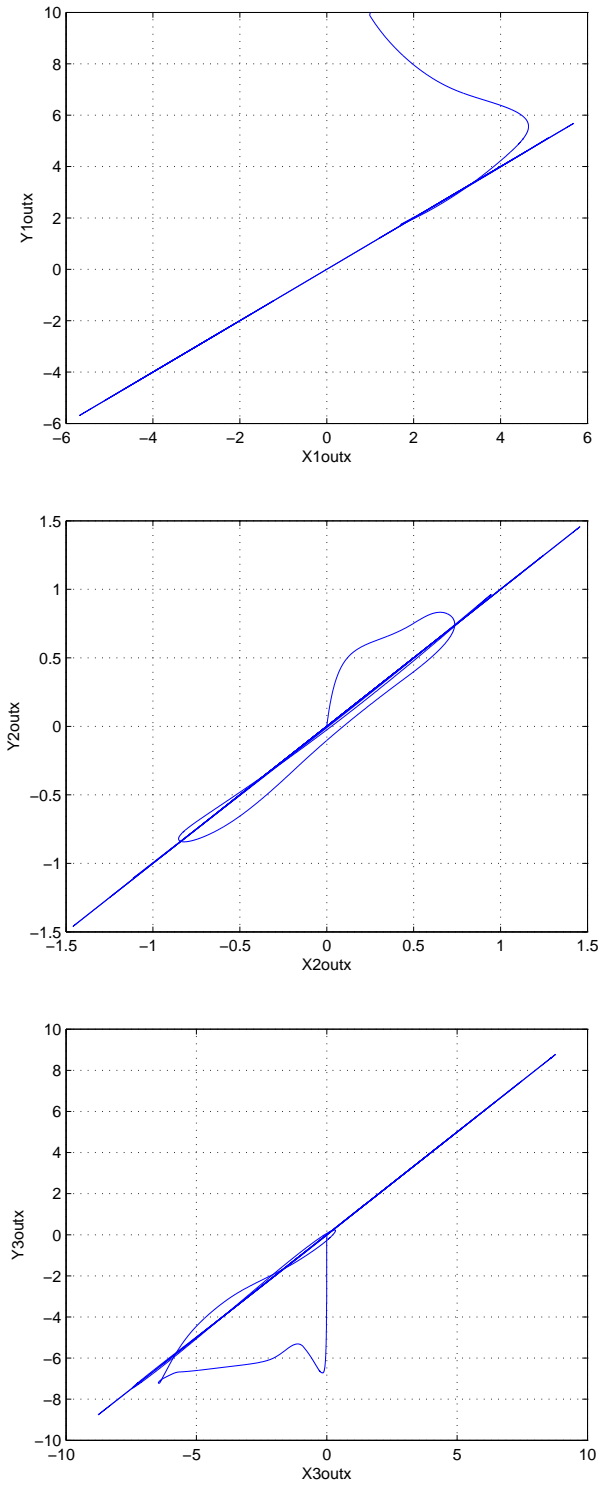


Figure 5.46. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 by System Generator.

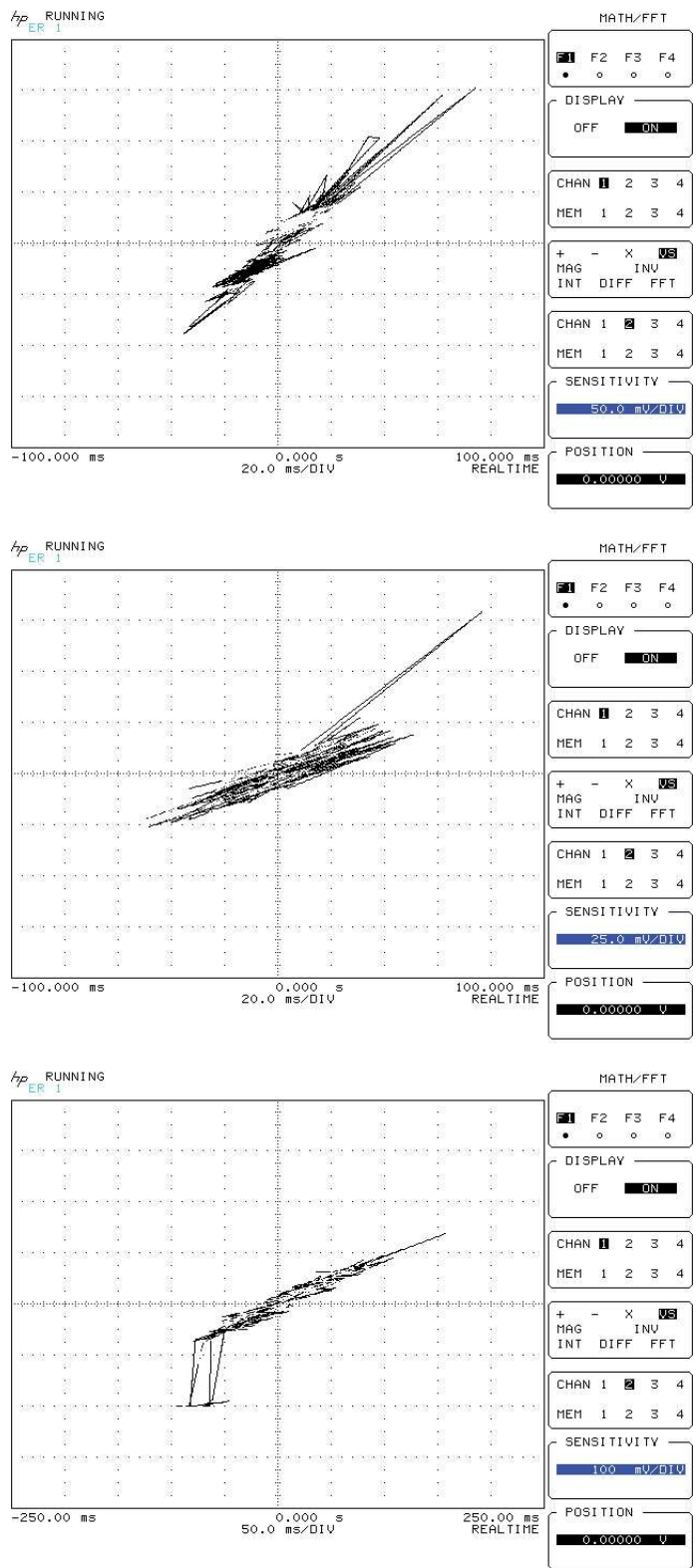


Figure 5.47. x_1 versus y_1 , x_2 versus y_2 and x_3 versus y_3 after implementation.

CHAPTER 6

CONCLUSION

A nonlinear geometric controller and chaotic systems were implemented by FPGA for synchronization. This is, a nonlinear geometric controller which was designed to synchronize each chaotic system individually. The controllers performance were such that the synchronization of Lorenz system, Rössler system, Linz and Sprott system and Chua system have been achieved. System Generators provides an efficient way to design dynamic nonlinear control system using a FPGA board with hardware simulation. Therefore re-configurable devices provide very powerful tool for the control of complex nonlinear systems.

Lorenz systems uses 42,289 logic gates in order to be implemented to FPGA. Lorenz system consumes so many logic gates because there are two multiplication calculations between state variables. However its experiment results are very good, it consumes so much source with respect to Rössler system and Linz and Sprott system.

Rössler system uses 21,645 logic gates. Rössler system is efficient at both using source and having sufficient results from synchronization implementation. There is only one nonlinearity in Rössler system which is multiplication of state variables. Although Rössler system consumes a bit more source than Linz and Sprott system, the experimental result of Rössler system is much more better than Linz and Sprott system's.

With respect to experimental results, Linz and Sprott system consumes the source of FPGA most efficiently. Linz Sprott system uses 18,468 logic gates. Since Linz and Sprott system is composed of simple equations with respect to other chaotic generators. However it consume a few source its results in experiment is not as efficient as other chaotic generators.

The last system is synchronized Chua system which uses 47269,065 logic gates. However Chua system does not include multiplication of state variables, it consumes so much source with respect to Lorenz system, Rössler system and Linz and Sprott system because Chua system's non-linear function contain so many elements to be implemented. The experimental results of Chua is not as good as Rössler, though.

REFERENCES

- Abel, A. and Schwarz, W. 2002. Chaos communications - principles, schemes and system analysis. *Proceedings of the IEEE* 90(5):691-710.
- Afraimovich, V. S., Verichev, N. N. and Rabinovich, M. I. 1987. Stochastic synchronization of oscillations in dissipative systems. *Radiophysics and Quantum Electronics* 29:795-803.
- Analog Devices. 2004. AD 7846 Datasheet.
http://www.analog.com/UploadedFiles/Data_Sheets/75667024AD7846_e.pdf
(accessed May 13, 2007).
- Andrievskii, B. R. and Fradkov, A. L. 2003. Control of Chaos : Methods and Applications. *Automation and Remote Control* 64(5):673-713.
- Blekhman, I. I., Landa, P. S. and Rosenblum, M. G. 1995. Synchronization and chaotization in interacting dynamical systems. *Applied Mechanics Review* 48:733-752.
- Brown, R. and Kocarev, L. 2000. A Unifying Definition of Chaos Synchronization for Dynamical Systems. *Chaos* 10(2):345-348.
- Chua, L. O., Kumuro, M. and Matsumoto, T. 1986. The Double Scroll Family. *IEEE Transactions on Circuits and Systems*. 33(11):1072-1118.
- Cummings M. and Haruyama S. 1999. FPGA in the Software Radio. *Communications Magazine, IEEE*. 37(2):108-112.
- Femat, R. and Solis-Perales, G. 1999. On the Chaos Synchronization Phenomena. *Physics Letters A*. 262(1):50-60.
- Leeser, M., Coric S., Miller E., Yu H. and Trepanier M. 2005. Parallel-Beam Backprojection: An FPGA Implementation Optimized for Medical Imaging. *Journal of VLSI Signal Processing* 39(3):295-311.
- Leonardo, S. I., Elvio Carlos, D. S. J. and Glesner, M. 2005. Advantages of the Linz-Sprott Weak Nonlinearity on the FPGA Implementation of Chaotic Systems: a Comparative Analysis. Paper presented at International Symposium on Signals, Circuits and Systems, July 14-15, in Iasi, Romania.
- Li, Y., Li, D. and Wang Z., 2000. A new approach to detect-mitigate-correct radiation-induced faults for SRAM-based FPGAs in aerospace application. Paper presented at National Aerospace and Electronics Conference, October 10-12, in Dayton, Ohio, USA.
- Linz, S. J. and Sprott, J. C. 1999. Elementary Chaotic Flow, *Physics Letters A* 259:240-245.
- Lorenz, N. E. 1963. Deterministic Nonperiodic Flow. *Journal of Atmospheric Science*. 20(2):130-141.

- Luethy, R. and Hoover C. 2004. Hardware and software systems for accelerating common bioinformatics sequence analysis algorithms. *Drug Discovery Today*. 2(1):12-17.
- Lund T., Aguirre M. and Torralba A. 2004. Fuzzy Logic Control via an FPGA: A Design using techniques from Digital Signal Processing. Paper presented at IEEE International Symposium on Industrial Electronics, May 4-7, in Ajaccio, France.
- Maistrenko, Yu and Popovych, O. 2000. On strong and weak chaotic partial synchronization. *International Journal of Bifurcation and Chaos*. 10:179-203.
- Marcus G. and Nolazco-Flores J. A. 2005. An FPGA-based Coprocessor for the SPHINX Speech Recognition System: Early Experiences. Paper presented at Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 20-26, in San Diego, CA, USA.
- Maxfield, Clive. 2004. *The Design Warrior's Guide to FPGAs*. Burlington: Elsevier.
- Mentes N., Sakiyama K., Batina L., Verbaauwhede I., and Preneel B. 2006. FPGA-oriented Secure Data Path Design: Implementation of a Public Key Coprocessor. Paper presented at 16th International Conference on Field Programmable Logic and Applications, August 28-30, in Madrid, Spain.
- Nijmeijer, H. and van der Schaft A. 1990. *Nonlinear Dynamical Control Systems*. New York: Springer.
- Nijmeijer, H. 2001. A dynamical control view on synchronization. *Physica D*. 154:219-228.
- Pecora, L. M. and Carrol, T. L. 1991. Synchronization in chaotic systems. *Physics Review Letter* 64(8):821-824.
- Rosenblum, M. G., Pikovsky, A. S. and Kurths, J. 1997. Phase synchronization in driven and coupled chaotic oscillators. *Physica D*. 104:219-238.
- Rössler, O. E. 1976. An Equation for Continuous Chaos. *Physics Letter 35A* 57(5):397-398.
- Sen M., Corretjer I., Haim F., Saha S., Schlessman J., Bhattacharyya S. S. and Wolf W. 2005. Computer Vision on FPGAs: Design Methodology and its Application to Gesture Recognition. Paper presented at Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 20-26, in San Diego, CA, USA.
- Solis Perales, G., Ayala, V., Kliemann, W. and Femat, R. 2003. Complete Synchronizability of Chaotic Systems: A Geometric Approach. *Chaos* 13(2):495-501.
- Vidyasagar, M., 1993. *Nonlinear System Analysis*. New Jersey: Prentice Hall.
- Wikipedia. 2007. Generic array logic. http://en.wikipedia.org/wiki/Generic_array_logic (accessed August 15, 2007).
- Wikipedia. 2007. Synchronization of Chaos. http://en.wikipedia.org/wiki/Synchronization_of_chaos (accessed September 15, 2007).

- Xilinx. 2006. ML 401/ML402/ML403 Evaluation Platform User Guide. <http://www.xilinx.com/bvdocs/userguides/ug080.pdf> (accessed May 24, 2006).
- Xilinx. 2007. Xilinx System Generator User's Guide 9.1.0.1. http://www.xilinx.com/support/sw_manuals/sysgen_ug.pdf (accessed April 10, 2007).
- Xilinx. 2007. Xilinx ISE 9.1i Software Manuals. http://www.xilinx.com/support/sw_manuals/xilinx9/download/qst.zip (accessed April 23, 2007).