ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

IMPLEMENTATIONS OF NOVEL CELLULAR NONLINEAR AND CELLULAR LOGIC NETWORKS AND THEIR APPLICATIONS

Ph.D. THESIS

Ramazan YENİÇERİ

Electronics and Communication Engineering Department

Electronics Engineering Doctorate Program

OCTOBER 2015

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

IMPLEMENTATIONS OF NOVEL CELLULAR NONLINEAR AND CELLULAR LOGIC NETWORKS AND THEIR APPLICATIONS

Ph.D. THESIS

Ramazan YENİÇERİ
(504092207)

Electronics and Communication Engineering Department

Electronics Engineering Doctorate Program

Thesis Advisor: Prof. Dr. Müştak Erhan Yalçın

OCTOBER 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YENİ HÜCRESEL DOĞRUSAL OLMAYAN VE HÜCRESEL LOJİK AĞLARIN GERÇEKLEMELERİ VE UYGULAMALARI

DOKTORA TEZİ

Ramazan YENİÇERİ
(504092207)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Doktora Programı

Tez Danışmanı: Prof. Dr. Müştak Erhan Yalçın

EKİM 2015

Ramazan YENİÇERİ, a Ph.D. student of ITU Graduate School of Science Engineering and Technology 504092207 successfully defended the thesis entitled **"IMPLEMENTATIONS OF NOVEL CELLULAR NONLINEAR AND CELLULAR LOGIC NETWORKS AND THEIR APPLICATIONS"**, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Müştak Erhan Yalçın** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. İsmail Serdar Özoğuz** Istanbul Technical University	.....
	Prof. Dr. Hulusi Hakan Kuntman Istanbul Technical University	.....
	Prof. Dr. Vedat Tavşanoğlu Isik University	.....
	Prof. Dr. Cüneyt Güzeliş Yasar University	.....

Date of Submission : **31 July 2015**
Date of Defense : **9 October 2015**

To my wife and daughter,

FOREWORD

From the beginning to the end of my doctorate, I always feel the enthusiasm of my advisor Prof. Müştak E. Yalçın. He never slows down the tempo of himself, of mine and also his research group. His scholarship and experience, now has been created an new science admirer. My special thanks will be permanent for him.

My wife, Vuslat, who is my adored half, has never withdraw her support and belief to me. Thank you very much my darling, for your hands tightly joint to mine. We always wish our bliss lasts forever in our family with our new member, our daughter, Hüma. Welcome my baby. Thank you for you incredible smiles that relieve my tiredness.

Prof. Vedat Tavşanoğlu has never spared his wisdom. Thank you professor for the long conversations which reshape my perception to many scientific subjects. Also, thank you Prof. Serdar Özoğuz for your quick and practical evaluations on critical subjects that I have encountered during my research.

I can not thank Prof. Muharrem Ök enough, the person who never leaves off guiding, always motivates for the science and enlightenment.

I owe everyone in our group, Embedded System Design Laboratory in ITU, which includes current and past members who are Prof. Berna Örs, Emre Göncü, Emrah Abtioğlu, Giray Başkır, Tuba Ayhan, Akif Özkan, Çağrı Bağbaba, Buse Ustaoğlu, Ahmet Arış, Selman Ergünay, Mehmet Tükel, Şahin Baş, Özen Özkaya, Onur Varol, Sercan Tunçay, Seyhan Çalışkan, Ercan Kalalı, Barış Karakaya and many others a debt of gratitude.

A period of my research has been completed in University of Notre Dame, with Prof. Wolfgang Porod and Gyorgy Csaba. Special thanks go to them and TÜBİTAK for grant from 2214a programme for this period and 2211 programme for the entire doctorate.

Finally, thank you very much dear reader, as you are interested in this thesis and are giving a value by reading it.

October 2015

Ramazan YENİÇERİ
Electronics Engineer, M.Sc.

TABLE OF CONTENTS

Page

ABBREVIATIONS

1D : One Dimensional
2D : Two Dimensional
3D : Three Dimensional
ACE : Analogic CNN Emulator Engine
ADD : Asynchronous Delay Doubler
CA : Cellular Automata
CFOA : Current Feedback OpAmp
CLN : Cellular Logic Network
CNN : Cellular Nonlinear Network
CNN-UM : Cellular Nonlinear Network Universal Machine
CNPN : Cellular Nonlinear Processor Network
CPU : Central Processing Unit
DDE : Delay Differential Equation
DE : Doppler Effect
DPR : Dynamic Partial Reconfiguration
DSP : Digital Signal Processor
EEPROM : Electrically Erasable Programmable Read Only Memory
EXOR : EXclusive OR
FPGA : Field Programmable Gate Array
FSR : Full Signal Range
GPU : Graphics Processing Unit
Gbps : Giga-bit per second
Gsps : Giga-sample per second
ICAP : Internal Configuration Access Port
ISE : Xilinx Integrated Synthesis Environment
LCL : Inductor-Capacitor-Inductor
LM311 : A kind of analog comparator
LUT : Look-Up Table
Mbit : Megabit
MG : Mackey-Glass
Msps : Mega sample per second
NPE : Nodal Processing Element
OpAmp : Operational Amplifier
RAM : Random Access Memory
RM : Reconfigurable Module
RP : Reconfigurable Partition
SDK : Software Development Kit
TRBG : True Random Bit Generator

LIST OF TABLES

Page

LIST OF FIGURES

Page

IMPLEMENTATIONS OF NOVEL CELLULAR NONLINEAR AND CELLULAR LOGIC NETWORKS AND THEIR APPLICATIONS

SUMMARY

This thesis is a consistent and coherent reorganization of studies on two topics of nonlinear systems. First topic includes Relaxation Oscillators and logic oscillators with similar behavior which are locally coupled and the resulting Cellular Nonlinear Networks (CNN) are utilized for a predictive motion planning algorithm. Nonlinear waves, especially autowave and traveling wave, have been studied and their system model, coupling schemes, parameters, and inputs generating both types of nonlinear waves are explained. The research covers two implementations of selected CNN and compares their digital circuit (FPGA prototyping), CPU simulation and GPU simulation performances. The research is focused on the Doppler Effect occurrence of the propagated nonlinear waves. A novel nonlinear wave propagation based feedback motion planning algorithm which utilizes the Doppler Effect and generates a prediction for the future state of target object has been proposed. The comparisons which reveals the effect of Doppler Effect are reported. The results prove that a tracker even slower than the target may catch it using the proposed algorithm. This new method of motion planning needs two layers of oscillator based CNNs. Two types of relaxation oscillators (one of them is a new model) and the logic oscillator have been tested for the algorithm.

Novel models of chaotic time-delay systems are introduced in the thesis as the second topic. The proposed binary output nonlinearity makes the oscillator generate a mono-scroll chaotic attractor. This thesis also proposes a generalization of the binary output nonlinear function, which is a quantized output nonlinearity. The generalized nonlinearity yields a multi-scroll attractor. Both systems are modelled as sampled-data models, because the binary delay lines are constructed by digital components (D-type flip-flops). The research on implementations of these oscillators has been expanded with binary inverting buffers (NOT gates) and asynchronous digital state machines. These systems successfully generate true random bit sequences without the need for post-processing. Up-to-date NIST's statistical test suite is used for the tests of bit sequences and successful throughput rates are reported. The jitter on the NOT gate based delay line is utilized as physical noise and all-digital implementation supported by the jitter also passed the statistical tests.

The thesis merges research parts and reorganize the outputs under four titles: cells, networks, implementations and applications.

YENİ HÜCRESEL DOĞRUSAL OLMAYAN VE HÜCRESEL LOJİK AĞLARIN GERÇEKLEMELERİ VE UYGULAMALARI

ÖZET

Bu tez, doğrusal olmayan sistemler ailesinden gevşemeli osilatörler, lojik osilatörler, zaman gecikmeli kaotik osilatörler; bu sistemlerden kurulan ağlar, bunların elektronik gerçeklemeleri ve uygulama alanlarında katkılar sunmaktadır.

Tez, iki hipotezi tartışır. Tezde, doğrusal olmayan dalga yayılımı için ortam olan iki boyutlu hücresel doğrusal olmayan ağlar, iki boyutlu hareket planlama problemlerinde hedefin gelecekteki durumlarını öngörmeye yarayan öznitelikler ürettiği gösterilmiştir. Ayrıca, zaman gecikmeli sistemlerde kullanılan, ürettiği ikili sembol dizileri gerçek rastgele bit dizisi olan, en az bir tane iki seviyeli çıkış veren geribesleme fonksiyonu vardır. İki hipotezli bu doktora çalışmasında, hücresel gevşemeli osilatör ağ uygulamaları ve zaman-gecikmeli kaotik osilatör gerçeklemeleri ağırlıklı araştırma sahaları olmuştur. Elde edilen çıktıların çoğu bu iki başlık altında toplanmıştır ve iki hipotez test edilmiştir.

Gevşemeli osilatörler ile ilişkili çalışmalar doktora başlangıcından sonuna kadar geçen süreye yayılmıştır. Başlangıçta hedeflenen yeni bir hücresel gevşemeli osilatör ağ modeline başarıyla ulaşılmıştır. Zaman gecikmeli kaotik sistemler ile ilişkili çalışmalar ise tez çalışmalarına sonradan dahil olmuş, sürenin orta ve son kısmında yoğun olarak yürütülmüştür. Özetin devamında, tezin yazım organizasyonuna göre ana bölümler ve alt bölümler kısaca anlatılacak ve aralarındaki ilişki sunulacaktır.

Giriş bölümünü takip eden ilk bölüm olan 'Hücreler' bölümünde beş osilatör modeli sunulmaktadır. İlk osilatör (Osilatör 1) çalışmalara referans olan gevşemeli osilatördür ve modelinde bir parça parça doğrusal fonksiyon bulunmaktadır. Bu fonksiyon, iki mutlak değer fonksiyonu ile gerçekleştirilebilir. Osilatör 2, yeni bir gevşemeli osilatör modelidir ve bu doktoranın orjinal önermelerindendir. Model yalnızca bir tane işaret (signum) fonksiyonu barındırır. Osilatör 3 ise lojik osilatör olmakla birlikte, Osilatör 1 ve 2'ye ait dinamik davranışın taklidini yapmaktadır. Kısaca, gevşemeli osilatörde mevcut iki durum değişkeninin birbirine yakın (tepe) değerlerde bulunduğu, biri pozitif diğeri negatif iki tepe durum, ve bunlar arasında farklı yörüngeler üzerinden gerçekleşen iki geçiş durumu, Osilatör 3'teki dört durum ile modellenmiştir. Lojik osilatörün, gevşemeli osilatöre davranışsal olarak benzetilerek sentezlenmesi tezin literatüre katkılarındandır. Osilatör 4 ise yeni bir zaman gecikmeli kaotik sistemi, önerdiği iki seviyeli çıkış veren bir doğrusal olmayan fonksiyon ile sunar. Modelinde bulunan doğrusal olmayan fonksiyonun seviye sayısı sistematik şekilde arttırılarak çok sarmallı çekici üreten kaotik model elde edilmiştir. Osilatör 5 olarak anılacak olan bu modelde doğrusal olmayan fonksiyonun genelleştirilmesi verilir. Yeni önerilen doğrusal olmayan foksiyonları ile hem Osilatör 4 modeli hem de Osilatör 5 modeli tezin literatüre katkılarındandır.

Üçüncü ana bölüm olan 'Ağlar'da, beş osilatörden ilk dördü kullanılmakta ve farklı iki tip ağ kurulmaktadır. Osilatör 1, 2 ve 3 ile hücresel doğrusal olmayan ağlar oluşturulmuş, Ağ 1, 2 ve 3 isimleri verilmiştir. Dördüncü osilatör (kaotik zaman gecikmeli osilatör) ile farklı bir tip ağ kurulmuştur. Ağ 1 referans modeldir ve tezde bilgilendirme amacıyla bulunur. Her üç ağ üzerinde, doğrusal olmayan dalgalardan, otodalga ve yürüyen dalganın üretilmesi ve yayılması gösterilmiştir. Ağ 2 ve Ağ 3 için otodalga ve yürüyen dalgaları üreten bağlantı kuralları ve parametreler tezde önerilen yeniliklerdendir. Üç ağda aranan ilerleme, ardı ardına ve lokasyonu değişen kaynak ile üretilen yürüyen dalgaların, 2 boyutlu uzayda iç içe geçmiş ve Doppler Etkisini ortaya çıkarmış dalga çeperleri oluşturmasıdır. Çalışmalarda üç ağda da Doppler Etkisinin gözlenmesi başarılmıştır. Ağların hücreleri otonom osilasyon yapan dinamikte iken otodalga yayılmakta, tezde açıklanan kurallar ile çift kararlı (bistable) dinamiğe sahip kılındıklarında ise yürüyen dalga yayılabilmektedir.

Ağ 1, 2 ve 3, beş farklı metrik ile karşılaştırılmıştır. Karşılaştırma esnasında hücreler çift kararlı davranışa ayarlanmış, yürüyen dalga yayılmıştır. Metrik 1, dalga çeperi geçiş periyodu olan d büyüklüğünün çözünürlüğüdür. Ağ 3 neredeyse 2 değere nicelenmiş d üretebilir, Ağ 2 dört farklı değerde, Ağ 1 çok daha fazla değerde d üretebilir. Tez, Doppler Etkisinin sonucu olarak kaynak hareketi ile ilişkilenen d değişkeninin analizini uygulama kısmında kullanır. Dolayısyla, d'nin niceleme seviyesindeki fazlalık, analiz işleminde sonuçların keskinliğini etkiler.

Metrik 2 elektronik gerçekleme karmaşıklığıdır. Ağ 3'ün lojik devre olması sebebiyle, modele uygun gerçekleme az sayıda transistor ile mümkündür. Ağ 1 ve 2 ise sürekli zamanlı modellere sahip olduğundan analog devre olarak gerçeklenebilir. Modele uygun, yüksek doğrulukta çalışacak, gerçeklemenin karşılaştırıcı, toplayıcı, integre edici, kuvvetlendirici, çoklayıcı gibi bileşenleri çok sayıda transistor gerektirir. Ağ 1 gerçeklemesi, daha karmaşık olan doğrusal olmayan fonksiyonu sebebi ile Ağ 2 gerçeklemesinden karmaşık olacaktır.

Metrik 3 uzaysal-zamansal çalışma bölgesinde ağ üzerinde yayılan dalga çeperlerinin yayılma hızıdır. Sürekli zaman modelli Ağ 1 ve Ağ 2'de hız saniye birim zamanda değerlendirilirken, ayrık zamanlı Ağ 3'te hız iterasyon adımına göre değerlendirilmektedir. Ancak, modellerin sayısal yöntemler ile çözümü, her üçünü de ayrık zamanlı ve kaşılaştırılabilir hale getirir. Buna göre Ağ 3 en hızlı dalga yayılan ağdır. Ağ 2'de de Ağ 1'e göre daha hızlı dalga yayılır.

Metrik 4, ağdaki hücrelerin (1 ve 2'de) eyer noktaları arasındaki hareketlerinde geçen süre ve (3'te) tepe durumlar arasındaki hareketlerinde geçen süredir. Metrik 3'teki gibi, yeni durumuna en hızlı yerleşen hücreler Ağ 3'tekiler, daha yavaş yerleşenler Ağ 2'dekiler, ve en yavaş yerleşenler Ağ 1'dekilerdir. Yerleşme hızı, giriş işareti ile yeni dalga yaratma sıklığını üstten sınırlandıran bir büyüklük olarak değerlendirilmelidir.

Yayılan dalga çeperlerinin eğriliği Metrik 5'tir. Ağ 3'te yayılan yürüyen dalga ve otodalga çeperleri dörtgen şeklindedir. Ağ 2'te otodalgalar dörtgen şekilde yayılırken, yürüyen dalga için parametre araştırmasında, uygulanan bir ofset ile sistem dinamiği sekizgen dalga çeperi üretecek hale getirilmiştir. Ağ 1 çember şekle sahip dalga formları yayabilmesi sayesinde diğer ikinsine göre uygulamalarda avantajlı konuma gelmektedir.

Ağlar ana bölümünün içerdiği son ağ bir boyutlu, tek yönlü bağlantıya sahip zaman gecikmeli hücrelerden kurulu ağdır. Bu ağ, kaotik osilatörler arasında sezgisel (anticipating) senkronizasyonun kurulabildiğini göstermektedir.

Takip eden ana bölümde Hücreler ve Ağlar bölümünden modellerin bir kısmının gerçeklemesi için yapılan çalışmalar sunulmaktadır. Ağ 1'in ileri Euler metodu ile ayrıklaştırılmış hali sayısal sistem olarak tasarlanmış ve seçilen Sahada Programlanabilir Kapı Dizisi (Field Programmable Gate Array, FPGA) üzerinde gerçeklenmiştir. Yapılan gerçeklemede, 2008'de gerçeklenen kayan nokta sayı formatıyla çalışan aritmetik devreler yerine sabit nokta aritmetiği kullanılmıştır. Devrenin çalışma performansı ve FPGA üzerinde kapladığı alan açısından referans tasarım ile karşılaştırması sunulmuştur. Ayrıca, Grafik İşleme Birimi (Graphics Processing Unit, GPU) üzerinde yine Ağ 1 modeline ilişkin benzetim sonuçları elde edilmiştir ve gerek Merkezi İşlem Birimi (Central Processing Unit, CPU) üzerinde çalışan benzetimlerden, gerek FPGA gerçeklemelerinden daha yüksek performans elde edilmiştir.

Ağ 3'ün gerçeklemesi FPGA'larda var olan ve günümüzde hala geliştirilmekte olan bir özelliğin ağ gerçeklemesine katkısı incelenerek yapılmıştır. Dinamik Kısmi Yeniden Yapılandırma (Dynamic Partial Reconfiguration, DPR) adlı bu özellik, ile sayısal devrenin bir kısmı çalışırken diğer bir kısmı değiştirilebilir. Bu özellik, Ağ 3'ün bazı hücrelerinin çalışma esnasında değiştirilmesi sağlanacak şekilde kullanılmıştır. Elde edilen sonuçlara göre, FPGA alanından tasarruf sağlanmış fakat öte yandan yalnızca özelliğin aktif tutulmasını sağlayan ek alan tüketimi sorunu da tespit edilmiştir.

Bu doktora çalışmasındaki elektronik gerçeklemelerin çoğunluğu zaman gecikmeli sistemler (Osilatör 4, 5) ve ağları (Ağ 4) için yapılmıştır. İki seviyeli doğrusal olmayan fonksiyonla önerilen yeni modelin en büyük avantajı gecikme hattının gerçeklenmesinde görülür. Sayısal devre elemanlarından DEĞİL kapısı (evirici tampon, inverting buffer) ve tutucular, özellikle D tipi tutucu (flip-flop) ile ikili işaretler geciktirilebilir. Senkron tutucular ile yapılan gerçeklemede örneklemeli (sampled-data) sistem modeli kullanılması uygun olur. Bu ana başlık altında anlatılan gerçeklemenin ilki hem DEĞİL kapısı gibi asenkron cevap verebilen (saat işaretsiz) hem de tutucu dizisi kadar uzun gecikme süresi sağlayabilen bir gecikme hattı yapıtaşıdır. Tezde, Asenkron Gecikme Çiftleyici (Asynchronous Delay Doubler, ADD) adı verilen bu yeni devre ile iç içe kullanım sayesinde üstel artan gecikme süreleri elde edilebilmiş, bu sayede zaman sabiti büyük olan ayrık analog integrator devrenin ihtiyaç duyduğu uzun gecikme sağlanabilmiştir. Osilatör 4'ün analog integrator, D tipi tutucu gecikme hattı gerçeklemesi; analog integrator, ADD gecikme hattı gerçeklemesi; sayısal integrator, DEĞİL kapısı gecikme hattı gerçeklemesi aynı ana bölümde alt bölümler olarak sunulmaktadır. Bunları Osilatör 5'in analog integrator, D tipi tutucu gecikme hattı gerçeklemesi; Ağ 4'ün analog integrator, D tipi tutucu gecikme hattı geçeklemesi ve yine Ağ 4'ün sayısal integrator, D tipi tutucu gecikme hattı gerçeklemesi takip eder.

Sonuçlardan önceki son bölüm olan 'Uygulamalar' ana bölümü, iki bölümden oluşur. İlkinde Ağ 1, kestirim yapılmaksızın geribeslemeli hareket planlama algoritmasında kullanılır. Ardından Doopler Etkisini ve onunla üretilen yeni özniteliği kullanan öngörülü geribeslemeli hareket planlama algoritması sunulmaktadır. Öngörülü planlama tezin içerdiği yeniliklerdendir. Geribeslemeli hareket planı, ayrıklaştırılmış uzayda uzayın her ayrık parçası için bir hareket vektörünün hesaplanmış olduğu

plandır. Uzayın, ayrıklaştırılmış olması sebebiyle hücresel doğrusal olmayan ağlarla modellenmesi mümkün olur. Bu ağlar üzerinde dalga hedef noktadan doğar. Dalga yayıldıkça, çeperin ulaştığı hücreler geliş açısını tespit ve kayıt ederek geribeslemeli hareket planı oluşturur. Bu yöntemde geribesleme ifadesinden kasıt, planlama için yayılan dalganın tüm ağa dağılması dolayısıyla modellenen fiziksel dünyanın tüm noktaları için çözümün bulunmuş olması, bu sayede hedefe giden yolların tek seferde, tüm hücreler için aynı anda tespit edilmesidir. Üretilen sonucu kullanan sistem rota üzerinde hata yapsa da elde edilen çözüm sayesinde yeniden hesaplamaya gerek kalmaksızın hedefe doğru ilerlemesi mümkün olmaktadır.

'Uygulamalar'daki bir diğer alt bölümde de zaman gecikmeli Osilatör 4'ün rasgele bit dizisi üretiminde kullanımı konusunda elde edilen araştırma sonuçları verilmiştir. Önerilen kaotik sistemlerin gecikme hattından çıkan bit dizisi rasgele sayı olarak kabul edilir ve NIST'in istatistiksel test ortamıyla dizi sınanır. Uygun düşük hızda yapılan örnekleme sonucunda testi başarıyla geçen bit dizileri elde edilebilektedir. Ayrıca sezgisel senkronizasyon sağlayan ağ ile Osilatör 4 tabanlı rastgele bit üreticisinin gelecekte üretteceği değerlerin önceden tespit edilebildiği gösterilmiştir.

Tez boyunca yürütülen çalışmalarda, yeni modeller, yenilikçi gerçeklemeler ve yeni uygulamalara ulaşılmıştır. Her ne kadar tez organizasyonu, hücreler, ağlar, gerçeklemeler ve uygulamalar bölümleriyle yapılmış olsa da içeriği oluşturan çalışmalar, farklı alt bölümlerin bir arada ele alındığı şekilde yürütülmüştür. Bu sebeple, tez çalışması boyunca yayınlanmış olan veya hakem değerlendirmesinde bulunan bildiri ve makaleler farklı alt bölümlerden parçalar ihtiva etmektedir. Çalışma süresince 8 uluslararası konferans bildirisi sunulmuş, 5 dergi makalesi ve 1 kitap bölümü yayınlanmıştır. Ayrıca henüz hakemlik süreci tamamlanmayan 1 dergi makalesi mevcuttur.

1. INTRODUCTION

This thesis is introduced through four steps. It starts with the purpose and then reviews the literature to construct the necessary background. They are followed by the hypothesis with achieved contributions. Introduction ends with outlining the thesis organization.

1.1 Purpose of Thesis

Besides the linear systems with narrower variety in dynamics, simple yet functional nonlinear systems provide diverse solutions to engineering problems. The overall aim of the thesis is to present a study to advance the knowledge about three specific types of nonlinear systems which are: relaxation oscillators, logic oscillator and time-delay chaotic oscillators. The contribution of the thesis comprises the mathematical models of individual systems, coupling schemes in order to constitute their networks, implementations and applications. Empowering these systems with novel functionalities together which implies expansion of the application field stands for the main motivation. At every stage of the study, implementability of the developed models has been regarded, although a limited number of the proposed systems have been implemented within this thesis. Hence, proposing and synthesizing processes of new systems and their networks comply with the implementability intention.

1.2 Literature Review

The lumped circuit approximation simplifies the electric circuit problems when the physical dimensions of the circuit is small enough to neglect the electromagnetic propagation. Although every electrical system is actually an electromagnetic system, the spatial behavior of the system is ignored when it is assumed to be a lumped circuit. An engineer needs such an approximation to be able to solve the problem easily within an acceptable error limit. Instead of ignoring the spatial features of the problem, discretizing the spatial dimensions also helps to suggest numerical solutions to the

problems. For example, modeling the three dimensional (3D) physical space with a regular grid form, then mapping the spatially discretized Maxwell's equations to this artificial discrete space establish the medium for numerical analysis or emulation. The given method is presented by Balsi in 1995 [7]. It is mainly based on a work by Roska in 1995, which investigates simulating nonlinear waves and partial differential equations using Cellular Neural Network (CNN) [8]. CNN has spread to many scientific fields, such as the wave simulation example given above, and becomes the template structure for spatial discretization, since its invention by Chua and Yang in 1988 [9].

Like many other engineering inventions, CNN is defined with its known applications too. Chua and Yang first targeted image processing and pattern recognition applications [10]. As CNN is inspired by cellular automata [11] for its network structure and artificial neuron models for its node (or cell) dynamics, image or two dimensional (2D) pattern processed by CNN has discrete-space continuous-time properties. In the following years, the neuron model has been changed to many different nonlinear systems, which causes CNN to be approved as Cellular Nonlinear Network [12]. Due to the advances in CMOS circuit technology, mixed-signal circuits has been started to employ in CNNs. Examples of mixed-signal implementations, which are called analogic in CNN field, are CNN-UM [13], ACE4k [14], ACE16k [15]. These implementations have given their place to digital only solutions in recent years [3, 16–18]. On the other hand, the application spectrum has been expanded to the control of the electromechanical systems, artificial pattern generation, biological system emulation, and motion planning. One of two focuses in this thesis is the motion planning using implementable CNN models which have oscillatory cells. Further literature review and background for this focus is presented at the beginning of the related chapter and sections.

There exist another focus in this thesis, which is random number generation, using implementable time-delay systems which exhibit chaotic oscillation. Beginning with the first-order Delay Differential Equation (DDE) modeling of some physiological systems by Mackey and Glass [19] in 1977, research on time-delay systems have gained momentum. The chaotic behavior of their system exhibits some relation to dynamical respiratory and hematopoietic diseases, and this points out the importance

of chaos control in medicine. The abstract model of time-delay chaotic systems consists of an integrator (which also sums signals), a delay line and a nonlinear feedback function. On the implementation side, Namajunas *et al.* introduced the first circuit for Mackey-Glass (MG) model in 1995 [20]. The exponential nonlinearity of MG model has been approximated by coupling two complementary junction field-effect transistors in a special way in this work. A saturated piecewise nonlinear feedback function has been used in time-delay chaotic system model by Lu in 1996 [21] and it has been analyzed as a cell of cellular neural networks with delay (DCNN) in [22] without any implementation. Tamasevicius *et al.* implemented the system using non-saturated piecewise nonlinear feedback function with the same delay line proposed in [20], which is assembled by T-type LCL filters, in 2006 [23]. Buscarino *et al.* widen the time-delay chaotic system implementations using a different odd-symmetric non-saturated piecewise nonlinear feedback function with cascaded low-pass second-order Bessel filters as delay line [24]. References indicate that the main bottleneck of the time-delay chaotic system implementations is the delay part. In order to overcome this bottleneck, a configuration of digital shift registers, which is coupled to the remaining analog part of the system with analog-to-digital converter (ADC) and digital-to-analog converter (DAC), has been introduced in 2012 [25]. From the implementation point of view, recent works draw the attention to the simplification of chaotic system which also covers the delay line design. Models with new nonlinearities, their implementations, and their successful application in true random bit generation have been covered within this thesis. Following chapters and sections include further related literature review and background.

1.3 Hypothesis and Contributions

2D Cellular Nonlinear Networks, which reveals the Doppler Effect of nonlinear wave propagation, provide features in order to predict the future positions of the target in 2D motion planning problems. In this thesis, the reader finds three cell models which are appropriate to be employed in CNN. The configuration of the first relaxation-oscillator cell in order to control the nonlinear wave propagation, the nonlinearity and the coupling scheme of the second relaxation oscillator cell, and control and coupling scheme of the third logic oscillator cell are the first level contributions of this

thesis. Revealing the Doppler Effect (DE) on the networks constructed by these three oscillators, employing the effect for prediction, then utilizing the prediction for increasing the tracking performance in 2D motion planning are the following contributions.

Furthermore, this thesis states that binary symbol sequences, autonomously generated by time-delay systems with at least one binary output feedback function, are true random bit sequences. The novel nonlinear feedback function, its generalization, simply implementable delay lines, sampled-data system model properly modeling the delay line, method for anticipating future states, and circuit implementations are the contributions related to this hypothesis.

1.4 Organization

The thesis is divided into four chapters, which are: Cells, Networks, Implementations, and Applications. Two relaxation oscillators, one logic oscillator, one chaotic oscillator with binary feedback function and a chaotic oscillator with generalized feedback function have been proposed, investigated, and presented in Chapter 2. In Chapter 3, the reader will find CNNs composed of two relaxation oscillators and the logic oscillator. Coupling methods, and their effects are given in related sections. A one dimensional (1D) network with unidirectional coupling scheme, which causes anticipating synchronization between the chaotic oscillators, is also proposed and presented in Chapter 3. Chapter 4 includes realization of some cells and networks from previous two chapters. Digital implementation of a relaxation oscillator based CNN on a Field Programmable Gate Array (FPGA) and its software implementation on a Graphics Processing Unit (GPU) is reported. A recent method for resource efficiency in FPGA implementations, which is called Dynamical Partial Reconfiguration (DPR), is studied for CNN composed of logic oscillator and reported in this chapter. Seven implementations related to time-delay chaotic oscillator and its network is also included to this Chapter. These implementations employ analog and digital parts in different configurations and yield impressive results. Applications which test the hypotheses and exhibit the results are organized in Chapter 5. Feedback motion planning enhanced with future state prediction of a moving target by Doppler Effect justifies the first hypothesis. On the other hand, up-to-date NIST statistical test suite,

justifies the randomness of bit sequences generated by proposed time-delay chaotic systems. A conclusion chapter resides after these four chapters including suggestions on possible future works.

2. CELLS

This chapter presents and investigates five nonlinear dynamical systems without any coupling. All the systems in this chapter have been evaluated as potential cells of proper networks. Two of them are continuous-time second-order nonlinear systems which exhibit relaxation oscillation behavior. One of these two is known as the simplest relaxation oscillator model proposed by Yalcin in 2008 [26]. Its alternative, which is a novel contribution of this thesis, explains the relaxation oscillation phenomenon with a simpler model in terms of the nonlinear function. These two relaxation oscillators are followed by the simplest 2-state logic oscillator. The third cell is a logic oscillator which is generalized to a 4-state one, thus some of its features are correlated with former relaxation oscillators. The last two are time-delay nonlinear systems which are capable of exhibiting chaotic oscillation. The main difference between these two cells is the nonlinearity where one type of nonlinearity causes mono-scroll attractor and the other causes multi-scroll attractor generation. Both mono-scroll and multi-scroll attractor generators are also modeled as sampled-data systems, which is the first time that chaotic time-delay systems are modeled as sampled-data system, according to the author's best knowledge. Besides sampled-data modeling, the time-delay cells have further novel contributions such as new binary output nonlinearity and its generalized form. This chapter is organized as follows. Two relaxation oscillators are presented in Section 2.1. Logical oscillator appears in Section 2.2. Section 2.3 explains the studies on time-delay chaotic oscillators.

2.1 Relaxation Oscillators

Relaxation oscillations were first observed by van der Pol in 1926. He discovered that a triode circuit may exhibit self-sustained oscillations in varying waveforms from almost sinusoidal to ones with sharp edges. As explained by Wang in 1999 [27], the period of the oscillation is proportional to the relaxation time (time constant) of the system

in the latter case. The term relaxation oscillation comes from this phenomenon. The properties of relaxation oscillations are defined by van der Pol as follows [28]:

1. The period of oscillations is determined by some form of relaxation time.
2. They represent a periodic autonomous repetition of a typical aperiodic phenomenon.
3. Drastically different from sinusoidal or harmonic oscillations, relaxation oscillators exhibit discontinuous jumps.
4. A nonlinear system with implicit threshold values, characteristic of the all-or-none law.

This section investigates two relaxation oscillator models. Subsection 2.1.1 is a reminder of Yalcin's model. In Subsection 2.1.2, a novel model is proposed which uses a signum function as the nonlinearity. For both systems, network models, parameters values, phase portraits and state signal graphs in time have been given in the following two subsections.

2.1.1 Oscillator using absolute value nonlinearity

This oscillator model originally inspired from ACE16k CNN chip [15] and proposed by Yalcin in 2008 [26]. The model has constructed CNNs in wave processing applications [3, 29, 30]. Below is the model which is called Oscillator 1 in this thesis.

Oscillator 1 is modelled by:

$$\begin{aligned} \dot{x}(t) &= \alpha x(t) + \beta y(t) + g(x(t)), \\ \dot{y}(t) &= \gamma x(t) + \varepsilon y(t), \end{aligned} \tag{2.1}$$

with the nonlinearity given by

$$g(x) = -\mu(|x+\lambda| - |x-\lambda| - 2x), \tag{2.2}$$

where $x, y \in \mathbb{R}$. The desired dynamics is obtained with parameters: $\alpha = 4.2$, $\beta = -3$, $\gamma = 1$, $\varepsilon = -0.5$, $\mu = -10$, $\lambda = 1$. In [26], the nonlinearity $g(\cdot)$ is given by a conditional expression. However, we prefer to write it using a basic special function, the absolute value, to be able to compare with other systems. The phase portrait with

nullclines and oscillator's limit cycle is plotted in Figure 2.1. Oscillating state in time domain is plotted in Figure 2.2. As shown in the figures, y-state slowly changes. When it across the threshold value, it causes x-state to change quickly. As in the definition, the relaxing quantity here is the y-state.

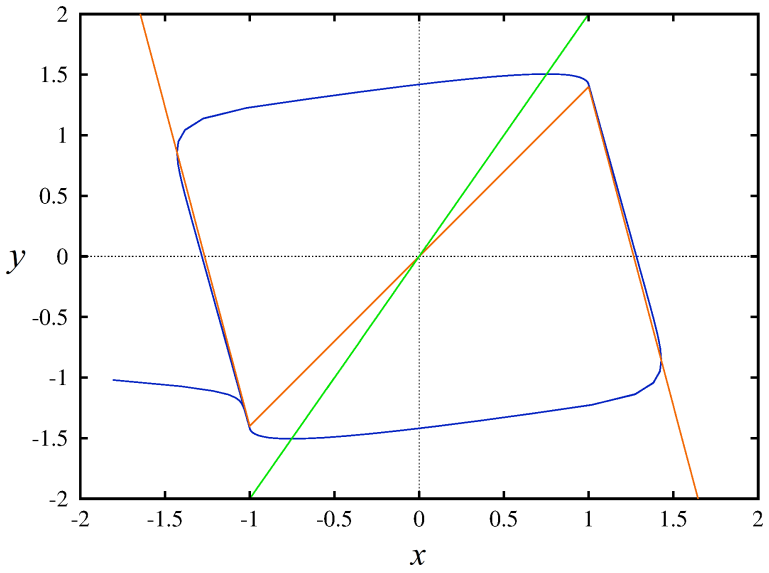


Figure 2.1 : Phase portrait of Oscillator 1.

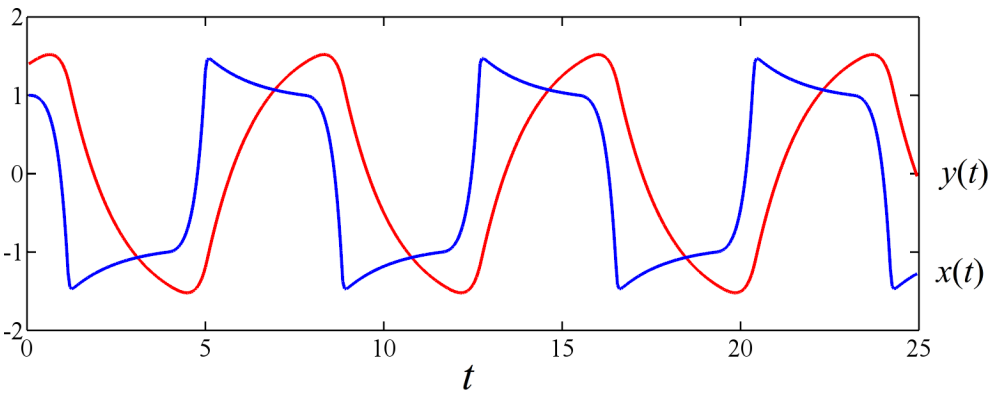


Figure 2.2 : $x(t)$ and $y(t)$ signals generated by Oscillator 1.

2.1.2 Oscillator using signum nonlinearity

In order to simplify the model, a new relaxation oscillator with a single signum function has been investigated. The absolute function can be decomposed into two signum functions with two multiplicative nonlinearities. Thus, in comparison with the nonlinearity of Oscillator 1, one signum function makes the system much simpler.

New model is called Oscillator 2 in this thesis. Oscillator 2 is modeled by

$$\begin{aligned} \dot{x}(t) &= (-\alpha+\mu)x(t) - \beta y(t) + g(x(t),y(t)), \\ \dot{y}(t) &= \gamma x(t) + \varepsilon y(t), \end{aligned} \tag{2.3}$$

with the nonlinearity given by

$$g(x,y) = -\mu \text{sign}(\alpha x + \beta y), \tag{2.4}$$

where $x, y \in \mathbb{R}$. The desired dynamics is obtained with parameters: $\alpha = 5.4$, $\beta = -6$, $\gamma = 2$, $\varepsilon = -2$, $\mu = -10$. This oscillator is simpler than Oscillator 1 as it has only one sign$(\cdot)$ nonlinearity. The phase portrait with nullclines and oscillator's limit cycle is plotted in Figure 2.3. State oscillation in time-domain is plotted in Figure 2.4.

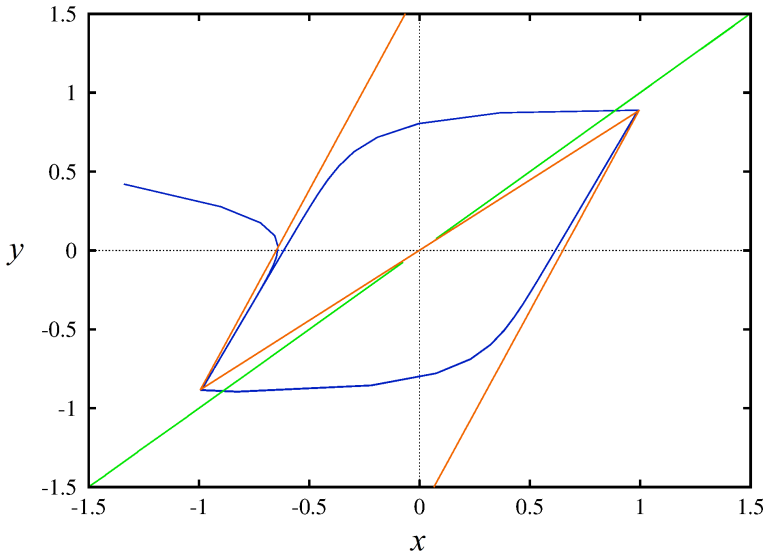


Figure 2.3 : Phase portrait of Oscillator 2.

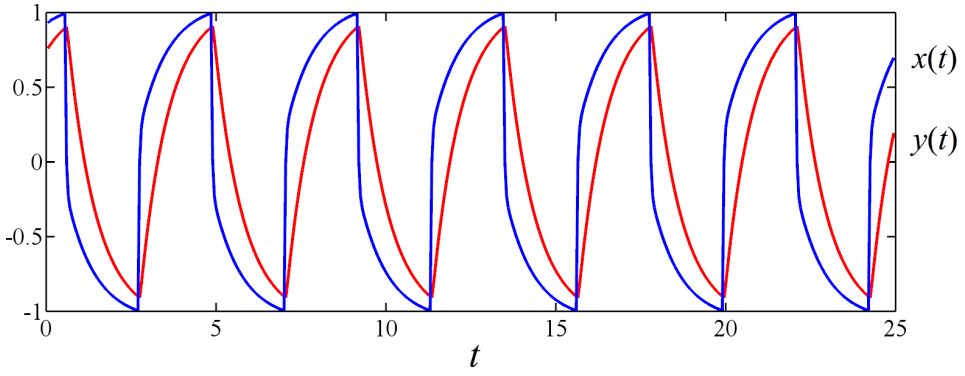


Figure 2.4 : $x(t)$ and $y(t)$ signals generated by Oscillator 2.

Further analysis on the effect of signum function is performed as follows: As $g(x,y) = \pm\mu$, x-nullcline is $y = ((-\alpha+\mu)/\beta)x \pm \mu/\beta$, which means x-nullcline is shifted up by μ/β in one half of the state space and shifted down by $-\mu/\beta$ in other half. The argument of signum function in g, which is $\alpha x + \beta y$, defines a line $y = (-\alpha/\beta)x$. Signum function divides the state space into two halves by this line. $g = \mu$ above the line and $g = -\mu$ below the line. Thus, x-nullcline is $y = ((-\alpha+\mu)/\beta)x + \mu/\beta$ above the line and vice versa below. As plotted in Figure 2.3, x-nullcline is shifted up by μ/β above the $y = (-\alpha/\beta)x$ line, and shifted down by $-\mu/\beta$ below.

As a result, $-\alpha/\beta$ is the slope of inner (middle) piece of x-nullcline, $(-\alpha+\mu)/\beta$ is the slope of outer (right and left) pieces of the nullcline. The breaking points are the intersections of inner and outer lines given by

$$\begin{aligned} y = (-\alpha/\beta)x &= ((-\alpha+\mu)/\beta)x \pm \mu/\beta, \\ -\alpha x &= (-\alpha+\mu)x \pm \mu, \\ x &= \mp 1, \\ y &= \pm\alpha/\beta. \end{aligned} \tag{2.5}$$

As parameter β is employed both in (2.3) and (2.4), slope of both inner and outer pieces are controlled by β. If β in (2.3) is switched by a new parameter $\phi > 0$, only slope of outer pieces change sign. Thus, the x-nullcline of Oscillator 2 resembles Oscillator 1's. In this case, the breaking point should be recalculated.

2.2 Logic Oscillator

The simplest digital oscillator is defined by its Boolean state equation

$$x(t_{k+1}) = x'(t_k), \tag{2.6}$$

where x' stands for logical inverse of $x \in \{0,1\}$ and t_k is the k-th sampling time. An abstraction may be done for relaxation oscillators in Subsection 2.1.1 and 2.1.2, such as two state variables approximate each other close to two different points in the state space and they follow two different trajectories in order to move between these two approximation regions. Abstraction leads one who needs to discretize the dynamics of relaxation oscillators to create 4 discrete states. Therefore, in order to mimic the given two relaxation-oscillators, the digital oscillator should have at least 4 states, in

which two states are for peaks and other two states are for the transitions between the peaks. Thus, rising and falling transitions in a relaxation oscillation may be represented separately. Because it is a logic system 4 states are coded by 2 binary state variables.

In (2.7), a simple 4-state logic oscillator, which is called Oscillator 3 in the thesis, has been given.

$$\begin{aligned} x(t_{k+1}) &= y(t_k) \\ y(t_{k+1}) &= x'(t_k) \end{aligned} \tag{2.7}$$

Figure 2.5 shows the state diagram of the Oscillator 3, whose state word is (x,y). It oscillates through the low peak state (00), rising state (01), high peak state (11), falling state (10), and continues to low peak state (00). State oscillation in time-domain is plotted in Figure 2.6.

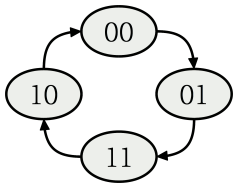


Figure 2.5 : State diagram of Oscillator 3.

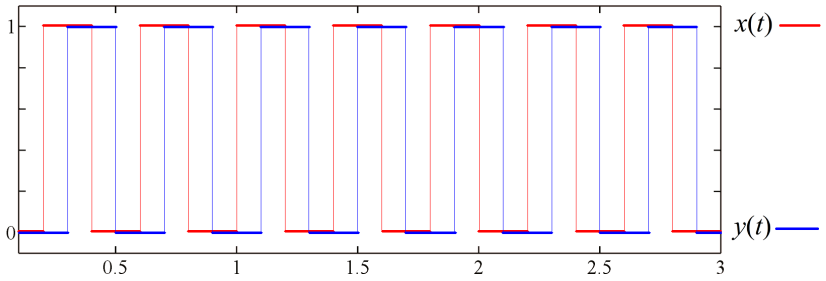


Figure 2.6 : $x(t)$ and $y(t)$ signals generated in Oscillator 3.

2.3 Time-delay Sampled-data Chaotic System

The research on chaotic behavior of a new time-delay systems is comprised in this section. In Subsection 2.3.1, a new binary function is employed in Mackey-Glass system. Then, the system is modeled as sampled-data system which is the first example for chaotic systems. Sampled-data modeling contributes to the realization of the proposed system. However, implementations are left to Chapter 4. The following

subsection 2.3.2 generalizes the system nonlinearity in order to generate multi-scroll chaotic attractor. In both subsections, the chaotic behavior are proved by numerically calculated largest Lyapunov exponents and bifurcation diagrams.

2.3.1 Generating mono-scroll attractor

There has been an increasing interest in DDE to generate chaotic dynamics since Mackey-Glass equation [19] describing physiological control systems was introduced. The first circuit realization to integrate Mackey-Glass equation was presented by Namajunas *et al.* [20] in 1995. Despite its simplicity, first-order nonlinear differential-delay equation of Mackey-Glass system can exhibit various dynamic behavior on a single state variable. Inspired by Mackey-Glass system, a new first-order nonlinear differential-delay equation with piecewise nonlinearity has been introduced by Uwe an der Heiden and M.C. Mackey [31] and its circuit realization was given by Losson *et al.* [32]. Another DDE with piecewise linear characteristic, motivated by Cellular Neural Network with delay, was introduced by Lu and He [21] and its chaotic behavior was experimentally verified by [22]. Autonomous and non-autonomous systems require to be at least third-order and second-order continuous-time systems, respectively, in order to exhibit chaos. However, a first-order continuous-time system with time-delay feedback can generate chaos. Although chaotic time-delay systems may be of smaller degree than the other types of systems, they need a particular component, the delay-line which requires attention when designing the system model. In this subsection a new time-delay system is proposed. Its delay line is modelled with sampled-and-hold blocks which converts the proposed system into a time-delay sampled-data system. Although sampled-data system is well-known in digital control systems and some related techniques have been recently used for synchronization of chaotic systems [33–35], a chaotic sampled-data system has yet not been introduced in the literature. One should notice that systems employing digital delay line, such as these in [25, 36], and systems employing a digital integrator [37] can also be modelled as sampled-data systems due to their hybrid (digital and analog) structure.

Consider the following delay differential equation:

$$\dot{x}(t) = -x(t) + \alpha f(x(t-\tau)) \tag{2.8}$$

where $\alpha \in R$ is the bifurcation parameter, τ is the delay and the nonlinearity $f(\cdot)$ (Figure 2.7) is given by

$$f(x) = \begin{cases} -1 & |x| \leq 1 \\ 1 & |x| > 1. \end{cases} \tag{2.9}$$

This equation with small difference has been used by Uwe an der Heiden and M.C.

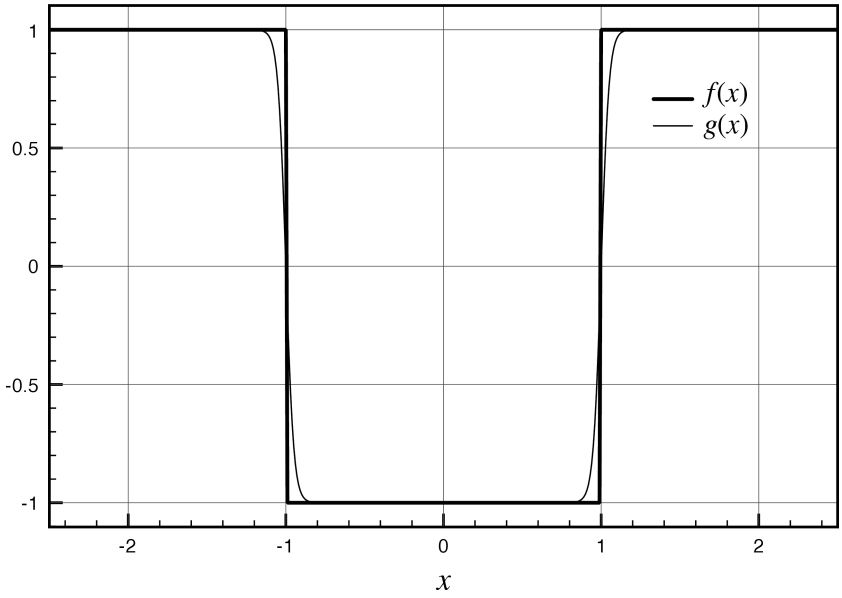


Figure 2.7 : The nonlinear function $f(x)$ and its approximate function $g(x)$ with $\gamma = 20$ given in (2.10).

Mackey [31] to model the processes in various areas of biology and chaotic dynamics of the system. Figure 2.8 shows the chaotic attractor of the presented system (2.8) in $x(t) - x(t-\tau)$ plane for $\alpha = 2$ and $\tau = 8$.

The nonlinearity of the system is a discontinuous function and it can be approximated by $\tanh(\cdot)$ functions such as

$$g(x) = \tanh(\gamma(-x-1)) + \tanh(\gamma(x-1)) + 1. \tag{2.10}$$

Figure 2.7 shows the $f(\cdot)$ function and its approximation $g(\cdot)$. The introduced system (2.8) with this smooth approximate function (2.10) evolves in the same type of chaotic attractor.

Here, $S = \{x(t) : x(t) = x(t-\tau)\}$ is defined as a Poincaré section for computing the bifurcation diagram. The bifurcation diagram of the system (2.8) versus parameter α is shown in Figure 2.9(a). For a quantitative measure of the system dynamics, Lyapunov exponent [38] versus the parameter α is displayed on the same figure (Figure 2.9(a))

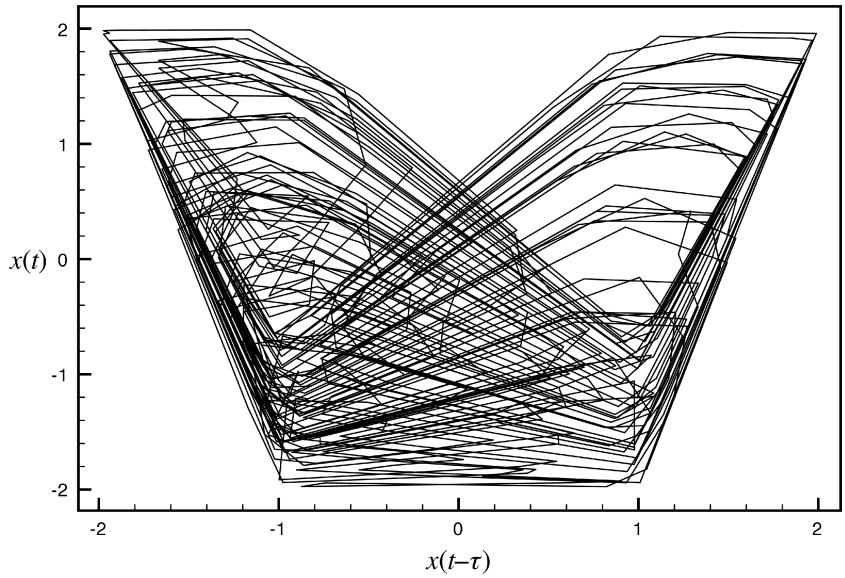


Figure 2.8 : Chaotic attractor of the system (2.8).

with the bifurcation diagram. These results show a good agreement between the values of the Lyapunov exponent and the observed bifurcation diagram. The presence of a positive Lyapunov exponent along with the observed strange attractor and bifurcation diagram indicates chaotic behavior in the system. As a result, the attractor of the introduced system (2.8), which is illustrated in Figure 2.8, is a chaotic attractor.

In order to understand the sensitivity of the dynamical behavior of the system (2.8) to the delay τ, the variation of the Lyapunov exponent versus the delay is displayed in Figure 2.9 (b) with the bifurcation diagram. Figures 2.9(a)-(b) indicate that the introduced system shows chaotic dynamics within a wide range of parameter values of α and τ.

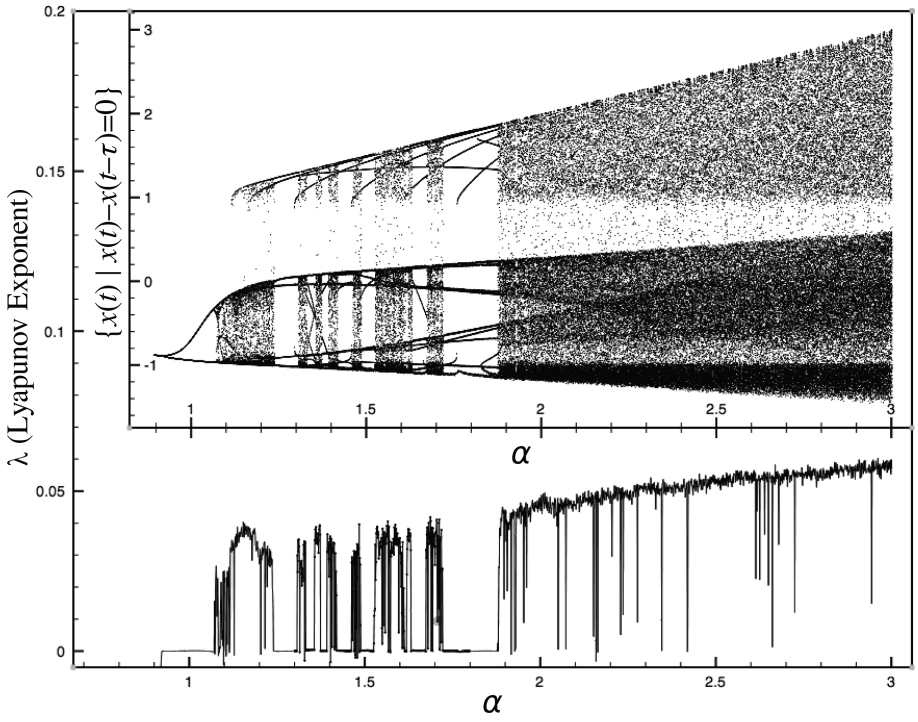
Next, a time-delay sampled-data system which is remodeled from the system (2.8) is introduced where the delay part of the system is composed of sample-and-hold blocks.

The block diagram of the system (2.8) is shown in Figure 2.10(a). The nonlinearity $f(x)$ of the system (2.8) is reformulated:

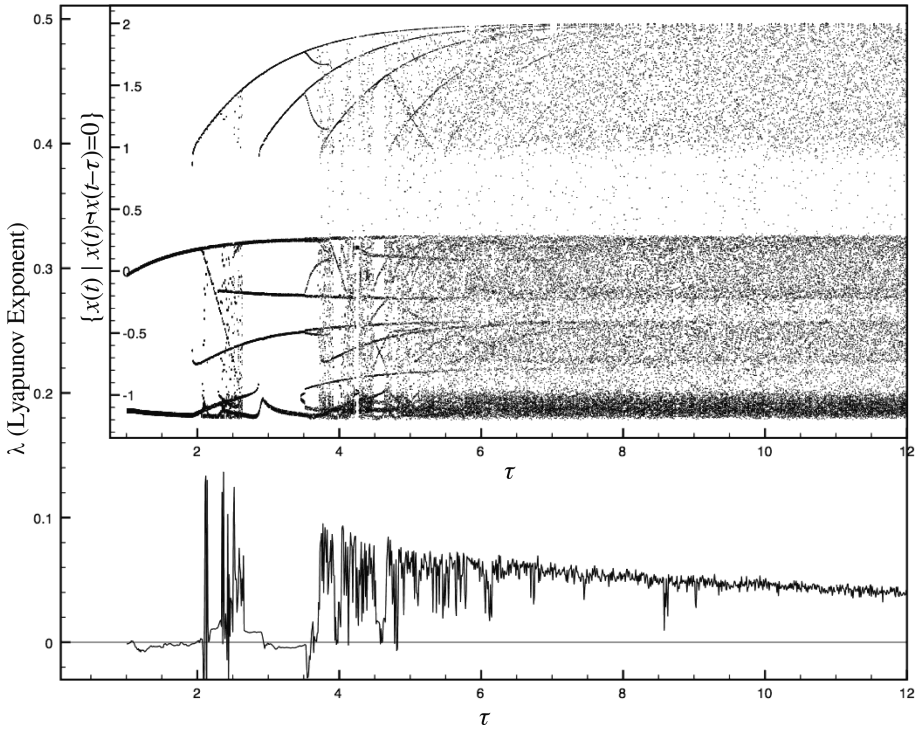
$$f(x) = f_c(h(x)) = \begin{cases} 1 & h(x) = 1 \\ -1 & h(x) = 0 \end{cases} \tag{2.11}$$

where $h(x(t))$ is a binary output function such that the output of function is 0 while $|x(t)| \leq 1$, else it is 1. This new formulation of the nonlinearity allows to implement $h(x)$ with two unit-step functions ($u(\cdot)$) such as

$$h(x) = u(x(t)-1) + u(-x(t)-1). \tag{2.12}$$

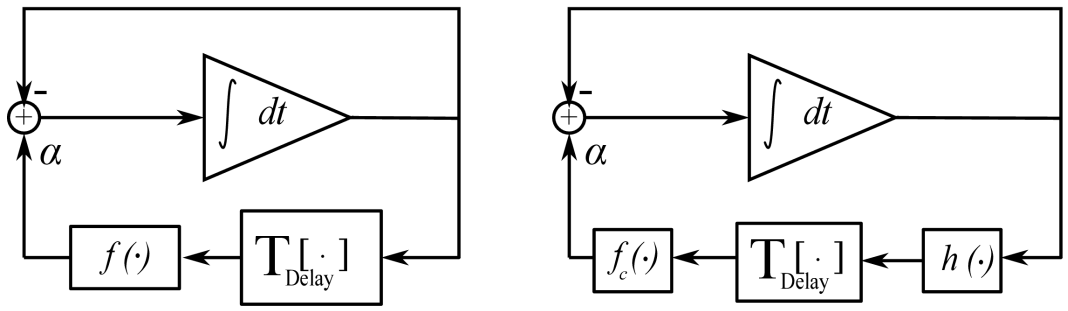


(a)



(b)

Figure 2.9 : Bifurcation diagram of the system (2.8) with the nonlinearity (2.10) versus a) α and b) the delay value τ. Figures also include spectrum of the Lyapunov exponent versus the same parameters. The observed dynamical behaviors of the system in bifurcation diagram and the calculated Lyapunov exponents are in a good agreement for both figures.



(a) (b)

Figure 2.10 : a) Block diagram of the system (2.8), b) the sequence of the blocks is rearranged in order to have binary input for the delay block. $f(\cdot)$ is decomposed into two functions $f_c(\cdot)$ and $h(\cdot)$ so that $f(\cdot) = f_c(h(\cdot))$. Then the delay block is placed after the function $h(\cdot)$ as in (2.12).

The nonlinearity of the system is a composition of the functions $h(\cdot)$ and $f_c(\cdot)$. Here, the function $h(\cdot)$ is the binary output function and nonlinear characteristic of the function $f(\cdot)$ can be realized by this function. The function $f_c(\cdot)$ given in (2.11) converts 0 to -1 and $f_c(1)$ is equal to 1.

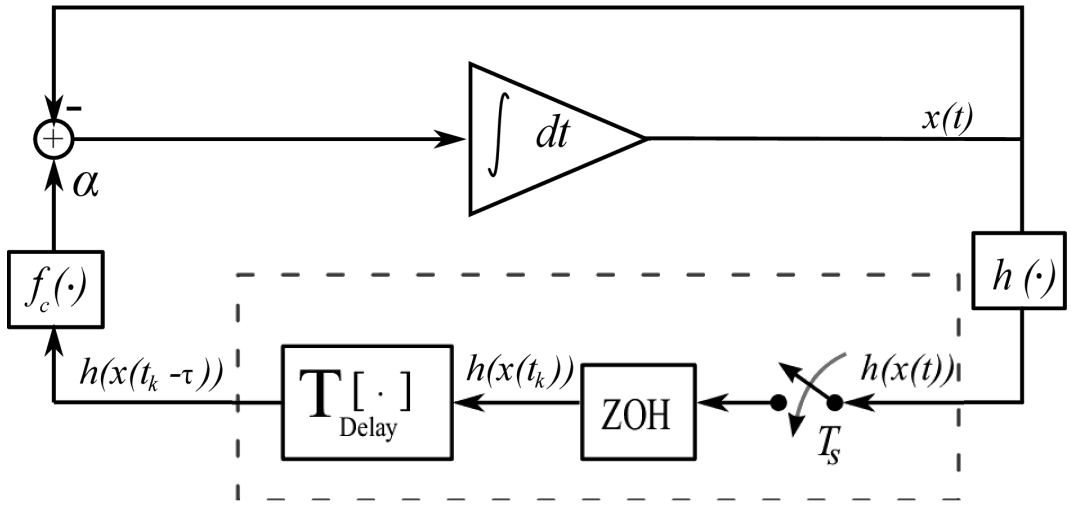


Figure 2.11 : Block diagram of the chaotic time-delay sampled-data system (2.15).

Furthermore, as $f(\cdot)$ is not a function of time, therefore one can write

$$f(\mathrm{T}_{\mathrm{Delay}}[x(t)]) = \mathrm{T}_{\mathrm{Delay}}[f(x(t))] \tag{2.13}$$

where $\mathrm{T}_{\mathrm{Delay}}[x(t)] = x(t-\tau)$. The above equation implies that the function of the delayed signal is equal to the delayed function of the signal. Therefore, the delay block $\mathrm{T}_{\mathrm{Delay}}[\cdot]$ can be placed after the nonlinear block which implements the function

$f(\cdot)$. Hence, the new block diagram representation of the system is shown in Figure 2.10(b). In this work, the delay block is placed just before $f_c(\cdot)$.

The block diagram of the system is given in Figure 2.11 after the sample-and-hold operation is introduced into the model in Figure 2.10(b). This type of system is known as sampled-data system [39], therefore the system given with block diagram in Figure 2.11 will henceforth be called chaotic time-delay sampled-data system. The signal at the delay line output is $h(x(t_k-\tau))$ (Figure 2.11) which is given by

$$h(x(t_k-\tau)) = \mathrm{T}_{\mathrm{Delay}}[h(x(t_k)]. \tag{2.14}$$

Thus, the mathematical model of the introduced chaotic time-delay sampled-data system is

$$\dot{x}(t) = -x(t) + \alpha f(x(t_k-\tau)), \quad t_k \le t < t_k + T_s \tag{2.15}$$

where T_s is the sampling period and t_k is the k^{th} sampling time. This system will also be called Oscillator 4 throughout this thesis. The system (2.15) has a similar strange attractor in $x(t)-x(t-\tau)$ plane for $\alpha = 2$, $\tau = 8$ and $T_s = 0.1$ as shown in Figure 2.12.

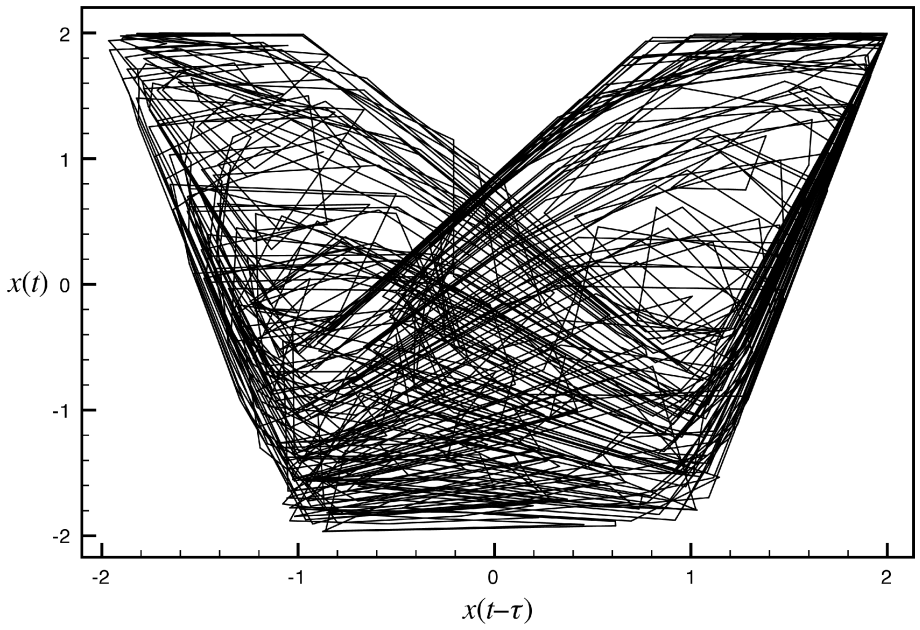


Figure 2.12 : Chaotic dynamics of the system (2.15) for $T_s = 0.1$ in $x(t)$-$x(t-\tau)$ space. The characteristic shape of strange attractors of the system (2.8) (see Figure 2.8) and the chaotic time-delay sampled-data system (2.15) for $T_s = 0.1$ are similar.

In order to examine the dynamics of the system (2.15), $S = \{x : x(t) = x(t-\tau)\}$ is again chosen as Poincaré section for computing the bifurcation diagram. Figure 2.13 shows the experimentally obtained bifurcation diagram versus parameter α for $T_s = 0.1$. Lyapunov exponent versus the parameter α is displayed on the same figure (Figure 2.13) with the bifurcation diagram. Figure 2.13 experimentally proves that the obtained attractor in Figure 2.12 is a chaotic attractor for the given parameter set. The Figures 2.9 and 2.13 show that the systems (2.8) and (2.15) have similar dynamical behavior. Specially, the system dynamics are almost identical for the small sampling period.

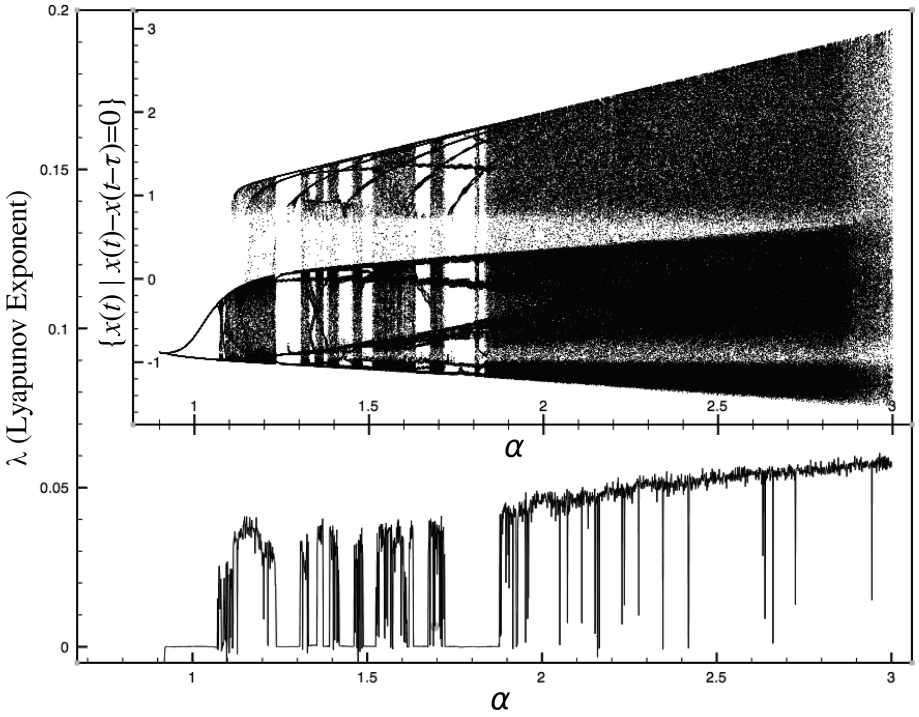


Figure 2.13 : Bifurcation diagrams of the system (2.15) with the nonlinearity (2.10), $\tau = 8$ and $T_s = 0.1$ versus α. Figure also includes the spectrum of the Lyapunov exponent versus the same parameters. The observed dynamical behavior of the system in bifurcation diagram and the calculated Lyapunov exponent are in a good agreement.

Dynamics of the sampled-data system depend on sampling period T_s. In the chaotic time-delay sampled-data system presented in (2.15), the signal is binary in the input of the zero-order hold block (see Figure 2.11), an error occurs when the input signal changes the state within the sampling interval. Otherwise the output of the delay line will be equal to the output of the ideal delay block ($h(x(t-\tau)) = h(x(t_k-\tau))$). The gray region in Figure 2.14 represents this situation. The input signal might change any time between the two samples, therefore $h(x(t-\tau)) = h(x(t_k-\tau))$ until a change in the

state occurs (for example, the output of the delay line will be correct $h(x(t-\tau))$ value between time t_2 and t_b (see Figure 2.14)). However, the output of the delay line will not be $h(x(t-\tau))$ between the time when the state change occurs and the next sampling time (t_b and t_3 in Figure 2.14, respectively). In fact, the output of the delay line will be $h(x(t_2-\tau))$ instead of $h(x(t-\tau))$ where $t_b \leq t \leq t_3$. This error appears since the $x(t)$ is a chaotic signal and/or $h(x(t))$ and sampling signal are not synchronized. The amount of error depends on the sampling period T_s.

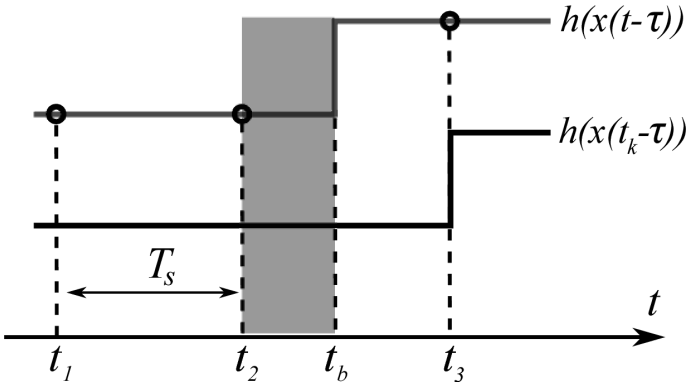


Figure 2.14 : The output of the ideal delay line $h(x(t-\tau))$ and the output of the sample-and-hold delay line $h(x(t_k-\tau))$ where $k=1,2,3$. $h(x(t_k-\tau))$ is obtained after the sampling of $h(x(t-\tau))$. These two signal will not be the same between t_b (when a state change occurs between two samples) and t_3 (the next sampling time).

Choosing the $S=\{x : x(t-\tau)=-1\}$ as a Poincaré section, the corresponding bifurcation diagram is depicted on Figure 2.15 to illustrate the behavior of the system with varying sampling period. The system keeps its chaotic regime while $T_s \leq 0.26$. In order to have similar dynamical behavior with the system (2.8), T_s should be chosen small. However, the system (2.15) exhibits chaotic dynamics depending on the bifurcation parameter α also. Figure 2.16 shows how the bifurcation parameter α effects the behavior of the system. In Figure 2.16, spectrums of Lyapunov exponent for $T_s=0.02$, $T_s=0.25$, $T_s=0.4$ and $T_s=0.6$ versus α are given in the same figure.

Lyapunov spectrums of the chaotic time-delay system (2.8) and the chaotic time-delay sampled-data system (2.15) for $T_s=0.02$ which are given in Figure 2.9(a) and Figure 2.16, respectively, indicate that these two systems show the same dynamical behavior. The system (2.15) might be in a chaotic region for a large T_s depending on α parameter (see Figure 2.16). However, a wide range of parameter values which keeps the system

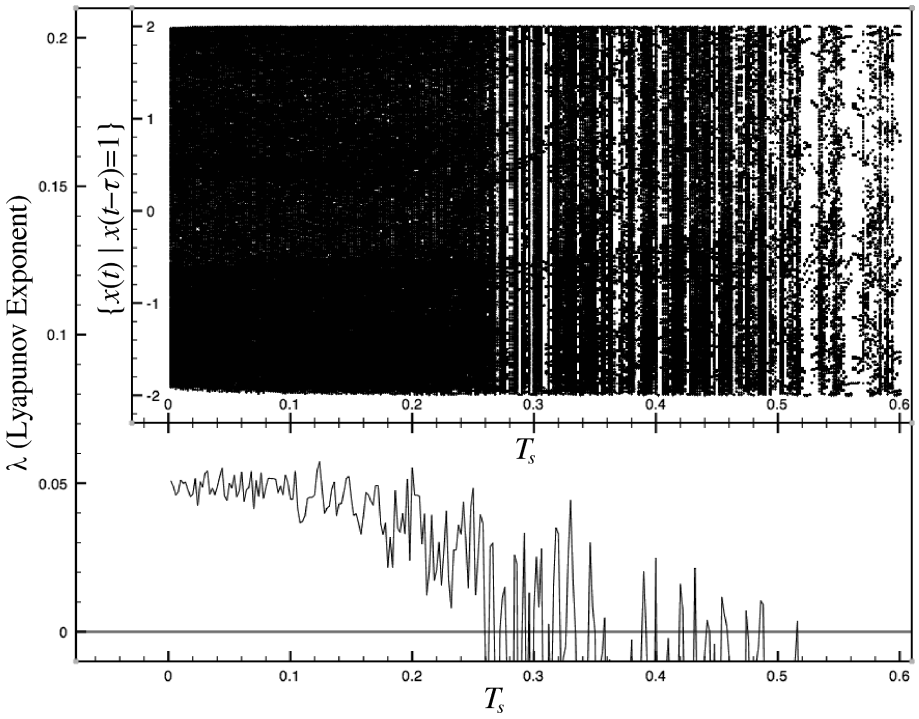


Figure 2.15 : Bifurcation diagram and the spectrum of Lyapunov exponent of the introduced system (2.15) with $\alpha = 2$ and $\tau = 8$ versus T_s.

(2.15) in chaos does not exists. In order to better understand the dynamics of the system (2.15), Lyapunov-exponent spectrum is calculated for both T_s and α, in the range of $[0.02,\ 1]$ and $[0.9,\ 3]$, respectively. The obtained Lyapunov exponents are displayed on $(T_s - \alpha)$-plane (Figure 2.17) by mapping the Lyapunov exponent which is larger than 0 to a black dot. Gray region indicates that the system produces a periodic motion.

2.3.2 Generating multi-scroll attractor

Since its invention, Chua's circuit [40] has become a paradigm for chaos studies. The simple 3-rd order nonlinear system using a 3-segment piecewise nonlinearity generates a double-scroll attractor in its 3-dimensional state space and has motivated the research in order to simplify the chaotic systems and increase the complexity of their behavior. The attention of scientists to multi-scroll chaotic attractors has considerably risen since the first *n*-scroll chaotic attractor was introduced by Suykens and Vandewalle in 1993 [41], which is the generalization of the original well-studied Chua's circuit. The systematic way of generating additional scrolls in the state-space is proposed as adding additional break points to the characteristics of nonlinear resistor. Recently, improved version of Chua's circuit has been employed for multi-scroll attractor generation [42].

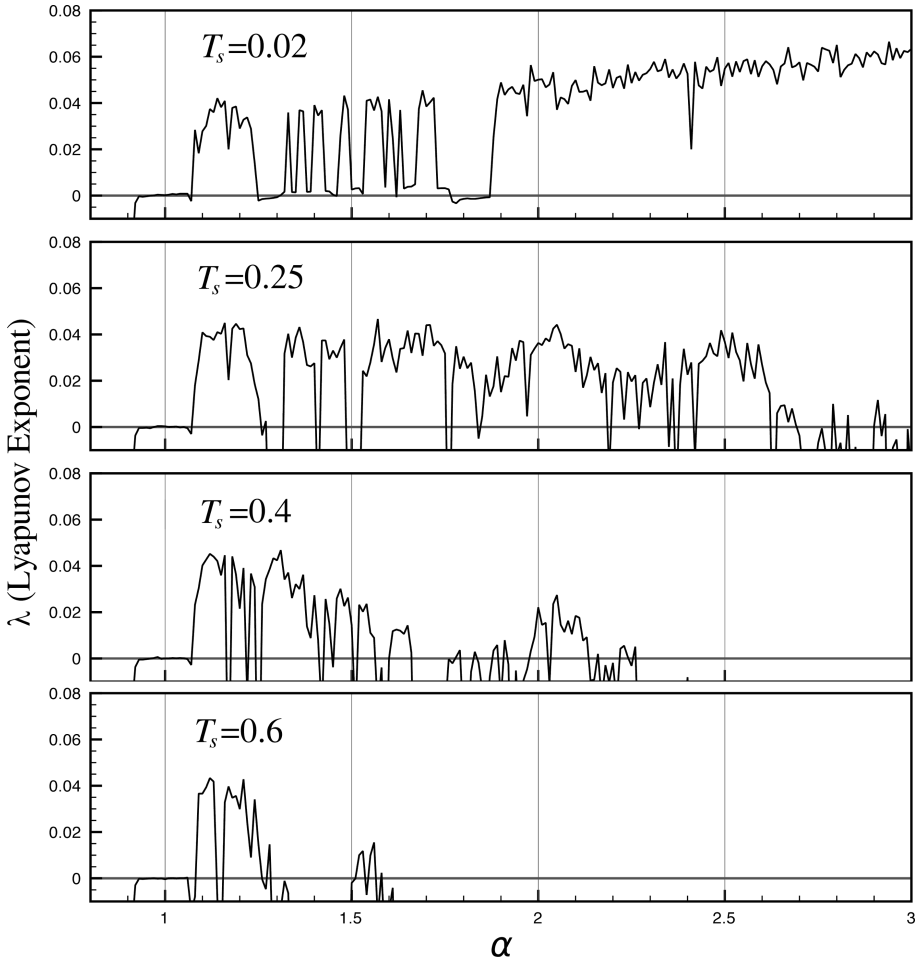


Figure 2.16 : Lyapunov exponent spectrums of the system (2.15) for $\tau = 8$ versus α calculated for $T_s = 0.02$, $T_s = 0.25$, $T_s = 0.4$ and $T_s = 0.6$.

A second chaotic system (later called Jerk circuit), which has been proposed by Elwakil and Kennedy in 2001 [43], has gained popularity due to its simple circuit implementation. Then, the research on multi-scroll attractors has yielded a generalization in Jerk circuit using hard limiter series [44]. Moreover, locating attractors in any state variable direction has become possible by the introduction of the family of scroll grid attractors in 2002 [45]. The given approaches for the generalization of the nonlinearity have been expanded by using hysteresis [46] and saturated circuit series [47]. In 2006, Deng and Lu have presented a systematic method for generating multi-directional multi-scroll chaotic attractors using fractional differential systems [48]. Besides autonomous systems, non-autonomous techniques for generating multi-scroll attractors are present in the literature, such as [49]. Stair-type and sawtooth functions, which are used in the implementation of multi-scroll chaotic generator, have been realized by floating gate-based CMOS structures in a

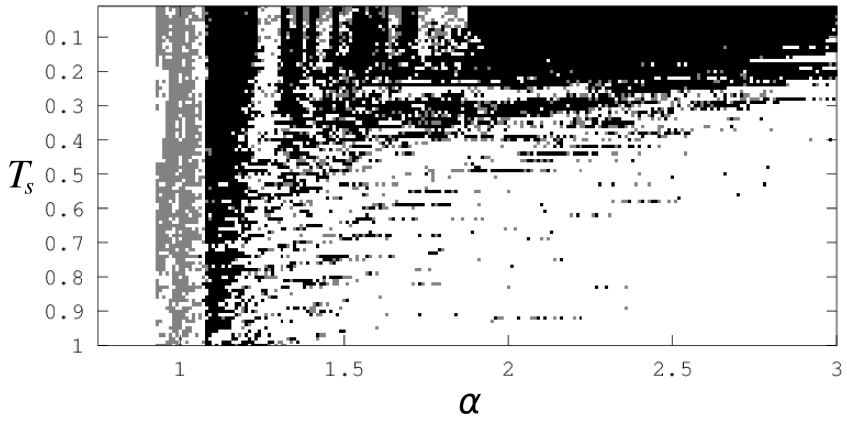


Figure 2.17 : Dynamical behavior of the time-delay sampled-data system (2.15) in $(T_s - \alpha)$-plane. Black region indicates that the system is in chaos. Gray region indicates that the system produces a periodic motion.

recent work [50]. Like stair-type or structure sawtooth functions, multi-segment quadratic functions are also capable of multi-scroll generation [51]. Since 2006, researchers find multi-scroll attractor generating systems based on variety of delay differential equations (DDE).

Since the work by Wang and Yang in 2006, multi-scroll attractors are realized also by time-delay systems [52]. Both [52] and [53] use the same cascaded LCL filter as delay line given in [20]. The main difference between [52] and [53] is that the nonlinear feedback function they used are multi-segment piecewise linear function and hard limiter function, respectively. Moreover, it has been shown that a modified nonlinearity of [53] with hysteresis function series also exhibits chaotic behavior in [54]. A chaotic system generating two double-scroll attractors has been introduced employing a three segment piecewise function, called threshold controller, by Srinivasan [55]. Besides the classical first-order systems with nonlinear delayed feedback function, multi-scroll attractors may also be generated by adding nonlinear delayed control term to higher order chaotic systems like Chen systems [56]. Another multi-scroll attractor generating system which uses the nonlinearity in [57] in the model given by [53] has been reported by Zhang in 2012 [58]. Zhang's work, using the delay line in [20], also suffers form the delay line implementation drawback. Apart from all the multi-scroll generating chaotic time-delay systems cited, Marquez *et al.* have recently proposed a novel system implementation, recently [37]. This new system has an optical delay line with Mach-Zehnder modulator in order to implement nonlinear feedback function and a

photodiode which connects the output of delay line to the electrical part. Moreover, the integrator is implemented by a digital signal processor (DSP).

In [52], a multi-scroll chaotic oscillator is proposed based on the first circuit realization of the Mackey-Glass model [20]. According to [52], the attractor in Figure 2.8 is a mono-scroll attractor that appears in a 'V' shape. If the mono-scroll attractor is considered with two wings, a double-scroll attractor consists of two mono-scroll attractors, one of them is in upside down position, like 'Λ', and one wing of each mono-scroll attractor overlap. For three scrolls 'V+Λ+V', a 'W' shape is yield. To generate multi-scroll attractor, Wang and Yang have used a piecewise linear feedback function [52]. Generation of multi-scroll attractor based on the system (2.15), i.e. Oscillator 5, with a parametrically quantized function is explained here.

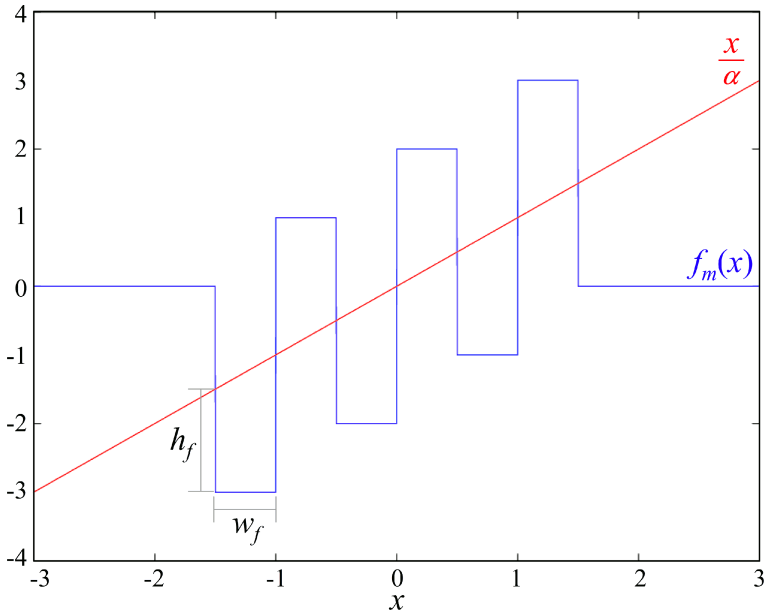


Figure 2.18 : A nonlinear function that generates multi-scroll attractor. It has 7 intersections with $\frac{x}{\alpha}$ line which separates the x-axis into 8 parts and generates 6-scrolls.

The proposed nonlinear function $f_m(x)$ for the multi-scroll attractor is discrete in amplitude and have discontinuous points. $f_m(x)$ which is composed by a few parameters and unit-step functions is given by

$$f_{\mathrm{m}}(x) = o_f + \left(h_f - \frac{n_f w_f}{\alpha}\right) + \sum_{k=0}^{2n_f} \left[\left(\frac{w_f}{\alpha} + (-1)^{k+1}\left(2h_f + \frac{w_f}{\alpha}\right)\right) \cdot u\left(x + w_f\left(n_f - k\right)\right) \right]$$

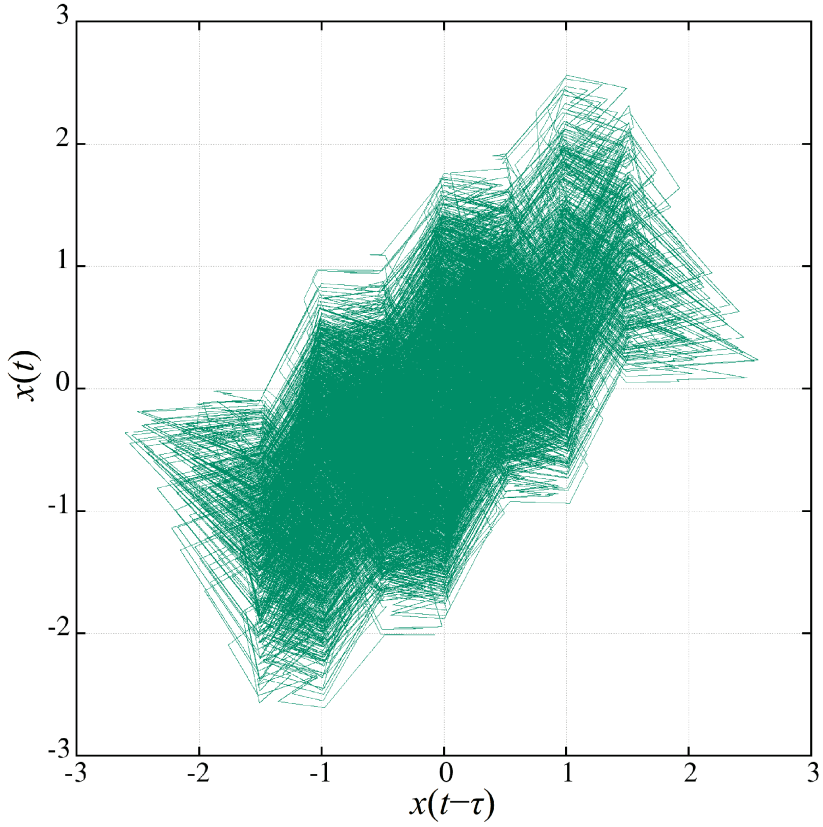


Figure 2.19 : A six-scroll attractor generated by the system (2.16).

where o_f is the offset, h_f is the absolute difference between $y = x/\alpha$ line and $f_m(x)$ at the left most discontinuity on $f_m(x)$, if $o_f = 0$. w_f is the horizontal distance between the points of discontinuity on $f_m(x)$. n_f is the parameter which determines the number of discontinuities. Number of discontinuities is equal to $2n_f + 1$, which separates x-axis into $2n_f + 2$ regions and generates $2n_f$-scrolls. Figure 2.18 depicts $f_m(x)$ with parameters $n_f = 3$, $o_f = 0$, $h_f = 1.5$, $w_f = 0.5$, and $\alpha = 1$.

Using the nonlinearity (2.16) in the system (2.15) yields the new multi-scroll chaotic system given by

$$\dot{x}(t) = -x(t) + \alpha f_m(x(t_k - \tau)), \quad t_k \leq t < t_k + \mathrm{T_s}. \tag{2.16}$$

Before further analysis, a six-scroll attractor is generated by the given system with a parameter set, $n_f = 3$, $o_f = 0$, $h_f = 1.5$, $w_f = 0.5$, $\alpha = 1$, $\tau = 20$, and $\mathrm{T_s} = 0.02$ whose trajectory is depicted in Figure 2.19.

Lyapunov exponents measure the exponential rates of divergence or convergence of nearby trajectories. Systems in a chaotic motion exhibit sensitivity to initial conditions which means trajectories should locally diverge away from each other [59].

Thus, the proposed system's behavior is analyzed by largest Lyapunov exponents and accompanying bifurcation diagram using the Poincaré section defined by $S = \{x(t) : x(t) = x(t-\tau)/\alpha\}$. Numerical calculation methods for largest Lyapunov exponent are given in [59–61]. Sprott's method which is for smooth systems is the basis of numerical analysis in this subsection. Although other methods, for example the one based on synchronization [62], appear for non-smooth systems, the estimation of Lyapunov exponents is not straightforward for them. Hence, a smooth approximation for the system (2.16) has been suggested, and the results obtained from the smooth version have been checked by the original model and the experimental circuit. An approximation of unit-step function is given by

$$u(x) = \begin{cases} 0, & x < 0; \\ 0.5 & x = 0; \\ 1, & x > 0; \end{cases} = \lim_{s\to\infty} \frac{\tanh(sx)+1}{2}. \tag{2.17}$$

Considering (2.17), the approximation of $f_m(x)$ is given by

$$\hat{f}_{\mathrm{m}}(x) = o_f + \left(h_f - \frac{n_f w_f}{\alpha}\right) + \sum_{k=0}^{2n} \left[\frac{1}{2}\left(\frac{w_f}{\alpha} + (-1)^{k+1}\left(2h_f + \frac{w_f}{\alpha}\right)\right) \cdot \left(\tanh\left(s_f\left(x + w_f\left(n_f - k\right)\right)\right) + 1\right)\right].$$

In (2.18), s_f parameter determines the slope of the *tanh* function at zero. For the largest Lyapunov exponents, bifurcation diagrams and phase portrait plots in Figure 2.20−2.25 in this subsection, $\hat{f}_{\mathrm{m}}(x)$ is used as the nonlinearity, and s_f is set to 50. Other parameter values are the same as Figure 2.19 unless it is the bifurcation parameter.

The offset parameter determines a bias for the scrolls. Positive offset values cause the scrolls on the negative half of the state space to disappear. Figure 2.20 shows the bifurcation diagram and largest Lyapunov exponents versus o_f. According to the figure, in the $-1 < o_f < 1$ range, the system is capable of generating chaotic attractors. However, the offset range is very tight $(-0.05, 0.05)$ for the desired multi-scroll attractor. Figure 2.20 also includes a phase portrait in which some of the scrolls have disappeared because of the high bias by o_f.

In order to have the desired multi-scroll attractor, the h_f parameter should be set in a proper range. Small values starting from 0 yield periodic motion in the state space. Also, very large values again come out with a limit cycle which surrounds the desired

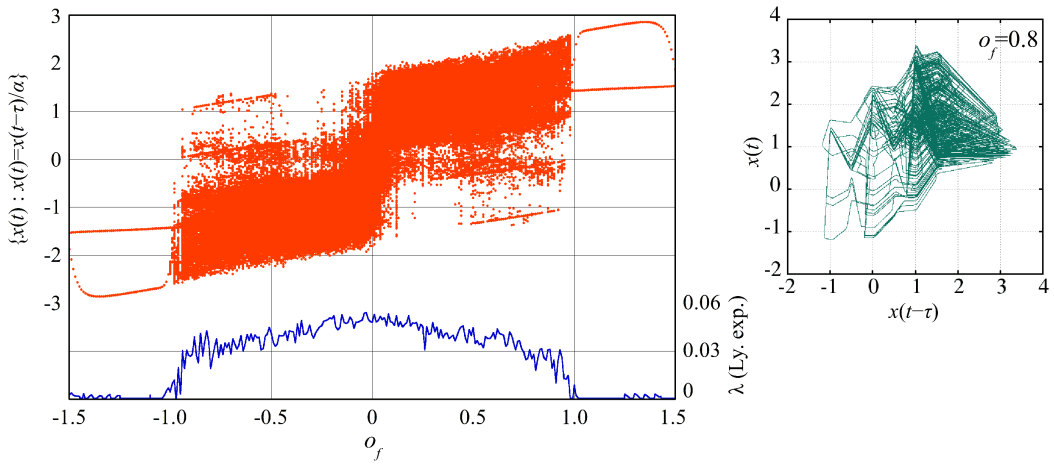


Figure 2.20 : Bifurcation diagram and accompanying largest Lyapunov exponent versus o_f is given on the left. A sample phase portrait for a chosen o_f value is given on the right.

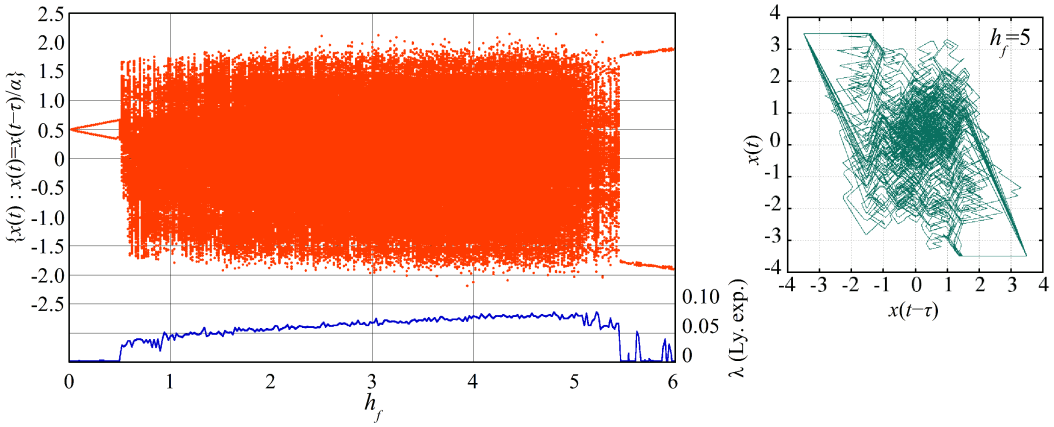


Figure 2.21 : Bifurcation diagram and accompanying largest Lyapunov exponent versus h_f is given on the left. A sample phase portrait for a chosen h_f value is given on the right.

multi-scroll attractor. For example, $h_f = 5$ in Figure 2.21 yields an attractor in which it is not possible to detect the scrolls. For the given parameter set, h_f around 1.5 provides a $2n_f$-scroll attractor.

Same with o_f and h_f, the width parameter w_f has a bounded range for the system to be in the chaotic motion which is shown in Figure 2.22. The width of scrolls increases with increasing w_f. For large values of w_f, it takes more time for trajectories to include every scroll. Despite the fact that multi scroll attractor can be better observed for the values of w_f around 0.5, it is still possible to observe them for $w_f = 1.2$ as shown by the phase portrait in Figure 2.22.

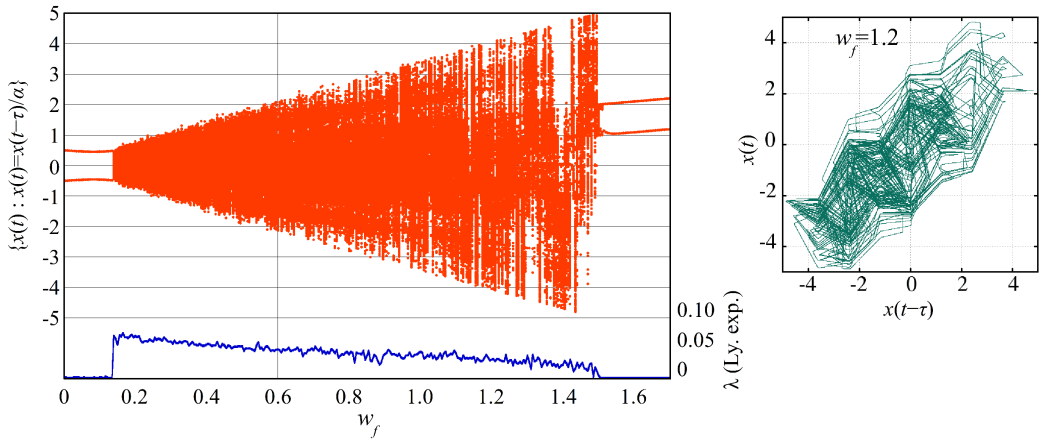


Figure 2.22 : Bifurcation diagram and accompanying largest Lyapunov exponent versus w_f is given on the left. A sample phase portrait for a chosen w_f value is given on the right.

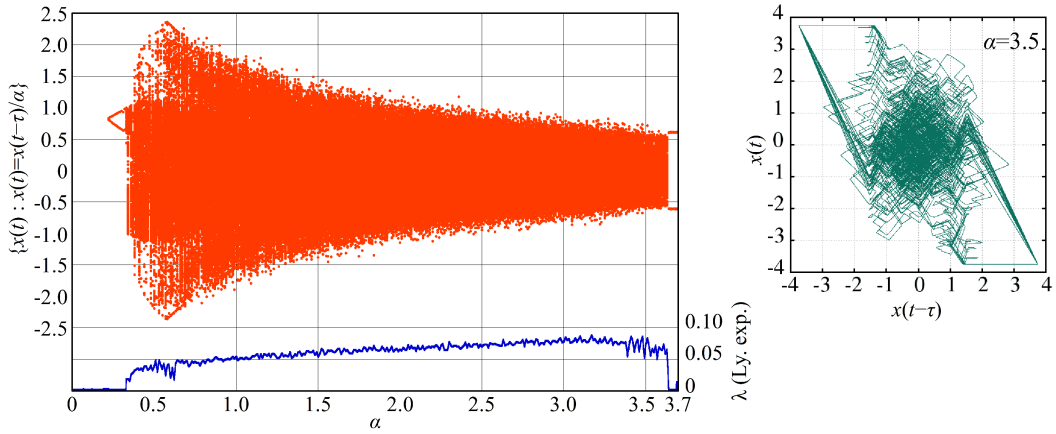


Figure 2.23 : Bifurcation diagram and accompanying largest Lyapunov exponent versus α is given on the left. A sample phase portrait for a chosen α value is given on the right.

α The weight of the delayed nonlinear function has similar effect as h_f on the system dynamics. Both determines the amplitude of the feedback term. The envelope of largest Lyapunov exponents is shrinking in Figure 2.23, because the Poincaré section is defined using α. As shown by the phase portrait in Figure 2.23, large values blend the scrolls and yield a different attractor which can not be defined as multi-scroll. Values greater than 3.63 yield periodic motion. $\alpha = 1$ keeps the system in the chaotic region, and provides a clear multi-scroll attractor with the given parameter set.

The delay amount effects the chaotic behavior fairly. In Figure 2.24, the largest Lyapunov exponents and bifurcation diagram show that the system may have chaotic behavior even for $\tau = 0.7$, however the multi-scroll attractor can not be observed with

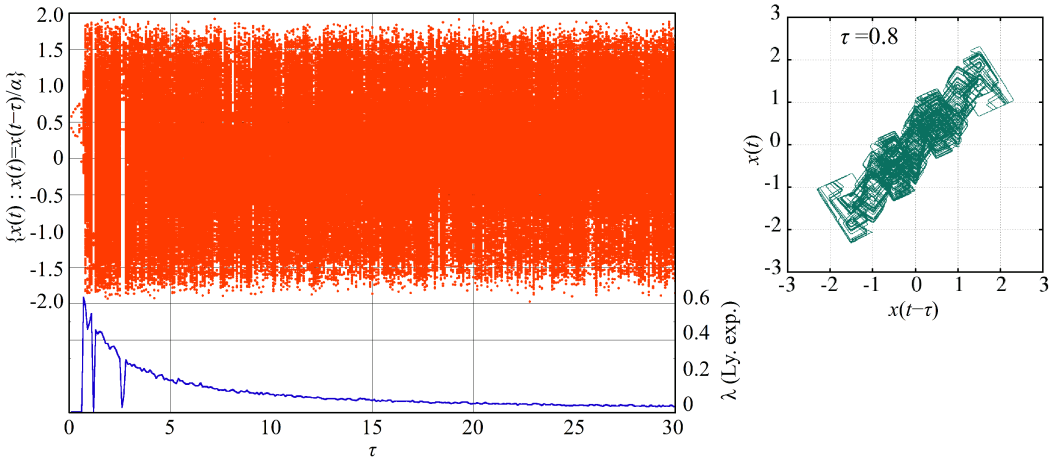


Figure 2.24 : Bifurcation diagram and accompanying largest Lyapunov exponent versus τ is given on the left. A sample phase portrait for a chosen τ value is given on the right.

this delay. In order to have the expected multi-scroll attractor, τ should be equal to 0.8 or greater. The phase portrait for $\tau = 0.8$ is plotted in Figure 2.24.

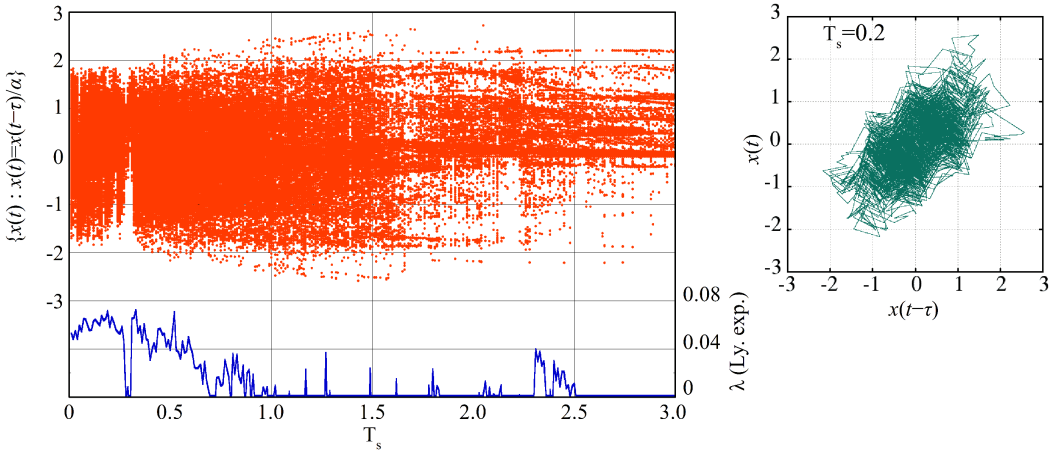


Figure 2.25 : Bifurcation diagram and accompanying largest Lyapunov exponent versus T_s is given on the left. A sample phase portrait for a chosen T_s value is given on the right.

According to the analysis whose results are plotted in Figure 2.25, sampling period T_s should be as small as possible. As a matter of fact, the sampled-data system has a continuous-time model basis. Thus, keeping the sampling period small protects the chaotic behavior of the system. $T_s = 0.2$ yields a reasonable multi-scroll attractor as given in Figure 2.25. On the other hand, even though it is not shown here, it is observed that the chaotic attractors are not the expected multi-scroll type for $T_s > 0.2$.

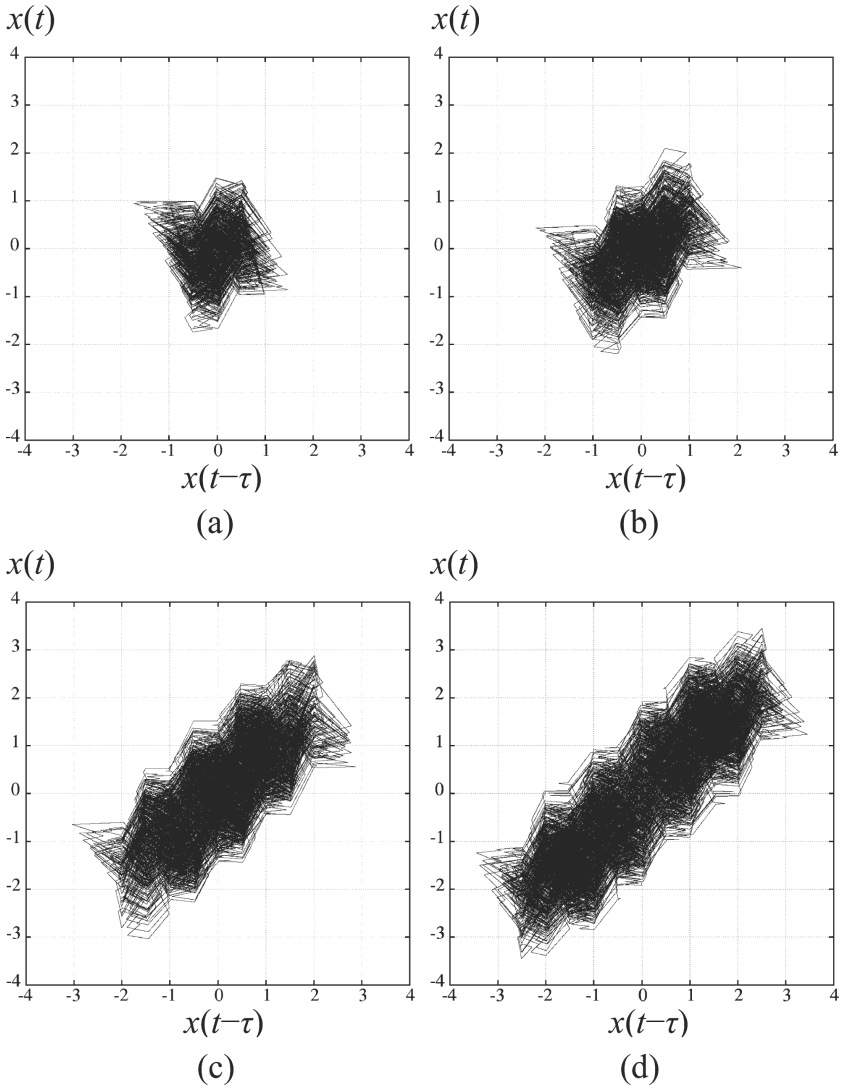


Figure 2.26 : Phase portraits in the $x(t-\tau)$ versus $x(t)$ state space. System parameters are the same as Figure 2.19 except n_f. n_f values in sub-graphs are 1 in (a), 2 in (b), 4 in (c), and 5 in (d). According to the V-shape of a mono-scroll attractor, each phase portrait has $2n_f$ scrolls.

Figure 2.20−2.25 show that the observed behavior of the system in bifurcation diagram and the numerically calculated Lyapunov exponent spectrum are in good agreement. Besides, simulations employing non-smooth nonlinearity (2.16) result in phase portraits which correspond to the given bifurcation diagrams and Lyapunov exponent spectrums. In Figure 2.26, n_f varies from 1 to 5, excepting 3, and phase portrait has been plotted using the non-smooth nonlinearity and parameter set in Figure 2.19, which includes the phase portrait for $n_f = 3$. Theoretically, number of scrolls has no limit, but the state variable is bounded by some facts, such as supply voltages in practical implementations.

3. NETWORKS

Two relaxation oscillator based networks, Network 1 and Network 2 (Subsections 3.1.1, 3.1.2), one logic oscillator based Network 3 (Subsection 3.1.3) and one time-delay chaotic system based Network 4 (Subsection 3.2) are presented in this Chapter. Oscillator $1-4$, which are given in Subsections 2.1.1, 2.1.2, Section 2.2 and Subsection 2.3.1, respectively, have been employed in these networks. Networks $1-3$ have regular grid structure and are members of Cellular Nonlinear Networks. Thus, they are investigated under the 'Cellular Nonlinear Networks' title. These three networks provide the base for research on spatio-temporal waves. In Subsection 3.1.4, three networks are compared and some results are presented. Dissimilarly, one dimensional unidirectionally coupled cells constitute Network 4, whose synchronization phenomenon has been studied with.

3.1 Cellular Nonlinear Networks

The Cellular Nonlinear Network (CNN), which has been outstanding candidate for non-Boolean computing paradigm since its invention in 1988 [9], are exploited in many applications especially the ones where massively parallel processing is essentially required. In order to solve the problems defined in one or more dimensional space, computational devices similarly having spatial dimension are sought. In image processing, for example, CNN proved its success for many ways. Analogical implementation of CNN [13] and its fully digital emulators [16, 63] are utilized to process image with high performance. One of the applications in which parallel processing is demanded is the wave processing. Solving problems exploiting linear or nonlinear waves on artificial excitable media (commonly electronic emulators) have recently attracted researchers' attention. For example, the shortest path planning problem is studied using Digital Reaction-Diffusion System [64], Relaxation-Oscillator CNN [3], and Reaction-Diffusion CNN [65]. Common feature of the given references concerns dynamic systems with real-valued state variables.

On the other hand, binary systems are also capable of wave processing. A kind of binary wave, which is called trigger wave in [66], is proved to solve many morphological image processing problems. This trigger wave processing technique has roots extending to 1999 [67]. Both the shortest path solving and morphological image processing applications motivate the research on wave-based computing.

Autowave is a kind of nonlinear waves, that propagates on active excitable media at the expense of the stored energy on the medium [68]. They can propagate without a driving signal, so the autowave term is used as the abbreviation for autonomous wave. Three properties listed below define the autowave [69]:

1. The shape and amplitude of autowaves remain constant during the propagation.

2. They do not reflect at the medium boundaries.

3. Two colliding autowaves annihilate each other.

Traveling waves, another nonlinear wave type, is propagated on active bistable (or stable) media but needs a driving signal. They are not self-sustaining like autowaves. Generating input signal that gives the opportunity to control the system is essential for this kind of wave. The network in this section focuses on the autowave and traveling wave propagation.

First, networks are designed as active media for autowaves, which have quite stationary source locations. In order to move the autowave source location on these networks, special spatio-temporal input patterns should be discovered, like for Network 1 in Subsection 2.1.1. The more generalizable method is configuring the network for successive traveling wave propagation by input patterns. These successive (i.e. nested) traveling waves mimic the autowaves in terms of their waveforms. For analog systems (e.g. Network 1 and Network 2) the key rule is to overlap some pieces of cell's nullclines in order to generate and propagate nested traveling waves. By this action, two saddle points and arbitrary numbers of unstable equilibrium points are revealed. Cells start to switch from one saddle point to the other by perturbation from the input or coupling terms. For the Network 3, cells should be modified such that the oscillation is controlled by an enable signal and this enable signal is determined in a traveling

wave generating way. Hence, three systems are converted to bistable mediums for propagating nested traveling waves.

Actually, the main purpose of the research on nonlinear waves in this section is to reveal the Doppler Effect for them. The Doppler Effect is the change in frequency of a periodic event when the source and the observer is in a relative motion. The successive wave-fronts of the periodic event are generated at different locations or if the source is moving or sensed at different locations if the observer is moving with respect to a reference point. Both cases may cause the observation of the wave-fronts with intervals different from the event period. The Doppler Effect observed on excitable media that generate nonlinear waves, such as Cellular Nonlinear Networks, is used as a kind of new feature related with the source movement.

3.1.1 Network using absolute nonlinearity

Coupling schemes are required for each cell model in order to construct an oscillatory medium. Simply, they form cellular nonlinear(/logic) networks after coupling. When coupling the cells, i.e., Oscillator 1 and 2 in order to have a network, it is supposed that every cell behaves same in the beginning. Then, the weight of one state term in one of the state equations, at least, is shared between the coupled cells and itself. It means the total weight in the network model for the selected state variable is equal to its weight in the single cell model. In Network 1 based on Oscillator 1, the weight of x state variable in the state equation for x has been shared. The coupling term I is included to the state equation for x in (2.1) which is given by

$$\begin{aligned} \dot{x}_{i,j} &= \alpha x_{i,j} + \beta y_{i,j} + g(x_{i,j}) + \omega I_{i,j}, \\ \dot{y}_{i,j} &= \gamma x_{i,j} + \varepsilon y_{i,j}. \end{aligned} \tag{3.1}$$

The coupling scheme, which is called synaptic law in CNNs, is given by

$$I_{i,j} = x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1}. \tag{3.2}$$

α in the network configuration is set to 3 when ω is chosen to be 0.3. Other parameters are the same with those of the oscillator. If the connected neighbors have the same state with the center cell, $x_{i,j} = x_{i-1,j} = x_{i+1,j} = x_{i,j-1} = x_{i,j+1}$ as supposed, the dynamics of single oscillator is preserved in the network. Indeed, cells on the network oscillates like uncoupled oscillators, only with small phase differences between neighbor cells. This

phenomenon forms a wave in the spatial domain which is observed by the top view of the network. In Figure 3.1, the state variable x of cells on Network 1 are depicted in three separate images at $t = 39$, $t = 40$, $t = 41$, respectively. Here, the numerical solution of Network 1 and Network 2 in Subsection 3.1.2 have been computed by Forward Euler Integration method with $h = 0.1$ second/iteration step. The waveform in the figure is called autowave due to its self generating ability. Once an autowave is generated on such a network, it does not require to be fed by any input signal in order to generate successive wave-fronts.

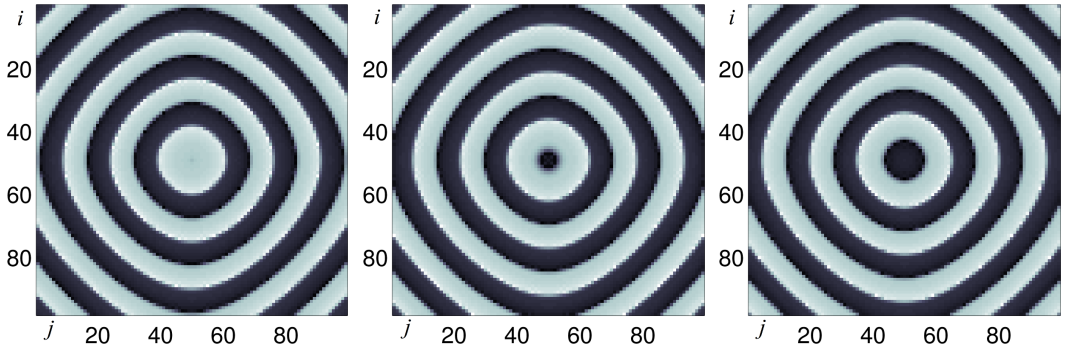


Figure 3.1 : Autowave propagation on Network 1 by top view of x state variables a) at $t = 39$, b) at $t = 40$, and c) at $t = 41$.

In order to create two saddle points for cells in Network 1, the parameters are selected as $\alpha = 3$, $\beta = -3$, $\gamma = 4.2$, $\varepsilon = -3$, $\mu = -10$, $\lambda = 1$. Also input term is added to the state equation for x given by

$$\dot{x}_{i,j} = \alpha x_{i,j} + \beta y_{i,j} + g(x_{i,j}) + \omega I_{i,j} + u_{i,j}(t). \tag{3.3}$$

Under the assumption that all the cells on the network have the same state, which means $\omega I_{i,j} = 4\omega x_{i,j}$, this configuration yields two saddle points at $(1, 1.4)$ and $(-1, -1.4)$. The phase portrait is separated into two equal parts, and each part converges to one of the saddle points. Phase portrait is depicted in Figure 3.2. Nullclines overlap on the line segment $[(1, 1.4), (-1, -1.4)]$. All the points on this line segment except the endpoints are unstable equilibrium points.

In this configuration, a slight difference in the value of state variable x of any neighbor turns the overlapped part of $\dot{x} = 0$ nullcline counterclockwise and converts the cell to an oscillator with only one unstable equilibrium point at $(0, 0)$. However, the cell does

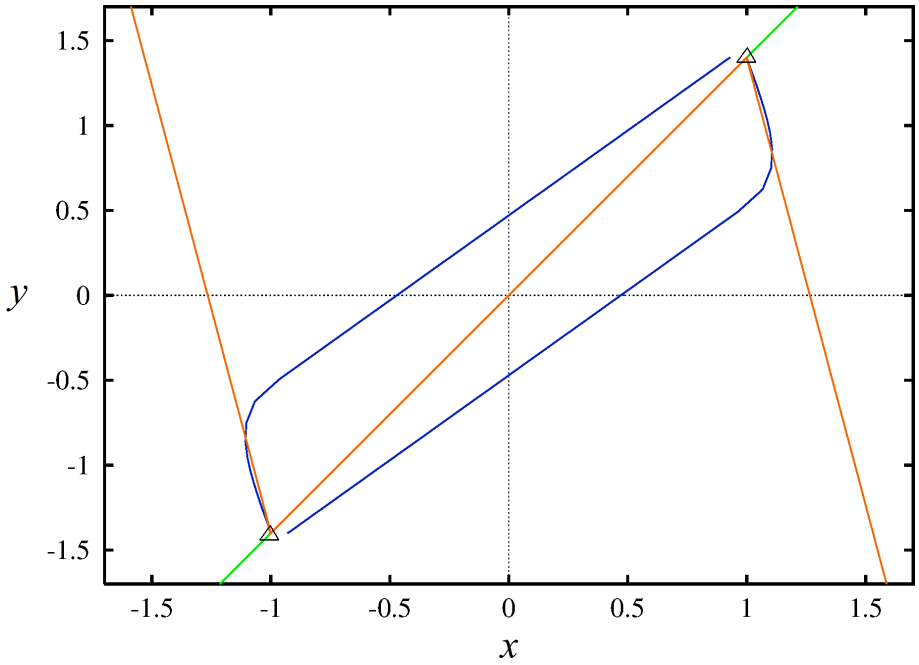


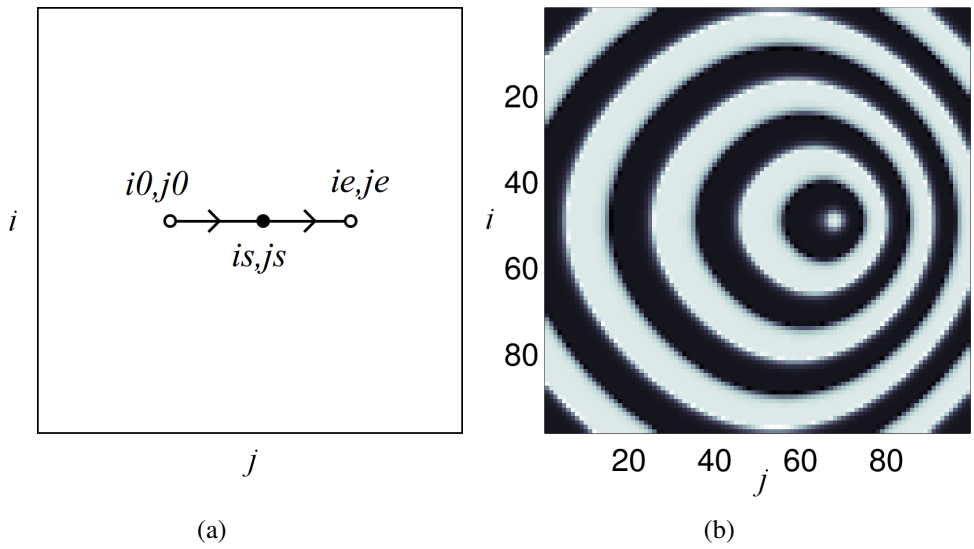
Figure 3.2 : Phase portrait of cells in Network 1. Saddle points are indicated with triangles.

not oscillate, just follows the neighbor cell while it is converging to other saddle point. As $t \to \infty$, cells are again converted back to two-saddle point systems and they evolve to the same saddle point together with phase differences. These phase differences in time give rise to the traveling wave in the spatio-temporal domain.

Similarly, a non-zero input shifts the $\dot{x} = 0$ nullcline on the x-axis. So, the input is able to convert the system to a system with a single stable equilibrium point. Hence, one of the cells on a network on which all cells are stationary at one of the saddle points can be triggered to converge to the other saddle point with a proper input. By means of the local coupling, cells from the closest neighbors to the farthermost ones start to converge to the same point with a phase shift.

The explained dynamics is the fundamental feature of the network configuration for the nested traveling wave propagation, hence for revealing the Doppler Effect. For this network, $u_{source}(t) = -0.5x_{source}(t_k)$, where $t_k \leq t < t_k + T$, and T is the traveling wave half-period. The Doppler Effect appears in Figure 3.3, in which $T = 5$ and source location is shifted to the right by two cells at each half-period. In the figure, nested traveling waves are depicted at $t = 15$.

After the nested traveling wave propagation method, whose source update scheme is a discrete process, Network 1 is again configured for generating autowaves below.



(a) (b)

Figure 3.3 : a) Moving input in time from left to right, b) nested traveling wave propagation on Network 1.

Autowaves whose generation is a continuous time process, are intent to provide more precision to the wave source movement. The coefficients $\alpha = 3$, $\beta = -3$, $\varepsilon = 3.6$, $\gamma = -3$, $\mu = -50$, $\lambda = 1$ and $w = 0.15$ are chosen for this time. Also, zero-flux boundary condition is applied.

In order to get input driven autowave generation a 23×24 sized network is built with the parameters, nonlinearity, synaptic law and boundary condition above. The input pair $u_{12,12} = 0.005$, $u_{12,13} = -0.005$ and $u_{\text{else}} = 0$ are applied. It should be noticed that the inputs are applied to the center of the network. Figure 3.4 illustrates the evolution of the network in time. The alternation to the negative states begins from the cell at $(12,13)$ which has input $u_{12,13} = -0.005$. On the other hand, the alternation to the positive states begins from the cell at $(12,12)$, as the input applied to the cell is 0.005. This is the reason for the non-symmetrical autowave generation observed on Figure 3.4. Here, one should consider that the non-symmetry happens in the wave generation not in the propagation which is determined by the symmetrical coupling given in the synaptic law (3.2).

To eliminate the non-symmetry, a symmetrical plus-sign shaped input pattern is proposed which is again applied continuously with constant values for autowave generation. A new (25×25) sized network is formed, which has 5 cells with non-zero input. Same network model, parameters and boundary condition are used. Input pattern has a negative value at the center $u_{13,13} = -0.25$ and absolutely smaller positive

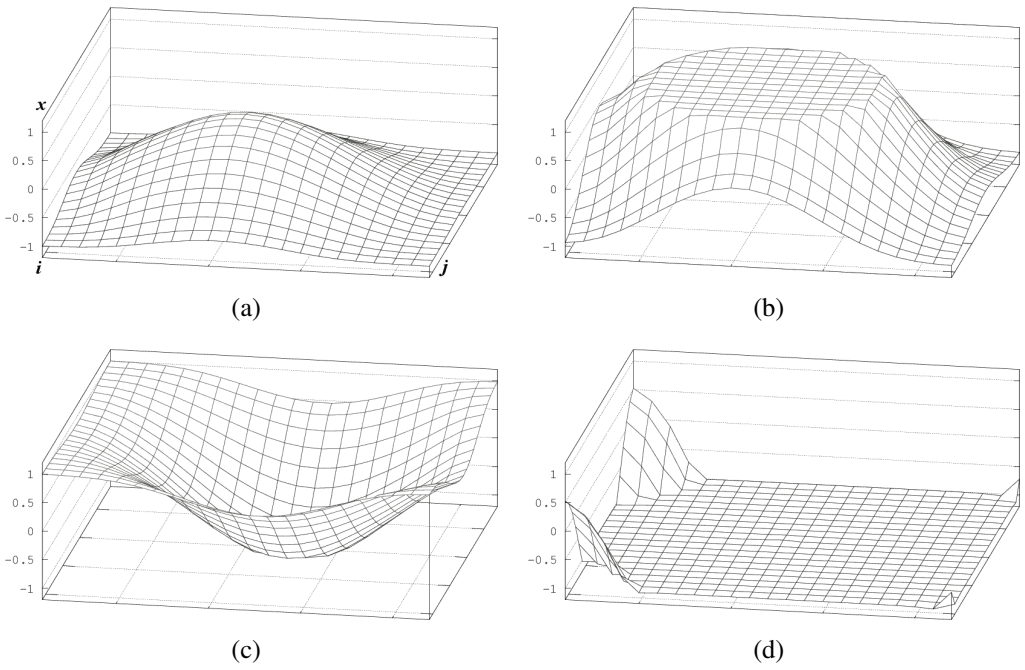


Figure 3.4 : Autowave generation with constant inputs $u_{12,12} = 0.005$ and $u_{12,13} = -0.005$. Subfigures are mesh plots of a (23×24) network's X state variable matrix a) for $t = 15$, b) for $t = 17$, c) for $t = 34$, d) for $t = 38$. Cells on boundary are not depicted. A **non-symmetrical** wave front is observed with this configuration.

values surrounding it $u_{12,13} = u_{13,12} = u_{13,14} = u_{14,13} = 0.05$. The result is depicted in Figure 3.5. Applying this pattern to a fixed location yields the symmetrical generation of the autowave. When all inputs are set to zero, the network goes to one of its saddle points, either $(x_{i,j}, y_{i,j}) = (-1,-1,2)$ or $(x_{i,j}, y_{i,j}) = (1,1,2)$. Then a new autowave can be generated from a new location using the same input pattern. But, this method annihilates the continuity of autowave evolution and has no advantage compared to the nested traveling wave generation in [70]. However, the advantage of utilizing input-driven autowaves appears when the wave source input pattern moves continuously.

In order to obtain the Doppler Effect, which means to carry information about wave source's movement across the medium, there should be continuity in wave generation and propagation. The nested traveling wave mimics the autowaves with the moving wave source very well [70]. But the mechanism still works in discrete-time. The wave source location should be updated and its amplitude should be alternated periodically. Instead of this process, the plus-sign shaped input pattern can be moved continuously (at each iteration for computer simulations). The autowave generation still goes on

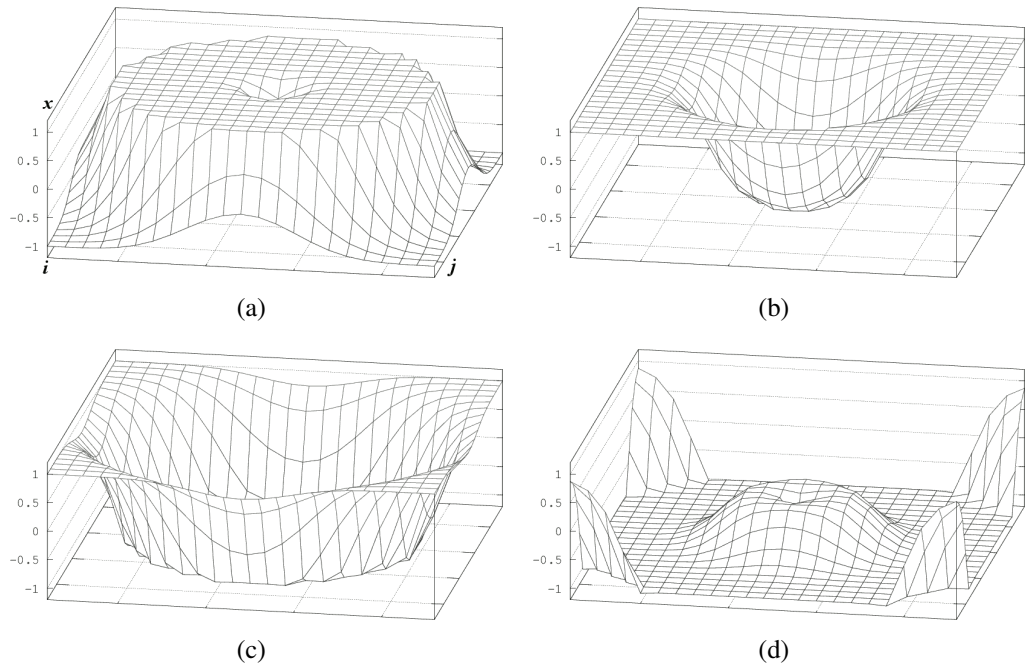


Figure 3.5 : Autowave generation with constant inputs $u_{12,13} = u_{13,12} = u_{13,14} = u_{14,13} = 0.05$ and $u_{13,13} = -0.25$. Subfigures are mesh plots of a (25×25) network's X state variable matrix a) for $t = 13$, b) for $t = 19$, c) for $t = 23$, d) for $t = 27$. Cells on boundary are not depicted. A **symmetrical** wave front is observed with this configuration.

but it is directionally doped by the input pattern's motion. The cells in front of the moving input pattern are forced to join state alternation sooner while the cells behind the moving input pattern are being relaxed to evolve slower. This yield higher and lower autowave frequencies before and after the moving input pattern.

The Doppler Effect sensing mechanism given in (3.11) still works. Using this mechanism a simple scenario is prepared. The input pattern (its center) is placed to the position $(5,5)$ on a 45×45 sized network and moved to the east direction with 0.1 cell/second speed. At 100-th second its direction is changed to the south-east but the speed is kept the same. When the input pattern's center cell reaches the $(12,22)$ position again, its direction is changed to the south with the same speed. The scenario ends when the pattern reaches the $(22,22)$ position. Figure 3.6 shows the route of the input pattern on the network. Figure 3.7 depicts the X matrices (constructed by x-states of the cells in regular grid form) of the network for different moments in this scenario.

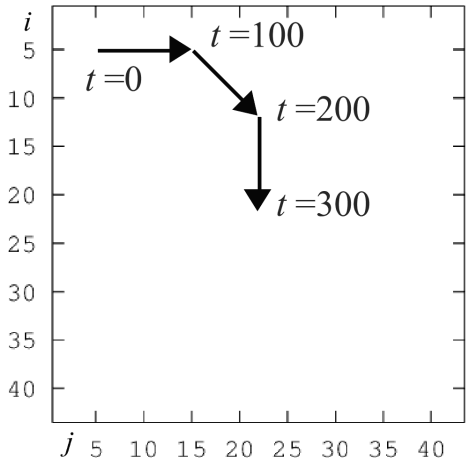


Figure 3.6 : The route of input pattern in the scenario.

It is obviously seen in the figure that the autowave generation is controlled by the continuously moving input pattern. In Figure 3.8, subfigures on the left side are the mesh plots of the D matrix (formed by d values in (3.11)). The U matrices of the network are depicted in the subfigures on the right. The D matrix plots show the high and low period regions which agree with the slow and continuous movement of the plus-sign shaped input pattern. For computer simulation the Forward Euler integration method is used for discretization of the analog model. The integration step is 0.02 second. Initial conditions are $x_{i,j} = -1$ and $y_{i,j} = -1.2$ which is a saddle point with zero input. Inputs applied to the cells that do not coincide with the plus-sign shaped pattern are zero. At each iteration step the input pattern moves only $(0.02) \cdot 0.1 = 0.002$ cell distance on a discrete 2D space. Therefore, the input pattern is smoothly distributed to the neighboring cells.

In this section, Network 1 in both bistable configuration and oscillatory configuration are proved to be able to propagate nonlinear waves that reveal the Doppler Effect. In both cases, special spatio-temporal input patterns are applied to continue the existence of wave source.

3.1.2 Network using signum nonlinearity

Similar to Network 1, the x state variables of cells are coupled in Network 2. However, coupling is applied to the arguments of the nonlinearity. Equation (3.4) is the network

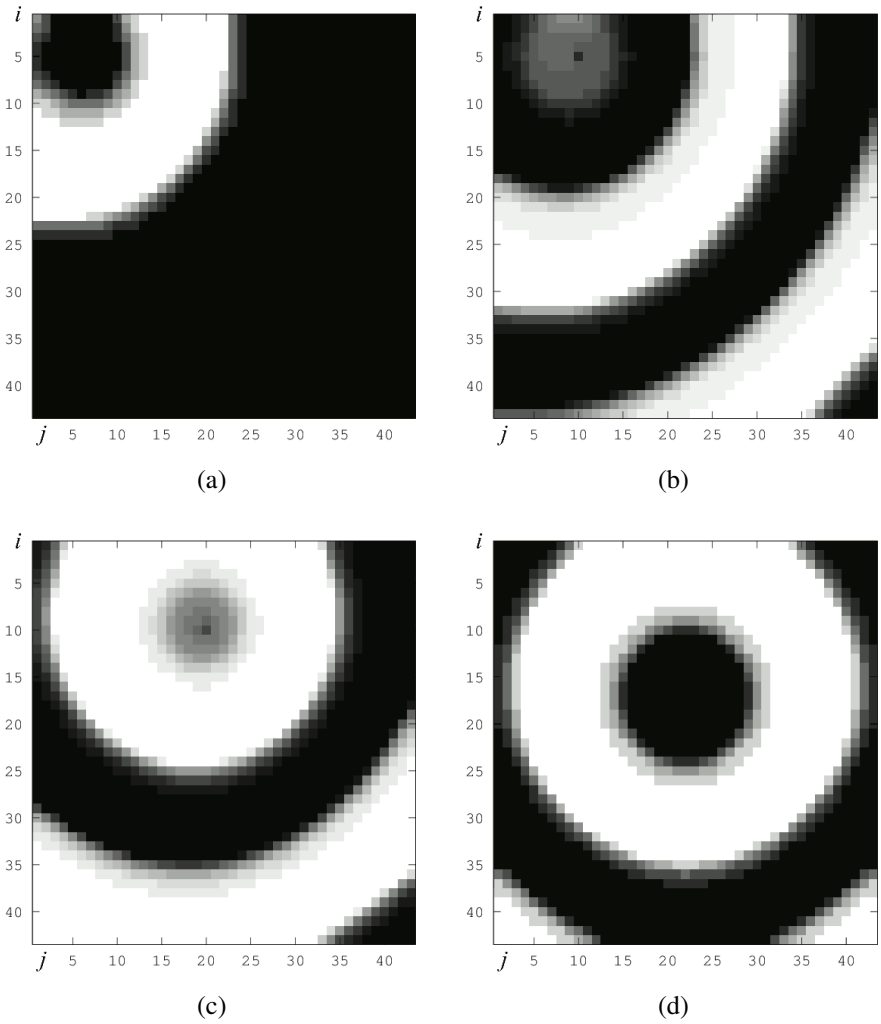


Figure 3.7 : The 2D plots of the X matrix of the 45×45 network with continuously moving DC input. Subfigures belong to $t = 20$ in a, $t = 50$ in b, $t = 170$ in c, $t = 260$ in d.

model and (3.5) is the nonlinearity using the same synaptic law given in (3.2).

$$\begin{aligned} \dot{x}_{i,j} &= (-\alpha+\mu)x_{i,j} - \beta y_{i,j} + g(x_{i,j}, y_{i,j}, I_{i,j}), \\ \dot{y}_{i,j} &= \gamma x_{i,j}(t) + \varepsilon y_{i,j}(t), \end{aligned} \tag{3.4}$$

$$g(x,y,I) = -\mu \text{sign}(\alpha x + \beta y + \omega I), \tag{3.5}$$

where $\alpha = 4$, $\beta = -6$, $\gamma = 2$, $\varepsilon = -2$, $\mu = -10$, $\omega = 0.35$. The weight of x in the nonlinearity of single cell is shared between the state variable x of the center cell and the ones of neighbor cells in the network with $\omega = 0.35$. Hence, the dynamics is preserved. Other parameters are the same with the oscillator. Figure 3.9 depicts the top view of Network 2's x state variables which form autowaves.

In order to switch cells in Network 2 to bistable systems, some parameters are changed: $\gamma = 5.25$, $\varepsilon = -6$, $\mu = 1.5$. Also input term is added to the state equation for x given

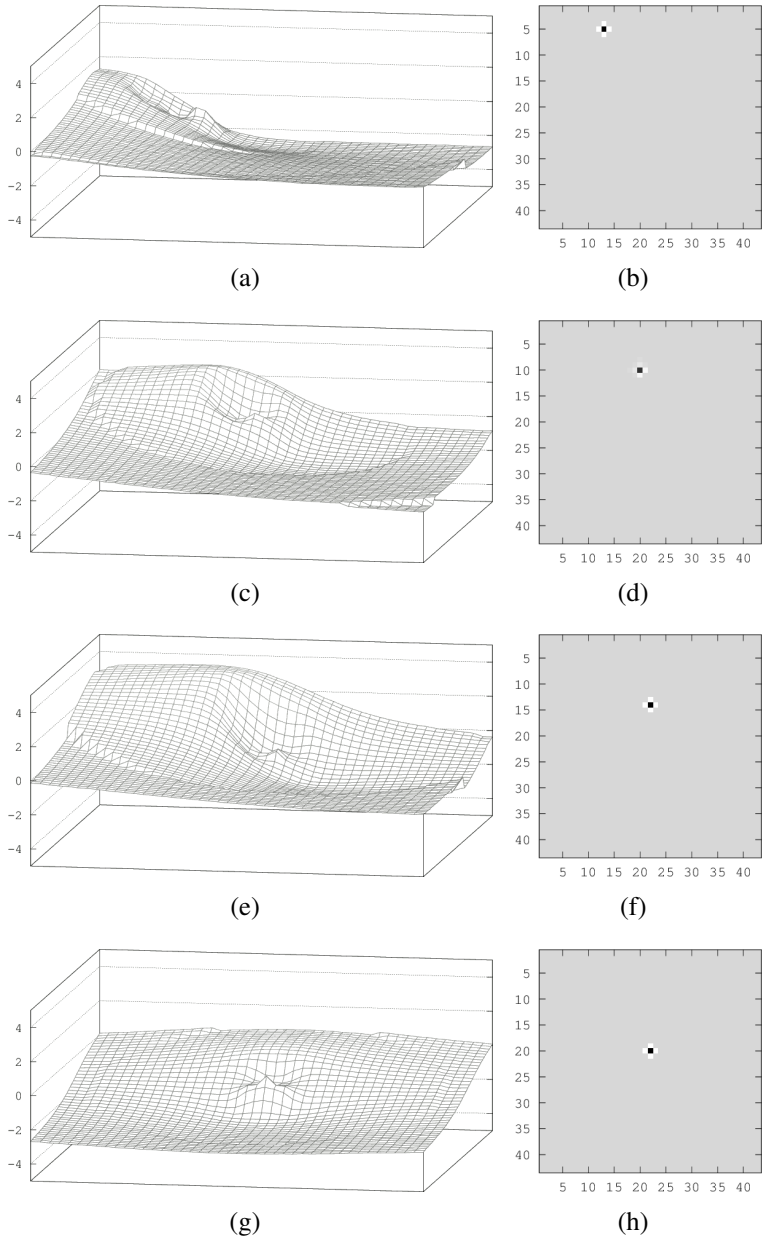


Figure 3.8 : The mesh plots of the D matrix, which are related to the Doppler Effect, (on the left column) of a 45×45 network with continuously moving DC input (on the right column). The d values, which compose the D matrix, are the last recorded omnidirectional wave-front passing periods on cells. d is inversely proportional to the approach speed of the wave source to the cell. The moving input pattern is the same one in Figure 3.5. Subfigures belong to $t = 80$ in a and b, $t = 170$ in c and d, $t = 220$ in e and f, $t = 280$ in g and h.

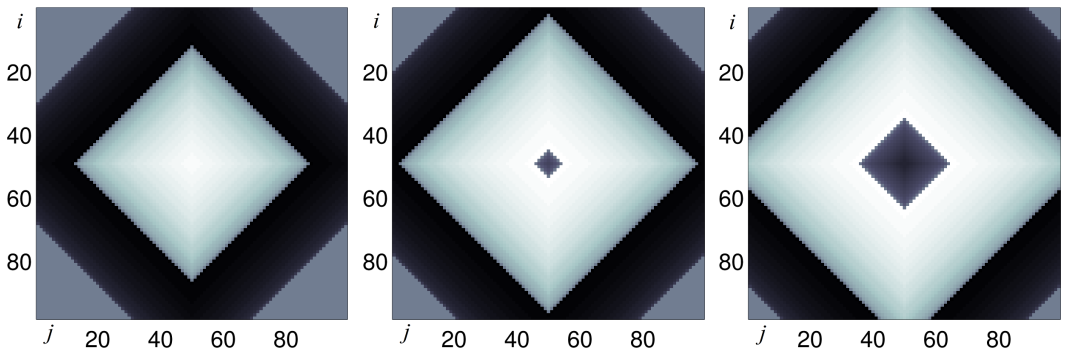


Figure 3.9 : Autowave propagation on Network 2 by top view of x state variables at a) $t = 3.5$, b) $t = 4.0$, and c) $t = 4.5$.

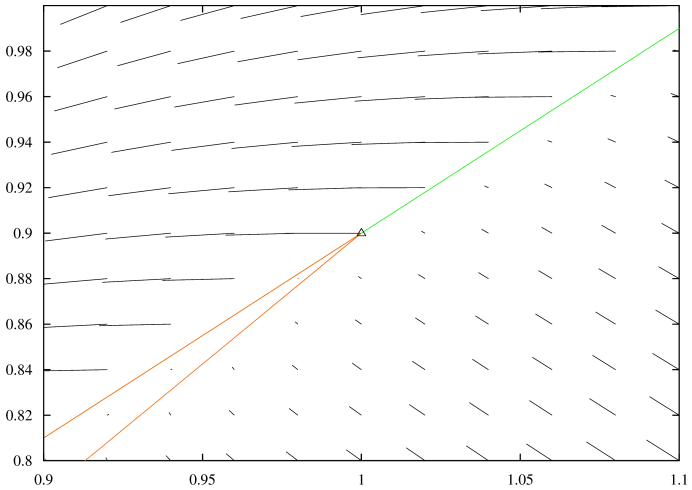
by

$$\dot{x}_{i,j} = (-\alpha + \mu)x_{i,j} - \beta y_{i,j} + g(x_{i,j}, y_{i,j}, I_{i,j}) + u_{i,j}. \tag{3.6}$$

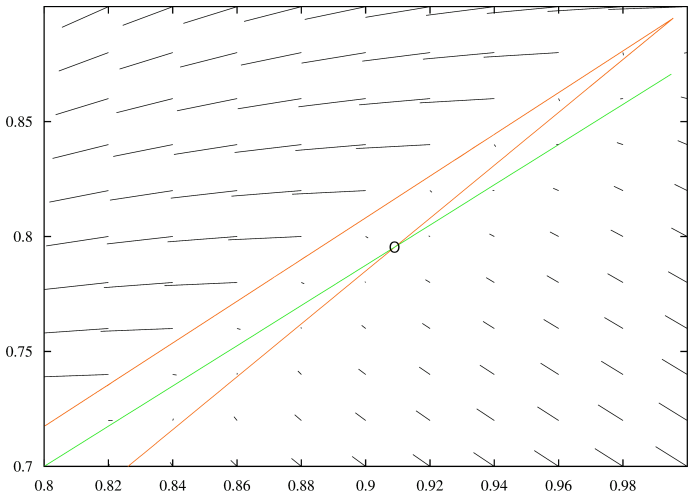
The coupling scheme in this network which occurs in the argument or the nonlinear function deviates from the classical coupling scheme, for example the one for Network 1. For Network 2 cell, the slope of outer pieces of the x-nullcline may have negative slope, thus resemble Network 1. To do this, required parametrization has been explained in Subsection 2.1.2. On the other side, x-nullcline of Network 1 can not have positive slope, as its nonlinearity is a function of only x. This feature gives flexibility in the design of cell (hence the network) dynamics.

On the contrary, the signum nonlinearity causes relatively high departure speeds when evolving from one saddle point to the other. If γ was set to 5.4, the nullclines would intersect on three points. Two of them would be saddle points and the third one would be the unstable equilibrium at $(0,0)$. However, $\gamma = 5.25$ makes the system a two stable point one. Figure 3.10(a) shows the phase portrait of the cell around the saddle point $(1.003, 0.903)$, and Figure 3.10(b) shows the phase portrait of the cell around the stable equilibrium point $(0.909, 0.795)$. γ is 5.4 in Figure 3.10(a), 5.25 in Figure 3.10(b).

As given in Figure 3.9, this system tends to generate tile shape wave-fronts with $\gamma = 5.4$. The speed of departure from the saddle points is too high to distinguish the numbers of neighbors effecting the center cell (see Figure 3.10(a)). Even the effect of one neighbor whose state has started to change is enough to trigger the cell to start to change its state immediately. But, if $\gamma = 5.25$, then center cell becomes immune to small effects from neighbors. Total effect of the neighbors should be high enough



(a)



(b)

Figure 3.10 : a) A saddle point of Network 2 with $\gamma = 5.4$, b) a stable equilibrium point of Network 2 with $\gamma = 5.25$.

to change the slope of inner piece of the x-nullcline in clockwise in order to switch the stable equilibrium point to a saddle one. When only this occurs, total effect of neighbors triggers a state transition at center cell. Similarly, the vertical shift effect of input term on x-nullcline should be high enough to switch the stable equilibrium point to a saddle one or remove that equilibrium. For further analysis, $\gamma = 5.25$ is chosen. A global view of phase nullclines is depicted in Figure 3.11. The necessity of this choice hides behind the wave-front curvature. By this way, a cell effected by a single neighbor and another cell effected by two cells do not behave similar. As a result, this network is able to propagate nested traveling waves with octagonal wave-front.

For this network, $u_{source}(t) = -0.5x_{source}(t_k)$, where $t_k \leq t < t+1$, and $u_{source}(t) = 0$, where $t_k + 1 \leq t < t_k + T$. The Doppler Effect on Network 2 appears in Figure 3.12,

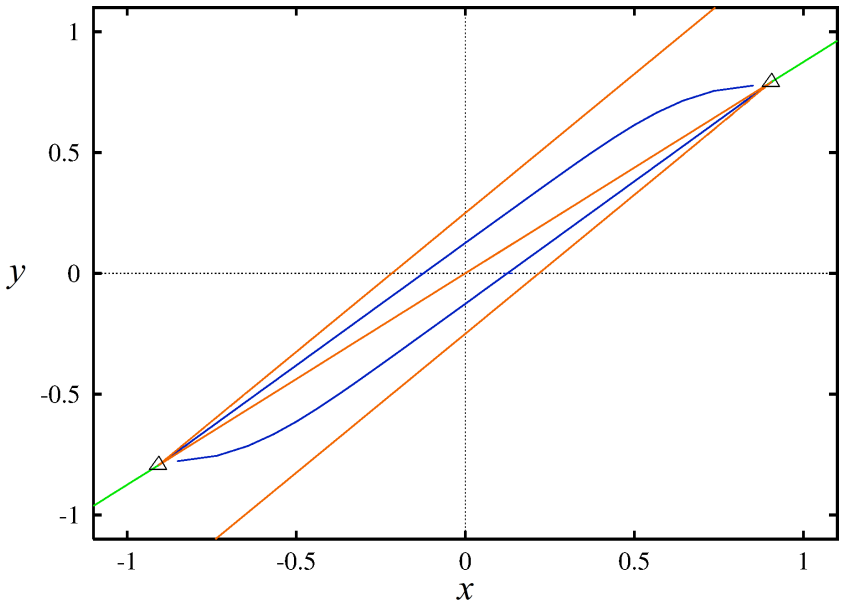
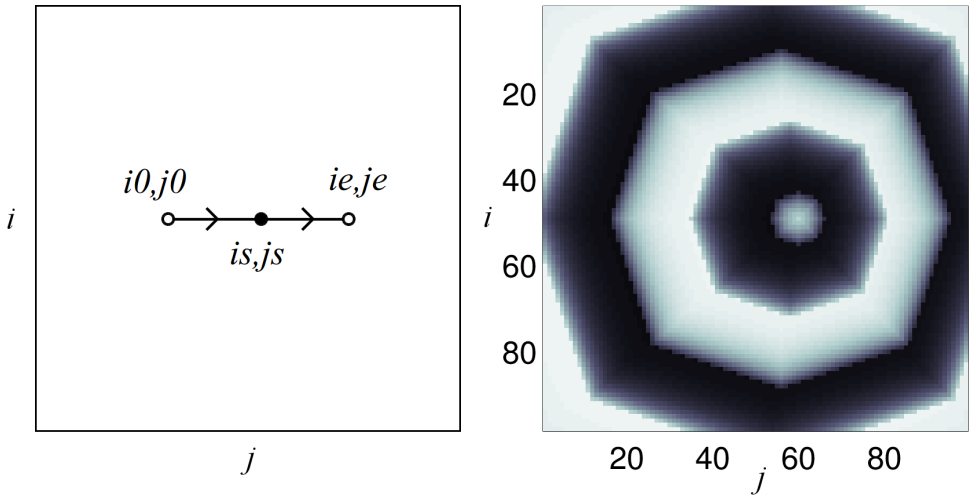


Figure 3.11 : Phase portrait of cells in Network 2. Saddle points are indicated with triangles.



(a) (b)

Figure 3.12 : a) Moving input in time from left to right, b) nested traveling wave propagation on Network 2.

in which $T = 5$ and source location is shifted to right by two cells at each half-period. In the figure, nested traveling waves are depicted at $t = 15$. Moving the source of autowave for Network 2 has not been investigated in the thesis.

3.1.3 Cellular logical network

In cells of both Network 1 and Network 2, oscillation is controlled by some facts. The weights tune the phase portraits and determine the waveform and frequency. Phase is controlled by the coupling. Thus, in order to observe the effect of coupling, and generate autowave, at least one control input should be added to the Oscillator 3. In Figure 3.13, the digital oscillator with an enable e signal is given with its state diagram. Briefly, system oscillates if $e = 1$, and preserves its state if $e = 0$. Its next-state equations are given by

$$\begin{aligned} x(t_{k+1}) &= ey(t_k) + e'x(t_k), \\ y(t_{k+1}) &= ex'(t_k) + e'y(t_k). \end{aligned} \tag{3.7}$$

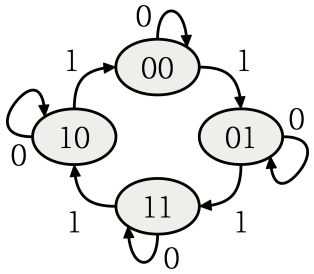


Figure 3.13 : State diagram of Oscillator 3 with enable e input. State word is (x,y). Transition is controlled by e.

Then, a coupling scheme can be defined on enable signal. A cell is enabled to oscillate, when it is in one of the peak states ('00' or '11') and one of the coupled cells are in a transition state ('01' or '10'); or when it is in one of the transition state and all of the coupled cells are in one of the peak state; or when an enable input is applied. For a cell $x \oplus y$ (exclusive-or operation) indicates that it is in one of the transition states. So,

$$e_{i,j} = x_{i,j} \oplus y_{i,j} \oplus I_{i,j} + u_{i,j} \tag{3.8}$$

gives the necessary autowave enable signal equation with the coupling scheme given by (3.9). The generated autowave is depicted in Figure 3.14.

$$\begin{aligned} I_{i,j} = {} & x_{i-1,j} \oplus y_{i-1,j} + x_{i+1,j} \oplus y_{i+1,j} \\ & + x_{i,j-1} \oplus y_{i,j-1} + x_{i,j+1} \oplus y_{i,j+1}. \end{aligned} \tag{3.9}$$

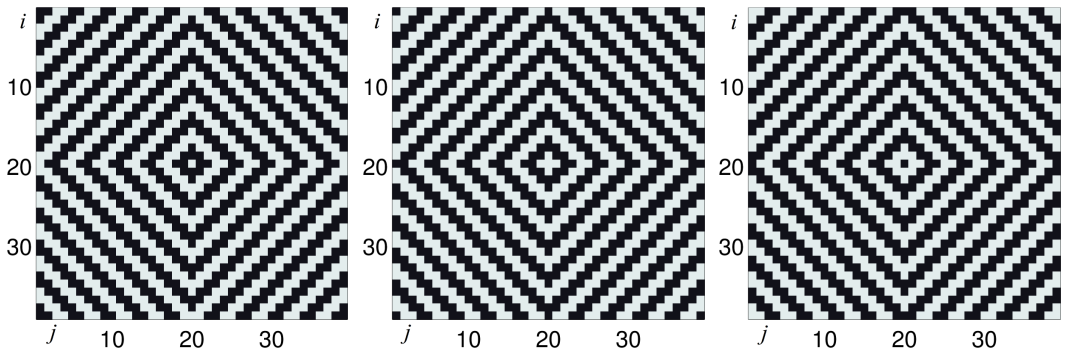


Figure 3.14 : Autowave propagation on Network 3 by top view of x state variables at $t = 3.8$ in a, $t = 3.9$ in b, and $t = 4.0$ in c.

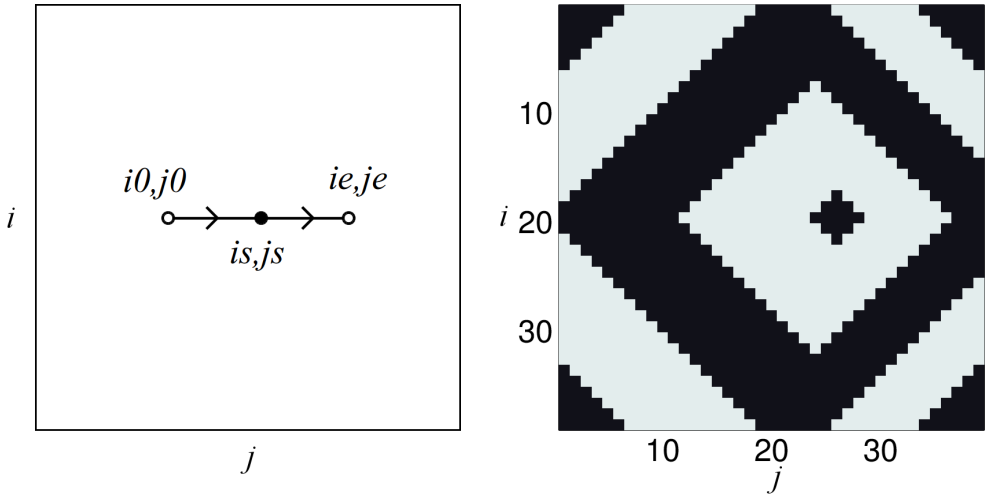
Autowaves can be generated by applying $u = 1$ pulse or $y[0] \neq x[0]$ initial condition to any cell, when the network is stationary in one of the peak states. Once the autowave is generated from a cell, u signal applied to other cells does not change the source location similarly to the situation in Network 1 and 2.

For Network 3, the autowave enable signal equation (3.8) is suitable for traveling wave propagation, but with the coupling scheme given by

$$\begin{aligned} I_{i,j} =& (y_{i-1,j} y_{i+1,j} y_{i,j-1} y_{i,j+1})' \\ & \cdot (y_{i-1,j} + y_{i+1,j} + y_{i,j-1} + y_{i,j+1}). \end{aligned} \tag{3.10}$$

This coupling scheme provides the information that not all of the coupled neighbors have the same output. If $I = 1$, one of them has a different output state which means a traveling wave-front is passing through the neighbor cells. According to (3.8), switching to a transition state is enabled by $I = 1$, when the center cell is at any peak state. Then, $I = 0$ is required to switch to other peak state from the transition state. Under the rule of (3.10), all cells may stay in one of the peak state, e.g. $(x = 0, y = 0)$. If one of them changes its state to a transition state, $(x = 0, y = 1)$, surrounding neighbors follows it. Like the domino effect, switching to transition state will propagate. The center cell reaches to the opposite peak state, $(x = 1, y = 1)$, when all coupled neighbors have output $y = 1$. Henceforward, the center cell becomes ready for an opposite amplitude traveling wave. Similarly, switching to opposite peak state also propagates. The instrument to generate a traveling wave on network staying at a peak state is a proper input pulse.

Figure 3.15 depicts nested traveling waves on Network 3, generated by a moving source with a speed of 2 cells/T. Network top view is captured at $t = 4$. The switching period of digital design is set to 0.1 in simulations So, the figure is generated at the 40-th iteration.



(a) (b)

Figure 3.15 : a) Moving input in time from left to right, b) nested traveling wave propagation on Network 3.

3.1.4 Results and comparison of networks

In order to make use of Doppler Effect on active media, effect should be sensed, stored and served in parallel by every cell. Two different Doppler Effect sensing mechanism are given in [71] and [70]. In [70], the wave-front passing half-period is calculated at every zero crossing of state variable x. On the other hand, only zero crossing x with positive derivative is used in [71]. If the temporal x waveform has the half wave symmetry, mechanism in [70] is suitable and has 2 times more update rate. Mechanism in [71] is safe for any waveform, but update rate is low.

Waveforms from all three networks employed in this section has half wave symmetry. Thus, the Doppler Effect sensing mechanism has been constituted by

$$d_{i,j}(t) = t_k - t_{k-1} - T, \quad t_k \leq t < t_{k+1} \tag{3.11}$$

for all t_k, $x(t_k) = 0$ and T is the traveling wave half period. Relative wave-front half-period is sampled and held by $d_{i,j}(t)$. $d_{i,j}$ is considered as the numerical result of the Doppler Effect.

Although $d_{i,j}$ update instants are not the same for all cells, as they have phase shift that provides the traveling wave, every cell runs the Doppler Effect sensing mechanism in parallel. Equation (3.11) have been added to the cell models for simulation and future implementations. Three networks developed with the sensing mechanism is simulated and the results (top view of d variables) are presented in three figures: Figure 3.16(b), 3.16(c), and 3.16(d). Before them, the wave source motion and the obstacles on the medium is depicted in Figure 3.16(a).

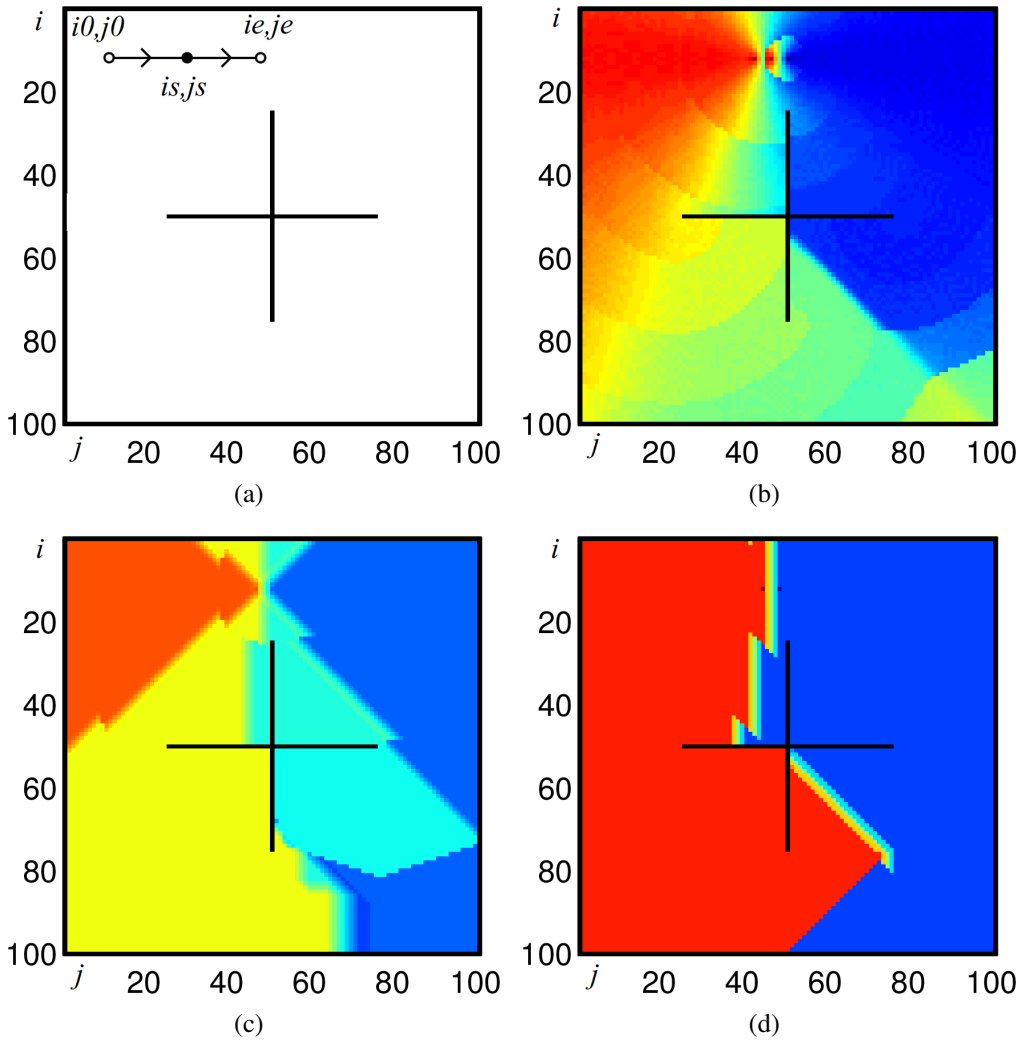


Figure 3.16 : a) Route of the wave generating input signal, b) D matrix of Network 1, which consists of d values of the cells, c) D matrix of Network 2, d) D matrix of Network 3.

The 'jet' colormap of given figure assigns blue to the lowest observed wave half period (the lowest d) value. Red implies the highest value. The worst result has obtained from Network 3 in Figure 3.16(d). In almost binary image Figure 3.16(d), majority of the cells has either blue or red color. Intermediate values are not represented as desired. Figure 3.16(c) shows that Network 2 is able to generate 4 different d values. Network 2 generates d feature that provides further information than the one by Network 3. The variety of d values of Network 1 shown in Figure 3.16(b) should be noticed. Network 1 outperforms other two in terms of information that generated d feature has. The variety of values in generated d feature is evaluated as Criteria 1 to compare these three networks.

The Second Criteria is the implementation complexity. A digital approximation of Yalcin's network has been implemented as a wave computer in 2009 [3]. Not the Network 3 but its simpler version have been implemented in 2014 [72]. Although, the novel Oscillator 2 and Network 2 have not been electronically implemented yet, all three networks are intended to be suitable for implementation. The circuit complexity of Network 3 seems to be the least. Due to the simpler nonlinearity of system 2, circuit complexity of the Network 2 is less than Network 1. Digital circuit requires CMOS gates and flip-flops. On the other side, Network 1 and Network 2 require analog building blocks such as inverting and non-inverting amplifiers, adders, integrators, comparators, and multiplexers.

Wave source motion information, which is encoded to the wave frequency, is carried by the propagation of the traveling wave. Due to the finite propagation velocity of the wave, a delay occurs between the generation of information and its reception by the observer. The freshness of the source's motion information may be essential for some applications. Hence, the wave-front propagation speed of active media should be considered as the Criteria 3. In general, fast networks are required to make the effect useful in applications. A cell is assumed to leave a saddle point (continuous systems) or a peak state (digital system) if its state variable x amplitude decreases to 90%. When the three networks are compared under that assumption, Network 3 evidently has the highest wave propagation speed with 1 cell/iteration. The wave-front speed of Network 1 is 0.158 cell/iteration. Network 2 achieves an intermediate speed with 0.435 cell/iteration. It should be reminded that the integration step of numerical

method used in Network 1 and Network 2 solution is 0.1. All the values reported above is measured on cardinal directions.

Criteria 4 to comparison networks is the speed of settling to a saddle point from the other one. One cell should converge enough to a saddle point before going back to the original one via the expected trajectory. In applications, traveling wave generating input pattern should not be applied before a cell settle to a saddle point or a peak state. Thus, speed of settling determines maximum traveling wave generation rate. According to the measurements in this work, settling time of a cell in Network 1 is 3.7 s in Matlab simulations and 8 s in XPPAUT simulations as Oscillator 1. The numbers for Network 2 is 4.4 s in Matlab simulations and 10 s in XPPAUT simulations for Oscillator 2. For logic system, only network simulations are computed using Matlab and settling time is measured as only 3 iterations.

The curvature of traveling waves, which is called Criteria 5, classifies the systems too. Network 1 has circular, Network 2 has octagonal, Network 3 has tetragonal wave-front. It is related by the coupling effect and cell dynamics explained above in the previous subsections. Curvature effects the wave-front normal vector, which is essential for applications. The best one is circular propagation. Circular propagation correctly provides the wave-front approach angle. In octagonal and tetragonal propagation, the angle values are quantized to 8 and 4, respectively.

3.2 1D Network with Unidirectional Coupling

Synchronization of chaos refers to a process wherein two or many chaotic systems (either equivalent or nonequivalent) adjust a given property of their motion to a common behavior due to a coupling [73]. Since the early work by Pecora and Carroll [74] on synchronization phenomena of chaotic systems, the research on synchronization has moved towards chaotic systems. This was a challenge for synchronization because it is well-known that chaotic systems are extremely sensitive with respect to initial conditions. To date, several synchronization schemes have been proposed and the theoretical analysis and experimental verification of these schemes has been given in the literature [75]. On the other hand, in [76] “anticipating synchronization” of chaotic systems is introduced. It is claimed in [76] that it might be possible to synchronize time-delayed master system and a coupled non-delayed

slave system. That is, synchronization is exhibited between drive and time-delayed response system, so that in this manner the slave dynamics act as a predictor of the master dynamics [77]. For an example, anticipating synchronization of the system given in [53] is studied in [78]. In this section anticipating synchronization of the time-delay chaotic system (2.15) is investigated. If one can couple to the state variable of the original system, another system can be constructed to anticipate the future states of the original system with a time difference of τ. It is also possible to increase number of coupled systems resulting in anticipation of states of the original system for integer multiples of τ. The 1D unidirectional coupling scheme for anticipating chaotic synchronization of the considered system is verified with numerical simulations in this section.

Now, let us assume that another system coupled with this one is introduced as follows

$$\begin{array}{ll} \dot{x_1}(t) & = -x_1(t) + \alpha f(x_1(t-\tau)) \\ \dot{x_2}(t) & = -x_2(t) + \alpha f(x_1(t)). \end{array} \tag{3.12}$$

The second system is realizing x_2 evolution while the first one is for x_1. If x_2 is delayed by τ, the second equation will take the following form

$$\dot{x_2}(t-\tau) = -x_2(t-\tau) + \alpha f(x_1(t-\tau)). \tag{3.13}$$

Taking the difference between the above one and the first equation in (3.12) leads to

$$\dot{e} = -e \tag{3.14}$$

where e is the error defined as $e \triangleq x_1(t) - x_2(t-\tau)$ [76]. (3.14) has a stable solution and as time goes to infinity the error goes to zero resulting in synchronization between $x_1(t)$ and $x_2(t)$. That is, the previous state of the response system is synchronized to the current state of the drive system. This means that the future state of the drive system can be predicted by the current state of the response system leading to anticipating synchronization. Figure 3.17shows numerical simulation results verifying this phenomenon.

It is possible to couple additional systems, in such a way that the error system is asymptotically stable, as given in (3.15).

$$\begin{array}{lcl} \dot{x_1}(t) & = & -x_1(t) + \alpha f(x_1(t-\tau)) \\ \dot{x_2}(t) & = & -x_2(t) + \alpha f(x_1(t)) \\ \dot{x_3}(t) & = & -x_3(t) + \alpha f(x_2(t)) \\ & \vdots & \\ \dot{x_n}(t) & = & -x_n(t) + \alpha f(x_{n-1}(t)) \quad \text{for } n \geq 3. \end{array} \tag{3.15}$$

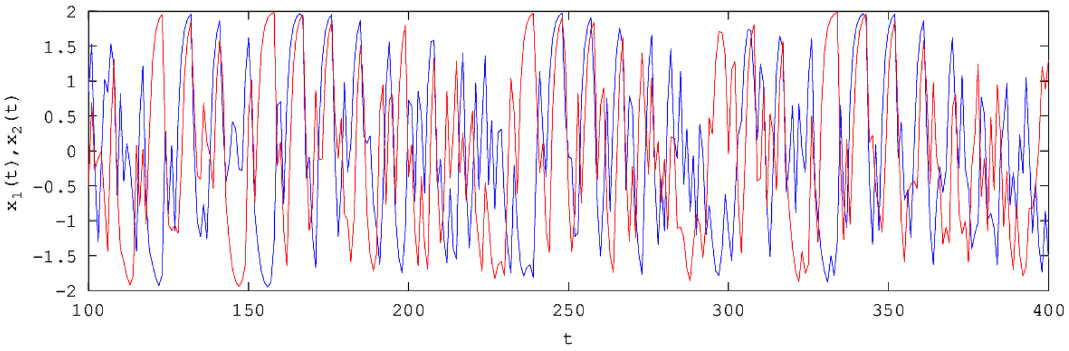
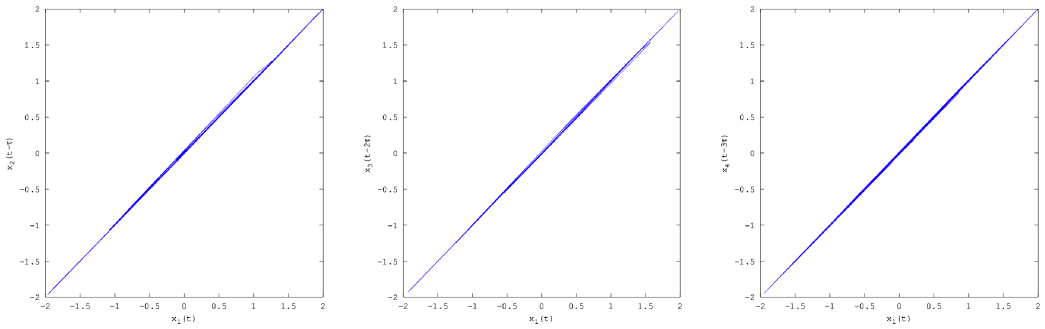


Figure 3.17 : Evolution of x_1 and x_2 is plotted in time. Due to the anticipating synchronization, the red signal which belongs to x_2 occurs $\tau = 8$ seconds before the blue signal of x_1. The first 100 seconds of the simulation is not plotted in order to ignore the transient effect of initial conditions.



(a) (b) (c)

Figure 3.18 : The anticipating synchronization between a) $x_1(t)$-$x_2(t-\tau)$, b) $x_1(t)$-$x_3(t-2\tau)$, c) $x_1(t)$-$x_4(t-3\tau)$.

By this way, the future state of the driver can be predicted for any integer multiple of τ [76]. Figures 3.18a, 3.18b and 3.18c depict the Matlab simulation results showing the anticipating synchronization for τ $[x_1(t) = x_2(t-\tau)]$, 2τ $[x_1(t) = x_3(t-2\tau)]$ and 3τ $[x_1(t) = x_4(t-3\tau)]$, respectively.

4. IMPLEMENTATIONS

This chapter presents the electronic implementations of cells from Chapter 2 and Chapter 3. As discussed in former chapters, relaxation oscillators are studied intensively in the form of cellular nonlinear networks, although time-delay sampled-data system are studied individually. Analogically, implementations of cellular nonlinear networks and individual time-delay chaotic systems are mainly reported in this chapter. Some innovative designs, which are accompanying time-delay system implementations, such as asynchronous delay doubler and binary low-pass filter, are also exposed in this chapter.

The implementations in this thesis, do not include VLSI designs. For analog circuits, a few off-the-shelf active and passive components are consumed. Thus, they result in low frequency proof of concept implementations. On the other hand, for digital implementations mature and modern FPGAs are employed. In one implementation, the target device is a GPU which exhibits outstanding performance. In contrast to analog circuits, the reader will find larger and more complex implementations for digital designs in this chapter.

The CNN implementations for relaxation oscillators are demonstrated in Subsections 4.1.1 and 4.1.2. Section 4.2 covers the CNN implementation of a slightly modified version of logic oscillators. Time-delay system and network implementations are treated in Section 4.3. First, a new area-efficient binary delay line is proposed in Subsection 4.3.1. A D-type flip-flop based delay line, the proposed delay line and binary inverter (NOT gate) based delay line are employed in mono-scroll attractor generating chaotic circuits in Subsections 4.3.2, 4.3.3, 4.3.4, respectively. Then, flip-flop based delay line is employed in the chaotic circuit generating a multi-scroll attractor in Subsection 4.3.5, one-dimensional unidirectional network with analog integrator in Subsection 4.3.6, and with digital integrator in Subsection 4.3.7.

Multi-scroll attractor generating circuit has a novel delay line design in which delayed data is encoded at the beginning and decoded at the end of the line.

4.1 Implementations for Relaxation Oscillators

This section covers the digital and software implementations of Network 1 (Subsection 3.1.1), targeting an FPGA from Xilinx Virtex2P family and a GPU form Nvidia (Subsection 4.1.2), respectively. FPGA implementation is an ongoing research started by [3]. The version implemented in this thesis uses fixed-point arithmetic, while its ancestor uses half-precision floating point arithmetic. Even FPGA implementations exhibit better results than CPU simulations, it is shown that an experimental GPU implementation outperforms the FPGA implementation in Subsection 4.1.2.

4.1.1 Digital implementation of relaxation oscillator network

The original wave computer core is proposed in [3] which is the successor of the design in [29]. Wave computer core is operated in path planning application [30], and the hardware/software co-design approach is implemented in [79]. One of the aims in this reference is to change the arithmetic design approach from floating-point to fixed-point. The mathematical model is changed from the Chua-Yang model to the Full Signal Range (FSR) model, and fixed-point arithmetic is used in [80]. In [81], the output of the CNN computation is 16-bits in width and fixed-point number representation is chosen. Another aim is to decrease the bit precision while the error is in a reasonable level. In [82], the target is to find whether such a template exists, and if so, what the minimal word length is. In [83], a simple algorithm is introduced to determine the optimal fixed-point precision and maximize computing performance. According to the research on this subject, fixed-point arithmetic satisfies the adequate precision for representing real network dynamics.

In this subsection, the floating-point arithmetic blocks in [3] are switched by their fixed-point arithmetic equivalents. The wave computing system composed with fixed-point arithmetic blocks was implemented on an FPGA platform with its host controller PC and on-line monitor, which is illustrated in Figure 4.1.

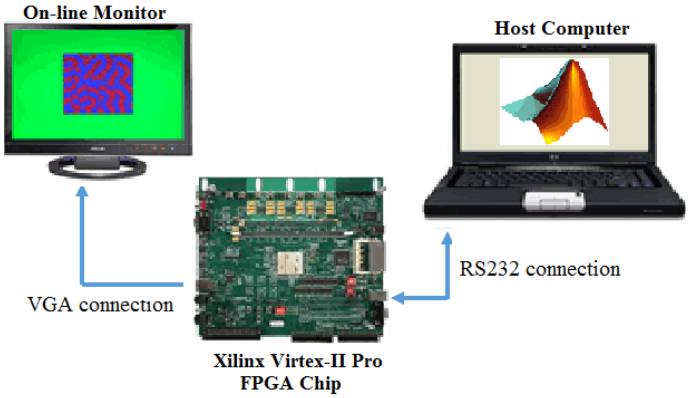


Figure 4.1 : The scheme of the wave computing system.

The discrete time model of Network 1 which is a obtained using Forwards Euler method is given by

$$\begin{aligned} x_{i,j}(k+1) &= x_{i,j}(k) + \tau[\alpha x_{i,j}(k) + \beta y_{i,j}(k) \\ &\quad + g(x_{i,j}(k)) + I_{i,j}(k) + u_{i,j}], \\ y_{i,j}(k+1) &= y_{i,j}(k) + \tau[\varepsilon x_{i,j}(k) + \sigma y_{i,j}(k)], \end{aligned} \tag{4.1}$$

with the nonlinearity,

$$g(x_{i,j}(k)) = \begin{cases} m.(x_{i,j}(k) - \lambda) & \text{if } x_{i,j}(k) > \lambda; \\ 0 & \text{if } |x_{i,j}(k)| \leq \lambda; \\ m.(x_{i,j}(k) + \lambda) & \text{if } x_{i,j}(k) < -\lambda; \end{cases} \tag{4.2}$$

and the synaptic law,

$$\begin{aligned} I_{i,j}(k) &= a_{i,j+1}x_{i,j+1}(k) + a_{i\text{-}1,j}x_{i\text{-}1,j}(k) \\ &\quad + a_{i,j\text{-}1}x_{i,j\text{-}1}(k) + a_{i+1,j}x_{i+1,j}(k), \end{aligned} \tag{4.3}$$

which defines the effect of coupled neighbors of the node.

Nodal Processing Element (NPE) is the fundamental component of the design. The mathematical model of Network 1 in (3.1) is realized by NPEs. Designed core contains 4×4 NPEs as nodes. NPEs concurrently execute iterations for a slice of the network.

In this design, these 16 NPEs constitute the Cellular Nonlinear Processor Network (CNPN). Parameters for the mathematical model are transferred from Parameter Register to NPE. Each NPE has two stacks called s_1 and s_2, and five temporary registers called f_1, f_2, a_1, a_2, and t, as shown in Figure 4.2 and Table 4.1.

In Table 4.1, α, β, ε, and σ are the state coefficients, τ is the integration time step, a is the coupling coefficient, m is the slope in nonlinearity, u is the input. The discontinuity point on nonlinearity (λ) is fixed to 1 and is not loaded to the stacks.

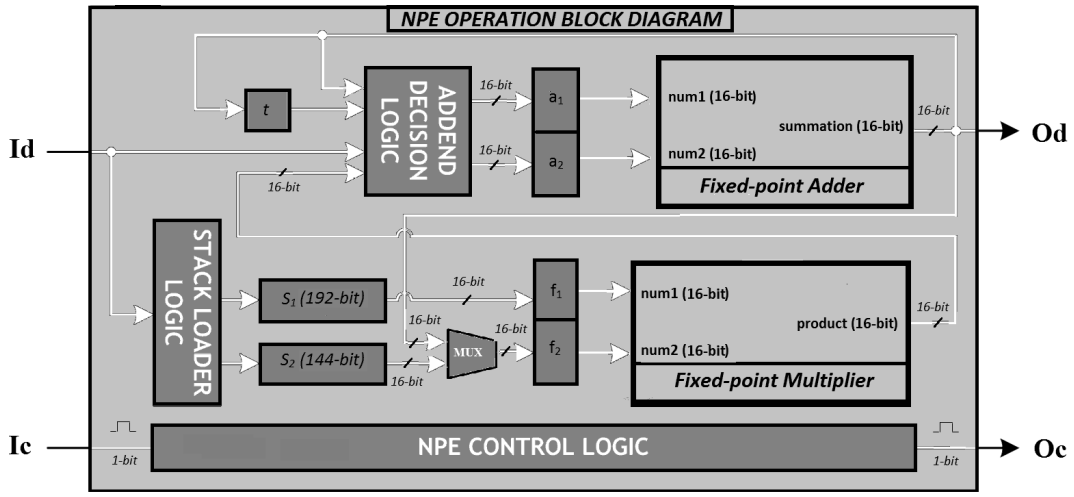


Figure 4.2 : Block diagram of the Nodal Processing Element (NPE), Id: variable and parameter inputs, Od: variable outputs, Ic: control signal inputs, Oc: control signal outputs [3].

Table 4.1 : Parameters and state variables hold by s_1 and s_2 stacks in the NPE.

Parameters and variables	$s_1(192-bit)$	$s_2(144-bit)$
word 1	$a_{\mathrm{i,j+1}}$	$x_{\mathrm{i,j+1}}(k)$ or x_{fixed}
word 2	$a_{\mathrm{i,j-1}}$	$x_{\mathrm{i,j-1}}(k)$ or x_{fixed}
word 3	$a_{\mathrm{i-1,j}}$	$x_{\mathrm{i-1,j}}(k)$ or x_{fixed}
word 4	$a_{\mathrm{i+1,j}}$	$x_{\mathrm{i+1,j}}(k)$ or x_{fixed}
word 5	α	$x_{\mathrm{i,j}}(k)$
word 6	β	$y_{\mathrm{i,j}}(k)$
word 7	m	$u_{\mathrm{i,j}}$
word 8	τ	$x_{\mathrm{i,j}}(k)$
word 9	1	$y_{\mathrm{i,j}}(k)$
word 10	ε	
word 11	σ	
word 12	τ	

Wave computer core as a whole includes Clock Generator, Control Circuit, Parameter Register, CNPN Circuit, CNPN Cache and communication interfaces as shown in Figure 4.3 [3]. The wave computer core is designed, implemented and programmed by Xilinx ISE Design Suite. When core is programmed, initial values of nodes, control signals and parameters are transferred to the core by using built-in communication interfaces and Matlab functions.

All types of data which are required by the wave computer are sent by the host computer to the control block. Control block receives the data and distributes it to the memory block and parameter register. As shown in Table 4.1, s_1 and s_2 stacks are

loaded with 12 and 9 words, respectively, by stack loader logic in NPE circuit. Each word is 16-bit in width and stores fixed-point number. These words are used by CNPN circuit sequentially. The data that belongs to network slices is carried continuously between the CNPN and the memory block, during the CNN emulation. Each one of 16 NPEs uses its parameters and state variables in order to execute iteration for its own x and y states. When the slowest NPE completes its calculation, CNPN control circuitry sends acknowledgement signals to make NPEs ready for the next iteration. Simultaneously, the network image is captured by observation block and sent to the monitor.

When compared with the previous work in [3], NPE is enhanced in order to obtain better results in the sense of speed and resource utilization. Both designs use the same FPGA. The motivation of this subsection is to change the number representation from floating-point to fixed-point in the NPE operations.

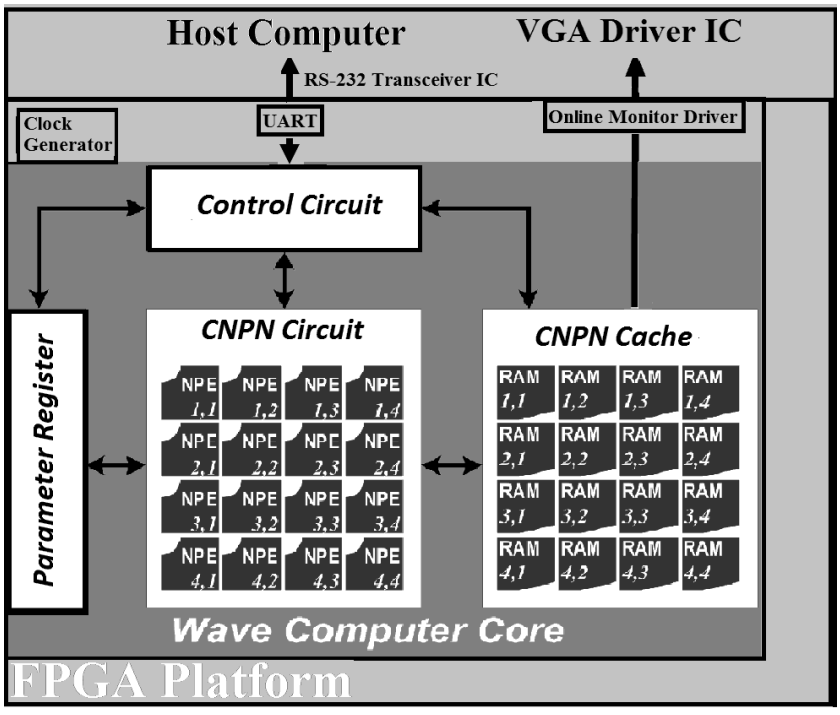


Figure 4.3 : Block diagram of the wave computer core.

State variables, initial values and parameters of the system are 16-bits in width and have $Q7.9$ fixed-point number format. As the represented numbers are signed, format

is changed to $Q6.9$ as 1 bit is reserved for the sign.

$$x_{[\mathrm{B10}]} = \frac{1}{2^n}[-2^{N-1} b_{\mathrm{N\text{-}1}} + \sum_{i=0}^{N-2} 2^i b_i] \tag{4.4}$$

Regarding (4.4) where $x_{[\mathrm{B10}]}$ represents fractional number in decimal, $x_{\min} = -2^m$, $x_{\max} = 2^m - 2^{-n}$, and $resolution = 2^{-n}$. It can be observed that $Q6.9$ signed two's complement format gives us $[-64, 63.99805]$ value range and $1.953 \cdot 10^{-3}$ resolution which are used in NPE operations during CNN emulation.

Three peripheral circuits are used in the Wave Computer Core design as shown in Figure 4.3. Clock Generator generates the low frequency clock signal required by the Core. In this design, the whole implementation covers the 64% of the FPGA chip better than previous work which covers 76% of the FPGA chip [3]. Routed signal paths have high delays for onboard 100 MHz clock signal, therefore clock frequency is divided into 4 to get 25 MHz by a frequency divider circuit. According to the timing analysis, the maximum clock frequency is $46,70$ MHz which has been recorded as 35.85 MHz for the design in [3]. Also, in order to observe the wave evolution, simultaneously the on-line monitor is used [3].

CNPN Cache needs 160 KB of total memory required by emulation of $16,384$ nodes. No external memory is needed as FPGA chip has 272 KB BlockRAM. 16 BlockRAM modules each having 10 KB capacity are defined for this implementation. Words at different addresses can be accessible at the same time because each BlockRAM has a dual-port interface. A-ports of BlockRAMs are used to read and write data that CNPN needs, while B-ports are used for reading the data to depict the network image onto the on-line monitor [3].

Each NPE circuitry has two subcircuits which are Adder Circuit and Multiplier Circuit. These arithmetic circuits work with *permission_input* signals coming from related NPE and operates addition and multiplication operations combinatorially. After addition or multiplication operation, arithmetic circuits send control signals in order to indicate that circuits are ready for new calculation. Adder circuit and multiplier circuit are designed specially for this fixed-point arithmetic Core implementation.

The whole design is implemented on a Xilinx XC2VP30-FF896 FPGA chip. When the whole system operation is examined, input image is scaled to a a 128×128 matrix by a Matlab script at the beginning. Each cell that is equal to one pixel on the network image

functions as a relaxation oscillator using mathematical model in Wave Computer Core design. FPGA is programmed by using Xilinx IMPACT interface and initial values of the network image, parameters and control signals are transferred from Matlab to FPGA chip through RS-232 transceiver chip.

Data and signals are received by UART on the Core. UART transfers this information to Control Circuit. Control Circuit sends parameters introduced in the mathematical model to Parameter Register and initial values of variables of the network image to CNPN Cache. After operating commands are received by CNPN Circuit, the parameters and initial values are transferred to CNPN. This data is processed regarding to mathematical model. While iterations are in progress, CNPN Cache sends instantaneous network image to a VGA monitor for real-time observation.

Table 4.2 represents resource utilization and latencies of arithmetic circuits that are implemented in the Core. Aim of this table is to compare adder, multiplier and NPE circuit between fixed-point and floating-point design approaches. Table 4.3 represents resource utilization of top-level circuits that are designed in the Core. Aim of this table is to compare CNPN circuit and overall Core design between floating-point and fixed-point design approaches.

Table 4.2 : Comparison of Resource Utilizations and Latencies of Arithmetic Circuits [1].

Arithmetic Circuits	Used Slices	Used FFs	Used LUTs	Maximum Latency
Fixed-point NPE Adder	9	1	18	2
Floating-point NPE Adder	126	128	202	20
Fixed-point NPE Multiplier	9	1	17	2
Floating-point NPE Multiplier	35	20	61	2
Fixed-point NPE Circuit	395	416	705	50
Floating-point NPE Circuit	482	590	818	271

Table 4.3 : Comparison of Resource Utilizations of Top-Level Design [1].

Components of the Core	Used Slices	Used FFs	Used LUTs
Fixed-point CNPN Circuit	6041	6952	10552
Floating-point CNPN Circuit	8192	10023	14745
Fixed-point Core Design	8825	9856	14295
Floating-point Core Design	10944	12444	18004

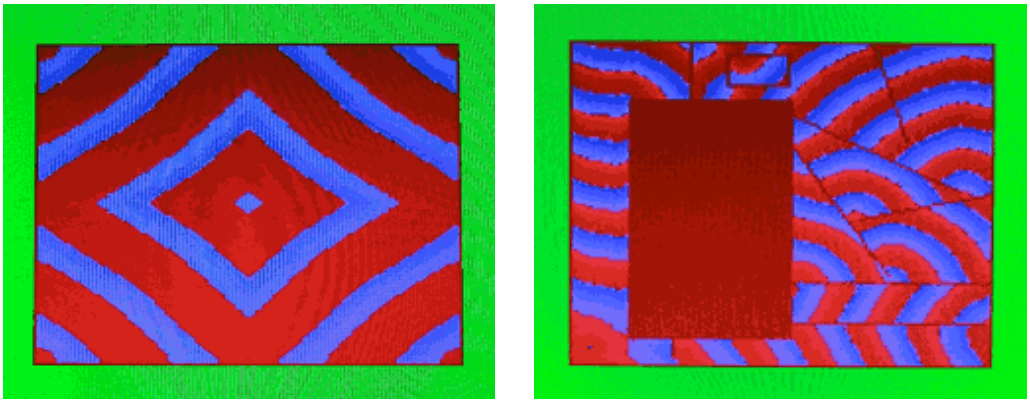
As represented in Table 4.2, Adder Circuit, Multiplier Circuit and NPE Circuitry are reduced approximately to 7%, 25%, 82% of their original ones, respectively, in the meaning of resource usage. Latencies of Adder Circuit and NPE Circuit are shrunk to 10% and 18%, respectively, while latency of Multiplier Circuit can not be decreased any more.

In Table 4.3, it is shown that the resource usage of CNPN circuit and overall Core design are reduced approximately to 74% and 81%, respectively. Also resolution of the numbers was decreased to 1.95310^{-3}. The average latency of the NPE is decreased compared to the previous work [3]. The latencies play important role in the performance as any pipeline has not been realized in the design. In Figure 4.4, example network images, which are depicted and emulated by the Core are represented. Autowave illustrations in Figure 4.4(a), Figure 4.4(b) and Figure 4.4(c) use different initial conditions and network configurations. Figure 4.4(d) illustrates a traveling wave propagation.

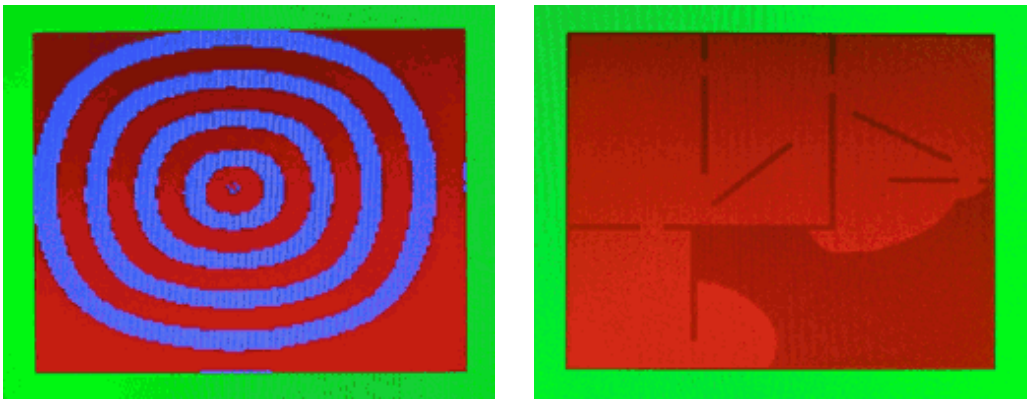
4.1.2 Implementation of relaxation oscillator network on GPU

Parallelism in image processing can be traced back to Chua and Yang's articles that proposed Cellular Neural Networks (CNNs) in 1988 [9, 10]. Chua and Yang merged neural networks and cellular automata's two dimensional grid structure, as a result analog nonlinear circuit clones came into use at the beginning of 90s. In 1993, Chua and Roska reviewed how CNN would be applied to many different image and video processing applications by software simulations and VLSI realizations [84]. CNN became a paradigm for image processing applications just in five years. In the same year, the first algorithmically programmable analog array computer called CNN-Universal Machine (CNN-UM) was proposed and implemented [13]. This machine combines analog array and logic parts without any analog to digital or digital to analog converters. The CNN-UM stores analog instructions that are executed on its analog array.

Cellular nonlinear network simulations need highly parallel computing structures in hardware to obtain practical result in real-time. Moreover, they should be flexibly programmable to serve in many wave computing applications from video processing to robotic locomotion. As shown in [85] and [86], using GPUs for their CNN simulation



(a) (b)



(c) (d)

Figure 4.4 : Autowaves and traveling wave are obtained on the network which is emulated by the Core by using different network configurations and initial conditions [4]: a) autowaves generated from corners, b) autowaves in medium with obstacles, c) autowave generated by the center cell, d) traveling wave in medium with obstacles.

is better than CPU simulations. In this subsection, it is investigated whether a GPU outperforms a CPU or the FPGA wave computer during active wave generation, and the possibilities of using GPU in top-view based path planning application in real-time dynamical environment.

Since Graphical Processing Units (GPU) started to be used in areas apart from visual processing, computation intensive applications formerly could be performed on super computers or on special hardware now can be done on GPUs. Depending on the size of the problem, difficult computations now can be done even on a video card of a laptop. Moreover, main GPU vendors make supercomputers using GPUs for very large scale computations [87]. By using dedicated libraries for GPUs such as NVIDIA CUDA [88], AMD APP SDK [89], it is easy to implement computations with just knowing GPU basics and C programming language. If vendor dependent libraries

are used, obviously the code will be device dependent. Open Computing Language (OpenCL) [5] solves device dependency problem. It is basically a C library with some extensions like the other libraries. It was not proposed just for GPUs. Software written with OpenCL can be run on CPU, GPU and any OpenCL compatible parallel computing device. Since it is not dependent to any hardware it is more portable. Besides, it has free license, which makes it possible to be used and distributed among other software companies and researchers. As an example, Qt is a well known library for C++ GUI applications, and it now supports OpenCL with QtOpenCL wrapper library. This library makes code easy and short, and also introduces OpenCL to C++. There are other commercial solutions that support GPU technologies such as Matlab. Currently, Matlab only supports CUDA for some of NVIDIA GPUs [90].

GPUs are now commonly used for scientific computations in a wide area from mathematical finance to physics. Spatio-temporal wave is a topic which is studied by physicians, mathematicians and engineers for different purposes. Active wave simulation and using it in real-time for robot navigation is one of those purposes. Waves generated in CNN have been used for robot navigation but not restricted to that topic. Electrical waves in the heart show similar patterns to the patterns that CNN generates [91]. Obviously the waves in the heart move in 3D. Simulation of waves created by CNN moving in a 3D media has been presented in [92] which employs a software on PC and has been intended to be a first step to investigate the waves in the heart and issues of robot navigation in 3D.

Hardware offered by GPUs and the hardware implementing CNNs, for example the one in Subsection 4.1.1, are very similar in functionality. Multiple processors, local memories for those processors and controllers (host) for the whole hardware are implemented in CNN emulating hardware. These are all given by a GPU. The issue to handle is just to program it for parallel computation. Conceptual device architecture of OpenCL given in Figure 4.5 shows that an architecture that supports OpenCL is the requirement for 2D and 3D active wave simulation.

Here, active waves in 2D and 3D media have been simulated on GPU. Basically, (3.1) is implemented with forward Euler iteration considering parallel processing. In order to create a portable implementation, it is decided to use OpenCL. By this way, both AMD and NVIDIA GPUs may be targeted. QtOpenCL library which is a

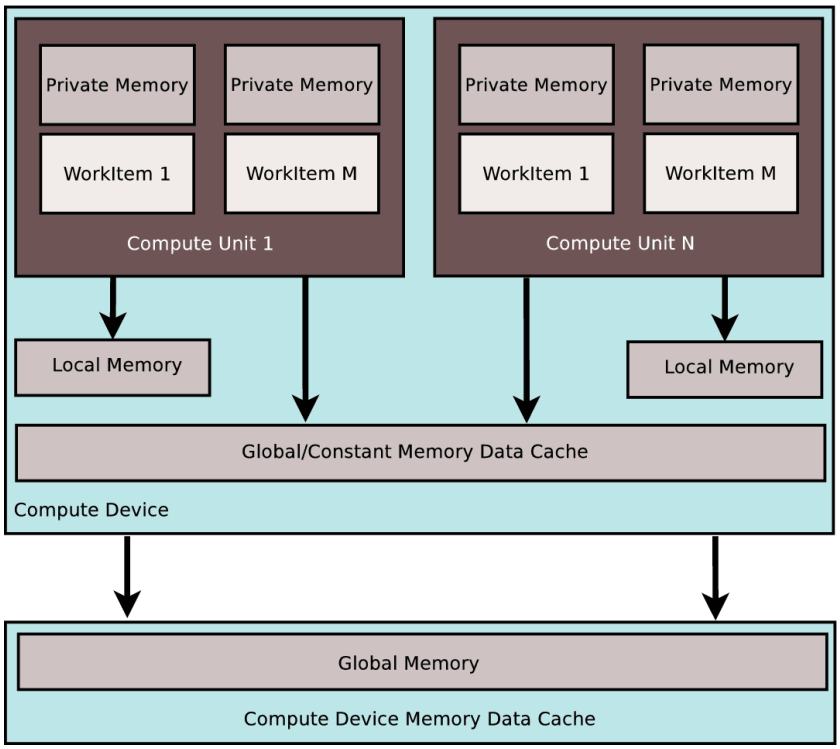


Figure 4.5 : Conceptual device architecture of OpenCL with computing units. Host is not shown. Image was redrawn from the figure in [5].

wrapper library of OpenCL has been used. Using arrays (QCLVector) and instantiating kernels (QCLKernel) are more convenient with this library. OpenCL kernel has been implemented to do one iteration step. As shown in Program 1, the kernel is called more than once and just change the arguments (array pointers) to give the previous result as an input for more iterations. argNewPtr and argOldPtr represent all the inputs and outputs. constVal represents the constants which are used in the iteration.

Program 1: Definition and instantiations of iteration kernel function

---

```
__kernel void
autoWave(__global float *dataNew,
         __global float *dataOld,          //outputs and inputs
         __const float constants)          //constant values
{...}

//function calls
```

autoWave(argNewPtr, argOldPtr, constVal);

autoWave(argOldPtr, argNewPtr, constVal);

autoWave(argNewPtr, argOldPtr, constVal);

...

32-bit (single-precision) floating point arithmetic is used because not every GPU supports 64-bit. Besides, according to experiments 32-bit is a sufficient precision for this work. As seen in Program 1, argNewPtr and argOldPtr are changed between two consecutive iterations to give previous result as an input. The numerical analysis does not include any complicated (time consuming) functions. It just includes addition, multiplication and comparison. However there are harder parts to solve in the spatio-temporal wave simulation, such as boundary values. In other words, whether the node is on the edge, corner or not. Since there is no neighbor connection to boundary nodes, the neighbor values should be the same as the cell itself or a constant. Therefore, in the kernel function code a location type variable as an input is put and the iterations are computed accordingly. It is obvious that location type variables have to be saved in an array like the other input-output values. Program 2 shows the details of one step of the iteration of a 2D network.

Program 2: One step of the iteration

$ti = a * (xijOld.w + xijOld.x + xijOld.y + xijOld.z);$

if $(xOld > 1.0f)$ {

  $tg = -m * (xOld - 1.0f);$

}

else {

  if $(xOld < -1.0f)$ {

    $tg = -m * (xOld + 1.0f);$

    }

    else {

      $tg = 0.0f$

```
        }
}
//new value of x
retval.x =
xOld + delta * (beta * yOld + alfa * xOld + tg + ti);
//new value of y
retval.y =
yOld + delta * (-epsilon * yOld + epsilon * xOld);
```

In program 2, $xOld$ and $yOld$ are state inputs, $xijOld$ is the contribution (input) from neighbors, $retval.x$ and $retval.y$ are outputs. ti and tg are temporary variables. Rest of the variables are constants that are explained in (4.1). In 3D case, contribution of the neighbors was computed as $ti = a * (xijOld.s0 + xijOld.s1 + xijOld.s2 + xijOld.s3 + xijOld.s4 + xijOld.s5)$ since there are bottom and top neighbors.

In this subsection, better performance is achieved than the systems that have been implemented on FPGA [3] and CPU [93]. Table 4.4 shows the performance comparison between GPU, CPU and FPGA implementations. Also, it should be noted that, a larger network $(256 \times 256 \times 256)$ on GPU has been simulated and 36 iterations-per-second has been achieved in 3D case.

Table 4.4 : Comparison of 128×128 sized network simulation performances of three different platforms [2].

	CPU simulation	FPGA emulation	GPU simulation
Iterations per second	~ 35	~ 137	~ 60000
Clock frequency	2.53 GHz	25 MHz	633 MHz

4.2 Partial Reconfiguration of Cellular Logic Network

In this section, a binary wave computing cellular nonlinear network is endowed with partial reconfiguration feature. PR helps designs to exhibit less area costs on reconfigurable logic devices [94]. The main aim is to achieve a primitive system that promises area efficiency in order to develop a massive network that has one

processor to one cell mapping. The proposed Cellular Logical Network (CLN) in this section consists of *fixed* and *alive* cells which are depicted in Figure 4.6. The implemented network is similar to Network 3. Cells on the network are locally coupled according to Von Neumann neighborhood. The proposed CLN is an array of 256 cells within 16×16 regular grid form in 2-dimensional space. At the boundaries, zero-flux boundary condition is applied.

In Figure 4.7, circuit schematic of the alive cell is given. x represents the state of the cell, z is its buffered output, i_j is the input with direction index j, and k is discrete-time variable. Each cell has four inputs from the coupled neighbors which are i_0, i_1, i_2, i_3. The behavior of the alive cell can be expressed as follows:
If $x[k] = 0$:

- $z[k+1]$ remains 0 when every input equals to 0.
- $z[k+1]$ becomes 1 for all input combinations except every input equals to 0.
- $x[k+1]$ remains 0 for all input combinations except every input equals to 1.
- $x[k+1]$ becomes 1 when every input equals to 1.

If $x[k] = 1$:

- $z[k+1]$ remains 1 when every input equals to 1.
- $z[k+1]$ becomes 0 for all input combinations except every input equals to 1.
- $x[k+1]$ remains 1 for all input combinations except every input equals to 0.
- $x[k+1]$ becomes 0 when every input equals to 0.

This behavior is temporally illustrated for the center cell in Figure 4.8. This cell is functionally equivalent to Network 3 with traveling wave configuration.

Fixed cells do not effect the states of their neighbors as an excitation (wave) source. As an example, let us consider a fixed cell and its north alive neighbor. Assuming that state of the neighbor is 0, and its output is 1, then north input of fixed cell becomes 1. Fixed cell sets its north output to 1, since it only reflects its inputs to its outputs. Here, like the boundary, zero-flux boundary condition occurs for fixed cells. This reflection

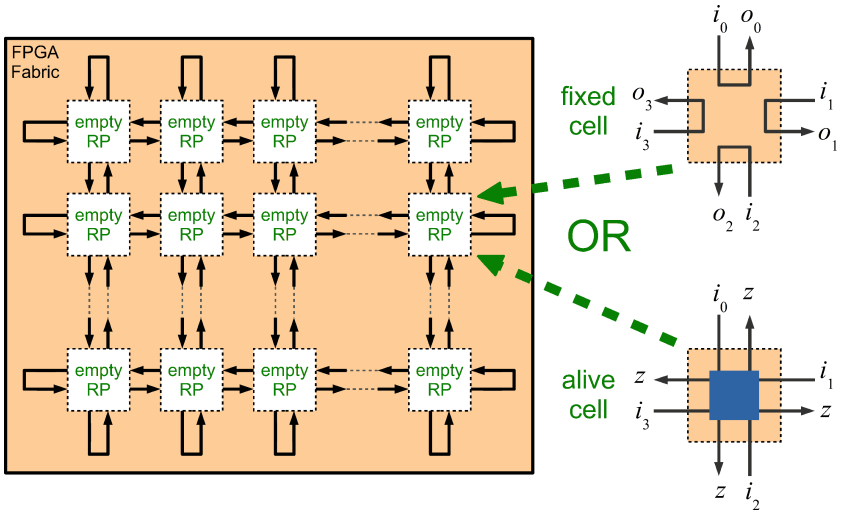


Figure 4.6 : A regular-grid network that consists of empty Reconfigurable Partitions (RPs) on the FPGA. The favor of partial reconfiguration is the opportunity to load the *fixed* cell design or *alive* cell design, both fits to empty RP, whenever it is required in run-time.

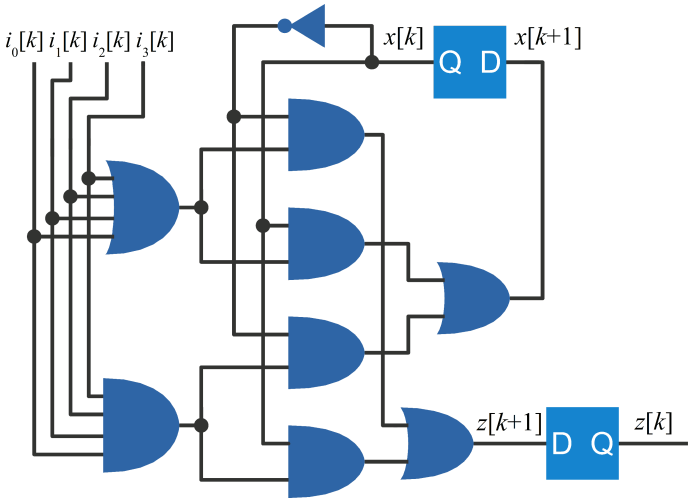


Figure 4.7 : Circuit schematic of the alive cell. The required memory is a 2-bit register. The function generation needs a few logical gates. On an FPGA-like device, it can be implemented by two 5-input Look-up-tables. Circuit has four single bit inputs (i_0 to i_3) and a 1-bit output z.

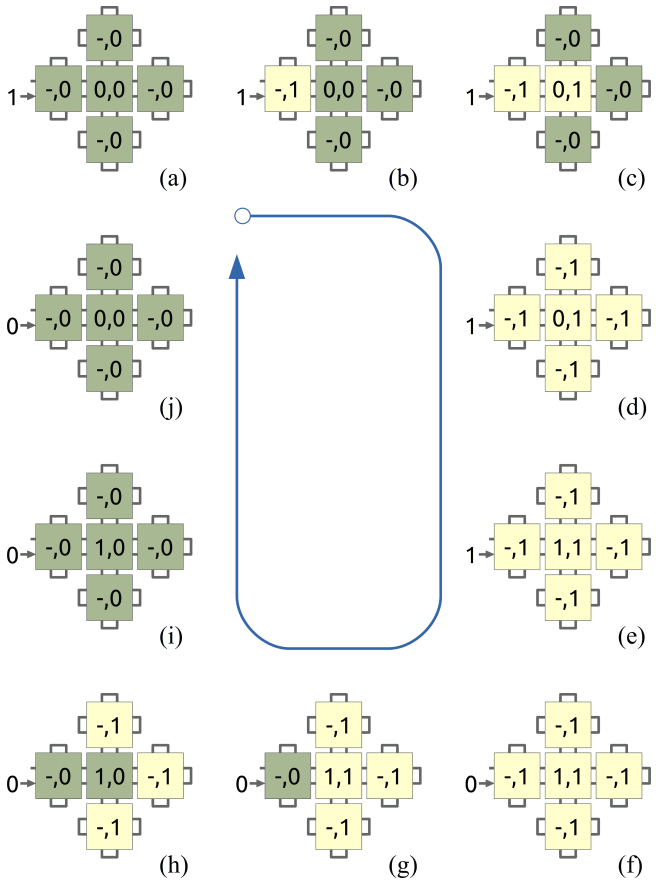


Figure 4.8 : From a to j each subplot represents a 5-cell network in plus-sign formation with zero-flux boundary condition. The initial condition is given in (a) and each consecutive subplot shows following discrete moment. The numbers given within the cells represent the state and the buffered output $(x[k], z[k]$, together. As noticed, the 1 input from the boundary affects the center cell's output in c and propagates immediately in d moment. When all the cells coupled to the center cell has 1 output, it switches its state x to 1 as in e. Similar dynamics is observed when 0 is injected to the network as in between f and j.

only changes the state of the cell if the other three inputs of the alive cells are all 1. Otherwise, the alive cell's state will keep its current value.

In this CLN with partial reconfiguration feature, not only the architecture but also the initial states of the alive cells are important and managed in the reconfiguration process. Therefore, states of alive cells are determined before partial reconfiguration takes place. If an alive cell with improper state is partially reconfigured, then this cell acts like a wave source after reconfiguration, and produces undesired behavior.

The proposed system is implemented on a Xilinx Virtex-5 ML-501 Evaluation Platform, which has Xilinx XCVLX50FFG676 FPGA chip. The implementation has the CLN and an UART circuitry. UART with 115200 baud rate is used to transfer state matrix and output matrix of the network to a PC. FPGA chip used in this implementation has 7200 slices. Each slice has 4 flip-flops and 4 six-input look-up-tables. The proposed system uses 1517 look-up-tables and 874 flip-flops. This corresponds to 5% of total look-up-tables, and 3% of total flip-flops available on the FPGA chip.

Partial reconfiguration is a feature that allows users modify a region of the design currently operating on an FPGA (See Figure 4.9). Tools that are used to design the CLN with partial reconfiguration feature are Xilinx ISE v13.2 and PlanAhead v13.2. Xilinx ISE is used for synthesizing design. PlanAhead is used for defining region of reconfigurable partition, implementing design, and generating configuration files for the target FPGA. The partial configuration files may be loaded either directly from PC or from the configuration EEPROM connected to the FPGA via the Internal Configuration Access Port (ICAP). ICAP is an in-FPGA circuitry which is used for partial reconfiguration of the FPGA while the configuration process is triggered and controlled by the system on the same FPGA. This feature can be considered as the self reconfiguration of the FPGA. The maximum clock frequency of the 32-bit wide ICAP module is 100 MHz which means reconfiguration by ICAP provides 3.2 Gbps bandwidth. In this work, ICAP module is not used and partial reconfiguration is controlled by PC, because the proof of CLN's partial reconfiguration concept is aimed, instead of gaining high performance output.

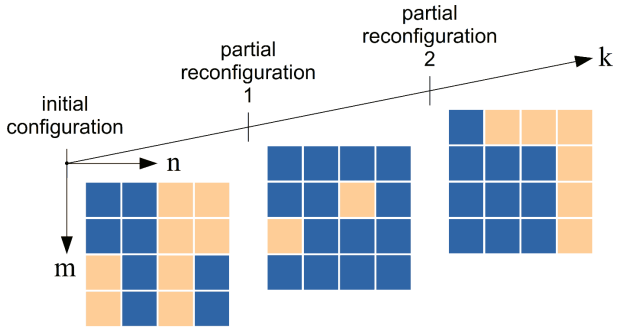


Figure 4.9 : The evolution of the network configuration in 3D spatiotemporal space. m and n axes are vertical and horizontal axes of the 2D network, respectively. k is the time axis. The configuration of the network can be changed any time during the operation as illustrated by two partial reconfiguration events. Partial reconfiguration only occurs for the cells that are decided to be swapped their function.

For the proof of the concept, an artificial scenario is composed. The goal is to change the network structure using partial reconfiguration and exhibit wave propagation in an obstructed medium. Therefore, the 8-th column of the CLN is defined as a reconfigurable partition (RP). Each configuration that can be placed into this RP is called Reconfigurable Module (RM). In this work, three different RMs have been used. First RM consists of 16 alive cells with initial states $x[k] = 0$. Here, it should be noticed that the initial time of a RM is the actual time of the design already running on the FPGA when the reconfiguration occurs. Hence, RM may be initialized after the initialization of other part of the design, so $x[k]$ is used instead of $x[0]$ in the expression. Second RM consists of 15 fixed cells, and one alive cell. Alive cell is placed in the 8-th row. Initial state of alive cell is 0. Third RM is same as second RM except the initial state of its alive cell which is 1.

Figure 4.10 depicts the evolution of the design with the composed scenario. According to the scenario, CLN that consists of only alive cells, with 0 initial states, has been initially loaded onto FPGA. Logical 1 input is applied from the boundary input to alive cell which is located in the first column second row. Light yellow squares represent alive cells whose output equals to 1. Green squares represent alive cells whose output equals to 0. Black squares represent fixed cell. Figure 4.10(a) shows the outputs of CLN after one clock period. Figure 4.10(b) shows the outputs of CLN at the end of 5-th clock period. After five clock periods, network is stopped. Considering that, states of all alive cells are 0 in the RP, second RM has been loaded onto FPGA. Figure 4.10(c) shows the output of the CLN after reconfiguration takes place. After reconfiguration,

system runs for 25 clock periods. From Figure 4.10(d) to Figure 4.10(h), propagation of the trigger-wave is observed. Then, third RM is loaded onto FPGA. Since initial state of alive cell is 1, and also the states of all cells on the network are 1, no new excitation is observed. In the following clock period, the second RM is loaded onto FPGA again with its 0-initialized alive cell. Figure 4.10(i) shows that this causes propagation of a zero wave generated from the alive cell in CLN. After reconfiguration, system runs for 20 clock periods. From Figure 4.10(i) to Figure 4.10(l) propagation of the zero-wave is depicted. According to the observed behavior of the system, it can be noted that if a cell is reconfigured with a different state, this cell acts like wave source in CLN. Therefore, for each partial reconfiguration, states of cells that are going to be reconfigured must be considered to prevent undesired behaviors.

It should be noticed that assigning the 8-th column as the RP is not an obligation. It is just a preference according to the scenario. For a full customization, each cell on the network should be implemented as a RM, and as depicted in Figure 4.6, every cell location can be prepared as an empty RP. However, the design-time increases proportionally to the RP count. The designer may overcome the long design treadmill by scripting the design tools. Certainly, all RMs for all RPs that the application requires should be pre-synthesized and pre-implemented during the run-time.

When the partial reconfiguration feature is used in the design, the trade-off occurs between the performance and area consumption. Obviously, the time consumed during the reconfiguration process reduces the system performance. On the other hand, this approach results in a better area consumption. An alternative to this design which consists of run-time switchable cells consumes more area because the cells should have the circuitry to emulate both the fixed and alive behavior. Moreover, the control logic and routing are required to select the cell behavior which means additional area cost to the design.

Partial reconfiguration feature yields more compact cell designs that increase the network size on a single chip, since it removes the need of implementing all possible functions for a single spot. It provides loading the required function to the spot whenever it is required in run-time. Thus, it promises to shelve the complex cell designs composed by all-possible functions and their multiplexing circuitries. However, the length of exhausting design process intimidates the designer. Thus,

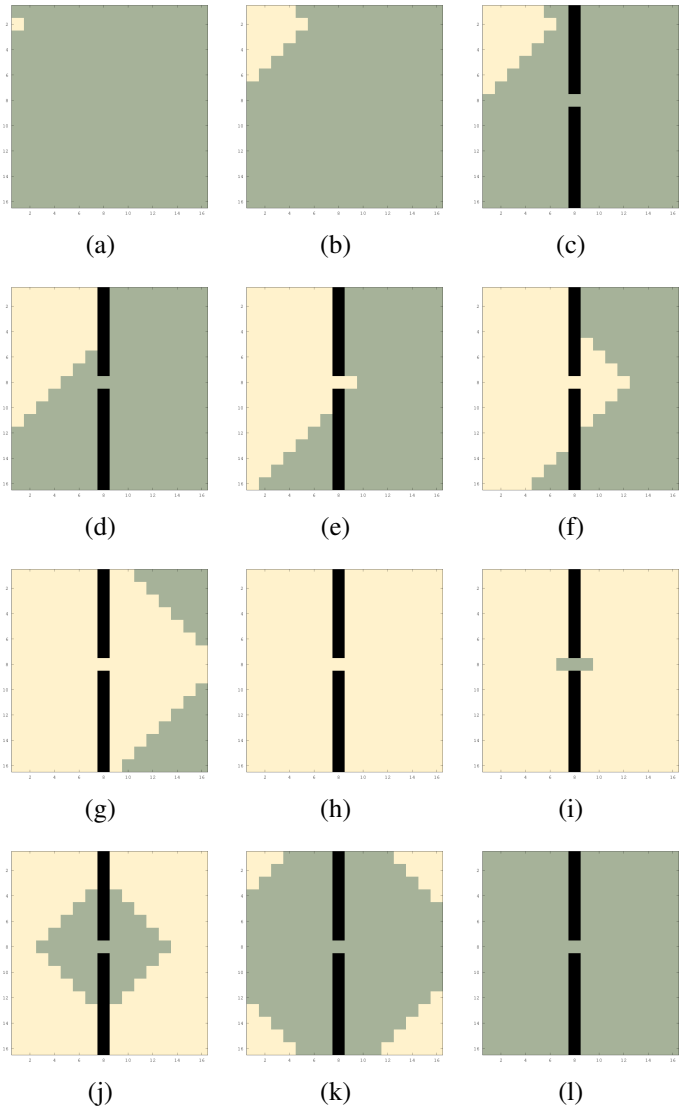


Figure 4.10 : The evolution of the trigger-wave on the Cellular Logical Network manipulated by partial reconfiguration. 2D subplots are the snapshots of the network output $Z[k]$ in discrete time sequence: a) $k = 1$, b) $k = 5$, c) $k = 6$, d) $k = 10$, e) $k = 15$, f) $k = 18$, g) $k = 23$, h) $k = 30$, i) $k = 32$, j) $k = 36$, k) $k = 42$, l) $k = 50$. In the figure, light yellow spot represents the alive cell whose output is 1. Green spot represents the alive cell whose output is 0. Black spots represent the fixed cells whose output is determined by the zero-flux boundary condition.

the desire to the better tools which meets the needs for repetitive structure of cellular networks is standing.

4.3 Implementations for Time-delay Chaotic System

From the implementation point of view, the complexity of the time-delay block is the main drawback of chaotic time-delay systems. Time-delay unit is commonly implemented using a cascade of filters, such as T-type LCL filters [20, 53] or Bessel-type filters [95] such that required delay is set to the group delay of the network. In [20] 30 LCL sub-circuits which is a total of 60 inductances and 30 capacitances has been used to implement the delay line. Mykolaitisa *et al.* [96] have used a coaxial transmission line which is a specialized cable to implement the time-delay unit. Bucket brigade device which is also known as analog delay line has been used by Losson *et al.* [32]. In this device, the analog delay line samples and delays the signal by storing it in the array of capacitor circuits. Furthermore, output of the delay circuit is fed through a Bessel-type filter. Delay part of the system can alternatively be implemented on a digital circuit [25]. To this end, analog signal in the input of the delay block is converted to a digital signal by an analog-to-digital converter (ADC), then this signal is delayed on a digital circuit. The delayed signal in the digital circuit is converted to an analog form and used in the system. From the circuit implementation point of view, implementation methods of the time-delay block of chaotic time-delay systems do not make chaotic time-delay systems an alternative for possible applications. Here, the aim is to use a digital circuitry to implement delay block $\mathrm{T}_{\mathrm{Delay}}[\cdot]$. In order to use a digital circuitry, the output signal of the $h(\cdot)$ function block might be converted into digital form by an ADC. Note that since the function $h(\cdot)$ is a binary output function, quantization and encoding operations performed in standard ADC are not required in this implementation.

Delay devices are based on the propagation delay of any physical component. Apart from the transmission lines for analog implementations, the fundamental delay device is the inverting buffer (NOT gate) for digital implementations. Without propagation delay, even flipflops can not react to the edge on the clock signal [97]. In this section, the digitally implementable buffer chain, flip-flop chain and a asynchronous delay circuit are discussed with their implementation in time-delay chaotic oscillator.

Figure 4.11a depicts a non-inverting buffer chain with N_{Buffer} pieces of buffers, each providing τ_b amount of delay. It is assumed that the buffers are identical and has no variation in τ_b. With these buffers, $\tau = N_{Buffer} \cdot \tau_b$. In Figure 4.11b, the delay line is a D-type flip-flop chain. These flip-flops are clocked by $c(t)$ signal with T_c period. The total amount of delay of these flip-flops is $N_{D-FF} \cdot T_c$. However, the flip-flops do not delay the signal $x(t)$. Instead, $x(t_k)$ is delayed. $x(t_k)$ is the k-th sample of $x(t)$ signal and is held during the concerned $c(t)$ period.

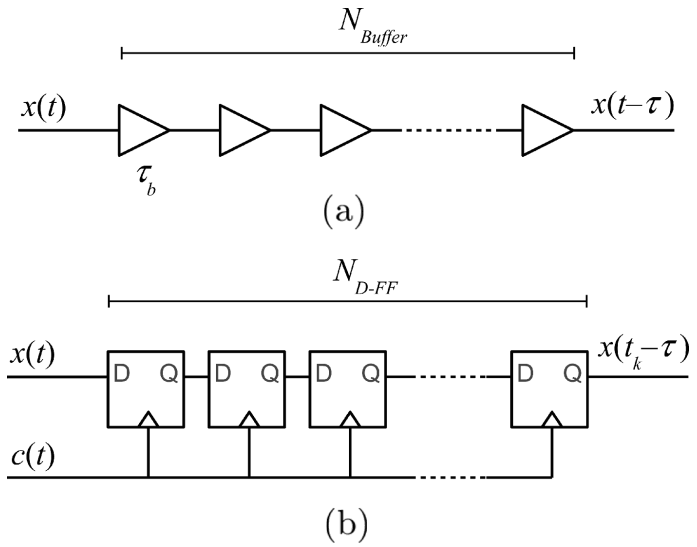


Figure 4.11 : Two different implementations of the binary delay line: a) Non-inverting buffer chain, b) D-type flip-flop chain. Both chains have finite number of units. The former chain functions asynchronously, but the latter one functions synchronously with $c(t)$.

The buffer chain responds to the asynchronous events on the input signal (either a positive edge or a negative edge) in continuous-time, but the flip-flop chain responds periodically (synchronously) sampled and held values. In spite of this difference, they have a fundamental similarity. They have spatial discontinuity which is also called granularity. As a result, the minimum event interval of the input signal is limited to τ_{min} (Figure 4.12) for proper functionality in both chains. For buffer chain, τ_{min} equals to τ_b. The input signal which has successive events with interval shorter than τ_{min} can not be delayed, because a buffer does not respond a second event if the previous event does not appear at its output yet [98]. For flip-flop chains, τ_{min} equals to T_c. Only one event per period can be turned out at flip-flops' output. Hence, the buffer chains and the flip-flop chains function properly if the shortest event interval recorded on the

input signal is greater than τ_{min}. There is a trade-off between the τ_{min} limitation and the number of delay units on the chain for a τ. If a transmission line, which is modeled by infinitesimal distributed elements, is considered, the spatial discontinuity disappears and τ_{min} becomes infinitesimal.

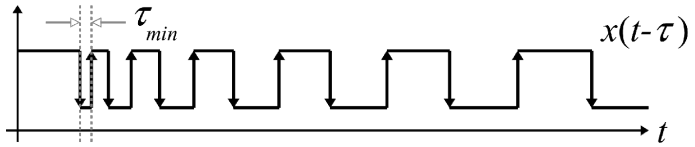


Figure 4.12 : There is a structural discontinuity (granularity) in the delay unit chains (Figure 4.11) which causes a limitation on the minimum event interval (pulse width) of the binary signal to be delayed. The given waveform belongs to the output of a chain delay line when a pulse-width sweep is applied to its input. The pulses narrower than τ_{min} is filtered out.

In Figure 4.13, the input signal $x(t)$, the output signal $x(t-\tau)$ of the buffer chain (Figure 4.11a) and the output signal $x(t_k-\tau)$ of the flip-flop chain (Figure 4.11b) are plotted. The deformation in the output signal of the flip-flop chain should be noticed. Although it is wider in the input side, the interval between the negative edge and the positive edge is $2T_c$ for the flip-flop chain. The event intervals in the output signal of flip-flop chain are always positive integer multiplies of T_c. Consequently, the buffer chains deform the signal if their τ_b is not stationary, but flip-flops deform even if T_c is stationary. The τ_{min} limitation can be controlled by setting T_c value for flip-flop chains. Owing to the relatively smaller values of τ_b in comparison with T_c, more pieces of buffers than the flip-flops have to be used for a same amount of delay.

It should be noted that, both flip-flop and buffer chain are suitable for the circuit-integration. According to these properties, they are candidates in the implementation of chaotic time-delay systems. The use of flip-flop (D-type) chain to delay the binary output of the nonlinear feedback part of the introduced system yields a new system which is a sampled-data feedback system [39]. And, this implementation provides the proposed time-delay sampled-data chaotic system.

The following subsections from Subsection 4.3.1 to Subsection 4.3.7, present the implementations for time-delay system, Oscillator 4 in Section 2.3, and its network in Section 3.2. The value contributed to literature by this chapter is listed below.

1) Two new asynchronous delay circuits proposing area efficiency with their FPGA

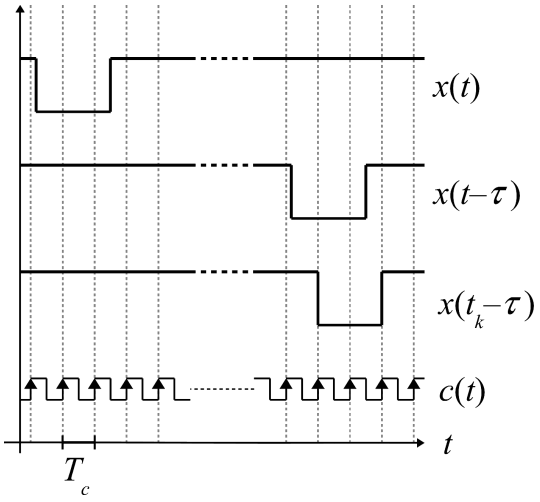


Figure 4.13 : The input signal $x(t)$, the output signal $x(t-\tau)$ of the buffer chain (Figure 4.11a), the output signal $x(t_k-\tau)$ of the flip-flop chain (Figure 4.11b) and the clock signal $c(t)$. Both outputs obey the $\tau_{\min}$ limitation. The buffer chain is able to respond its input asynchronously and propagate the input signal always with the same amount of delay, theoretically. The deviation of a buffer's delay (τ_b) is related to the physical implementation facts and environmental conditions. On the other hand, the flip-flop chain samples and holds the input during the T_c period which always causes deformation on output pulse widths.

implementations.

2) The analog compare and integrator circuit with flip-flop chain implementing sampled-data time-delay chaotic oscillator.

3) The analog circuit with novel asynchronous delay components implementing the time-delay chaotic oscillator.

4) 8-bit fixed-point digital implementation of time-delay chaotic system's integrator and comparator parts.

5) The fast digital circuit with only inverter chain for time-delay chaotic system which can be entirely implemented on an FPGA.

6) The new delay line coding scheme for multi-scroll attractor generating chaotic oscillator and its novel nonlinearity.

7) 1D network of sampled-data systems practicing the anticipating synchronization.

8) 1D network of digital emulators of sampled-data systems practicing the anticipating synchronization.

4.3.1 Asynchronous delay doubler for binary delay lines

Implementing a τ amount of delay with a buffer chain is more area consuming than implementing it with a flip-flop chain because of the difference in the number of units on the chain. Besides, the asynchronous response of the buffer chain is a desired feature. Here, the idea which is using the delay unit on the delay line more than one time emerges [99]. This idea can be applied to a buffer chain with proper signal routing. The buffer chain is cut into cascaded subchains at first. Then, every subchain is equipped with a binary signal router. The router can drive the binary signal event to the subchain at the beginning. When the event appears at the end of the subchain, the router can drive inverse of the event to the subchain again. Inverting the event and re-driving it to the subchain again and again can be managed by the router. The buffer based subchains respond asynchronously; therefore this router should respond asynchronously, too.

The Asynchronous Delay Doubler, is a simple binary signal router that activates its slave delay line two times per event. Figure 4.14 shows the cascaded ADDs, each with a slave delay line. The slave delay lines may be buffer chains, transmission lines or any other binary signal delayers.

The ADD has four terminals (Figure 4.15). Two of them are for the slave unit connection. The reminder two terminals are for the master unit connection. The ADD is designed as a simple asynchronous finite state machine.

For preserving simplicity and preventing race conditions, ADD has only four states shown in Figure 4.16. ADD design is based on an assumption: A second event at FM input does not occur until previously delayed event appears at TM output. This interval is the delay provided by the ADD with its slave, and it determines the τ_{min} constraint.

The states are coded with two bits (S_1S_0) as in Figure 4.16. The inputs controlling the state transitions and outputs are given along the arrows in the figure (FM FS / TM TS). The state transitions occur between codes with Hamming distance 1. Possible hazards are prevented in the ADD design. The output functions TM and TS with the transition

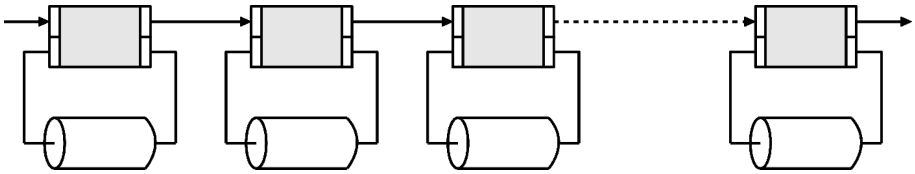


Figure 4.14 : Asynchronous Delay Doubler doubles the amount of delay using one delay unit twice.

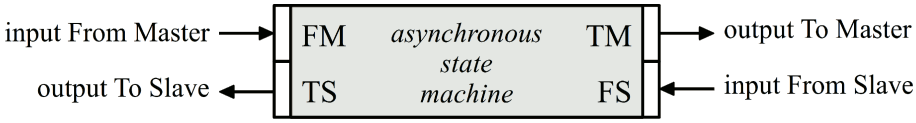


Figure 4.15 : The ADD is an asynchronous state machine with an input of binary signal to be delayed from master (FM), an output of delayed binary signal to master (TM), an output of binary signal to be delayed to slave (TS) and an input of delayed binary signal from slave (FS).

functions S_1 and S_0 are given in (4.5).

$$\begin{array}{ll} TM & = S_1, \\ TS & = (\overline{FM} \cdot \overline{S_0}) + (FM \cdot S_0), \\ S_1 & = (S_1 \cdot FS) + (S_1 \cdot \overline{S_0}) + (\overline{S_0} \cdot \overline{FS}), \\ S_0 & = (\overline{FM} \cdot S_0) + (S_1 \cdot FS) + (S_0 \cdot \overline{FS}). \end{array} \tag{4.5}$$

Every function in (4.5) has arguments less than five. As $TM = S_1$, only three 4-input Look-Up Tables (LUT) are sufficient to implement an ADD. The LUT-based schematic of the ADD is given in Figure 4.17. LUT-based implementation is suitable for reconfigurable devices such as Field Programmable Gate Arrays (FPGAs). Although FPGAs are designed for synchronous circuits, LUT-based ADD can also be implemented on FPGA using some design constraints. These constraints keep the loops (S_1 and S_0 lines) safe from being recognized as errors by the synthesizer tool and preserve the buffer chains from optimization.

As mentioned, the ADD can use any binary delay line. So, an ADD can use another ADD in a recursive way. A delay line with cascaded M recursive ADD blocks is depicted in Figure 4.18. In this structure, the ADD $D^{i,j}$ is the slave of the $D^{i,j+1}$ one from $j = 1$ to $j = Ni - 1$. Each ADD has a propagation delay τ_{ADD} and at the bottom of the recursive structure, there exists a buffer chain with $\tau^{i,0}$ amount of delay. The overall delay provided by the structure in the Figure 4.18 is

$$\tau_{\text{total}} = \sum_{i=1}^{M} \tau_i. \tag{4.6}$$

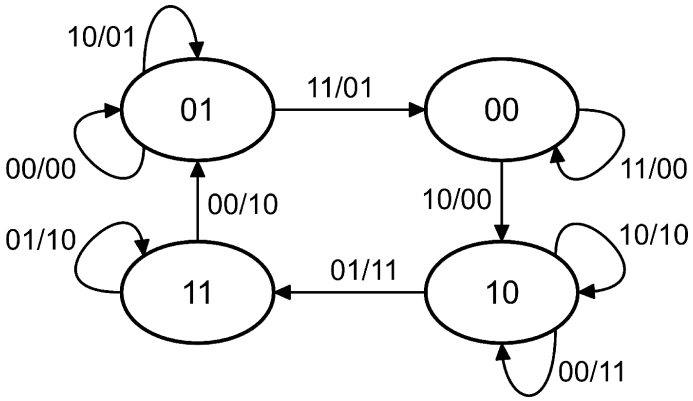


Figure 4.16 : The state diagram of ADD with assigned state codes. Transitions occur through the arrows with FM FS / TM TS signals. Don't-care conditions by means of the minimum input interval assumption reduce the number of states to 4, and the number of state transitions from 16 to 10.

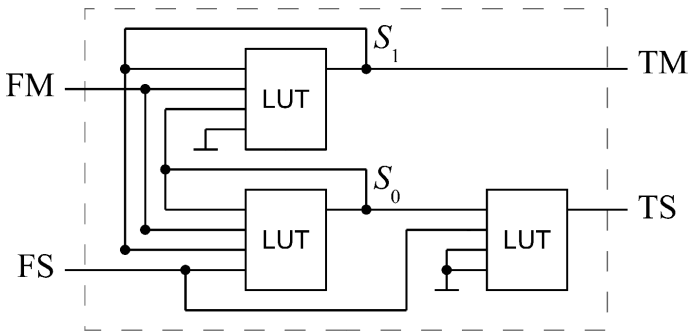


Figure 4.17 : The realization of the ADD is done using three 4-input look-up tables.

Each recursive block on the chain has a delay given by

$$\tau_i = \ldots 2(2(2\cdot \tau^{i,0} \overbrace{+\tau_{\mathrm{ADD}})+\tau_{\mathrm{ADD}})+\tau_{\mathrm{ADD}}\ldots}^{Ni \text{ terms}} \tag{4.7}$$

which can be simplified as

$$\tau_i = 2^{Ni}\left(\tau^{i,0}+\tau_{\mathrm{ADD}}\right)-\tau_{\mathrm{ADD}}. \tag{4.8}$$

Here, the delay provided by nested ADDs increases exponentially while the LUT consumption increases linearly. However, the cost of this efficiency is the growth in minimum event interval limitation which is given by

$$\tau_{min} = \max\left(\tau_i\right). \tag{4.9}$$

When the event interval occurred on TM of ADD is smaller than the τ_{min} constraint, the assumption is violated and the recursive ADD block malfunctions. Therefore, a binary low-pass filter (BLPF) unit is designed to cover the recursive ADD block. The

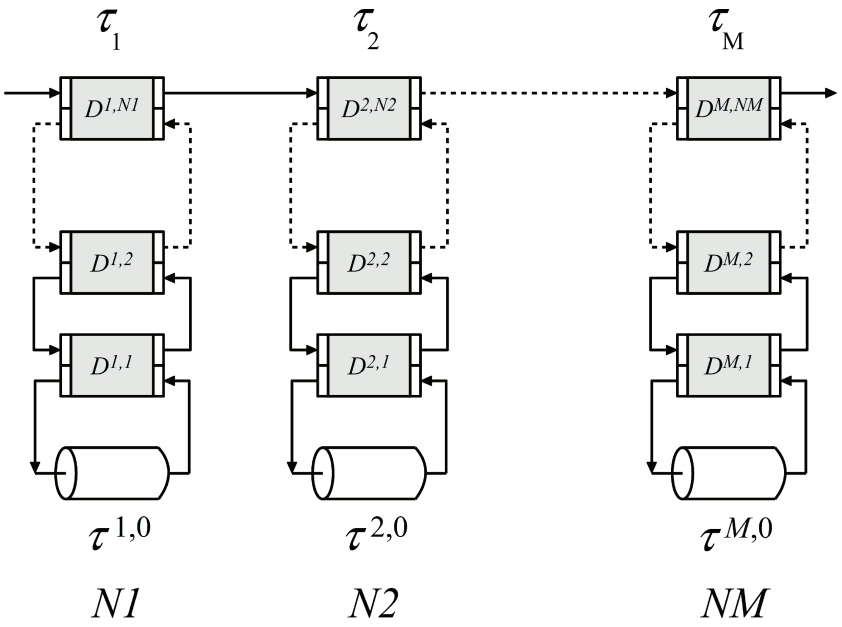


Figure 4.18 : A cascaded recursive structure is constituted with ADDs. Each cascaded block has a delay time τ_i related to its recursion order (Ni), the propagation delay of ADD (τ_{ADD}) and the base delay time ($\tau^{i,0}$).

BLPF is a pulse-width filter which filters out pulses that are narrower than the delay of the BLPF's slave delayer. As the ADD block runs asynchronously, the BLPF should also run asynchronously in continuous-time. The BLPF block is used as the recursion terminator in the recursive ADD block design. The connection of the BLPF is given in Figure 4.19.

The BLPF has FM, FS inputs and TM, TS outputs same as ADD. When an event occurs on the FM input of BLPF, it triggers an event on the TS output. The slave delayer returns this event through the FS input of the BLPF after its delay time. When the BLPF captures this returned event, it checks its FM input. If the signal at FM is at the same level as the signal right after the first event, then it is transferred to the TM output. Otherwise, the pulse starting with the first event is assumed to be a narrow pulse and filtered out. This behavior is illustrated in Figure 4.20. The outputs are also the states for BLPF design and the state transition functions are given by

$$\begin{aligned} TM &= (\overline{S}\cdot TS\cdot(TM+FS)) \\ &\quad +(S\cdot\overline{TS}\cdot(\overline{FS}+TM)) \\ &\quad +(TM\cdot(FS\oplus S)), \\ TS &= (TS\cdot\overline{FS})+((\overline{FS}+TS)\cdot(S\oplus FM)), \\ S &= (\overline{FM}\cdot TS\cdot(\overline{FS}+S)) \\ &\quad +(FM\cdot\overline{TS}\cdot(FS+S)) \\ &\quad +(FS\cdot(S\odot FM)). \end{aligned} \tag{4.10}$$

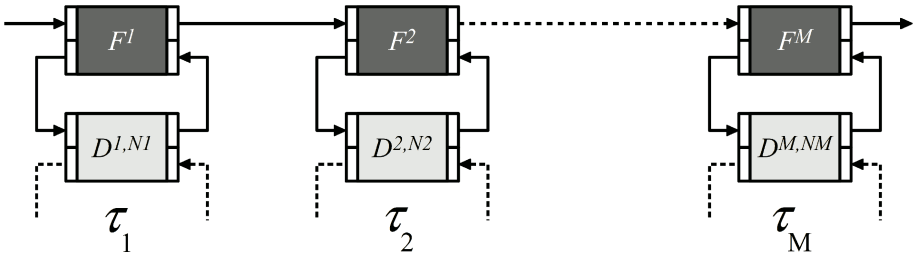


Figure 4.19 : Binary low-pass filters (F^is) protecting the nested ADD blocks against the pulses narrower than τ_i. These filters do not increase the amount of delay significantly. If the blocks ideally have the same amount of delay, filter should be applied to each block. Else, only the block having the greatest delay should have a BLPF.

Same with the ADD, the BLPF is proper to implement using only three 4-input LUTs on an FPGA. Its LUT based schematic is given in Figure 4.21.

4.3.2 Mono-scroll attractor using analog integrator and flip-flop chain

In order to verify the feasibility of the proposed chaotic system, Oscillator 4, a practical circuit that realizes the model in (2.15) is designed and built using off-the-shelf components. The diagram of the designed circuit is given in Figure 4.22 which is implemented using one capacitor, five resistors, three voltage comparators and two CFOAs (current feedback op amps, special op amps offering high-slew rate performance (AD844) [100]) operating in open-loop configuration.

The subcircuit depicted within dashed lines in Figure 4.22 built around two voltage comparators, three resistors denoted by R_1, R_2 and R_3, and the CFOA_1 realizes the required nonlinear function $1-h(x)$, which is the binary inversion of $h(x)$. The nonlinear voltage transfer of this subcircuit is given by:

$$v_h = -V_{CC}R_3\left(\frac{1}{R_1}u\left(v_C - V_b\right) - \frac{1}{R_2}u\left(v_C + V_b\right)\right). \tag{4.11}$$

For $R_1 = R_2$, the nonlinear voltage transfer becomes

$$v_h = \frac{V_{CC}R_3}{R_1}\left[1 - h\left(\frac{v_C}{V_b}\right)\right]. \tag{4.12}$$

It should be noted that $v_h(t)$ corresponds to $[1-h(x(t))]$, where $h(x)$ is defined in (2.12). Output levels of the comparator subcircuit are $\frac{R_3}{R_1}V_{CC}$ and 0. Hence, it has two stable states and it would be possible to implement $\text{T}_{\text{Delay}}[\cdot]$ block with a digital circuit after the output of the function $[1-h(x)]$, which can be represented by the binary digits.

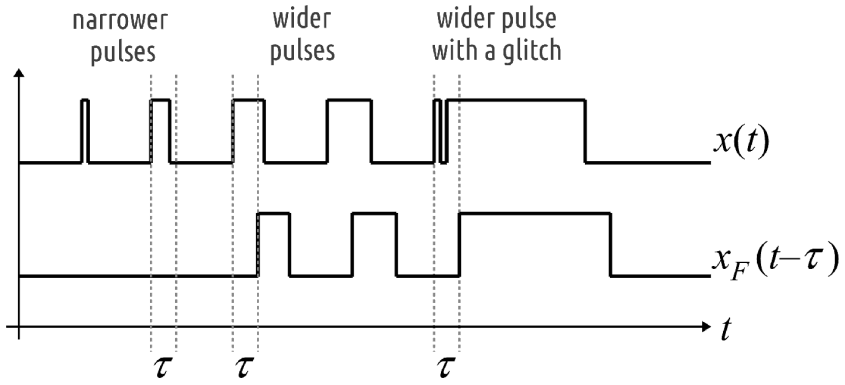


Figure 4.20 : Input signal $x(t)$ with various pulses and glitches, and $x_F(t-\tau)$ which is the output signal of a recursive ADD block with BLPF is plotted. BLPFs filter out the pulses narrower than the delay provided by its slave delayer, which is τ for this case.

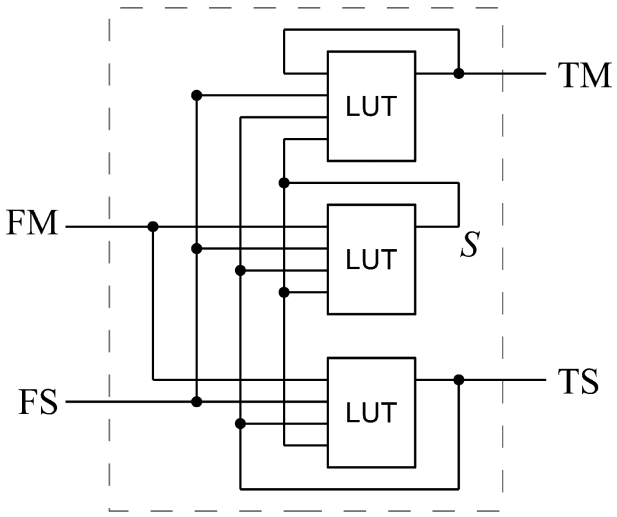


Figure 4.21 : LUT-based schematic of the BLPF. Similar to the ADD, only three 4-input look-up tables are required to implement BLPF.

The subcircuit shown within the gray box realizes the delay function with binary output, $\mathrm{T}_{\mathrm{Delay}}[\cdot]$, and is composed of an FPGA based structure. The delay line is implemented on a low-cost 100K-gate FPGA (XC3S100E-4TQ144) and clocked by 100 MHz frequency signal generated by the oscillator on the board. The length of flip-flop chain (N_{FF}) is set to $20,000$ in order to obtain needed delay time, $\hat{\tau} = 200\mu$s. The output of the flip-flop chain is given by:

$$v_d(\hat{t}) = v_h(\hat{t}_k - \hat{\tau}), \tag{4.13}$$

where $\hat{t}$ is the time, $\hat{t}_k$ is the sampling time and $\hat{\tau}$ is the amount of delay in seconds.

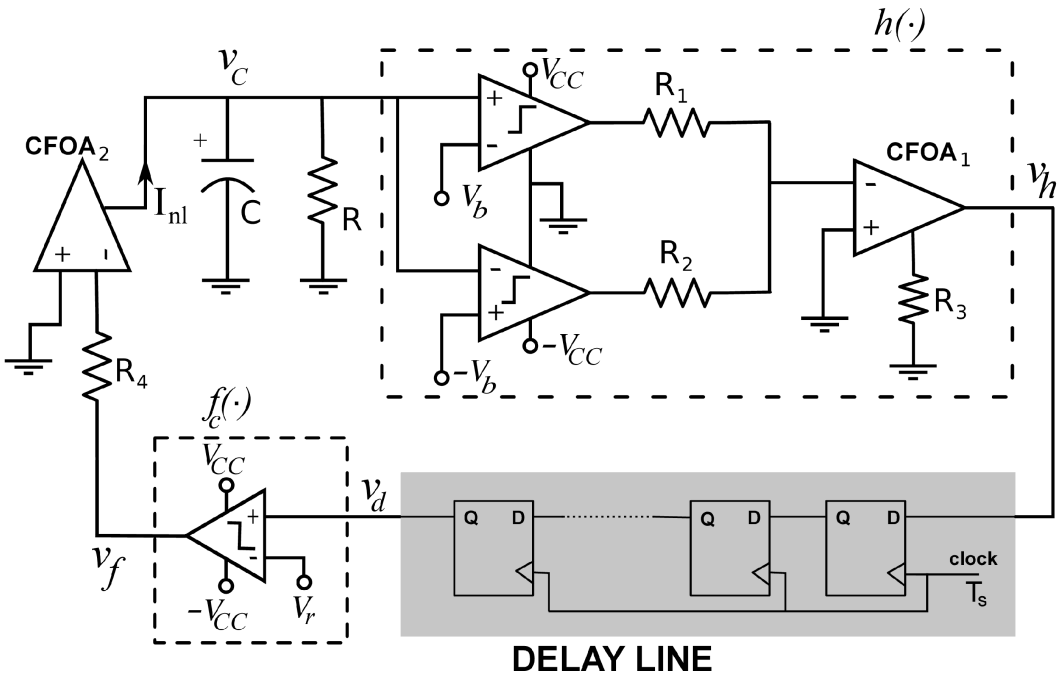


Figure 4.22 : The experimental circuit used to realize the proposed chaotic system (2.15).

A voltage comparator ($V_r = 0.5$V) which implements the block $f_c(\cdot)$ is driven by the output of the delay line which yields

$$v_f(\hat{t}) = V_{CC}\left[2\cdot u\left(\frac{V_{CC}R_3}{R_1}\left[1-h\left(\frac{v_C(\hat{t}_k-\hat{\tau})}{V_b}\right)\right]-V_r\right)-1\right]. \tag{4.14}$$

Briefly, (4.14) can be written by

$$v_f(\hat{t}) = -V_{CC}f\left(\frac{v_C(\hat{t}_k-\hat{\tau})}{V_b}\right). \tag{4.15}$$

The output of this block (v_f) is connected to CFOA$_2$ which is used for implementation of a voltage controlled current source. Routine analysis of the circuit (Figure 4.22) yields to the following equation:

$$\dot{v}_C(\hat{t}) = -\frac{v_C(\hat{t})}{RC}+\frac{V_{CC}}{R_4C}f\left(\frac{v_C(\hat{t}_k-\mathrm{N_{FF}}\hat{T}_s)}{V_b}\right) \tag{4.16}$$

where $\hat{T}_s$ is the flip-flop clock frequency in seconds with $\hat{t}_k \le \hat{t} < \hat{t}_k + \hat{T}_s$.

By defining the following normalized variables $x \equiv v_C/V_b$, $\tau \equiv \mathrm{N_{FF}}\hat{T}_s/RC$, $\alpha = V_{CC}R/R_4$, normalizing time using $t = \hat{t}/RC$, and choosing $V_b = 0.75$V it can readily be shown that (4.16) is equivalent to the model (2.15).

The circuit in Figure 4.22 is built using the following passive component values: $C =$ 5nF, $R_1 = R_2 = 50$kΩ, $R_3 = 33$kΩ and V_{CC} is set to 5V. Hence, the voltage of delay

line input is in the range of $[0, 3.3]$V. The value of R and R_4 are made adjustable to set desired RC and α values.

Agilent's DSO6104A oscilloscope, which has 1 GHz bandwidth through its 4 Gsps rate, is utilized to analyze the system. Time waveform of the capacitor voltage, $v_C(t)$ corresponding to the variable x in (2.8) is captured by this digital oscilloscope. But, the captured signal is downsampled to 1 Msps by the oscilloscope in order to reduce the data size of the waveform. The value of adjustable resistor R is set to 5KΩ with $C = 5$nF and $\hat{\tau} = 200\mu$s to make $\tau = 8$. A 200 ms of sampled signal record, which is 200K samples long, is taken to computer. In this system, only $v_C(\hat{t})$ can be measured, because the binary signal $h(v_C(\hat{t}_k))$ is delayed instead of $v_C(\hat{t})$. Therefore, $v_C(\hat{t} - \hat{\tau})$ is obtained from the signal recorded and transferred to the computer. Phase portrait of a 40ms interval, which is plotted using the recorded data, is shown in Figure 4.23.

Based on the spectrum of the Lyapunov exponents on the $(T_s - \alpha)$-plane (Figure 2.17), it is possible to decrease the number of flip-flops on the delay line. The amount of the delay is the product of N_{FF} and T_s. With a constant proper α value, T_s can be increased, which provides decrease in N_{FF}, and can be still in the chaotic regime. The α values greater than 2 yield a chaotic behavior band for T_s values smaller than approximately 0.2. As an example, for $\alpha = 2.5$, the phase portraits for different T_s values are depicted in Figure 4.24. The phase portraits on the left column of the figure are drawn with the captured data from the circuit realization. The phase portraits on the right column belong to the computer simulation results. $T_s = 0.001$ for Figures 4.24(a) and 4.24(b), $T_s = 0.020$ for Figures 4.24(c) and 4.24(d), $T_s = 0.250$ for Figures 4.24(e) and 4.24(f) while $\tau = 8$ for all experiments. So, the numbers of utilized flip-flops are 8000, 400 and 32, respectively. Apparently, both the simulation and the circuit realization still exhibit the chaotic attractor when N_{FF} is decreased from 8000 to 32.

Moreover, experiments with smaller number of flip-flops than 32 validate Figure 2.17 with non-chaotic behavior. Although it is easy to implement chains longer than 8000 flip-flops using current FPGAs, requiring few number of flip-flops strengthens the simplicity of the presented chaotic system.

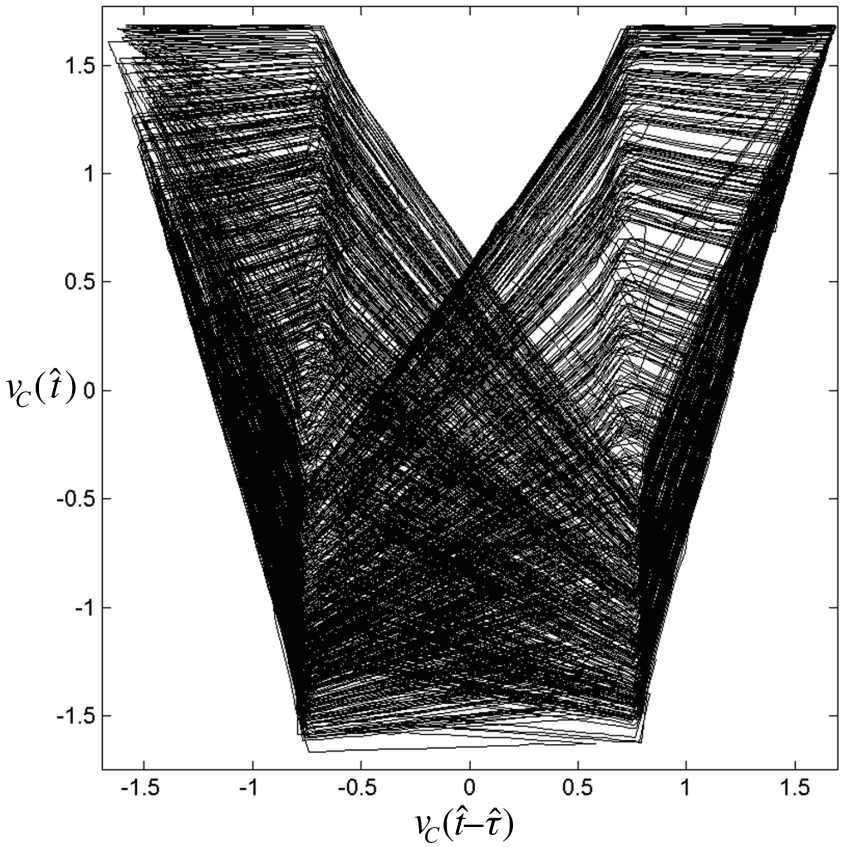


Figure 4.23 : Phase portrait in $v_C(\hat{t}-\hat{\tau})-v_C(\hat{t})$ plane.

4.3.3 Mono-scroll attractor using analog integrator and ADD chain

A delay line using 2048 LUT-based buffers is implemented on Xilinx XC3S500E-4 FPGA which achieves 3.68μs delay. When an ADD is used with a chain of 1024 LUT-based buffers as its slave unit, the delay is still 3.68μs. A 2-nd order nested ADD (using 2 ADDs) with 512 LUT-based buffers also achieves 3.70μs delay. High nesting orders of ADDs create extra delay because ADD propagation delay gains significance, given in (4.8). The flashy result is achieved using 4 LUT-based buffers and a 10-th order nested ADD. Totally 34 LUTs provide 6.29μs amount of delay by this configuration which is 102 times more area efficient than a chain using buffers only. If all logical resources of the target FPGA is occupied by this 34-LUT configuration and if they are connected in cascaded form, 1.71ms total amount of delay is achieved which is already impossible with pure buffer chain.

The ADD design is tested within a time-delay sampled-data feedback system [101] in which the delay line on the feedback is implemented by a D-type flip-flop chain

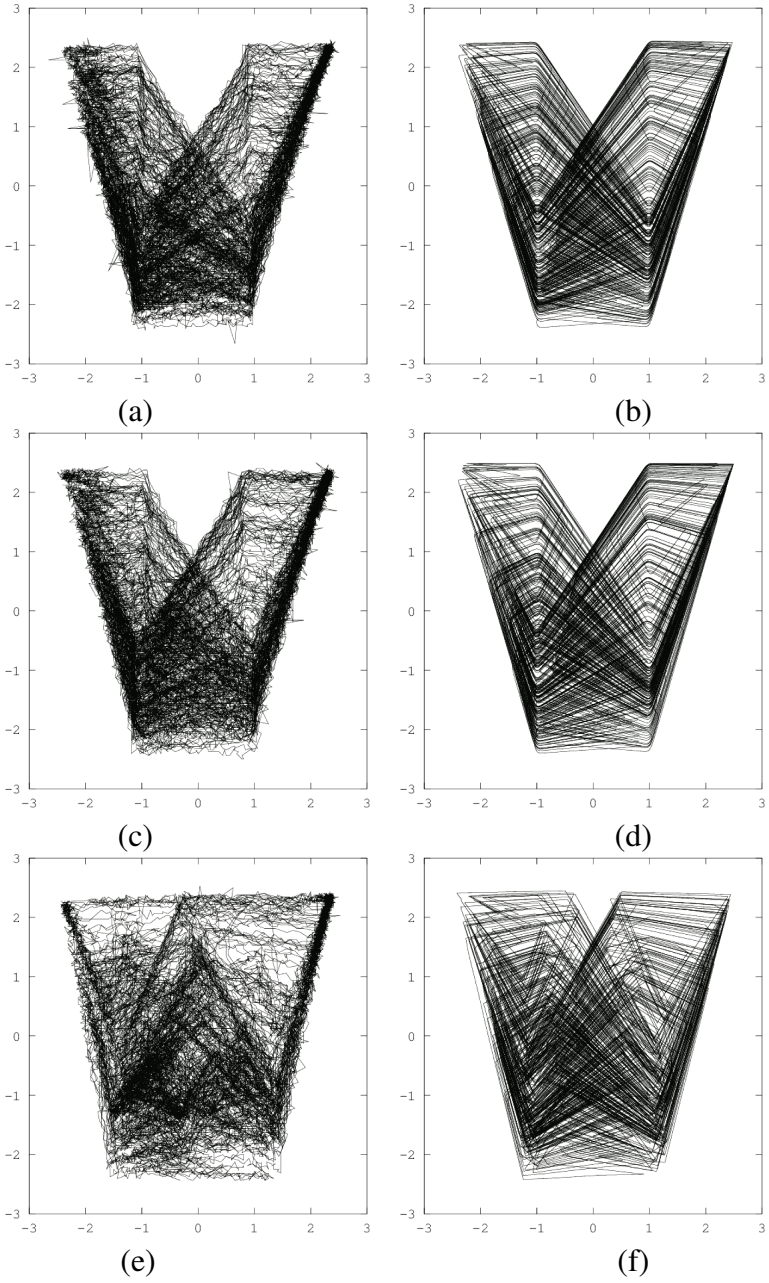


Figure 4.24 : $v_C(t_k-\tau)-v_C(t)$ phase portraits of circuit realization for $T_s = 0.001$ in a, $T_s = 0.020$ in c, $T_s = 0.250$ in e and of simulations for $T_s = 0.001$ in b, $T_s = 0.020$ in d, $T_s = 0.250$ in f. For all experiments $\alpha = 2.5$ and $\tau = 8$. This results demonstrate that the chaotic behavior of the system can be obtained with a small number of flip-flops down to 32 in conformity with Lyapunov exponent spectrum in Figure 2.17.

that functions as a sample-and-hold block. This delay line in [101] is swapped with a cascaded 152 pieces of 9-th order recursive ADD blocks. The schematic of the system with nested ADDs is given in Figure 4.25. At the bottom of each recursive ADD block, a chain of 12 inverting buffers in 3 slices using LUTs and XOR gates is used. Therefore, a continuous-time delay line with 555.6μs delay and 4.4μs minimum interval limitation (τ_{min}) is created and protected with binary low-pass filters which are located at the top of each recursive ADD block. Totally, $[6+(9\times 3)+3]\times 152 = 5472$ LUTs and $6\times 152 = 912$ XOR gates are used in the delay line. The circuit configuration (component values and parameters) are also found in Figure 4.25.

The delay line works properly and the same chaotic behavior in [101] is observed. Figure 4.26a shows the phase portrait of the original system with flip-flop chain and Figure 4.26b shows the phase portrait of the system with proposed ADD based delay line. [101] The random candidate bits are sampled at the end of the delay line with 50μs period similar to the one in [6]. Recorded 4Mb string is subjected to the NIST 800-22rev1a test suite and it is observed that the bit string has successfully passed all the tests.

More distinct advantage appears if ADD is implemented as an CMOS integrated circuit. ADD-BLPF block chain is the most area saving one among the three binary delay lines, flip-flop based, inverter based and ADD based ones. The circuit in [101] should have at least 25000 D-type flip-flops ($T_c = 20$ns for $\tau = 50\mu$s) in its delay line in order to success NIST's statistical test suite at 20 kbps throughput. The classical positive-edge triggered D-type flip-flop using three SR latches needs 26 transistors, however a very simple static D-type flip-flop without reset and preset feature is composed of 10 transistor [102]. Flip-flop based delay line, which is required for generating the chaotic attractor and the 20 kbps random bit generation, consumes $250,000$ transistor. If only even number of inverting buffers are utilized, $500\mu\text{s}/3.68\mu\text{s}\cdot 2048\cdot 2 = 556,521$ transistors are required. One should notice that, the CMOS-IC inverting buffer has quite less propagation delay than the FPGA's LUT. Hence, the implementation of buffer-chain delay line probably consumes much more transistors than $556,512$ for a 50μs delay. On the other hand, the similar attractor is observed using a cascaded 152 pieces of 9-th order recursive ADD blocks and their BLPFs. In this configuration, the total required transistor count is $92,720$. Using

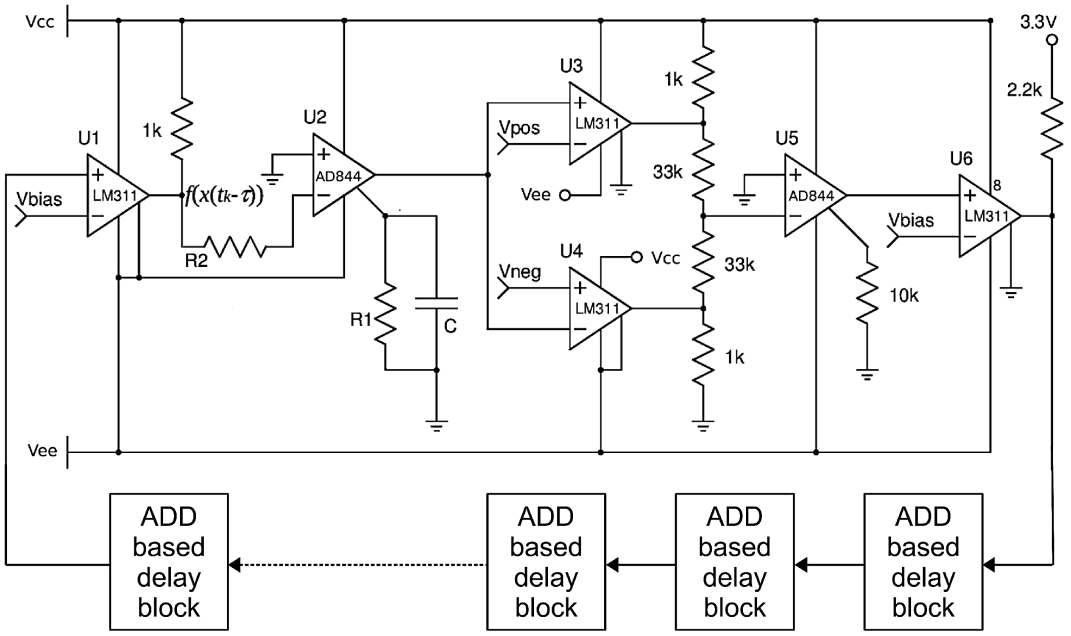


Figure 4.25 : Using ADDs on the delay line of a time-delay sampled-data system which was proposed as a True Random Bit Generator [6]. ADD and buffer chain based delay line converts the system from sampled-data to continuous-time. In this implementation, $V_{\mathrm{CC}} = 5V$, $V_{\mathrm{EE}} = -5V$, $V_{\mathrm{bias}} = 0.5V$, $V_{\mathrm{pos}} = 1.10V$, $V_{\mathrm{neg}} = -0.97V$, R1 $= 2.76\mathrm{k}\Omega$, R2 $= 5.73\mathrm{k}\Omega$, $\tau_{\mathrm{line}} = 555.6\mu$s, $\tau_{\mathrm{min}} = 4.4\mu$s, and the bit sampling rate is 20kHz.

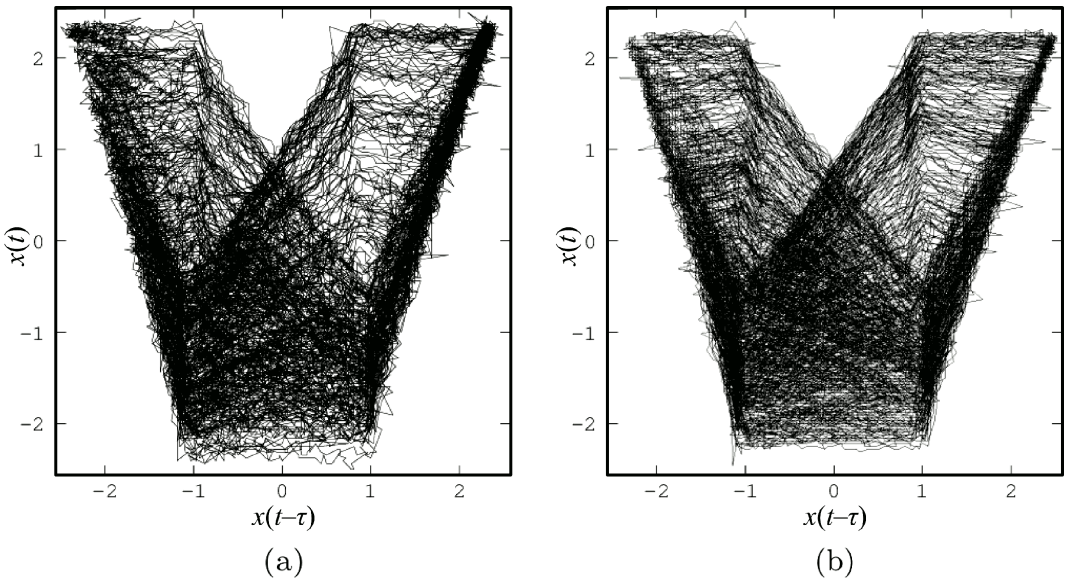


Figure 4.26 : Phase portraits of chaotic signals obtained a) by the original system with flip-flop chain, b) by the system with the proposed ADD based delay line.

NAND and NOT gates, BLPF can be implemented by 136 transistors, ADD by 50 transistors and the bottom delay line of each ADD block by 24 transistors. The proposed design need the least transistor in order to generated true random bits at 20 kbps rate by its expected chaotic dynamics. Proposed design fits into 37% of flip-flop based delay line area, and 17% of only buffer based delay line area.

On a target FPGA, the delay line constituted with recursive ADD blocks, small buffer chains at the base, and BLPFs at the top yield such a high amount of delay ($1.71ms$) that can not be produced practically using only buffers in the same FPGA. In recursive ADD structure, the delay increases exponentially in exchange for linearly increasing area consumption which provides outstanding area efficiency. By means of the asynchronous design, the ADD responds its input immediately without waiting for the clock edge, and relieves the system from being a periodic sampled-data system. This delay line can be used in RO-based PUF designs, low-frequency time-delay chaotic oscillators and low-frequency clock signal generators with the help of run-time controllable delay cells that the DLL designs already include. As expressed by Roska, the time delay of connections could play also a role when the processors of Cellular Wave Computer are connected either locally, within a receptive field, or in a sparse global, bus-like way [103].

4.3.4 Mono-scroll attractor using digital integrator and inverter chain

At the beginning, the sampled-data system $\dot{x}(t) = -x(t) + f(x(t_k - \tau)),\ t_k \leq t < t_k + T_s$ which is given in [101] has been discretized by forward Euler method with integration step h, and given by

$$x(t_{k+1}) = x(t_k) + h(-x(t_k) + f(x(t_k - \tau/h))). \tag{4.17}$$

Both systems have the same nonlinearity

$$f(x) = 4\,(u(x-1) + u(-x-1)) - 2. \tag{4.18}$$

According to the original system, $x(t) \in [-2,2]$. Here, amplitude has been quantized in System (4.17). $[-4,4)$ range has been quantized by 8-bit digital state using 2's complement representation (signed Q3.5 format). System is realized by the integrate and compare (INTCOMP) block whose schematic diagram is drawn in Figure 4.27.

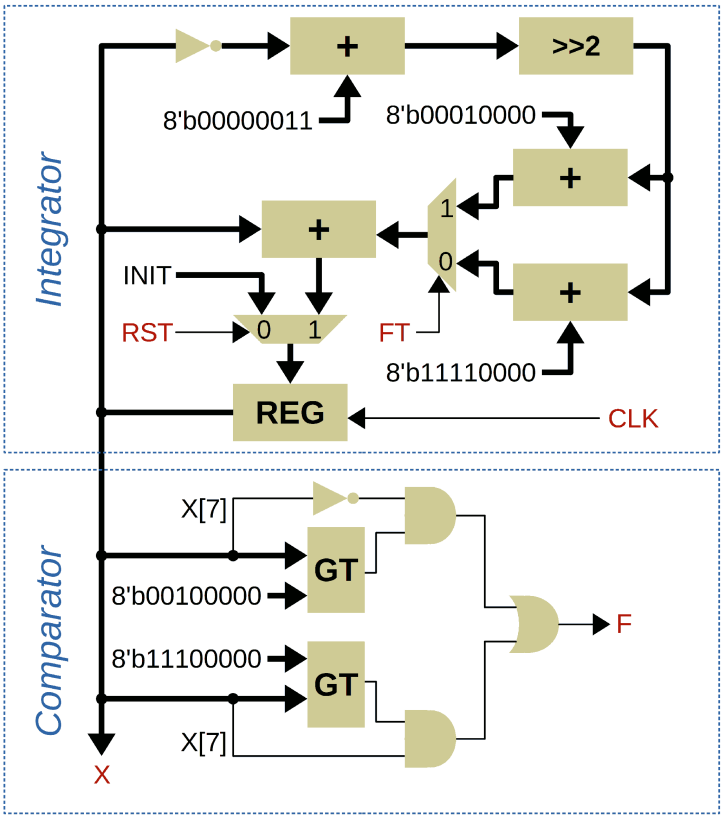


Figure 4.27 : Schematic diagram of integrate and compare block (INTCOMP).

INTCOMP has one 8-bit state register, REG, which can be reseted to INIT value according to RST signal. Current value of the state register is increased by either $h(-x(t_k)+2)$ or $h(-x(t_k)-2)$ according to the FT signal, which is the delayed nonlinear feedback. The CLK signal period is equal to h which is $1/4$ (normalized value) in this implementation, and multiplication by h is computed by shifting the operand to the right by 2 bits. F output is generated by the comparator block below the REG in the figure. Outside the INTCOMP, F is driven to the delay line.

Two delay lines are implemented. The first one (DDFF) has 36 D-type flipflops (DFF), depicted in Figure 4.28. System obtains $\tau = 36h = 9$ using DDFF. This structure has also been employed in the sampled-data system implementation [101]. Delay amount depends on the CLK signal period and jitter.

A system composed of an INCTOMP and a DDFF has 44-bit memory in total. At most in 2^{44} (the number of states in the state space) iterations, the system must repeat a state which indicates that system has a periodic motion. The solution proposed by this subsection utilizes propagation delay. Numbers of cascaded inverting buffers provide both the delayed signal and a delay uncertainty that breaks the periodic trajectory. The

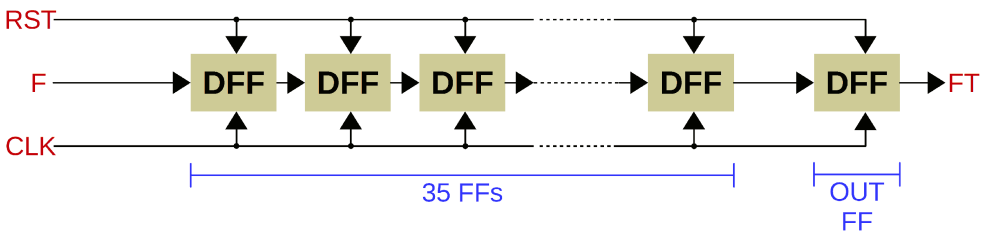


Figure 4.28 : Schematic diagram of D-type flipflop based delay line (DDFF).

alternative delay line with Look-up-tables (LUT) that function as non-inverting buffers is called DLUT and depicted in Figure 4.29.

Propagation delay is based on the low-pass characteristics of digital gates. In reality, switching takes time due to the parasitic capacitors appearing in CMOS circuits. The voltage exponentially increases or decreases at each gate output, and it takes some time to across the threshold voltage level of the next gate. This effect creates delay line on the LUT chain.

The RST signal resets the flipflops in Figure 4.28 and drives the LUTs to 0 in Figure 4.29. 1024 LUTs are employed to have a delay which is obtained by 35 DFFs in Figure 4.28. For the target Field-Programmable Gate Array (FPGA), the CLK period h is set to 8.197ns in order to equalize both delays. However, it is not possible to get a constant delay from DLUT. Although the mean delay may be approximated to the delay gained by DDFF, there exists a significant jitter on this line. This jitter represents the random process which is sourced by the noise on the FPGA. INTCOMP's chaotic basis has sensitivity to initial conditions. Hence, INTCOMP dramatically changes its trajectory when the delayed signal has a slight change with respect to the expected one.

System behavior is examined by a test circuit whose schematic diagram is given in Fig 4.30. INTCOMP1 is the main sub-block which runs with DLUT1 in order to observe the uncertainty effect. INTCOMP1 and DLUT1 generates $x_1(t_k)$ (X1 in the

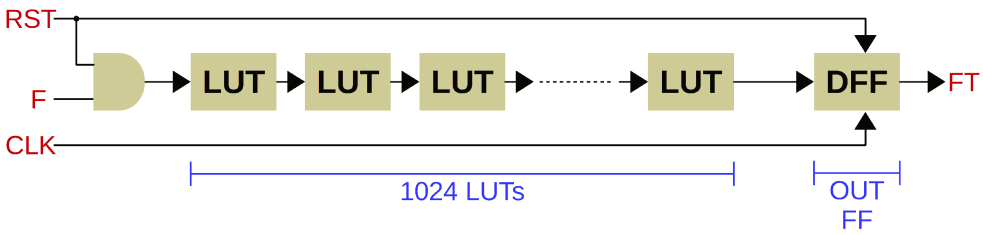


Figure 4.29 : Schematic diagram of Look-up-table based delay line (DLUT).

schematic). Evolution of X1 is compared with X2 ($x_2(t_k)$) which is the state variable of INTCOMP2 and DDFF2 couple. INTCOMP2 and DDFF2 couple tend to have a periodic motion, and this periodic motion will be the reference behavior of the system. DDFF0 in the diagram is required to adjust the CLK frequency. A square wave with a long period is generated by TSWG and driven to the DLUT1 and DDFF0 delay lines while the CLK period is adjusted using CLKGEN block. Output of the delay lines are stored in MEM (memory) block and sent to a computer through TX line to be analyzed. If delay of both lines are equalized, then proper CLK period has been determined. This process is required because DLUT is sensitive to PVT variations. However, one should notice that a stabilization scheme like delay-locked-loop [104] is not required, as the jitter occurred on DLUT line is vital for the intended aperiodic behavior. CU block is the control unit and has a user interface including LEDs, buttons, and switches. MEM does not only store the delay line outputs (FT signals), but also nonlinear function output of INTCOMP1 (or output of TSWG), and state variables of both INTCOMPs. Depth of MEM is 48k samples.

4.3.5 Multi-scroll attractor using analog integrator and flip-flop chain

The circuit of the proposed multi-scroll chaotic oscillator is implemented by frequently used analog components and a low-cost reconfigurable digital device. Although the *tanh* approximation is employed in numerical analysis, because the calculation of largest Lyapunov exponents requires it, the nonlinearity of circuit is composed by comparators. They are supposed to implement unit step functions. The system (2.16) is composed by a Current Feedback Operational Amplifier (CFOA) which realizes analog accumulation and integration, some analog comparators which emulate the unit step functions in the nonlinearity, a logical priority encoder, flip-flop chains and a logical decoder. Even it is assumed that the flip-flops run in discrete time in general, they hold their outputs along the sampling period obviously and they switch their values in an analog manner.

The system (2.16) shows that the evolution of x is controlled by itself and the nonlinear function of its delayed value. The difficulty in time-delay circuit implementation is to delay the analog state signal. At first view, the same difficulty is valid for the system (2.16). However, the fact that both the nonlinearity and the delay operation are time

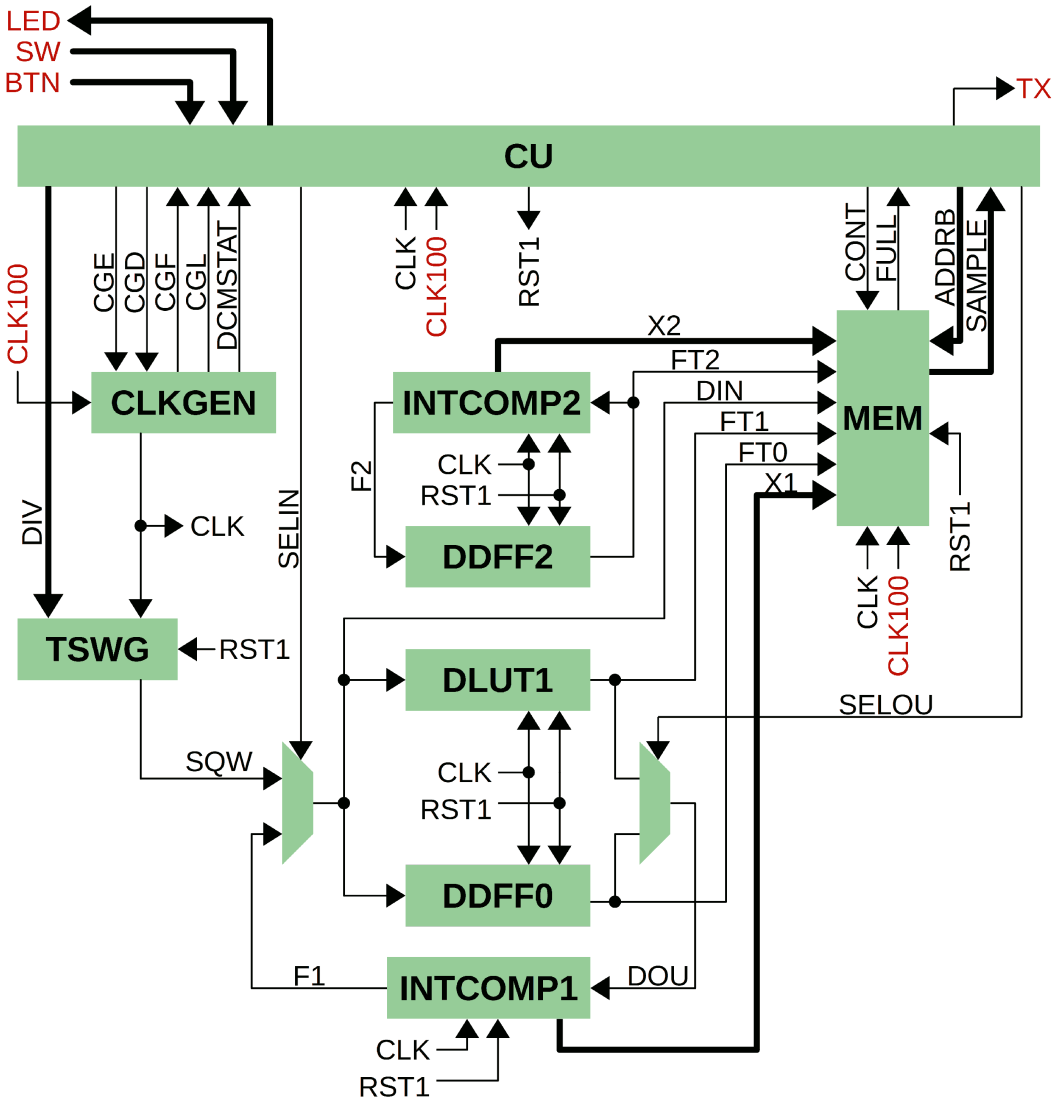


Figure 4.30 : Schematic diagram of top level design.

invariant provides a convenience. The nonlinear function and the delay operation may be swapped. If it is assumed that a delay operation $\mathrm{T}_\tau[\cdot]$ satisfies $\mathrm{T}_\tau[x(t)] = x(t-\tau)$, System (2.16) can be rewritten as

$$\dot{x}(t) = -x(t) + \alpha f_m(\mathrm{T}_\tau[x(t_k)]), \;\; t_k \leq t < t_k + \mathrm{T}_s. \tag{4.19}$$

By means of the time invariance, the equation takes the form

$$\dot{x}(t) = -x(t) + \alpha \mathrm{T}_\tau[f_m(x(t_k))], \;\; t_k \leq t < t_k + \mathrm{T}_s. \tag{4.20}$$

According to (4.20), the output of the nonlinear function is delayed instead of the analog state signal. The output of the nonlinearity (2.16) is discrete in amplitude. The number of different output levels of the nonlinearity is dependent to the n_f, w_f, h_f and α parameters. It should be noticed that this number has an upper limit $2n_f + 2$, which is also the number of regions on the $f_m(x)$.

A digital circuit having a D-type flip-flop chain is able to delay the output of a binary valued nonlinear functions, such as the one in [101]. The sample-and-hold behavior of the D-type flip-flop is proper for the system (4.20), as the sampled x signal is subjected to the nonlinear function in the model. In this implementation, D-type flip-flop chains are utilized with a slightly different duty. At any time, the sampled state $x(t_k - \tau)$ is in one of the regions which are separated by the discontinuity points on $f_m(x)$. The main idea is to delay the information where the sampled $x(t_k)$ is standing in. This information is represented by a digital signal which can be delayed using parallel flip-flop chains.

The circuit diagram is given in Figure 4.31. The $p_i(t)$ signals ($i = 1, 2, ..., 7$) are the comparison results of the state with the discontinuity points. They indicate the region that $x(t)$ stands in. As $x(t)$ can stand in only one region, the information that $p_i(t)$s carry can be encoded to $d_m(t)$ signals ($m = 0, 1, 2$). The number of comparators P, which is 7 in this case, should be equal or greater than $2n_f + 1$ to implement the desired nonlinearity. It can be considered as a P-bit asynchronous digital signal bus. The bit number of encoded signal bus M is $\lceil log_2(P) \rceil$. The comparator outputs are adjusted for the digital device. High level voltage value H is set to 3.3V and low level voltage L is 0V. The priority encoder which is a combinatorial logic block runs clockless and does not sample the $p_i(t)$ signals. But, the encoded $d_m(t)$ signals are sampled and held by flip-flop chains that yield $d_m(t_k - \tau)$ signals. The decoder is also a combinatorial logic

block, however its outputs are also sampled-data as inherited from its inputs. Only one of the $q_i(t)$ signals ($i = 0, 1, ..., 7$), which is indicating that $x(t_k - \tau)$ is in the i-th region, can be logically high at any time. The others remain logically low. So, $i_n(t)$ current in (4.21), which is the sum of currents flowing over the resistors R_i ($i = 0, 1, ..., 7$), only depends on the high bit on the q-bus and the resistor connected to that bit.

$$i_n(t) = \sum_{i=0}^{7} \frac{q_i(t)}{R_i} \tag{4.21}$$

CFOA provides $i_z = i_n$ and $i_z(t)$ is integrated on the *RC* block. The output of the CFOA is the state variable $x(t)$ which also equals to the $v_z(t)$.

The brief circuit introduction should be explained in details. In general case, a priority encoder ignores the values of inputs which are less significant than the most significant input with high level value. If the threshold voltages are adjusted in the order $V_i < V_{i+1}$ for $i = 1, 2, ..., 6$, then the priority encoder satisfies the truth table given in Table 4.5.

Table 4.5 : Truth table of priority encoder used in the circuit.

$p_7(t)$	$p_6(t)$	$p_5(t)$	$p_4(t)$	$p_3(t)$	$p_2(t)$	$p_1(t)$	$d_2(t)$	$d_1(t)$	$d_0(t)$
L	L	L	L	L	L	L	L	L	L
L	L	L	L	L	L	H	L	L	H
L	L	L	L	L	H	H	L	H	L
L	L	L	L	H	H	H	L	H	H
L	L	L	H	H	H	H	H	L	L
L	L	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	L
H	H	H	H	H	H	H	H	H	H

Using a field programmable gate array (FPGA) makes it easy to adjust the number of flip-flops N_{FF} on the chains. In this implementation $N_{\text{FF}} = 1000$ and the common clock signal for all flip-flops (clk) has a period $\text{T}_\text{s} = 200\text{ns}$. With the given sampling period, the delay τ obtained from each chain is $200\mu\text{s}$. There may be differences between $d_m(t - \tau)$ and $d_m(t_k - \tau)$ due to the sampling phenomenon explained in Figure 2.14. With the cooperation of a logical decoder at the output of flip-flop chains, a delay line coding mechanism is constructed. It should be noticed that linear growth in delay line number provides an exponential growth in scroll number with a cost of exponential growth in p-bus and q-bus widths. Since the required flip-flop chain count is less than

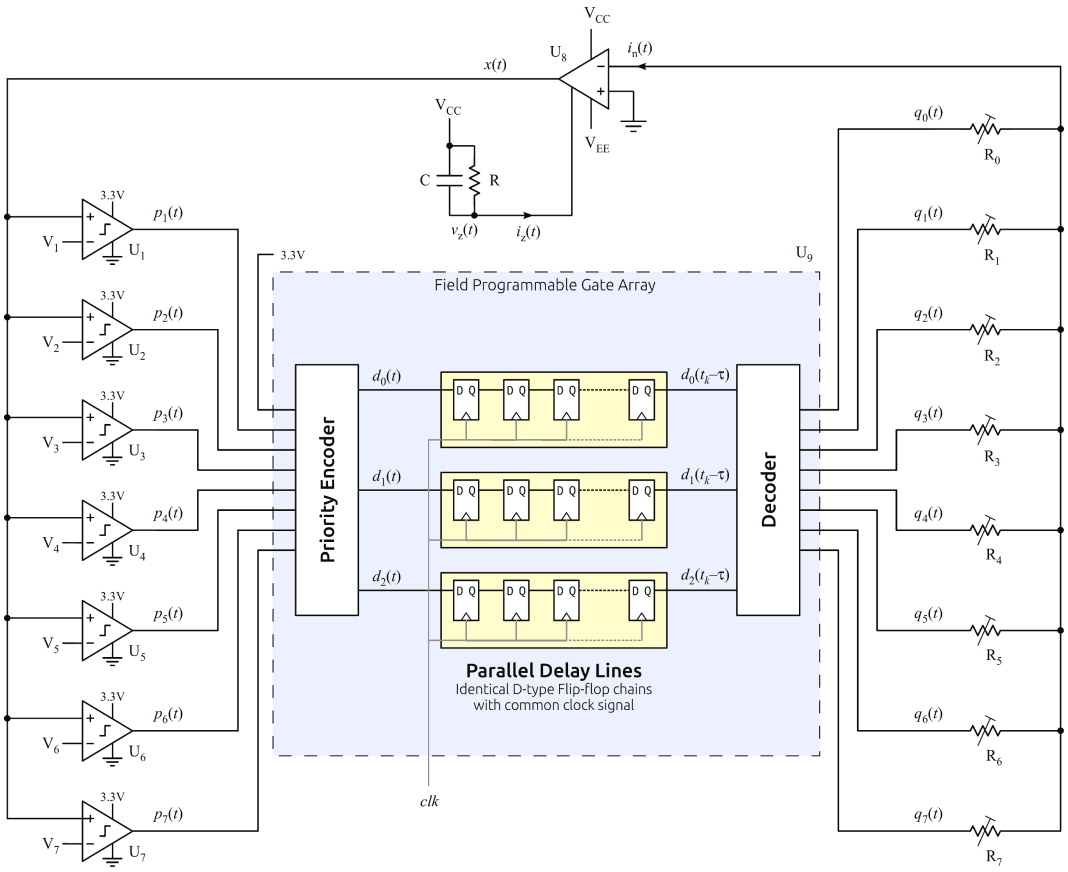


Figure 4.31 : Circuit diagram of multi-scroll time-delay sampled-data chaotic system. The Circuit has 7 comparators that determines the maximum number of scrolls as 6. With data coding approach, the delay line number is less that the comparator number and increases linearly when the comparator need increases exponentially for exponential growth in scroll number.

the comparator count, the proposed mechanism efficiently reduces the total component count when the scroll number is increased.

The truth table of the decoder is given in Table 4.6. In terms of comparator threshold voltages V_i, decoded signals $q_i(t)$ can be defined by

$$q_i(t) = \begin{cases} 3.3\text{V}, & V_i \leq x(t_k - \tau) < V_{i+1}, \\ 0\text{V}, & \text{else}, \end{cases} \tag{4.22}$$

Table 4.6 : Truth table of decoder used in the circuit.

$d_2(t_k-\tau)$	$d_1(t_k-\tau)$	$d_0(t_k-\tau)$	$q_7(t)$	$q_6(t)$	$q_5(t)$	$q_4(t)$	$q_3(t)$	$q_2(t)$	$q_1(t)$	$q_0(t)$
L	L	L	L	L	L	L	L	L	L	H
L	L	H	L	L	L	L	L	L	H	L
L	H	L	L	L	L	L	L	H	L	L
L	H	H	L	L	L	L	H	L	L	L
H	L	L	L	L	L	H	L	L	L	L
H	L	H	L	L	H	L	L	L	L	L
H	H	L	L	H	L	L	L	L	L	L
H	H	H	H	L	L	L	L	L	L	L

if it is assumed that $V_0 = -\infty$ and $V_8 = \infty$. The current into the z terminal of the CFOA is given by

$$i_z(t) = \frac{V_{\text{CC}} - v_z(t)}{R} - C\frac{dv_z(t)}{dt}. \tag{4.23}$$

It should be noticed that one of the terminals of the capacitor C is connected to V_{CC}. For CFOA, it is defined that $i_n(t) = i_z(t)$ which yields

$$\frac{dv_z(t)}{dt} = \frac{-v_z(t)}{RC} + \left[\frac{V_{\text{CC}}}{RC} - \frac{i_n(t)}{C}\right]. \tag{4.24}$$

Using (4.21) and (4.22), the state equation becomes

$$\begin{aligned} \frac{dv_z(t)}{dt} = & \frac{-v_z(t)}{RC} + \left[\frac{V_{\text{CC}}}{RC}\right. \\ & \left. - \sum_{i=0}^{7} \frac{3.3}{R_i C}\Big[u(x(t_k-\tau) - V_i) - u(x(t_k-\tau) - V_{i+1})\Big]\right]. \end{aligned}$$

The state variable $x(\hat{t})$ equals to $v_z(t)/RC$ with time normalization, $t = \hat{t}RC$. The proposed system does not require an amplitude normalization. Using (2.16) and setting

$o_f = 0$, the comparison between (2.16) and (4.25) reveals that

$$f_m(x) = \frac{1}{\alpha}\left[V_{\mathrm{CC}} - 3.3\sum_{i=0}^{7}\frac{R}{R_i}\Big[u(x-V_i) - u(x-V_{i+1})\Big]\right]$$
$$= \left(h_f - \frac{n_f w_f}{\alpha}\right) + \sum_{i=0}^{2n_f}\Bigg[\left(\frac{w_f}{\alpha} + (-1)^{i+1}\left(2h_f + \frac{w_f}{\alpha}\right)\right) \cdot u\big(x + w_f\left(n_f - i\right)\big)\Bigg].$$

The value of R is fixed to 10kΩ and the capacitor C to 1nF. For a chosen $\{\alpha, n_f, h_f, w_f\}$ set, R_is and V_i are calculated to match the numerical analysis and circuit implementation results below.

In order to adjust the circuit parameters, the simulation parameters are selected as follows: $\alpha = 1$, $n_f = 3$, $h_f = 1.5$, $w_f = 0.5$, $o_f = 0$. As the result, the comparator threshold voltages are calculated as $V_1 = -1.5$V, $V_2 = -1.0$V, $V_3 = -0.5$V, $V_4 = 0.0$V, $V_5 = 0.5$V, $V_6 = 1.0$V, $V_7 = 1.5$V. And the values of adjustable resistors are calculated as $R_0 = 6600\Omega$, $R_1 = 4125\Omega$, $R_2 = 8250\Omega$, $R_3 = 4714\Omega$, $R_4 = 11000\Omega$, $R_5 = 5500\Omega$, $R_6 = 16500\Omega$, $R_7 = 6600\Omega$. When a ramp signal is applied to the input of the comparators on the circuit which has the values given above and in which the connection between the CFOA and the comparators is cut, the $f_m(x)$ function is observed on the output of the CFOA. In this way, the $f_m(x)$ from the circuit is captured by an oscilloscope and depicted in Figure 4.32a.

The circuit which implements the nonlinear function in Figure 4.32a generates the phase portrait in Figure 4.32b with $\tau = 200\mu$s, $N_{\mathrm{FF}} = 1000$, $\mathrm{T}_s = 200$ns, $RC = 10\mu$s. The normalized delay, $\tau/RC = 20$, is equal to delay in numeric simulations in Figure 2.20 to 2.25. A 6-scroll chaotic attractor in Figure 4.32b from the circuit matches up with the simulation result in Figure 2.19. 2-scroll and 4-scroll ones from the circuit (Figure 4.32c and Figure 4.32d) are in agreement with the given simulation results in Figure 2.26a and Figure 2.26b. Circuit realization highly matches up with numerical analysis. However, one should notice that signals are influenced by random noise generated by electrical circuit, thus the signal has a fluctuating trajectory as seen on Figure 4.32b−d.

In this subsection, a chaotic time-delay sampled-data system which is generating multi-scroll attractor is proposed. The system is modelled by a first-order delay differential equation which consists of a nonlinear time-delay sampled-data feedback

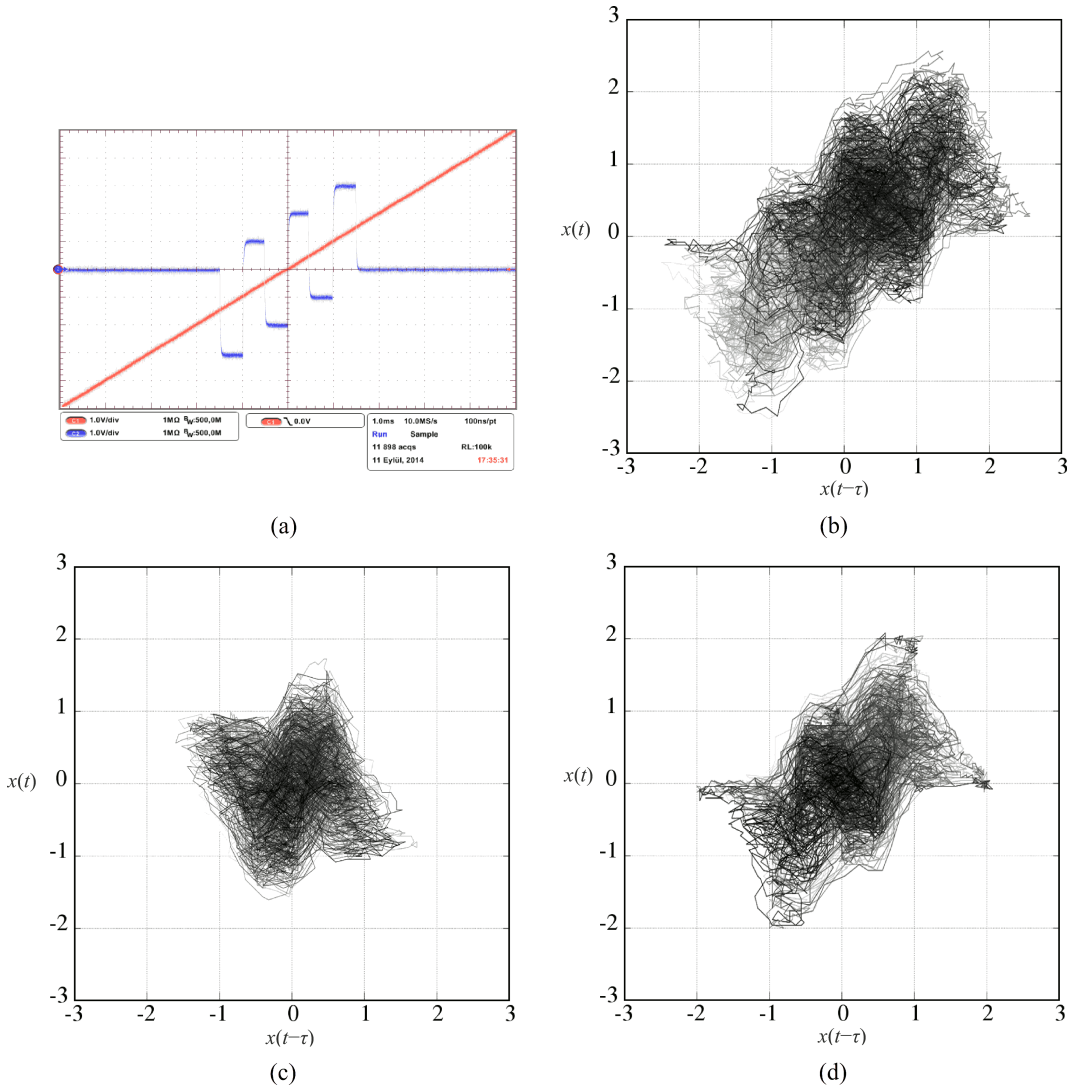


Figure 4.32 : Results from the circuit implementation: a) The implemented nonlinear function $y = f_m(x)$ (blue), and $y = x/\alpha$ line (red). Nonlinear function has 7 discontinuities which yields 6 scrolls. b−d) Phase portraits in the $x(t-\tau)$ versus $x(t)$ state space generated by the circuit implementation with common parameters: $\alpha = 1$, $h_f = 1.5$, $w_f = 0.5$, $o_f = 0$, $\tau = 200\mu$s, $N_{\text{FF}} = 1000$, $\text{T}_s = 200$ns, $RC = 10\mu$s, $V_{supp} = \pm 3.3$V, $n_f = 3$ in b, $n_f = 1$ in c, $n_f = 2$ in d.

function. The nonlinearity is a discrete amplitude function and formed by unit step functions. Number of scrolls can be increased by inserting additional step to the nonlinearity. There is no limit in the number of scroll, theoretically, but in practice, the operation range of the circuit puts realization limits. The 2, 4, 6, 8 and 10-scroll simulation results and circuit implementation results of the first three of them are demonstrated.

This sampled-data system removes the need for analog delay lines. Instead, delay lines composed of simple D-type flip flops are used in the circuit implementation. The nonlinear function with finite discrete amplitudes results in finite numbers of delay lines. Furthermore, a binary coding method is applied to the feedback channel and the number of delay lines are significantly decreased. As a result, the number of scrolls increases exponentially while the number of delay lines increases linearly. Both the nonlinear feedback function and its binary coded sampled-data delay lines are novel contributions to the literature.

4.3.6 1D network using analog integrator and flip-flop chain

In [6], the considered time-delay chaotic system is employed to generate random numbers. Actually, it has a binary output, which can directly be used for random number generation. If one can couple to the signal of this original system and apply it to another appropriately constructed system, it is possible to predict the numbers that will be generated by the original system τ second before. This is the consequence of the fact that time-delayed response system can be synchronized to non-delayed drive system by making use of anticipating synchronization. It is possible to increase number of coupled systems, therefore with a chain of oscillators prediction can be accomplished for integer multiples of τ, i.e. 2τ, 3τ, etc. [76]. This subsection, presents experimental verification of the idea.

The circuit realization of the original time-delay chaotic system is given in [101] and [6]. Figure 4.33 shows the circuit implementation of four coupled systems in such a way that they exhibit anticipating synchronization. The given implementation realizes

the sampled-data form of the systems as follows

$$\begin{array}{rcl} \dot{x_1}(t) & = & -x_1(t)+\alpha f(x_1(t_k-\tau)) \\ \dot{x_2}(t) & = & -x_2(t)+\alpha f(x_1(t_k)) \\ \dot{x_3}(t) & = & -x_3(t)+\alpha f(x_2(t_k)) \\ \dot{x_4}(t) & = & -x_4(t)+\alpha f(x_3(t_k)) \end{array} \tag{4.25}$$

where t_k is the k-th sampling-time [101]. The delay line shown in Figure 4.33 is a flip-flop chain which delays $f(x_1(t_k))$ by sampling and shifting the binary signal. Using single flip-flop in each system and applying the same clock signal to all makes the nonlinear feedback signal generated by $f(\cdot)$ sampled-data. Again the error system defined with

$$e \triangleq x_{\mathrm{n}-1}(t)-x_{\mathrm{n}}(t-\tau) \tag{4.26}$$

is a stable system and anticipating synchronization occurs. Considering the first two coupled time-delay sampled-data systems, which are given in (4.25), the error evolution is obtained as

$$\dot{x_1}(t)-\dot{x_2}(t-\tau) = -[x_1(t)-x_2(t-\tau)]+\alpha\{f(x_1(t_k-\tau))-f(x_1(t_k-\tau))\}. \tag{4.27}$$

This shows that (3.14) is also valid for sampled-data form of the system.

The circuit in Figure 4.33 is easily built using on-the-shelf analog components and a very low-cost Field Programmable Gate Array (FPGA) Board. Each system has four LM311 comparators for the nonlinearity and two AD844 current-feedback OpAmps, one for the integration and the other for the nonlinearity implementation. There exist four potentiometers. Two of them are used for reference voltages in nonlinearity implementation, and remaining two are for the α parameter and the time constant of the circuit. The component values are the same as given in [6]. The delay line and single flip-flops (sample-and-hold units) are implemented on the FPGA. Even this low-featured FPGA is quite enough for the required flip-flop chain which provides 500μs delay with a 50MHz sampling rate.

In Figure 4.34, some qualitative measurements made on the implementation in Figure 4.33 are given. Figure 4.34a, 4.34b and 4.34c are oscilloscope screen photos. Figure 4.34a plots $x_1(t)$ vs. $x_2(t)$, Figure 4.34b plots $x_2(t)$ vs. $x_3(t)$, and Figure 4.34c plots $x_1(t)$ vs. $x_3(t)$. The attractor observed on $x_1(t)$ vs. $x_3(t)$ state-space, which is a new attractor, is compared with its computer simulation results which is depicted in Figure 4.34d. The realization results for this new attractor is in accord with the computer

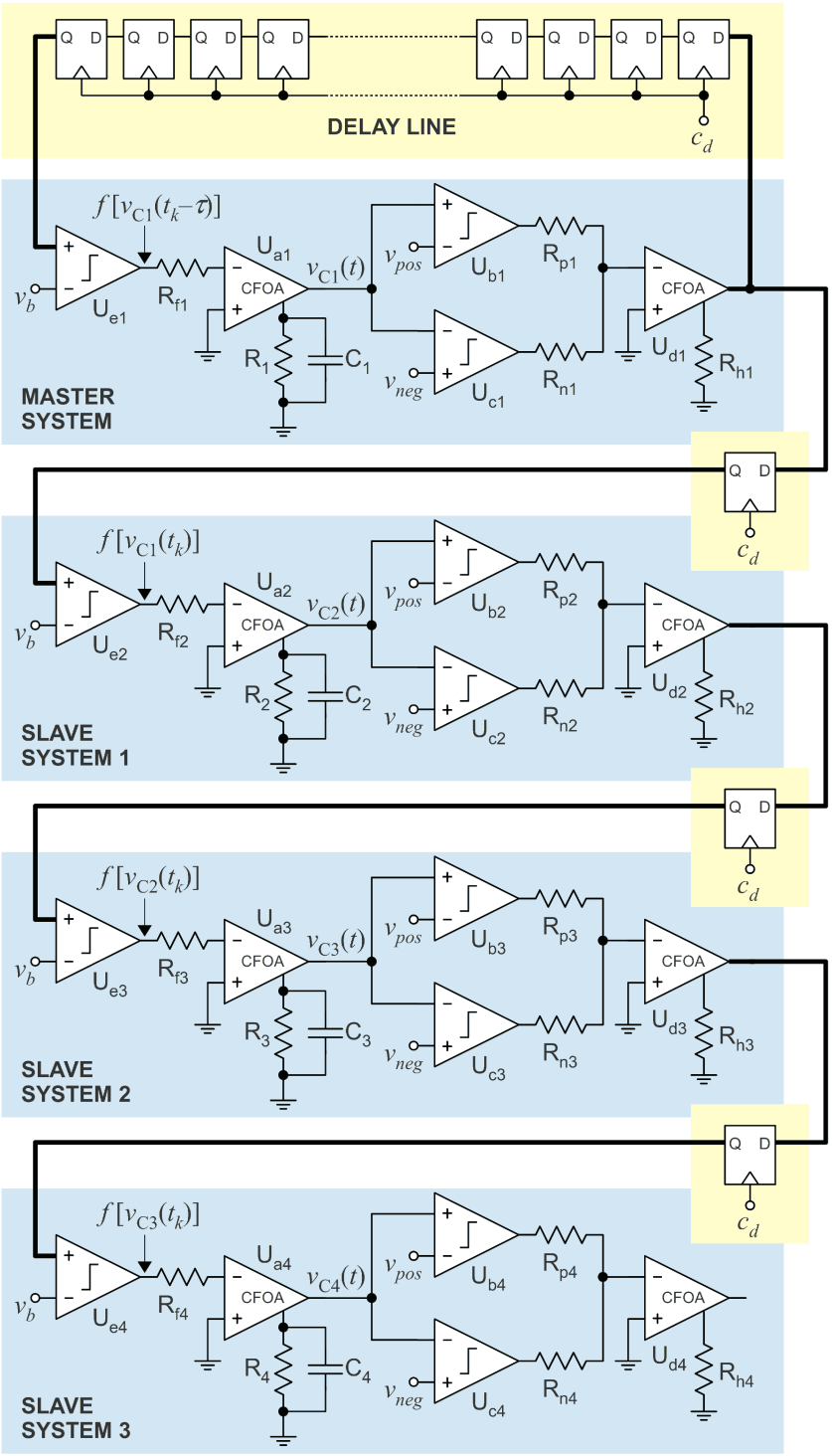


Figure 4.33 : Four sampled-data feedback systems are used for the realization of the idea. The Master System has the time-delay feedback. Slave System 1 is coupled to the Master System in a non-delayed form. Just one D flip-flop is used to make the drive signal sampled-data which is synchronous to the delay line. Slave System 2 and Slave System 3 are implemented in the same manner. As a result, Slave System 3 predicts (anticipates) the state of the Master System 3τ before.

simulation. Figure 4.35 depicts experimental waveforms of $x_1(t) = v_{C1}(t)$, $x_2(t) = v_{C2}(t)$, $x_3(t) = v_{C3}(t)$, and $x_4(t) = v_{C4}(t)$ on the oscilloscope screen for $\tau = 500\mu$s, showing the anticipating synchronization for integer multiples of τ. It is seen from the Figure 4.35 that anticipating chaotic synchronization is maintained for τ, 2τ, and 3τ such that $x_1(t) = x_2(t-\tau) = x_3(t-2\tau) = x_4(t-3\tau)$ in ideal conditions. This can be seen from the highlighted parts of the waveforms given in Figure 4.35 with some effect of the electrical noise exposed in the implementation.

4.3.7 1D network using digital integrator and flip-flop chain

In Subsection 4.3.6, one master and three slave systems have been implemented and anticipated signals have been demonstrated. In this subsection, the slave systems are the ones discretized using forward Euler method and implemented as digital circuits.

The delayed signal is expressed by a linear time invariant delay operator, $x(t_k - \tau) = \mathrm{T}_\tau[x(t_k)]$. Then changing the order of the nonlinearity $f(\cdot)$ and the time operator $\mathrm{T}_\tau[\cdot]$ yields $f(x(t_k-\tau)) = \mathrm{T}_\tau[f(x(t_k))]$. This provides the binary output of the nonlinearity to be delayed, instead of the analog state variable itself. The required delay line for Oscillator 4 in (2.15) samples the binary signal from the nonlinearity and hold it along the sampling period. The previous samples in the delay line moves one step all together towards output at each sampling moment. This structure is simply constructed using fundamental memory element for the digital circuits, delay-type (D-type) flipflop, and called *shift register* in digital electronic field. Figure 4.11b depicts the D-type flipflop chain used in this implementation as the delay line.

In this implementation, p is selected as 3. Integration step of discrete systems h is selected as T_s, which is 0.125 in the model. Delay (τ) is 10. The discrete-time slave systems with the sampled-data master system is given in (4.28) with the coupling scheme.

$$\begin{array}{ll} \dot{x_0}(t) & = -x_0(t) + \mathrm{T}_\tau\Big[f\big(x_0(t_k)\big)\Big] \\ x_1(t_k+T_s) & = (1-T_s)x_1(t_k) + T_s f(x_0(t_k)) \\ x_2(t_k+T_s) & = (1-T_s)x_2(t_k) + T_s f(x_1(t_k)) \\ x_3(t_k+T_s) & = (1-T_s)x_3(t_k) + T_s f(x_2(t_k)) \end{array} \tag{4.28}$$

If the feedback in the Oscillator 4 is broken and f is considered as an input, the solution of sampled-data system becomes

$$x(t_k+T_s) = e^{-T_s}x(t_k) + (1-e^{-T_s})f(t_k). \tag{4.29}$$

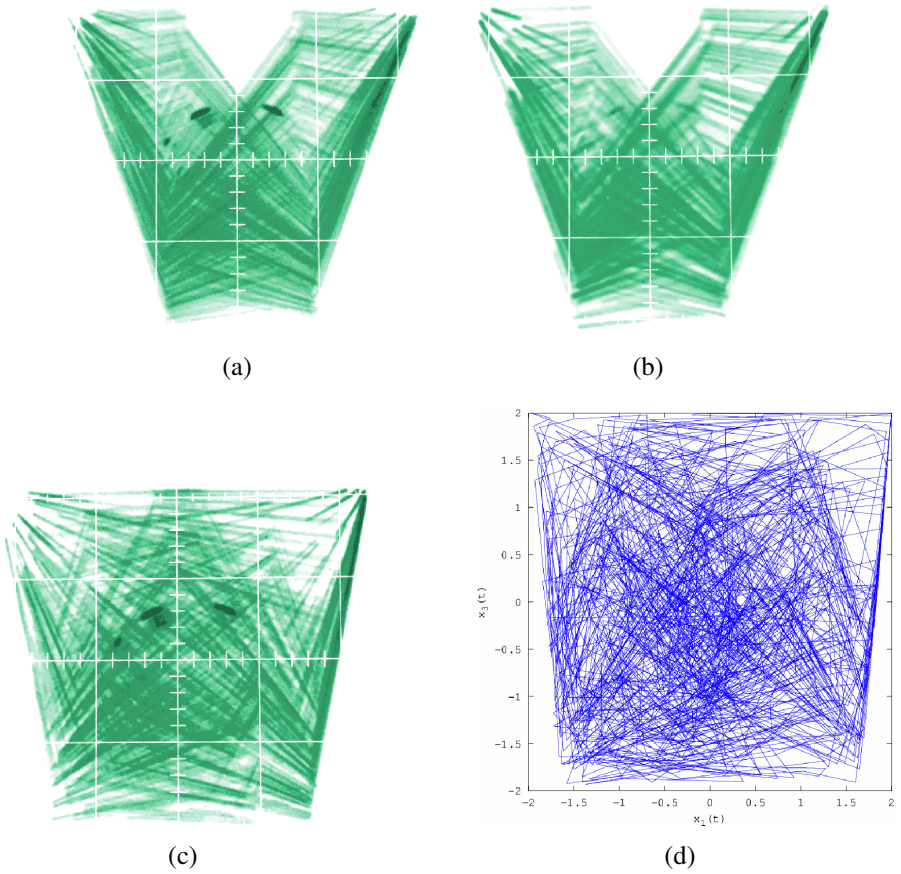


Figure 4.34 : a) $x_1(t)$ vs. $x_2(t)$ plot, b) $x_2(t)$ vs. $x_3(t)$ plot, c) $x_1(t)$ vs. $x_3(t)$ captured on an analog oscilloscope screen, d) $x_1(t)$ vs. $x_3(t)$ plot drawn by computer simulation.

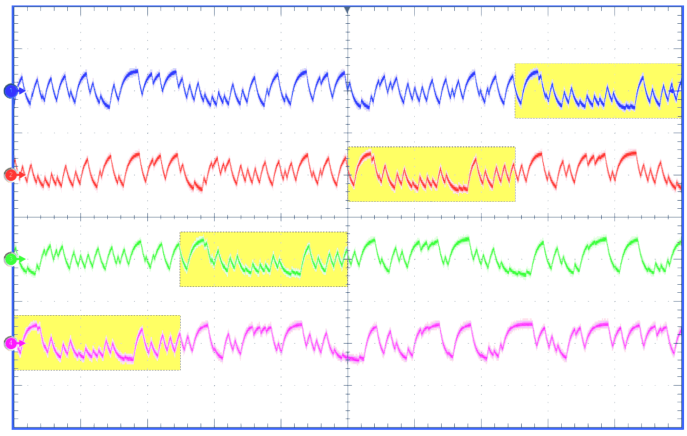


Figure 4.35 : Digital oscilloscope screen captures depicting $x_1(t)$ (blue), $x_2(t)$ (red), $x_3(t)$ (green) and $x_4(t)$ (purple) synchronously. The yellow box is shifted τ second left at each channel and used for focusing a short-time signal record that is seen on every system's output.

Using $\alpha = e^{-T_s}$, the general solution is given by

$$x(nT_s) = \alpha^n x(0) + \sum_{r=0}^{n-1} \alpha^{n-1-r}(1-\alpha) f(t_r). \tag{4.30}$$

Similarly, the forward Euler integration provides the solution of the discretized system by

$$x_{\mathrm{D}}(nT_s) = \beta^n x_{\mathrm{D}}(0) + \sum_{r=0}^{n-1} \beta^{n-1-r}(1-\beta) f(t_r), \tag{4.31}$$

where $\beta = (1-h)^{T_s/h}$. As $\alpha < 1$ and $\beta < 1$, the zero-input solution of both systems asymptotically converges to 0. If the error is defined as $e(nT_s) = x_{\mathrm{D}}(nT_s) - x(nT_s)$, and the input applied $f(t_r)$ is ± 2 to both systems, $0 < e(nT_s) \leq e_{\max} = 0.097082$, when $h = T_s = 1/8$. Input signal is defined ± 2 in order to simulate the nonlinear function in (2.8), as $\alpha = 2$ in Subsection 2.3.1. Selecting $h = T_s = 1/8$ gives a very rough digital approximation.

The discrete-time (digital) system resembles the original sampled-data system in the error range given above. Simulations show that digital system's state variable may incorrectly pass the thresholds in the f function due to the error, and generate faulty f values for finite sampling periods. This reduces the anticipating synchronization performance of the digital systems while p is increasing. However, for $p = 3$ as in this implementation, the 3-rd slave signal successfully predicts the master systems state signal. Synchronization performance may be polished by smaller h values in exchange for implementation complexity. The preferred feature is the simplicity of slave system, thus the system is implemented as explained below with $h = 1/8$.

The System (4.28) is implemented by a mixed signal circuit whose block diagram is drawn in Figure 4.36. The red blocks are analog (continuous time, continuous amplitude signal) circuits. The green blocks in the Field Programmable Gate Array (FPGA) region are digital (discrete-time, quantized amplitude signal) circuits. FPGA is a configurable digital device, that you can build the designed digital circuit using its functional blocks and programmable interconnections. One should note that, the discrete-time systems hold their signal value between the sampling moments. Therefore, the digital Delay Line acts the sample and hold role for the analog S_0 system, which is integrating the Delay Line output signal in continuous time. The time constant of analog S_0 system, which is used as the time normalization factor, is 10μs. The real delay is 100μs that means $\tau = 10$ in the implementation. T_s equals to 1250ns

(equal to $1/8$ before time normalization), so the flipflop count on the delay line is 80. Digital S_1, S_2, and S_3 circuits iterate at the beginning of each T_s period, and generate the signals from $f(x_1(t_k))$ to $f(x_3(t_k))$. Their states ($x_1(t_k)$ to $x_3(t_k)$) are not observable in the proposed implementation. In order to observe these states, they are re-generated outside the digital device (FPGA) using the drive signals ($f(x_0(t_k))$ to $f(x_2(t_k))$) by analog integrators. The analog integrators are simple serial RC circuits. The voltage signal over the capacitors of these RC couples represent the internal (non-observable) x_1, x_2, and x_3 states.

Figure 4.37 have four phase portraits which are generated by the given implementation. The original attractor in [101] that exist on the $x(t-\tau)$-$x(t)$ state space has the shape like the letter 'V'. The same attractor is observed on $x_0(t)$-$x_1(t)$ plane (Figure 4.37(a)), on $x_1(t)$-$x_2(t)$ plane (Figure 4.37(b))on $x_2(t)$-$x_3(t)$ plane (Figure 4.37(c)), and on $x_3(t)$-$x_4(t)$ plane (Figure 4.37(d)). Recorded signals over time using an analog oscilloscope have been plotted in Figure 4.38. From top to bottom, each signal is anticipated by the one below, 100μs before. The time shift between the signals equals to the delay line length (100μs).

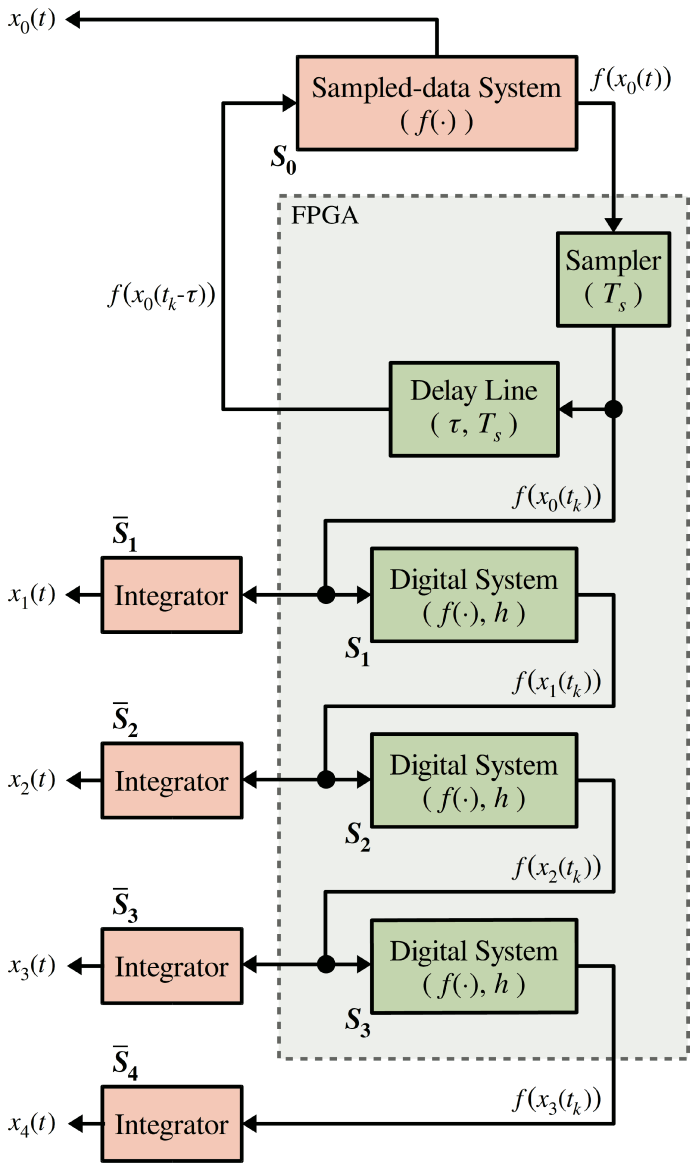
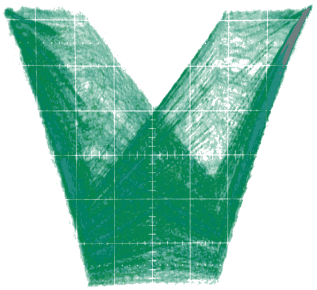
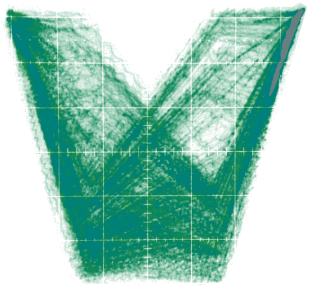


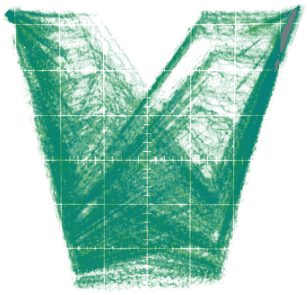
Figure 4.36 : Block diagram of coupled systems for anticipating synchronization. S_1, S_2, and S_3 are digital slave systems whose states are hidden. In order to measure the hidden states, integrators ($\overline{S_1}$, $\overline{S_2}$, and $\overline{S_3}$) are employed. They work like sampled data system, as they continuously integrate the sampled and held binary signal.



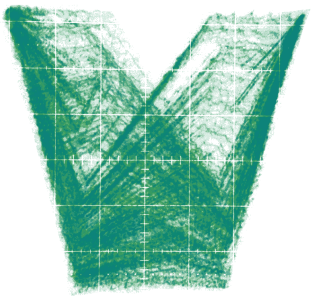
(a) Chaotic attractor on $x_0(t)$-$x_1(t)$ plane.



(b) Chaotic attractor on $x_1(t)$-$x_2(t)$ plane.



(c) Chaotic attractor on $x_2(t)$-$x_3(t)$ plane.



(d) Chaotic attractor on $x_3(t)$-$x_4(t)$ plane.

Figure 4.37 : The chaotic attractors observed by an analog oscilloscope. Phase planes used for observation have been noted below subfigures.

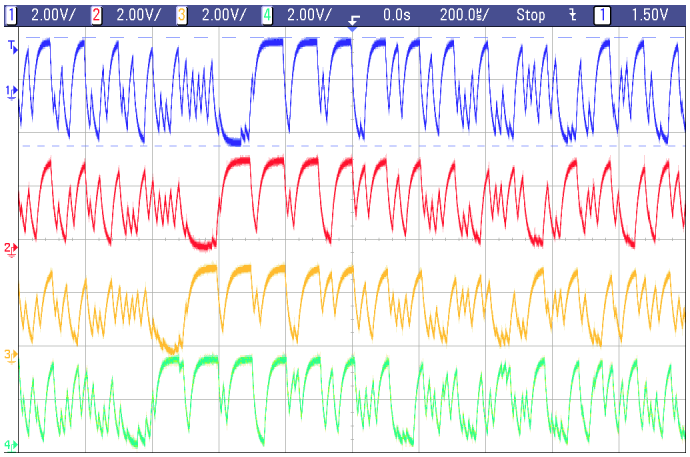


Figure 4.38 : From top to bottom, $x_0(t)$ (blue), $x_1(t)$ (red), $x_2(t)$ (yellow), $x_3(t)$ (green) in a 2ms-long record. Each have 100μs left-shift in reference to the one up signal.

5. APPLICATIONS

This chapter covers application examples of the cells, the networks, and their implementations given in Chapter 2−4 to the real world problems.

Feedback Motion Planning, which is introduced in Section 5.1, has novel contributions which are proposed in the same section. Doppler Effect observed on Cellular Nonlinear Networks, based on both relaxation-oscillators and logic oscillator, comes in useful when predicting the future location of target in 2D motion planning problem. The details of employing the effect to extract a new feature from the network has been presented in Subsection 5.1.2. Before, the classical feedback motion planning methods employing Network 1 (Subsection 3.1.1) has been given in Subsection 5.1.1. Two methods, called Wavefront Diffusion Based Algorithm and Wave Accumulation and Gradient Based Algorithm, are included. Functional properties of two methods are unified and a reference classical method is built in order to compare with the predictive method in Subsection 5.1.2. Test simulations prove that the prediction based on Doppler Effect really satisfies the expectation.

Applications of CNNs are followed by True Random Bit Generators (TRBGs). TRBGs are introduced in Section 5.2. The previously proposed circuit of time-delay sampled-data system and the circuits derived from it have been tested as TRBG. Original circuit with analog integrator/comparator and flip-flop base delay line performs good statistical results at low throughput, which is enhanced by EXOR operation. The circuit with digital integrator/comparator and inverter based delay line is exhibited in aperiodic behavior, which acts a key-role for generating random bits using only an FPGA. The original circuit, its enhanced configuration, and aperiodic digital implementation are covered by Subsection 5.2.1.

Moreover, the network given in Subsection 4.3.6 is employed in an attack to the proposed TRBGs. Especially, attack on the TRBG utilizing analog integrator/comparator has been investigated in Subsection 5.2.2. A correlation between

number of anticipating slave systems on the network and the anticipated bit error rate has also been reported.

5.1 Feedback Motion Planning

Motion planning, which has applications from robotics, biochemistry [105], video animations, artificial intelligence [106], to autonomous vehicle navigation [107], is the name of producing a plan that moves object form an initial configuration to a desired configuration while obeying the movement constraint [108]. A state (or configuration) space, either discrete or continuous, which includes all possible values of the variables, such as positions, orientations, velocities of objects like robots, targets, obstacles, is the first fundamental requirement of motion planning. Time is the following requirement. Implicit time definition (having just the order of event sequence) or explicit time definition (having quantities as functions of time) should be provided. State transforming operators, which are called actions, are also required in order to compose a plan to evolve an initial state to the desired one. Henceforth, two criteria are considered to evaluate any algorithm if it is a motion planner. The first one, feasibility, is the indication of success to arrive at the goal state, without efficiency consideration. The second is optimality, which is the indication of a feasible plan which optimizes performance in a clearly specified way. In general, the effort for proposing a feasible solution is more than for an optimal solution in robotics and related fields [108].

In today's applications, sampling-based motion planning, one of the two main approaches in continuous state spaces, is much more referred than combinatorial motion planning, due to its short running times and implementation simplicity [108]. Both methods need geometric modeling of application's world, and associated geometric transformations. On the other hand, without geometric representation, the simplest planning algorithms on discrete state space lie at the base of many complex methods or inspire them. Simplicity arises from not only the lack of geometric representations, but also the lack of support for differential equations and uncertainty. To build such a motion planner, every unique situation of the world is mapped to a discrete state. The set constituted by those discrete states are called the state space. A state transition function is created, whose inputs are the current state and the action, and output is the next state. A search algorithm which is capable of recording the

state transitions is proposed such that the result is a sequence of actions that draws a feasible plan [108]. For an optimal solution, a cost function is defined for actions. Then, algorithm is enhanced in order to seek for the minimum cost of plan, for example Dijkstra's algorithm and A* algorithm [109].

The continuous paths generated by both sampling-based and combinatorial motion planning require a feedback controller in real world applications, because errors and deviations are taken into account. Discrete space motion planning steps forward when embedding such a feedback control to the core of the planner. That planner is called feedback motion planner and produces a feedback plan that involves feasible paths avoiding obstacles by giving an action for every single state. Therefore, any unpredictable deviation in the state of the real world that the planner interacts can be healed by the feedback plan [108]. Potential function Φ, which is a function from the discrete state space to $[0,\infty]$ can be called a navigation function if it satisfies three conditions as follows. 1) $\Phi(x) = 0$ for all x in the goal states set, 2) $\Phi(x) = \infty$ if and only if no point in the goal set is reachable from x, 3) and the local operator gives a next state whose potential is less than the current state for every state excluded the goal set. The local operator may be a minimization operator like the negative gradient operator in continuous space. Navigation function defines a feedback plan if the action is determined by the local operator [108].

Special Cellular Nonlinear Networks (CNNs) serve as feedback motion planners, in the case of $\mathbb{R}^2$ state space, where the states represent the discrete positions on a 2D Euclidean plane, not the velocity or acceleration, and the only action defined on this state space is 2D translation of a point object. CNN promises a computation style beyond Boolean logic [103], which is handled by wave computers [110], while algorithmic researches develop algorithms that runs sequentially on Boolean processor or runs in parallel on reconfigurable logic [111]. Wave-front propagation algorithm, maximum clearance algorithm, and Dial's algorithm in literature yield optimal feedback plans [108]. This phenomenon is also observed in propagation of nonlinear waves, which is also called active waves and spatio-temporal waves. Three different types of wave propagating nonlinear grid network, which are also called active media, are defined as follows. The first one is excitable networks whose elements (cells) have one stable equilibrium point. Any excitements from outside

and from the coupled cells bring the cell out of stable equilibrium, then the cell evolves back to the initial stable state, while the excitation wave is propagating on the network. The second one is bistable networks whose cells have two stable equilibrium points. Excitement brings the cell out of a stable state and evolves to the other stable state. The product wave is called traveling wave or switching wave. The last one is self-oscillatory networks with cells without any stable equilibrium point. The cells typically have a limit cycle in phase portraits causing a periodic oscillation. In this type networks, cells usually synchronize to each other with a proper coupling scheme, and a spatio-temporal event, called autowave, propagates on the network. CNN can properly represent those system and propagate active waves [112]. Systems capable of propagating binary traveling waves (triggering waves) are also suitable for feedback motion planning [67].

The fact that wave propagation directly generates the feedback plan or the navigation function is not stated in many CNN based motion planning studies. CNN is a functional tool for planning. An early work has been published in 1993 that declares a two dimensional grid array of Chua's circuit is capable of finding optimal path which needs the least energy even the ground level is wrinkled using different coupling resistors [113]. A simple CNN based wave propagation algorithm, which has been proposed for real time robot control, works as a backward search algorithm in which the solution starts form the target point of the searched path [114]. The continuation of that work demonstrates a gathering application of multiple robots [115], which is an easy problem after producing the feedback plan using wave-front propagation. Another CNN which is capable of contracting autowaves as well as propagating them is proposed in [116]. This network does not generate feedback plan as the cells do not have memory to save the wave propagation vector. Instead of this, propagated wave is contracted with two fixed endpoint. At the end of contraction, the shortest path is revealed. In 2010, the similar results from a FitzHugh-Nagumo network have been announced by [117]. Not only electronic implementations, also chemical setups coupled with electronically implemented active mediums are researched for solving the shortest path problem. In [118] and [65], reaction-diffusion mediums are realized by chemical processors with collaborating Cellular Automata (CA) and CNN, respectively. A Cellular Logic Network (CLN) for binary traveling (trigger) wave

propagation is designed and applied for morphological image processing in [119], and [66]. The CLN is also proper for motion planning [72]. Moreover, the architecture of CLN and CA resemble each other [120]. A recent paper demonstrates that planning with CA performs both optimality and the time efficiency in a better way than other planning methods [121]. A distance propagation dynamic system [122] which is an algorithm for robot navigation for dynamic environment has been developed and formalized as a CNN in [123].

CNN serves exceptionally well for discrete motion planning and the Doppler Effect supports the success of CNN with providing new feature generating ability. Before explaining how this happens,the Doppler Effect is reviewed. It is the change in frequency of a periodic event, such as wave, for an observer in a motion relative to the source. In 1982, Christian Doppler, who is the eponymous physicist of the effect, defined the frequency shift effect based on the colorful light observation of moving stars [124]. By means of Doppler's proposal, binary stars who are star systems consisting of two stars rotating around their common center of mass are clarified. Although the origin of Doppler's proposal is about the visible bandwidth of electromagnetic waves, it is a phenomenon for all kind of periodic events. If one keeps the propagation speed of an event constant and generates that periodic event in different spatial points, any observer who is stationary in reference to the origin of the space in which the event occurs observes the Doppler Effect. The general form of observed frequency f_o is given by $f_o = \frac{c-v_o}{c-v_s} f_s$, where c is the propagation speed of event, v_o is the observer's radial velocity, v_s is the source's radial velocity, f_s is the frequency of the event at source. Doppler Effect also serves when detecting the speed of atmospheric objects. Doppler radars solve this problem by transmitting electromagnetic wave to the target object, then sensing the frequency shift of the reflected and received signal. Electromagnetic waves employed for Doppler Radar may be continuously transmitted or be a repeated pulse [125]. Doppler Effect occurred in sound waves derives applications about flow measurement. In Acoustic Doppler Velocimetry, signal is transmitted to a flowing liquid in a cylindrical pipe and using four receivers, the instant velocity is measured [126]. Echocardiogram employs ultrasonic waves to determine the speed and direction of blood flow [127]. Laser Doppler Velocimetry

overcomes the difficult conditions such as high pressure or high temperature [128]. Some optical computer mice are also based upon the Laser Doppler Velocimetry.

As is given, Doppler Effect of physical waves in the real world has various applications. However, this section resumes the Doppler Effect research of non-linear waves on active media. The shape and amplitude of autowaves remain constant during he propagation. They do not reflect at the medium boundaries and two colliding autowaves annihilate each other [69]. The source of autowave generates successive wave-fronts without a periodic input. Traveling waves, which is another kind of nonlinear waves, is triggered by an input. Then, only one or a few wave-fronts are generated and propagated in the medium. Successive traveling waves produced by a periodic input may have pattern similar to an autowave. Another kind of nonlinear waves is spiral wave that has a special self synchronization. The researches show that the Doppler Effect may modulate the wavelength and the amplitude of the spiral waves in a reaction-diffusion medium [129]. In the literature, it is possible to find solutions to robot navigation problem that uses the Doppler Effect of acoustic waves in a bio-inspired way [130]. Navigation applications using the Doppler Effect are mainly improved on known sonar techniques. Scientific surveys about the Doppler Effect on active media are performed particularly in physics and medicine. A paper represents that it is possible to modulate the frequency of the autowaves in chemical active media by the change in light intensity [131]. Obtained results from these fields of science cause excitement about exercising the Doppler Effect on CNN-based active media. Utilizing the Doppler Effect on active media for path planning and navigation is one of the novelties of this thesis.

5.1.1 Generating feedback plan by relaxation oscillator networks

The fundamental questions are if there is any path for a robot that takes it to the target point, and which one is the shortest if there are more than one path. To solve these problems two path planning algorithms are presented in this subsection. The first one is Wavefront Diffusion Based Algorithm and the second one is Wave Accumulation and Gradient Based Algorithm. Traveling waves are employed in both algorithms. In Subsection 5.1.2, they are called classical algorithm as they are solving the shortest

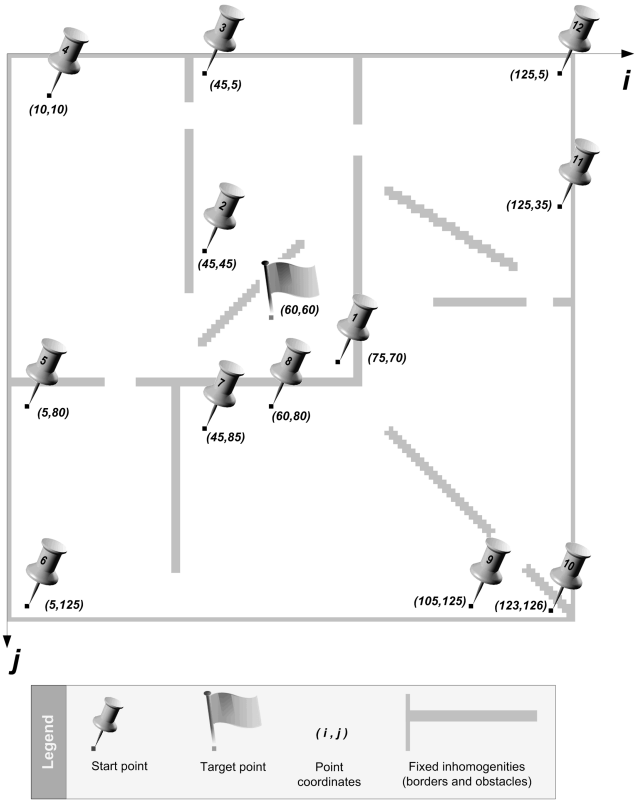


Figure 5.1 : Reference map.

path problem without any prediction. In Figure 5.1 a reference map is presented which will be used to describe and test the algorithms.

Traveling wave-front itself offers a solution for path planning problem. To use this solution, the history of propagating wave can be recorded and processed. Ito *et al.* [64] expressed a method based on this phenomena in 2006. For Wavefront Diffusion Based Algorithm, all y-state and u-input values are set to 0s. Initial x-state matrix has three different nodes: fixed nodes, active non-source nodes and an active source node. The map, which is shown in Figure 5.1, have these nodes. Fixed nodes which are represented with gray color in Figure 5.1 form the obstacles and boundaries, and during wave computing they get the constant value $x_{\text{fixed}} = 0$. The dark-gray node which has a flag on the reference map is the target point. In other word it is the source of the wave and gets the initial value $x_{source} = -1.122$. All other nodes including the white ones and black ones are active non-source nodes and they get the initial value $x_{initial} = 0$. The black colored start points on the reference map get their difference than the white colored nodes at the end of the wave propagation step. At the initialization step these black colored start points are ignored and assumed to be white. The state

coefficients are set as $\alpha = 4$, $\beta = 0$, $\varepsilon = 0$ and $\sigma = 0$. The weights of the synaptic law are $a_{i,j+1} = 0.8$, $a_{i-1,j} = 0.8$, $a_{i,j-1} = 0.8$ and $a_{i+1,j} = 0.8$. The slope of the function $g(\cdot)$ is $m = -20$, and its limit value is $limit = 0.90$.

Figure 5.2(a) shows the source node and its neighbors at the beginning of the iterations. The x state variables of these nodes get the values shown in Figure 5.2(b) at the end of the first iteration. Also, the second iteration values are given in Figure 5.2(c). At each iteration, nodes which are touched by the wave are determined. The threshold value for the wave is set to 0. The traveling wave appears on the node whose x-state value is smaller than 0. Therefore, only the source node $(60,60)$ holds the wave at the beginning. Then the wave propagates to its 4-neighbors. Because of the synaptic law, the diagonal neighbors do not affected directly by the center node. The wave image is constructed by thresholding the x-state matrix. The wave image of the previous iteration is also kept. A pixel-wise exclusive-or operation are performed using the previous and the current wave images, and then the wave-front image is generated. This wave-front image highlights the nodes which are newly touched by the wave. At this step the direction of the wave must be computed. The neighbors of a newly touched node carries the direction information. For this algorithm, the direction of the wave-front propagation is represented by a vector which starts from the neighbor node which has the maximum absolute x value and finishes at the newly touched node.

In Figure 5.3(a) the node $(56,45)$ has not been touched by the wave yet. It has got the value -0.001 in Figure 5.3(b), so the wave has propagated to this node. Because the east neighbor has bigger absolute value than the north neighbor, the wave propagation

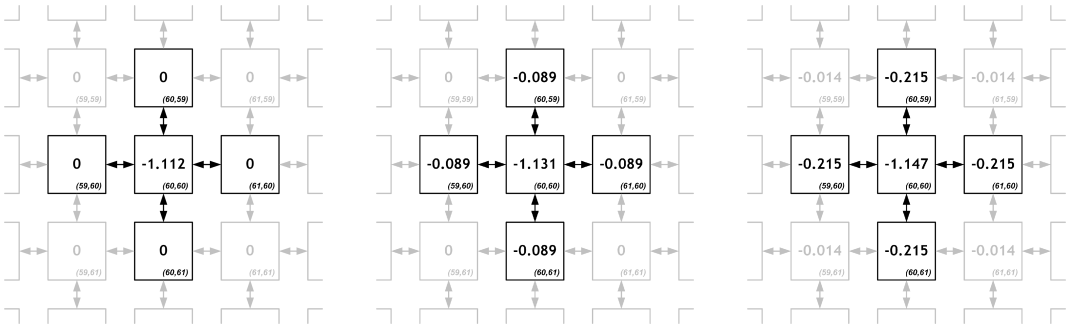


Figure 5.2 : x state values of the nodes surrounding the source node during the first two iterations: a) initial values, b) the first iteration results, c) the second iteration results.

direction for the node $(56,45)$ is determined from east to this node, that is a west directional vector. Figure 5.3(c) shows the following iteration results.

In this wise, traveling wave covers all of the active nodes and vectors which show the propagation directions for all nodes are calculated. After the wave propagation finishes, a vectorial image is generated as shown in Figure 5.4(d) and using this image, the path from starting node to the target node is computed. The starting node can be any of the nodes but the target node is always the source node. The path is the sum of vectors multiplied by -1 from starting node to the target node, through the neighbors which are shown by the vector multiplied by -1. The outputs of this algorithm on the reference map in Figure 5.1 are given in Figure 5.5.

The history of wave propagation is recorded in previous algorithm. In Wave Accumulation and Gradient Based Algorithm, the history is recorded on the Cellular Nonlinear Network itself. Like previous algorithm, all y-state and u-input values are set to 0s. Initial x-state matrix again has three different nodes: fixed nodes, active non-source nodes and an active source node as shown on reference map in Figure 5.1. For this algorithm each node accumulates the values of its x-state during whole emulation and stores it on y-state. To achieve this, the state coefficients are set to $\alpha = 0$, $\beta = 0$, $\varepsilon = 1$ and $\sigma = 0$. The weights of the synaptic law are $a_{i,j+1} = 1$, $a_{i-1,j} = 1$, $a_{i,j-1} = 1$ and $a_{i+1,j} = 1$. The slope of the $g(\cdot)$ function is $m = -20$, and its limit value is $limit = 1$. Source node gets the initial value $x_{source} = -1.122$ and other active nodes gets the value $x_{initial} = 0$. Differently from the previous algorithm, fixed boundary and obstacle nodes are set to $x_{\text{fixed}} = 0.00005$ at initialization. With this initial values the

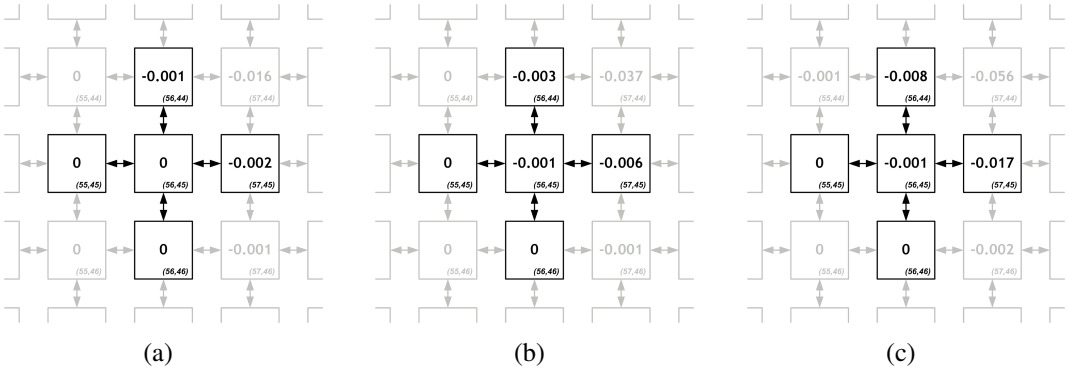


Figure 5.3 : x state values of the nodes surrounding the node $(56,45)$ during the iterations 103 to 105: a) at iteration 103, b) at iteration 104, c) at iteration 105.

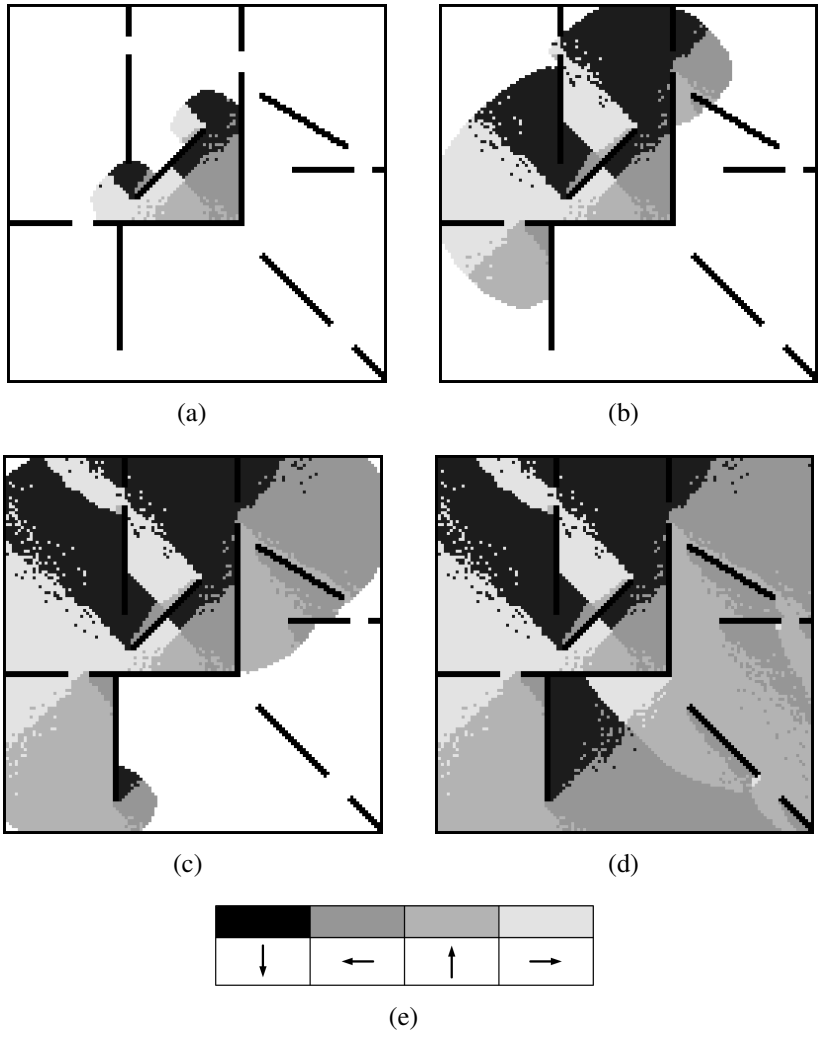


Figure 5.4 : Output vectorial images of wavefront diffusion based algorithm with sequencing iterations numbers: a) step 100, b) step 200, c) step 300, d) step 527 which is the final step, e) the vector legend.

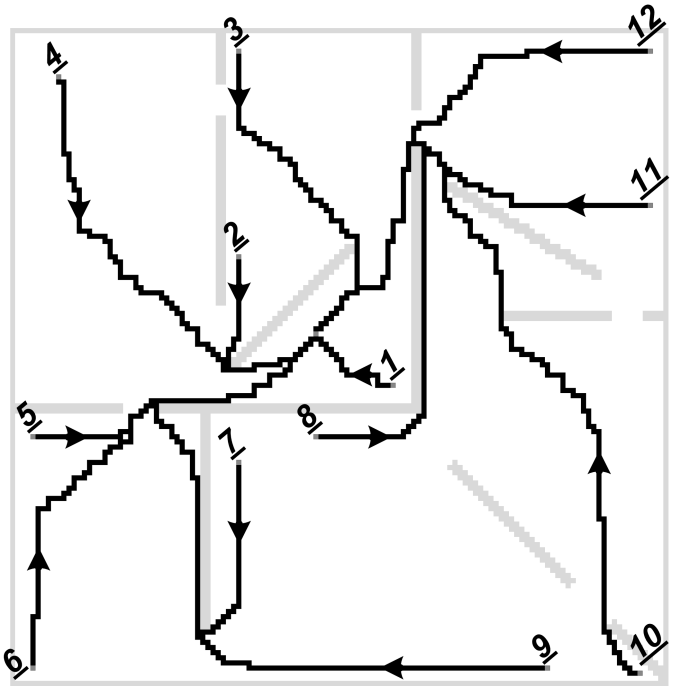


Figure 5.5 : Traces found by the Wavefront Diffusion Based Algorithm.

network is emulated for 750 iterations. This iteration number gives sufficient time to the wave while covering the whole network.

As shown in Figure 5.6, traveling wave is generated by the node $(60, 60)$ and it propagates on the x-states. The y-states are the integral of the x-states as presented in (3.1). At each iteration, the sum of current x-state values and current y-states values are stored on the y-states. Because the initial value of the source node is negative, the y-state of this node becomes smaller at each iteration. Also that makes the neighbors become smaller but they do not exceed the source node. This rule is valid for all nodes. y-states decrease for all nodes, but none of them exceeds its neighbor which the wave is propagated by. So, the source node always has the minimum y-state value. When the wave covers the whole network, y-state matrix becomes a topographical map. The lowest point on this map is the source node. The highest points are the obstacles and boundaries. Then, the node which is touched last by the wave, is the second highest point on the map. Figure 5.7 shows the wave propagation and three dimensional topographical map evolution step by step. In this configuration, cell's of Network 1 behaves as 1-st order stable systems, state equation for y is removed form the cell dynamics and it is employed for just integrating the state variable x.

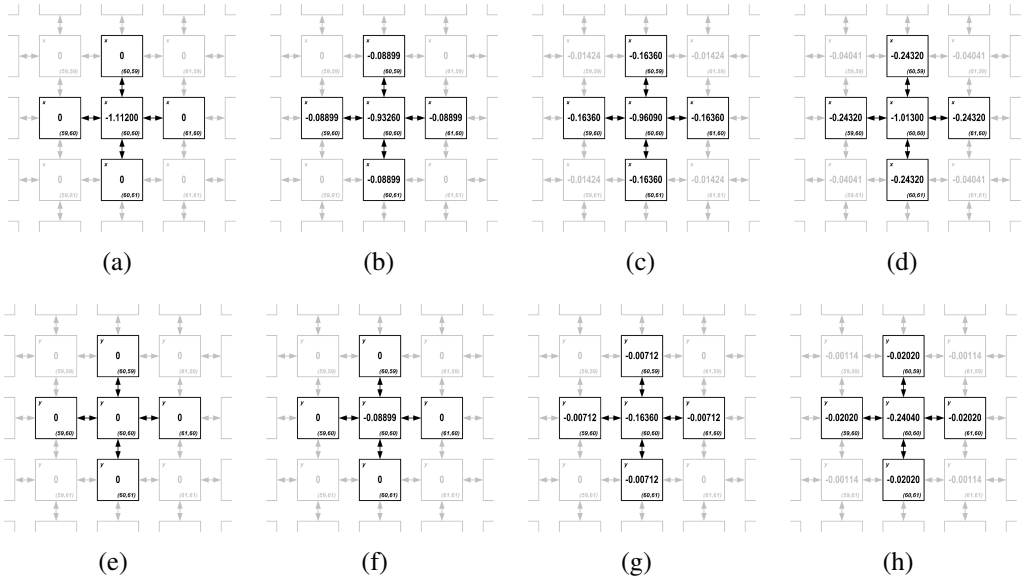
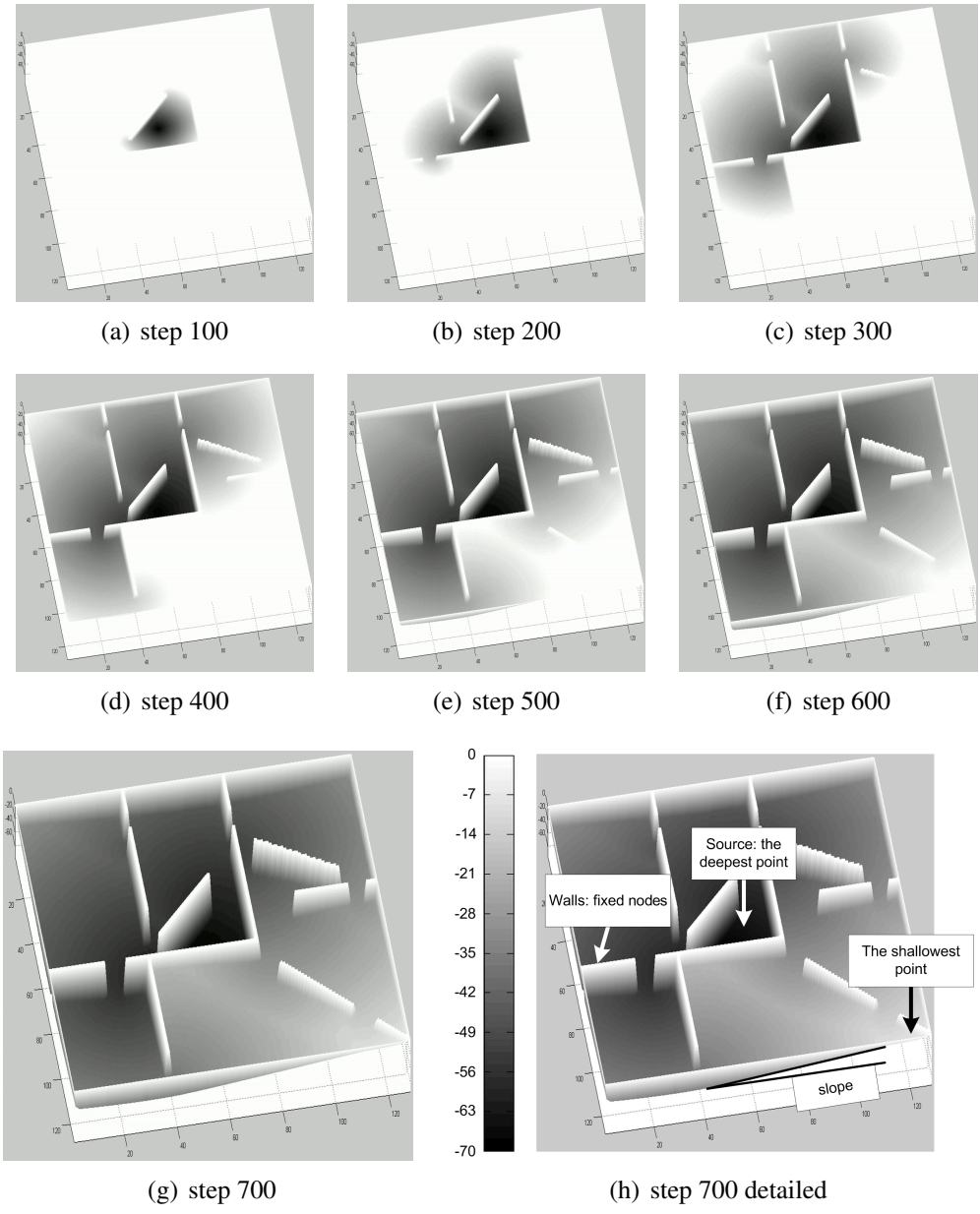


Figure 5.6 : *x* and *y* state values of the nodes surrounding the source node during the first three iterations: a) initial *x*-states, b) *x*-states at the 1*st* iteration, c) *x*-states at the 2*nd* iteration, d) *x*-states at the 3*rd* iteration, e) initial *y*-states, f) *y*-states at the 1*st* iteration, g) *y*-states at the 2*nd* iteration, h) *y*-states at the 3*rd* iteration.

After the propagation ends, the coefficients and weights are updated and gradients of this topographical map are computed. To do this 4 final iterations are executed. The first final iteration copies the *y*-states onto *x*-states, the second one computes only the horizontal gradients of the map, the third one swaps *y*-states and *x*-states and the last one computes only the vertical gradients of the topographical map. The horizontal and vertical gradient components are given in Figure 5.8(a) and Figure 5.8(b), respectively. To execute the first final iteration the parameters are set to $\alpha = -1$, $\beta = 1$, $\varepsilon = 0$, $\sigma = 0$, $a_{i,j+1} = 0$, $a_{i-1,j} = 0$, $a_{i,j-1} = 0$, $a_{i+1,j} = 0$, and $m = 0$. The horizontal gradient is computed at the second final step by using the same parameters except $\beta = 0$, $a_{i-1,j} = -1$, and $a_{i+1,j} = 1$. The third iteration swaps the states by using the same parameters except $\beta = 1$, $\varepsilon = 1$, $\sigma = -1$, $a_{i,j+1} = 0$, $a_{i-1,j} = 0$, $a_{i,j-1} = 0$, and $a_{i+1,j} = 0$. The last iteration which computes the vertical gradients are executed after the parameters are set to $\beta = 0$, $\varepsilon = 0$, $\sigma = 0$, $a_{i,j-1} = -1$, and $a_{i,j+1} = 1$ while others remain the same.

The horizontal and vertical gradient components, which are computed for any active node, are used to draw the path for this algorithm. Following these vectors form starting point guides the robot to target point. Also, giving the initial value $x_{\text{fixed}} = 0.00005$ to the boundaries and obstacles makes a positive accumulation effect to their

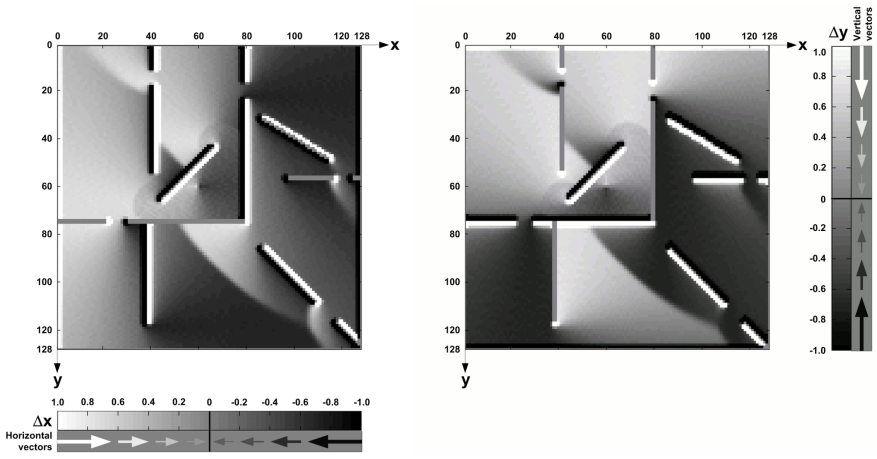


(a) step 100 (b) step 200 (c) step 300

(d) step 400 (e) step 500 (f) step 600

(g) step 700 (h) step 700 detailed

Figure 5.7 : Accumulation images of Wave Accumulation and Gradient Based algorithm with sequencing iterations numbers: a−g) from top view, h) step 700 detailed with labels.



(a) (b)

Figure 5.8 : Output gradient images of Wave Accumulation and Gradient Based algorithm: a) only horizontal gradient (Δx), b) only vertical gradient (Δy).

neighbors and the output gradient vectors move away from them. In Figures 5.8(a) and 5.8(b), strong horizontal vectors can be seen next to the vertical walls and strong vertical vectors can be seen next to the horizontal walls. This makes the robot avoid from obstacles easily.

Reference map in Figure 5.1 has 12 start points. Both algorithms ignore these points and consider them as white active nodes while wave propagation. After vectorial data has been produced the software draws the path from a determined start point to the target point by using these vectors. In Figure 5.5, the paths from the given 12 start points to the same target point are found by Wavefront Diffusion Based Algorithm and drawn on the same map.

In Figure 5.9, the paths are found by Wave Accumulation and Gradient Based Algorithm and again drawn on the same map. The Wave Accumulation and Gradient Based Algorithm produces shorter, smoother paths than the first algorithm. Also its paths do not touch the walls while the paths of Wavefront Diffusion Based Algorithm are touching. In the Wavefront Diffusion Based Algorithm, the hardware requires sending each iteration result to the Host Computer over the serial communication line. Because communication bandwidth is not large enough, this algorithm is slower than the Wave Accumulation and Gradient Based Algorithm. The speed of Wave Accumulation and Gradient Based Algorithm is proper to use the algorithm in real-time robot navigation algorithm. The proposed algorithms in this subsection show

that the cellular nonlinear network model given by Network 1 in Subsection 3.1.1 is suitable to use in path planning applications.

5.1.2 Predictive planning in 2D discrete space

Doppler Effect brings information about target's velocity by propagating wave-fronts. In this subsection, CNNs are further investigated for nonlinear wave-front propagation based feedback motion planning method. For the first time, a prediction ability is added to the investigated networks using the Doppler Effect. Three different networks, Network 1, Network 2 and Network 3, perform satisfying results at a chasing scenario. One should note that information about target's velocity, transported by wave-fronts, should be still recent when received by the tracker or other observers on the network. In order to take advantage of Doppler Effect in motion planning, fast evolving networks are desired. When simulating, network process time is dependent on the simplicity of the cells and networks. When implementing, the simplicity effects the utilized component/transistor/gate count. Three cell models in this subsection are foreseen to be easily implemented as electronic circuits, like the example implementations of wave computers [3], [15].

From one point of view, network simulators and network implementations may be evaluated as feature generators. Then, motion planning algorithm becomes the consumer of those features as a decision block and outputs the actions, similar to the structures in odor processing [132] and image processing [66] examples. If the network ability is just the generation of a feedback plan or the shortest path, then the motion planning is assumed to be completed by the network. On the other hand, network generating more than the feedback plan as in Section 3.1 should cooperate with a separate decision block that evaluates generated features. In this subsection, a prediction performed and action is selected according to this prediction by a motion planner using the feedback plan and frequency values provided by the network. This modular organization allows to interchange networks and develop new planners using the same networks. The most suitable role for the networks is being a feature generator co-processor array next to a main processor which is running the motion planning algorithm. This role is still valid for simulations. The network model runs

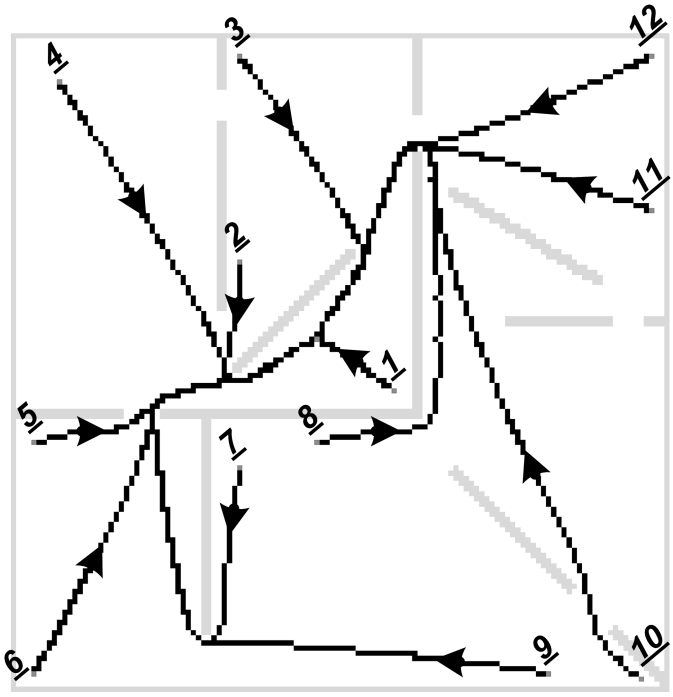


Figure 5.9 : Traces found by the Wave Accumulation and Gradient Based Algorithm.

for generating gradient, angle and period features. In this subsection a simple yet successful motion planning algorithm is presented using these features.

In the test scenario, target moves on a predetermined route in an environment with stationary obstacles depicted in Figure 5.10. Target route starts at the blue cell on the upper left corner. Light blue cells are followed by the target. The final location is the west neighbor of the red cell. Red cell is the initial location of the tracker. There are three different speed values to test. Both target and tracker move at 1 cell/T speed in the first one. In the second one, target moves at 2 cell/T speed and in the third one, tracker moves at 2 cell/T speed. Proposed motion planning algorithm using Doppler Effect is tested at given speed values against the classical algorithm which is a mixture of Wavefront Diffusion Based Algorithm and Wave Accumulation and Gradient Based Algorithm in Subsection 5.1.1. The calculation of gradient function in Wave Accumulation and Gradient Based Algorithm is embedded to the cells and carried of at every iteration by the cells whom touched by the wave as in Wavefront Diffusion Based Algorithm. The slight modification to build the classical algorithm provides a good reference for the comparison.

The feature generator consists of a two layer active media as depicted in Figure 5.11. In one layer, traveling wave source location is the target location. In other layer,

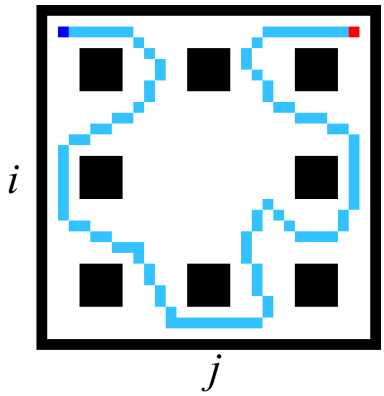


Figure 5.10 : Scenario: Obstacles and the predetermined route of the target.

wave source location is the tracker location. They are called target layer and tracker layer, respectively. Both layers are equipped with Doppler Effect sensing mechanism, wave-front gradient sensing mechanism and wave-front normal angle calculator in cells.

The gradient and angle features are calculated based on the methods in Subsection 5.1.1. Wave-front gradient sensing mechanism is constituted by two simple nonlinear equations given by

$$\begin{aligned} v_{i,j} &= -\text{sign}(\dot{x}_{i,j})(x_{i-1,j} - x_{i+1,j}), \\ h_{i,j} &= -\text{sign}(\dot{x}_{i,j})(x_{i,j-1} - x_{i,j+1}), \end{aligned} \tag{5.1}$$

where $h_{i,j}$ is the horizontal, $v_{i,j}$ is the vertical wave gradient on (i,j)-th cell. For Network 3, instead of $-\text{sign}(\dot{x}_{i,j})$ term, $(x_{i,j}[k] - x_{i,j}[k+1])$ is used.

Wave-front normal angle calculation starts with finding raw angle value $\hat{a}$ using $\arctan(\cdot)$ function given by

$$\hat{a}_{i,j} = \arctan(\frac{-h_{i,j}}{v_{i,j}}). \tag{5.2}$$

Then, the exact angle is determined using Table 5.1.

Actually, wave-front normal angle calculation is not as essential as gradient sensing. It is employed to have better visualization in this algorithm. Only gradient vectors provide the information that the algorithm needs.

In this scenario, obstacles are stationary. Methods using reinitialization may update the obstacle pattern at every reinitialization process. Nonetheless, it causes a discontinuity in wave evolution that wipe out the Doppler Effect. Thus, methods in this subsection

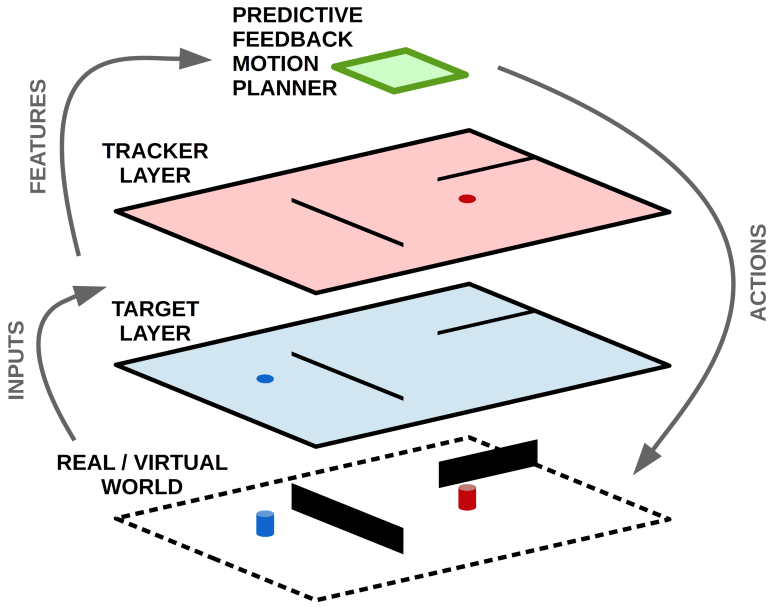


Figure 5.11 : Mapping real world to network layers.

Table 5.1 : Exact angle values using horizontal, vertical gradients and temporary angle.

Gradient vectors	Exact angle value
$h<0, x<0$	$a=\pi+\hat{a}$
$h<0, x=0$	$a=\pi/2$
$h<0, x>0$	$a=\hat{a}$
$h=0, x<0$	$a=\pi$
$h=0, x=0$	$a=$ not available
$h=0, x>0$	$a=0$
$h>0, x<0$	$a=\pi+\hat{a}$
$h>0, x=0$	$a=3\pi/2$
$h>0, x>0$	$a=2\pi+\hat{a}$

are not suitable for reinitialization and the obstacles in real world are assumed to be stationary.

Two layers are initialized by switching the cells that correspond to the obstacles to the boundary cells that realize zero-flux boundary condition. Hence, any obstacle does not become an undesired wave source. The rest of the network's cell states are initialized to a saddle point (continuous systems) or a peak state (digital system). All inputs are set to 0. According to the measured cell settling time values, T is set to 5. Continuous-time systems are discretized by Forward Euler Method and the integration step h is set to 0.1. At the same time, h is the clock period for the digital network. All network parameters used in these simulations are the ones given in Subsection 3.1.1 to Subsection 3.1.3.

At every T seconds, the target location is updated according to the route given in Figure 5.10. A traveling wave generating input signal, specific to the network type, is applied to the cell at the target's location on the target network. The target network evolves by updating the target location successively and propagates nested traveling waves whose wave-front updates the gradients (h, v), normal angles (a) and relative half-periods (d) as the features.

On the tracker network, the wave evolution continues similarly to the target network. Same type of features are generated. However, deciding the next location of tracker is the duty of motion planning algorithm. Features are fed to the algorithm, and it generates the next location. As illustrated in Figure 5.11, there is a loop closed by the motion planning algorithm.

The proposed motion planning algorithm is simple. At the end of every T seconds, features from the two layers are sampled to the planner. Planner separates the network grid into three regions. Region 1 consists of cells that both target and tracker are approaching on opposite directions. Region 2 consists of cells that only target is approaching but the tracker is receding again on opposite directions. Region 3 is the rest of the grid. Regions do not have to be connected components. Then, the planner computes the centroid of Region 1 and Region 2. For Region 3, the current location of the target is used instead of its centroid. The priority of computed points decreases from Region 1 to 3. If Region 1 exists and the calculated centroid is not on any obstacle, then planner assign that centroid as the temporary target location. Unless Region 1 exists, then centroid of Region 2 is assigned as the temporary target location. The target's current location is used if Region 1 and Region 2 do not exist. The paths from any cell of the tracker layer to the tracker are solvable due to the angle (or gradient) information provided by tracker network. Angles from tracker network forms the feedback plan. The last step of planner is determining the route from the temporary target location to the tracker location. Planner moves the tracker on the determined route depending on the tracker speed.

Proposed motion planner aims to drive the tracker to predicted meeting point instead of target itself by giving the highest priority to the Region 1. Region 2 gives opportunity to consider an important route alternative. Having a region that approaching target and receding tracker wave-fronts are on opposite directions is a strong indicator that some

of the Region 2 cells may turn into Region 1 cells if tracker starts to move on opposite direction. Without these two predictions, aiming at the targets current location is the classical method and usually means just tagging behind and no catching.

Figure 5.12 exhibits the motion planning algorithm results in time. The network employed in the simulation is Network 2. Region 1 is painted with green, and Region 2 is painted with yellow. Dark blue squares show the executed part of the predetermined route steps of the target. Dark red dots are the previous tracker positions. Brighter blue and red dots are the current positions of the target and the tracker, respectively. Obstacles are black and Region 3 is white. The temporary target is indicated by with a red square frame. Subfigures belong form $t = 20$ to $t = 120$. Tracker catches the target and the final routes are shown in Figure 5.14(b).

Figure 5.13, 5.14, 5.15 show the results fo test scenario simulations in which Network 1, 2, and 3 are employed, respectively. The network type, object speeds and motion planning algorithm vary in their subfigures. In both three figures, objects speeds vary from top to bottom as follows. In the first row, both target and the tracker has 1 cell/T speed. In second row, target has the same but the tracker has 2 cell/T speed. In the third row, target has 2 cell/T and tracker has 1 cell/T speed. The first column shows the results using classical planner, while the second row shows results for the proposed planner. Consequently, simulation results have been exhibited in order to compare network successes.

According to those figures, trackers using classical planner are not able to catch the target before it reaches to its destination, if their speeds are equal (Figures 5.13(a), 5.14(a), 5.15(a)). The tracker routes are different from each other which is a result of different network dynamics. The trackers using classical planner unexceptional catches the tracker when they move 2 times faster than the target (Figure 5.13(c), 5.14(c), 5.15(c)). In all three simulations, the meeting point does not change too much. It is total opposite if the target speed is faster. The tracker is mostly unsuccessful (Figure 5.13(e), 5.14(e)) with an exceptions. In Figure 5.15(e), the slowly moving tracker meets with the target around its initial position, because the destination of the target is almost initial location of tracker. It should be noted that, even a stationary tracker has a very small chance to meet the target.

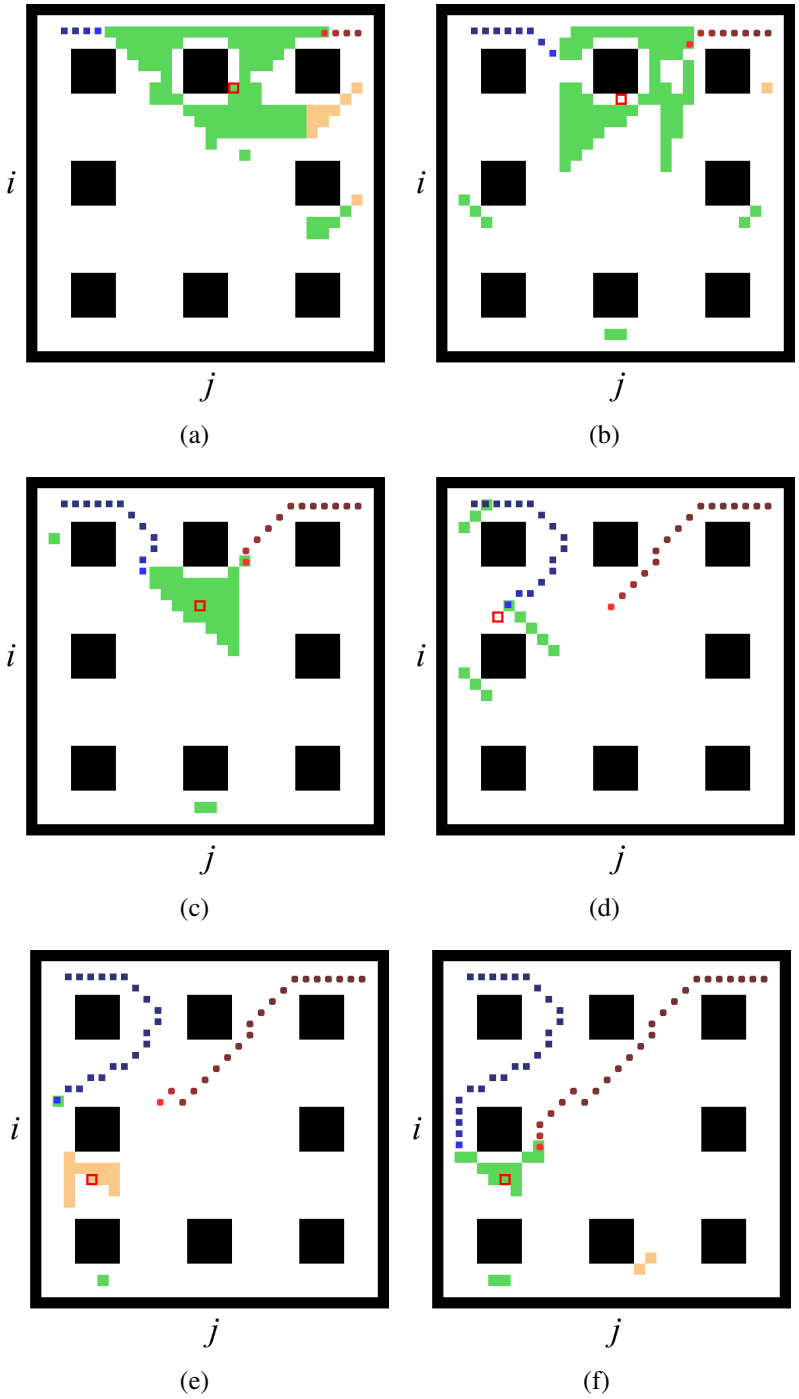


Figure 5.12 : Simulation snapshots of proposed predictive motion planning. Both are at equal velocities. Region 1 is green, Region 2 is yellow, Region 3 is white. Blue points are the steps of target. Red steps belong to the tracker.

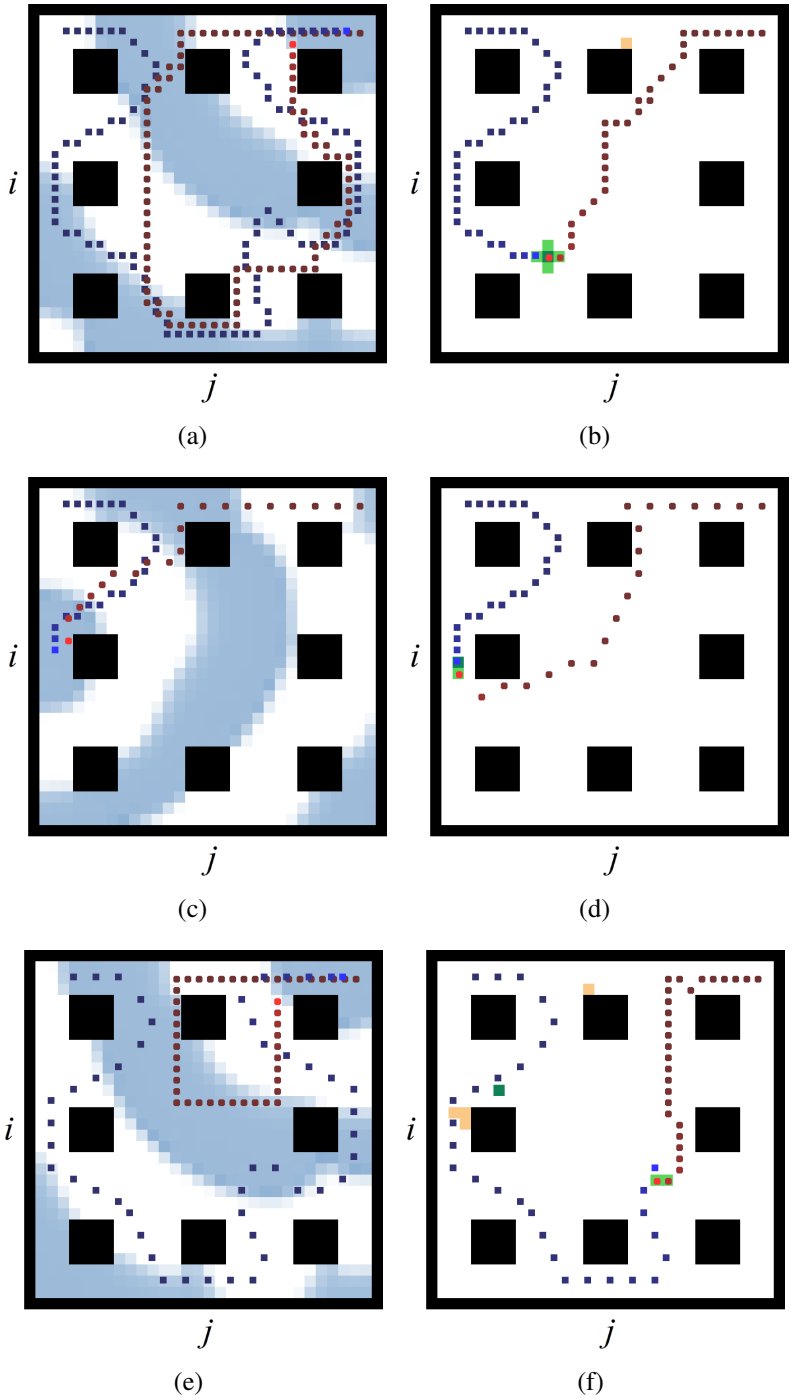


Figure 5.13 : Simulation results of Network 1.

Simulations show that, proposed motion planner succeeds in every situation. Using Network 1 (Figures 5.13(b), 5.13(d), 5.13(f)) and Network 2 (Figures 5.14(b), 5.14(d), 5.14(f)), tracker catches the target without passing redundant cells. The significant result is that the catching time is smaller when Network 2 is used instead of Network 1, in all situations. Even though Network 3 has the least settling time, the highest wave propagation speed, and seemingly the simplest implementation; the tagging performances are not so satisfying. In the equal speed situation, catching time is longer than the others (Figure 5.15(b)). The worst behavior occurs when the tracker is faster than the target and the planner employs Network 3 (Figure 5.15(d)). In this situation tracker passes many redundant cells before the meeting. Affirmatively, solution using Network 3 resembles the ones using other two networks (Figure 5.15(f)).

According to the simulation results, if the only choice is the classical planner, then the tracker should be faster than its target. On the other hand, utilizing the Doppler Effect gives the opportunity to predict a meeting point and catch the target at equal speed. If Doppler Effect will be utilized, then the proposed planner with Network 2 is the optimum solution among the three network. It is simpler than Network 1, catches faster than Network 1, has less redundancy than Network 3. One should choose Network 3, if the least computational cost is desired.

5.2 Random Bit Generation

Random number generators (RNGs) are employed in a variety of applications such as stochastic optimization, computer simulation, cryptography, etc. In these areas, a good random number generation is essential for cryptographic security. The security of cryptographic algorithms relies on generating secret quantities, which are produced by RNGs. Random numbers can be generated by a random bit generator, which can be defined as a device or algorithm whose output is a sequence of statistically independent and unbiased binary digits. To ensure that an RNG is secure, its output must be computationally indistinguishable from a true random sequence [133]. Practically, NIST's test suite SP 800-22rev1a is used as an up-to-date tool for qualification of RNGs [134]. Besides the NIST's test suite, others such as DIEHARD [135] and TestU1 exist in the randomness testing field [136].

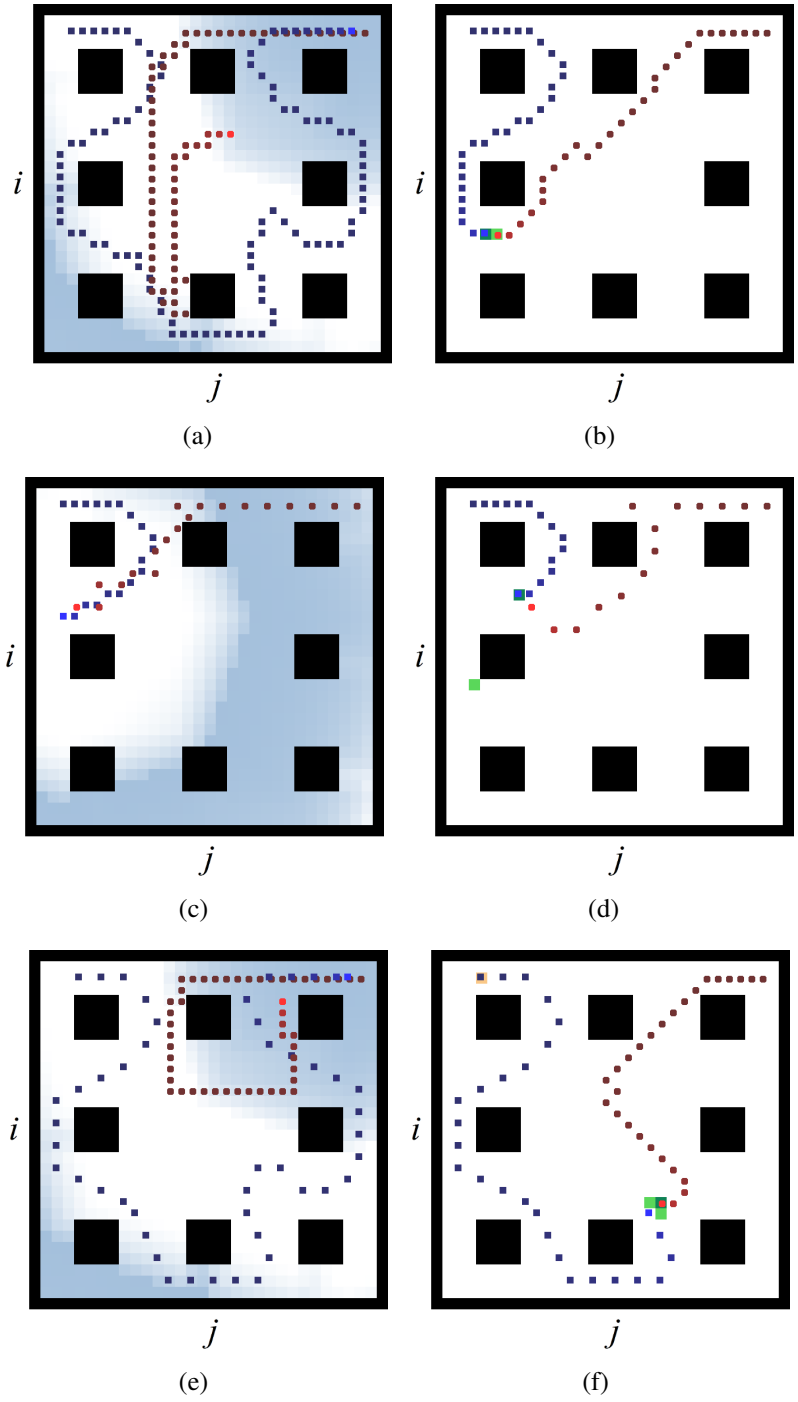


Figure 5.14 : Simulation results of Network 2.

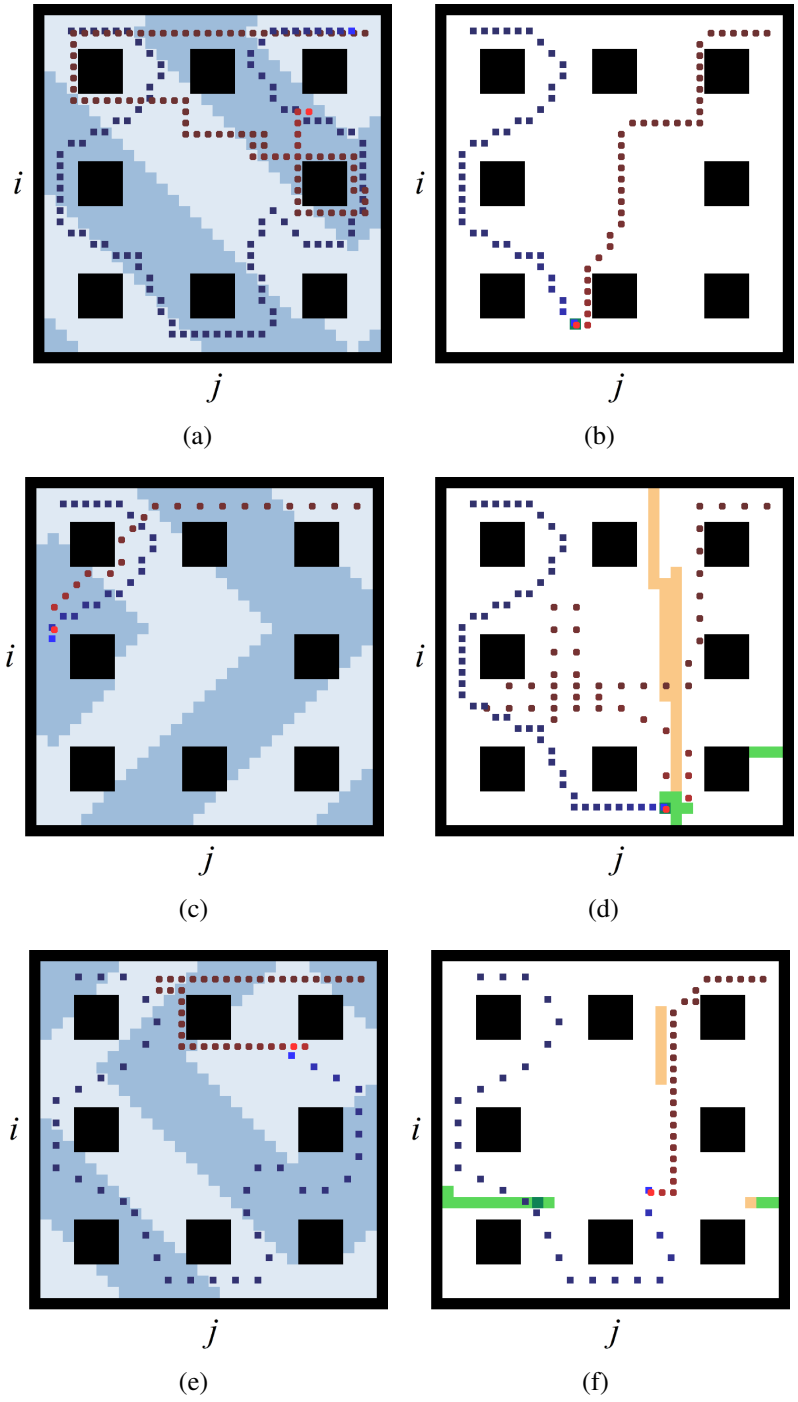


Figure 5.15 : Simulation results of Network 3.

In cryptography, there are true RNGs (TRNGs) and pseudo RNGs (PRNGs). TRNGs are based on measuring unpredictable natural processes, for example thermal noise, frequency instability of an oscillator, elapsed time between emission of particles during radioactive decay and variations in systems response times. On the other hand, PRNGs are based on deterministic processes and they generate a series of outputs from an initial seed state. TRNGs are generally costly and slow responding devices when compared with PRNGs. But, the actual entropy of the PRNG output can never exceed the entropy of the seed, because the output is a function of the seed state, [137]. Hence, the randomness level of the pseudo-random numbers depends on the level of randomness of the seed [133].

It is reported in the literature that physical noise can be used as the entropy source in random number generation. An RNG structure based on this idea is given in [138]. On the other hand, sensitivity with respect to the initial conditions causes unpredictability in chaotic systems. They produce irregular signals with a noise-like spectrum [139]. Therefore, one possible way to generate random numbers is to use chaotic signal (instead of physical noise) as the unpredictable source. There are many chaos based RNGs reported in the literature [140–143].

To generate chaotic dynamics, systems defined by delay-differential equations can also be used. The paper by Mackey and Glass [19] is an early work in the area of time-delay chaotic systems. Since then, there has been a growing interest in delay-differential equations for the generation of chaotic signals because of their simple model. There are many time-delay chaotic systems reported in the literature [19, 21, 52, 53].

In Subsection 5.2.1, the promising circuit realization in Subsection 4.3.2 is tested as a random bit generator. First, it is demonstrated that the sampled binary sequence on the feedback has successfully passed the statistical tests. Then, two instances of the same circuit have been employed in order to enhance throughput of the random bit generator. The third component of the section is employing the digital implementation of the same system in Subsection 4.3.4 as a TRBG. In Subsection 5.2.2, the quantification of anticipating synchronization is given by bit error since the coupled systems produce binary outputs. It is treated in the frame of an attack on the TRBG in Subsection 5.2.1.

5.2.1 True random bit generator

Today, the well accepted test suite for randomness is the NIST's 800-22rev1a Statistical Test Suite. Therefore, the bit strings recorded from the given circuit's delay line has been examined using this test suite for this subsection. An auxiliary FPGA-based implementation samples 40 million bits with T_b period from the output of the last flip-flop on the delay-line and transfers them to a computer in order to apply the tests.

R_1, R_2, V_{pos}, V_{neg}, τ and T_b are the dimensions of the configuration space of the circuit. Tentative searches were made on this configuration space to gain the best statistical results. To speed-up the investigation, FIPS-140-1 Test Suite, which is the predecessor of the 800-22, is firstly applied to some 20-thousand bit parts of the long bit strings. There is a correlation between the Poker test results in FIPS-140-1 and general NIST 800-22rev1a success. Computer controlled investigations on τ and T_b parameters' effect with manually adjusted resistor and voltage values yield very successful test results. When the given circuit was configured with $V_{CC} = 5$V, $V_{EE} = -5$V, $V_{bias} = 0.5$V, $T_s = 20$ns, C = 5.6nF, $R_1 = 2.78$kΩ, $R_2 = 5.78$kΩ, $V_{pos} = 1.12$V, $V_{neg} = -0.81$V, $\tau = 500\mu$s and $T_b = 50\mu$s, the best Poker test results are recorded. Then, the NIST's 800-22rev1a tests are applied to a 40 million bits long string which is recorded in the given configuration. The test suite partitions the string into 100 sub-strings. It has been observed that the 40Mbit string passed all of the NIST 800-22rev1a tests. Table 5.2 shows the pass proportion of the sub-strings and the p-values. It is accepted that the whole string passes the statistical test with a sub-string success proportion equal or greater than $96/100$. The Non Overlapping Template, Random Excursions and Random Excursions Variant tests are the worst results of actually applied series of statistical tests with the same name.

This subsection proposes the novel easily implementable circuit in Subsection 4.3.2 performs very well in random bit generation. Based on the statistical test results, this true random bit generator's discrete component based version ensures a 20000 bit per second throughput. The spectrum of the recorded $x(t)$ signal is also analyzed and observed that %99 of the signal power stands in the 62 kHz bandwidth. It is estimated

Table 5.2 : The NIST 800-22rev1a tests' summary results for a 40 Mbit string sampled at 20Mbit/s rate.

Statistical Test	p-value	Success Proportion	Result
Frequency	0.275709	100/100	Pass
BlockFrequency	0.657933	99/100	Pass
CumulativeSums	0.455937	100/100	Pass
CumulativeSums	0.401199	97/100	Pass
Runs	0.085587	99/100	Pass
LongestRun	0.304126	100/100	Pass
Rank	0.181557	100/100	Pass
FFT	0.162606	96/100	Pass
NonOverlappingTemplate	0.678686	96/100	Pass
OverlappingTemplate	0.383827	98/100	Pass
Universal	0.191687	96/100	Pass
pproximateEntropy	0.637119	100/100	Pass
RandomExcursions	0.330628	42/43	Pass
RandomExcursionsVariant	0.460664	42/43	Pass
Serial	0.262249	97/100	Pass
Serial	0.334538	99/100	Pass
LinearComplexity	0.455937	99/100	Pass

a much better throughput can be achieved if the VLSI implementation of the circuit with a wider bandwidth is studied.

The circuit realization in Subsection 4.3.2 has been improved by employing two circuits with an EXOR operator. It is shown above that a single circuit can generate bits at a rate of 20000 bps which pass the NIST 800-22rev1a Statistical Test Suite [134]. Nevertheless, the generated bits fail in the test suite when the bit capture period T_b is decreased from 50μs. For an example, the statistical test results are given in Table 5.3 for a 40Mbit record which has $T_b = 20\mu$s capture period. The complete parameter values of this record are given as $V_{\mathrm{CC}} = 5$V, $V_{\mathrm{EE}} = -5$V, $V_{\mathrm{bias}} = 0.5$V, $V_{\mathrm{H}} = 3.3$V, $T_s = 20$ns, C $= 5.6$nF, $\mathrm{R}_1 = 2.78\mathrm{k}\Omega$, $\mathrm{R}_2 = 5.78\mathrm{k}\Omega$, $V_{\mathrm{pos}} = 1.12$V, $V_{\mathrm{neg}} = -0.81$V, and $\tau = 500\mu$s. It also should be noticed that the system exhibits a strange attractor behavior in the $x(t) - x(t-\tau)$ state space as shown in Figure 5.16. On the other hand, there should be a strong relation between the randomness of the generated bits and the non-periodicity of the phase portrait of this system. With given component values, the time constant equals to 15.568μs which results in 10.23kHz cut-off frequency of passive RC filter connected to the U2 for x-integration. Observation on recorded $x(t)$ data sampled with 1MHz shows that 99% of the signal energy is in 65kHz bandwidth. The following part proposes a solution to make the generated bits clearly pass the

randomness tests by doubling the signal source with another circuit and utilizing this twin in an unsynchronized manner.

Firstly, a second circuit which is identical to the first one is prepared. For the analog part, a new circuit board is printed, but the digital part is implemented next to the first one on the same FPGA. Figure 5.17 depicts the circuit pair with the common bit recording buffer. Here, the bits generated by each circuit are subjected to a logical EXOR operation. Hence, a statistically better sequence is obtained from two bit sequences. Still, the FPGA has a single bit record buffer. This buffer captures the output of the EXOR gate. The EXOR operation could be considered as a post-processing operation.

As expected, this new configuration obtains better NIST 800-22rev1a test results. The recorded 40Mbit strings with $T_b = 20\mu$s sampling period clearly pass the statistical tests. Table 5.4 shows the success in tests with numerical results for an example bit string. The similar successful statistical results are achieved with many bit strings recorded by the same setup.

The predecessor of NIST test suite is the FIPS 140-1 tests. Exactly, the Poker sub-test of this old suite, which is an easily implementable fast statistical test, is applied primarily. The average Poker test results of single and double circuit setup for different bit sampling rates are given in Figure 5.18. The black horizontal line on the graph indicates the maximum Poker result to pass. When the second circuit is joined, better Poker results are achieved. In spite of good Poker results, the NIST tests' results do not match at considerable lower sampling periods than ones indicated by vertical lines on the same figure. The relation between the success in NIST tests and other metrics should be analyzed in the future works.

Expanding the TRBG system from one to two first order nonlinear subsystems yields two continuous states ($x_1(t)$, $x_2(t)$) and a 2D continuous state space. The unsynchronized behavior of the entire system can be observed by examining the trajectory of the system state in this state-space. Figure 5.19 shows the screen-shot of an analog oscilloscope which is depicting $x_1(t) - x_2(t)$ space. The homogeneity and the symmetry of the trajectory in this state-space is clearly seen. The trajectory pattern can be observed as a qualitative analysis on the system behavior. The pattern in Figure

Table 5.3 : The NIST 800-22rev1a Test Suite's summary results for a 40Mbit string generated by single circuit at 40Mbit/s rate.

Statistical Test	p-value	Success Proportion	Result
Frequency	0.000000	58/80	Fail
BlockFrequency	0.000000	58/80	Fail
CumulativeSums	0.000000	61/80	Fail
CumulativeSums	0.000000	57/80	Fail
Runs	0.000000	0/80	Fail
LongestRun	0.000000	0/80	Fail
Rank	0.048716	79/80	Fail
FFT	0.000000	0/80	Fail
NonOverlappingTemplate	0.000000	15/80	Fail
OverlappingTemplate	0.000000	43/80	Fail
Universal	0.000000	60/80	Fail
ApproximateEntropy	0.000000	0/80	Fail
RandomExcursions	0.834308	20/22	Pass
RandomExcursionsVariant	0.017912	22/22	Pass
Serial	0.000000	0/80	Fail
Serial	0.509162	80/80	Pass
LinearComplexity	0.739918	79/80	Pass

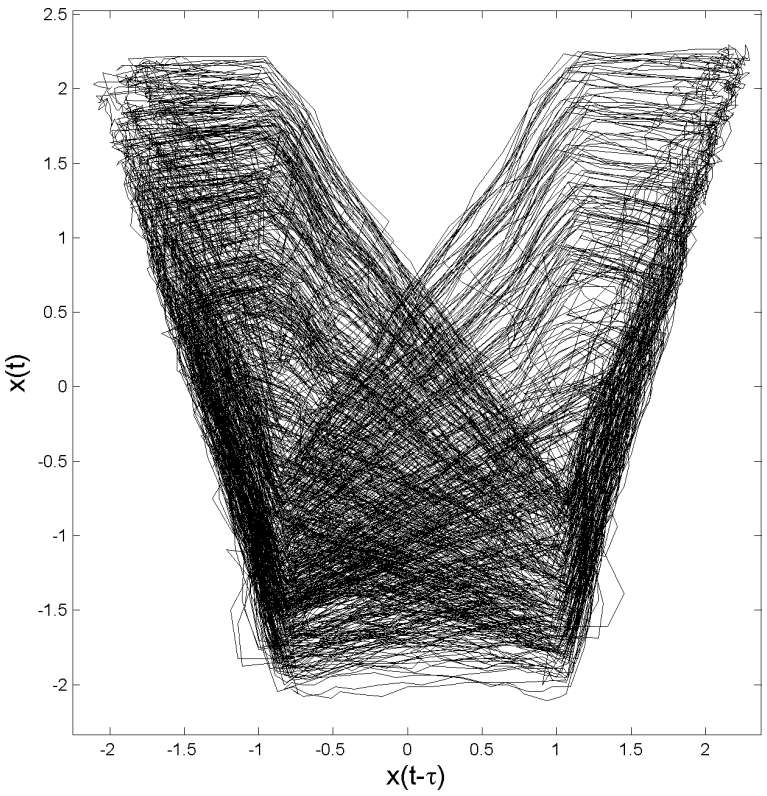


Figure 5.16 : The strange attractor observed on the $x(t) - x(t-\tau)$ plane.

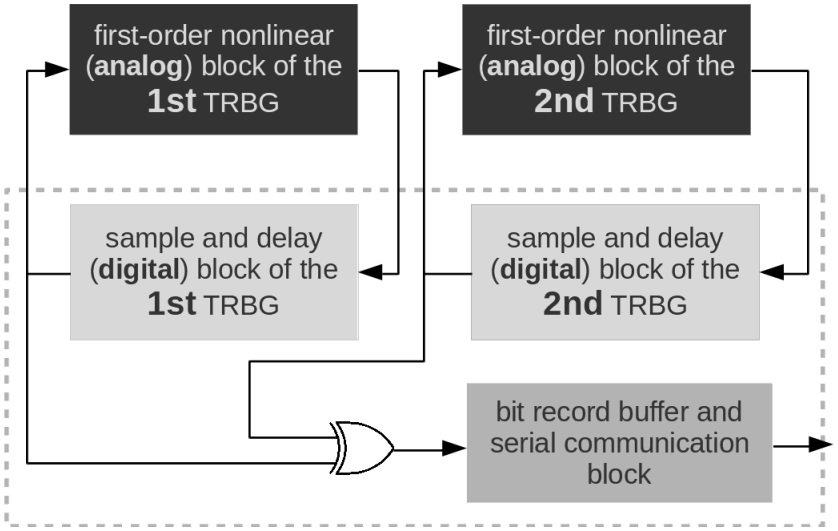


Figure 5.17 : The block diagram of the unsynchronized circuit pair, the EXOR gate and the bit recording buffer.

Table 5.4 : The NIST 800-22rev1a Test Suite's summary results for a 40Mbit string co-generated by a pair of circuits at 50Mbit/s rate.

Statistical Test	p-value	Success Proportion	Result
Frequency	0.941144	79/80	Pass
BlockFrequency	0.986869	77/80	Pass
CumulativeSums	0.739918	79/80	Pass
CumulativeSums	0.258961	79/80	Pass
Runs	0.484646	78/80	Pass
LongestRun	0.012650	78/80	Pass
Rank	0.764655	79/80	Pass
FFT	0.371101	79/80	Pass
NonOverlappingTemplate	0.023149	77/80	Pass
OverlappingTemplate	0.764655	80/80	Pass
Universal	0.275709	79/80	Pass
ApproximateEntropy	0.663130	78/80	Pass
RandomExcursions	0.407091	35/36	Pass
RandomExcursionsVariant	0.100508	35/36	Pass
Serial	0.764655	77/80	Pass
Serial	0.350485	79/80	Pass
LinearComplexity	0.141256	79/80	Pass

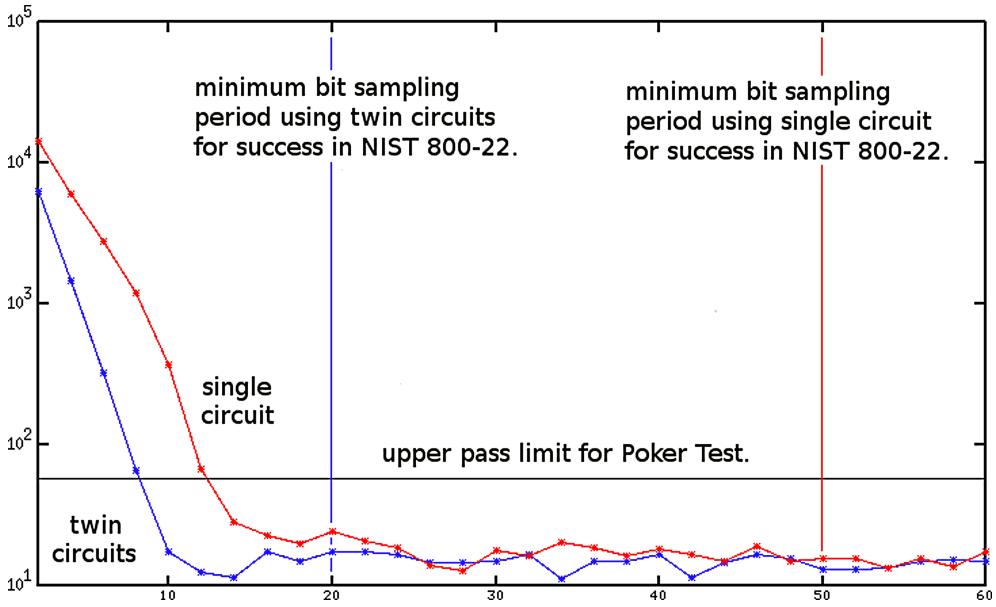


Figure 5.18 : The average Poker Test results for $2\mu s \leq T_b \leq 60\mu s$ interval for single (red) and double (blue) circuit forms.

5.19 can be interpreted as a proof of unsynchronized and unbiased dynamic behavior which has positive contribution to the randomness.

Using two identical time-delay sampled-data feedback systems as independent bit sources and combining their outputs using an EXOR gate increases the throughput by 2.5 times. The randomness of the generated bit strings are tested by the up-to-date statistical test suite NIST 800-22rev1a. Exactly, increasing the number of unsynchronized circuits will yield much entropy, then better statistical characteristics, so the proposed system can be expanded by utilizing more identical subsystems to achieve higher random bit generation rates. It is also expressed that the proposed system can be easily integrated on a silicon chip, owing to the digital delay line and simple analog part. Decrease in the time constant value of the integrator and using faster operational amplifiers will widen the bandwidth of the analog circuits and also the random bit generation rate as well.

The third and the last TRBG proposed in this thesis is the digital implementation of the Oscillator 4, explained in Subsection 4.3.4. Target FPGA is Xilinx XC6SLX45-2CSG324 in implementation of the design in Figure 4.30. According to [144], the propagation delay for one LUT is 0.26ns. Average LUT delay with its routing has been experimented as 0.28ns. INTCOMP and DLUT couple utilize 9 D-type flipflops and 1039 LUTs on the target FPGA. Due to the LUT-intense

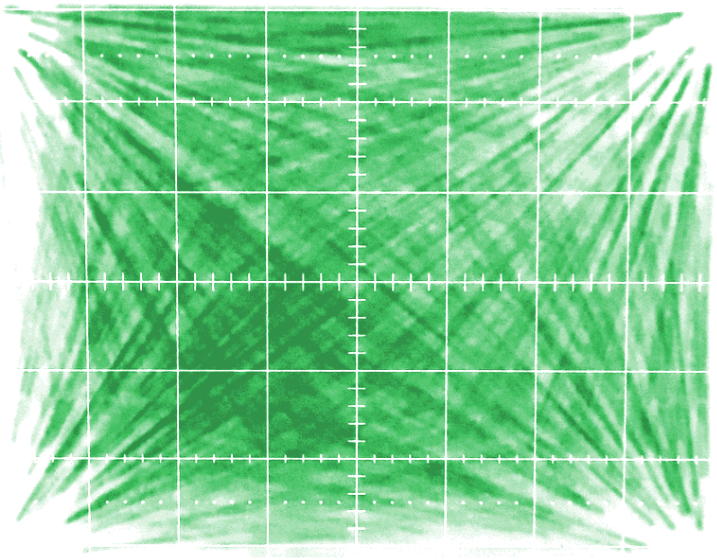


Figure 5.19 : The observed phase portrait of the two circuit system on $x_1(t) - x_2(t)$ state space.

implementation, FPGA may house 26 INTCOMP-DLUT couples without auxiliary blocks in Figure 4.30. The total consumption of test circuit is 284 D-type flipflops, 1757 LUTs, 960kb BlockRAM, 1 DSP48, and 3 BUFGs.

As expected, INTCOMP2 and DDFF2 generate a periodic signal. The period has been detected as 9608 CLK cycles by the signal analyzing process on computer. Moreover, the behavior of INTCOMP1 and DLUT1 seem to be random. INTCOMP1 visits the states which are on the periodic trajectory of INTCOMP2, but this happens in an aperiodic way. It is also observed that the aperiodic system's trajectory has states which are excluded by the periodic trajectory. Figure 5.20 depicts the periodic trajectory by INTCOMP2 in black and the aperiodic trajectory by INTCOMP1 in red. Black trajectory is plotted over the red one, but does not cover it, which means that the red trajectory visits the states which are unreached by the black one. Furthermore, aperiodic system performs a different trajectory at each measurement.

The periodic FT2 signal is subjected to EXOR operation, like the circuit in Figure 5.17, with periodic FT0 signal (using INTCOMP1 and DDFF0) and aperiodic FT1 signal (using INTCOMP1 and DLUT1). In both cases, 950272 bits (48k bits in Subsection 4.3.4) are stored in MEM after a slight modification of the MEM block. The stored bits are transferred to computer and NIST's SP800-22rev1a Statistical Test Suites is applied. Table 5.5 shows the statistical test result. All measurements from the

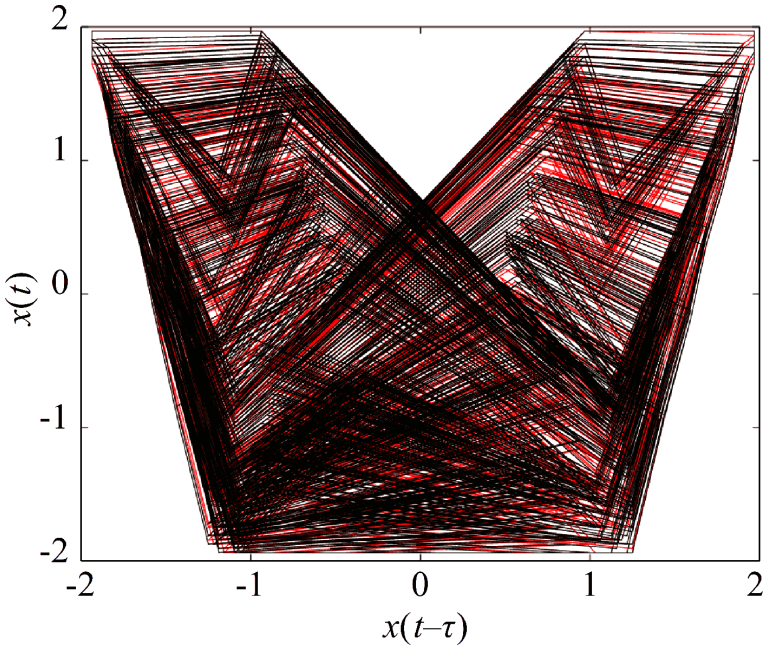


Figure 5.20 : Overlapping phase portraits. Black trajectory which belongs to the periodic system employing DDFF has been plotted over red trajectory which belongs to the aperiodic system employing DLUT.

aperiodic+periodic circuit configuration have passed all sub-test of the suite except the Universal test. However, two periodic circuits are not able to pass some test when their outputs are subjected to EXOR operation like the previous one. The proposed system provides 250kbps throughput which is statistically suitable for cryptographic applications. This rate is 5 times better than the one in Figure 5.17, and 12.5 times better than its single circuit version above.

As a result, digital circuit modeling a chaotic system has to be periodic, because its state space is finite. Employing propagation based delay lines in digital implementations contributes the complex behavior of time-delay chaotic systems. This subsection has demonstrated how a time-delay chaotic system turns into a periodic oscillator when it is digitalized, then how it escapes back to the aperiodic behavior, and the statistical properties of different implementations and configurations. The proposed circuits in Figure 2.15, and Figure 4.30 are proper for random bit generation according to the statistical tests. As a future work, the relation may be investigated between the unique physical properties of the target device and the trajectory of proposed system in order to realize a physically unclonable function.

Table 5.5 : NIST's SP800-22rev1a statistical test results. The effect of time-variant delay in DLUT (buffer chain delay line) is shown on the second column.

Statistical Name	Two DDFF Configuration	DDFF & DLUT Configuration
Frequency	(F) 33/50	47/50
BlockFrequency	49/50	50/50
CumulativeSums	(F) 31/50	47/50
Runs	(F) 45/50	50/50
LongestRun	50/50	50/50
Rank	49/50	50/50
FFT	50/50	50/50
NonOverlappingTemp.	(F) 0/50	50/50
OverlappingTemplate	50/50	50/50
ApproximateEntropy	(F) 46/50	47/50
Serial	49/50	50/50
LinearComplexity	50/50	49/50

5.2.2 Attack on true random bit generator

The circuit in Figure 4.33 generates binary signals of $f(x_1(t_k-\tau))$, $f(x_1(t_k))$, $f(x_2(t_k))$, and $f(x_3(t_k))$ with the similar relation $f(x_1(t_k)) = f(x_2(t_k-\tau)) = f(x_3(t_k-2\tau)) = f(x_4(t_k-3\tau))$ as explained in Subsection 4.3.6. In Figure 5.21, a snapshot of binary signals are plotted. But, small differences in real component values and behaviors of the subcircuits in Figure 4.33 and also the electrical interference generates some error in the bit sequences. Here, the proposed attack to the original system, generating random bits from $f(x_1)$ signal, occurs when the anticipated (predicted) future values of $f(x_1)$ is obtained with some error by coupled slave circuits.

These binary $f(x_i)$ signals are recorded and quantification of anticipating synchronization is given by the bit errors as shown in Figure 5.22. In Subsection 5.2.1, it is shown that the circuit in Figure 2.15 is capable of a random bit generation at 20000bps rate. For this reason, the signals depicted in Figure 5.21 is sampled at 20kbps. The strings with 10^6 bits, which are recorded and shifted proportional to the anticipation time, are compared. The Figure 5.22a shows the histogram of the erroneous bits between $f(x_2(t_p-\tau))$ and $f(x_1(t_p))$. Here, t_p is the p-th sampling time of the binary signal with 20kHz sampling frequency. To calculate the histogram the string is split into 1000 substrings with 1000 bits. Similar to Figure 5.22a, Figure 5.22b shows the histogram of the erroneous bits between $f(x_3(t_p-\tau))$ and $f(x_2(t_p))$. And, Figure 5.22c

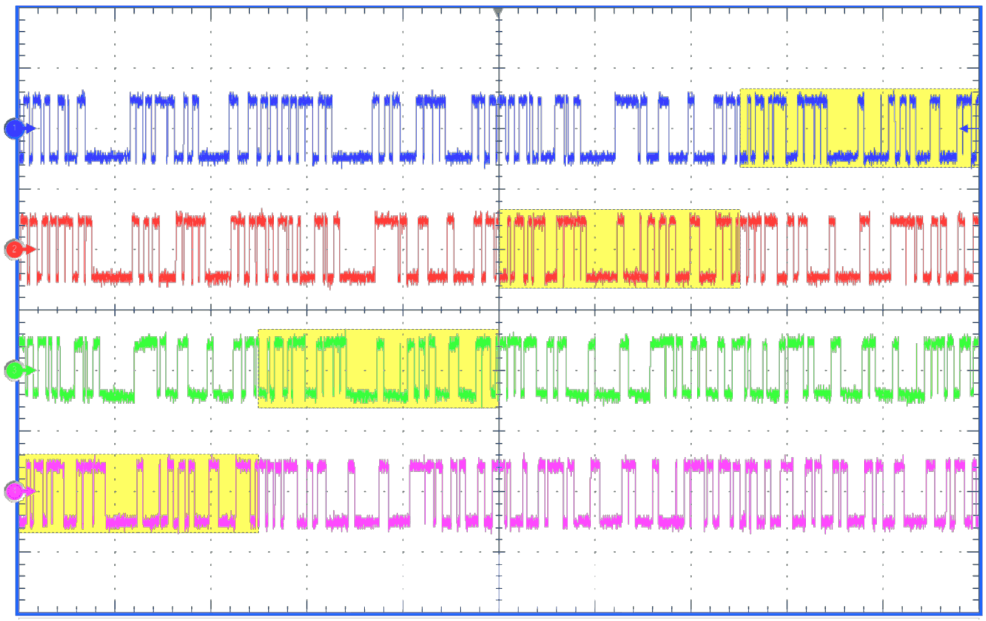
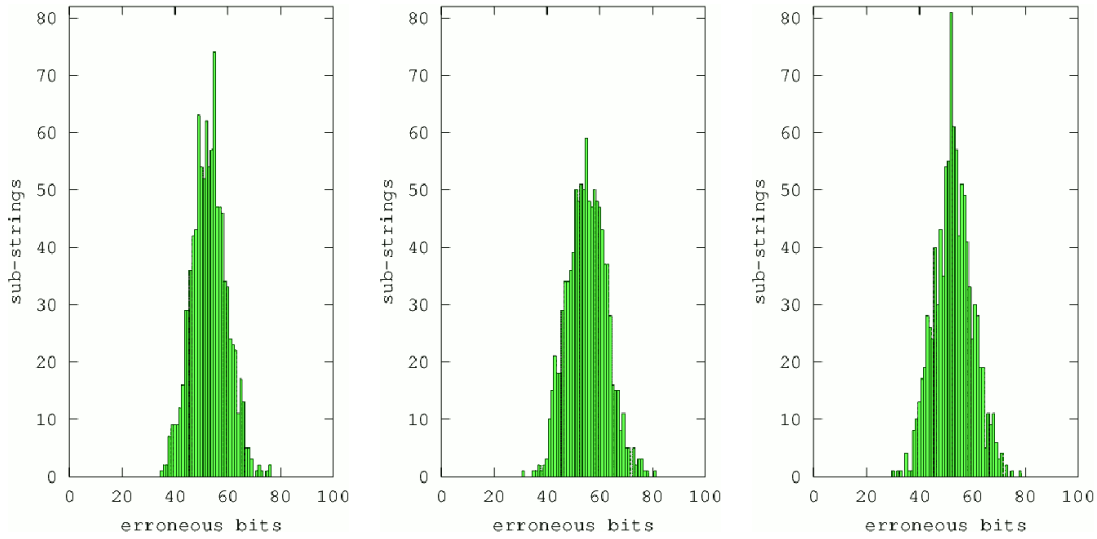


Figure 5.21 : Similar to x signals, digital oscilloscope screen captures depicting $f(x_1(t_k - \tau))$ (blue), $f(x_1(t_k))$ (red), $f(x_2(t_k))$ (green) and $f(x_3(t_k))$ (purple) synchronously. The yellow box is shifted τ second left at each channel and used for focusing a short-time signal record that is seen on every system's output.

belongs to the erroneous bits between $f(x_4(t_p - \tau))$ and $f(x_3(t_p))$. The mean value of the erroneous bits proportion to the substring length is approximately 5% which is reasonable because the analog part of the circuit has component with a tolerance of 5%. Hence, it can be claimed that the attack to the master system's bit sequence by anticipating synchronization achieves 95% success τ before, 90% success 2τ before, and a 85% success 3τ before the sequence occurrence. The error rate increases linearly according to the prediction length.

Experimental verification of the attack to a TRBG using anticipating synchronization has been presented in this subsection. The quantification of synchronization, given by the bit error between the bit streams generated by the original and coupled systems, has showed that the cumulative error increases 5% at each τ step. Thus, the attack on a time-delay sampled-data chaos-based RNG using anticipating synchronization seems to be successful taking into account the tolerances of the elements in the non-ideal implementations.



(a) (b) (c)

Figure 5.22 : Distribution of erroneous bits between strings of $f(x_2(t_p - \tau))$ and $f(x_1(t_p))$ in a, $f(x_3(t_p - \tau))$ and $f(x_2(t_p))$ in b, $f(x_4(t_p - \tau))$ and $f(x_3(t_p))$ in c. String length is 1000 bits.

6. CONCLUSION

This thesis is concluded with the following two sections: First, the results obtained from the studies are exposed in Obtained Results section. The second section, Open Research Fields, suggests future works inspired by the thesis.

6.1 Obtained Results

Instead of the section order, which is cells, networks, implementations and applications, the hypotheses order is preferred in this section to report the results.

Thesis covers a novel relaxation oscillator model, Oscillator 2 given in Section 2.1.2. This model employs only one signum function as nonlinearity, which can be electronically implemented by a simple analog voltage comparator. Yalcin's model (Oscillator 1) had been the simplest relaxation oscillator model since 2008. However, Oscillator 2 outperforms Oscillator 1 from the implementation viewpoint.

The simplest oscillating logic circuit includes an inverter and an artificial delay component which determines the oscillation period. This logic oscillator is constructed by the closed-loop circuit of these two components. For synchronous sequential circuits, required components are one flip-flop and a NOT gate, which yields a 2-state oscillator. If one mimic the relaxation oscillation in another circuit, at least four states have to be defined, two for peaks and two for transitions. Thus, a simple 4-state logic oscillator with a control input (Oscillator 3) is studied to construct Cellular Logic Networks in this thesis.

Thesis evolves Oscillator 1 to Network 1 with known coupling scheme and parameters. Yet, Network 2 has a new different coupling scheme which occurs in the arguments of the nonlinear function. Although Network 1 and Network 2 need change only in parameter values in order to propagate autowave or traveling wave, Network 3 need to change the coupling scheme. Network 3 with these two coupling schemes is a novel proposal.

All three networks are capable of propagating nested traveling waves. Moreover, the location of traveling wave is controlled by the location of the cell where the input signal is applied. In this case, Doppler Effect emerges on the traveling wave-fronts. Doppler Effect, which is the frequency modulation of wave sources velocity, is useful as the source motion information is propagated to entire network by wave-fronts. Cells on the networks have a new feature extraction duty which is carried out measuring the temporal wave period. Hence, this thesis utilizes Doppler Effect, in making a prediction for the wave source's future state (position). According to the results in Section 5.1.2, the optimum solution is employing Network 2, in novel predictive feedback motion planning algorithm, which is also proposed by this thesis. Although, obtained simplicity in cell dynamics loses the curvature quality of the propagated wavefront, the wave propagation speed on Network 2 is higher than Network 1. Furthermore, as cell's settling to a stationary state on Network 2 is faster than the ones on Network 1. Actually, Network 3 is the fastest one. But, it has a tetragonal wave-front pattern which causes to generate undesired feedback plans. All three networks, which reveal the Doppler Effect, support the hypothesis as they successfully predict the future positions of the target in motion planning test.

The second hypothesis, generating random bit sequences is successfully carried out by a time-delay chaotic system with a binary feedback function. This feedback function, enables using logic components on their binary delay lines, which simplifies the implementation of the systems. Moreover, mono-scroll chaotic attractor by the binary output nonlinearity has been generalized by a quantized output nonlinearity. This generalization which yields a multi-scroll chaotic attractor is one of the novel contributions of the thesis. An asynchronous delay circuit, for delay line composition is first proposed by this thesis. This circuit is able to double the delay amount acquired from a delay line. Thus, area efficient delay line implementation has successfully demonstrated. Both flip-flops and logic NOT gates are employed in the implementation of the time-delay chaotic circuits. A sampled-data model for the system has been realized by analog integrator and comparator circuits accompanying to flip-flop based delay line. The sampled-data modeling approach is also one of the contributions of the thesis.

The bit sequences generated by implemented time-delay chaotic systems has been analyzed using up-to-date NIST's Statistical Test Suite. The output of the binary feedback line is directly used for capturing the candidate bits. Tests prove that system provides true random bits proper to cryptographic applications. On the other hand, an attack to TRBG has also been achieved using anticipating synchronization. Thesis includes one throughput enhancement method and an all-digital implementation on a single FPGA. It has been presented that even all-digital implementation is successful in true random bit generation if its dynamics is supported by jitter.

6.2 Publications on The Thesis

Implementation of Network 1 on GPU has been presented in [2] and on an FPGA using fixed-point arithmetic modules and presented in [1]. Dynamical partial reconfiguration of logic network (Network 3) has been presented in [72, 145].

Two conference papers have been presented till now about the Doppler Effect observation on nonlinear waves propagated by Network 1 [70, 71]. Network 2, systematic construction of Network 3, predictive feedback motion planning algorithm and result gained by employing Network $1-3$ have been submitted to [146].

Oscillator 4, which is the time-delay sampled-data chaotic oscillator has been investigated and in [101]. True random bit generation using Oscillator 4 has been exhibited in [6]. In [147], throughput enhancement using two Oscillator 4 has been achieved. Oscillator 5 generating multi-scroll chaotic attractor has been recently submitted to [148]. An asynchronous digital circuit which provide area efficiency in implementing binary delay line has is now under revision [149].

Network structure, implementation and application of Network 4, which causes anticipating synchronization between time-delay chaotic cells, have been published in [150]. Anticipating synchronization phenomenon in Network 4 has been implemented using digital approximation of Oscillator 4 and presented in [151]. The book chapter [152] includes the details for Oscillator 4, its TRBG application and throughput enhancement circuit.

6.3 Open Research Fields

Time-variant systems (especially varying coupling weights) can be investigated in future works of relaxation oscillator based CNNs. Using unidirectional coupling, moving obstacles without reinitialization may be possible. The boundaries of the obstacle may evolve under the coupling with active cells. On the other side, the active cells still obey the zero-flux boundary condition and are not effected by the obstacle. When the obstacle is moved by 1 cell, the new active cells will be ready to oscillate along with the currently active ones.

Proposed predictive motion planning technique in real-world robotics application may need to study adaptive control of tracker speed, which provides the energy-efficiency in a robot's motion. As well as, wave computer implementations, a unified digital system with relaxation based CNNs and feedback motion planner can be designed and implemented on a single FPGA, then integrated in robots.

The curvature of the propagated wave-front effects the paths generated by motion planner. Hence, the state number of logic oscillator can be increased. Then, the high propagation speed may be supported with octagonal or circular wave-front a proper coupling scheme and a new digital oscillation dynamics. This solution may outperform the best results acquired from Network 2.

Reconfigurable configuration studies may focus on single-cell reconfigurability. Also, reconfiguration overhead needs to be decreased. Developing FPGAs with strong support for dynamic reconfiguration and developing new methodologies and tools for dynamic reconfiguration are also open research fields.

A possible cellular network composed of time-delay chaotic cells (Oscillator 4 and 5) may exhibit complex synchronization phenomena. These kinds of networks may be studied in order to solve problems via synchronization. Defining the problem set that can be solved by this kind of network is an open research field.

The proposed Asynchronous Delay Doubler, in this thesis, is open to develop. It functions based on an assumption which defines the minimum pulse width which can be delayed. A complex asynchronous state machine may record the positive edge or negative edge received on the input and than delay them more than one times. This state machine should track the events received, events delayed, and events sent to

output. This approach may enhance the minimum pulse width constraint and may provide more area efficiency.

REFERENCES

[1] **Karakaya, B.**, **Yeniceri, R. and Yalcin, M.E.** (2015). Wave computer core using fixed-point arithmetic, *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pp.1514–1517.

[2] **Tukel, M.**, **Yeniceri, R. and Yalcin, M.E.** (2012). Nonlinear spatio-temporal wave computing for real-time applications on GPU, *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*, pp.1–5.

[3] **Yeniceri, R. and Yalcin, M.E.** (2009). An emulated digital wave computer core implementation, *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pp.831–834.

[4] **Yeniceri, R.**, (2009). Yol Bulma Uygulamaları için Bir Hücresel Yapay Sinir Ağının Sayısal Tasarımı ve Gerçeklenmesi, Master's thesis, Istanbul Technical University.

[5] **Khronos Group**, (2012), The open standard for parallel programming of heterogeneous systems, `http://www.khronos.org/opencl/`, date retrieved 02.08.2015.

[6] **Yeniceri, R. and Yalcin, M.E.** (2013). True random bit generation with time-delay sampled-data feedback system, *Electronics Letters*, ***49***(8), 543–545.

[7] **Balsi, M.**, **Marongiu, A. and V., C.** (1995). Electromagnetic Field Simulation using 3D Cellular Neural Networks, *Proc. of 1995 European Conference on Circuit Theory and Design (ECCTD95)*, pp.987–990.

[8] **Roska, T.**, **Chua, L.O.**, **Wolf, D.**, **Kozek, T.**, **Tetzlaff, R. and Puffer, F.** (1995). Simulating nonlinear waves and partial differential equations via CNN. I. Basic techniques, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***42***(10), 807–815.

[9] **Chua, L.O. and Yang, L.** (1988). Cellular neural networks: theory, *Circuits and Systems, IEEE Transactions on*, ***35***(10), 1257–1272.

[10] **Chua, L.O. and Yang, L.** (1988). Cellular neural networks: applications, *Circuits and Systems, IEEE Transactions on*, ***35***(10), 1273–1290.

[11] **Wolfram, S.** (1896). *Theory and Applications of Cellular Automata*, volume 1 of *Advanced Series on Complex Systems*, World Scientific.

[12] **Rekeczky, C.**, **Szatmari, I.**, **Foldesy, P. and Roska, T.** (2002). Analogic cellular PDE machines, *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 3, pp.2033–2038.

[13] **Roska, T. and Chua, L.O.** (1993). The CNN universal machine: an analogic array computer, *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, ***40***(3), 163–173.

[14] **Liñán, G.**, **Espejo, S.**, **Domínguez-Castro, R. and Rodríguez-Vázquez, A.** (2002). ACE4k: An analog I/O 64×64 visual microprocessor chip with 7-bit analog accuracy, *International Journal of Circuit Theory and Applications*, ***30***(2-3), 89–116.

[15] **Rodriguez-Vazquez, A.**, **Linan-Cembrano, G.**, **Carranza, L.**, **Roca-Moreno, E.**, **Carmona-Galan, R.**, **Jimenez-Garrido, F.**, **Dominguez-Castro, R. and Meana, S.E.** (2004). ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***51***(5), 851–863.

[16] **Nagy, Z. and Szolgay, P.** (2003). Configurable multilayer CNN-UM emulator on FPGA, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***50***(6), 774–778.

[17] **Malki, S. and Spaanenburg, L.** (2004). On the packet-switched implementation of a discrete-time CNN, *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*, pp.234–241.

[18] **Yildiz, N.**, **Cesur, E. and Tavsanoglu, V.** (2010). A new control structure for the pipelined CNN processor arrays, *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, pp.1–4.

[19] **Mackey, M.C. and Glass, L.** (1977). Oscillation and chaos in physiological control systems, *Science*, ***197***(4300), 287–289.

[20] **Namajunas, A.**, **Pyragas, K. and Tamasevicius, A.** (1995). An electronic analog of the Mackey-Glass system, *Physics Letters A*, ***201***(1), 42 – 46.

[21] **Lu, H. and He, Z.** (1996). Chaotic behavior in first-order autonomous continuous-time systems with delay, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***43***(8), 700–702.

[22] **Lu, H.**, **He, Y. and He, Z.** (1998). A chaos-generator: analyses of complex dynamics of a cell equation in delayed cellular neural networks, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***45***(2), 178–181.

[23] **Tamasevicius, A.**, **Mykolaitis, G. and Bumeliene, S.** (2006). Delayed feedback chaotic oscillator with improved spectral characteristics, *Electronics Letters*, ***42***(13), 736–737.

[24] **Buscarino, A.**, **Fortuna, L.**, **Frasca, M. and Sciuto, G.** (2011). Design of Time-Delay Chaotic Electronic Circuits, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***58***(8), 1888–1896.

[25] **Pham, V.T.**, **Fortuna, L. and Frasca, M.** (2012). Implementation of chaotic circuits with a digital time-delay block, *Nonlinear Dynamics*, ***67***(1), 345–355.

[26] **Yalcin, M.E.** (2008). A Simple Programmable Autowave Generator Network for Wave Computing Applications, *Circuits and Systems II: Express Briefs, IEEE Transactions on*, ***55***(11), 1173–1177.

[27] **Wang, D.**, (1999). Relaxation Oscillators and Networks, volume 18 of *Encyclopedia of electrical and electronic engineers*, chapter 1, John Wiley & Sons, Inc., pp.396–405.

[28] **van der Pol, B.** (1940). Biological Rhythms Considered as Relaxation Oscillations, *Acta Medica Scandinavica*, ***103***(S108), 76–88.

[29] **Yeniceri, R. and Yalcin, M.E.** (2008). An implementation of 2D locally coupled relaxation oscillators on an FPGA for real-time autowave generation, *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, pp.29–33.

[30] **Yeniceri, R. and Yalcin, M.E.** (2009). Path planning on cellular nonlinear network using active wave computing technique, *Bioengineered and Bioinspired Systems IV*, ***7365***(1), 736508.

[31] **an der Heiden, U. and Mackey, M.C.** (1982). The Dynamics of production and destruction: Analytic insight into complex behavior, *J. Math. Biology*, ***16***, 75–101.

[32] **Losson, J. and Mackey, M.C.** (1993). Solution multistability in first-order nonlinear differential delay equations., *Chaos*, ***3***(2), 167.

[33] **Lu, J.G. and Hill, D.J.** (2008). Global asymptotical synchronization of chaotic Lur'e systems using sampled data: a linear matrix inequality approach, *Circuits and Systems II: Express Briefs, IEEE Transactions on*, ***55***(6), 586–590.

[34] **Chen, W.H.**, **Wang, Z. and Lu, X.** (2012). On Sampled-Data Control for Master-Slave Synchronization of Chaotic Lur'e Systems, *Circuits and Systems II: Express Briefs, IEEE Transactions on*, ***59***(8), 515–519.

[35] **Barajas-Ramírez, J.G.**, **Chen, G. and Shieh, L.S.** (2003). Hybrid Chaos Synchronization, *International Journal of Bifurcation and Chaos*, ***13***(05), 1197–1216.

[36] **Wagemakers, A.**, **Buldú, J.M. and Sanjuán, M.A.F.** (2008). Experimental demonstration of bidirectional chaotic communication by means of isochronal synchronization, *EPL (Europhysics Letters)*, ***81***(4), 40005.

[37] **Márquez, B.A.**, **Suárez-Vargas, J.J. and Ramírez, J.A.** (2014). Polynomial law for controlling the generation of n-scroll chaotic attractors in an optoelectronic delayed oscillator, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, ***24***(3), 033123.

[38] **Farmer, J.D.** (1982). Chaotic attractors of an infinite-dimensional dynamical system, *Physica D: Nonlinear Phenomena*, ***4***(3), 366–393.

[39] **Astrom, K. and Wittenmark, B.** (1984). *Computer Controlled Systems: Theory and Design*, Prentice-Hall, N.J.

[40] **Chua, L.O.** (1994). Chua's circuit 10 years later, *International Journal of Circuit Theory and Applications*, ***22***(4), 279–305.

[41] **Suykens, J. and Vandewalle, J.** (1993). Generation of n-double scrolls (n=1, 2, 3, 4, ...), *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***40***(11), 861–867.

[42] **Chen, D.**, **Sun, Z.**, **Ma, X. and Chen, L.** (2014). Circuit implementation and model of a new multi-scroll chaotic system, *International Journal of Circuit Theory and Applications*, ***42***(4), 407–424.

[43] **Elwakil, A. and Kennedy, M.** (2001). Construction of classes of circuit-independent chaotic oscillators using passive-only nonlinear devices, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***48***(3), 289–307.

[44] **Yalcin, M.**, **Ozoguz, S.**, **Suykens, J.A.K. and Vandewalle, J.** (2001). n-scroll chaos generators: a simple circuit model, *Electronics Letters*, ***37***(3), 147–148.

[45] **Yalcin, M.E.**, **Suykens, J.A.**, **Vandewalle, J. and Ozoguz, S.** (2002). Families of scroll grid attractors, *International Journal of Bifurcation and Chaos*, ***12***(01), 23–41.

[46] **Lü, J.**, **Han, F.**, **Yu, X. and Chen, G.** (2004). Generating 3-D multi-scroll chaotic attractors: A hysteresis series switching method, *Automatica*, ***40***(10), 1677 – 1687.

[47] **Lu, J.**, **Chen, G.**, **Yu, X. and Leung, H.** (2004). Design and analysis of multiscroll chaotic attractors from saturated function series, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***51***(12), 2476–2490.

[48] **Deng, W. and Lü, J.** (2006). Design of multidirectional multiscroll chaotic attractors based on fractional differential systems via switching control., *Chaos (Woodbury, N.Y.)*, ***16***(4), 043120.

[49] **Elwakil, A. and Ozoguz, S.** (2006). Multiscroll Chaotic Oscillators: The Nonautonomous Approach, *Circuits and Systems II: Express Briefs, IEEE Transactions on*, ***53***(9), 862–866.

[50] **Trejo-Guerra, R.**, **Tlelo-Cuautle, E.**, **Jiménez-Fuentes, M.**, **Muñoz-Pacheco, J.M. and Sánchez-López, C.** (2013). Multiscroll floating gate–based integrated chaotic oscillator, *International Journal of Circuit Theory and Applications*, ***41***(8), 831–843.

[51] **Yu, S.**, **Tang, W.K.S.**, **Lü, J. and Chen, G.** (2010). Generating 2n-wing attractors from Lorenz-like systems, *International Journal of Circuit Theory and Applications*, ***38***(3), 243–258.

[52] **Wang, L. and Yang, X.** (2006). Generation of multi-scroll delayed chaotic oscillator, *Electronics Letters*, ***42***(25), 1439–1441.

[53] **Yalcin, M.E. and Ozoguz, S.** (2007). n-scroll chaotic attractors from a first-order time-delay differential equation, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, ***17***(3), 033112.

[54] **Kilinc, S.**, **Yalcin, M.E. and Ozoguz, S.** (2010). Multiscroll Chaotic Attractors From A Hysteresis Based Time-Delay Differential Equation, *International Journal of Bifurcation and Chaos*, ***20***(10), 3275–3281.

[55] **Srinivasan, K.**, **Raja Mohamed, I.**, **Murali, K.**, **Lakshmanan, M. and Sinha, S.** (2011). Design of time delayed chaotic circuit with threshold controller, *International Journal of Bifurcation and Chaos*, ***21***(03), 725–735.

[56] **Liu, X.**, **(Sherman) Shen, X. and Zhang, H.** (2012). Multi-scroll Chaotic and Hyperchaotic Attractors Generated From Chen System, *International Journal of Bifurcation and Chaos*, ***22***(02), 1250033.

[57] **Sprott, J.** (2007). A simple chaotic delay differential equation, *Physics Letters A*, ***366***(4-5), 397 – 402.

[58] **Zhang, H.**, **Liu, X.**, **Shen, X. and Liu, J.** (2012). A family of novel chaotic and hyperchaotic attractors from delay differential equation, *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, ***19***(3), 411–430.

[59] **Sandri, M.** (1996). Numerical calculation of Lyapunov exponents, *Mathematica Journal*, ***6***(3), 78–84.

[60] **Sprott, J.C.** (2003). *Chaos and time-series analysis*, Oxford University Press, Oxford, New York.

[61] **Benettin, G.**, **Galgani, L.**, **Giorgilli, A. and Strelcyn, J.M.** (1980). Lyapunov Characteristic Exponents for smooth dynamical systems and for hamiltonian systems; A method for computing all of them. Part 2: Numerical application, *Meccanica*, ***15***(1), 21–30.

[62] **Stefański, A. and Kapitaniak, T.** (2003). Estimation of the dominant Lyapunov exponent of non-smooth systems on the basis of maps synchronization, *Chaos, Solitons and Fractals*, ***15***(2), 233–244, cited By 34.

[63] **Cesur, E.**, **Yildiz, N. and Tavsanoglu, V.** (2010). Architecture of The Next Generation Real Time CNN Processor RTCNNP-v2, *International Symposium on Nonlinear Theory and its Applications (NOLTA'2010)*, Krakow, Poland, pp.1–4.

[64] **Ito, K.**, **Hiratsuka, M.**, **Aoki, T. and Higuchi, T.** (2006). A Shortest Path Search Algorithm Using an Excitable Digital Reaction-Diffusion System, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, ***E89-A***(3), 735–743.

[65] **Adamatzky, A.**, **Arena, P.**, **Basile, A.**, **Carmona-Galan, R.**, **Costello, B.D.L.**, **Fortuna, L.**, **Frasca, M. and Rodriguez-Vazquez, A.** (2004). Reaction-diffusion navigation robot control: from chemical to VLSI analogic processors, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***51***(5), 926–938.

[66] **Lopich, A. and Dudek, P.** (2011). Asynchronous cellular logic network as a co-processor for a general-purpose massively parallel array, *International Journal of Circuit Theory and Applications*, ***39***(9), 963–972.

[67] **Rekeczky, C. and Chua, L.O.** (1999). Computing with Front Propagation: Active Contour And Skeleton Models In Continuous-Time CNN, *J. VLSI Signal Process. Syst.*, ***23***(2/3), 373–402.

[68] **Munuzuri, A.**, **Perezmunuzuri, V.**, **Gomezgesteira, M.**, **Chua, L. and Perezvillar, V.** (1995). Spatiotemporal Structures in Discretely-coupled Arrays of Nonlinear Circuits: A Review, *International Journal of BIifurcation and Chaos*, ***5***(1), 17–50.

[69] **Yalcin, M.E. and Suykens, J.A.K.** (2006). Spatiotemporal pattern formation in the ACE16k CNN chip, *International Journal of Bifurcation and Chaos*, ***16***(05), 1537–1546.

[70] **Yeniceri, R. and Yalcin, M.E.** (2012). A new CNN based path planning algorithm improved by the Doppler Effect, *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*, pp.1–5.

[71] **Yeniceri, R. and Yalcin, M.E.** (2013). The Doppler effect with input driven autowaves, *Circuit Theory and Design (ECCTD), 2013 European Conference on*, pp.1–4.

[72] **Yeniceri, R.**, **Abtioglu, E.**, **Govem, B. and Yalcin, M.E.** (2014). A 16×16 Cellular Logical Network with partial reconfiguration feature, *Cellular Nanoscale Networks and their Applications (CNNA), 2014 14th International Workshop on*, pp.1–2.

[73] **Boccaletti, S.**, **Kurths, J.**, **Osipov, G.**, **Valladares, D.L. and Zhou, C.S.** (2002). The synchronization of chaotic systems, *Physics Reports*, ***366***(1-2), 1–101.

[74] **Pecora, L. and Carroll, T.** (1990). Synchronization in chaotic systems, *Physical Review Letters*, ***64***(8), 821–824.

[75] **Yalcin, M.**, **Suykens, J. and Vandewalle, J.** (2005). *Cellular Neural Networks, Multi-Scroll Chaos And Synchronization*, volume 50 of *World Scientific Series on Nonlinear Science, Series A*, World Scientific.

[76] **Voss, H.U.** (2000). Anticipating chaotic synchronization, *Phys. Rev. E*, ***61***(5), 5115–5119.

[77] **Huijberts, H.**, **Nijmeijer, H. and Oguchi, T.** (2007). Anticipating synchronization of chaotic Lur'e systems, *Chaos: An Interdisciplinary Journal of Nonlinear Science*, ***17***(1), 013117.

[78] **Kilinc, S.**, **Yalcin, M.E. and Ozoguz, S.** (2008). Synchronization of first-order time-delay systems generating n-scroll chaotic attractors, *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp.756–759.

[79] **Ergunay, S.**, **Yeniceri, R. and Yalcin, M.E.** (2010). Hardware-Software Co-design of Nonlinear Active Wave Generator with Microblaze Soft Core Processor, *Nonlinear Theory and its Applications, 2010. NOLTA 2010. International Symposium on*, pp.157–160.

[80] **Zarandy, A.**, **Keresztes, P.**, **Roska, T. and Szolgay, P.** (1998). CASTLE: an emulated digital CNN architecture design issues, new results, *Electronics, Circuits and Systems, 1998 IEEE International Conference on*, volume 1, pp.199–202.

[81] **Malki, S. and Spaanenburg, L.** (2003). CNN Image Processing on a Xilinx Virtex-II 6000, *Proceedings of the 16th European Conference on Circuit Theory and Design, ECCTD'03*, pp.261–264.

[82] **Fang, W.**, **Wang, C. and Spaanenburg, L.** (2006). In search for a robust digital CNN system, ***V. Tavsanoglu and S. Arik****, editors, Proceedings of the 2006 10th IEEE International Workshop on Cellular Neural Networks and Their Applications*, IEEE, 345 E 47TH ST, NEW YORK, NY 10017 USA, pp.328–333, 10th IEEE International Workshop on Cellular Neural Networks and Their Applications, Istanbul, TURKEY, AUG 28-30, 2006.

[83] **Nagy, Z.**, **Vörösházi, Z. and Szolgay, P.** (2006). Emulated digital CNN-UM solution of partial differential equations, *International Journal of Circuit Theory and Applications*, ***34***(4), 445–470.

[84] **Chua, L.O. and Roska, T.** (1993). The CNN paradigm, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***40***(3), 147–156.

[85] **Soos, B.**, **Rak, A.**, **Veres, J. and Cserey, G.** (2008). GPU powered CNN simulator (SIMCNN) with graphical flow based programmability, *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, pp.163–168.

[86] **Dolan, R. and DeSouza, G.** (2009). GPU-based simulation of cellular neural networks for image processing, *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pp.730–735.

[87] **Nvidia Coorp.**, TESLA High Performance Computing, `http://www.nvidia.com/object/tesla-supercomputing-solutions.html`, date retrieved 02.08.2015.

[88] **Nvidia Coorp.**, Compute Unified Device Architecture (CUDA), `https://developer.nvidia.com/cuda-zone`, date retrieved 02.08.2015.

[89] **Advanced Micro Devices, Inc.**, APP SDK, A Complete Development Platform, `http://developer.`

amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/, date retrieved 02.08.2015.

[90] **The MathWorks, Inc.**, Matlab GPU computing support for Nvidia CUDA-enabled GPUs, http://www.mathworks.com/discovery/matlab-gpu.html?BB=1, date retrieved 02.08.2015.

[91] **Sato, D.**, **Xie, Y.**, **Weiss, J.**, **Qu, Z.**, **Garfinkel, A. and Sanderson, A.** (2009). Acceleration of cardiac tissue simulation with graphic processing units, *Medical and Biological Engineering and Computing*, ***47***, 1011–1015, 10.1007/s11517-009-0514-4.

[92] **Kilic, V. and Yalcin, M.E.** (2011). An active wave computing based path finding approach for 3-D environment, *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pp.2165–2168.

[93] **Kilic, V.**, **Yeniceri, R. and Yalcin, M.E.** (2010). A new active wave computing based real time mobile robot navigation algorithm for dynamic environment, *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, pp.1–6.

[94] **Becker, J.**, **Hubner, M.**, **Hettich, G.**, **Constapel, R.**, **Eisenmann, J. and Luka, J.** (2007). Dynamic and Partial FPGA Exploitation, *Proceedings of the IEEE*, ***95***(2), 438–452.

[95] **Lindberg, E.**, **Mykolaitis, G.**, **Bumelien, S.**, **Pyragien, T.**, **Tamaseviius, A.**, **Tamaseviit, E. and Kirvaitis, R.** (2010). Higher-order chaotic oscillator using active bessel filter, *Journal of Circuits, Systems and Computers*, ***19***(4), 859–869.

[96] **Mykolaitisa, G.**, **Tamaseviciusa, A.**, **Cenysa, A.**, **Bumelien, S.**, **Anagnostopoulosb, A.N. and Kalkanc, N.** (2003). Very high and ultrahigh frequency hyperchaotic oscillators with delay line, *Chaos, Solitions & Fractals*, ***17***(2-3), 343–347.

[97] **Razmdideh, R. and Saneei, M.** (2014). Two novel low power and very high speed pulse triggered flip-flops, *International Journal of Circuit Theory and Applications*, n/a–n/a.

[98] **Schell, B. and Tsividis, Y.** (2008). A Low Power Tunable Delay Element Suitable for Asynchronous Delays of Burst Information, *Solid-State Circuits, IEEE Journal of*, ***43***(5), 1227–1234.

[99] **Chang, H.H. and Liu, S.I.** (2005). A wide-range and fast-locking all-digital cycle-controlled delay-locked loop, *Solid-State Circuits, IEEE Journal of*, ***40***(3), 661–670.

[100] **Toumazou, C.**, **Lidgey, F.J. and Haigh, D.** (1990). *Analogue IC design: the current-mode approach*, Peter Peregrinus Ltd.

[101] **Yalcin, M.**, **Yeniceri, R. and Ozoguz, S.** (2014). A chaotic time-delay sampled-data system and its implementation, *International Journal of Bifurcation and Chaos*, ***24***(3), 1450039.

[102] **Mazin, M. and Engeler, W.**, (1984), CMOS latch cell including five transistors, and static flip-flops employing the cell, uS Patent 4,484,087.

[103] **Roska, T.** (2007). Circuits, computers, and beyond Boolean logic, *International Journal of Circuit Theory and Applications*, ***35***(5-6), 485–496.

[104] **Xing, N.**, **Shin, W.Y.**, **Jeong, D.K. and Kim, S.** (2010). High-resolution time-to-digital converter utilising fractional difference conversion scheme, *Electronics Letters*, ***46***(6), 398–400.

[105] **Latombe, J.C.** (1999). Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts, *The International Journal of Robotics Research*, ***18***(11), 1119–1128.

[106] **LaValle, S.** (2011). Motion Planning, *Robotics Automation Magazine, IEEE*, ***18***(1), 79–89.

[107] **Koyuncu, E.**, **Ure, N. and Inalhan, G.** (2010). Integration of Path/Maneuver Planning in Complex Environments for Agile Maneuvering UCAVs, *Journal of Intelligent and Robotic Systems*, ***57***(1-4), 143–170.

[108] **LaValle, S.M.** (2006). *Planning Algorithms*, Cambridge University Press, New York, NY, USA, 1 edition.

[109] **Cormen, T.H.**, **Leiserson, C.E.**, **Rivest, R.L. and Stein, C.** (2009). *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 3 edition.

[110] **Roska, T.** (2007). Cellular wave computers for nanotera- scale technology–-beyond Boolean, spatial-temporal logic in million processor devices, *Electronics Letters*, ***43***(8), 427–429.

[111] **Atay, N. and Bayazit, B.** (2006). A motion planning processor on reconfigurable hardware, *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp.125–132.

[112] **Chua, L.O.**, **Hasler, M.**, **Moschytz, G.S. and Neirynck, J.** (1995). Autonomous cellular neural networks: a unified paradigm for pattern formation and active wave propagation, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***42***(10), 559–577.

[113] **Perez-Munuzuri, V.**, **Perez-Villar, V. and Chua, L.O.** (1993). Autowaves for image processing on a two-dimensional CNN array of excitable nonlinear circuits: flat and wrinkled labyrinths, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***40***(3), 174–181.

[114] **Gacsadi, A.**, **Maghiar, T. and Tiponut, V.** (2002). A CNN path planning for a mobile robot in an environment with obstacles, *Cellular Neural Networks and Their Applications, 2002. (CNNA 2002). Proceedings of the 2002 7th IEEE International Workshop on*, pp.188–194.

[115] **Gavrilut, I.**, **Tiponut, V.**, **Gacsadi, A. and Grava, C.** (2007). CNN Processing Techniques for Multi-robot Coordination, *Signals, Circuits and Systems, 2007. ISSCS 2007. International Symposium on*, pp.1–4.

[116] **Munuzuri, A.P. and Vazquez-Otero, A.** (2008). The CNN solution to the shortest-path-finder problem, *Cellular Neural Networks and Their Applications, 2008. CNNA 2008. 11th International Workshop on*, pp.248–251.

[117] **Vazquez-Otero, A. and Munuzuri, A.P.** (2010). Navigation algorithm for autonomous devices based on biological waves, *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, pp.1–5.

[118] **Adamatzky, A. and Costello, B.D.** (2003). Reaction-diffusion path planning in a hybrid chemical and cellular-automaton processor, *Chaos Solitons & Fractals*, ***16***(5), 727–736.

[119] **Dudek, P.** (2006). An asynchronous cellular logic network for trigger-wave image processing on fine-grain massively parallel arrays, *Circuits and Systems II: Express Briefs, IEEE Transactions on*, ***53***(5), 354–358.

[120] **Tzionas, P.G.**, **Thanailakis, A. and Tsalides, P.G.** (1997). Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata, *Robotics and Automation, IEEE Transactions on*, ***13***(2), 237–250.

[121] **Ahmed, S.**, **Akhter, A. and Kunwar, F.** (2012). Cellular automata based real time path planning for mobile robots, *Control Automation Robotics Vision (ICARCV), 2012 12th International Conference on*, pp.142–147.

[122] **Willms, A.R. and Yang, S.X.** (2008). Real-Time Robot Path Planning via a Distance-Propagating Dynamic System with Obstacle Clearance, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, ***38***(3), 884–893.

[123] **Cao, Y.**, **Zhou, X.**, **Li, S.**, **Zhang, F.**, **Wu, X.**, **Li, A. and Sun, L.** (2010). Design of path planning based Cellular Neural Network, *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pp.6539–6544.

[124] **Eden, A.** (1992). *The Search for Christian Doppler*, Springer-Verlag Wien, 1 edition.

[125] **Schetzen, M.** (2006). *Airborne Doppler Radar : Applications, Theory, and Philosophy*, American Institute of Aeronautics and Astronautics.

[126] **Eckert, S. and Gerbeth, G.** (2002). Velocity measurements in liquid sodium by means of ultrasound Doppler velocimetry, *Experiments in Fluids*, ***32***(5), 542–546.

[127] **Garcia, M.**, **Thomas, J. and Klein, A.** (1998). New Doppler echocardiographic applications for the study of diastolic function, *Journal of the American College of Cardiology*, ***32***(4), 865–875.

[128] **Albrecht, H.E.**, **Damaschke, N.**, **Borys, M. and Tropea, C.** (2003). *Laser Doppler and Phase Doppler Measurement Techniques*, Springer-Verlag Berlin Heidelberg, 1 edition.

[129] **Brusch, L.**, **Torcini, A. and Bar, M.** (2003). Doppler effect of nonlinear waves and superspirals in oscillatory media, *Physical Review Letters*, ***91***(10).

[130] **Carmena, J.M. and Hallam, J.C.T.** (2004). The use of Doppler in Sonar-based mobile robot navigation: inspirations from biology, *Information Sciences*, ***161***(1–2), 71–94.

[131] **Muñuzuri, A.P.**, **Davydov, V.A.**, **Gómez-Gesteira, M.**, **Pérez-Muñuzuri, V. and Pérez-Villar, V.** (1996). Frequency-modulated autowaves in excitable media, *Phys. Rev. E*, ***54***(6), –5921.

[132] **Ayhan, T.**, **Yeniceri, R.**, **Ergunay, S. and Yalcin, M.E.** (2012). Hybrid processor population for odor processing, *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pp.177–180.

[133] **Yalcin, M.E.**, **Suykens, J.A.K. and Vandewalle, J.** (2004). True random bit generation from a double-scroll attractor, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***51***(7), 1395–1404.

[134] **Bassham III., L.E.**, **Rukhin, A.L.**, **Soto, J.**, **Nechvatal, J.R.**, **Smid, M.E.**, **Barker, E.B.**, **Leigh, S.D.**, **Levenson, M.**, **Vangel, M.**, **Banks, D.L.**, **Heckert, N.A.**, **Dray, J.F. and Vo, S.** (2010). SP 800-22 Rev. 1a: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, **Technical Report**, U.S. National Institute of Standards and Technology, Gaithersburg, MD, United States, `http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf`.

[135] **Marsaglia, G.**, (1995), The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness, on-line, `http://www.stat.fsu.edu/pub/diehard/`, accessed July 15, 2014.

[136] **L'Ecuyer, P. and Simard, R.** (2007). TestU01: A C Library for Empirical Testing of Random Number Generators, *ACM Trans. Math. Softw.*, ***33***(4).

[137] **Jun, B. and Kocher, P.** (1999). The Intel Random Number Generator, **White paper**, Cryptography Research, Inc., `http://www.cryptography.com/public/pdf/IntelRNG.pdf`.

[138] **Petrie, C.S. and Connelly, J.A.** (2000). A noise-based IC random number generator for applications in cryptography, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***47***(5), 615–621.

[139] **Fang, X.**, **Wetzel, B.**, **Merolla, J.M.**, **Dudley, J.**, **Larger, L.**, **Guyeux, C. and Bahi, J.** (2014). Noise and Chaos Contributions in Fast Random Bit Sequence Generated From Broadband Optoelectronic Entropy Sources, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***61***(3), 888–901.

[140] **Gerosa, A.**, **Bernardini, R. and Pietri, S.** (2002). A fully integrated chaotic system for the generation of truly random numbers, *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, ***49***(7), 993–1000.

[141] **Callegari, S.**, **Rovatti, R. and Setti, G.** (2005). Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos, *Signal Processing, IEEE Transactions on*, ***53***(2), 793–805.

[142] **Addabbo, T.**, **Alioto, M.**, **Fort, A.**, **Rocchi, S. and Vignoli, V.** (2006). A feedback strategy to improve the entropy of a chaos-based random bit generator, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***53***(2), 326–337.

[143] **Pareschi, F.**, **Setti, G. and Rovatti, R.** (2010). Implementation and Testing of High-Speed CMOS True Random Number Generators Based on Chaotic Systems, *Circuits and Systems I: Regular Papers, IEEE Transactions on*, ***57***(12), 3124–3137.

[144] **Xilinx, Inc.**, (2015). Spartan-6 FPGA Data Sheet: DC and Switching Characteristics, 3.1.1 edition, `http://www.xilinx.com/support/documentation/data_sheets/ds162.pdf`.

[145] **Yeniceri, R.**, **Abtioglu, E. and Yalcin, M.E.** (2015). Cellular Network of Networks on Dynamically Partial Reconfigurable FPGA, *Proceedings of 22nd European Conference on Circuit Theory and Design (ECCTD2015)*, Trondheim, Norway, pp.1–4.

[146] **Yeniceri, R. and Yalcin, M.E.** (2015). Predictive Feedback Motion Planning in 2D Space using Nonlinear Waves with Doppler Effect, *IEEE Transactions on Circuits and Systems-I: Regular Papers*, submitted to.

[147] **Yeniceri, R.**, **Ustaoglu, B. and Yalcin, M.E.** (2013). Throughput enhancement for a new time-delay sampled-data system based True Random Bit Generator, *Circuit Theory and Design (ECCTD), 2013 European Conference on*, pp.1–4.

[148] **Yeniceri, R. and Yalcin, M.E.** (2015). Multi-scroll Chaotic Attractors from A Generalized Time-delay Sampled-data System, *International Journal of Circuit Theory and Applications*, submitted to.

[149] **Yeniceri, R. and Yalcin, M.E.** (2015). Asynchronous Delay Doubler and Binary Low-pass Filter for A Time-delay Chaotic Circuit, *International Journal of Circuit Theory and Applications*, submitted to.

[150] **Yeniceri, R.**, **Kilinc, S. and Yalcin, M.E.** (2015). Attack on a Chaos-Based Random Number Generator Using Anticipating Synchronization, *International Journal of Bifurcation and Chaos*, ***25***(02), 1550021.

[151] **Yeniceri, R. and Yalcin, M.E.** (2015). Anticipating Synchronization Between Sampled-Time Master And Discrete-Time Slave Chaotic Systems,

Proceedings of 7th International Scientific Conference on Physics and Control (PhysCon 2015), Istanbul, Turkey, pp.1–4.

[152] **Yeniceri, R.**, **Ozoguz, S. and Yalcin, M.E.**, (2014). A Chaotic Time-Delay Sampled-Data Systems with Applications, chapter 3, Physics Research and Technology Series, Nova Science Publishers, New York, pp.59–72.

CURRICULUM VITAE

Name Surname	**:**	Ramazan Yeniçeri
Place and Date of Birth	**:**	Denizli, 1985
E-Mail	**:**	yenicerir@itu.edu.tr

EDUCATION:

- **B.Sc.:** 2007, Istanbul Technical University, Electrical-Electronics Faculty, Electronics and Communication Engineering Department
- **M.Sc.:** 2009, Istanbul Technical University, Electronics and Communication Engineering Department, Electronics Engineering Master Program

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2009–2015: Research and Teaching Assistant in the Department of Electronics and Communication Engineering at Istanbul Technical University.
- 2014: Visiting researcher with a scholarship from Scientific and Technological Research Council of Turkey (TUBITAK) at University of Notre Dame, IN.
- 2009–2014: National Scholarship Programme for Ph.D. Students of TUBITAK.

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- Abtioglu, E., **Yeniceri, R.**, and Yalcin, M. E. "Cellular Network of Networks on Dynamically Partial Reconfigurable FPGA," *22nd European Conference on Circuit Theory and Design (ECCTD2015)*, Trondheim, Norway, August 24-26, 2015.
- Karakaya, B., **Yeniceri, R.**, and Yalcin, M. E. "Wave Computer Core Using Fixed-point Arithmetic," *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, Lisbon, Portugal, May 24–27, 2015.
- Tukel, M., **Yeniceri, R.**, and Yalcin, M. E. "Nonlinear spatio-temporal wave computing for real-time applications on GPU," *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*, August 29-31, 2012, Turin, Italy.
- Yalcin, M. E., **Yeniceri, R.**, and Ozoguz, S. "A chaotic time-delay sampled-data system and its implementation," *International Journal of Bifurcation and Chaos*, vol.24, no.3, 2014. DOI: 10.1142/S0218127414500394

- **Yeniceri, R.**, Abtioglu, E., Govem, B., Yalcin, M. E. "A 16×16 Cellular Logical Network with partial reconfiguration feature," *Cellular Nanoscale Networks and Their Applications (CNNA), 2014 14th International Workshop on*, July 29-31, 2014, Notre Dame, IN, USA.

- **Yeniceri, R.**, Kilinc, S., and Yalcin M. E. "Attack on a Chaos-Based Random Number Generator Using Anticipating Synchronization," *International Journal of Bifurcation and Chaos*, vol.25, no.2, 2015. DOI: 10.1142/S0218127415500212

- **Yeniceri, R.**, Ozoguz, S., and Yalcin, M. E. "A Chaotic Time-Delay Sampled-Data Systems with Applications," *New Research Trends in Nonlinear Circuits: Design, Chaotic Phenomena and Applications*, ed. Kyprianidis I., Stouboulos I., and Volos C., chapter 3, Physics Research and Technology Series, pp.59-72, Nova Science Publishers, New York, 2014, ISBN: 978-1-63321-406-4.

- **Yeniceri, R.**, Ustaoglu, B., and Yalcin, M. E. "Throughput enhancement for a new time-delay sampled-data system based True Random Bit Generator," *Circuit Theory and Design (ECCTD), 2013 European Conference on*, September 8-12, 2013, Dresden, Germany.

- **Yeniceri R.**, and Yalcin, M. E. "A new CNN based path planning algorithm improved by the Doppler Effect," *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*, August 29-31, 2012, Turin, Italy.

- **Yeniceri, R.**, and Yalcin, M. E. "The Doppler effect with input driven autowaves," *Circuit Theory and Design (ECCTD), 2013 European Conference on*, September 8-12, 2013, Dresden, Germany.

- **Yeniceri, R.**, and Yalcin, M. E. "True random bit generation with time-delay sampled-data feedback system," *Electronics Letters*, vol.49, no.8, pp.543–545, 2013. DOI: 10.1049/el.2012.3448

- **Yeniceri, R.**, and Yalcin, M. E. "Anticipating Synchronization Between Sampled-Time Master And Discrete-Time Slave Chaotic Systems," *7th International Scientific Conference on Physics and Control (PhysCon 2015)*, Istanbul, Turkey, August 19-22, 2015.

- **Yeniceri, R.**, and Yalcin, M. E. "Asynchronous Delay Doubler and Binary Low-pass Filter for A Time-delay Chaotic Circuit," *International Journal of Circuit Theory and Applications*, online, 2015. DOI: 10.1002/cta.2158

- **Yeniceri, R.**, and Yalcin, M. E. "Multi-scroll Chaotic Attractors from A Generalized Time-delay Sampled-data System," *International Journal of Circuit Theory and Applications*, online, 2015. DOI: 10.1002/cta.2160

- **Yeniceri, R.**, and Yalcin, M. E. Predictive Feedback Motion Planning in 2D Space Using Nonlinear Waves with Doppler Effect, *submitted to IEEE Transactions on Circuits and Systems-I: Regular Papers*.

OTHER PUBLICATIONS:

- Ayhan, T., **Yeniceri, R.**, Ergunay, S., Yalcin, M. E. "Hybrid Processor Population for Odor Processing," *2012 IEEE International Symposium on Circuits and Systems (ISCAS 2012)*, Seoul, Korea, May 20–23, 2012.

- Csaba, G., Papp, A., **Yeniceri, R.**, Porod, W. "Non-Boolean Computing Based on Linear Waves and Oscillators," *accepted to 45th European Solid-State Device Conference*, Graz, Austria, September 14-18, 2015.

- Ergunay, S., **Yeniceri, R.**, and Yalcin, M. E. "Hardware-Software Co-design of Nonlinear Active Wave Generator with Microblaze Soft Core Processor," *2010 International Symposium on Nonlinear Theory and its Applications (NOLTA2010)*, Krakow, Poland, Sept. 5–8, 2010.

- Kilic, V., **Yeniceri, R.**, and Yalcin, M. E. "A New Active Wave Computing Based Real Time Mobile Robot Navigation Algorithm for Dynamic Environment," *12th International Workshop on Cellular Nanoscale Networks and Applications (CNNA 2010)* Berkeley, USA, February 3-5, 2010.

- **Yeniceri, R.**, and Yalcin, M. E. "An Implementation of 2D Locally Coupled Relaxation Oscillators on an FPGA for Real-time Autowave Generation," *11th International Workshop on Cellular Neural Networks and their Applications (CNNA 2008)*, Santiago de Compostela, Spain, July 14-16, 2008.

- **Yeniceri, R.**, and Yalcin, M. E. "An Emulated Digital Wave Computer Core Implementation," *European Conference on Circuit Theory and Design 2009 (ECCTD'09)*, Antalya, Turkey, August 23-27, 2009.

- **Yeniceri, R.**, and Yalcin, M. E. "Path Planning on Cellular Nonlinear Network Using Active Wave Computing Technique," *Bioengineered and Bioinspired Systems IV, Proc. SPIE*, vol. 7365, Dresden, Germany, May 3-5, 2009.