



**HÜCRESEL YAPAY SİNİR AĞI VE KAOTİK
DEVRE TEMELLİ RASGELE SAYI
ÜRETECİ TASARIMLARI**

Bariş KARAKAYA

Doktora Tezi

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Arif GÜLTEN

MART-2019

**T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

**HÜCRESEL YAPAY SİNİR AĞI VE KAOTİK DEVRE TEMELLİ RASGELE
SAYI ÜRETECİ TASARIMLARI**

DOKTORA TEZİ

Barış KARAKAYA

(142113201)

Tezin Enstitüye Verildiği Tarih : 23 Ocak 2019

Tezin Savunulduğu Tarih : 1 Mart 2019

Tez Danışmanı : Doç. Dr. Arif GÜLTEN (F.Ü)

Diğer Jüri Üyeleri : Prof. Dr. Müştak Erhan YALÇIN (İ.T.Ü)

Prof. Dr. Ahmet Bedri ÖZER (F.Ü)

Doç. Dr. Mustafa TÜRK (F.Ü)

Doç. Dr. Erdiñç AVAROĞLU (M.Ü)

MART-2019

ÖNSÖZ

2015 yılında gösterime giren “Yapay Oyun: Enigma” adlı sinema filmi, şifreleme bilimine ilgi duymamı sağladı. Bu ilgi, “Kriptoloji” adlı lisansüstü dersi ile birlikte artarak doktora tez konumu rasgele sayı üretimi alanında belirlememe vesile oldu.

Doktora eğitimim boyunca bilgi, birikim ve yardımlarını esirgemeyen; akademik kariyerim boyunca beni daima doğruya yönlendiren çok değerli danışman hocam Doç. Dr. Arif GÜLTEN’e en içten saygı ve teşekkürlerimi sunarım. Ayrıca lisans eğitimimden beri bana güvenen, başarılı olmama büyük katkıları olan ve hiçbir yardımı benden esirgemeyen Doç. Dr. Mustafa TÜRK’e en içten saygı ve teşekkürlerimi sunarım. Doktora tez konumu belirlemede büyük role sahip ve bu konuda sağladığı her türlü bilgi birikimi ve yardımlarından ötürü Prof. Dr. Ahmet Bedri ÖZER’e teşekkürlerimi borç bilirim.

Yaşamım ve eğitimim boyunca birlikte karşılaştığımız her türlü maddi manevi zorluklara karşın hep dirayetli durup sabırlı olan, çocuklarını kendilerinden daha çok düşünen ve her türlü desteklerini benden esirgemeyen babam Etem KARAKAYA’ya annem Serpil KARAKAYA’ya ve kardeşlerime çok teşekkür ederim. Evlendiğimiz günden beri iyi ve kötü günümde daima yanımda olan ve bana dünyadaki en değerli hediye oğlum Yunus Emre KARAKAYA’yı veren eşim Ayşe KARAKAYA’ya en içten sevgi ve teşekkürlerimi sunarım.

Doktora tez çalışmalarımın 6 aylık bir periyodu, University of Catania’da tamamlanmıştır. Bu vesileyle, sağladıkları burs desteği için TÜBİTAK BİDEB’e ve danışmanlığı için Prof. Dr. Mattia FRASCA’ya teşekkürlerimi sunarım. Ayrıca doktora tezime MF.18.15 proje numarasıyla sağladıkları destek için Fırat Üniversitesi Bilimsel Araştırma Projeleri (FÜBAP) birimine teşekkürlerimi borç bilirim.

Barış KARAKAYA

ELAZIĞ-2019

İÇİNDEKİLER

	Sayfa No
ÖNSÖZ	I
İÇİNDEKİLER	II
ÖZET	IV
SUMMARY	V
ŞEKİLLER LİSTESİ	VI
TABLolar LİSTESİ	VIII
SEMBOLLER LİSTESİ	IX
KISALTMALAR	X
1. GİRİŞ	1
1.1. Tezin Amacı	3
1.2. Tezin İçeriği	3
1.3. Tezin Yapısı	4
2. RASGELE SAYI ÜRETEÇLERİ	5
2.1. Rasgele Sayı Üretecinin Tanımı	5
2.2. Rasgele Sayı Üretecinden Beklenen Özellikler	5
2.3. İstatistiksel Testler	6
2.4. Rasgele Sayı Üretecinin Yapısı	8
2.4.1. Entropi Kaynağı	9
2.4.2. Örnekleyici	10
2.4.3. Son İşlem	10
2.5. Rasgele Sayı Üreteci Türleri	11
2.5.1. Sözde Rasgele Sayı Üreteci	11
2.5.2. Gerçek Rasgele Sayı Üreteci	12
2.5.3. Hibrit Rasgele Sayı Üreteci	16
3. KAOTİK DEVRELER VE SAYISAL TASARIMLARI	17
3.1. Kaotik Devrelerin Özellikleri	17
3.2. Diferansiyel Denklemlerin Ayrıklaştırılması	18
3.2.1. Taylor Serisi Açılımı	18
3.2.2. Runge-Kutta Yöntemi	19
3.2.3. Euler Yöntemi	19
3.3. Literatürdeki Bazı Kaotik Devreler ve Analizleri	20
3.3.1. Lojistik Harita	21
3.3.2. Lorenz Sistemi	23
3.3.3. Chua Osilatör Devresi	26
3.3.4. 3D Kaotik Sistem	29
3.3.5. Memristörlü Kaotik Devre	32
3.4. Sahada Programlanabilir Kapı Dizileri (FPGA) ve Gerçekleştirilen Tasarımlar	36
3.4.1. Sayısal Tasarım Aşamaları	37
3.4.2. Xilinx Sistem Üreteci (XSG) ve Kaotik Devre Tasarımları	39
3.4.3. 3D Kaotik Sistemi ile RSÜ Tasarımı	46
4. HÜCRESEL YAPAY SİNİR AĞI TABANLI RSÜ UYGULAMALARI	51
4.1. HYSA Modeli	51
4.2. Üç Hücreli Otonom HYSA Modeli ve RSÜ Uygulaması	53

4.2.1. Üç Hücreli Otonom HYS A Modelinin Donanım Tabanlı Benzetimi	56
4.2.2. Üç Hücreli Otonom HYS A Modeli ile FPGA Ortamında RSÜ Tasarımı.....	58
4.3. Memristörlü HYS A Modeli ve RSÜ Uygulaması.....	71
4.3.1. Memristörlü HYS A Modelinin Donanım Tabanlı Benzetimi	74
4.3.2. Memristörlü HYS A Modeli ile FPGA Ortamında RSÜ Tasarımı	76
5. SONUÇ VE ÖNERİLER.....	83
KAYNAKLAR.....	85
ÖZGEÇMİŞ	89



ÖZET

Bu tezde, kriptografik uygulamalarda kullanılan Rasgele Sayı Üretecinin (RSÜ) yapısı, türleri ve özellikleri detaylı olarak incelenmiş olup kaotik sistem tabanlı RSÜ tasarımları ile FPGA (Field Programmable Gate Array- Sahada Programlanabilir Kapı Dizileri) kartında gerçek zamanlı olarak rasgele sayı/bit üretimi gerçekleştirilmiştir. Ayrıca literatürde var olan bazı kaotik sistemler detaylı olarak incelenmiş ve sistemlerin XSG (Xilinx System Generator, Xilinx Sistem Üreteci) platformunda donanım tabanlı benzetimleri gerçekleştirilmiştir.

Tez çalışmasında üç farklı RSÜ tasarımı önerilmiştir. Tasarımlar, XSG platformunda donanım tabanlı benzetimleri gerçekleştirildikten sonra Xilinx ISE FPGA Editor arayüzünde Verilog HDL (Verilog Hardware Description Language- Verilog Donanım Tanımlama Dili) ile kodlanmıştır. RSÜ tasarımları sonucunda elde edilen bit dizilerinin rasgeleliliği NIST (National Institute of Standards and Technology- Ulusal Standartlar ve Teknoloji Enstitüsü) 800.22 istatistiksel test paketi ile test edilmiştir. Ayrıca bit dizilerinin aperiodyiklik derecesi, ölçek indeksi (scale index) algoritması kullanılarak hesaplanmıştır.

Tezde, önerilen ilk RSÜ tasarımı bir benzetim çalışmasıdır. Tasarımda entropi kaynağı olarak üç durum değişkenine sahip kaotik bir sistem ve son işlem olarak Trivium şifreleme algoritması kullanılmıştır. Tasarımın XSG platformunda donanım tabanlı benzetimi gerçekleştirilerek üretilen bitlerin rasgeleliliği test edilmiştir. İkinci RSÜ tasarımında entropi kaynağı olarak kaotik davranış sergileyen üç hücreli otonom Hücresel Yapay Sinir Ağı (HYSA) modeli ve son işlem olarak Trivium şifreleme algoritması kullanılarak üç farklı senaryo için bit üretimi gerçekleştirilmiştir. Böylece tasarım, 2.015 Mbps hızında rasgele bit üretimi sağlamıştır. Üçüncü RSÜ tasarımında ise entropi kaynağı olarak Memristörlü kaotik HYSA modeli ile birlikte lojistik harita sistemi kullanılmıştır. İki kaotik sistemin çıkışının örneklenip sayısallaştırılması sonucu elde edilen bit dizileri XOR son işlemine uygulanarak rasgele bit üretimi gerçekleştirilmiştir. Tasarım sonucunda 125 kbps hızında ve çok düşük FPGA kaynak kullanımı ile rasgele bit üretimi sağlandığı gösterilmiştir.

Anahtar Kelimeler: Kaos, Memristör, Rasgele Sayı Üreteci, İstatistiksel Test, FPGA.

SUMMARY

Designs of Cellular Neural Network and Chaotic Circuit Based Random Number Generator

In this thesis, the architecture, types and features of Random Number Generator (RNG) used in cryptographic applications have been examined in detail, real time random number/bit generation has been implemented on Field Programmable Gate Array (FPGA) board by exercising chaos based RNG designs. Moreover, some of chaotic systems in the literature have been examined in detail and hardware based simulations of systems have been realized in Xilinx System Generator (XSG) platform.

In this thesis study, three different RNG designs were proposed. After the hardware based realizations of the designs in XSG platform, RNG designs have been coded by using Verilog Hardware Description Language (HDL) through the Xilinx ISE FPGA Editor interface program. The statistical randomness of bit streams derived from the RNG designs were validated with National Institute of Standards and Technology (NIST) 800.22 test suite. Furthermore, the degree of non-periodicity of the bit streams has been determined by adopting scale index algorithm.

In the thesis, the first proposed RNG design is a simulation study. In the design, a chaotic system with three state variables and Trivium encryption algorithm have been made use of as entropy source and post processor, respectively. By examining hardware based simulation of the design in XSG platform, the generated bit stream has been tested statistically. In the second proposed RNG design, making use of three cell autonomous Cellular Neural Network (CNN) model that represents chaotic behavior as entropy source and Trivium encryption algorithm as post processor, bit stream generation has been realized for three different scenarios. Thus, the design performed random bit generation with an output bit rate of 2.015 Mbps. In the third proposed RNG design, Memristive chaotic CNN model and logistic map were utilized as entropy source. The random bit generation has been examined by sampling and digitizing the output of two chaotic systems then applying to XOR post processor. As a result of performed design, it is demonstrated that random bit generation was achieved with an output bit rate of 125 kbps and very low FPGA resource utilization.

Keywords: Chaos, Memristor, Random Number Generator, Statistical Test, FPGA.

ŞEKİLLER LİSTESİ

	Sayfa No
Şekil 2.1. Kriptografik uygulamaların temel amacı	8
Şekil 2.2. Basit bir RSÜ yapısı	9
Şekil 2.3. SRSÜ tasarımına ait genel bir gösterim	11
Şekil 2.4. IRO XOR tabanlı GRSÜ mimarisi	14
Şekil 2.5. İkili entropi çekirdekli ayırık zamanlı kaos tabanlı GRSÜ mimarisi	15
Şekil 2.6. Hibrit RSÜ'nün mimarisi	16
Şekil 3.1. Lojistik harita sisteminin $r \in [1,4]$ aralığı için çatallaşma diyagramı	22
Şekil 3.2. Lojistik harita sisteminin $r = 4$ için Lyapunov üstel değişimi	23
Şekil 3.3. Lorenz sisteminin üç boyutlu yörünge değişimi	25
Şekil 3.4. Lorenz sisteminin $b \in [0,1]$ aralığı için çatallaşma diyagramı	25
Şekil 3.5. Lorenz sisteminin Lyapunov üstellerinin değişimi	26
Şekil 3.6. Chua osilatörü devre şeması	26
Şekil 3.7. Chua osilatör devresinin a) $x - y$ ve b) $x - z$ faz portreleri	28
Şekil 3.8. Chua osilatör devresinin $\alpha \in [6,11]$ aralığı için çatallaşma diyagramı	28
Şekil 3.9. Chua osilatör devresinin Lyapunov üstellerinin değişimi	29
Şekil 3.10. 3D kaotik sisteminin üç boyutlu yörünge değişimi	30
Şekil 3.11. 3D kaotik sisteminin $b \in [0,2,3,5]$ aralığı için çatallaşma diyagramı	31
Şekil 3.12. 3D kaotik sisteminin Lyapunov üstellerinin değişimi	31
Şekil 3.13. Memristörün içyapısı	32
Şekil 3.14. Memristörün a) iki uçlu eleman gösterimi b) akı (q) – yük (φ) karakteristiği	33
Şekil 3.15. Memristörlü kaotik devre şeması	34
Şekil 3.16. Memristörlü kaotik devrenin üç boyutlu yörünge değişimi	35
Şekil 3.17. Memristörlü kaotik devrenin $\alpha \in [0,1]$ aralığı için çatallaşma diyagramı	35
Şekil 3.18. Memristörlü kaotik devrenin Lyapunov üstellerinin değişimi	36
Şekil 3.19. FPGA'nın içyapısı	37
Şekil 3.20. FPGA ortamında sayısal tasarımın akış şeması	38
Şekil 3.21. $Qm.n$ sabit noktalı sayı aritmetiğinin gösterimi	39
Şekil 3.22. XSG platformunda tasarımın akış şeması	40
Şekil 3.23. Lojistik harita sisteminin XSG modeli	40
Şekil 3.24. Lojistik haritanın XSG modelinin benzetim sonucu	41
Şekil 3.25. XSG modelinin derleme ekranı	42
Şekil 3.26. Lojistik haritanın XSG modelinin son hali	42
Şekil 3.27. Lojistik haritanın XSG modelinin donanım tabanlı benzetim sonucu	43
Şekil 3.28. Lorenz sisteminin XSG modeli	44
Şekil 3.29. Lorenz sisteminin üç boyutlu donanım tabanlı benzetim sonucu	45
Şekil 3.30. Memristörlü kaotik devrenin XSG modeli	45
Şekil 3.31. Memristörlü kaotik devrenin üç boyutlu donanım tabanlı benzetim sonucu	46
Şekil 3.32. 3D kaotik sisteminin XSG modeli	47
Şekil 3.33. 3D kaotik sisteminin üç boyutlu donanım tabanlı benzetim sonucu	48
Şekil 3.34. Trivium algoritması için giriş dizilerinin oluşturulması	49
Şekil 4.1. $m \times n$ boyutlu HYSA mimarisine ilişkin standart bir gösterim	51
Şekil 4.2. Üç hücreli otonom HYSA modeline ait akış diyagramı	54
Şekil 4.3. Üç hücreli otonom HYSA modelinin $p2 \in [0,5,2]$ aralığı için çatallaşma diyagramı	55

Şekil 4.4. Üç hücreli otonom HYS A modelinin Lyapunov üstellerinin değişimi.....	55
Şekil 4.5. Üç hücreli otonom HYS A'nın XSG modeli	57
Şekil 4.6. Üç hücreli otonom HYS A'nın XSG modeline ait üç boyutlu yörüngesi	57
Şekil 4.7. Üç hücreli otonom HYS A modelinin $x_1[n]$ durum değişkeninin değişimi.....	58
Şekil 4.8. $x_1[n]$ durum değişkeninin karesel ortalama değerinin değişimi	59
Şekil 4.9. Önerilen üç hücreli otonom HYS A tabanlı RSÜ tasarımı na ait ilk senaryo	59
Şekil 4.10. Trivium algoritmasının kapı elemanları düzeyinde gösterimi	60
Şekil 4.11. Önerilen üç hücreli otonom HYS A tabanlı RSÜ tasarımı na ait ikinci senaryo ...	62
Şekil 4.12. Önerilen üç hücreli otonom HYS A tabanlı RSÜ tasarımı na ait üçüncü senaryo.	63
Şekil 4.13. Önerilen üç hücreli otonom HYS A tabanlı RSÜ tasarımı na ait uygulama platformu	63
Şekil 4.14. Üç hücreli otonom HYS A modelinin FPGA gerçekleştirim sonucu a) $x_1 - x_2$ b) $x_1 - x_3$ c) $x_2 - x_3$ faz portreleri	64
Şekil 4.15. Üç hücreli otonom HYS A tabanlı RSÜ tasarımı na ait blok gösterimi	65
Şekil 4.16. Tasarlanan aritmetik işlem modülleri a) toplama b) çarpma modülü	66
Şekil 4.17. Önerilen üç hücreli otonom HYS A tabanlı RSÜ tasarımı na ait üçüncü senaryonun benzetim sonucu.....	67
Şekil 4.18. a) Önerilen üç hücreli otonom HYS A tabanlı RSÜ tasarımı nı gerçekleştirme platformu b) üçüncü senaryoya ait üretilen rasgele bitlerin osiloskop görüntüsü	67
Şekil 4.19. Üç hücreli otonom HYS A modeli ile tasarlanan RSÜ çıktı larının ölçek indekslerinin değişimi a) ilk b) ikinci ve c) üçüncü senaryo için sonuç lar	70
Şekil 4.20. Kanonik Chua osilatör devresi	71
Şekil 4.21. Memristörlü HYS A modeline ait devre şeması	72
Şekil 4.22. Memristörlü HYS A'nın XSG modeli	75
Şekil 4.23. Memristörlü HYS A'nın XSG modeline ait faz portreleri a) $x - y$ b) $x - z$ c) $x - w$ d) $y - z$ e) $y - w$ f) $z - w$	76
Şekil 4.24. Önerilen Memristörlü HYS A tabanlı RSÜ tasarımı	77
Şekil 4.25. a) Memristörlü HYS A modeli ve b) lojistik harita sisteminin x durum değişkenlerinin değişimi.....	77
Şekil 4.26. α ve r parametre değişimlerine karşın elde edilen ortalama ölçek indeksi değerleri.....	78
Şekil 4.27. Önerilen Memristörlü HYS A tabanlı RSÜ tasarımı nın XSG donanım tabanlı benzetim modeli	79
Şekil 4.28. Önerilen Memristörlü HYS A tabanlı RSÜ tasarımı na ait ölçek indeksinin değişimi	80
Şekil 4.29. Önerilen Memristörlü HYS A tabanlı RSÜ tasarımı nın FPGA editöründeki benzetim sonucu	80
Şekil 4.30. Önerilen Memristörlü HYS A tabanlı RSÜ tasarımı nın gerçek zamanlı çalışma platformu ve osiloskop ekran çıktısı.....	81

TABLÖLAR LİSTESİ

	Sayfa No
Tablo 3.1. Lojistik haritanın XSG modelinin FPGA kaynak kullanımı	43
Tablo 3.2. XSG modellerinin FPGA kaynak kullanımları	46
Tablo 3.3. 3D kaotik sisteminin XSG modelinin FPGA kaynak kullanımı	48
Tablo 3.4. 3D kaotik sistem tabanlı RSÜ çıkışının NIST istatistiksel test sonuçları	50
Tablo 4.1. Üç hücreli otonom HYSA'nın XSG modelinin FPGA kaynak kullanımı	58
Tablo 4.2. Tasarlanan aritmetik işlem modüllerinin FPGA kaynak kullanımı	66
Tablo 4.3. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait üçüncü senaryonun FPGA kaynak kullanımı	66
Tablo 4.4. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımlarına ait NIST istatistiksel test sonuçları	68
Tablo 4.5. Memristörlü HYSA'nın XSG modelinin FPGA kaynak kullanımı	75
Tablo 4.6. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımının literatürdeki benzer çalışmalar ile karşılaştırılması	81
Tablo 4.7. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımına ait NIST istatistiksel test sonuçları	82

SEMBOLLER LİSTESİ

K	: Anahtar
p_i	: i. olayın meydana gelme olasılığı
$H(p)$	: p olasılığının Shannon ölçütü
S_0	: SRSÜ tohum dizisi
S_1	: SRSÜ iç durum dizisinin ilk değeri
S_n	: SRSÜ iç durum dizisi
R_n	: Rasgele sayı dizisi
$\Psi_H()$	: Çıkış fonksiyonu
$\Phi_H()$	: Geçiş fonksiyonu
V	: Gerilim, Volt
MHz	: MegaHertz
Mbps	: Mega bit per second
$x(t)$	: Zamana bağlı durum değişkeni
h	: Ayrık zaman adım aralığı
k_1, k_2	: 1. ve 2. dereceden Runge-Kutta eğimleri
x_0	: Durum değişkeni başlangıç değeri
$\bar{x}$	: Durum değişkeni denge noktası
λ	: Lyapunov üstel değeri

KISALTMALAR

RSÜ	: Rasgele Sayı Üreteci
SRSÜ	: Söзде Rasgele Sayı Üreteci
GRSÜ	: Gerçek Rasgele Sayı Üreteci
HRSÜ	: Hibrit Rasgele Sayı Üreteci
FPGA	: Field Programmable Gate Array (Sahada Programlanabilir Kapı Dizileri)
XSG	: Xilinx System Generator (Xilinx Sistem Üreteci)
FE	: Forward Euler (İleri Euler)
RK	: Runge-Kutta
RK4	: 4. dereceden Runge-Kutta
NIST	: National Institute of Standards and Technology
HYSA	: Hücresel Yapay Sinir Ağı
XOR	: Exclusive-OR
AES	: Advanced Encryption System (Gelişmiş Şifreleme Standardı)
DES	: Data Encryption System (Veri Şifreleme Standardı)
RSA	: Rivest-Shamir-Adleman
PSoC	: Programmable System on Chip (Çip üzerinde Programlanabilir Sistem)
LUT	: Look-up-Table (Doğruluk Tablosu)
IRO	: Inverter Ring Oscillator (Evirici Halka Osilatörü)
VHDL	: VHSIC Hardware Description Language
IEEE	: Institute of Electrical and Electronics Engineers
CT	: Continuous Time (Sürekli Zamanlı)
DT	: Discrete Time (Ayrık Zamanlı)
DTC	: Discrete Time Chaos (Ayrık Zamanlı Kaos)
TRNG	: True Random Number Generator
CMOS	: Complementary Metal Oxide Semiconductor
ANN	: Artificial Neural Network (Yapay Sinir Ağı)
PLD	: Programmable Logic Device (Programlanabilir Mantık Aygıtı)
CLB	: Configurable Logic Block (Konfigüre edilebilir Mantık Bloğu)
IOB	: Input Output Block (Giriş Çıkış Bloğu)
BRAM	: Block Random Access Memory (Rasgele Erişimli Hafıza Bloğu)
FF	: Flip Flop

LUT : Look up Table (Doğruluk tablosu)
SC-CNN : State Controlled Cellular Nonlinear/Neural Network
DK-HYSA : Durum Kontrollü Hücresel Yapay Sinir Ağı



1. GİRİŞ

Günümüzde teknoloji, belirli bir zaman aralığına bağlı olmaksızın çok hızlı değişim göstermektedir. İnsanlar, en son üretilen ve daha iyi özelliklere sahip ürünleri kullanarak elde edecekleri bilgileri en hızlı şekilde gerek sosyal medya olarak adlandırılan platformlarda gerekse çalışma alanları ile ilgili platformlarda paylaşmak isterler. Bu durum gerçek veya tüzel kişilere ait bilginin güvenli ortamlarda ve güvenli bir şekilde saklanması, iletilmesi ve paylaşılmasını önemli kılar. Bu nedenle sosyal içerikli internet sitelerinden askeri alanlarda kullanılan yazılımlara kadar tüm haberleşme ortamları, güvenlik gereksinimlerini üst seviyede tutmalarını sağlayacak yapılara ihtiyaç duymaktadırlar. Bu yapılar güvenli haberleşme (*secure communication*) sistemi olarak adlandırılırlar [1].

Kriptografi, bilgi güvenliği için kullanılan şifreleme bilimidir. Kriptografi, haberleşme sistemlerinde alıcı ile verici arasında paylaşılan bilginin üçüncü şahıslardan veya siber saldırılardan korunması için bilgiyi, verici çıkışında şifreleyip (*encryption*) ve alıcı kısmında ise şifre çözerek (*decryption*) güvenli bir şekilde iletilmesini sağlar [1].

Şifreleme, gizli bir anahtar (*key*) yardımı ile saf veriyi (*plane text*) güvensiz bir ortamda alıcıya ulaştırılması için anlamsız hale (şifreli metin, *cipher text*) getirmektir. Şifre çözme ise bu işlemin tam tersidir. İki işlemde de aynı anahtar kullanılırsa, alıcı ve verici arasındaki ortam güvenli olmasa da şifreli haberleşmiş diyebiliriz. Bu işlemlerde kullanılan anahtar, açığa çıkarılamayacak şekilde gizli ve tahmin edilemez ise güvenli haberleşme sağlanmış olur. Bu nedenle kullanılan anahtar rasgele seçilmelidir. Anahtar ne kadar rasgele seçilir ise haberleşme sistemi o kadar güvenlidir. Sonuç olarak kriptografi rasgele üretilen anahtar değerine ihtiyaç duymaktadır [2].

Kriptografik anahtar, kriptografi ile rasgele sayılar arasında bir bağlantı kurar. Rasgele sayılar ise rasgele sayı üreteçleri (RSÜ) ile üretilirler. Rasgele sayı üreteçleri yapısal bakımdan Sözde Rasgele Sayı Üreteci (SRSÜ), Gerçek Rasgele Sayı Üreteci (GRSÜ) ve Hibrit Rasgele Sayı Üreteci (HRSÜ) olmak üzere üç ana gruba ayrılırlar. SRSÜ'ler tohum (*seed*) olarak adlandırılan giriş değeri ile deterministik algoritmalar kullanarak uzun sayı dizileri üretirken GRSÜ'ler fiziksel veya fiziksel olmayan gerçek olayları kullanarak rasgele sayı dizileri üretirler. Hibrit RSÜ'ler ise hem GRSÜ hem de SRSÜ sistemlerini birlikte kullanarak rasgele sayı üretirler [1]. SRSÜ'lerin ürettiği sayı dizileri deterministik özelliklerinden dolayı periyodik bir yapıya sahiptir. Ayrıca üretilen sayı dizileri ile

girişinde kullanılan tohum değeri arasında bağlantı kurulabildiğinden kriptografik uygulamalar için elverişli değildir. GRSÜ sistemleri rasgelelilik kaynağı olarak fiziksel olmayan bir modeli kullansalar bile yapılarında örnekleme ve son işlem algoritmaları olarak adlandırılan ve üretilen sayıların istatistiksel özelliklerini iyileştiren işlemler barındırdıkları için kriptografik uygulamalarda tercih edilmektedirler. Ancak GRSÜ'ler SRSÜ'lere kıyasla yavaş, maliyetli ve çevresel etkenlere hassas bağlıdır.

Son yıllarda, GRSÜ uygulamalarında kaos tabanlı sistemler sıkça kullanılmaktadır [3-5]. Kaos, doğrusal olmayan bir sistemde tahmin edilemeyen ve başlangıç şartlarına hassas bağlı davranış olarak tanımlanabilir [6]. Kriptografik uygulamalarda kullanılan anahtarların sahip olması gereken tahmin edilemezlik ve karmaşıklık özellikleri, kaotik davranış sergileyen sistemlerin kullanılmasıyla anahtar üretimi yapılabileceği düşüncesini ortaya çıkarmıştır [5]. Ayrıca kaotik sistemlerin parametre değerlerinin birbirinden bağımsız olması, kaos tabanlı rasgele sayı üretimini istatistiksel olarak elverişli hale getirmiştir.

Rasgele sayı üretiminde bir diğer önemli unsur ise gizli kalması gereken anahtarın nasıl bir ortamda üretilceğidir. Anahtar haberleşme sistemi dışında güvensiz ortamda üretilirse, sistemin güvenliği tartışılır hale gelir. Bu çıkarımdan yola çıkarak araştırmacılar sıklıkla FPGA (Field Programmable Gate Array, Sahada Programlanabilir Kapı Dizileri) gibi programlanabilir donanımsal ortamlarda RSÜ tasarımı yaparak anahtar üretimi gerçekleştirmişlerdir [4, 7].

GRSÜ tasarımlarının sayısal olarak gerçekleştirilmeleri için çoğu zaman ayrık zamanlı sistemler tercih edilmektedir. Bu nedenle kullanılan sürekli zamanlı sistemler ayrık zamanlı matematiksel modellere dönüştürülürler. Ayrıklaştırma işlemi olarak adlandırılan bu dönüşüm için FE (Forward Euler, İleri Euler) integrasyonu, RK (Runge-Kutta) integrasyonu ve Taylor serisi açılımı yöntemleri kullanılır. Bu yöntemler, gömülü sistemlerde tercih edilirken, gerekli donanım elemanı sayısı ve hesaplanan doğruluk oranı bakımından kıyaslanırlar. Tasarımcı dönüşüm uygulayacağı doğrusal olmayan sisteme göre ayrıklaştırma yöntemlerinden birini seçebilir [8].

1.1. Tezin Amacı

Son yıllarda kaotik devrelerden elde edilen gürültü benzeri sinyaller RSÜ uygulamalarında sıklıkla kullanılmaktadır. Kaotik sinyaller; kolay üretilip depolanabilmesi ve donanım tabanlı sistemlere uyumlu olması gibi avantajları sayesinde araştırmacıların ilgisini çekmektedir. Kaotik devreler parametre ve başlangıç değerlerinin uygun seçilmesi ile doğasında rasgelelilik özelliği barındıran sinyaller üretir. Kaotik devrelerin başlangıç şartlarına hassas bağlı olması nedeniyle [6] aynı kaotik devreden farklı istatistiki değerlere sahip çok sayıda sinyal üretilebilir [9]. Bu özelliği nedeniyle kaotik sinyaller entropi bakımından iyi istatistiki değerlere sahiptir. Fakat bu her kaotik devre kriptografik uygulamalarda kullanılır anlamına gelmez. Hem yeni oluşturulacak olan hem de literatürde var olan kaotik devrelerde üretilen kaotik sinyallerin rasgeleliliği analiz edilmelidir. Analiz sonucunda kaotik davranışın derecesi belirlenebilir.

Ayrıca herhangi bir RSÜ'den elde edilen rasgele sayı veya bit dizisinin belirli standartlarda olup olmadığını test etmek için literatürde FIPS140, AIS 31, NIST, Diehard, Crypt-X gibi çeşitli test paketleri geliştirilmiştir [10]. Tez çalışmasında istatistiksel test için bu paketlerden Ulusal Standartlar ve Teknoloji Enstitüsü tarafından geliştirilen NIST 800.22 test paketi kullanılmıştır.

Tez çalışmasında, literatürde var olan fakat RSÜ tasarımlarında kullanılmamış kaotik sistemler ve sayısallaştırma yöntemlerinden faydalanılarak üç RSÜ tasarımı gerçekleştirilmesi amaçlanmıştır. RSÜ tasarımlarında entropi kaynağı olarak kaotik sistemler kullanılmıştır. Kullanılan kaotik sistemler, MATLAB ortamında hem sürekli zamanlı hem de ayrık zamanlı olarak analiz edilmiştir.

1.2. Tezin İçeriği

Tezde gerçekleştirilen ilk RSÜ tasarımında, üç durum değişkenine sahip kaotik sistemin ayrık zamanlı olarak Xilinx sistem üretici (Xilinx System Generator, XSG) platformunda donanım tabanlı benzetimi gerçekleştirilmiştir. Durum değişkenleri belirli kriterlerden geçirilerek sayısallaştırılmıştır. Elde edilen bit dizileri Trivium [11] algoritmasına giriş olarak uygulanmış ve 50000 bit uzunluğunda rasgele bit dizisi üretilmiştir. İkinci RSÜ tasarımında, üç hücreli otonom HYSA modeli kullanılmıştır. Model, sürekli zamanlı olarak incelenip diferansiyel denklemler ayrık zamanlı denklemlere

dönüştürülmüştür. Bu model kullanılarak üç farklı senaryo için RSÜ tasarımı yapılmış ve FPGA ortamında gerçekleştirilmiştir. Gerçekleştirilen RSÜ tasarımında son işlem olarak Trivium algoritmasının ilk fazı kullanılmıştır. Üçüncü RSÜ tasarımında ise kaotik davranış sergileyen Memristörlü HYSA modeli kullanılmıştır. Modele ait diferansiyel denklemler ayrık zamanlı denklemlere dönüştürülmüştür. Uygulamada ayrık zamanlı Memristörlü HYSA modelinin yanı sıra lojistik harita da kullanılmıştır. Bu iki kaotik sinyal örneklenip sayısallaştırıldıktan sonra XOR son işlemi uygulanarak rasgele bit dizisi üretilmiştir.

Tez çalışmasında tüm tasarımlar Xilinx Kintex-7 KC705 FPGA Geliştirme Kartında gerçekleştirilmiştir. Tasarım sonucunda elde edilen bit dizilerinin rasgeleliliği NIST 800.22 test paketi ile test edilmiştir. Ayrıca son iki uygulama için üretilen bit dizilerinin periyodiklik derecesi ölçek indeksi (scale index) algoritması ile analiz edilmiştir.

1.3. Tezin Yapısı

Tez çalışması 5 bölümden oluşmaktadır. Bu giriş ile birlikte, Bölüm 2’de rasgelelilik, rasgele sayı üreteçleri ve NIST istatistiksel testleri hakkında genel bilgilere yer verilmiştir. Bölüm 3’te kaos, kaotik devreler ve ayırıklaştırma yöntemleri hakkında bilgiler verilmiştir. Bu bilgilerin yanı sıra, literatürdeki 5 kaotik sistemin davranışı analiz edilmiştir. Ayrık zamanlı denklemleri verilen kaotik sistemlerin XSG platformu kullanılarak FPGA kartında sayısal tasarımları, ayrıca kaotik devrelerden biri kullanılarak RSÜ tasarımı gerçekleştirilmiştir. Bölüm 4’te ise HYSA temelli kaotik devreler kullanılarak RSÜ uygulamaları ve bu uygulamaların FPGA kartında gerçekleştirimi detaylı olarak anlatılmıştır. Son bölümde ise gerçekleştirilen RSÜ tasarımlarının sonuçları karşılaştırılarak önerilere yer verilmiştir.

2. RASGELE SAYI ÜRETEÇLERİ

2.1. Rasgele Sayı Üreticinin Tanımı

En basit anlamda rasgelelilik, kesin olarak bilinmemek ve tahmin edilemezliktir [10]. Rasgelelilik günümüz teknolojilerinde oyun platformlarından güvenli haberleşmeye kadar çok geniş bir yelpazede sıkça kullanılmaktadır. Rasgele sayı üretici ise tahmin edilemeyen veya tekrar etmeyen sayı dizisi üreten bir sistemdir [12]. Rasgeleliliğe mühendislik alanında hava olaylarının tahmini, sistemlerin benzetim aşamaları, matematiksel modellerin başlangıç şartlarının oluşturulması ve haberleşme sistemlerinin güvenliği gibi gerek uygulamaların gerçeğe en yakın şekilde çalıştırılması, gerekse iletilmek istenen bilginin güvensiz bir ortamda dahi güvenilir bir şekilde alıcıya ulaştırılması amacıyla ihtiyaç duyulmaktadır.

2.2. Rasgele Sayı Üreticinden Beklenen Özellikler

Rasgele sayı üreticilerinin amacı, istatistiksel olarak bağımsız, düzgün dağılımlı ve tahmin edilemez sayı veya bit dizisi üretmektir. Bu temel amaç çerçevesinde üretilen rasgele sayılar, kriptografik uygulamalarda anahtar dağıtımı ve başlangıç değer atamaları gibi yüksek güvenlik gerektiren sistemlerde kullanılmaktadır. Bu nedenle, sistemin güvenilirliği, yapısında barındırdığı RSÜ'nün ürettiği sayıların rasgeleliği kadardır. Ayrıca gelişen teknoloji ile birlikte bilgisayar yazılımlarının kullanıldığı ortamların dışında donanım tabanlı sistemler de rasgele sayılara ihtiyaç duymaktadır. RSÜ'nün gömülü sistemlere entegrasyonu, daha güvenli donanımsal yapıların oluşturulmasını sağlamaktadır. RSÜ'nün kullanılacağı sistem ne kadar hassas ise RSÜ'den beklenen özellikler o kadar artar. RSÜ'nün temel özelliklerinin yanı sıra birtakım kriterleri sağlaması sistemin güvenliğini artırır.

- RSÜ'de üretilen çıkış değeri, bir önceki ya da bir sonraki çıkış değerinden türetilmemelidir.
- RSÜ, çevresel koşullardan performans ve kullanım ömrü bakımından olumsuz etkilenmemelidir.
- RSÜ, kullanılacağı uygulama gereksinimlerine uygun veri sayısını yüksek veri hız oranında sağlayabilmelidir.

2.3. İstatistiksel Testler

İstatistiksel test, bir stokastik modeli gerçek zamanlı bir süreç ile ilişkilendirir. İstatistiksel testlerle doğrulanmamış modelin sonuçları sadece bir varsayımdan ibarettir [13]. Bu yüzden, GRSÜ ya da SRSÜ'ler tarafından üretilen sayı ya da bit dizilerinin istatistiksel testler yardımıyla rasgele olup olmadıkları incelenir. Rasgelelik olasılıksal bir terim olduğundan, yalnızca rasgele diziler bir olasılık terimi ile tanımlanabilirler. Literatürde çok fazla sayıda istatistiksel test vardır. Bu nedenle herhangi bir istatistiksel testten başarılı bir şekilde geçen bir veri dizisi için kesin olarak rasgele demek mümkün değildir. Araştırmacılar çalışmalarında baz aldıkları istatistiksel testleri belirterek ilgili testlere göre rasgele bir veri dizisi elde ettiklerini özellikle belirtirler.

İstatistiksel analiz, deneysel bir çalışmanın betimleyici değişkenler (*explanatory variables*) ile tepkisel değişkenlerini (*response variables*) birbirinden ayırır. Tepkisel değişkenler deneysel bir çalışmanın çıktısını ölçmektedir. RSÜ için tepkisel değişken bir rasgele sinyal, sayısallaştırılmış sinyal ya da dâhili rasgele sayıdır. Betimleyici değişken ise deneysel çalışmanın hangi şartlar altında gerçekleştiğini tanımlar. Örneğin sıcaklık, fiziksel bir RSÜ'de üretelin rasgele sayıları etkileyebilir. Bu nedenle iyi istatistiksel bir test platformu, iki değişkeni analiz etmelidir [13].

Tez çalışmasında literatürde GRSÜ tasarımları için sıklıkla kullanılan Diehard, FIPS140, AIS 31, Crypt-X ve NIST gibi test paketlerinden NIST 800.22 sürümü kullanılmaktadır [10]. Bu test paketiyle istatistiki hesaplamalar yapılarak en sonda elde edilen olasılık değeri, eşik değer olarak belirlenen değer ile kıyaslanır. Her test için elde edilen olasılık değeri eşik değerinden büyük ise, veri dizisi o testten başarı ile geçmiş anlamına gelir. NIST 800.22 test paketinde 16 adet test yer almaktadır ve iki test dışında eşik değeri 0.01'dir. Eğer veri dizisi bu testlerin tümünden başarılı bir şekilde geçerse dizi istatistiksel olarak rasgeledir, aksi halde rasgele değildir. Aşağıda NIST 800.22 test paketindeki istatistiksel testlerin kısaca amaçlarından bahsedilmektedir. Testlerde kullanılan denklem ve hesaplamalar Avaroğlu [14] tarafından detaylı bir şekilde anlatılmıştır.

- **Frekans (*monobit*) testi:** Testin amacı, teste tabi tutulan ikili (*binary*) düzendeki veri dizisinde yer alan “0” ve “1” sayılarının yaklaşık olarak eşit

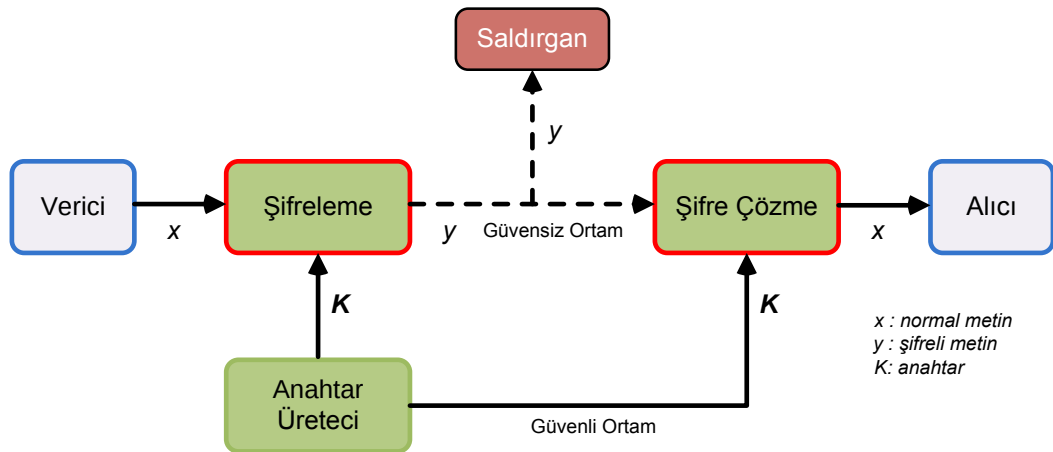
olup olmadığını hesaplamaktır. Test edilecek dizinin en az 100 bit uzunluğa sahip olması gerekir.

- **Bir blok içindeki frekans testi:** Dizide bulunan “0” ve “1” lerin oranını sabit uzunlukta seçilen bloklar için ayrı ayrı frekans testini çalıştırarak hesaplama yapar. Test edilecek dizinin en az 100 bit uzunluğa sahip olması gerekir.
- **Akış (Runs) testi:** Akış, dizi içerisindeki sürekli ve özdeş bit dizisi anlamına gelmektedir. Akış testinin amacı, dizi içerisindeki “0” ve “1” ler arasındaki değişimin hızlı veya yavaş olduğuna karar verecek hesaplamayı yapmaktır.
- **Bir blok içinde en uzun birlerin akış (Longest run of ones) testi:** Bu test, dizi içerisindeki en uzun “1”lerin akış değeri üzerine hesaplamaya dayalıdır. Dizi sabit sayıdaki bit kümesi halinde bloklara ayrılır. Akış testi, bu blokların her biri için tek tek uygulanarak, elde edilen sonuç eşik değeri ile karşılaştırılır.
- **İkili matris rankı testi:** Bu test, diziyi $M \times M$ -bitlik alt matrislere ayırır. Tüm alt matrislerin rankı hesaplanır. Böylece orijinal dizi ile alt diziler arasında bir ilişim olup olmadığı kontrol edilir.
- **Ayrık Fourier dönüşümü (Spektral) testi:** Bu test dizinin ayrık Fourier dönüşümünü hesaplayarak elde edilen genlik tepe değerlerini kontrol eder. Tepe değerlerine bakarak dizinin periyodik olup olmadığına karar verilir.
- **Örtüşmeyen şablon eşleştirme testi:** Bu testin amacı test öncesi belirlenen hedef alt dizisinin gözlenmesini kontrol ederek periyodik olmayan örnekleri tespit etmektir. Bu örnek sayısına göre dizinin periyodikliği test edilir.
- **Örtüşen şablon eşleştirme testi:** Örtüşmeyen şablon eşleştirme testi ile benzer mantıkta çalışmaktadır. İki test arasında hesapsal yöntem farklılığı vardır.
- **Maurer’in “Evrensel İstatistik” testi:** Testin amacı, sıkıştırılmış bir dizi uzunluğuna bağlı ölçüm yaparak birbiriyle örtüşen bit sayısını hesaplamaktır. Bu amaçla dizinin bilgi kaybı olmadan sıkıştırılıp sıkıştırılamayacağı kontrol edilir.
- **Lempel-Ziv sıkıştırma testi:** Dizide bulunan kümülatif ayrık örneklerin sayısını hesaplayarak dizinin ne kadar sıkıştırılabileceğini ölçer.
- **Doğrusal karmaşıklık (Linear complexity) testi:** Dizinin rasgele olarak kabul edilmesi için yeterince karmaşık olup olmadığını kontrol eder.

- **Seri (Serial) test:** Bu test, tüm dizideki m-bitlik örtüşen olası örneklerin frekansını hesaplar.
- **Yaklaşık entropi (Approximate entropy) testi:** Bu testin amacı, seri testinde olduğu gibi m-bitlik örtüşen olası örneklerin frekansını iki ardışık veya komşu uzunluktaki bitler için hesaplar.
- **Kümülatif toplamlar (Cumulative sums) testi:** Testin amacı, dizideki kısmi sıralı verilerin kümülatif toplamının, tanımlı kümülatif toplam değerleriyle kıyaslamaktır.
- **Rasgele gezinimler (Random excursions) testi:** Rasgele yürüyüş döngüsü, dizi içerisinde rasgele kabul edilen bir noktadan tam bir döngü olana kadar benzer bir noktaya kaç defa rastladığını hesaplar. Bu testin eşik değeri 500'dür.
- **Rasgele gezinimler değişken (Random excursions variant) testi:** Kümülatif toplam rasgele yürüyüşünde, rasgele gezinimler testine göre daha farklı kaç durumun oluştuğunu hesaplar.

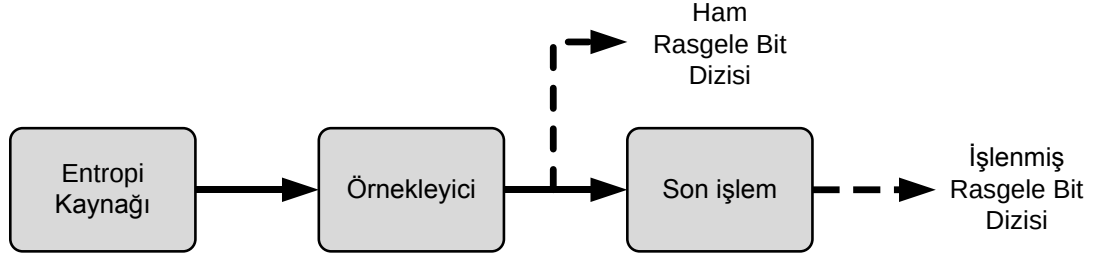
2.4. Rasgele Sayı Üreticinin Yapısı

RSÜ özellikle kriptografik uygulamalarda, bilgi güvenliğini sağlamak için alıcı ile verici arasındaki ortam güvenli olmasa dahi bilginin içeriğini saldırgan için anlamsız hale getirmek adına kullanılan anahtarın oluşumundaki temel işlem bloğu olarak düşünülebilir. Bilgi güvenliğini sağlamak için oluşturulan kriptografik uygulamaların amacı Şekil 2.1'deki gibi şematize edilebilir.



Şekil 2.1. Kriptografik uygulamaların temel amacı

Bir kriptografik sistemin güvenliği, ürettiği sayıların gerçek rasgeleliğine bağlıdır. Bu sayılar rasgele üretilen bitlerin birleşiminden oluşmaktadır. Basit olarak bir RSÜ'nün yapısı Şekil 2.2'deki gibi gösterilebilir.



Şekil 2.2. Basit bir RSÜ yapısı

2.4.1. Entropi Kaynağı

RSÜ sistemlerinde en önemli kısım entropi kaynağıdır. Bilgi teorisinde entropi, rasgele değişkenin belirsizliğinin ölçüsüdür. Bu belirsizlik ölçüsü, sistemin yapı taşı olan elemanların durum veya değer değişimlerinden ziyade sistemin kendisinin durum veya değer değişimiyle artar. Bilgi teorisinde entropi, Shannon entropi olarak kullanılmaktadır. Shannon, bir olayın belirsizlik içermesi durumunda o olay hakkında bilgi edinilebileceğini öngörmüştür. Bu yaklaşım ile oluşma olasılığı yüksek olan bir olayın meydana gelmesi durumunda elde edilecek bilginin az, tersi durumda ise bilginin daha fazla olacağı matematiksel olarak Denklem (2.1) ile ifade edilmektedir [15].

$$H(p) = -\sum_{i=1}^N p_i \log_2 p_i \quad (2.1)$$

Shannon entropi denkleminde yer alan p_i , N adet ihtimalin olduğu bir durum uzayında i . olayın meydana gelme olasılığı olarak tanımlanır. Denklem, elde edilecek belirsizliğin en az kaç bit ile tanımlanması gerektiğini hesaplamaktadır. Bu nedenle logaritmanın tabanı 2 olarak seçilir ve denklem sonucunda elde edilen bilgi miktarının birimi bit olarak verilir.

RSÜ yapılarında entropinin yüksek olması istenir. Bu nedenle entropi kaynağının belirsizlik ölçüsünün yüksek olması gerekir. Maksimum entropi değerine mümkün olan olayların meydana gelme olasılıklarının birbirine eşit olduğu durumda ulaşılabilir. Minimum entropi değeri ise mümkün olan olaylardan birinin meydana gelme olasılığının 1

olması durumunda elde edilir. Bu durumda belirsizlik yoktur ve entropi 0 olarak elde edilir.

RSÜ çıkışında elde edilecek sayıların temelini rasgele üretilen bit dizileri oluşturmaktadır. Entropisi yüksek olan bir sistemin entropi kaynağı olarak kullanılması, üretilen bit dizisinin her bitinin yüksek entropi değeri ile elde edilmesi anlamına gelir. Bir bitlik bir belirsizlik bilgisine sahip durum uzayında iki ihtimal vardır; bu da bit değerinin 0 veya 1 olmasıdır. RSÜ sistemlerindeki asıl amaç, bu iki ihtimalin meydana gelme olasılığını mümkün olduğu kadar birbirine yakın değerlerde elde etmektir. Yani, bir önceki bit değeri ne olursa olsun yeni üretilecek bitin değerinin belirsizliğinin yüksek olması RSÜ sistemini saldırılara karşı güçlü kılar.

2.4.2. Örnekleyici

Örnekleyici, düşük entropiye sahip kaynak kullanımında ihtiyaç duyulan ve elde edilen bit dizisindeki tekrarlı değerleri önlemek amacıyla kullanılan bir yapıdır. Genellikle entropi kaynağı olarak fiziksel sistemlerin kullanıldığı RSÜ yapılarında sayısallaştırıcı olarak da adlandırılır. Bu sistemlerde analog sinyaller ve işaretler kullanıldığından, üretilen rasgele değerlerin bir eşik değeri ile kıyaslanmasıyla bit dizisi üretilir ve art arda gelen 0 ve 1 değerlerinden kurtulmak için belirlenen örnekleme sayısı kadar bit çıkışa verilmez. Bu durumda çıkış bit uzunluğu azalsa da üretilen bit dizisinin rasgeleliliği artmış olur.

2.4.3. Son İşlem

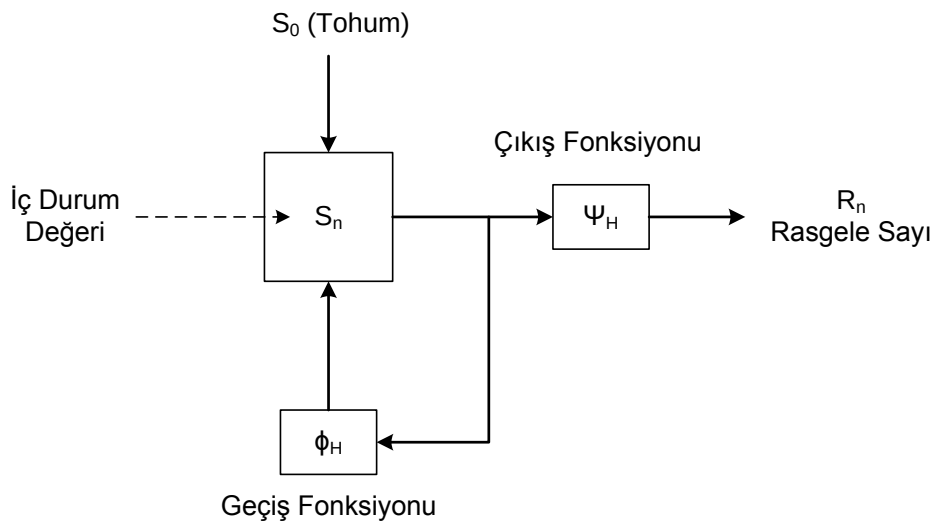
Son işlem, üretilen bit dizilerini düzenli (*uniform*) dağılımlı hale getirmek için kullanılan mantıksal ve/veya aritmetik işlemdir. Literatürde XOR doğrulama, Resilient fonksiyonu, Von Neumann doğrulama ve özet fonksiyonu gibi birçok son işlem algoritmaları kullanılmıştır [16-20]. Son işlem yapısı, üretilen bit dizilerinin istatistiksel özelliklerini güçlendirmesine karşın özellikle fiziksel sistemlerde eleman kullanımı, güç tüketimi ve kapsadığı alan bakımından artırıcı etki yaparak RSÜ sistemini daha karmaşık hale getirebilmektedir. Bu nedenle kullanılacak son işlem algoritması veya yapılarının basitliği, seçilen entropi kaynağı ve örnekleyici yapılarının kalitesi ile doğru orantılıdır.

2.5. Rasgele Sayı Üreteci Türleri

Rasgele sayı üreteçleri, Sözde Rasgele Sayı Üreteci (SRSÜ), Gerçek Rasgele Sayı Üreteci (GRSÜ) ve Hibrit Rasgele Sayı Üreteci olmak üzere üç temel sınıfa ayrılmaktadır. SRSÜ'ler tohum (seed) olarak adlandırılan belirli uzunluktaki bir başlangıç sayı dizisini bir algoritma içinde kullanarak uzun sayı dizileri oluştururlar. GRSÜ'ler ise elektronik devrelerdeki gürültüler, osilatör devrelerindeki faz farkı, anahtarlamalı devrelerdeki gecikme süresi ve kaotik devrelerin durum değişkenleri gibi fiziksel kaynakları kullanarak iyi istatistiki özelliklere sahip rasgele sayı dizisi oluştururlar. Hibrit RSÜ'ler, hem SRSÜ hem de GRSÜ yapılarını birlikte kullanarak GRSÜ yapılarının tahmin edilemezlik SRSÜ yapılarının ise karmaşık hesaplama yöntemlerinin sağladığı güvenilirlik özelliklerini birlikte gösterirler.

2.5.1. Sözde Rasgele Sayı Üreteci

SRSÜ'ler deterministik RSÜ olarak da adlandırılırlar. Çünkü SRSÜ'ler belirli bir algoritmaya ve bir tohum girişine bağlı olarak sayı dizisi çıkışı verirler. SRSÜ'ler donanım tabanlı GRSÜ'lere kıyasla çok daha uygun maliyetli ve çok hızlıdırlar. Fakat SRSÜ çıkışı girişteki tohum dizisinin bir fonksiyonu olduğundan, çıkışın anlık entropi değeri hiçbir zaman tohumun entropi değerini geçemez [21, 22]. Bu nedenle SRSÜ'nün rasgelelik düzeyi kullandığı tohumun rasgelelik düzeyine bağlıdır.



Şekil 2.3. SRSÜ tasarımına ait genel bir gösterim

Şekil 2.3’de SRSÜ tasarımına ait genel bir gösterim yer almaktadır. Sistemde $n-1$ adet rasgele sayı üretildikten sonra sistemin iç durum değeri S_n elde edilir. Bu değer elde edildikten sonra S_n çıkış fonksiyonundan geçirilerek bir sonraki rasgele sayı olan R_n elde edilir. Aynı zamanda S_n , geçiş fonksiyonundan geçirilerek de S_{n+1} değeri üretilir ve yeni iç durum değeri olarak kullanılır. Bu döngü devam ettirilerek rasgele sayı dizisi elde edilmektedir. İç durum değerinin ilk değeri olan S_1 ise girişte kullanılan tohum değerinin geçiş fonksiyonundan geçirilmesiyle üretilir [14]. Özet olarak SRSÜ’nin ürettiği bütün durum değerleri ve rasgele sayılar tohum değerine bağlıdır. Tohum değerinin bilinmesi durumunda sistemin tümü tehdit altındadır. Bu nedenle tohum değeri genellikle bir GRSÜ yardımıyla üretilir. Böylece SRSÜ sisteminin güvenliği artırılmış olur. Ayrıca iç durum değerinin de ataklara karşı korunması gerekmektedir. Sistemin güvenliği kullanılan geçiş ve çıkış fonksiyonlarının daha karmaşık algoritmalara sahip olması ile artırılabilir.

Literatürde SRSÜ olarak yer alan algoritmaların başında AES (*Advance Encryption Standard*) [23], DES (*Data Encryption Standard*) [24] ve RSA (Rivest-Shamir-Adleman) [22] gelmektedir. Bu algoritmalar uluslararası şifreleme yarışmalarında kabul görerek standart haline gelmişlerdir. Algoritmaların içyapısında bit kaydırma (*shifting*), yer değiştirme (*substitution*), XOR ve karıştırma (*permutation*) gibi algoritmik işlemler bir ve birden fazla döngü içinde yapılarak üretilen sayıların rasgeleliği artırılmaktadır.

2.5.2. Gerçek Rasgele Sayı Üreteci

GRSÜ sistemleri, tekrar aynı değerde üretilemeyecek biçimde rasgele çıkış üretirler. Bu tanım bir örnekle açıklanacak olursa, hilesiz bir madeni paranın atılması durumunda iki ihtimal ile karşılaşılır. Bu işlem 100 defa ardı ardına tekrarlanarak 100 bit uzunluğunda rasgele bit dizisi elde edilebilir ve bu rasgele bit dizisinin aynısının üretilme ihtimali $1/2^{100}$ dür [25]. Örnekte yer alan olay gerçek bir olaydır ve GRSÜ’ler benzer mantıkla hem fiziksel hem de fiziksel olmayan sistemleri kullanarak entropi değeri yüksek rasgele sayı veya bit dizileri üretirler. Tez çalışmasında GRSÜ tasarımına odaklanıldığından bu bölümde literatürdeki bazı GRSÜ tasarımlarına yer verilmektedir.

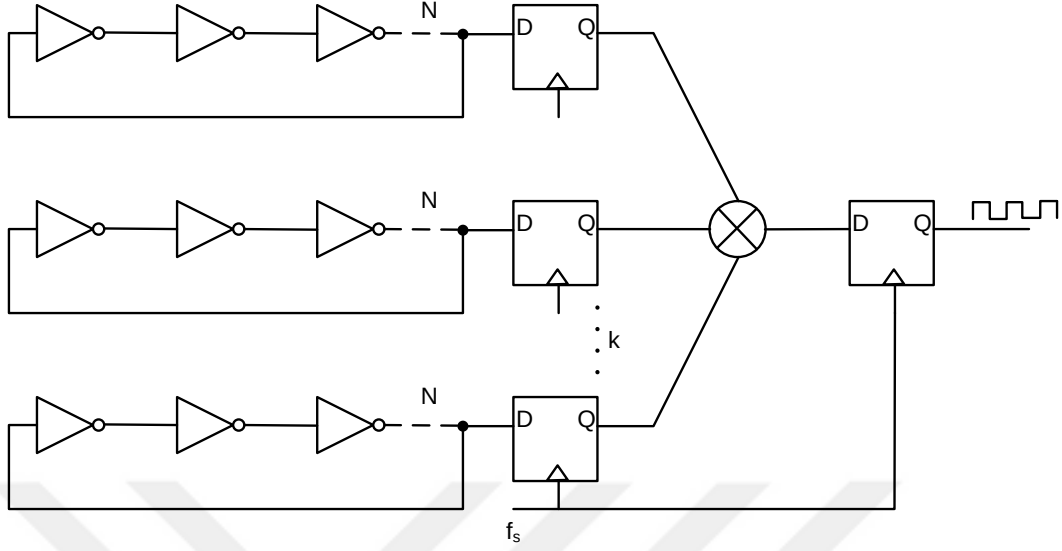
Yalçın ve arkadaşlarının 2004 yılında yayınladıkları gerçek rasgele bit üretimi tasarımında [26], Chua devresine benzer basit bir devre modelinin çıkışında gözlenen çift sarmal çekici (*double scroll attractor*) olarak adlandırılan kaotik davranışı entropi kaynağı

olarak kullanmışlardır. Kaotik davranışın doğasında var olan biaslı bitleri ve bitler arasındaki korelasyonu yok etmek için Von Neumann doğrulama algoritmasına benzer bir son işlem yapısını uygulamışlardır. Çalışma sonucunda tamamen donanımsal entegre yapısında olan ve gerçek zamanlı olarak çalışan tasarım ile rasgele bit üretimi gerçekleştirilmiştir.

2006 yılında yapılan bir çalışmada [3], entropi kaynağı olarak kaotik davranış sergileyen bir analog devre kullanılmıştır. Markov kaotik haritaya dayalı analog devre, PSoC (Programmable System on Chip, Çip Üzerinde Programlanabilir Sistem) tekrar konfigüre edilebilir aygıtında harici bir elemana ihtiyaç duymadan rasgele bit dizisi üretmektedir. İdeal RSÜ özelliklerini elde edebilmek için XOR doğrulama yöntemi son işlem olarak kullanılmıştır. Üretilen rasgele bit dizilerinin istatistiki analizleri yapılarak devre parametreleri optimize edilmiştir. Aynı araştırmacılar 2007 yılında ise aynı analog devrede kullanılan anahtarlama kapasitör (switched capacitor) sayılarını artırarak daha güçlü bir GRSÜ tasarımı önermişlerdir [27].

Danger ve arkadaşları 2009 yılında FPGA'daki yarı kararlı açık çevrim yapıları kullanarak yüksek çıkış bit oranına sahip bir GRSÜ tasarımını öne sürmüşlerdir [28]. Bu çalışmada D tipi flip-flop ve LUT (Look up Table, Doğruluk Tablosu) yapıları kullanılarak Latch zincir tasarımı elde edilmiştir. Tasarımda örnekleyici olarak da yine D tipi flip-flop kullanılmıştır. Benzer bir diğer çalışmada ise [29] ikili yarı kararlı FPGA eleman yapısı tercih edilerek GRSÜ tasarımı önerilmiştir.

Murphy 2012 yılında [30], gerilim kontrollü programlanabilir GRSÜ tasarımını önermiştir. Bu tasarımda entropi kaynağı, Şekil 2.4'de verilen ısı gürültü sinyali üreten evirici halka osilatörlü XOR (Inverter Ring Oscillator Exclusive-OR, IRO Exclusive-OR) yapısıdır. Çalışma sonucunda oda sıcaklığına yakın bir sıcaklıkta 0.1 V/91 MHz frekans ayarı ile gerilim kontrolüne dayalı rasgele bit üretimi gerçekleştirilmiştir.

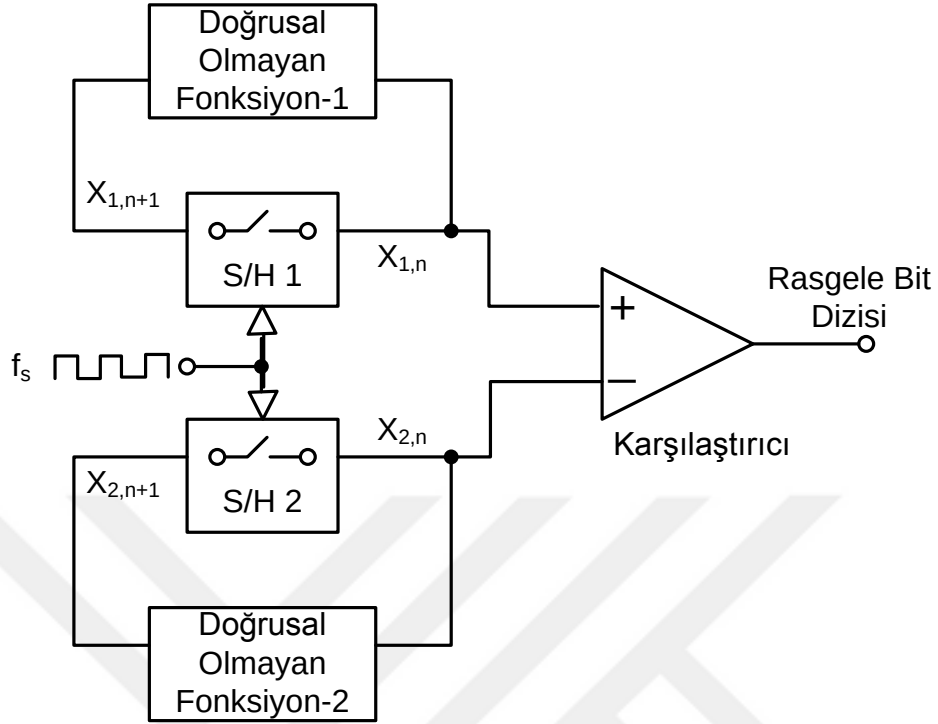


Şekil 2.4. IRO XOR tabanlı GRSÜ mimarisi

2014 yılında ise Sprot94 G kaotik sistemi [31] ayrık zamanlı modellenerek FPGA ortamında VHDL (*VHSIC Hardware Description Language*) programlama dili ile kodlanmıştır. Ayrık zamanlı modeldeki aritmetik için Xilinx firmasının geliştirdiği IP Core Generator blokları kullanılmıştır. Çalışmada 32-bit IEEE 754 kayan noktalı sayı standardı ile aritmetik işlemler gerçekleştirilerek son işlemci olarak seçilen XOR doğrulama çıkışından elde edilen rasgele bit dizileri benzetim sonuçlarına göre istatistiki testlerden başarılı bir şekilde geçmiştir.

Tuncer ve arkadaşları [7] halka osilatörlü GRSÜ tasarımlarındaki klasik örnekleme yönteminden farklı olarak örnekleme işlemini periyodik olmayan bir yapı ile gerçekleştirmişlerdir. Çalışmada kullanılan halka osilatörü sayısı değiştirilerek elde edilen üç farklı senaryo için FPGA ortamında çalıştırılıp rasgele bit dizisi elde edilmiştir.

2017 yılında Çiçek ve arkadaşları [32], ikili entropi çekirdekli ayrık zamanlı kaos tabanlı GRSÜ (*dual entropy core DTC-based TRNG*) tasarımını ön ürün olarak 180-nm CMOS teknolojisi ile imal etmişlerdir. Bu tasarım, mimarisi Şekil 2.5’de de görüldüğü gibi ikisi de doğrusal olmayan fonksiyon olan Bernoulli ve Tent kaotik haritaların saat darbe işareti (clock pulse) ile örneklenip birbirleriyle karşılaştırılması sonucunda rasgele bit üretimini sağlamaktadır. Gerçek zamanlı sonuçlar analiz edilerek GRSÜ tasarımı ile 35 Mbps oranında çıkış bitleri elde edilmiştir. Bu tasarımın en avantajlı yanı ise son işlem yahut ek girdi gibi herhangi bir harici yapıya ihtiyaç duymamasıdır.



Şekil 2.5. İkili entropi çekirdekli ayrık zamanlı kaos tabanlı GRSÜ mimarisi

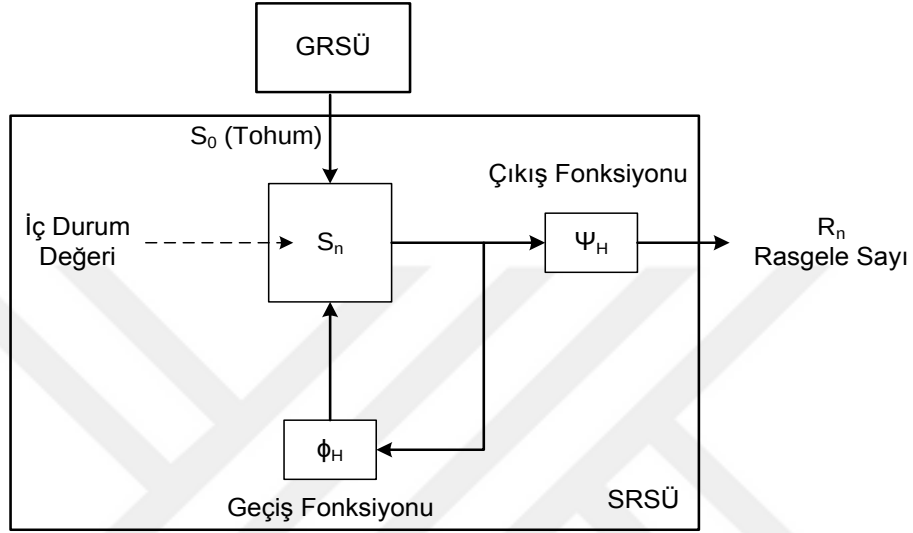
Kriptografik uygulamalarda hem çok güvenilir hem de çok yüksek hızda rasgele bit üretiminin zor olduğu algısını çürütmek için bir GRSÜ tasarımı öneren Alçın ve arkadaşları [33], yapay sinir ağı (*Artificial Neural Network*, ANN) tabanlı kaotik sistem kullanarak 115 Mbps oranında rasgele bit üretimi gerçekleştirmişlerdir.

Literatürde yukarıda kısaca bahsedilen çalışmalara benzer çok çalışma olsa da son olarak Bonny ve arkadaşlarının çok yakın bir zaman önce önerdiği GRSÜ tasarımı [4] dikkat çekmektedir. Bu çalışmada basit doğrusal olmayan özelliklerde, biri üç diğeri ise dört adet durum değişkenine sahip iki kaotik sistemin çıkışları, belirli uzunlukta iki kayıtçı (*register*) oluşturarak XOR son işlemine tabi tutulmuştur. Böylece döngü yardımıyla 879.5 Mbps çıkış oranına sahip rasgele bit üreten GRSÜ tasarımını FPGA ortamında gerçekleştirmişlerdir.

Güncel GRSÜ tasarımları içeren çalışmalarda araştırmacıların, tasarımlarını gerçek zamanlı olarak çalıştırmak için paralel çalışma kabiliyeti yüksek olan FPGA platformlarını, son işlem yapısı olarak basitliği ile öne çıkan XOR doğrulama yöntemini ve daha az eleman kullanımı sağlayan entropi kaynağı olarak ise ayrık zamanlı kaotik sistemleri daha çok tercih ettikleri görülmektedir.

2.5.3. Hibrit Rasgele Sayı Üreteci

Hibrit RSÜ tasarımları içyapılarında hem SRSÜ hem de GRSÜ sistemlerini barındırırlar. Şekil 2.6’da Hibrit RSÜ yapısına ait genel bir gösterim yer almaktadır.



Şekil 2.6. Hibrit RSÜ'nün mimarisi

Önceki bölümlerde anlatıldığı gibi SRSÜ tasarımları ne kadar karmaşık algoritmalara dayalı rasgele sayı üretebilseler de, üretilen sayının tohum ve iç durum değerine bağlılığı SRSÜ tasarımlarının dezavantajıdır. GRSÜ tasarımlarının tekrar elde edilemeyecek derecede rasgele sayı üretebilmelerinden dolayı, Hibrit RSÜ tasarımlarının çoğunda SRSÜ'nün ihtiyaç duyduğu tohum girişi bir GRSÜ tasarımı ile elde edilmektedir. Şekilden de görüldüğü gibi Hibrit RSÜ tasarımları iki RSÜ yapısını birlikte kullandığından, bu tasarım ile GRSÜ'lere kıyasla daha hızlı ve SRSÜ'lere kıyasla daha iyi istatistiksel özelliklere sahip rasgele sayı üretimi gerçekleştirilebilir.

3. KAOTİK DEVRELER VE SAYISAL TASARIMLARI

Kaos teorisi, başlangıç değerlerine hassas duyarlılık gösteren dinamik sistemlerin davranışını inceler. Başlangıç değerlerindeki küçük değişimlere, nümerik hesaplama yöntemlerindeki sayı yuvarlama hatası örnek verilebilir. Bu değişimler kaotik sistemlerde beklenenden daha büyük değerlikli değişimlerin gözlenmesine neden olmaktadır [34, 35]. Kaos terimi ilk olarak Edward Lorenz tarafından 1963 yılında yapılan bir çalışma [36] ile tanımlanmıştır. 1963'ten günümüze kadar, kaotik sistemler birçok araştırmacının ilgisini çekmiş, tanımlanan kaotik sistem modelleri senkronizasyon ve güvenli haberleşme gibi önemli alanlarda sıkça kullanılmıştır.

Araştırmacılar son zamanlarda kaotik devre gerçeklemelerini donanım ortamlarında yaparak hem eleman sayısını minimum seviyelere indirgemiş hem de yüksek çalışma frekansları ile hızlı neticeler elde etmişlerdir. Bu donanımlardan en çok tercih edileni ise FPGA çiplerinin yer aldığı geliştirme kartlarıdır.

Tez çalışmasında kaotik devre temelli RSÜ tasarımları FPGA kitinde gerçekleştirilmiştir. Bu bölümde, RSÜ tasarımlarında sıkça kullanılan lojistik haritanın yanı sıra Lorenz sistemi [36], Chua osilatör devresi [37], 3D kaotik sistem [38] ve memristörlü kaotik devre [39] modelleri için kaotik davranış analizlerine yer verilmektedir. Kaotik sistemlerin denklemleri, ayırıklaştırma yöntemleri ve sayısal tasarım aşamaları anlatılarak XSG platformundaki donanım tabanlı benzetimleri ile birlikte verilmektedir. Ayrıca 3D kaotik sistemi kullanılarak RSÜ tasarımı da gerçekleştirilmiştir.

3.1. Kaotik Devrelerin Özellikleri

Kaos terimi, güncel yaklaşımla 1976'da Yorke ve Li tarafından tanımlanmıştır [40]. Tam olarak anlaşılamayan bir terim olan kaos, çeşitli fiziksel sistemlerde meydana gelen ve bazen istenilmeyen bir durum olarak yorumlanmaktadır. Poincare'ye göre kaos, dinamik sistemlerde kararlı ve kararsız dallanmaların (manifold) kesişimidir [41]. Bu çıkarımla, kaotik davranışın sistemin kararlı olduğu bir durumda meydana gelmediği açıktır. Özellikle kontrol sistemlerinde kaotik davranış bu yönüyle istenmeyen bir durumdur.

Robert L. Devaney bir dinamik sistemi kaotik olarak sınıflandırabilmek için şu özelliklere sahip olması gerektiğini savunmaktadır [42]:

- Sistem, başlangıç koşullarına hassas bağlı olmalıdır.
- Sistem topolojik olarak karmaşık yapıda olmalı ve verdiği tepkinin tahmin edilemez olması gerekmektedir.
- Sistem çok sayıda periyodik yörüngeye sahip olmalıdır.

Devaney her ne kadar üç özellik sıralamış olsa da çoğu araştırmacı son iki durumun da başlangıç şartlarına hassas bağlılık özelliğini işaret ettiğini savunmaktadırlar. Bu nedenle başlangıç şartlarına hassas bağlılık kaotik davranışın varlığının en büyük göstergesidir.

Başlangıç şartlarına hassas bağlılık, genellikle “kelebek etkisi” olarak bilinir [43]. Lorenz’in bu yaklaşımında kelebeklerin kanat çırpmaları, bir fırtına oluşumunun başlangıcındaki küçük bir durum değişimi olarak kabul edilmektedir.

3.2. Diferansiyel Denklemlerin Ayırıştırılması

Kaotik davranış, başlangıç değerlerine hassas bağlı doğrusal olmayan diferansiyel denklemlerle tanımlanan dinamik sistemlerde gözlenmektedir. Bu sistemlerin yazılımsal olarak gerçekleştirilmesi için araştırmacılar çoğu zaman sürekli zamanlı (*Continuous Time*, CT) denklemler yerine ayrık zamanlı (*Discrete Time*, DT) denklemleri tercih etmektedirler. Sürekli zamanlı denklemlerden ayrık zamanlı denklemlere dönüşüm için en sık kullanılan yöntemler; Taylor serisi açılımı, RK (Runge-Kutta) ve FE (*Forward Euler*, İleri Euler) ayırıştırma yöntemleridir.

3.2.1. Taylor Serisi Açılımı

Diferansiyel denklemler, fonksiyonların bir veya birden çok değişkene göre türevleri ile ilişkili denklemlerdir. Kaotik sistemlerin çoğu, birinci mertebeden adi diferansiyel denklem takımlarından oluşur. Taylor serisi açılımı, 1715 yılında İngiliz matematikçi Brook Taylor tarafından resmi olarak tanımlanmıştır. Herhangi bir birinci mertebeden diferansiyel denklem takımı Taylor serisi açılımına göre Denklem (3.1)’de verilen matematiksel ilişki kullanılarak ayrık zamanlı denklemlere dönüştürülebilir.

$$x(t+h) = x(t) + \sum_{m=1}^{\infty} \frac{1}{m!} \cdot h^m \cdot x^{(m)}(t) \quad (3.1)$$

Denklemden $x(t)$ zamana bağılı durum değişkenini, $x^{(m)}(t)$ durum değişkeninin m . dereceden türevini, m Taylor serisi açılım derecesini ve h adım aralığını ifade etmektedir. Seri açılımında m değeri ne kadar büyük seçilirse, diferansiyel denklemler o kadar yüksek doğrulukla ayrık zamanlı denklemlere dönüştürülür. Adım aralığı değeri ile doğruluk arasındaki ilişki ise ters orantılıdır.

3.2.2. Runge-Kutta Yöntemi

1900'lü yıllarda C. Runge ve M.W. Kutta adlı matematikçiler tarafından geliştirilen Runge Kutta yöntemlerinin en sık kullanılanı "RK4" olarak kısaltılan 4. dereceden klasik Runge-Kutta yöntemidir. Yöntemin kolay anlaşılması adına 2. dereceden Runge-Kutta denklemleri Denklem (3.2)'deki gibi ifade edilmiştir.

$$\begin{aligned}
 \dot{x}(t) &= \frac{dx(t)}{dt} = f(x(t), t) \\
 k_1 &= h \cdot f(x(t_0), t_0) \\
 x_1(t_0 + \frac{h}{2}) &= x(t_0) + k_1 \cdot \frac{h}{2} \\
 k_2 &= f\left(x_1(t_0 + \frac{h}{2}), t_0 + \frac{h}{2}\right) \\
 x(t_0 + h) &= x(t_0) + k_2 \cdot h
 \end{aligned} \tag{3.2}$$

Denklemden $x(t)$ zamana bağılı durum değişkenini, $x(t_0)$ durum değişkeninin t_0 anındaki değerini, k_1 ve k_2 sırasıyla 1. ve 2. dereceden eğimleri ve h adım aralığını ifade etmektedir. Yöntemin derecesi ne kadar büyük seçilirse, diferansiyel denklemler o kadar yüksek doğrulukla ayrık zamanlı denklemlere dönüştürülür. Adım aralığı değeri ile doğruluk ise ters orantılıdır.

3.2.3. Euler Yöntemi

Euler yöntemi 2. dereceden Runge-Kutta yöntemi ile eşdeğerdir. Bu yöntemde ileri veya geri yönlü eğim hesabı yapılır. İleri Euler yönteminde kullanılan tanımlama Denklem (3.3)'deki gibidir.

$$\dot{x}(t) = \frac{x(t+h) - x(t)}{h} \tag{3.3}$$

Denklemde $x(t)$ zamana bağılı durum değişkenini, $\dot{x}(t)$ durum değişkeninin birinci mertebeden türevini ve h ise adım aralığını ifade etmektedir. Bu yöntemde h değeri ne kadar küçük seçilirse, diferansiyel denklemler o kadar yüksek doğrulukla ayrık zamanlı denklemlere dönüştürülebilir.

Ayrık zaman dönüşümlerinde denklemleri verilen üç yöntem de kullanılabilir. Yapılan bir çalışmada Lorenz kaotik sistemi üç yöntem ile ayrı ayrı gerçekleştirilerek sistemin faz portreleri incelenmiştir [8]. Çalışma sonucunda iteratif denklem basitliği dolayısıyla FPGA çipinde daha az donanım kullanımı sağlayan İleri Euler yöntemi ile diğer iki yönteme göre yüksek doğruluk elde edilmiştir. Bu yüzden tez çalışmasında ayırıklaştırma için İleri Euler Yöntemi kullanılmıştır.

3.3. Literatürdeki Bazı Kaotik Devreler ve Analizleri

Bir dinamik sistemde kaosu gözlenebilirliği, sistemin başlangıç değerlerine hassas bağılılığı ile ilişkilidir. Ayrıca sistemdeki parametre değişimleri de sistemin davranışını etkilediğinden seçilen parametre değerine bağılı olarak kaotik davranışın varlığı söz konusu olacaktır. Bu tanımlardan yola çıkarak kaotik devrelerin analizinde kaosu varlığını göstermek için bazı hesaplama teknikleri geliştirilmiştir. Bunlar, çatallaşma diyagramı (bifurcation diagram) ve Lyapunov üstel değerlerinin (Lyapunov exponents) analizidir.

Çatallaşma diyagramı, bir sistemin dinamik davranışının kontrol parametresi olarak seçilen bir parametre değerine göre nasıl değiştiğini gösterir. Bu analizde kontrol parametresi küçük adım aralıkları ile değişirken sistemin kararlı durumdaki değişimi gözlenir. Çatallaşmanın oluşması sistemin her çatallaşma anından sonra farklı periyotlarda davranması anlamına gelmektedir. Çatallaşma sayısı arttıkça sistem artık periyodiklik özelliği göstermekten uzaklaşır ve kaotik davranış sergilemeye başlar [44].

Lyapunov üstel değerleri ise sistemin başlangıç değerlerine hassas bağılılığının ölçümü sonucu elde edilir. Üstel değeri, başlangıç değerlerinde yapılan çok küçük değişim ile sistemin belirli yörüngelerden ne kadar uzaklaştığının hesaplanmasıyla elde edilir. Sistemden elde edilebilecek Lyapunov üstel değeri sayısı, sistemin sahip olduğu durum değişkeni sayısı kadardır [44].

Bu teknikler kullanılarak lojistik harita, Lorenz sistemi, Chua osilatör devresi, 3D kaotik sistem ve memristörlü kaotik devre analiz edilmektedir.

3.3.1. Lojistik Harita

Ayrık zamanlı dinamik bir sistem olan lojistik harita;

$$x_{i+1} = rx_i(1 - x_i) \quad (3.4)$$

denklemleri ile tanımlanmıştır. Bu denklemde x_i durum değişkenini ve r ise sistem parametresini temsil eder. Sistemin tek şartı $0 \leq x_i \leq 1$ eşitsizliğinin sağlanmasıdır. Sistem, bir durum değişkenine sahip olmasına rağmen $r \in [3,4]$ aralığı için kaotik davranış göstermektedir. Bu davranışın varlığı denge noktasının (*equilibrium point*) tespitinden başlayarak analiz edilebilir.

Dinamik sistemlerde denge noktası sabit nokta olarak da adlandırılır ve

$$x_{i+1} = x_i = \bar{x} \quad (3.5)$$

denklemleri ile ifade edilebilir. Bu durumda Denklem (3.4) tekrar yazılırsa;

$$\bar{x} = r\bar{x}(1 - \bar{x}) \quad (3.6)$$

denklemlerden

$$\bar{x}_1 = \frac{r-1}{r} \quad (3.7)$$

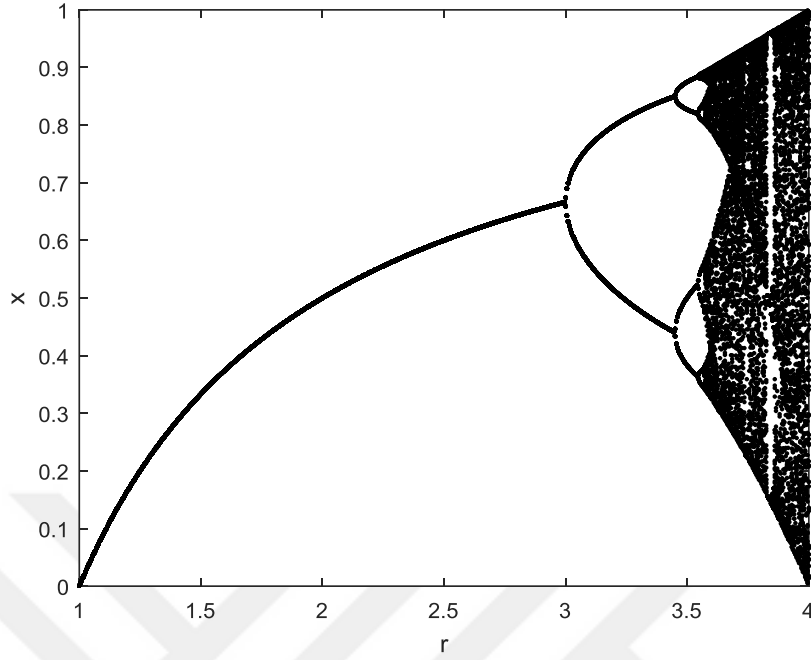
ve

$$\bar{x}_2 = 0 \quad (3.8)$$

denge noktaları elde edilir. Denge noktalarının lokal kararlılıkları ise Denklem (3.4)'ün Jacobian matrisinde yerine yazılırsa;

$$\frac{dx_{i+1}}{dx_i} = (r - 2rx_i) \Big|_{x_i = \bar{x}} \quad (3.9)$$

$|r| < 1$ ve $|2 - r| < 1$ şartları için kararlı olduğu hesaplanır. İki şart birleştirilirse $1 < r < 3$ şartı elde edilir. Sonuç olarak, sistem $r > 3$ için kararlı değildir. Sistemin kararlı olmadığı aralıkta denge noktası sayısını çatallaşma diyagramına bakarak gözlemleyebiliriz.



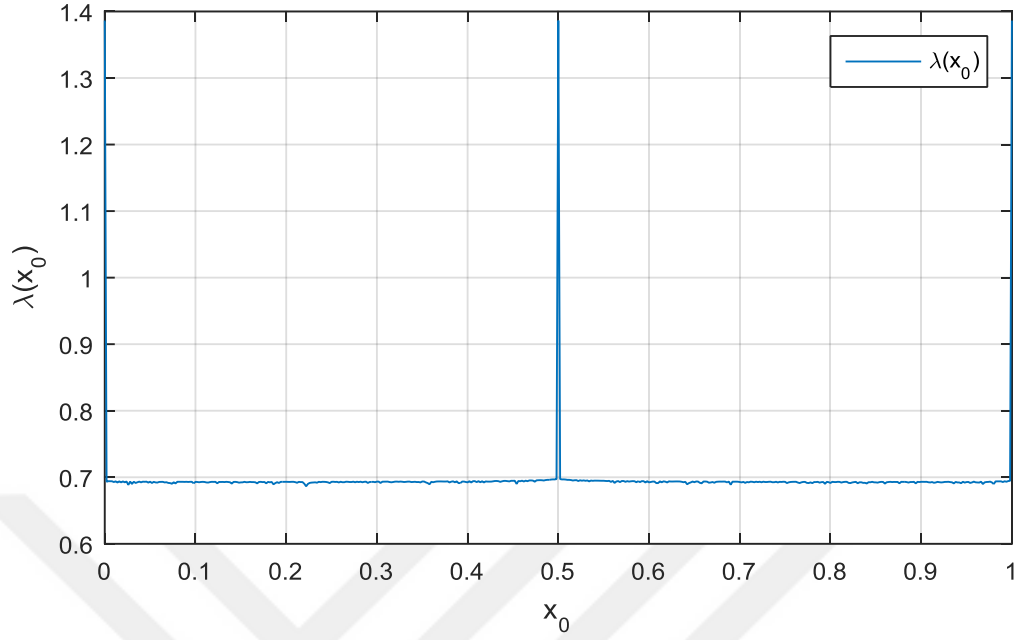
Şekil 3.1. Lojistik harita sisteminin $r \in [1,4]$ aralığı için çatallaşma diyagramı

Şekil 3.1’de görüldüğü gibi, lojistik harita sistemi r parametresine bağlı olarak $r \in [1,3]$ aralığında kararlı, $r \in [3,4]$ aralığında ise kararsız davranarak periyot çevrim sayısı sırasıyla 2 ve 4 olarak artmaktadır. Böylece sistemin kaotik davranışının, parametre değerine bağlı olduğu söylenebilir.

Bir dinamik sistemin Lyapunov üstelleri sayısı, durum değişkenleri sayısı kadardır. Lojistik harita sisteminde bir Lyapunov üstel değeri vardır. Bu değeri hesaplamak için Denklem (3.10) kullanılır.

$$\lambda(x_0) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n \ln |\dot{f}(x_{k-1})| \quad (3.10)$$

Denklemden $\lambda(x_0)$, x_0 başlangıç değerindeki Lyapunov üstelini, n iterasyon sayısını ve $\dot{f}(x_{k-1})$ ise sistemin durum değişkenine göre x_{k-1} değerindeki türevini temsil etmektedir.



Şekil 3.2. Lojistik harita sisteminin $r = 4$ için Lyapunov üstel değişimi

Şekil 3.2’den de görüldüğü gibi lojistik harita sabit parametre değeri için başlangıç değerine hassas bağlıdır. Lyapunov üstel değerleri $x_0 = (0, 0.5, 1)$ için 1.386, diğer tüm başlangıç değerleri için 0.639 olarak elde edilmiştir. Sisteme verilen başlangıç değerleri için pozitif değerlikli Lyapunov üsteli hesaplanıyorsa, sistem kaotik davranış göstermektedir. Yapılan analizlerle lojistik harita sisteminin kaotik davranışı incelenmiştir.

3.3.2. Lorenz Sistemi

Dinamik sistemlerde doğrusal olmayan terimler içeren durum değişkeni sayısı arttıkça daha karmaşık davranışlar gözlenmektedir [45]. Ayrık zamanlı sistemlerde gözlenen kaosu sürekli zamanlı sistemlerde de aşağıdaki özellikleri sağlayarak elde edilmesi mümkündür.

- Durum değişkenlerinin periyodik olmayan osilasyonlar yapması
- Başlangıç değerlerine hassas bağıllık
- Parametre değişimlerine duyarlılık
- Uzun zaman aralığında tahmin edilemezlik

Üç veya daha fazla durum değişkenine sahip otonom sürekli zamanlı sistemlerde gözlenen periyodik olmayan davranışa tuhaf çekici (*strange attractor*) denir [44]. Bu

terim, sistem yörüngesinin sınırlı fakat aperiodyik veya ıraksak olması anlamında kullanılır. Bu sistemlere örnek olarak Lorenz sistemi verilebilir.

Lorenz, doğrusal olmayan sistemi [36] Denklem (3.11)'de verilen denklem takımıyla tanımlanmaktadır.

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= xy - bz\end{aligned}\tag{3.11}$$

Sistemde σ , ρ ve b parametreleri pozitif değerlidir ve $\sigma = 10$, $\rho = 28$ ve $b = 8/3$ değerlerinde kaotik davranış görülmektedir. Lorenz denklem takımı, ayrık zamanlı denklemlere dönüştürülerek sistem, istenilen sayısal işlem platformunda gerçekleştirilebilir. Ayrıklaştırma yöntemi olarak İleri Euler kullanılıp dönüşüm sağlanırsa;

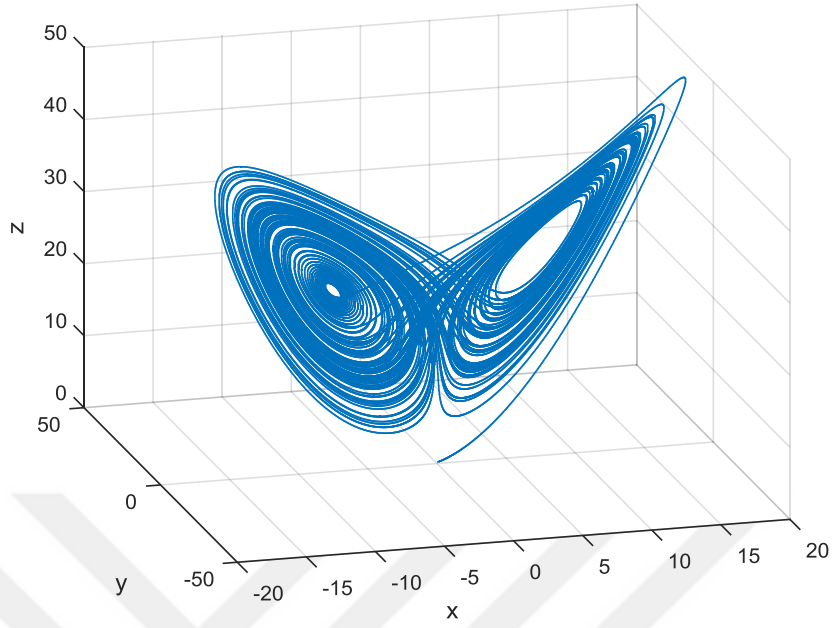
$$\begin{aligned}x[n+1] &= x[n] + h\sigma(y[n] - x[n]) \\ y[n+1] &= y[n] + h(\rho x[n] - y[n] - x[n]z[n]) \\ z[n+1] &= z[n] + h(x[n]y[n] - bz[n])\end{aligned}\tag{3.12}$$

denklem takımının, $\sigma = 10$, $\rho = 28$, $b = 8/3$ parametre değerleri ve $(x_0, y_0, z_0) = [0.1, 0.1, 0.2]$ başlangıç değerleri için MATLAB ortamında $n = 100000$ iterasyon sayısı ve $h = 0.001$ adım aralığında davranışı Şekil 3.3'deki gibi elde edilir.

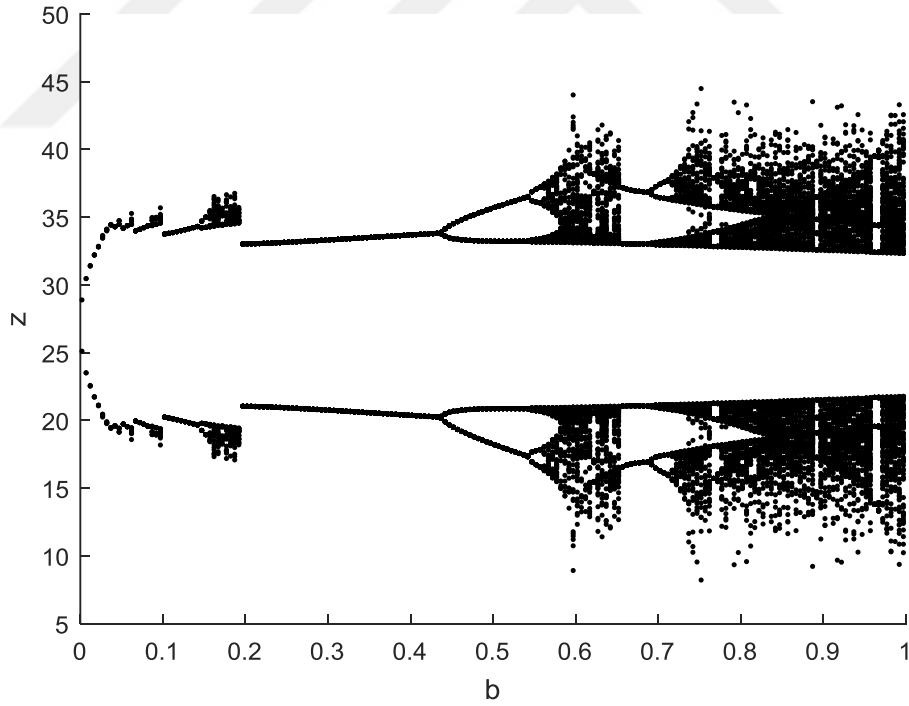
Sistemin bu davranışında, yörüngesinin başlangıç değerine hiçbir zaman geri dönmediği ve aynı noktadan hiç geçmediği gözlenmektedir. Sistem, verilen parametre ve başlangıç değerlerinde kaotik özellik göstermekte ve bu davranış tuhaf çekici olarak adlandırılmaktadır.

Sistemin parametre değişimine karşın nasıl değiştiği çatallaşma diyagramı ile gözlemlenir. Sistemin b parametresinin değişimine karşın davranışı Şekil 3.4'de gösterilmiştir.

Çatallaşma diyagramında b parametresinin sıfırdan büyük tüm değerlerinde periyot çiftlemenin olduğu gözlenmektedir. Bu durum için sistem b parametresine duyarlıdır. Tez çalışmasında çatallaşma diyagramlarının elde edilmesi için Frasca ve arkadaşlarının oluşturduğu MATLAB kodları kullanılmıştır [44].

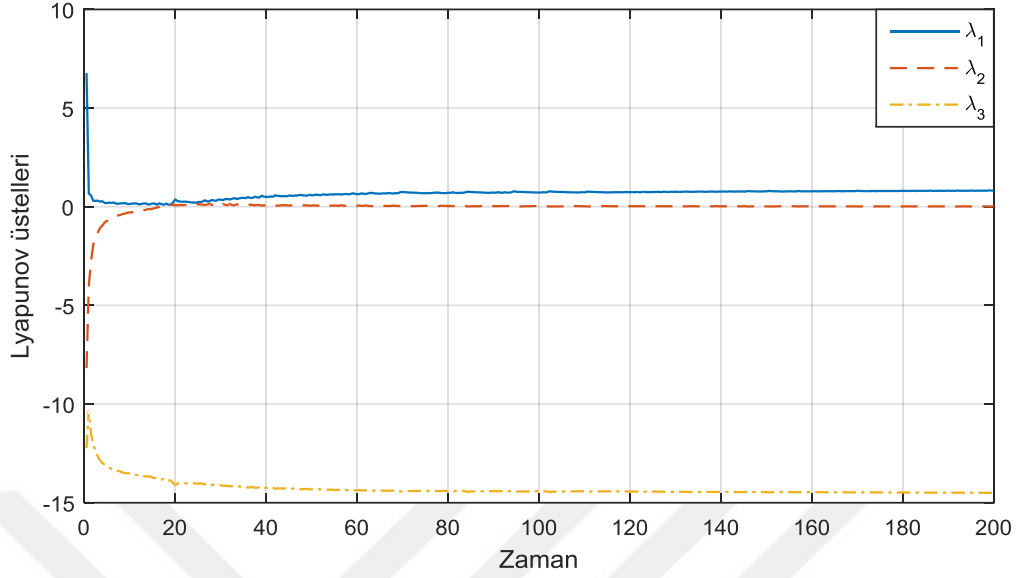


Şekil 3.3. Lorenz sisteminin üç boyutlu yörünge değişimi



Şekil 3.4. Lorenz sisteminin $b \in [0,1]$ aralığı için çatallaşma diyagramı

Sürekli zamanlı kaotik sistemlerin Lyapunov üstelleri hesabında Wolf ve arkadaşlarının önerdiği yöntem [46], araştırmacılar tarafından sıkça kullanılmaktadır. Bu yöntemle göre Lorenz sisteminin Lyapunov üstelleri Şekil 3.5’deki gibi elde edilmiştir.

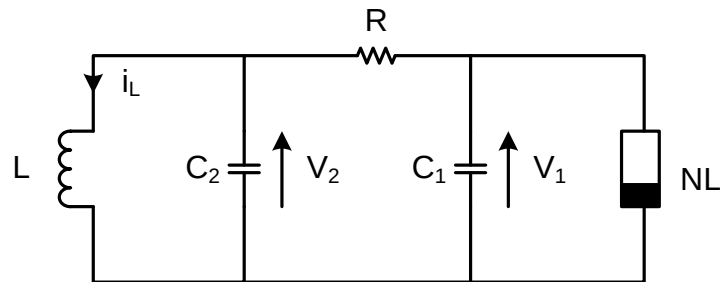


Şekil 3.5. Lorenz sisteminin Lyapunov üstellerinin değişimi

Lorenz sistemi üç durum değişkenine sahip olduğundan, sistemin üç Lyapunov üsteli bulunmaktadır. Bu üstel değerleri kararlı durumda yaklaşık $\lambda_1 = 0.783$, $\lambda_2 = 0.014$ ve $\lambda_3 = -14.47$ olarak elde edilmiştir. Üç veya daha fazla Lyapunov üstel değeri hesaplanan sistemde $\lambda_1 > 0$, $\lambda_2 \cong 0$, $\lambda_i < 0$ şartı $i = 3 \dots n$ için sağlanıyorsa kaotik davranış gösterdiği söylenir. Böylece sistemin üç boyutlu değişimi, çatallaşma diyagramı ve Lyapunov üstelleri incelendiğinde, Lorenz sistemi tuhaf çekici tanımına uyacak şekilde kaotik davranış gösterdiği görülmektedir.

3.3.3. Chua Osilatör Devresi

Chua osilatör devresi, kaotik davranış üretmeye mahsus, Leon O. Chua tarafından geliştirilen ilk kaotik devre olarak bilinir [37]. Devre şeması Şekil 3.6’da gösterildiği gibidir ve aşağıdaki denklem takımı ile tanımlanır.



Şekil 3.6. Chua osilatörü devre şeması

$$\begin{aligned}
\frac{dv_1}{dt} &= \frac{1}{C_1} [G(v_2 - v_1) - g(v_1)] \\
\frac{dv_2}{dt} &= \frac{1}{C_2} [G(v_1 - v_2) + i_L] \\
\frac{di_L}{dt} &= \frac{-1}{L} v_2
\end{aligned} \tag{3.13}$$

Denklem (3.13), $x = v_1$, $y = v_2$, $z = i_L R$, $\alpha = C_2/C_1$, $\beta = C_2 R^2/L$ şeklinde düzenlenirse;

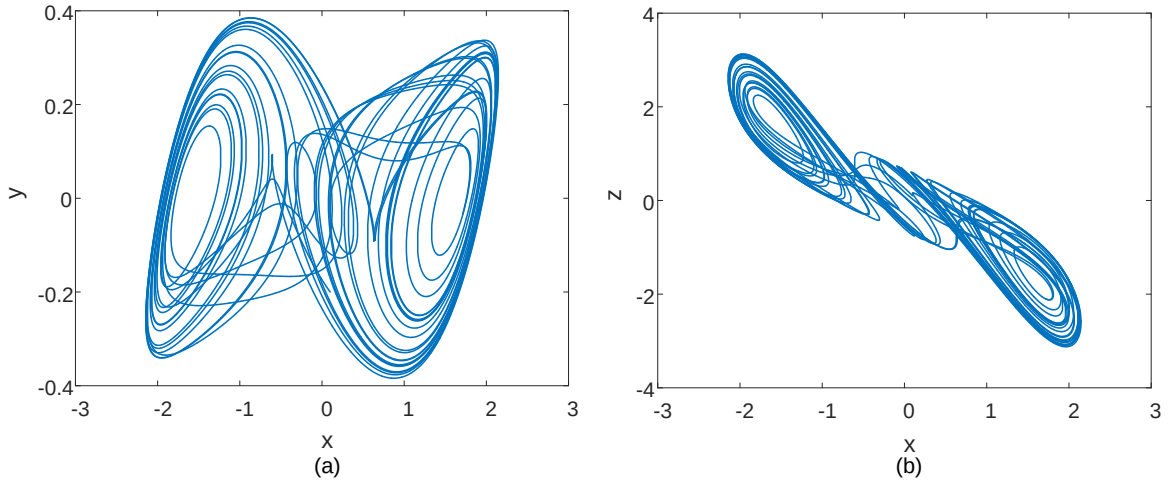
$$\begin{aligned}
\dot{x} &= \alpha[y - h(x)] \\
\dot{y} &= x - y + z \\
\dot{z} &= -\beta y
\end{aligned} \tag{3.14}$$

şeklini alır. Burada $h(x)$ doğrusal olmayan direncin akım-gerilim karakteristiğini $i = g(v_1)$ temsil etmekte ve sisteme doğrusal olmayan özellik katmaktadır. Doğrusal olmayan özellik sağlayan $h(x)$ kübik formda $h(x) = c_1 x^3 + c_0 x$ olarak seçilebilir. Bu durumda doğrusal olmayan parametreler sabit olarak $c_1 = 1/16$ ve $c_0 = -1/6$ şeklinde kullanılmaktadır. Sistemde değişken parametreler ise α ve β 'dir.

Denklem (3.14) İleri Euler yöntemi ile ayırık zamanlı olarak aşağıdaki gibi ifade edilir.

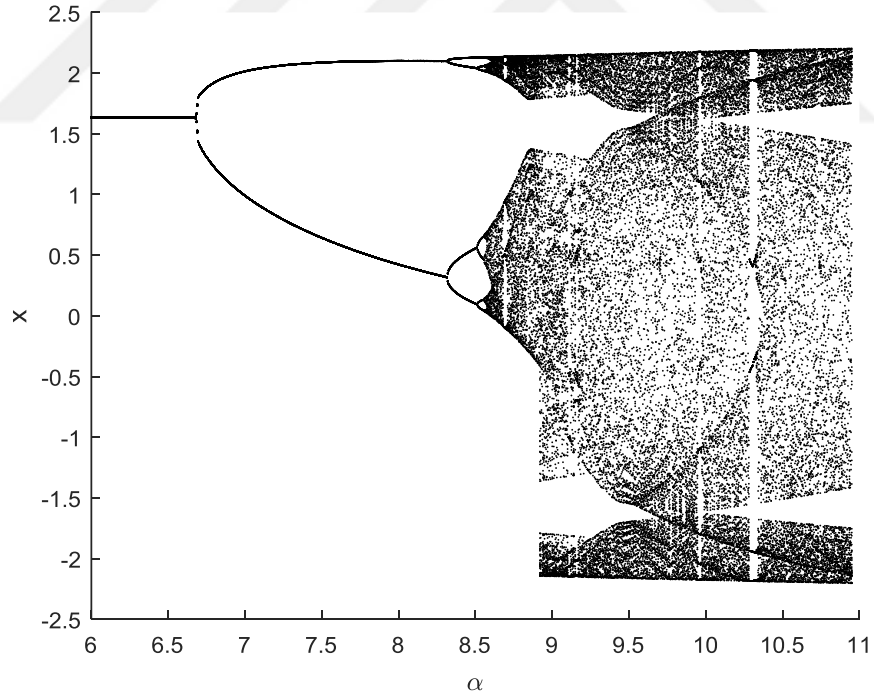
$$\begin{aligned}
x[n+1] &= x[n] - h\alpha(c_1 x^3[n] + c_0 x[n]) + h\alpha y[n] \\
y[n+1] &= (1-h)y[n] + h(x[n] + z[n]) \\
z[n+1] &= z[n] - h\beta y[n]
\end{aligned} \tag{3.15}$$

Ayrık zamanlı denklem takımının $\alpha = 9$, $\beta = 14.286$ parametre değerleri ve $(x_0, y_0, z_0) = [0.1, -0.2, 0.1]$ başlangıç değerleri için MATLAB ortamında $n = 100000$ iterasyon sayısı ve $h = 0.001$ adım aralığında kaotik çeker davranışları Şekil 3.7'deki gibi elde edilir.



Şekil 3.7. Chua osilatör devresinin a) $x - y$ ve b) $x - z$ faz portreleri

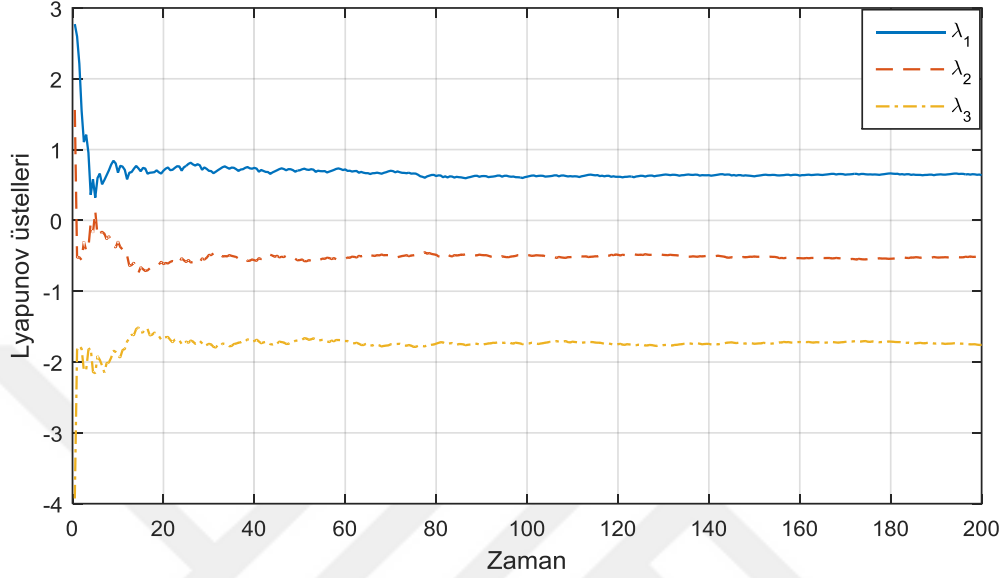
Sistemde $\beta = 14.286$ sabit değeri için α parametresinin $\alpha \in [6,11]$ aralığındaki değişimine göre çatallaşma diyagramı Şekil 3.8’de verilmiştir.



Şekil 3.8. Chua osilatör devresinin $\alpha \in [6,11]$ aralığı için çatallaşma diyagramı

Çatallaşma diyagramında sistemin α ’nın artan değerine karşın kararlı bir denge noktasından sonra limit çevrim daha sonra periyot çiftlemesi yaparak $\alpha \geq 9$ için kaotik çeker davranışı gösterdiği görülmektedir.

Sistemin $\alpha = 9$ ve $\beta = 14.286$ sabit parametre değerleri için başlangıç değerlerine karşın Lyapunov üstellerinin değişimi ise Şekil 3.9’da gösterilmiştir.



Şekil 3.9. Chua osilatör devresinin Lyapunov üstellerinin değişimi

Chua osilatör devresi üç durum değişkenine sahip olduğundan, sistemin üç Lyapunov üsteli bulunmaktadır. Bu üstel değerleri yaklaşık olarak $\lambda_1 = 0.649$, $\lambda_2 = -0.546$ ve $\lambda_3 = -1.736$ şeklinde elde edilmiştir.

3.3.4. 3D Kaotik Sistem

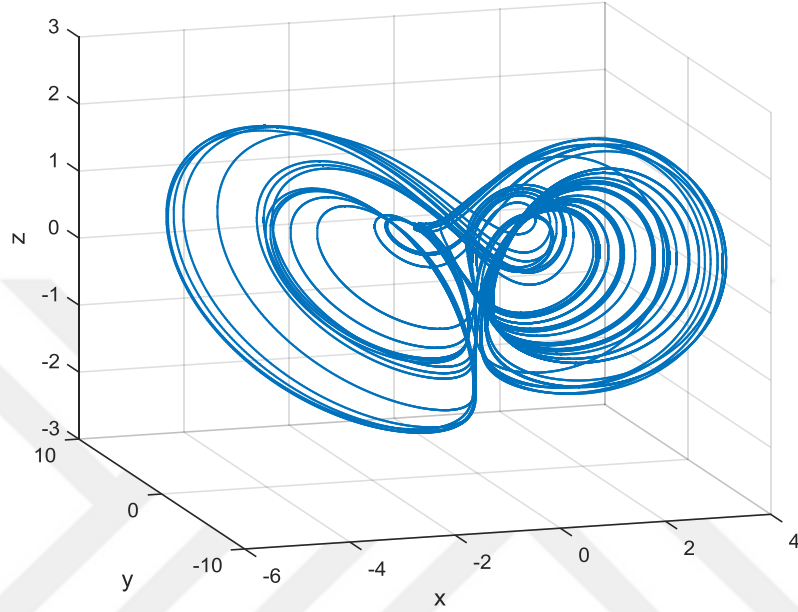
Pehlivan ve arkadaşlarının 2017 yılında önerdiği üç boyutlu (3D) kaotik sistem [38] Denklem (3.16) ile ifade edilir.

$$\begin{aligned}\dot{x} &= y + ax + bxz \\ \dot{y} &= cxz + dx + yz + 1 \\ \dot{z} &= 1 + xy\end{aligned}\tag{3.16}$$

Denklem (3.16) İleri Euler yöntemi ile ayırık zamanlı olarak Denklem (3.17)’deki gibi ifade edilir.

$$\begin{aligned}x[n+1] &= x[n](1 + ah + bhz[n]) + hy[n] \\ y[n+1] &= x[n](chz[n] + dh) + y[n](1 + hz[n]) + h \\ z[n+1] &= hx[n]y[n] + z[n] + h\end{aligned}\tag{3.17}$$

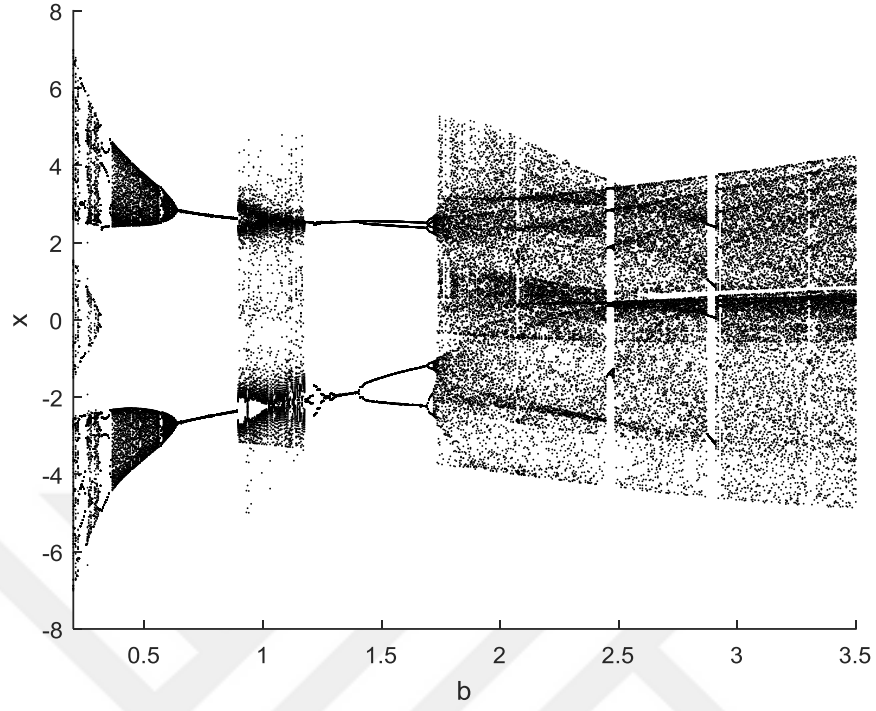
Sistemde kaotik yörünge, $a = -0.6$, $b = 3$, $c = -10$, $d = -0.3$ parametre değerleri ve $(x_0, y_0, z_0) = [0,0,0]$ başlangıç değerleri için MATLAB ortamında $n = 100000$ iterasyon sayısı ve $h = 0.001$ adım aralığında Şekil 3.10'daki gibi elde edilir.



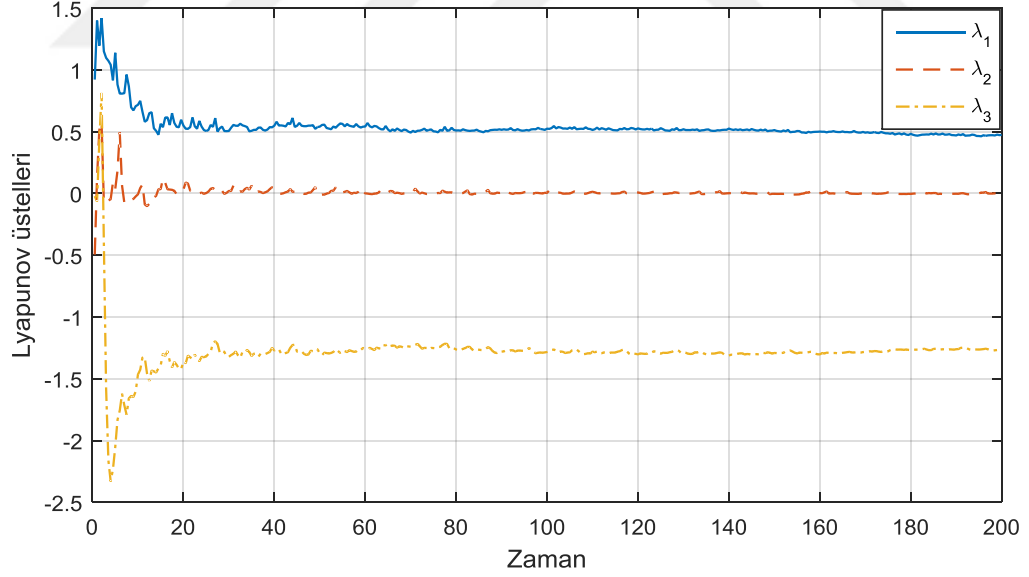
Şekil 3.10. 3D kaotik sisteminin üç boyutlu yörünge değişimi

Sistemde diğer parametreler sabit tutularak b parametresinin $b \in [0.2, 3.5]$ aralığındaki değişimine göre çatallaşma diyagramı Şekil 3.11'de verilmiştir. Çatallaşma diyagramında sistemin b 'nin artan değerine karşın $b \cong 1.4$ değerinde görülen kararlı bir denge noktasından sonra periyot çiftlemesi yaparak $b \geq 3.5$ için kaotik çeker davranışı gösterdiği görülmektedir.

Sistemin sabit parametre değerleri için başlangıç değerlerine karşın Lyapunov üstellerinin değişimi ise Şekil 3.12'de gösterilmiştir.



Şekil 3.11. 3D kaotik sisteminin $b \in [0.2, 3.5]$ aralığı için çatallaşma diyagramı



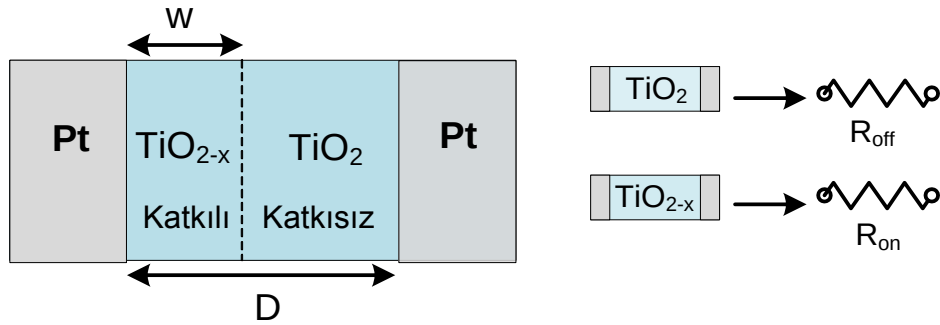
Şekil 3.12. 3D kaotik sisteminin Lyapunov üstellerinin değişimi

Şekil 3.12’de görüldüğü gibi, 3D kaotik sisteminin Lyapunov üsteli değerleri yaklaşık olarak $\lambda_1 = 0.474$, $\lambda_2 = 0.008$ ve $\lambda_3 = -1.258$ elde edilmiştir.

3.3.5. Memristörlü Kaotik Devre

Memristör, hafızalı direnç (*memory resistor*) kelimesinin kısaltılmış hali olarak adlandırılan ve 1971 yılında Leon O. Chua tarafından literatüre sunulan dördüncü temel devre elemanıdır [47]. Bu tarihten sonra uzun yıllar boyunca memristör tanımı teorik bir ifadeden öteye varamamıştır. 2008 yılında Williams ve arkadaşları, memristans olarak adlandırdıkları pratik bir yapıyı literatüre sunmuşlardır [48, 49].

Memristörün içyapısı, silisyum dioksit (SiO_2) ya da titanyum dioksit (TiO_2) bileşiklerinden oluşmaktadır. Araştırmacılar, analog devre yapılarına daha uyumlu olması dolayısıyla TiO_2 bileşiği kullanılarak tasarlanan memristör yapılarını tercih etmektedirler. Bu yapılarda oksijen atomları küçük boşluklar olarak bulunur ve memristörden akım geçirildiğinde TiO_2 katkılı tabakadaki oksijen boşlukları akımın geçtiği yöne doğru hareket ederek katkısız tabaka ile arasında bulunan mesafeyi artırırlar. Bu hareket, yapının akıma karşı koyduğu direnç değerini değiştirir [49]. Bu değişim mantığıyla memristör, bir yönde akım uygulandığında direnci artan tersi yönde akım uygulandığında ise direnci azalan ve uygulanan akım kesildiğinde tekrar akım uygulanılana kadar aynı direnç değerinde kalan bir yapıdır. Bu yapı Şekil 3.13’de görselleştirilmiştir.



Şekil 3.13. Memristörün içyapısı

Memristör temel olarak iletkenin yükü ile akısı arasında bir ilişki kurar. Bu ilişki;

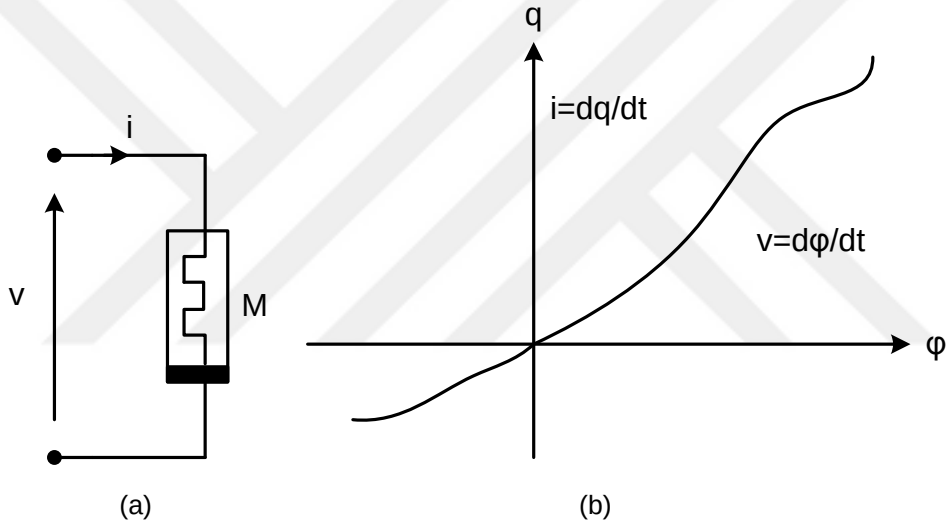
$$d\phi = Mdq \quad (3.18)$$

denklemleri ile ifade edilmektedir. Bu denklemlerden yola çıkarak memristörün davranışı iki şekilde ifade edilebilir.

$$M(q) = \frac{d\varphi(q)}{dq} \quad (3.19)$$

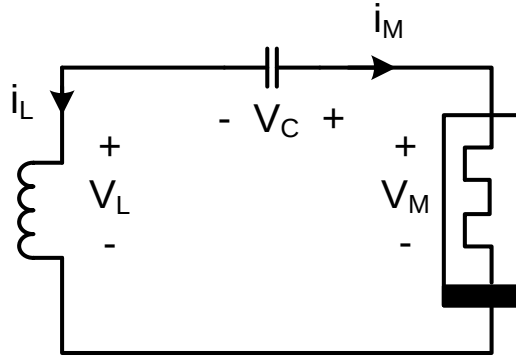
$$W(\varphi) = \frac{dq(\varphi)}{d\varphi} \quad (3.20)$$

Denklem (3.19), yük kontrollü memristörü tanımlarken Denklem (3.20) ise akı kontrollü memristörü tanımlar. Bu denklemlerde M memristansı, W ise memdüktaansı ifade eder. Memristör, iki uçlu eleman olarak Şekil 3.14.a'daki gibi sembolize edilirken, karakteristiği Şekil 3.14.b'deki gibi tanımlanmıştır.



Şekil 3.14. Memristörün a) iki uçlu eleman gösterimi b) akı (q) – yük (φ) karakteristiği

Memristörlerin henüz gerçek eleman olarak üretilmiş bir örneği olmaması nedeniyle araştırmacılar memristör gibi davranan taklitçi devreler tasarlamışlardır. Tasarlanan taklitçi devreler sayesinde memristör karakteristiği içeren birçok devre şemaları literatürde tanıtılıp bu devreler ile kaotik davranış gösteren dinamik sistemler önerilmiştir. Bu tür kaotik devrelere Muthuswamy ve Chua tarafından literatüre sunulan ve Şekil 3.15'de verilen üç elemanlı devre örnek gösterilebilir [39].



Şekil 3.15. Memristörlü kaotik devre şeması

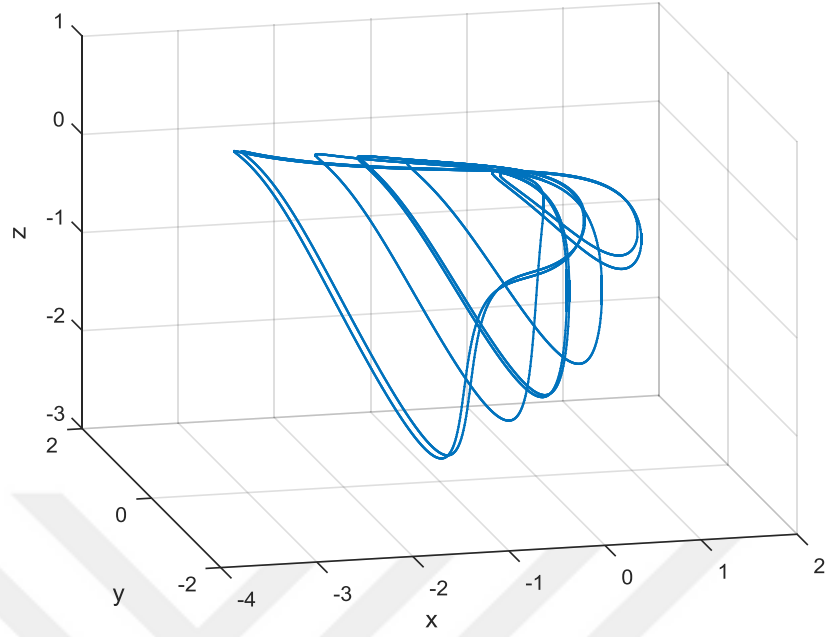
Devrede M , L ve C sırasıyla memristör, endüktans ve kapasitansı temsil etmektedir. Üç elemanlı bu devreye ait durum denklemleri Denklem (3.21)'deki gibi elde edilmektedir.

$$\begin{aligned} \dot{x} &= y/C \\ \dot{y} &= \frac{-1}{L} [x + \beta(z^2 - 1)y] \\ \dot{z} &= -y - \alpha z + yz \end{aligned} \quad (3.21)$$

Denklemde $x = V_C$, $y = i_L$ ve z ise memristörün iç durumunu temsil eder. Bu denklemde memristörün akım-gerilim ilişkisi $V_M = \beta(z^2 - 1)i_M$ ve $\dot{z} = -i_M - \alpha z + i_M z$ eşitlikleri ile tanımlanır. Devrenin kaos parametreleri $C = 1$ ve $L = 3$ sabit, $\alpha = 0.6$ ve $\beta = 1.5$ değişken parametre olarak verilmektedir [39]. Denklem (3.21) İleri Euler yöntemi ile ayrık zamanlı olarak Denklem (3.22)'deki gibi ifade edilir.

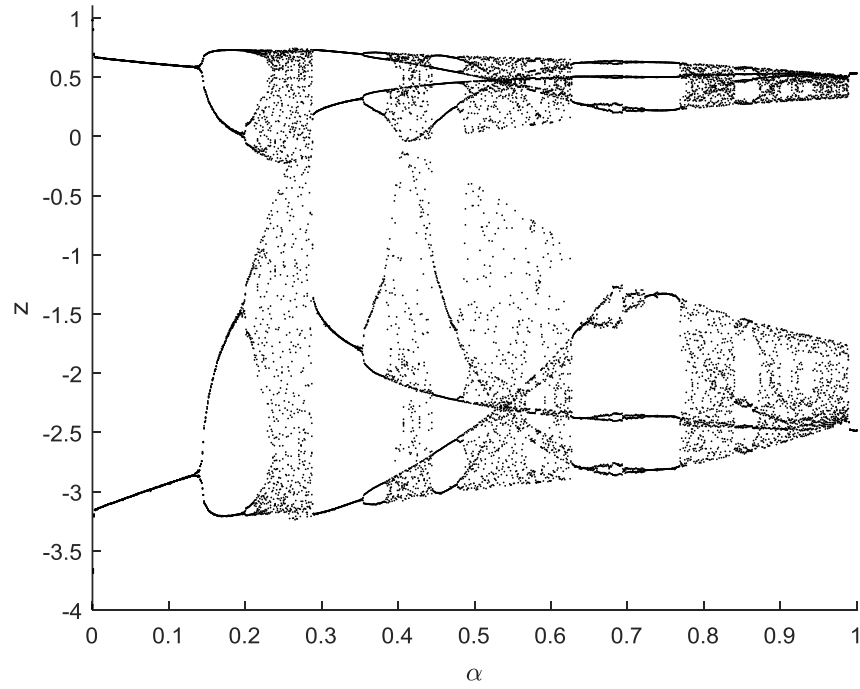
$$\begin{aligned} x[n+1] &= x[n] + hy[n] \\ y[n+1] &= (-h/3) (x[n] + \beta(z[n]^2 - 1)y[n]) + y[n] \\ z[n+1] &= (h(y[n] - \alpha) + 1)z[n] - hy[n] \end{aligned} \quad (3.22)$$

Sistemde kaotik çeker $\alpha = 0.6$ ve $\beta = 1.5$ parametre değerleri ve $(x_0, y_0, z_0) = [0.1, 0, 0]$ başlangıç değerleri için MATLAB ortamında $n = 100000$ iterasyon sayısı ve $h = 0.001$ adım aralığında Şekil 3.16'daki gibi elde edilir.



Şekil 3.16. Memristörlü kaotik devrenin üç boyutlu yörünge değişimi

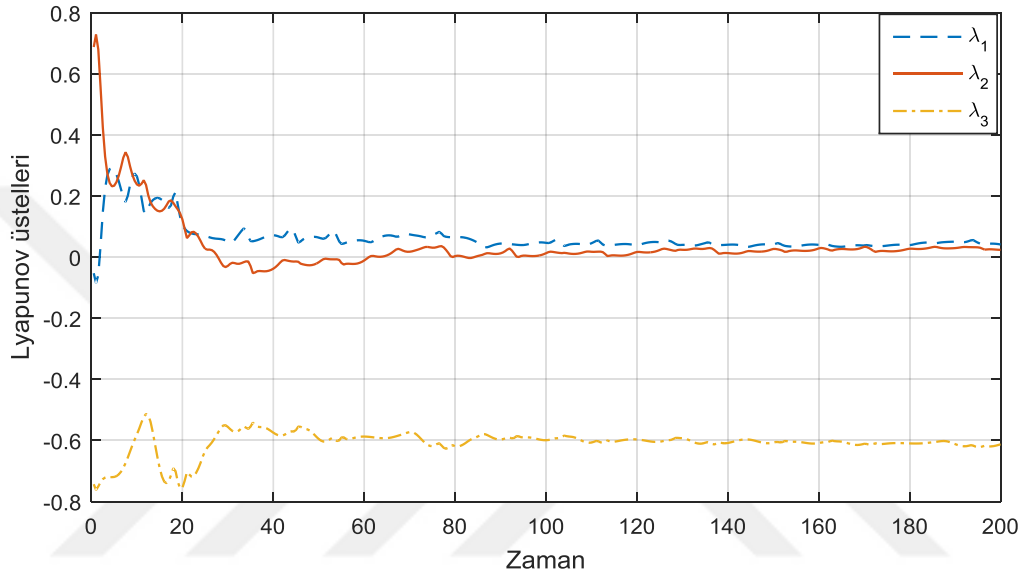
Sistemde diğer parametreler sabit tutularak α parametresinin $\alpha \in [0,1]$ aralığındaki değişimine göre çatallaşma diyagramı Şekil 3.17’de verilmiştir.



Şekil 3.17. Memristörlü kaotik devrenin $\alpha \in [0,1]$ aralığı için çatallaşma diyagramı

Çatallaşma diyagramında sistemin α 'nın artan değerine karşın kararlı iki denge noktasından başlayıp $\alpha \geq 0.17$ değerinden sonra periyot çiftlemesi yaparak kaotik çeker davranışı gösterdiği görülmektedir.

Sistemin sabit parametre değerleri için başlangıç değerlerine karşın Lyapunov üstellerinin değişimi ise Şekil 3.18'de gösterilmiştir.

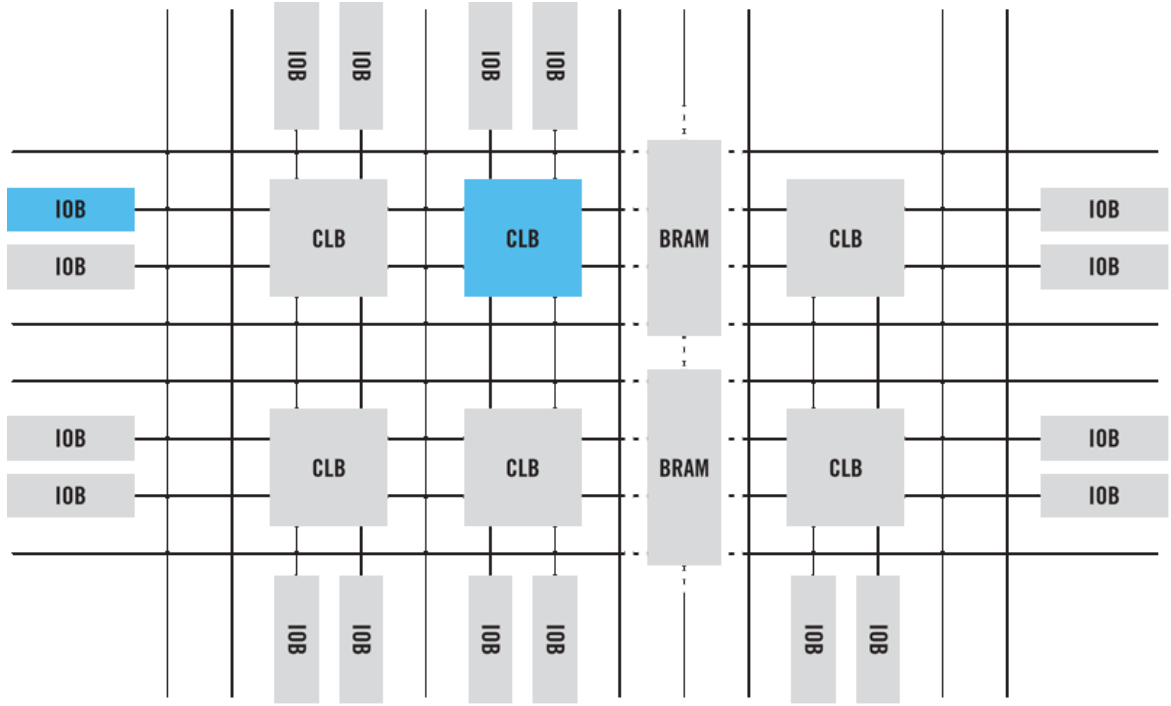


Şekil 3.18. Memristörlü kaotik devrenin Lyapunov üstellerinin değişimi

Memristörlü kaotik devrenin Lyapunov üstel değerleri yaklaşık $\lambda_1 = 0.041$, $\lambda_2 = 0.013$ ve $\lambda_3 = -0.613$ olarak elde edilmiştir.

3.4. Sahada Programlanabilir Kapı Dizileri (FPGA) ve Gerçekleştirilen Tasarımlar

FPGA'lar programlanabilir mantık aygıtları (Programmable Logic Device, PLD) ailesinin bir üyesi olarak 1980'li yılların ortasında Xilinx firması tarafından geliştirilmiştir. FPGA kullanıcıya, tasarımını gerçekleştirdikten sonra dahi müdahale ve değiştirme imkanı vermektedir. Bu özelliği nedeniyle sayısal tasarım yapan araştırmacılar tarafından sıkça kullanılmaktadır. Şekil 3.19'da FPGA'nın içyapısı verilmiştir.



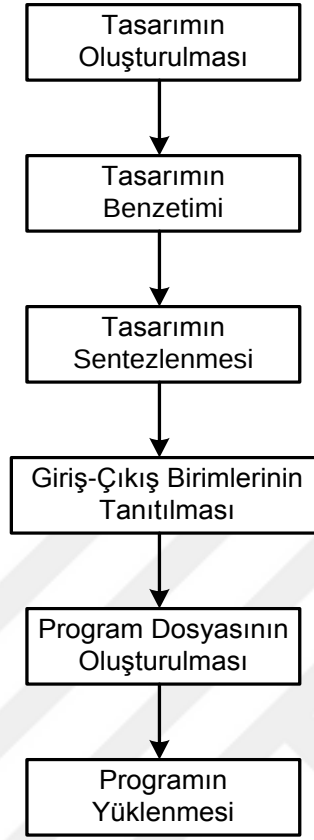
Şekil 3.19. FPGA'nın içyapısı

FPGA'lar içyapısında konfigüre edilebilir mantık blokları (Configurable Logic Block, CLB), giriş çıkış blokları (Input Output Block, IOB) ve rasgele erişimli hafıza blokları (Block Random Access Memory, BRAM) içerir. CLB'lerin yapısında mantık fonksiyonları bulunmaktadır. Kullanıcının tanımladığı bu fonksiyonlar, IOB, BRAM veya diğer CLB fonksiyonlarından gelen bilgiyi işleyerek BRAM, IOB ya da diğer CLB'lere aktarırlar. BRAM, FPGA'da yapılan aritmetik işlemler için gerekli veri depolama ve aktarma işlemleri için kullanılır. IOB ise CLB'ler arası veri alışverişini sağlamak için kullanılır.

Kullanıcı yapacağı tasarım ile aynı zaman diliminde tanımladığı sayı kadar CLB fonksiyonunu işletebilir. Bu durum, FPGA'nın paralel çalışma kabiliyeti olarak adlandırılır. Böylece yüksek çalışma frekanslarında çok hızlı işlemler gerçekleştirirler.

3.4.1. Sayısal Tasarım Aşamaları

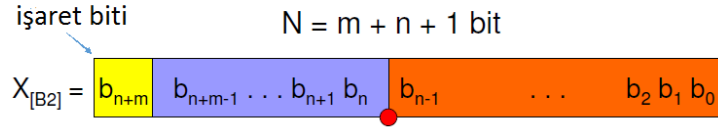
FPGA'da oluşturulan sayısal tasarımlar donanım elemanları kullanırlar. Bu donanım elemanları yüksek seviyeli donanım tanımlama dilleri ile bir arayüz ortamında programlanırlar. Donanım tanımlama dilleri VHDL ve Verilog dilleridir. Tez çalışmasında Verilog dili kullanılarak sayısal tasarımlar gerçekleştirilmiştir. FPGA tasarımlarında sayısal tasarım aşamaları Şekil 3.20'de gösterilmektedir.



Şekil 3.20. FPGA ortamında sayısal tasarımın akış şeması

Sayısal tasarımın en önemli aşaması benzetim aşamasıdır. Kullanıcı, oluşturduğu tasarımın hedeflenen aritmetik işlemleri doğru sırada ve sürede gerçekleşip gerçekleşmeyeceğini bu aşamada izler. Bu süreçten sonra tasarımın aşamaları adım adım işletilerek program dosyası FPGA çipinin bulunduğu kite yüklenir. Böylece gerekli girişler uygulanarak hedeflenen çıkışlar elde edilir.

Tez çalışmasında RSÜ'nün sayısal tasarımında entropi kaynağı olarak kaotik davranış gösteren HYSA modeli kullanılmıştır. HYSA modeli ayrık zamanlı denklemlere dönüştürülerek FPGA programlama arayüzünde Verilog dili ile kodlanmıştır. Modele ait parametre, başlangıç ve durum değişkenlerine ait tüm değerler ondalıklı sayılardan oluştuğu için kodlama aşamasında ondalıklı sayı gösterim formatı kullanılmıştır. Ondalıklı sayı gösterim formatları iki çeşittir. Bunlar, kayan noktalı (floating point) ve sabit noktalı (fixed point) sayı aritmetikleridir. Tez çalışmasında 2'ye tümleyen işaretli sabit noktalı sayı gösterim formatı kullanılmıştır. Bu format $Qm.n$ olarak sembolize edilip Şekil 3.21'deki gibi gösterilmektedir.



Şekil 3.21. $Qm.n$ sabit noktalı sayı aritmetiğinin gösterimi

Sabit noktalı sayı aritmetiğinde m , sayının tam kısmını n ise ondalıklı kısmını temsil etmek için kullanılan bit sayısıdır.

$$x_{[B10]} = \frac{1}{2^n} \left[-2^{N-1} b_{N-1} + \sum_{i=0}^{N-2} 2^i b_i \right] \quad (3.23)$$

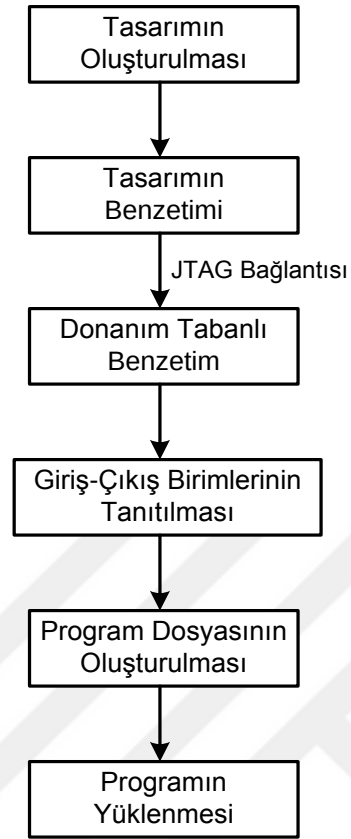
Elde edilen ikili tabanda $Qm.n$ formatındaki ondalıklı sayı, Denklem (3.23) kullanılarak onluk tabana çevrilebilir. Sabit noktalı sayı aritmetiğinde n sayısı ne kadar büyük seçilirse 2^{-n} formülüyle o kadar küçük adım aralığı elde edilir ve o kadar yüksek doğrulukla dönüşüm yapılmış olur.

3.4.2. Xilinx Sistem Üretici (XSG) ve Kaotik Devre Tasarımları

FPGA ortamında çalıştırılmak istenen sayısal tasarım, Xilinx ISE ve Vivado program arayüzleriyle gerek mantıksal operatörler ile şematik olarak, gerekse VHDL ve Verilog dilleri kullanılarak kodlarla oluşturulabilir.

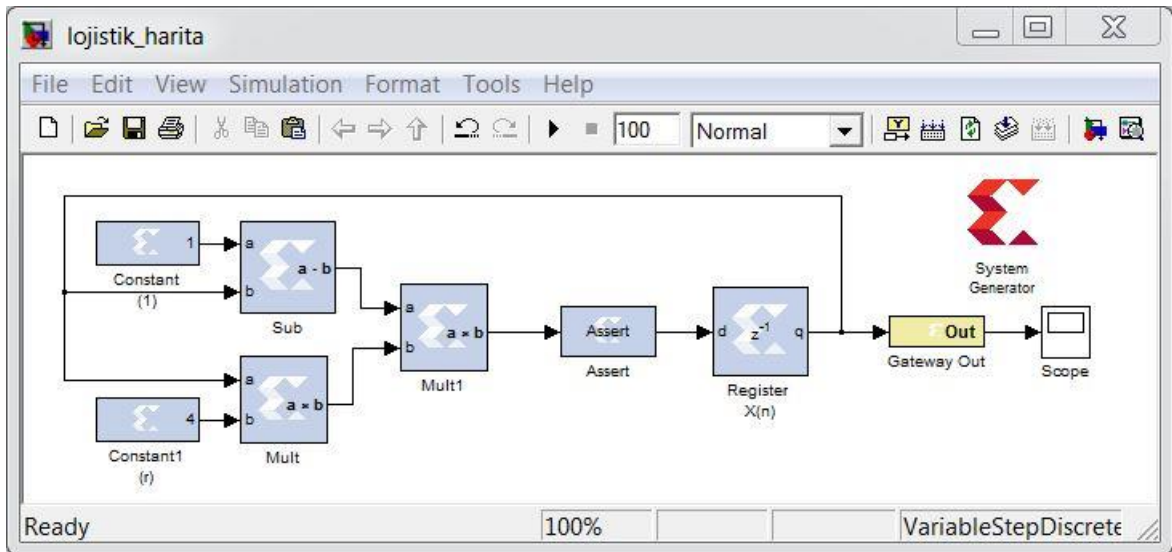
Xilinx firması tasarımcılara büyük bir kolaylık sağlayan Xilinx sistem üretici (Xilinx System Generator, XSG) platformunu üretmiştir. Bu platform ile tasarımcılar uzun kodlamalardan kurtularak MATLAB/Simulink ortamında FPGA tasarımlarını yapabilmektedirler. XSG platformunda MATLAB/Simulink ortamında Xilinx Blockset modülünde yer alan aritmetik ve mantıksal bloklar aracılığıyla tasarım yapılabilir. Oluşturulan tasarım benzetim, donanım tabanlı benzetim, programlama ve gerçek zamanlı çalıştırma gibi adımlarla seçilen FPGA çipinde çalıştırılır.

XSG bloklarının MATLAB/Simulink klasik bloklarından en önemli farkı, aritmetik ve mantıksal blokların ayrı zamanlı olarak bir döngü içerisinde sabit noktalı sayı aritmetiğiyle tasarımda kullanılmasıdır. Böylece Xilinx ISE ve Vivado program arayüzlerinde oluşturulan ayrı zamanlı tasarımlar ile eşdeğer tasarım gerçekleştirilir. XSG platformunda yapılan tasarımın aşamaları Şekil 3.22’de gösterilmektedir.



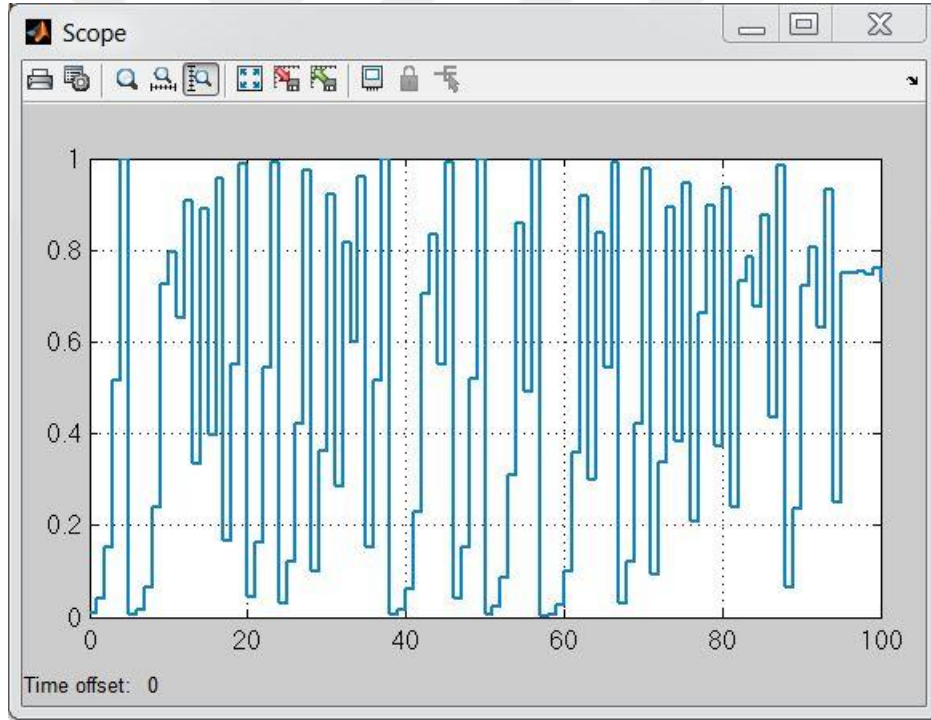
Şekil 3.22. XSG platformunda tasarımın akış şeması

Devam eden kısımda XSG tasarımlarına bir örnek olması açısından, lojistik harita sistemi için tasarım aşamalarına yer verilmiştir. Denklem (3.4)’de verilen ayrık zamanlı sistem Şekil 3.23’deki XSG modeli ile ifade edilmiştir.



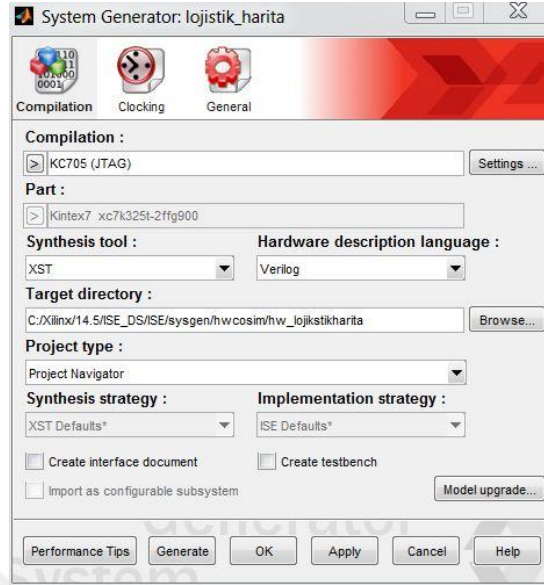
Şekil 3.23. Lojistik harita sisteminin XSG modeli

Modelde, kayıtçı (register) bloğu ile x durum değişkeninin başlangıç değeri kaydedilerek her döngü için üretilen yeni değer tekrar başlangıç değeri olarak saklanır. İki adet sabit sayı (constant) bloğu ise sistemin tek parametresi olan r ve sabit sayısı 1 için kullanılmıştır. Bu blokların dışında “Sub” ve “Mult” blokları sırasıyla çıkarma (subtraction) ve çarpma (multiplication) işlemleri için kullanılmaktadır. “Assert” bloğu ise modeldeki tüm işlemlerde tercih edilen sabit noktalı sayı formatını ve örnekleme sayısını kaydederek kayıtçıya gelen yeni değeri kontrol eder. “Gateway Out” bloğu da elde edilen değerlerin XSG blokları dışında kullanımı ve izlenmesi için veri iletimini sağlar. Modelde, Q8.24 2’ye tımleyen işaretili sabit noktalı sayı aritmetiği tercih edilmiştir. Tasarımın $x_0 = 0.01$ başlangıç değeri, $r = 4$ parametre değeri ve 100 iterasyon için benzetim sonucu Şekil 3.24’deki gibi elde edilir.



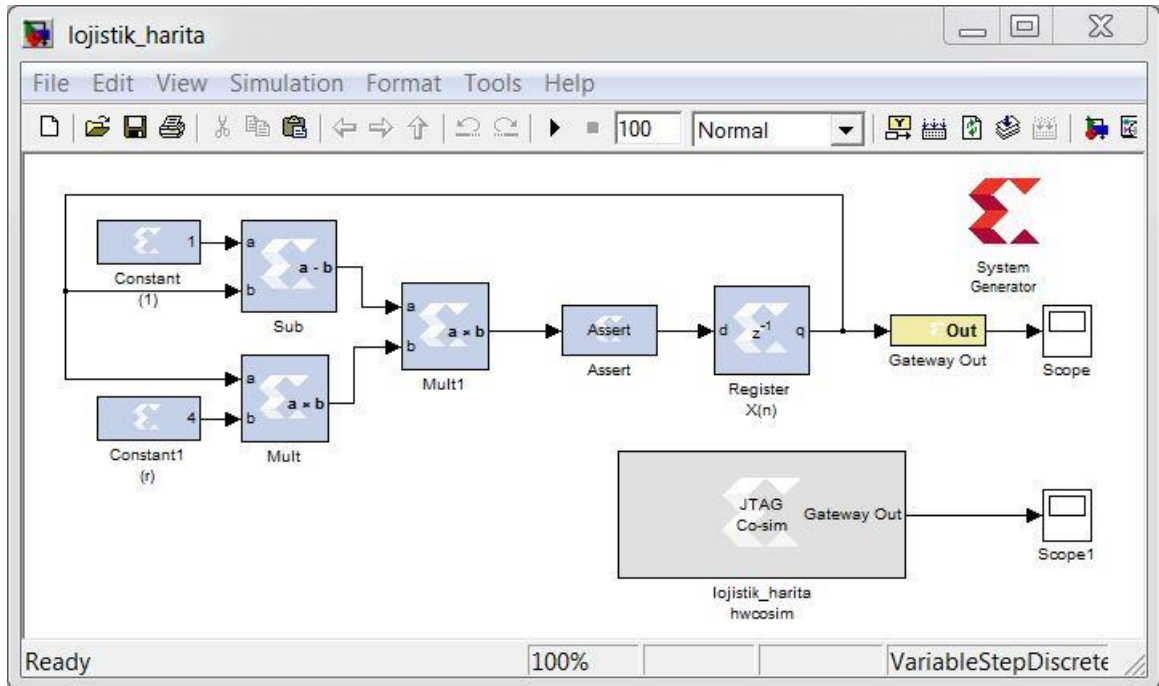
Şekil 3.24. Lojistik haritanın XSG modelinin benzetim sonucu

Tasarımın benzetimi yapıldıktan sonra modeldeki “System Generator” ikonu çift tıklanarak derleme yöntemi “Hardware Co-Simulation → KC705 → JTAG” şeklinde Şekil 3.25’deki gibi seçilir. Bu aşamadan sonra seçilen FPGA kiti bilgisayara JTAG kablosu ile bağlı tutulur.



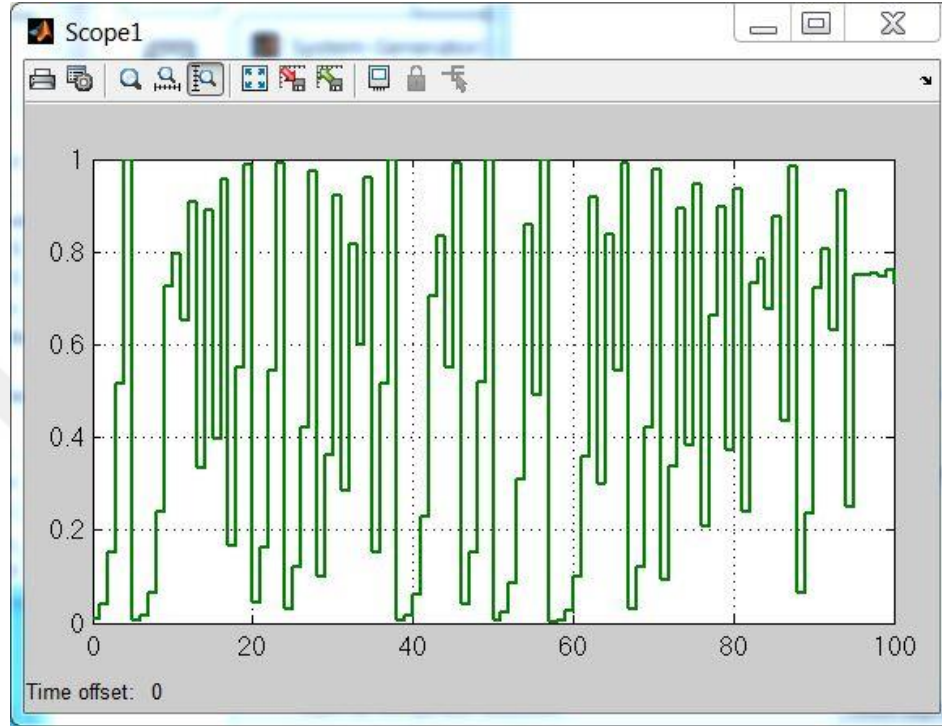
Şekil 3.25. XSG modelinin derleme ekranı

Tasarımın Xilinx KC705 Evaluation Kit’inde yer alan Kintex-7 FPGA çipinde donanım tabanlı benzetimi yapılmaktadır. Benzetimi başlatmak için ekranda görülen “Generate” sekmesine tıklanır ve modele ait eklenmesi gereken JTAG Co-Sim bloğu üretilir. Bloğun eklenmiş haline ilişkin modelin görüntüsü Şekil 3.26’daki gibidir.



Şekil 3.26. Lojistik haritanın XSG modelinin son hali

Model aynı değerlerle tekrar çalıştırıldığında ise “Scope1” ekranında Şekil 3.27’deki çıkış elde edilir.



Şekil 3.27. Lojistik haritanın XSG modelinin donanım tabanlı benzetim sonucu

Elde edilen bu sonuç ile Şekil 3.24’deki sonuç tamamen aynıdır. Böylece donanım tabanlı benzetim ile beklenen sonuçlar elde edilmiştir. Bu aşama sırasında XSG, modeli donanım ortamında ne kadar mantıksal eleman kullanarak gerçekleştireceğini hesaplar. Bu eleman sayıları için FPGA kaynak kullanım sayısı tabiri kullanılır ve kullanım sayısı Tablo 3.1’de verilmiştir.

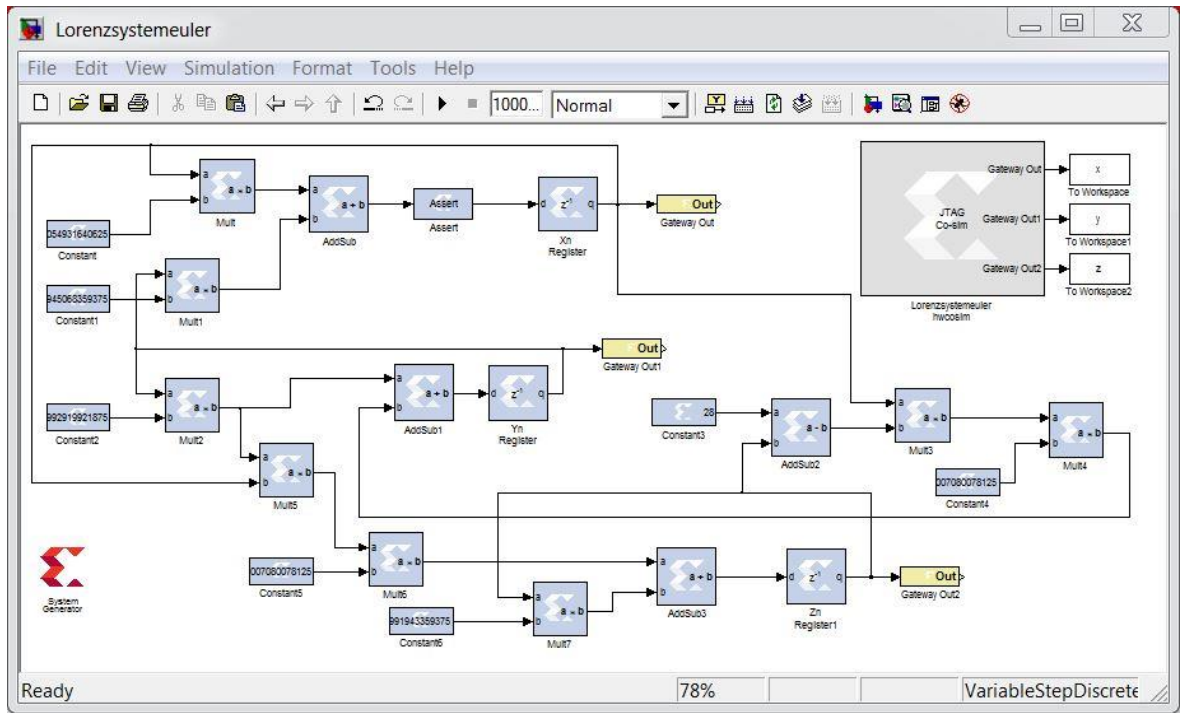
Tablo 3.1. Lojistik haritanın XSG modelinin FPGA kaynak kullanımı

Kaynak Adı	Kullanım Sayısı
FF (Flip Flop)	212
LUT (Look up Table)	286

Tasarım, geriye kalan adımlar doğrultusunda gerçek zamanlı çıkışların FPGA’da gözlenmesi şeklinde devam ettirilir. Böylece verilen örnek tasarım ile XSG platformunun kullanım kolaylığı vurgulanmıştır.

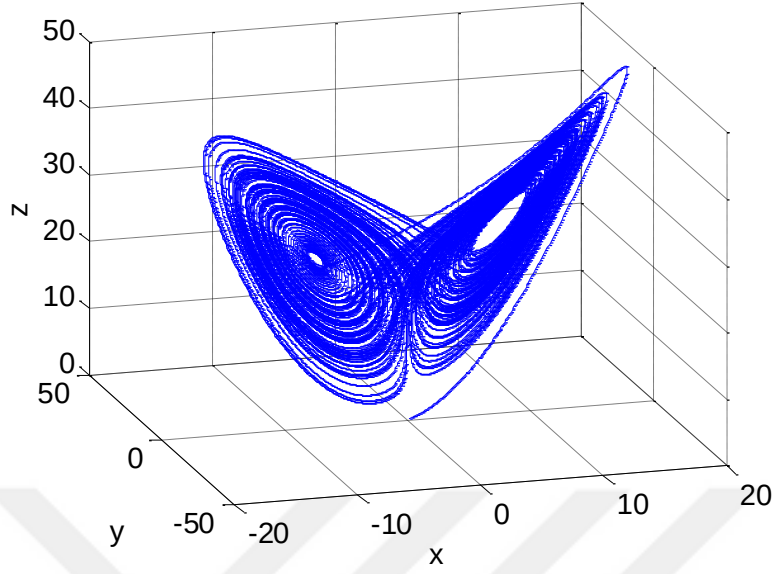
Tez sürecinde yapılan çalışmalar kapsamında önceki bölümlerde detaylı analizleri yapılan Lorenz ve memristörlü kaotik devre modelleri de XSG platformunda tasarlanarak donanım tabanlı sonuçlar elde edilmiştir [8, 50]. Yapılan çalışmalarda kullanılmış FPGA kitleri farklı olsa da, bu kısımda karşılaştırma amacıyla aynı modeller Kintex 7 FPGA çipi seçilerek tekrar çalıştırılmıştır.

Denklem (3.12)'de verilen ayrık zamanlı Lorenz kaotik sisteminin, Şekil 3.28'de gösterilen XSG tasarımı ile donanım tabanlı benzetimi gerçekleştirilmiştir.



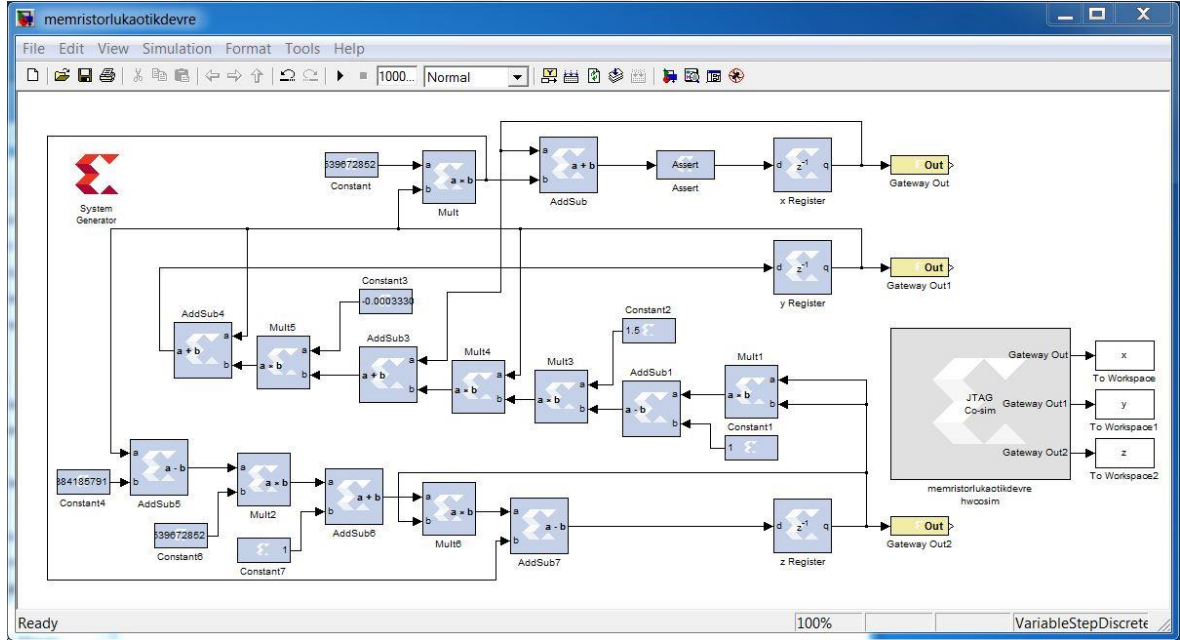
Şekil 3.28. Lorenz sisteminin XSG modeli

Şekil 3.28'de verilen donanım tabanlı benzetim çıktısı MATLAB çalışma alanına iletilerek üç durum değişkeninin değişimi Şekil 3.29'daki gibi elde edilmiştir.



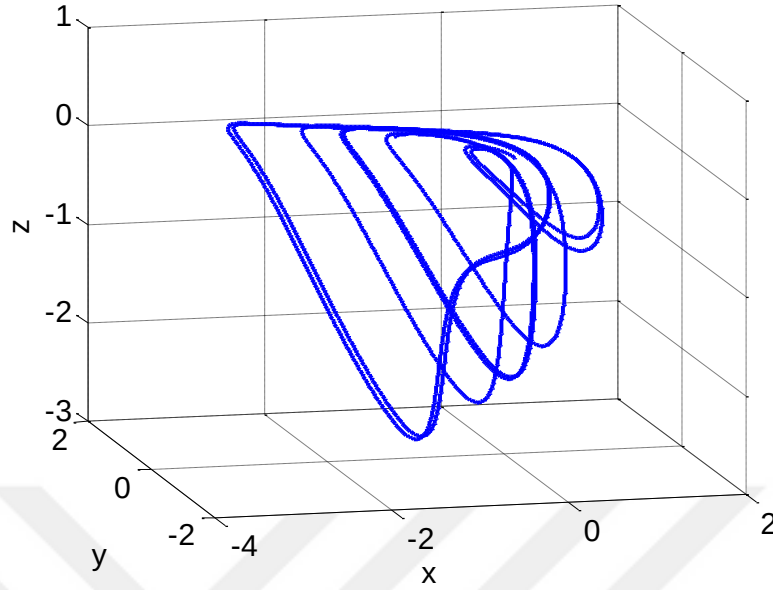
Şekil 3.29. Lorenz sisteminin üç boyutlu donanım tabanlı benzetim sonucu

Ayrıca, denklem (3.22)'de verilen ayrık zamanlı memristörlü kaotik devrenin, Şekil 3.30'da gösterilen XSG tasarımı ile donanım tabanlı benzetimi gerçekleştirilmiştir.



Şekil 3.30. Memristörlü kaotik devrenin XSG modeli

Donanım tabanlı benzetim çıktısı MATLAB çalışma alanına iletilerek üç durum değişkeninin birbirine göre değişimi Şekil 3.31'deki gibi elde edilmiştir.



Şekil 3.31. Memristörlü kaotik devrenin üç boyutlu donanım tabanlı benzetim sonucu

Üç durum değişkeninin birbirine göre değişimi XSG platformunda donanım tabanlı benzetim ile elde edilen kaotik devrelerin FPGA kaynak kullanım sayıları Tablo 3.2’de verilmiştir.

Tablo 3.2. XSG modellerinin FPGA kaynak kullanımları

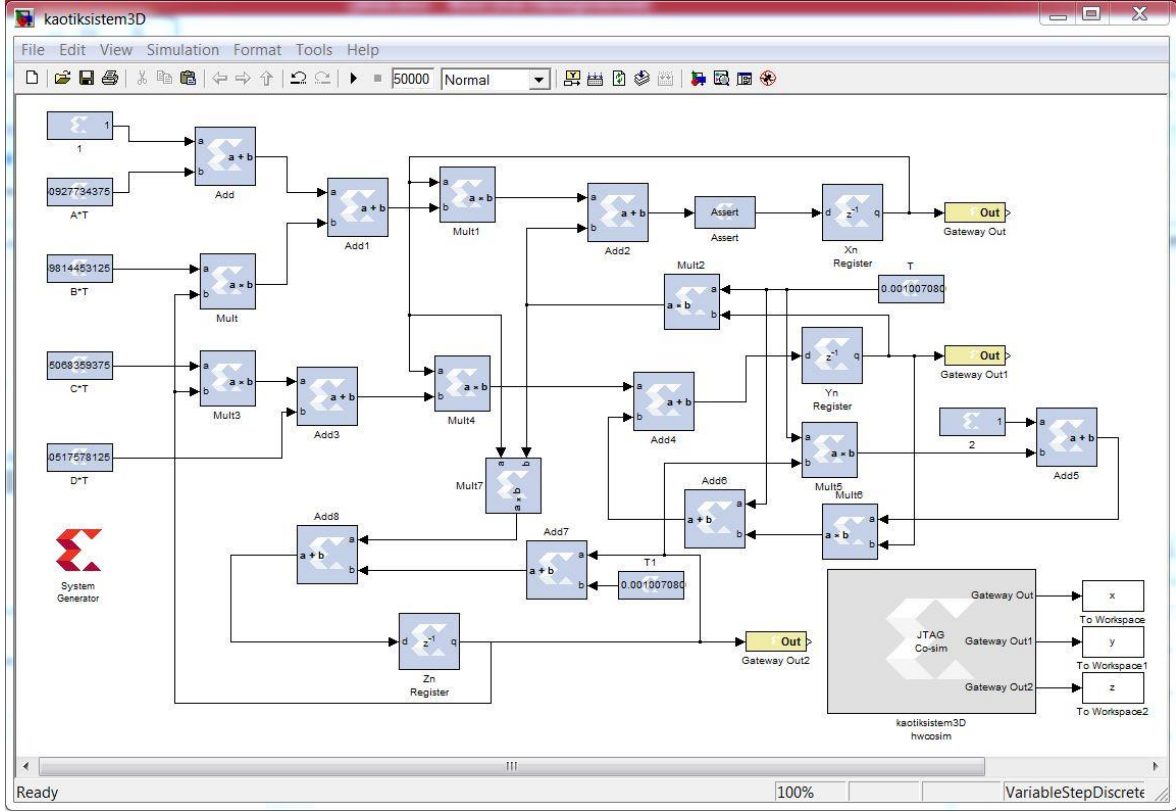
Modelin Adı	FFs sayısı	LUTs Sayısı
Lorenz kaotik sistem	486	508
Memristörlü kaotik devre	618	656

Tablo 3.1 ile Tablo 3.2 karşılaştırıldığında, XSG modelinde yer alan durum değişkeni ve aritmetik işlem sayılarının artmasıyla FPGA kaynak kullanımının doğru orantılı bir şekilde arttığı görülür. Fakat bu değerler XSG platformunun sentezleme sonucunda elde edilmiştir. Sistem gerçekleştirme aşamasında modeli optimize ederek daha az eleman sayısı ile gerçekleştirir.

3.4.3. 3D Kaotik Sistemi ile RSÜ Tasarımı

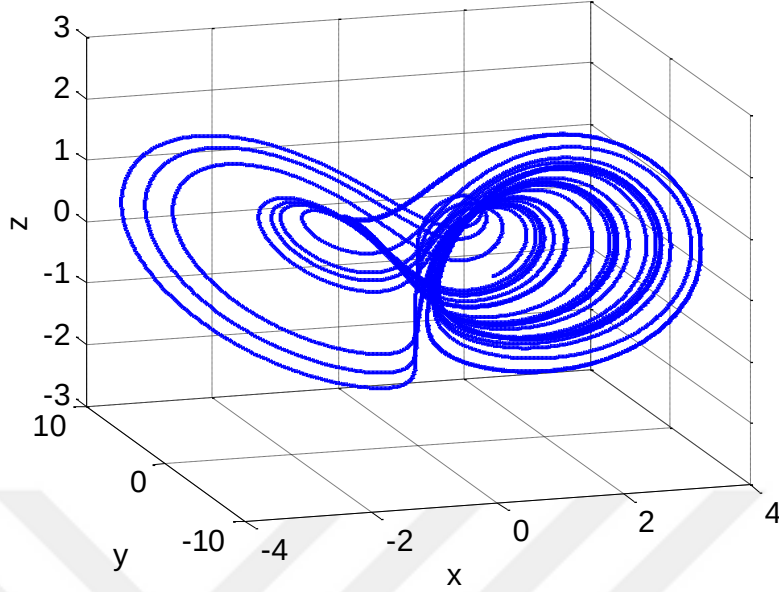
Tez çalışmasında önerilen ilk RSÜ tasarımı [51], 3D kaotik sistem tabanlı RSÜ tasarımıdır. Bu tasarım, entropi kaynağı olarak 3D kaotik sistemini kullanır. Sisteme ait analizler önceki bölümlerde verilmiş olup Denklem (3.17)’deki ayrık zamanlı denklemler

XSG platformunda Şekil 3.32'deki gibi modellenenerek 50000 iterasyon için donanım tabanlı benzetim gerçekleştirilmiştir.



Şekil 3.32. 3D kaotik sisteminin XSG modeli

Durum değişkenlerine ait donanım tabanlı benzetim sonuçları MATLAB çalışma alanına iletilerek üç durum değişkeninin birbirine göre değişimi Şekil 3.33'deki gibi elde edilmiştir. Ayrıca 3D kaotik sistemin FPGA kaynak kullanımı Tablo 3.3'de verilmiştir.



Şekil 3.33. 3D kaotik sisteminin üç boyutlu donanım tabanlı benzetim sonucu

Tablo 3.3. 3D kaotik sisteminin XSG modelinin FPGA kaynak kullanımı

Kaynak Adı	Kullanım Sayısı
FF	496
LUT	619

Durum değişkenlerinin değerlerini MATLAB çalışma alanına ileterek değişkenlerin ortalama değeri hesaplanır. Hesaplanan ortalama değerler Denklem (3.24) ve (3.25) kullanılarak kaotik anahtar bit dizisi üretilir.

$$anahtardizisi[n] = \begin{cases} x[n]; & y[n] \geq x_{ort} \\ z[n]; & y[n] < x_{ort} \end{cases} \quad (3.24)$$

$$anahtar[n] = \begin{cases} Lojik '1'; & anahtardizisi[n] \geq z_{ort} \\ Lojik '0'; & anahtardizisi[n] < z_{ort} \end{cases} \quad (3.25)$$

Denklemlerde x_{ort} ve z_{ort} sırasıyla x ve z ayırık zamanlı durum değişkenlerinin 50000 iterasyon için hesaplanan değerlerin ortalamasıdır. Böylece 50000 bit uzunluğunda *anahtar* verisi üretilir.

RSÜ tasarımının ikinci rasgele bit dizisi ise 3D kaotik sisteminin x ve y ayırık zamanlı durum değişkenleri kullanılarak Şekil 3.34’de verilen MATLAB kodu ve Trivium

algoritması [11] çalıştırılarak elde edilir. Trivium algoritması sonraki bölümde detaylı bir şekilde verilecektir.

Şekil 3.34'deki kodlar çalıştırıldığında 80 bit uzunluklu iki bit dizisi elde edilir. Bu diziler, Trivium algoritmasına giriş olarak tanımlanır. Algoritma 50000 iterasyon için çalıştırılarak ikinci rasgele bit dizisi elde edilir.

Önerilen RSÜ tasarımı son işlem olarak XOR fonksiyonu kullanılmıştır. Elde edilen 50000 bit uzunluklu rasgele bit dizileri (anahtar ve Trivium algoritma çıkış dizisi) XOR fonksiyonuna uygulanır ve nihai sonuç olarak yine 50000 rasgele bit dizisi üretilmiş olur. Nihai bit dizisinin NIST istatistiksel test sonuçları Tablo 3.4'deki gibidir.

```
x1=abs(x);  
xm=mean(x1);  
dimension=length(x);j=1;  
for k=1:630:dimension  
    if x1(k)>=xm  
        secretkey(j,1)=1;  
    else  
        secretkey(j,1)=0;  
    end  
    j=j+1;  
end  
  
y1=abs(y);  
ym=mean(y1);  
dimension=length(y);j=1;  
for k=1:630:dimension  
    if y1(k)>=ym  
        initvalue(j,1)=1;  
    else  
        initvalue(j,1)=0;  
    end  
    j=j+1;  
end
```

Şekil 3.34. Trivium algoritması için giriş dizilerinin oluşturulması

Önerilen RSÜ tasarımıyla NIST testlerinin 13'ünü başarılı olarak geçen 50000 bit uzunluklu rasgele bit üretimi gerçekleştirilmiştir. Başarısız olunan diğer 3 test için daha fazla bit üretimi sağlanarak başarılı sonuç alınabilir. Özellikle Lempel Ziv testi için en az bir milyon bit uzunluğu gereklidir.

Önerilen RSÜ tasarımı, entropi kaynağı olarak hem fiziksel olmayan GRSÜ (3D kaotik sistem modeli) hem de SRSÜ (deterministik Trivium algoritması) tasarımı içermektedir. Bu nedenle 3D kaotik sistem tabanlı RSÜ tasarımı Hibrit RSÜ olarak tanımlanabilir.

Tablo 3.4. 3D kaotik sistem tabanlı RSÜ çıkışının NIST istatistiksel test sonuçları

NIST Testleri	3D Kaotik Sistem Tabanlı RSÜ (50000 bit)	Sonuç
Frekans	0.7781	Başarılı
Bir blok içindeki frekans	0.2407	Başarılı
Akış	0.5107	Başarılı
Bir blok içinde en uzun birlerin akış	0.529	Başarılı
İkili matris rankı	0.9466	Başarılı
Ayrık Fourier dönüşümü	0.5511	Başarılı
Örtüşmeyen şablon eşleştirme	0.4657	Başarılı
Örtüşen şablon eşleştirme	0.7486	Başarılı
Maurer evrensel istatistik	0.4936	Başarılı
Doğrusal karmaşıklık	0.6119	Başarılı
Seri (1/2)	0.949 / 0.7793	Başarılı
Yaklaşık entropi	0.9056	Başarılı
Kümülatif toplamlar	0.7192	Başarılı

4. HÜCRESEL YAPAY SİNİR AĞI TABANLI RSÜ UYGULAMALARI

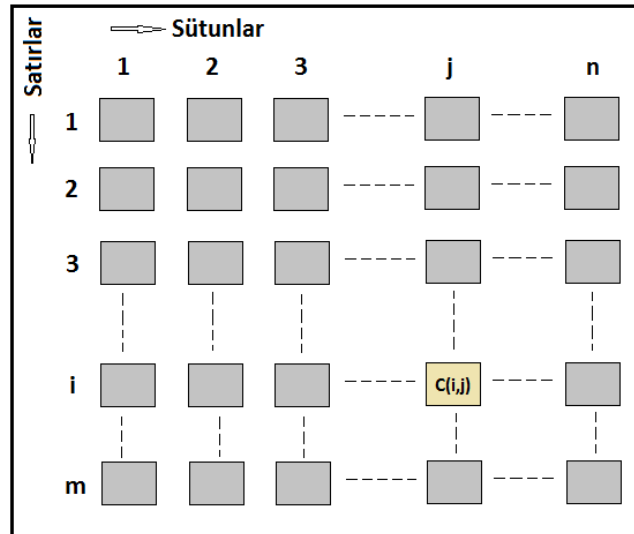
Bu bölümde HYSA modeli, HYSA tabanlı iki RSÜ tasarımı ve FPGA ortamında gerçekleştirmeleri detaylı bir şekilde anlatılmaktadır.

İlk RSÜ tasarımında, üç hücreli otonom HYSA modeli kullanılmıştır. Model, sürekli zamanlı olarak incelenip diferansiyel denklemler ayrık zamanlı denklemlere dönüştürülmüştür. Bu model kullanılarak üç farklı senaryo için RSÜ tasarımı yapılmış ve FPGA ortamında gerçekleştirilmiştir. Gerçekleştirilen RSÜ tasarımında son işlem olarak Trivium algoritması kullanılmıştır.

İkinci RSÜ tasarımında ise kaotik davranış sergileyen Memristörlü HYSA modeli kullanılmıştır. Modele ait diferansiyel denklemler ayrık zamanlı denklemlere dönüştürülmüştür. Uygulamada ayrık zamanlı Memristörlü HYSA modelinin yanı sıra lojistik harita da kullanılmıştır. Bu iki kaotik sinyal örneklenip sayısallaştırıldıktan sonra XOR son işlemi uygulanarak rasgele bit dizisi üretilmiştir.

4.1. HYSA Modeli

Hücresel yapay sinir ağı, Leon O. Chua ve Lin Yang tarafından yapılan çalışmalar [52, 53] ile yeni bir sinir ağı tasarımı olarak 1988 yılında tanıtılmıştır. HYSA yapısında her bir hücre en yakın komşu hücreler ile doğrudan diğer hücreler ile dolaylı olarak bağlıdır. Şekil 4.1’de HYSA mimarisine ilişkin standart bir gösterim verilmektedir.



Şekil 4.1. $m \times n$ boyutlu HYSA mimarisine ilişkin standart bir gösterim

1993’de ise Leon O. Chua ve Thomas Roska [54] önceki 5 yıl içinde HYSA ile ilgili yapılan çalışmaları özetleyerek araştırmacıların ilgisini çekmeyi başarmışlardır. Bu çalışmadan sonra HYSA mimarisinin analog ve sayısal işlem yeteneği keşfedilmiş [55], bu özelliği sayesinde HYSA görüntü işleme uygulamasında kullanılmaya başlanmıştır [56].

Chua-Yang modeli olarak da adlandırılan HYSA mimarisinde yer alan her bir $C(i,j)$ hücrelerine ait matematiksel model Denklem (4.1)’deki gibidir.

$$\frac{dx_{ij}}{dt} = -x_{ij} + \sum_{C(k,l) \in S_r(i,j)} A(i,j;k,l)y_{kl} + \sum_{C(k,l) \in S_r(i,j)} B(i,j;k,l)u_{kl} + I_{ij} \quad (4.1)$$

Denklemde;

$x_{ij} \in R$, $C(i,j)$ hücrenin durum değişkeni
 $y_{kl} \in R$, etki küresi içerisindeki hücre çıkışı
 $u_{kl} \in R$, etki küresi içerisindeki hücre girişi
 $I_{ij} \in R$, bias değeri
 $S_r(i,j)$, $C(i,j)$ hücrenin r yarıçaplı etki küresi
 $C(k,l)$, $S_r(i,j)$ etki küresi içerisindeki komşu hücre
 $A(i,j;k,l)$, geri besleme operatörü
 $B(i,j;k,l)$, giriş operatörü

olarak tanımlanır. Modele ait çıkış denklemi ise

$$y_{ij} = f(x_{ij}) = \frac{1}{2}(|x_{ij} + 1| - |x_{ij} - 1|) \quad (4.2)$$

olarak verilmektedir. Modelde eğer A ve B operatörleri birbirine eşit ise hücre modeli uzamda değişmez (*space-invariant*) olarak adlandırılır. Eğer $B = 0$ ise, model otonom (*autonomous*) olarak adlandırılır.

HYSA modeli, her hücre sadece bir durum değişkenine sahipse model *single-layer* (tek katmanlı), birden fazla durum değişkenine sahip ise *multi-layer* (çok katmanlı) olarak adlandırılır. Yapılan tanımlamalar ışığında özet olarak HYSA, eşdeğer dinamik hücrelerden oluşan, çok boyutlu, çoğu bağlantısı sınırlı bir r yarıçapında, lokal ve tüm durum değişkenleri sürekli değerlikli sinyal olan bir hücre dizisidir [57].

Araştırmacılar HYSA mimarisi kullanarak farklı operatör ve şablonlar ile *pattern generation* (desen üretimi), *wave propagation* (dalga yayılımı), *spiral waves* (sarmal dalgalar) ve *visual processing* (görsel işleme) [58, 59] uygulamalarının yanı sıra doğrusal olmayan kompleks dinamiklerin benzetimi gibi [60, 61] uygulamalar gerçekleştirmişlerdir. Bu tür uygulamalarda HYSA, kaotik davranış göstermektedir ve bu HYSA mimarisi *SC-CNN State Controlled Cellular Neural/Nonlinear Network* (Durum Kontrollü HYSA, DK-HYSA) olarak adlandırılır.

Denklem (4.1)'de verilen Chua-Yang modelinde her hücrenin durum değişkeni, direkt olarak komşuluğundaki hücrelerin giriş ve çıkışlarına bağlıdır. DK-HYSA modeli ise daha genel bir formda durum değişkenlerine bağlılığı da kapsayacak şekilde

$$\frac{dx_j}{dt} = -x_j + \sum_{C(k) \in S_r(j)} \{A(j;k)y_k + B(j;k)u_k + C(j;k)x_k\} + I_j \quad (4.3)$$

$$y_j = f(x_j) = \frac{1}{2}(|x_j + 1| - |x_j - 1|) \quad (4.4)$$

olarak tanımlanır. Denklemde x_j , y_j , u_j sırasıyla $C(j)$ hücresinin durum değişkeni, çıkış ve girişidir. N hücre sayısı, f çıkışa ait doğrusal olmayan fonksiyon ve $I(j)$ ilgili hücrenin bias değerini temsil etmektedir. $1 \leq j \leq N$ koşulu sağlanmak üzere DK-HYSA modeli her biri birinci mertebeden olmak üzere N hücreli N . mertebeden dinamik bir sisteme karşılık gelir. DK-HYSA'na ait şablonlar (template) uygun seçilerek kaotik davranış sergileyen HYSA modelleri tanımlanabilir [57]. Önerilen iki RSÜ tasarımı da entropi kaynağı olarak kaotik davranış sergileyen DK-HYSA modeli kullanılmaktadır.

4.2. Üç Hücreli Otonom HYSA Modeli ve RSÜ Uygulaması

Zou ve Nossek 1993 yılında yaptıkları bir çalışmada [62] DK-HYSA modelini kullanarak üç otonom hücreli HYSA modelini önermişlerdir. Önerilen modelde Denklem (4.3)'de verilen DK-HYSA modeli için

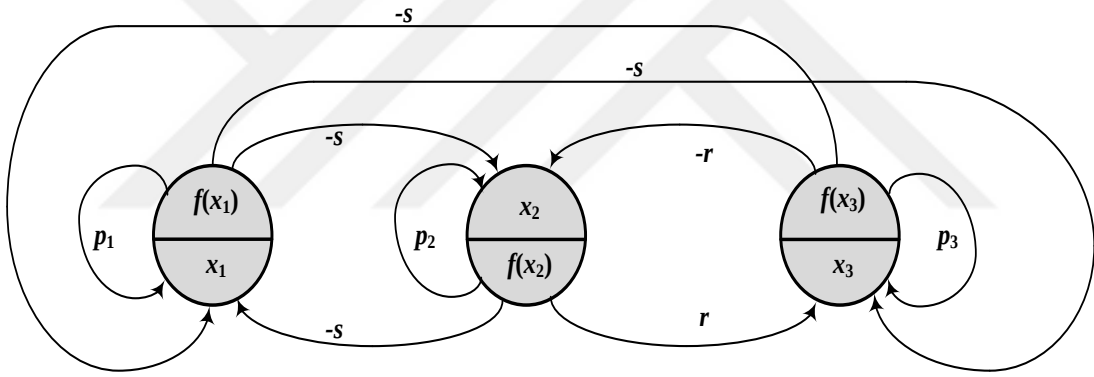
$$I_j = B_j = C_j = 0 \quad 1 \leq j \leq 3 \quad (4.5)$$

$$A_{jk} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} p_1 & -s & -s \\ -s & p_2 & -r \\ -s & r & p_3 \end{bmatrix} \quad (4.6)$$

şablon tanımlamaları yapılarak

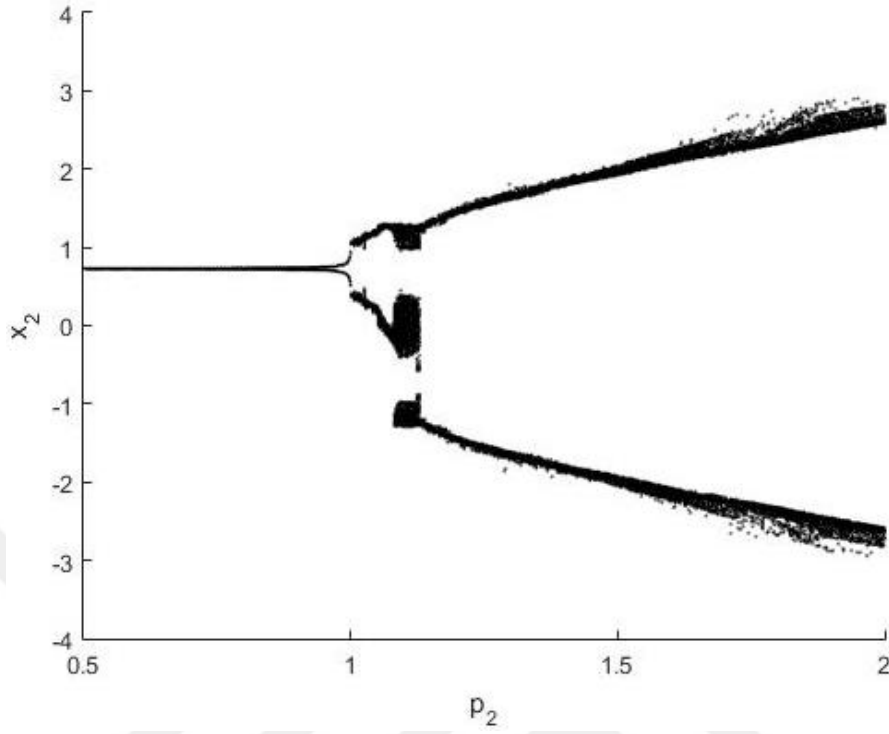
$$\begin{aligned} \dot{x}_1 &= -x_1 + p_1 f(x_1) - sf(x_2) - sf(x_3) \\ \dot{x}_2 &= -x_2 - sf(x_1) - p_2 f(x_2) - rf(x_3) \\ \dot{x}_3 &= -x_3 - sf(x_1) + rf(x_2) - p_3 f(x_3) \end{aligned} \quad (4.7)$$

şeklinde durum değişkenleri tanımlanmış ve Denklem (4.4)'deki çıkış fonksiyonu kullanılmıştır. Modele ait akış diyagramı Şekil 4.2'de gösterilmektedir.

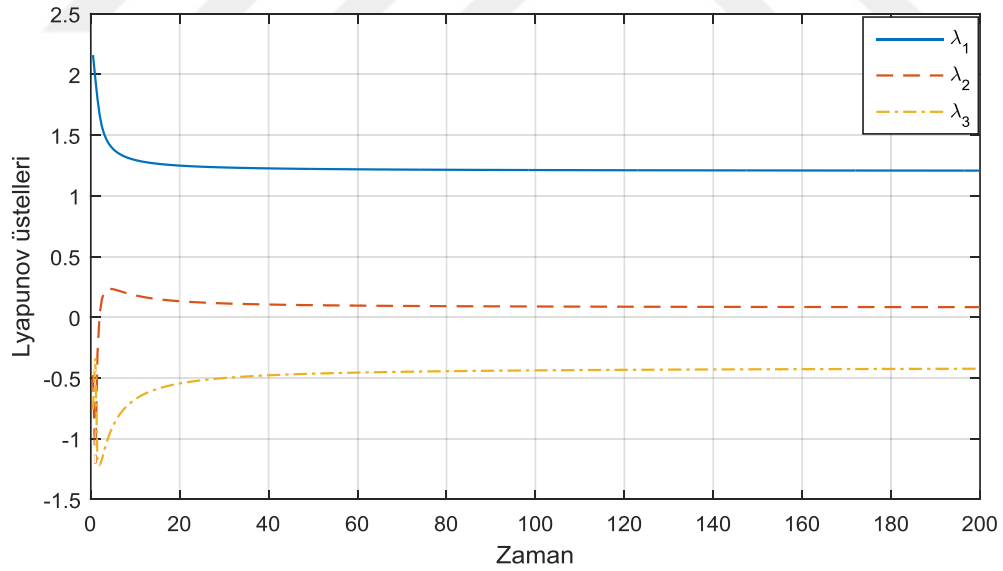


Şekil 4.2. Üç hücreli otonom HYSA modeline ait akış diyagramı

HYSa modeli, $p_1 = 1.25$, $p_2 = 1.1$, $p_3 = 1$, $r = 4.4$, $s = 3.2$ parametre değerleri ve $(x_1(0), x_2(0), x_3(0)) = [0.1, 0.1, 0.1]$ başlangıç değerleri için kaotik davranış göstermektedir. Bu davranışını incelemek için Şekil 4.3'de p_2 parametresine karşılık çatallaşma diyagramı ve Şekil 4.4'de Lyapunov üstellerinin değişimi verilmektedir.



Şekil 4.3. Üç hücreli otonom HYSA modelinin $p_2 \in [0.5,2]$ aralığı için çatallaşma diyagramı



Şekil 4.4. Üç hücreli otonom HYSA modelinin Lyapunov üstellerinin değışimi

Üç hücreli otonom HYSA modelinin üç Lyapunov üsteli bulunmaktadır. Bu üstel değeri yaklaşık olarak $\lambda_1 = 1.208$, $\lambda_2 = 0.083$ ve $\lambda_3 = -0.424$ şeklinde elde edilmiştir.

Önerilen RSÜ uygulamasında [63], üç hücreli otonom HYSA modeli ayrık zamanlı olarak modellenip x_1 durum değışkeni entropi kaynağı olarak kullanılmaktadır. Ayrıca

tasarımda Trivium algoritması son işlem olarak yer alıp rasgele sayı üretimi üç farklı senaryo için gerçekleştirilmektedir.

4.2.1. Üç Hücreli Otonom HYSA Modelinin Donanım Tabanlı Benzetimi

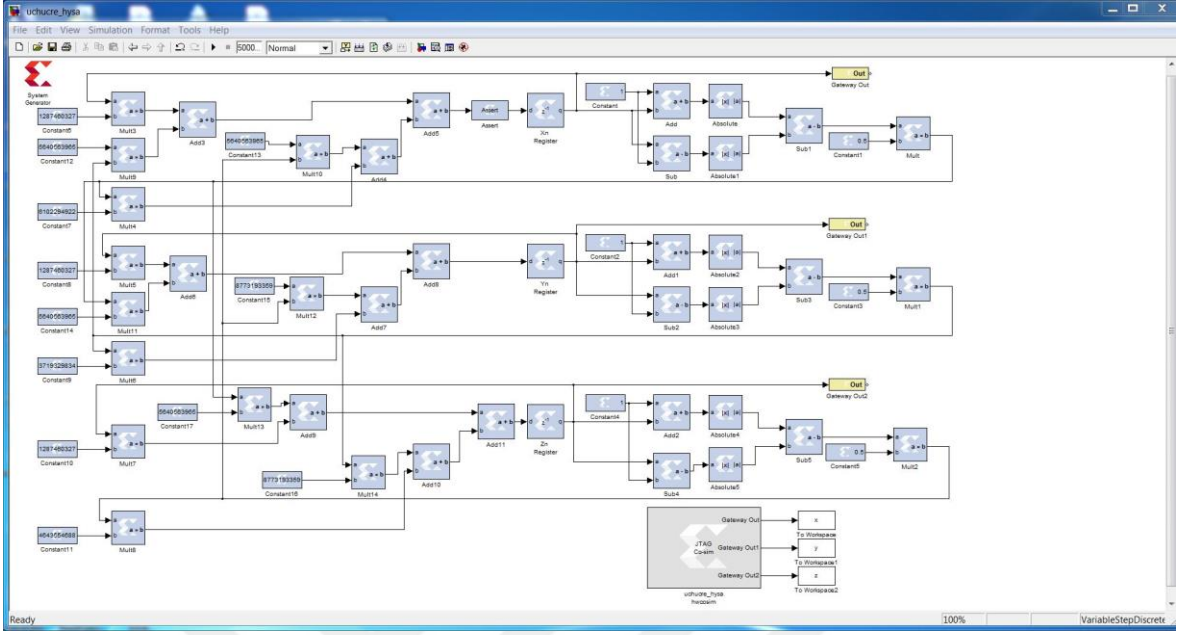
Önerilen RSÜ tasarımı için kullanılan HYSA modelinin uygulanabilirliğini test etmek ve davranışını analiz etmek amacıyla Denklem (4.7)'de verilen üç hücreli otonom HYSA modelinin ayrık zamanlı olarak XSG platformunda donanım tabanlı benzetimi gerçekleştirilecektir. Model İleri Euler yöntemi ile ayrıklaştırılırsa;

$$\begin{aligned} x_1[n+1] &= \frac{1}{a} \left[\left(\frac{x_1[n]}{h} \right) + p_1 f(x_1[n]) - sf(x_2[n]) - sf(x_3[n]) \right] \\ x_2[n+1] &= \frac{1}{a} \left[\left(\frac{x_2[n]}{h} \right) - sf(x_1[n]) + p_2 f(x_2[n]) - rf(x_3[n]) \right] \\ x_3[n+1] &= \frac{1}{a} \left[\left(\frac{x_3[n]}{h} \right) - sf(x_1[n]) + rf(x_2[n]) + p_3 f(x_3[n]) \right] \end{aligned} \quad (4.8)$$

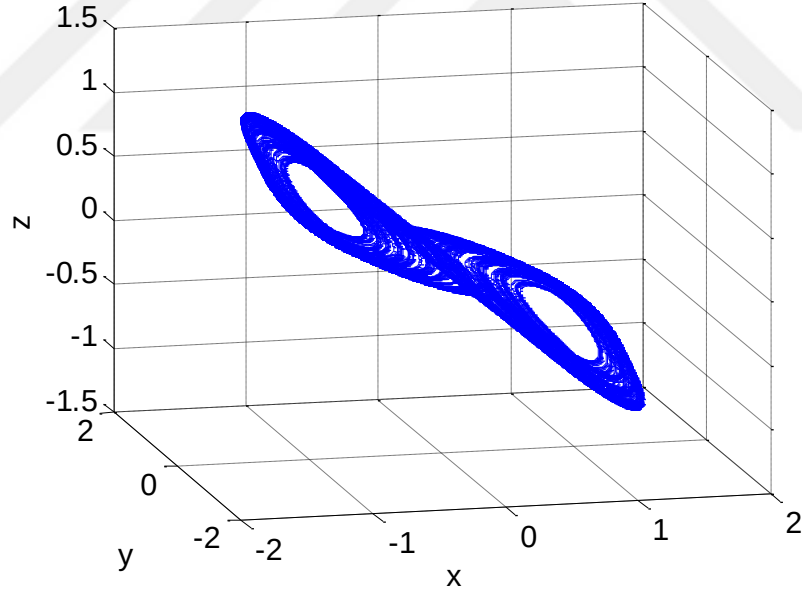
denklem takımı elde edilir. Doğrusal olmayan çıkış fonksiyonu ise

$$f(x_j[n]) = \frac{1}{2} \left(|x_j[n] + 1| - |x_j[n] - 1| \right) \quad j = 1, 2, 3 \quad (4.9)$$

olarak ayrıklaştırılır. Ayrık zamanlı HYSA modelinin, $p_1 = 1.25$, $p_2 = 1.1$, $p_3 = 1$, $r = 4.4$, $s = 3.2$ parametre ve $(x_1(0), x_2(0), x_3(0)) = [0.1, 0.1, 0.1]$ başlangıç değerleri, $h = 0.001$ adım aralığı, $a = 1 + 1/h = 1001$ örnekleme parametresi ve $n = 500000$ iterasyon için Şekil 4.5'de verilen XSG modeli ile donanım tabanlı benzetimi gerçekleştirilmiştir. XSG modelinin benzetim sonucunda elde edilen üç boyutlu yörüngenin değişimi Şekil 4.6'da gösterilmektedir.

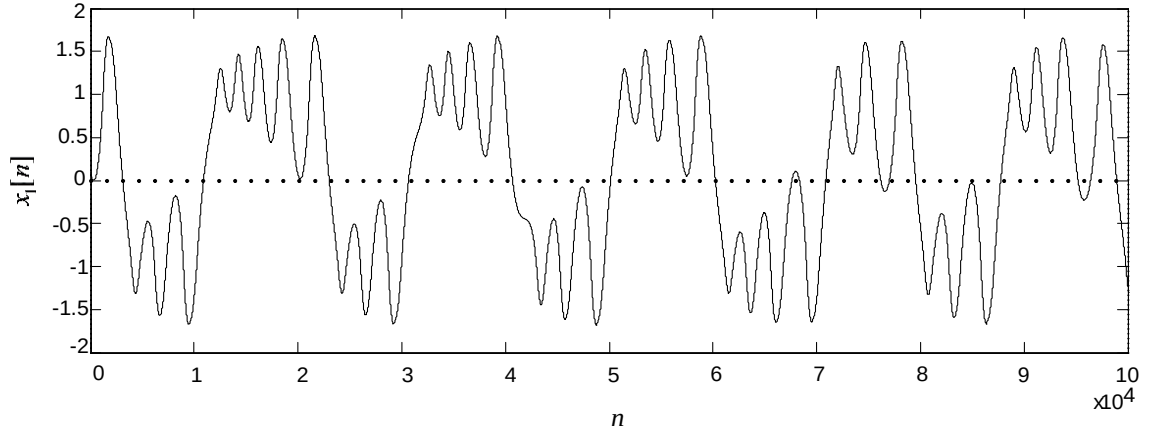


Şekil 4.5. Üç hücreli otonom HYSA'nın XSG modeli



Şekil 4.6. Üç hücreli otonom HYSA'nın XSG modeline ait üç boyutlu yörüngesi

Önerilen RSÜ tasarımında $x_1[n]$ durum değişkeni kullanılacağından, $x_1[n]$ durum değişkeninin değişimi Şekil 4.7'de gösterilmektedir.



Şekil 4.7. Üç hücreli otonom HYSA modelinin $x_1[n]$ durum değişkeninin değişimi

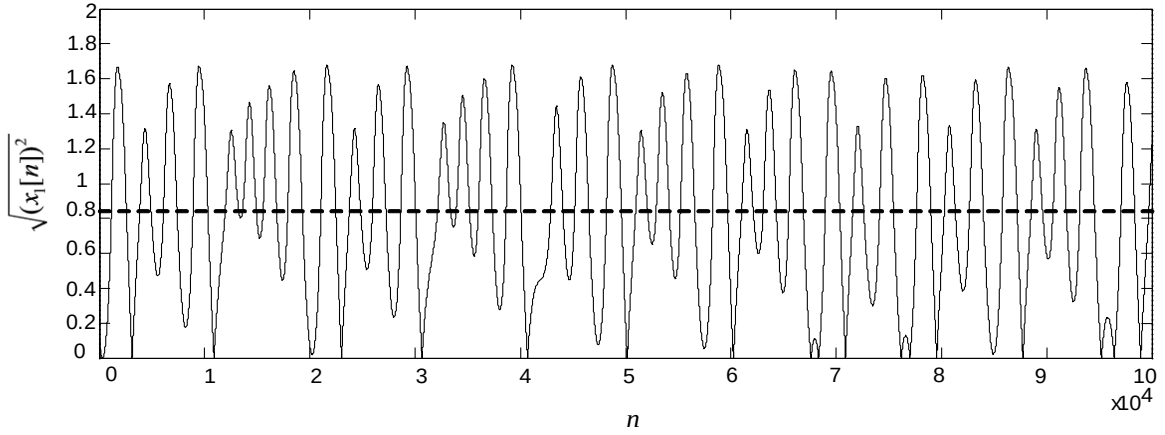
Şekil 4.6 ve 4.7’den de görüldüğü üzere üç hücreli otonom HYSA modelinin üç boyutlu yörüngesi kaotik çeker davranışı göstermekte ve modelin ayrık zamanlı $x_1[n]$ durum değişkeni osilasyon yapmaktadır. Üç hücreli otonom HYSA’nın XSG ortamında donanım tabanlı benzetimi Tablo 4.1’de verilen kaynak kullanımı ile başarılı bir şekilde gerçekleştirilmiştir.

Tablo 4.1. Üç hücreli otonom HYSA’nın XSG modelinin FPGA kaynak kullanımı

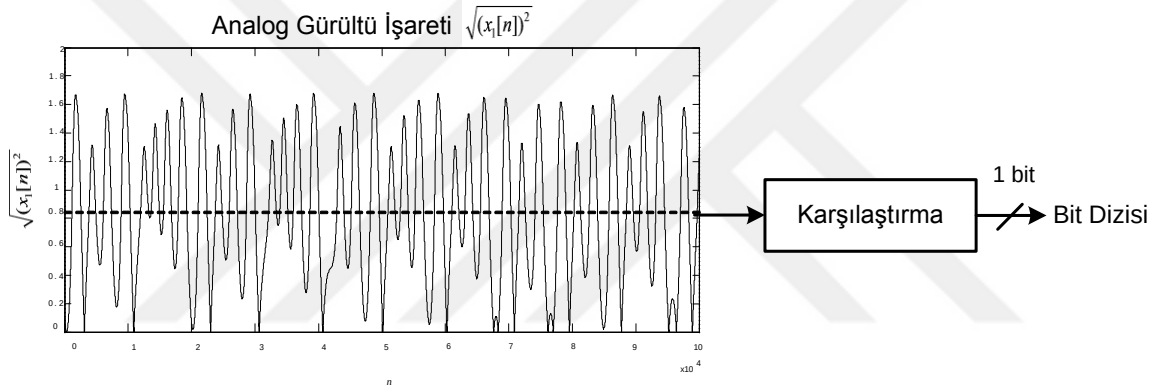
Kaynak Adı	Kullanım Sayısı
FF	486
LUT	1200

4.2.2. Üç Hücreli Otonom HYSA Modeli ile FPGA Ortamında RSÜ Tasarımı

Önerilen RSÜ tasarımında ilk senaryo için entropi kaynağı olarak Şekil 4.7’de verilen üç hücreli otonom HYSA modelinin $x_1[n]$ ayrık zamanlı durum değişkeni kullanılmaktadır. Şekilden de görüleceği üzere durum değişkeni hem pozitif hem de negatif değerler almaktadır. Bu nedenle $x_1[n]$ ayrık zamanlı durum değişkeninin her iterasyonda karesel ortalama değeri hesaplanır. Şekil 4.8’de $x_1[n]$ ayrık zamanlı durum değişkeninin karesel ortalama değerinin değişimi verilmektedir. Bu değişimin ortalama değeri ise 0.83935 olarak hesaplanmış ve şekilde gösterilmiştir. Elde edilen bu değer, durum değişkeninin sayısallaştırılması için eşik değeri olarak kullanılmaktadır. İlk senaryoya ait gösterim Şekil 4.9’da verilmektedir.



Şekil 4.8. $x_1[n]$ durum değişkeninin karesel ortalama değerinin değişimi

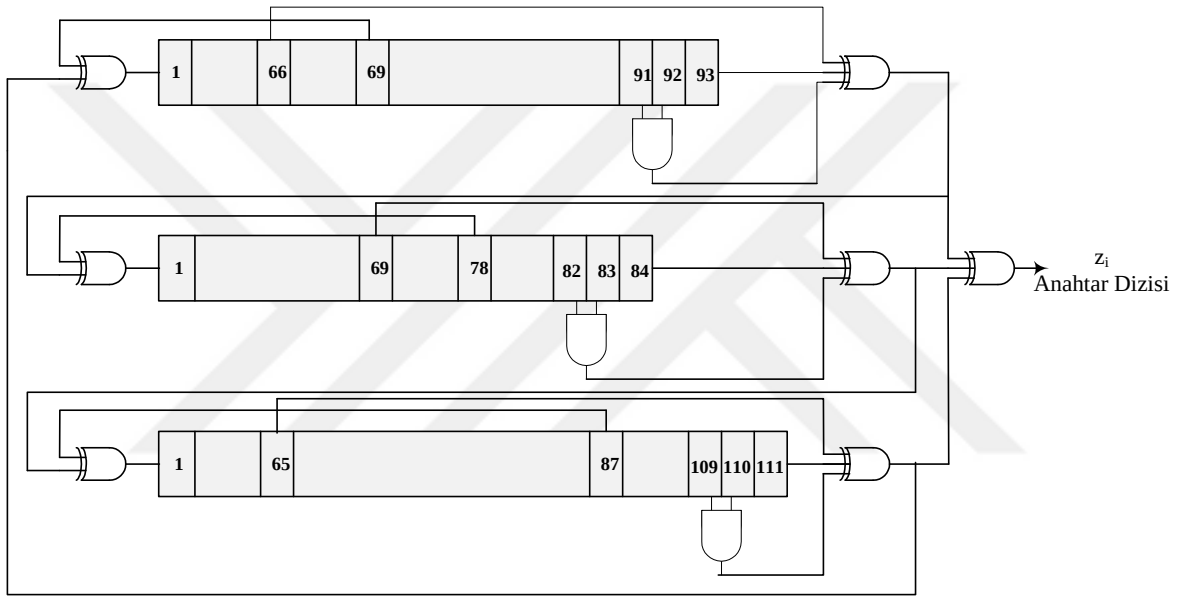


Şekil 4.9. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait ilk senaryo

Şekil 4.9'dan da görüldüğü gibi RSÜ tasarımında entropi kaynağı olarak üç hücreli otonom HYSA modeline ait Şekil 4.8'de verilen $x_1[n]$ ayrık durum değişkeninin karesel ortalaması alınmış analog işaret kullanılmıştır. Analog işaret, belirlenen eşik değeri (0.83935) ile karşılaştırılıp sayısallaştırılır. Bu işlem her iterasyon için tekrarlanarak istenilen uzunlukta veri elde edilebilir.

İkinci senaryoda ise ilk senaryonun çıkışı kullanılarak Trivium şifreleme algoritmasının girişine uygulanır. Trivium şifreleme algoritması [11], donanım bazlı senkron bir dizi şifreleme sistemidir. Senkron olmasının sebebi, anahtar dizisinin açık metinden bağımsız olarak üretilmesidir. Trivium algoritmasının üretilme amacı hız ve alan arasında esnek bir optimizasyon sağlamaktır. Trivium, güvenlik, hız ve esneklikten fedakârlık etmeden, bir dizi şifreleme sisteminin ne kadar basitleştirileceğini araştırmak için tasarlanmıştır.

Trivium dizi şifreleme algoritması, 80-bit gizli anahtar (tohum) ve 80-bit başlangıç değeri (*Initialization Value*, IV) olmak üzere 288 bitlik iç durum dizisi kullanarak 2^{64} bite kadar anahtar dizisi üretebilen senkron bir dizi şifreleme sistemidir. Trivium iki aşamadan oluşur: İlk olarak gizli anahtar K ve başlangıç değeri IV kullanılarak algoritma başlangıç durumuna getirilir, daha sonra başlangıç durumu tekrar güncellenerek anahtar bitleri üretilir. Algoritmaya ait kapı elemanları düzeyinde bir gösterim Şekil 4.10'da verilmektedir.



Şekil 4.10. Trivium algoritmasının kapı elemanları düzeyinde gösterimi

Trivium dizi şifreleme algoritmasında ilk olarak 288-bitlik iç durum (internal state, s) bilgisi oluşturulur. 288-bitlik iç durum 80-bitlik gizli anahtarın, 80-bitlik ilk değerin (IV) yüklenmesi ve s_{286} , s_{287} ve s_{288} hariç diğer bitlerin '0' atanması ile başlangıç konumuna getirilmiş olur. Daha sonra 288-bitlik iç durum 4 tam tur boyunca döndürülür. Başlangıç koşuluna getirilme ve 4 tur döndürme aşağıdaki gibi matematiksel olarak ifade edilebilir;

$$\begin{aligned}
 (s_1, s_2, \dots, s_{93}) &\leftarrow (K_1, \dots, K_{80}, 0, \dots, 0) \\
 (s_{94}, s_{95}, \dots, s_{177}) &\leftarrow (IV_1, \dots, IV_{80}, 0, \dots, 0) \\
 (s_{178}, s_{179}, \dots, s_{288}) &\leftarrow (0, \dots, 0, 1, 1, 1) \\
 \text{for } i=1 \text{ to } 4 \times 288 \text{ do}
 \end{aligned}$$

$$\begin{aligned}
t_1 &\leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171} \\
t_2 &\leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264} \\
t_3 &\leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69} \\
(s_1, s_2, \dots, s_{93}) &\leftarrow (t_3, s_1, \dots, s_{92}) \\
(s_{94}, s_{95}, \dots, s_{177}) &\leftarrow (t_1, s_{94}, \dots, s_{176}) \\
(s_{178}, s_{179}, \dots, s_{288}) &\leftarrow (t_2, s_{178}, \dots, s_{287})
\end{aligned} \tag{4.10}$$

end for

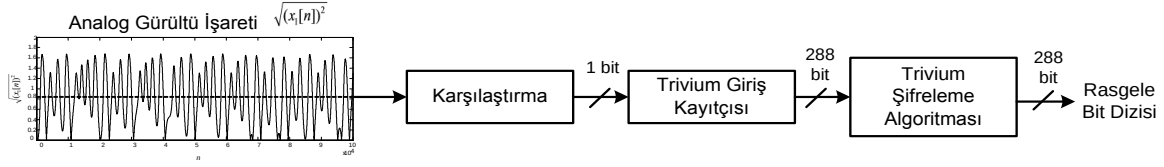
denklemde kullanılan K ve IV sırasıyla 80 bit uzunluğundaki gizli anahtar ve başlangıç değeridir. t_1 , t_2 ve t_3 değişkenleri ise geçici değerleri tutan kayıtlılardır.

Anahtar dizisi üretilme işlemi ise 15 özel bitin ayrılması ve bu bitlerin hepsinin 3 bitin güncellenmesinde kullanılmasının sonucunda 1 bit anahtar dizisi z_i 'nin hesaplanması işlemlerini içerir. Daha sonra durum bitleri kaydırılarak döndürülür ve istenen bit uzunluğunda anahtar dizisi üretilene kadar işlem tekrarlanır. Algoritmanın bu safhasının matematiksel tanımını aşağıdaki işlemlerle ifade edilebilir;

$$\begin{aligned}
&\textbf{for } i=1 \text{ to } N \text{ do} \\
&\quad t_1 \leftarrow s_{66} + s_{93} \\
&\quad t_2 \leftarrow s_{162} + s_{177} \\
&\quad t_3 \leftarrow s_{243} + s_{288} \\
&\quad z_i \leftarrow t_1 + t_2 + t_3 \\
&\quad t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171} \\
&\quad t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264} \\
&\quad t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69} \\
&\quad (s_1, s_2, \dots, s_{93}) \leftarrow (t_3, s_1, \dots, s_{92}) \\
&\quad (s_{94}, s_{95}, \dots, s_{177}) \leftarrow (t_1, s_{94}, \dots, s_{176}) \\
&\quad (s_{178}, s_{179}, \dots, s_{288}) \leftarrow (t_2, s_{178}, \dots, s_{287}) \\
&\textbf{end for}
\end{aligned} \tag{4.11}$$

Modülo-2'ye göre yazılan yukarıdaki algoritmada N üretilmek istenen bit sayısını temsil ederken (+) işlemi XOR ve (.) işlemi AND mantıksal operatörleri temsil etmektedir. Ayrıca s_1 en düşük değerlikli biti, s_{288} ise en yüksek anlamlı biti göstermektedir.

Üç hücreli otonom HYSA modeli kullanılarak ikinci senaryo için RSÜ tasarımı yapılmıştır. Tasarıma ait gösterim Şekil 4.11'de verilmektedir.



Şekil 4.11. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait ikinci senaryo

İkinci senaryoda, ilk senaryoda elde edilen 1 bitlik veri, Trivium algoritmasında tanımlanan iç durum dizisini oluşturmak için kullanılmaktadır. 288 bitlik iç durum dizisi oluşturulduktan sonra Trivium algoritmasına ait aşağıdaki işlemler yapılarak 288 bitlik iç durum dizisi girişine karşın yine 288 bitlik rasgele bit dizisi üretilir.

for i=1 to 288 **do**

$$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$$

$$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$$

$$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$$

$$(s_1, s_2, \dots, s_{93}) \leftarrow (t_3, s_1, \dots, s_{92})$$

$$(s_{94}, s_{95}, \dots, s_{177}) \leftarrow (t_1, s_{94}, \dots, s_{176})$$

$$(s_{178}, s_{179}, \dots, s_{288}) \leftarrow (t_2, s_{178}, \dots, s_{287})$$

(4.12)

end for

for i=1 to 288 **do**

$$t_1 \leftarrow s_{66} + s_{93}$$

$$t_2 \leftarrow s_{162} + s_{177}$$

$$t_3 \leftarrow s_{243} + s_{288}$$

$$z_i \leftarrow t_1 + t_2 + t_3$$

$$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$$

$$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$$

$$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$$

$$(s_1, s_2, \dots, s_{93}) \leftarrow (t_3, s_1, \dots, s_{92})$$

$$(s_{94}, s_{95}, \dots, s_{177}) \leftarrow (t_1, s_{94}, \dots, s_{176})$$

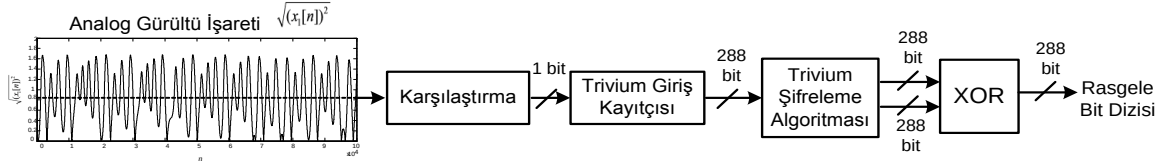
$$(s_{178}, s_{179}, \dots, s_{288}) \leftarrow (t_2, s_{178}, \dots, s_{287})$$

(4.13)

end for

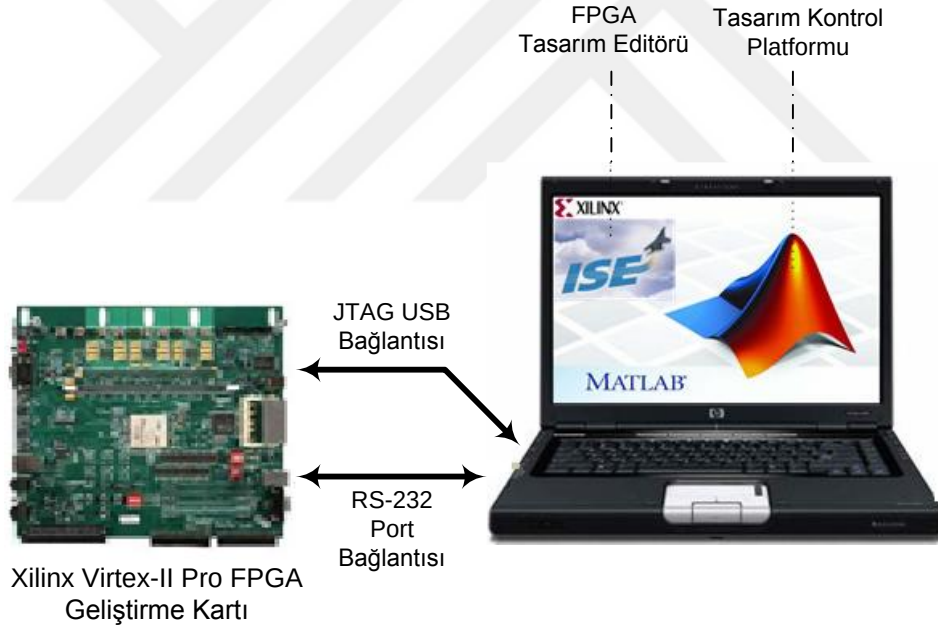
Üçüncü senaryoda ise, ikinci senaryodaki gibi 288 bitlik iç durum dizisi oluşturulduktan sonra Trivium algoritmasına ait sadece Denklem (4.13)'de verilen

algoritmik işlemler gerçekleştirilir. Fakat bu işlem Denklem (4.13)'den farklı olarak 576 bit üretimi yapılarına kadar devam edilir. Elde edilen bit dizisi ikiye ayrılarak XOR işlemine tabi tutularak sonuçta yine 288 bitlik dizi üretilir. Tasarıma ait gösterim Şekil 4.12'de verilmektedir.



Şekil 4.12. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait üçüncü senaryo

Üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait tüm senaryolar için Şekil 4.13'de verilen platform gerçek zamanlı olarak çalıştırılmıştır.

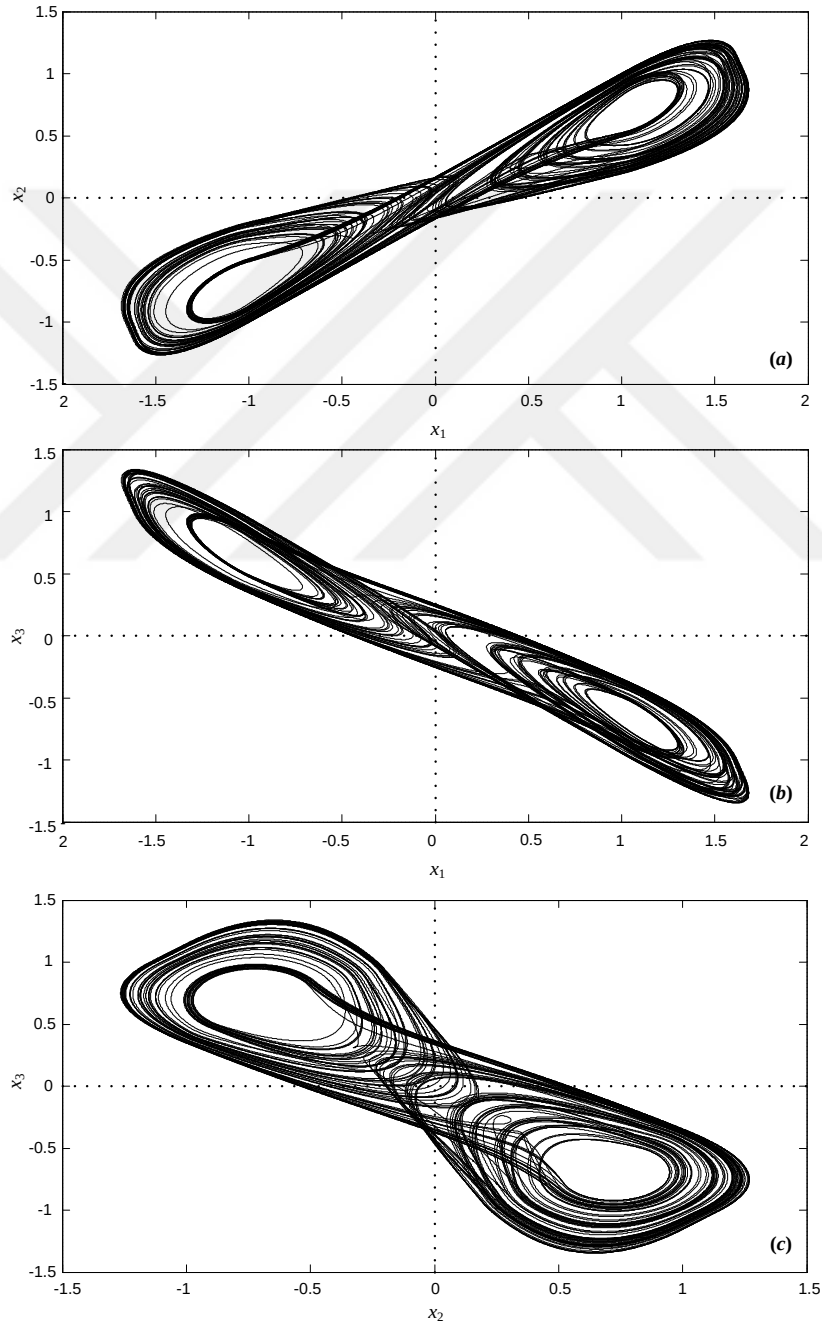


Şekil 4.13. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait uygulama platformu

Senaryolara ait tasarımlar Xilinx ISE 9.2i FPGA editöründe Verilog HDL ile kodlanmıştır. Tasarımlar Xilinx Virtex-II Pro FPGA geliştirme kartında yer alan XC2VP30FF896 FPGA çipinde gerçek zamanlı olarak gerçekleştirilmiştir. Uygulama platformunda MATLAB programı kontrol ve çıkış platformu olarak kullanılmıştır. Tasarım sonucu elde edilen çıkışlar Xilinx IP Core Generator tarafından üretilen Block RAM

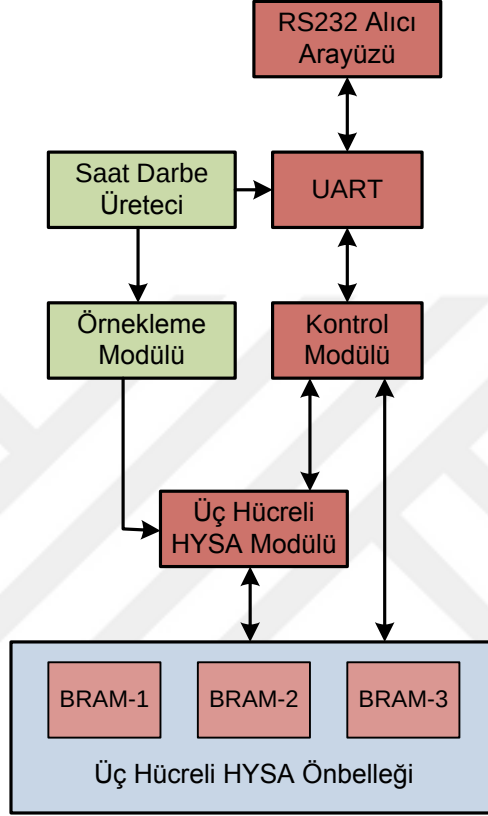
(BRAM) modülleri aracılığıyla MATLAB platformuna aktarılmıştır. Tanımlanan BRAM modülleri her biri 10 KB kapasiteye sahip tek port (*single-port*) yapısındadır.

İlk senaryo ile HYSA modelindeki üç durum değişkenine ait değerler üretilen 3 BRAM modülüne kaydedilir. MATLAB platformuna aktarılan değerlere göre HYSA modeline ait faz portreleri Şekil 4.14'deki gibidir.



Şekil 4.14. Üç hücreli otonom HYSA modelinin FPGA gerçekleştirim sonucu a) $x_1 - x_2$ b) $x_1 - x_3$ c) $x_2 - x_3$ faz portreleri

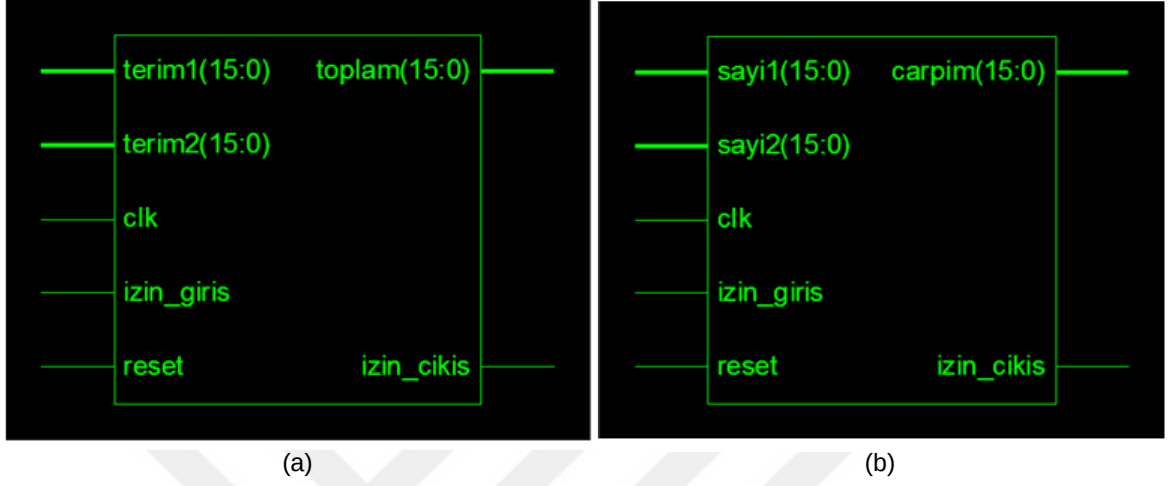
İlk senaryo tekrar çalıştırılarak bu kez 1001600 bitlik veri dizisi üretilir ve BRAM'e kaydedilir. Tüm senaryolar için oluşturulan tasarıma ait Verilog kod modüllerinin blok gösterimi Şekil 4.15'de gösterilmektedir.



Şekil 4.15. Üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait blok gösterimi

Blok gösteriminde yer alan örnekleme modülü, üç hücreli otonom HYSA modülünün çıkışlarını örnekleme için gerekli sayıcı devresini içermektedir. Sayıcı devresi, klasik *for* döngüsü işlevini yerine getirmek için kodlanmıştır. Kontrol modülü ise HYSA modülünün çıkışlarının BRAM'e kaydedilmesi ve/veya adreslenerek UART modülüne iletilmesini sağlar. İkinci ve üçüncü senaryo için kullanılan Trivium algoritması, Üç Hücreli HYSA modülünün içerisinde aritmetik işlemler olarak kodlanmıştır. Üç hücreli otonom HYSA modelinin gerçekleştirimi için gerekli toplama, çıkarma ve çarpma gibi aritmetik işlemler için Q2.14 16 bitlik 2'ye tımlayan sabit noktalı sayı formatı kullanılmıştır. Çıkarma işlemi, çıkarılacak sayının negatif işarete çevrilerek toplama işlemi uygulanmasına dayalı olarak gerçekleştirilmiştir. Toplama ve çarpma modüllerine ait blok gösterimi Şekil 4.16'da verilmektedir. Toplama ve çarpma modülleri girişine verilen sabit noktalı sayıları 2 saat

darbe işareti (clock-pulse) süresi sonunda çıkış vermektedir. Aritmetik işlem modüllerinin FPGA kaynak kullanımları ise Tablo 4.2’de verilmiştir [64].



Şekil 4.16. Tasarlanan aritmetik işlem modülleri a) toplama b) çarpma modülü

Tablo 4.2 Tasarlanan aritmetik işlem modüllerinin FPGA kaynak kullanımı

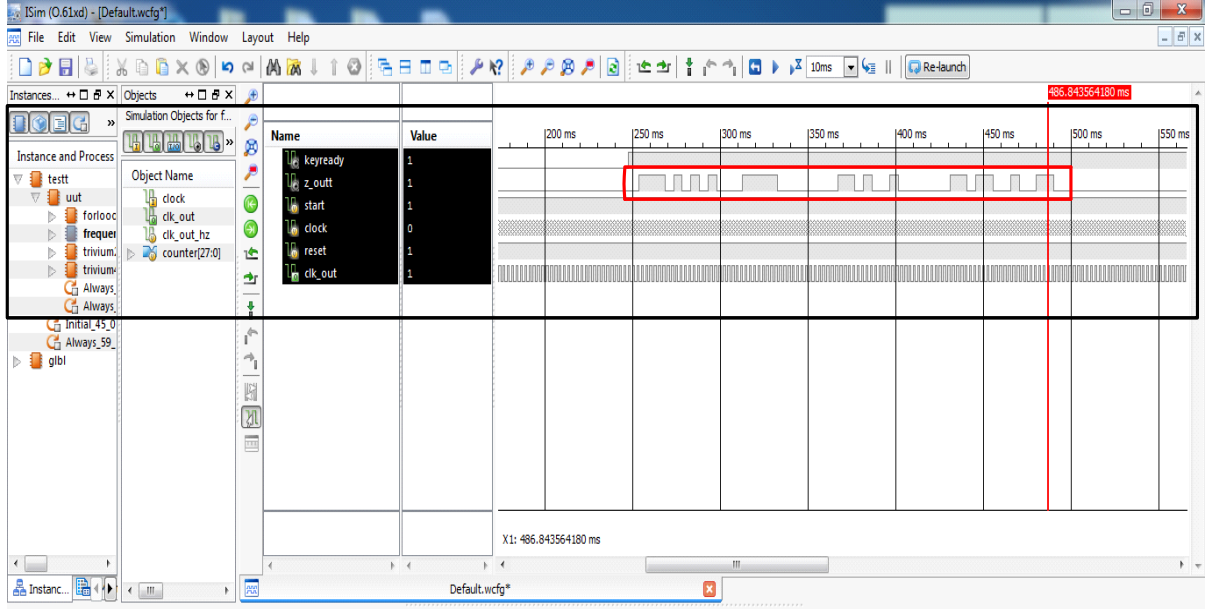
Aritmetik İşlem Modülü	FF	LUT
Toplama modülü	1	18
Çarpma modülü	1	17

İkinci ve üçüncü senaryo için tasarıma Trivium algoritmasına ait Verilog modülleri eklenmiştir. İç durum dizisini (s) oluşturmak için ana modülde counter (sayıcı) kullanılarak döngü işlemi oluşturulur. Tasarımlara ait ilgili aritmetik işlemler gerçekleştirilerek 1001600 bitlik veri dizisi BRAM’e kaydedilir ve bit dizisi MATLAB platformuna aktarılır. Üçüncü senaryo için son işlem uygulanmış önerilen RSÜ tasarımının FPGA kaynak kullanımı Tablo 4.3’de gösterilmektedir.

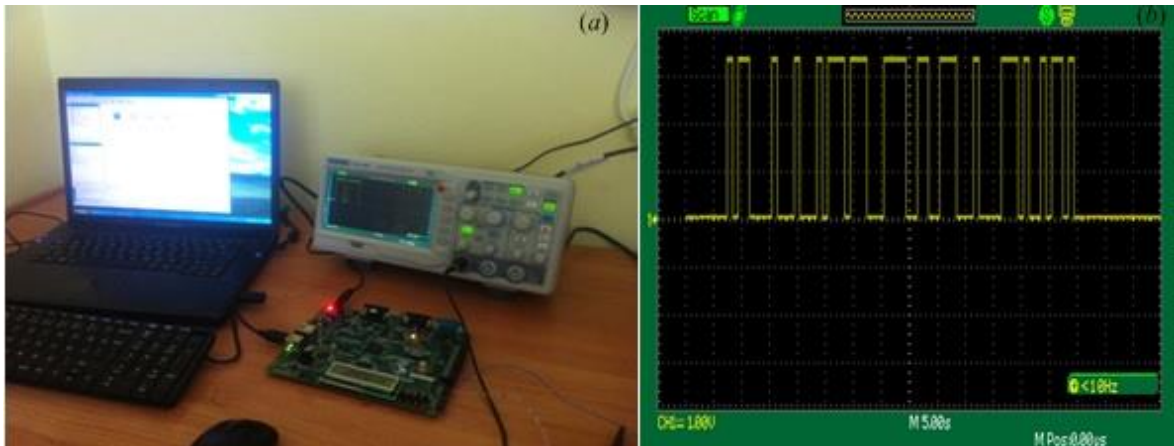
Tablo 4.3. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait üçüncü senaryonun FPGA kaynak kullanımı

Kaynak Adı	Kullanım Sayısı
FF	5069
LUT	13519

Üçüncü senaryoya ait tasarımın FPGA editöründe gerçekleştirilen benzetimi sonucu Şekil 4.17’de, uygulamanın gerçekleştirildiği platform ve üretilen bitlerin 50 bitlik kısmına ait osiloskop görüntüsü Şekil 4.18’de verilmektedir.



Şekil 4.17. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımına ait üçüncü senaryonun benzetim sonucu



Şekil 4.18. a) Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımını gerçekleştirme platformu b) üçüncü senaryoya ait üretilen rasgele bitlerin osiloskop görüntüsü

Kullanılan FPGA geliştirme kartında 50 MHz frekansına sahip kristal osilatörler tarafından üretilen saat darbe işareti (clock pulse) bulunmaktadır. Üçüncü senaryoya ait tasarımın sentezleme sonucuna göre tasarım maksimum 16.124 MHz frekansında

gerçekleştirilebilir. Aritmetik toplama ve çarpma işlemlerinde belirtilen gecikmelerden kaynaklı olarak bit üretimi 2.015 Mbps hızında gerçekleştirilmiştir. Tablo 4.4’de üç senaryo için üretilen 1001600 bit uzunluğundaki veri dizilerinin NIST istatistiksel test sonuçları yer almaktadır.

Tablo 4.4. Önerilen üç hücreli otonom HYSA tabanlı RSÜ tasarımlarına ait NIST istatistiksel test sonuçları

NIST Testleri (1001600 bit)	İlk senaryo	Sonuç	İkinci senaryo	Sonuç	Üçüncü Senaryo	Sonuç
Frekans	0.865	Başarılı	0.861	Başarılı	0.342	Başarılı
Bir blok içindeki frekans	-	Başarısız	0.099	Başarılı	0.25	Başarılı
Akış	-	Başarısız	0.373	Başarılı	0.568	Başarılı
Bir blok içinde en uzun birlerin akış	-	Başarısız	0.474	Başarılı	0.026	Başarılı
İkili matris rankı	-	Başarısız	0.028	Başarılı	0.756	Başarılı
Ayrık Fourier dönüşümü	-	Başarısız	0.856	Başarılı	0.452	Başarılı
Örtüşmeyen şablon eşleştirme	-	Başarısız	0.575	Başarılı	0.685	Başarılı
Örtüşen şablon eşleştirme	-	Başarısız	0.26	Başarılı	0.401	Başarılı
Maurer evrensel istatistik	-	Başarısız	0.25	Başarılı	0.605	Başarılı
Doğrusal karmaşıklık	0.197	Başarılı	0.314	Başarılı	0.192	Başarılı
Seri (1/2)	-	Başarısız	0.525/0.734	Başarılı	0.302/0.202	Başarılı
Lempel Ziv	-	Başarısız	71253/2/1	Başarılı	74356/2/1	Başarılı
Yaklaşık entropi	-	Başarısız	0.308	Başarılı	0.527	Başarılı
Kümülatif toplamlar	0.695	Başarılı	0.935	Başarılı	0.542	Başarılı
Rasgele gezinimler (ortalama)	-	Başarısız	0.782	Başarılı	0.822	Başarılı
Rasgele gezinim değişkenler (ortalama)	-	Başarısız	0.126	Başarılı	0.038	Başarılı

Üretilen rasgele bit dizisi, NIST istatistiksel testlerinin yanı sıra ölçek indeksi (*scale index*) tekniği kullanılarak analiz edilebilir. Bu teknik, bit dizisinin periyodikliği hakkında bilgi verir.

Ölçek indeksi tekniği [65], sayısal işaretlerin periyodiklik derecesini ölçekleyen bir algoritmadır. Üretilen rasgele bit dizisi sayısal bir işaret olarak düşünülerek *scalogram* (ölçekleme) algoritması ile analiz edilebilir. Verilen bir f fonksiyonu için ölçekleme işlemi;

$$S(s) = \left(\int_{-\infty}^{+\infty} |Wf(u, s)|^2 du \right)^{1/2} \quad (4.14)$$

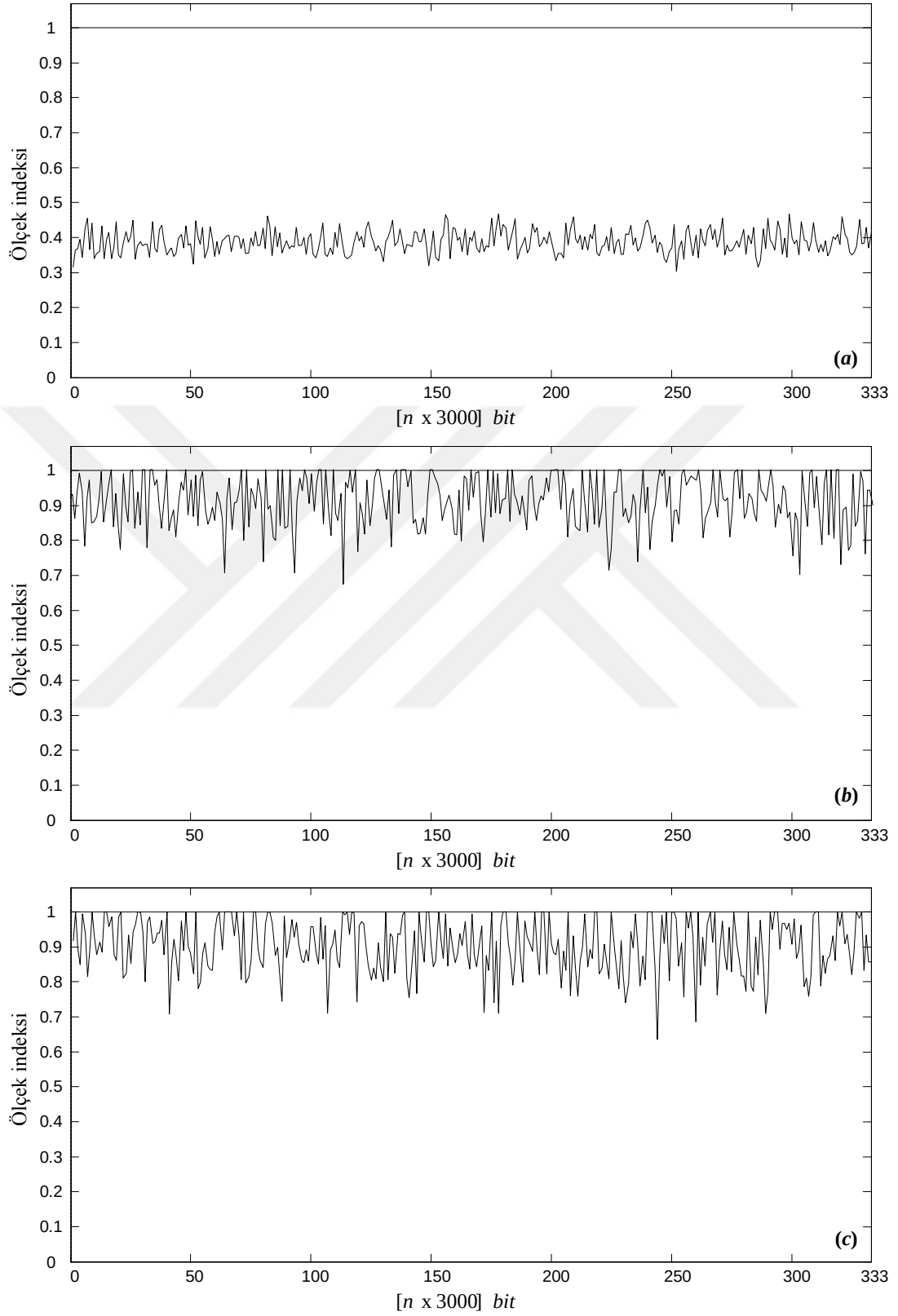
denklemleri ile tanımlanır. Denklemlerde $Wf(u, s)$, u anında s ölçeği ile f fonksiyonunun wavelet (dalgacık) dönüşümünü ifade etmektedir. Ölçekleme sonucunda elde edilen indeks ise

$$i_s = \frac{S(s_{\min})}{S(s_{\max})} \quad (4.15)$$

denklemleri ile hesaplanır. İndeks değerinin değişim aralığı $[0,1]$ 'dir. Herhangi bir fonksiyon için elde edilen indeks değeri ne kadar 1'e yakın ise, fonksiyon o kadar aperiyoiktir, denir.

Üç hücreli otonom HYSA modeli kullanılarak üç senaryo için üretilen 1001600 bitlik diziler, 3000 bit uzunluğunda parçalara ayrılarak ölçek indeksi hesaplanmıştır. Elde edilen ölçek indekslerinin değişimi Şekil 4.19'daki gibidir.

İlk senaryo ile elde edilen bit dizisi sadece 3 istatistiksel testten geçerek ortalama olarak 0.475 ölçek indeksi değerine sahiptir. İkinci ve üçüncü senaryolar, tüm istatistiksel testlerden başarılı bir şekilde geçerek sırasıyla 0.9107 ve 0.9027 ortalama ölçek indeks değerlerine sahiptir. Sonuç olarak sadece HYSA modelinin çıktısı kullanılarak, güvenli derecede rasgele bit üretimi sağlanamamıştır. HYSA modeli ile birlikte Trivium algoritması kullanıldığında veya model ile birlikte Trivium algoritmasının sadece bir kısmı kullanılarak XOR son işleminin uygulanması ile periyodik olmayan değişime sahip, istatistiksel olarak rasgele bit üretimi gerçekleştirilmiştir.



Şekil 4.19. Üç hücreli otonom HYSA modeli ile tasarlanan RSÜ çıktıların ölçek indekslerinin değişimi a) ilk b) ikinci ve c) üçüncü senaryo için sonuçlar

4.3. Memristörlü HYSA Modeli ve RSÜ Uygulaması

Memristörlü HYSA modeli, Leon O. Chua tarafından önerilmiş memristör [47], ilk kaotik elektronik devre [66] ve Hücrel Doğrusal Olmayan Ağlar (CNN) [54] çalışmaları arasında bağlantı kurularak literatürde yer almış kaotik davranış sergileyen bir modeldir. Memristörlü HYSA modeli, 2012 yılında yapılan bir çalışmada [67] detaylı olarak incelenmiş ve modelin devre gerçekleştirilmesi yapılmıştır.

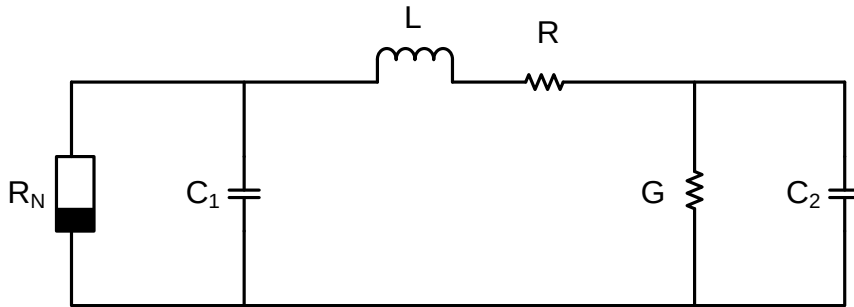
Chua'nın tanımlamalarına göre, pasif bir iki-uçlu devre elemanı olan memristör Denklem (4.16)'daki gibi doğrusal olmayan akım-gerilim (i - v) karakteristiği ile ifade edilir:

$$\begin{aligned} v &= M(q)i \\ i &= W(\varphi)v \end{aligned} \quad (4.16)$$

denklemlerde v , i , q , φ sırasıyla memristöre ait gerilim, akım, yük ve akıyı temsil ederken, Denklem (3.19) ve (3.20)'de verilen $M(q)$ memristansı ve $W(\varphi)$ memdüktaansı ifade etmektedir. Hem yük hem de akı kontrollü memristör için sırasıyla $\varphi(q)$ ve $q(\varphi)$ Denklem (4.17)'deki gibi parça-parça doğrusal (piece-wise linear, PWL) fonksiyon olarak tanımlanır.

$$\begin{aligned} \varphi(q) &= bq + 0.5(a-b)(|q+1| - |q-1|) \\ q(\varphi) &= d\varphi + 0.5(c-d)(|\varphi+1| - |\varphi-1|) \end{aligned} \quad (4.17)$$

Denklemlerde verilen her parametre çiftinden ($a-b$ ve $c-d$) en az biri negatif değerlikli ise bu durumda memristör aktiftir. Aksi halde memristör pasif olarak nitelendirilir [68]. Kanonik Chua osilatör devresi orijinal olarak 1990 yılında literatüre sunulmuş olup [69], devre şeması Şekil 4.20'de verilmektedir.

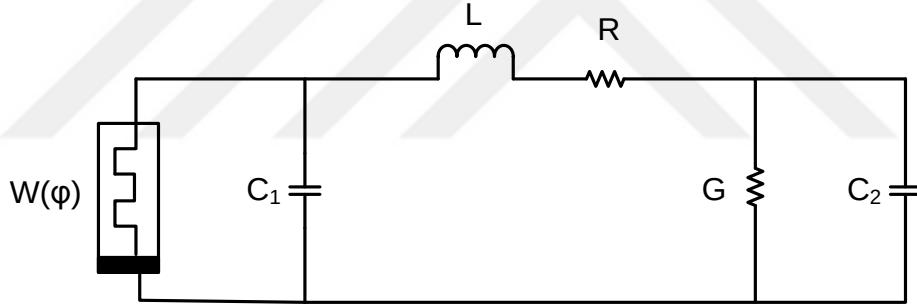


Şekil 4.20. Kanonik Chua osilatör devresi

Devreye ait durum denklemleri ise;

$$\begin{aligned} C_1 \dot{v}_1 &= i_3 - h(v_1) \\ C_2 \dot{v}_2 &= -i_3 + Gv_2 \\ L \dot{i}_3 &= v_2 - v_1 \end{aligned} \quad (4.18)$$

olarak verilmektedir. Denklemden v_1 , v_2 , i_3 sırasıyla C_1 ve C_2 kondansatörleri üzerine düşen gerilimleri ve L indüktansının akımını temsil eder. Ayrıca $h(v_1) = m_1 v_1 + 0.5(m_0 - m_1)(|v_1 + 1| - |v_1 - 1|)$ doğrusal olmayan fonksiyon olarak tanımlanır. Bu fonksiyon Chua diyodundan geçen akımı ifade etmektedir. Fonksiyonda yer alan m_0 ve m_1 sistem parametrelerini ve G devrenin iletkenliğini temsil etmektedir [69]. Eğer Chua diyodunun yerine akı kontrollü memristör yerleştirilirse, dört durum değişkenine sahip Memristörlü HYSA devresi oluşturulur ve bu devre Şekil 4.21'deki gibi gösterilmektedir.



Şekil 4.21. Memristörlü HYSA modeline ait devre şeması

Devreye ait durum denklemleri

$$\begin{aligned} C_1 \dot{v}_1 &= i_3 - W(\varphi) v_1 \\ C_2 \dot{v}_2 &= -i_3 + Gv_2 \\ L \dot{i}_3 &= v_2 - v_1 \\ \dot{\varphi} &= v_1 \end{aligned} \quad (4.19)$$

olarak elde edilir ve $W(\varphi)$, Faraday'ın indüksiyon yasası ve Denklem (4.17) kullanılarak düzenlenirse,

$$W(\varphi) = \frac{dq(\varphi)}{d\varphi} = \begin{cases} d, & |\varphi| \geq 1 \\ c, & |\varphi| < 1 \end{cases} \quad (4.20)$$

halini alır. Denklem (4.19) ve (4.20) daha basite indirgenerek,

$$\begin{aligned}\dot{x} &= \alpha(y - W(w)x) \\ \dot{y} &= z - x \\ \dot{z} &= -\beta y + \gamma z \\ \dot{w} &= x\end{aligned}\tag{4.21}$$

$$W(w) = \frac{dq(\varphi)}{d\varphi} = \begin{cases} d, & |\varphi| \geq 1 \\ c, & |\varphi| < 1 \end{cases}\tag{4.22}$$

$x = v_1, y = i_3, z = v_2, w = \varphi, \alpha = 1/C_1, \beta = 1/C_2, \gamma = G/C_2$ ve $L = 1$ tanımlamalarıyla son halini alır [68]. Böylece model,

$$\begin{aligned}\dot{x} &= \alpha(y - V_M) \\ \dot{y} &= z - x \\ \dot{z} &= -\beta y + \gamma z\end{aligned}\tag{4.23}$$

durum denklemlerine sahip üç hücreli HYSA ve $V_M = f(x, \int x dx) = W(\int x dt)x$ dinamik doğrusal olmayan fonksiyon ile de dördüncü hücre temsil edilmektedir [67]. Böylece Memristörlü HYSA modeli, Denklem (4.3)'te verilen DK-HYSA modeli kullanılarak,

$$I_j = B_j = 0 \quad 1 \leq j \leq 4\tag{4.24}$$

$$A_{jk} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} -\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\tag{4.25}$$

$$C_{jk} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} = \begin{bmatrix} 1 & \alpha & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -\beta & (\gamma+1) & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}\tag{4.26}$$

şablon tanımlamaları ile ifade edilir. Memristörlü HYSA modeli; $\alpha = 4, \beta = 1, \gamma = 0.65, c = 0.2, d = 10$ parametre değerleri ve $(x(0), y(0), z(0), w(0)) = [0.1, 0.01, 0.2, 0.7]$ başlangıç değerlerinde kaotik davranış göstermektedir [68].

Önerilen RSÜ uygulamasında [70] Memristörlü HYSA modeli ile lojistik harita sistemi birlikte entropi kaynağı olarak kullanılırken, XOR fonksiyonu son işlem olarak uygulanmıştır.

4.3.1. Memristörlü HYSA Modelinin Donanım Tabanlı Benzetimi

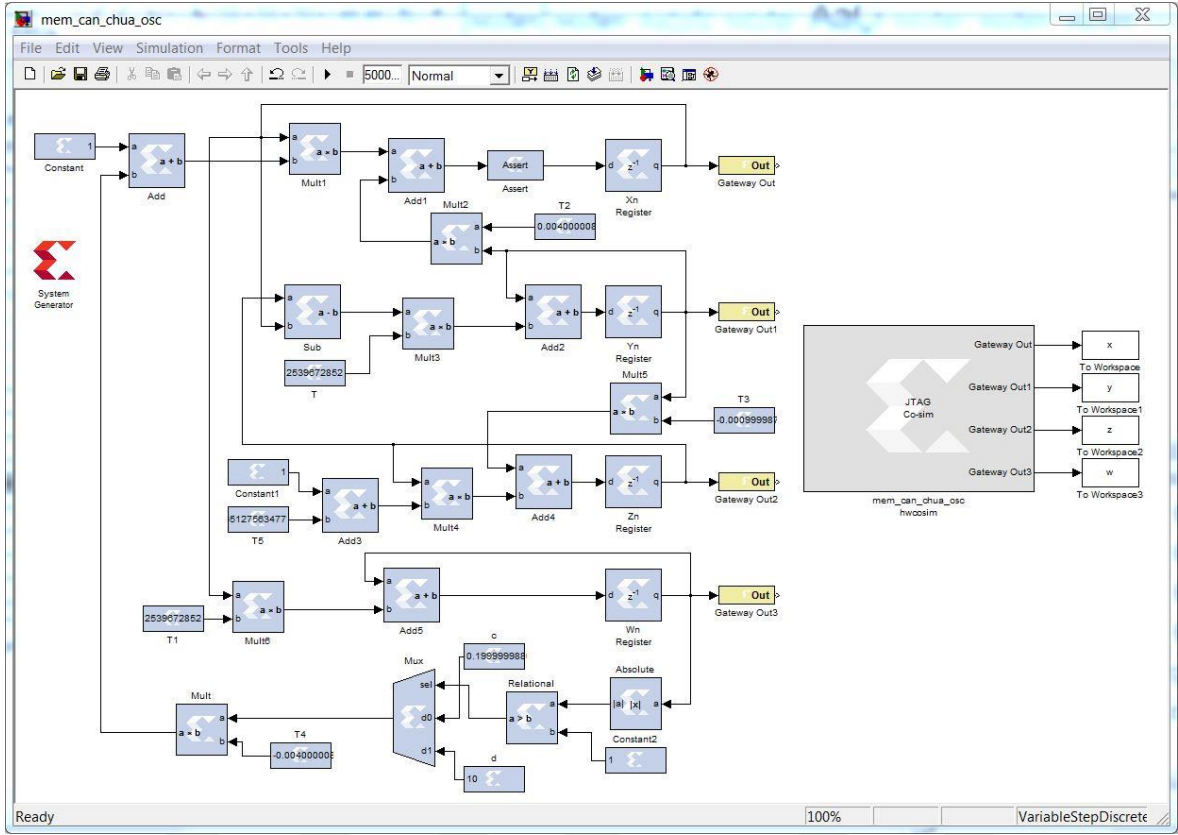
Önerilen RSÜ tasarımı için kullanılan HYSA modelinin uygulanabilirliğini test etmek ve davranışını analiz etmek amacıyla Denklem (4.21) ve (4.22)'de verilen Memristörlü HYSA modelinin ayrık zamanlı olarak XSG platformunda donanım tabanlı benzetimi gerçekleştirilecektir. Model İleri Euler yöntemi ile ayrıklaştırılırsa

$$\begin{aligned}x[n+1] &= x[n](1 - \alpha h W(n)) + \alpha h y[n] \\y[n+1] &= y[n] + h(z[n] - x[n]) \\z[n+1] &= z[n](1 + \gamma h) - \beta h y[n] \\w[n+1] &= w[n] + h x[n]\end{aligned}\tag{4.27}$$

denklem takımı elde edilir. Doğrusal olmayan fonksiyon ise

$$W(n) = \begin{cases} d, & |w(n)| \geq 1 \\ c, & |w(n)| < 1 \end{cases}\tag{4.28}$$

olarak ayrıklaştırılır. Ayrık zamanlı HYSA modeli; ; $\alpha = 4$, $\beta = 1$, $\gamma = 0.65$, $c = 0.2$, $d = 10$ parametre değerleri ve $(x(0), y(0), z(0), w(0)) = [0.1, 0.01, 0.2, 0.7]$ başlangıç değerleri, $h = 0.001$ adım aralığı ve $n = 500000$ iterasyon için Şekil 4.22'de verilen XSG modeli ile donanım tabanlı benzetimi gerçekleştirilmiştir. XSG modelinin benzetim sonucunda elde edilen faz portreleri Şekil 4.23'de gösterilmektedir.

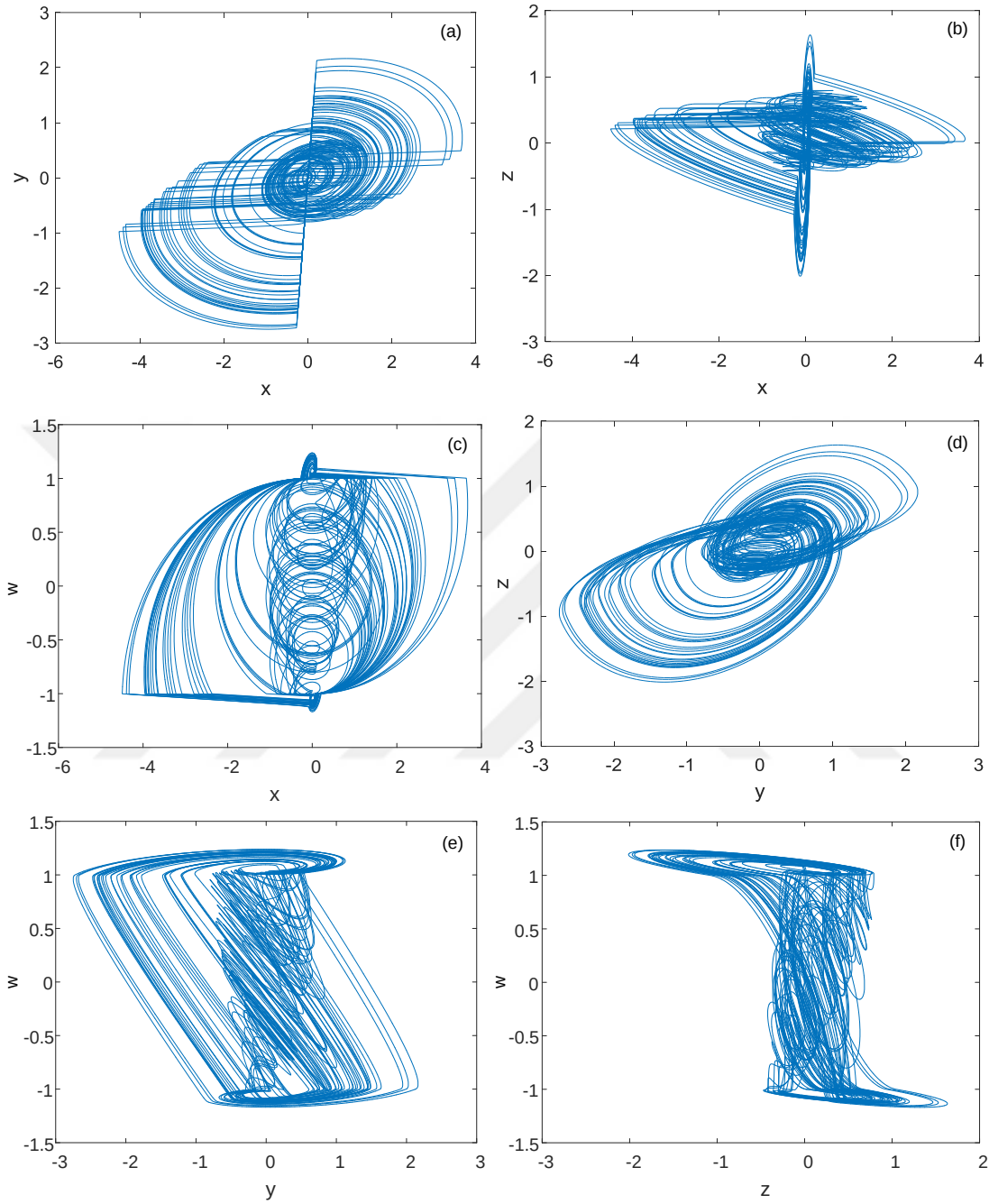


Şekil 4.22. Memristörlü HYSA'nın XSG modeli

Şekil 4.23'de görüldüğü üzere Memristörlü HYSA modeline ait faz portrelerinin değişimi kaotik çeker davranışı göstermektedir. Memristörlü HYSA'nın XSG ortamında donanım tabanlı benzetimi Tablo 4.5'de verilen kaynak kullanımı ile başarılı bir şekilde gerçekleştirilmiştir.

Tablo 4.5. Memristörlü HYSA'nın XSG modelinin FPGA kaynak kullanımı

Kaynak Adı	Kullanım Sayısı
FF	518
LUT	652

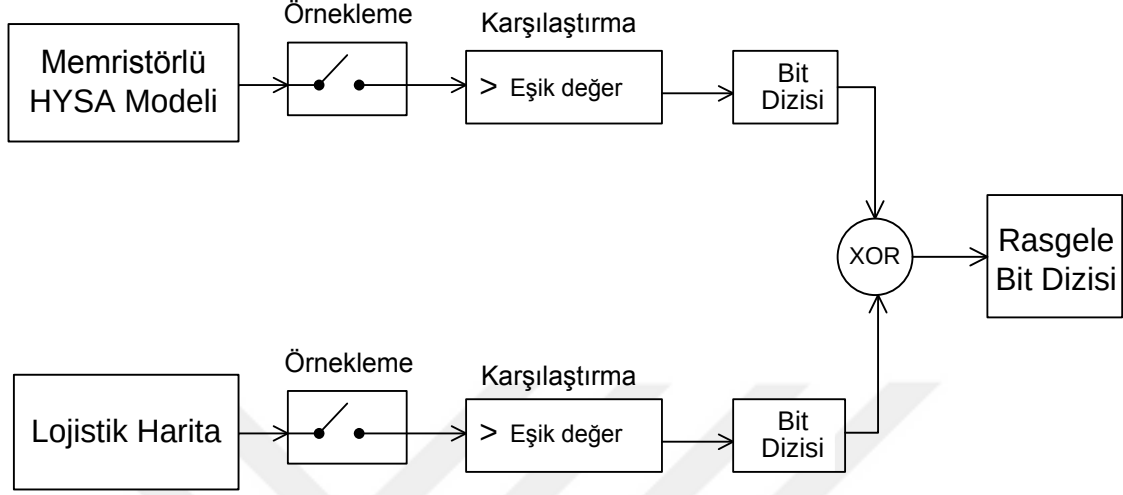


Şekil 4.23. Memristörlü HYSA'nın XSG modeline ait faz portreleri a) $x - y$ b) $x - z$ c) $x - w$ d) $y - z$ e) $y - w$ f) $z - w$

4.3.2. Memristörlü HYSA Modeli ile FPGA Ortamında RSÜ Tasarımı

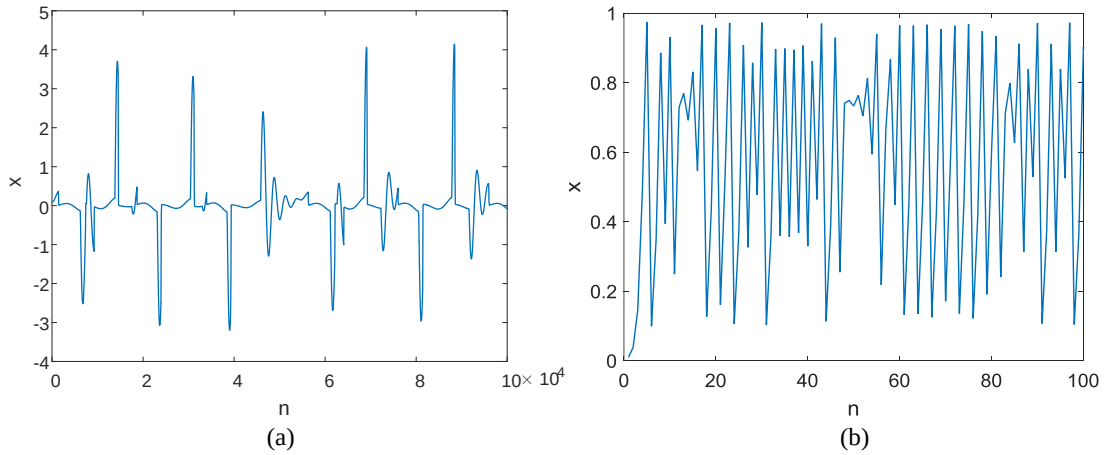
Önerilen RSÜ tasarımı Memristörlü HYSA modeli ile lojistik harita sistemi entropi kaynağı olarak kullanılıp gerçek rasgele bit üretimi gerçekleştirilmiştir. Tasarımda son

işlem olarak XOR fonksiyonu kullanılmıştır. Tasarıma ilişkin blok diyagram Şekil 4.24’de gösterilmektedir.



Şekil 4.24. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımı

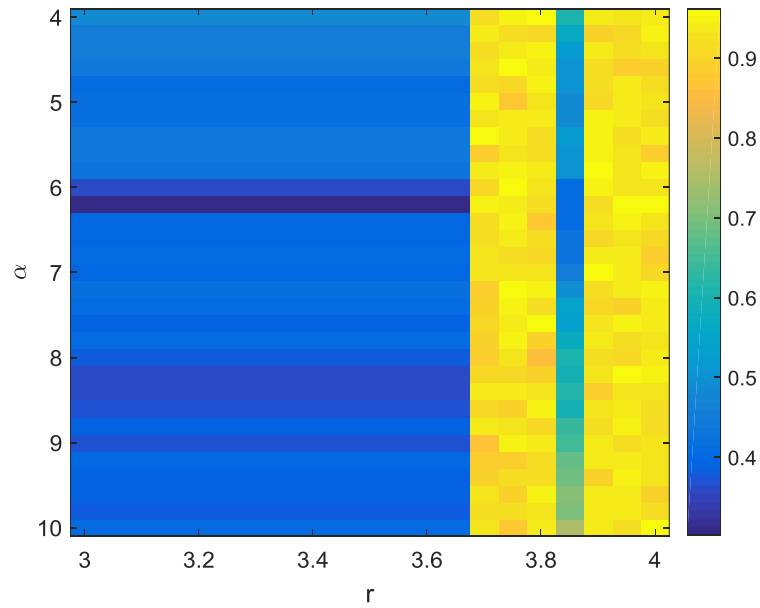
Tasarımda entropi kaynağı olarak kullanılan Memristörlü HYSA modelinin verilen parametre ve başlangıç değerleri için elde edilen x durum değişkeni ile Denklem (3.4)’de verilen lojistik harita sistemine ait durum değişkeninin $r = 4$ için elde edilen değişimleri Şekil 4.25’de gösterilmektedir.



Şekil 4.25. a) Memristörlü HYSA modeli ve b) lojistik harita sisteminin x durum değişkenlerinin değişimi

Şekil 4.25’den de görüldüğü gibi elde edilen değişimler gürültü benzeri sinyallerdir. Tasarımda iki sinyal için de aynı örnekleme ve sayısallaştırma yöntemi kullanılmıştır. Lojistik harita sistemi yapısal olarak ayırık zamanlı olması nedeni ile adım aralığı 1’dir.

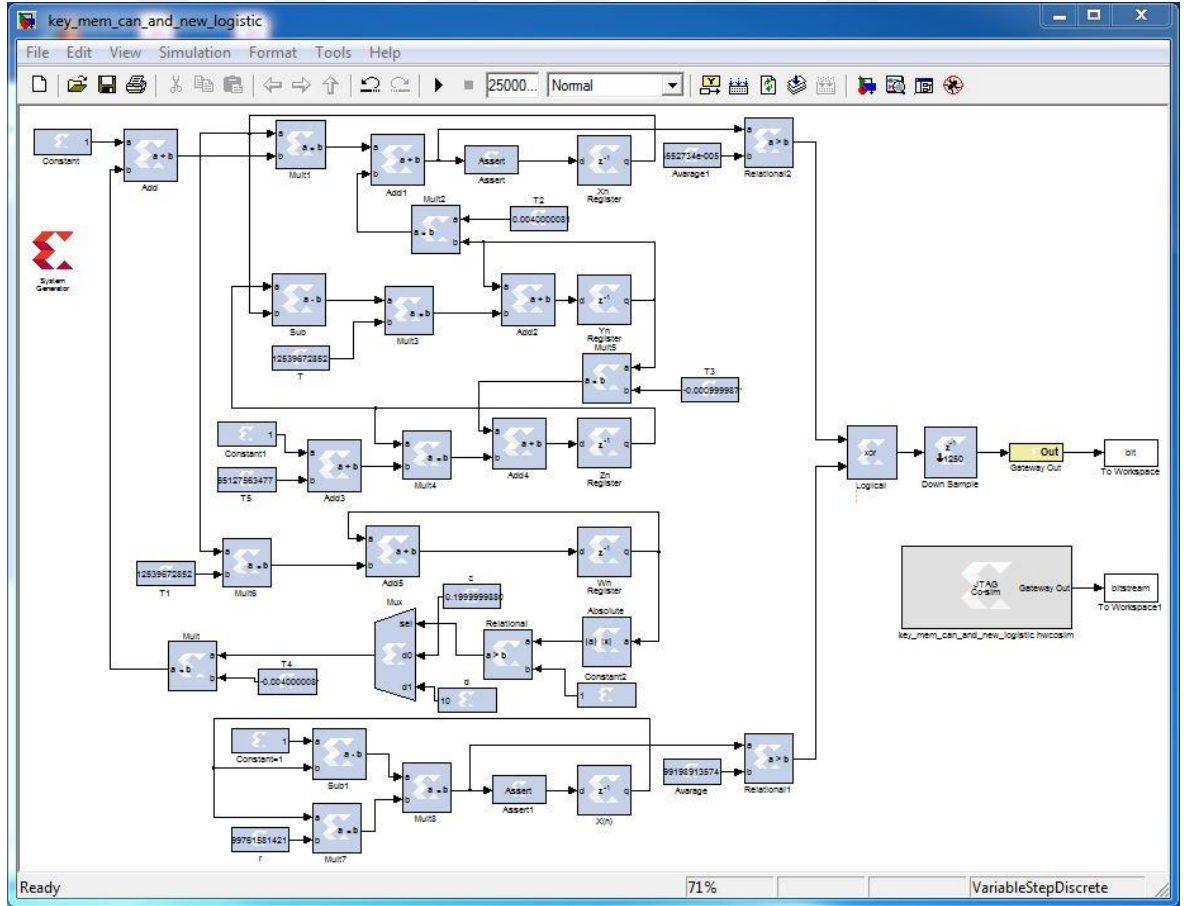
Fakat Memristörlü HYSA modeline ait durum değişkeni için örnekleme adım aralığı $h = 0.001$ olarak uygulanmıştır. Bu nedenle iki durum değişkeni için farklı iterasyon sayısı kullanılmıştır. Örnekleme frekans değeri, eşit dağılımlı “0” ve “1” bitleri elde etmek için 1250 olarak seçilmiş ve elde edilen ondalıklı sayılar karşılaştırma bloğuna uygulanmıştır. Karşılaştırma bloğunda belirtilmiş eşik değeri, her iki durum değişkeni için de ayrı ayrı hesaplanmış ortalama değeri temsil etmektedir. Tasarım sonucunda üretilen bit dizisinin istatistiksel rasgeleliliğini artırmak amacıyla en uygun parametre seçimi gerçekleştirilmiştir. Bu amaçla tasarıma ait ayrık zamanlı denklemler MATLAB ortamında nümerik olarak iteratif bir şekilde çalıştırılmıştır. Her iterasyonda Şekil 4.24’de verilen tasarım için; lojistik harita sisteminde r , Memristörlü HYSA modelinde ise α parametreleri sistematik olarak küçük adımlarla değiştirilip, üretilen bit dizileri ölçek indeksi algoritmasına uygulanmıştır. Her iterasyon için r parametresi kaotik davranışın elde edildiği $r = [3,4]$ aralığında 0.05 adım aralığıyla değiştirilirken, α parametresi [71] çalışmasında verilen $\alpha = [4,10]$ aralığında 0.2 adım aralığıyla değiştirilmiştir. Böylece 200000 bit uzunluğunda olan 651 rasgele bit vektörü üretilip her 200000 bit için ölçek indeksi algoritması çalıştırılmıştır. Ölçek indeksi algoritmasına giriş olarak verilen 200000 bitlik her dizi, 4000 bitlik parçalara ayrılarak iteratif bir şekilde 50 adet indeks değeri elde edilmiştir. Şekil 4.26’da verilen ölçeklendirilmiş değerlerin her biri ise, elde edilen 50 indeks değerinin ortalamasını temsil etmektedir.



Şekil 4.26. α ve r parametre değişimlerine karşın elde edilen ortalama ölçek indeksi değerleri

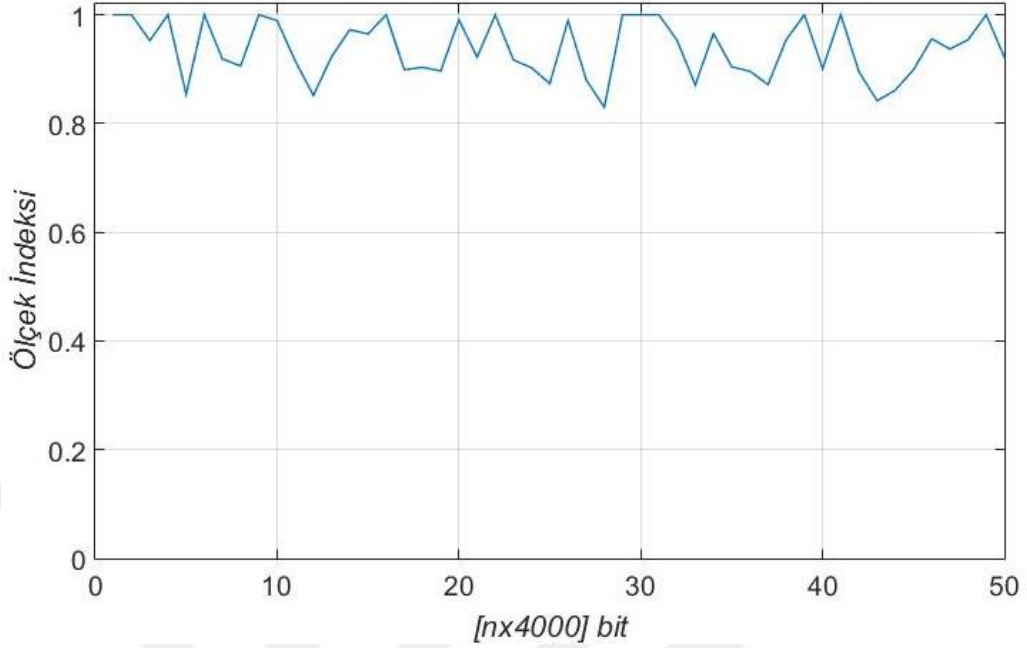
Ortalama ölçek indeksi hesaplamalarının sonucunda maksimum ortalama ölçek indeksi $\alpha = 6.2$ ve $r = 3.95$ parametre değerlerinde $i_s = 0.9618$ olarak elde edilmiştir. Ayrıca maksimum ölçek indeksinin hesaplandığı bu şartlarda Memristörlü HYSA modeline ait x durum değişkeninin eşik (ortalama) değeri $-6.3 \cdot 10^{-6}$, lojistik harita sistemine ait x durum değişkeninin eşik değeri ise 0.5836 olarak elde edilmiştir.

Tasarımın donanım tabanlı benzetimi elde edilen parametre ve eşik değerler kullanılarak Şekil 4.27’de verilen XSG modeli ile gerçekleştirilmiştir. Tasarımda tüm aşamalar için Xilinx Kintex-7 KC705 FPGA geliştirme kartı kullanılmıştır. Ayrıca XSG modeli 250000000 iterasyon için çalıştırılıp üretilen 200000 bitlik veri MATLAB platformuna aktararak kaydedilmiştir.



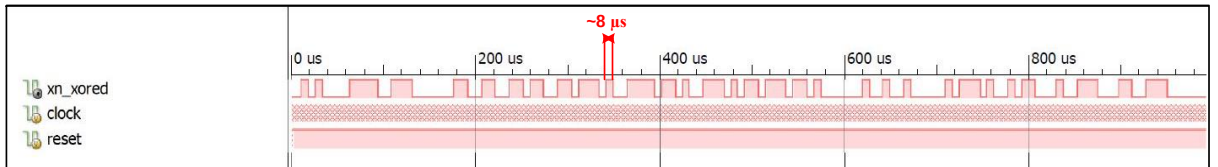
Şekil 4.27. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımının XSG donanım tabanlı benzetim modeli

XSG modelinin çıkışında elde edilen 200000 bitlik rasgele dizi için ölçek indeksi algoritması çalıştırılıp her 4000 bit için elde edilen indeks değerlerinin değişimi Şekil 4.28’de gösterilmektedir.



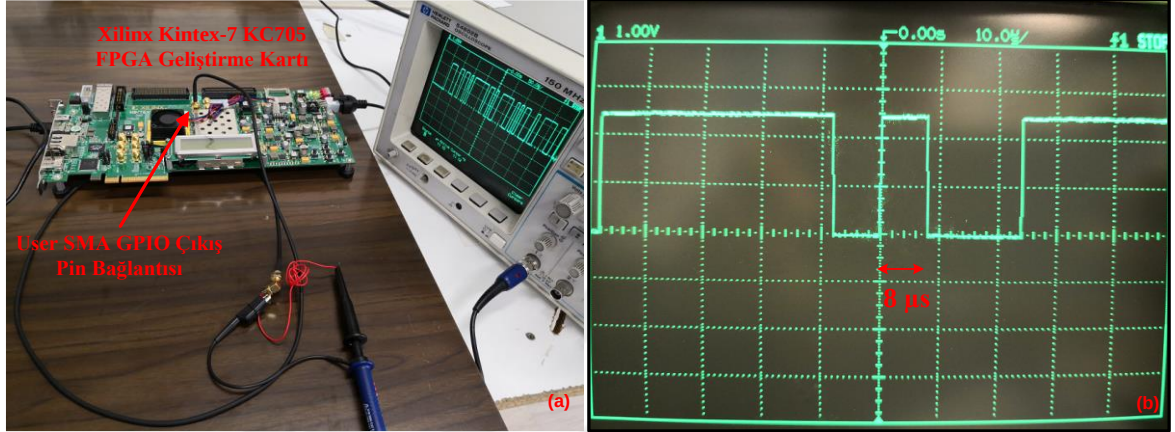
Şekil 4.28. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımına ait ölçek indeksinin değişimi

Şekil 4.28’de gösterilen ölçek indekslerinin ortalama değeri 0.9367 olarak hesaplanmıştır. Önerilen RSÜ tasarımı Verilog HDL dili ile Xilinx ISE 14.5 FPGA editöründe kodlanmıştır. Tasarıma ait aritmetik işlem aşamaları için Şekil 4.16’da verilen aritmetik toplama ve çarpma modülleri kullanılmıştır. Aritmetik işlemlerde önceki tasarımlardan farklı olarak daha hassas değerler elde etmek için Q8.24 32 bitlik 2’ye tımlıyen sabit noktalı sayı formatı tercih edilmiştir. Tasarımın benzetim aşamasında ise Kintex-7 XC7K325T-FFG900 FPGA çipi seçilmiştir. Önerilen RSÜ tasarımında *clock* ve *reset* adında 1’er bitlik girişler ve *xn_xored* adında 1 bitlik rasgele bit çıkışı tanımlanmıştır. Kullanılan FPGA kartında 156.25 MHz frekansına sahip saat darbe osilatörü kullanılarak *clock* girişi tanımlanmıştır. Kart üzerinde bulunan butonlardan biri ise *reset* girişi olarak seçilmiştir. Tasarımın çıkış biti olan *xn_xored* değişkeninin değişimi benzetim platformunda Şekil 4.29’daki gibi elde edilmiştir.



Şekil 4.29. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımının FPGA editöründeki benzetim sonucu

Şekil 4.29’da görüldüğü gibi, benzetim sonucunda 8 μ s’de bir çıkış biti değişmektedir. Böylece 125 kbps veri hızına sahip bit üretimi gerçekleştirilmiştir. Tasarıma ait programlama dosyası, sentezleme ve giriş-çıkış tanımlamalarının ardından oluşturulup Şekil 4.30’da verilen platformda gerçek zamanlı rasgele bit üretimi gerçekleştirilmiştir.



Şekil 4.30. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımının gerçek zamanlı çalışma platformu ve osiloskop ekran çıktısı

Önerilen RSÜ tasarımı, Tablo 4.6’da verilen FPGA kaynak kullanımı ile başarılı bir şekilde gerçek zamanlı olarak uygulanmış ve literatürde yer alan en güncel RSÜ tasarımları ile karşılaştırılmıştır.

Tablo 4.6. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımının literatürdeki benzer çalışmalar ile karşılaştırılması

Çalışma	RSÜ Türü	Entropi Kaynağı	FPGA Kaynak Kullanımı
[72]	SRSÜ	Bernoulli Haritası	315 Lojik Dilim, 575 LUT
[72]	SRSÜ	Tent Haritası	402 Lojik Dilim, 769 LUT
[72]	SRSÜ	Zigzag Haritası	476 Lojik Dilim, 928 LUT
[4]	GRSÜ	Ayrık Zamanlı Kaotik Sistem	236 FF, 406 LUT
Üç Hücreli Otonom HYSA tabanlı RSÜ Tasarımı [63]	GRSÜ	Ayrık Zamanlı Üç Hücreli Otonom Kaotik Sistem	5069 FF, 13519 LUT
Memristörlü HYSA tabanlı RSÜ Tasarımı [70]	GRSÜ	Ayrık Zamanlı Memristörlü Kaotik Sistem ve Lojistik Harita	133 FF, 311 LUT

Önerilen RSÜ tasarımının literatürde yer alan en güncel ve benzer çalışmalara kıyasla daha az FPGA kaynak kullanımı ile rasgele bit üretimi gerçekleştirdiği görülmektedir.

Tasarım için XSG platformunun donanım tabanlı çalıştırılması sonucunda kaydedilmiş 200000 bitlik veri Tablo 4.7’deki sonuçlarla NIST istatistiksel testlerinden başarılı bir şekilde geçmiştir.

Tablo 4.7. Önerilen Memristörlü HYSA tabanlı RSÜ tasarımı için NIST istatistiksel test sonuçları

NIST Testleri (200000 bit)	Memristörlü HYSA tabanlı RSÜ tasarımı	Sonuç
Frekans	0.1329	Başarılı
Bir blok içindeki frekans	0.0812	Başarılı
Akış	0.3433	Başarılı
Bir blok içinde en uzun birlerin akış	0.5697	Başarılı
İkili matris rankı	0.7679	Başarılı
Ayrık Fourier dönüşümü	0.5518	Başarılı
Örtüşmeyen şablon eşleştirme	0.9657	Başarılı
Örtüşen şablon eşleştirme	0.9527	Başarılı
Maurer evrensel istatistik	0.0931	Başarılı
Doğrusal karmaşıklık	0.5612	Başarılı
Seri (1/2)	0.9062 / 0.7082	Başarılı
Yaklaşık entropi	0.8879	Başarılı
Kümülatif toplamlar	0.1165	Başarılı
Rasgele gezinimler (ortalama)	0.4037	Başarılı
Rasgele gezinim değişkenler (ortalama)	0.4366	Başarılı

Önerilen RSÜ tasarımı; Memristörlü HYSA modeli ve lojistik harita sisteminin entropi kaynağı, XOR fonksiyonunun son işlemci olarak kullanılmasıyla 125 kbps çıkış bit oranında gerçek rasgele bit üretimi gerçekleştirilmiştir. Ayrıca tasarım sonucunda elde edilen bitlerin 0.9367 ölçek indeksi ile periyodik olmadığı söylenebilir. Elde edilen sonuç güncel çalışmalarla ve üç hücreli otonom HYSA tabanlı RSÜ tasarımı ile kıyaslanırsa, daha az donanım elemanı kullanımı ile tasarımın gerçek zamanlı olarak başarılı bir şekilde gerçekleştirildiği görülmektedir.

5. SONUÇ VE ÖNERİLER

Bu tezde entropi kaynağı olarak, kaotik sistemler ve kaotik davranış sergileyen HYSA modelleri kullanılarak rasgele sayı/bit üretimi gerçekleştirilmiştir. Tez çalışmasında üç adet özgün RSÜ tasarımı önerilmiştir.

İlk RSÜ tasarımı XSG platformunda modellenmiş olup donanım tabanlı benzetimi gerçekleştirilerek rasgele bit üretimi sağlanmıştır. Tasarımda kullanılan üç boyutlu kaotik sisteme ait durum değişkenlerinin ayrık zamanlı ortalama değerleri hesaplanarak durum değişkenleri sayısallaştırılmıştır. Son işlem olarak kullanılan Trivium algoritması ile sayısallaştırılmış durum değişkenlerinin rasgeleliliği artırılmıştır.

İkinci RSÜ tasarımında ise üç hücreli otonom HYSA modeli entropi kaynağı olarak kullanılmıştır. Bu tasarımda üç senaryonun her biri için 1001600 bit üretimi gerçekleştirilmiştir. Sadece HYSA modelinin sayısallaştırılmış durum değişkeninin rasgeleliliği test edildiğinde başarısız sonuçlar görülmüştür. Ancak frekans testine göre durum değişkeninin son işlemde geçirilerek iyi istatistiksel sonuçlar vereceği öngörülmüştür. İkinci senaryoda, ilk senaryonun çıkışı Trivium şifreleme algoritmasına uygulanarak elde edilen bit dizisi NIST istatistiksel testlerinin tümünden başarılı bir şekilde geçmiştir. Üç hücreli otonom HYSA modelinin kaotik davranışını ön plana çıkarmak amacıyla bu senaryoda son işlem olarak sadece Trivium algoritmasının anahtar üretim fazına ait algoritmik işlemler kullanılmıştır. Böylece üçüncü senaryoya ait RSÜ tasarımı sonucunda tüm NIST istatistiksel testlerinden başarılı bir şekilde geçen 2.015 Mbps çıkış hızında rasgele bit üretimi FPGA kartında gerçekleştirilmiştir.

Üçüncü RSÜ tasarımında entropi kaynağı olarak iki kaotik sistem kullanılmıştır. Bu sistemlerden biri Memristörlü HYSA modeli diğeri ise lojistik harita sistemidir. İki kaotik sistem, aynı örnekleme frekansı için örneklendikten sonra sistemlere ait x durum değişkenlerinin ortalama değerleri hesaplanmıştır. Ortalama değerleri ile karşılaştırılıp sayısallaştırılan durum değişkenlerine XOR son işlemi uygulanmıştır. Bu tasarımın asıl amacı literatürde var olan RSÜ tasarımlarından daha az FPGA kaynak kullanımı sağlamaktır. Nitekim bu çalışma sonucunda, tasarım yapısı benzer ve güncel olan çalışmalara kıyasla en düşük FPGA donanım elemanı kullanımı ile 125 kbps çıkış hızında rasgele bit üretimi gerçekleştirilmiştir.

Bu tez çalışmasında elde edilen sonuçlardan yola çıkılarak aşağıdaki öneriler ile yeni tasarımların gerçekleştirilmesi düşünülebilir:

- Üç RSÜ tasarımı için de elektronik devre elemanları ile sayısal devreler oluşturularak rasgele bit üretimi sağlanabilir.
- Önerilen tasarımlar için farklı son işlem algoritmaları uygulanarak daha yüksek hızlarda bit üretimi gerçekleştirilebilir.
- Trivium şifreleme algoritması için elektronik devre kartı tasarlanıp basit bir kaotik sistem kullanılarak yeni bir kaotik harita sistemi gerçekleştirilebilir.



KAYNAKLAR

- [1] C. K. Koc, "About cryptographic engineering," *Cryptographic engineering*, Springer, pp. 1-4, 2009.
- [2] K. Wold, "Security Properties of a Class of True Random Number Generators in Programmable Logic," Doctor of Philosophy, Information Security, Gjøvik University College, 2011.
- [3] M. D. P. Galajda, "Chaos-Based True Random Number Generator Embedded in a Mixed-Signal Reconfigurable Hardware," *Journal of Electrical Engineering*, vol. 57, no. 4, pp. 218-225, 2006.
- [4] T. Bonny, R. Al Debsi, S. Majzoub, and A. S. Elwakil, "Hardware Optimized FPGA Implementations of High-Speed True Random Bit Generators Based on Switching-Type Chaotic Oscillators," *Circuits, Systems, and Signal Processing*, vol. 38, no. 3, pp. 1342-1359, 2018.
- [5] İ. Koyuncu, A. T. Özcerit, İ. Pehlivan, and E. Avaroğlu, "FPGA Üzerinde Kaos Tabanlı Gerçek Rasgele Sayı Üretici Tasarımı ve Gerçeklemesi " *IEEE 22nd Signal Processing and Communications Applications Conference (SIU 2014)*, pp. 236-239, 2014.
- [6] S. H. Strogatz and D. E. Herbert, "Nonlinear dynamics and chaos," *Medical Physics-New York-Institute of Physics*, vol. 23, no. 6, pp. 993-995, 1996.
- [7] T. Tuncer, E. Avaroglu, M. Türk, and A. B. Ozer, "Implementation of non-periodic sampling true random number generator on FPGA," *Informacije MIDEM*, vol. 44, no. 4, pp. 296-302, 2015.
- [8] B. Karakaya, M. A. Turk, M. Turk, and A. Gulten, "Selection of optimal numerical method for implementation of Lorenz Chaotic system on FPGA," *International Advanced Researches and Engineering Journal*, vol. 2, no. 2, pp. 147-152, 2018.
- [9] B. Alataş, "Kaotik Haritalı Parçacık Sürü Optimizasyon Algoritmaları Geliştirme," Doktora Tezi, Fırat Üniversitesi Fen Bilimleri Enstitüsü, 2007.
- [10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Booz-Allen and Hamilton Inc Mclean Va2001.
- [11] De Canni'ere C and B. Preneel, "Trivium specifications," 2005. Online. Available: http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.
- [12] K. Marton, A. Suci, C. Sacarea, and O. Cret, "Generation and testing of random numbers for cryptographic applications," *Proceedings of the Romanian Academy, Series A*, vol. 13, no. 4, pp. 368-377, 2012.
- [13] W. Killmann and W. Schindler, "AIS 31: Functionality classes and evaluation methodology for true (physical) random number generators, version 3.1," *Bundesamt für Sicherheit in der Informationstechnik (BSI)*, Bonn, 2001.
- [14] E. Avaroğlu, "Donanım Tabanlı Rasgele Sayı Üreticinin Gerçekleştirilmesi," Doktora Tezi, Fırat Üniversitesi Fen Bilimleri Enstitüsü, 2014.
- [15] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656-715, 1949.
- [16] V. B. Suresh and W. P. Burleson, "Entropy extraction in metastability-based TRNG," *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 135-140, 2010.
- [17] R. B. Davies, "Exclusive OR (XOR) and hardware random number generators," 2002. Online. Available: <http://www.robertnz.net/pdf/xor2.pdf>.

- [18] S. Yıldırım, "A True Random Number Generator In Fpga For Cryptographic Applications, A Thesis Submitted To The Graduate School Of Natural And Applied Sciences Of Middle East Technical University," 2012.
- [19] J. Von Neumann, "Various techniques used in connection with random digits," *Applied Math Series*, vol. 12, no. 36-38, p. 1, 1951.
- [20] M. Varchola, "FPGA Based True Random NumberGenerators for Embedded Cryptographic Applications, Phd Thesis, Faculty of Electrical Engineering and Informatics Department of Electronics and Multimedia Communications, Technical University of Kosice," 2008.
- [21] B. Jun and P. Kocher, "The Intel random number generator," *Cryptography Research Inc. white paper*, 1999.
- [22] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.
- [23] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.
- [24] D. E. Standard, "Federal information processing standards publication 46," *National Bureau of Standards, US Department of Commerce*, vol. 23, 1977.
- [25] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.
- [26] M. E. Yalçın, J. A. Suykens, and J. Vandewalle, "True random bit generation from a double-scroll attractor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 7, pp. 1395-1404, 2004.
- [27] M. Drutarovsky and P. Galajda, "A robust chaos-based true random number generator embedded in reconfigurable switched-capacitor hardware," *IEEE 17th International Conference Radioelektronika*, pp. 1-6, 2007.
- [28] J.-L. Danger, S. Guilley, and P. Hoogvorst, "High speed true random number generator based on open loop structures in FPGAs," *Microelectronics journal*, vol. 40, no. 11, pp. 1650-1656, 2009.
- [29] P. Wiczorek, "Dual-metastability fpga-based true random number generator," *Electronics Letters*, vol. 49, no. 12, pp. 744-745, 2013.
- [30] J. Murphy, "Field-programmable true random number generator," *Electronics Letters*, vol. 48, no. 10, pp. 565-566, 2012.
- [31] J. C. Sprott, "Some simple chaotic flows," *Physical review E*, vol. 50, no. 2, pp. 647-650, 1994.
- [32] I. Cicek, A. E. Pusane, and G. Dundar, "An integrated dual entropy core true random number generator," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 3, pp. 329-333, 2017.
- [33] M. Alcin, I. Koyuncu, M. Tuna, M. Varan, and I. Pehlivan, "A novel high speed Artificial Neural Network-based chaotic True Random Number Generator on Field Programmable Gate Array," *International Journal of Circuit Theory and Applications*, 2018.
- [34] M. M. Zirkohi, "Model reference type-2 fuzzy sliding mode control for a novel uncertain hyperchaotic system," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 1, pp. 389-400, 2017.
- [35] N. Jia and T. Wang, "Chaos control and hybrid projective synchronization for a class of new chaotic systems," *Computers & Mathematics with Applications*, vol. 62, no. 12, pp. 4783-4795, 2011.
- [36] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the atmospheric sciences*, vol. 20, no. 2, pp. 130-141, 1963.

- [37] L. O. Chua, "The genesis of Chua's circuit," *Arch Elektron Ubertragung*, vol. 46, no. 3, pp. 250-257, 1992.
- [38] S. Çiçek, A. Ferikoğlu, and I. Pehlivan, "A new 3D chaotic system: dynamical analysis, electronic circuit design, active control synchronization and chaotic masking communication application," *Optik-International Journal for Light and Electron Optics*, vol. 127, no. 8, pp. 4024-4030, 2016.
- [39] B. Muthuswamy and L. O. Chua, "Simplest chaotic circuit," *International Journal of Bifurcation and Chaos*, vol. 20, no. 05, pp. 1567-1580, 2010.
- [40] T.-Y. Li and J. A. Yorke, "Period three implies chaos," *The American Mathematical Monthly*, vol. 82, no. 10, pp. 985-992, 1975.
- [41] C. Jung, "Poincaré map for scattering states," *Journal of physics A: mathematical and general*, vol. 19, no. 8, p. 1345, 1986.
- [42] R. Devaney, *An introduction to chaotic dynamical systems*. Westview press, 2008.
- [43] E. Lorenz, "Predictability: does the flap of a butterfly's wings in Brazil set off a tornado in Texas " *Paper presented at: Annual Meeting of the American Association for the Advancement of Science, December 1979, Washington, DC*.
- [44] A. Buscarino, L. Fortuna, and M. Frasca, *Essentials of Nonlinear Circuit Dynamics with MATLAB® and Laboratory Experiments*. CRC Press, 2017.
- [45] P. A. Cook, "Nonlinear dynamical systems," *Prentice Hall*, 1994.
- [46] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285-317, 1985.
- [47] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507-519, 1971.
- [48] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, p. 80, 2008.
- [49] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature nanotechnology*, vol. 3, no. 7, p. 429, 2008.
- [50] B. Karakaya, M. E. Şahin, H. Güler, and A. Gülten, "Memristör Tabanlı Kaotik Devre Modelinin FPGA Ortamında Tasarımı ve Gerçeklenmesi," *Fırat Üniversitesi Fen Bilimleri Dergisi*, vol. 29, no. 2, pp. 39-44, 2017.
- [51] B. Karakaya and A. Gülten, "3D Chaotic System based Random Bit Generation," in *International Engineering and Natural Sciences Conference (IENSC)*, 2018, pp. 1592-1599.
- [52] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transactions on circuits and systems*, vol. 35, no. 10, pp. 1257-1272, 1988.
- [53] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Transactions on circuits and systems*, vol. 35, no. 10, pp. 1273-1290, 1988.
- [54] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 3, pp. 147-156, 1993.
- [55] T. Roska and L. O. Chua, "The CNN universal machine: an analogic array computer," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp. 163-173, 1993.
- [56] T. Roska and Á. Rodríguez-Vázquez, "Toward visual microprocessors," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1244-1257, 2002.
- [57] F. Luigi, F. Mattia, and X. M. Gabriella, *Chua's Circuit Implementations: Yesterday, Today And Tomorrow*. World Scientific, 2009.

- [58] L. O. Chua and T. Roska, *Cellular neural networks and visual computing: foundations and applications*. Cambridge university press, 2002.
- [59] G. Manganaro, P. Arena, and L. Fortuna, *Cellular neural networks: chaos, complexity and VLSI processing*. Springer Science & Business Media, 2012.
- [60] J. Nossek, H. Magnussen, P. Nachbar, and A. Schuler, "Learning algorithms for cellular neural networks," *Proc. of NOLTA'93*, pp. 11-16, 1993.
- [61] J. Nossek, "Design and learning with cellular neural networks," *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and their Applications, 1994. CNNA-94*, pp. 137-146, 1994.
- [62] F. Zou and J. A. Nossek, "Bifurcation and chaos in cellular neural networks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 3, pp. 166-173, 1993.
- [63] B. Karakaya, V. Çelik, and A. Gülten, "Chaotic cellular neural network-based true random number generator," *International Journal of Circuit Theory and Applications*, vol. 45, no. 11, pp. 1885-1897, 2017.
- [64] B. Karakaya, "Bir Hücreli Yapay Sinir Ağı'nın Sabit Noktalı Sayı Aritmetiğiyle Sayısal Tasarımı Ve Gerçeklenmesi," Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, 2014.
- [65] R. Benítez, V. Bolós, and M. Ramírez, "A wavelet-based tool for studying non-periodicity," *Computers & Mathematics with Applications*, vol. 60, no. 3, pp. 634-641, 2010.
- [66] L. Chua, M. Komuro, and T. Matsumoto, "The double scroll family," *IEEE transactions on circuits and systems*, vol. 33, no. 11, pp. 1072-1118, 1986.
- [67] A. Buscarino, L. Fortuna, M. Frasca, L. V. Gambuzza, and G. Sciuto, "Memristive chaotic circuits based on cellular nonlinear networks," *International Journal of Bifurcation and Chaos*, vol. 22, no. 03, p. 1250070, 2012.
- [68] M. Itoh and L. O. Chua, "Memristor oscillators," *International journal of bifurcation and chaos*, vol. 18, no. 11, pp. 3183-3206, 2008.
- [69] L. O. Chua and G.-N. Lin, "Canonical realization of Chua's circuit family," *IEEE transactions on Circuits and Systems*, vol. 37, no. 7, pp. 885-902, 1990.
- [70] B. Karakaya, A. Gülten, and M. Frasca, "A true random bit generator based on a memristive chaotic circuit: Analysis, design and FPGA implementation," *Chaos, Solitons and Fractals*, vol. 119, pp. 143-149, 2019.
- [71] Ł. Czerwinski and M. J. Ogorzałekt, "Study of Bifurcation Phenomena in Memristor Oscillators," *International Symposium on Nonlinear Theory and its Applications NOLTA 2011*, pp. 244-247, 2011.
- [72] L. G. de la Fraga, E. Torres-Pérez, E. Tlelo-Cuautle, and C. Mancillas-López, "Hardware implementation of pseudo-random number generators based on chaotic maps," *Nonlinear Dynamics*, vol. 90, no. 3, pp. 1661-1670, 2017.

ÖZGEÇMİŞ

Barış KARAKAYA

Fırat Üniversitesi

Mühendislik Fakültesi

Elektrik-Elektronik Mühendisliği Bölümü

Elazığ

0 424 237 0000/5234

E-posta: bkarakaya@firat.edu.tr

- 1990 : Elazığ'da doğdu.
- 2004-2008 : Elazığ Korgeneral Hulusi Sayın Lisesi'ni bitirdi.
- 2008-2012 : Fırat Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümünde lisans eğitimini tamamladı.
- 2012 : Fırat Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümünde araştırma görevlisi olarak göreve başladı. Bu tarihten itibaren Fırat Üniversitesi, Mühendislik Fakültesi, Elektrik-Elektronik Mühendisliği Bölümünde araştırma görevlisi olarak görev yapmaktadır.
- 2012 : İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik & Haberleşme Mühendisliği Ana Bilim Dalı'nda yüksek lisans eğitimine başladı.
- 2014 : İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik & Haberleşme Mühendisliği Ana Bilim Dalı'nda yüksek lisans eğitimini tamamladı.
- 2015 : Fırat Üniversitesi, Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Ana Bilim Dalı, Devreler ve Sistemler Bilim Dalı'nda doktora eğitimine başladı.

- 2018 : Haziran-Aralık 2018 tarihleri arasında Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) 2214-A Yurt Dışı Doktora Sırası Araştırma Burs Programı kapsamında Department of Electrical Electronics and Information (DIEEI), University of Catania’da bursiyer araştırmacı olarak görev yaptı.
- 2018 : Fırat Üniversitesi Bilimsel Araştırma Projeleri (FÜBAP) kapsamında M.F.18.15 numaralı Doktora Tezi Araştırma Projesi’nde Yardımcı Araştırmacı olarak görev aldı.

TEZ KONUSUNDA YAPILAN ÇALIŞMALAR

- **B. Karakaya**, A. Gülten, “FPGA-based Trivium Algorithm for Random Number Generation”, *International Conference on Life Science and Engineering ICLSE*, 2016.
- **B. Karakaya**, M.E. Şahin, H. Güler, A. Gülten, “Memristör Tabanlı Kaotik Devre Modelinin FPGA Ortamında Tasarımı ve Gerçeklenmesi”, *Fırat Üniversitesi Fen Bilimleri Dergisi*, vol. 29, no. 2, pp. 39-44, 2017.
- **B. Karakaya**, M.E. Şahin, H. Güler, A. Gülten, “Memristor Based True Random Number Generator”, *6th International Conference on “Advances in Engineering and Technology” (AET-17)*, 2017.
- M.E. Şahin, **B. Karakaya**, H. Güler, A. Gülten, “Memristor Based Chaotic Circuit Modelling with LabVIEW”, *International Advanced Researches & Engineering Congress (IAREC 2017)*, 2017.
- **B. Karakaya**, V. Çelik, A. Gülten, “Chaotic cellular neural network-based true random number generator”, *International Journal of Circuit Theory and Applications*, vol. 45, no. 11, pp. 1885-1897, 2017.
- **B. Karakaya**, V. Çelik, A. Gülten, “Implementation of Two Cell Non-Autonomous Cnn Model on FPGA”, *Fırat University Turkish Journal of Science & Technology*, vol. 13, no. 1, pp. 25-29, 2018.
- **B. Karakaya**, V. Çelik, A. Gülten, “Realization of Delayed Cellular Neural Network Model on FPGA”, *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT)*, 2018.

- **B. Karakaya**, M.A. Turk, M. Turk, A. Gulden, “Selection of optimal numerical method for implementation of Lorenz Chaotic system on FPGA”, *International Advanced Researches and Engineering Journal*, vol. 2, no. 2, pp. 147-152, 2018.
- **B. Karakaya**, A. Gülten, “3D Chaotic System based Random Bit Generation”, *International Engineering and Natural Sciences Conference (IENSC)*, pp. 1592-1599, 2018.
- **B. Karakaya**, A. Gülten, M. Frasca, “A True Random Bit Generator based on a Memristive Chaotic Circuit: Analysis, Design and FPGA Implementation”, *Chaos, Solitons and Fractals*, vol. 119, pp. 143-149.

DİĞER ÇALIŞMALAR

- **B. Karakaya**, R. Yeniçeri, M.E. Yalçın, “Wave computing core using fixed-point arithmetic”, *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1514-1517, 2015.
- E. Özpolat, **B. Karakaya**, T. Kaya, A. Gülten, “FPGA-based Digital Filter Design for Biomedical Signal”, *2016 XII International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, pp. 70-73, 2016.
- E. Özpolat, **B. Karakaya**, A. Gülten, “Filter implementation for ECG Signal on FPGA Chip”, *6th International Conference on “Advances in Engineering and Technology” (AET-17)*, 2017.
- E. Özpolat, **B. Karakaya**, A. Gülten, “FIR Filtre Tasarımı ve FPGA Ortamında Gerçeklenmesi”, *Fırat Üniversitesi Mühendislik Bilimleri Dergisi*, vol. 29, no. 2, pp. 269-275, 2017.
- **B. Karakaya**, T. Kaya, A. Gülten, “FPGA-based ANN Design for Detecting Epileptic Seizure in EEG Signal”, *Balkan Journal of Electrical and Computer Engineering*, vol. 6, no. 2, pp. 15-19, 2018.