

HAFIZALI HÜCRESEL OTOMAT SAYISAL TASARIMI

YÜKSEK LİSANS TEZİ

Emre GÖNCÜ

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

HAZİRAN 2013

HAFIZALI HÜCRESEL OTOMAT SAYISAL TASARIMI

YÜKSEK LİSANS TEZİ

Emre GÖNCÜ
(504101234)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Müştak Erhan YALÇIN

HAZİRAN 2013

İTÜ, Fen Bilimleri Enstitüsü'nün 504101234 numaralı Yüksek Lisans Öğrencisi **Emre GÖNCÜ**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “**HAFIZALI HÜCRESEL OTOMAT SAYISAL TASARIMI**” başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Doç. Dr. Müştak Erhan YALÇIN**
İstanbul Teknik Üniversitesi

Jüri Üyeleri : **Prof. Dr. İsmail Serdar ÖZOĞUZ**
İstanbul Teknik Üniversitesi

Prof. Dr. Vedat TAVŞANOĞLU
Işık Üniversitesi

.....

Teslim Tarihi : **03 Mayıs 2013**
Savunma Tarihi : **04 Haziran 2013**

Aileme,

ÖNSÖZ

Tez çalışmam boyunca bilgi birikimi ve tecrübesi ile desteğini benden esirgemeyen, değerli hocam, tez danışmanım Doç. Dr. Müştak Erhan Yalçın'a teşekkür ederim.

Aileme her an yanımda olduklarını hissettirdikleri ve maddi, manevi bütün desteklerinden dolayı teşekkür ederim.

Çalışmalarım boyunca benden yardımlarını esirgemeyen başta Yük. Müh. Ramazan Yeniçeri 'ye ve tüm İTÜ Gömülü Sistemler Tasarımı Labarotuvanı lisansüstü ve lisans öğrencilerine teşekkürü bir borç bilirim.

Haziran 2013

Emre GÖNCÜ
(Elektronik Mühendisi)

İÇİNDEKİLER

Sayfa

ÖNSÖZ	vii
İÇİNDEKİLER	ix
KISALTMALAR.....	xi
ÇİZELGE LİSTESİ.....	xiii
ŞEKİL LİSTESİ.....	xv
ÖZET	xvii
SUMMARY	xix
1. GİRİŞ.....	1
2. HÜCRESEL OTOMAT.....	5
2.1 Uygulamalar	7
2.2 Hücresel Otomatın Temel Kavramları.....	7
2.3 Temel Hücresel Otomat.....	8
2.3.1 Totalistik kurallar.....	10
2.3.2 Eklenebilirlik	11
2.3.3 Tersinirlik	11
2.3.4 Hücresel otomatın sınıflandırılması.....	13
2.4 Hücresel Otomatın Sayısal Gerçeklemeleri.....	15
3. HAFIZALI HÜCRESEL OTOMAT.....	19
3.1 R. Alonso-Sanz'ın Hafızalı Temel Hücresel Otomatı	19
3.2 Minimal Hafızalı Sıklık Fonksiyonu	22
4. HAFIZALI HÜCRESEL OTOMATIN SAYISAL GERÇEKLEMESİ.....	25
4.1 Sayısal Sistemin Oluşturulması.....	25
4.2 Benzetim Sonuçları	29
4.3 Gerçekleme Sonuçları	30
5. FİZİKSEL OLARAK KLONLANAMAYAN HÜCRESEL OTOMAT.....	35
5.1 Fiziksel Olarak Klonlanamayan fonksiyonlar	35
5.2 Fiziksel Olarak Klonlanamayan Hücresel Otomat	37
5.3 Deneysel Sonuçlar	41
5.4 FKHO Sistemi Kullanılarak Tümdevre Kimliklendirmesi.....	45
6. SONUÇ VE ÖNERİLER	47
KAYNAKLAR.....	51
EKLER	55
EK A.....	57
EK B.....	61
ÖZGEÇMİŞ	65

KISALTMALAR

FPGA	: Sahada Programlanabilir Kapı Dizileri
FKHO	: Fiziksel Olarak Klonlanamayan Hücresel Otomat
PUF	: Fiziksel Olarak Klonlanamayan Fonksiyonlar

ÇİZELGE LİSTESİ

	<u>Sayfa</u>
Çizelge 4.1: Kural 150'yi gerçekleyen kombinezonsal kısım için doğruluk tablosu.....	28
Çizelge 4.2: Minimal sıklık fonksiyonunu gerçekleyen kombinezonsal kısım için doğruluk tablosu.....	28
Çizelge 5.1: Tümdevre kimliklendirme için yapılan ölçümlerin sonuçları.....	45

ŞEKİL LİSTESİ

	<u>Sayfa</u>
Şekil 2.1 : 1, 2 ve 3-boyutlu hücresel otomatların temsilleri [2].	5
Şekil 2.2 : Temel hücresel otomat kural örneği [1].Bu gösterim Wolfram gösterimi olarak anılır. Çizgilerin üstünde bir sonraki durumu bulunacak hücrenin komşuluğundaki hücrelerin değerleri görülmektedir. Çizgilerin altında ise komşuluğundaki hücrelerin değerlerine göre hücrenin alacağı bir sonraki durum görülmektedir. .	9
Şekil 2.3 : Temel hücresel otomatta kural gösterimi.Kural 254 [17]......	9
Şekil 2.4 : Kural 30'un zamana ve konuma bağlı dinamik davranışı [17]. x eksenini hücrelerin uzayda bulunduğu yeri y eksenini ise hücrelerin durumlarının zamandaki değişimini göstermektedir.....	10
Şekil 2.5 : (a) Kural 150'nin Wolfram gösterimi.(b) Kural 150'nin zamana ve konuma bağlı dinamik davranışı [18].	11
Şekil 2.6 : Tersinir ve Tersinir olmayan iki hücresel otomatın karşılaştırılması. (a) Kural 51, (b) Kural 254 [19]......	12
Şekil 2.7 : Tersinir hücresel otomatlar, (a) Kural 15, (b) Kural 51,(c) Kural 85, (d) Kural 170, (e) Kural 204, (f) Kural 240 [19]......	12
Şekil 2.8 : Kural 90R'nin zamanda değişimi [17].	13
Şekil 2.9 : Hücresel otomatın sınıfları [21].	14
Şekil 2.10 : Shackleford ve arkadaşlarının FPGA'da hücresel otomat gerçeklemeleri [22].	16
Şekil 2.11 : Shackleford ve arkadaşlarının 1-boyutlu hücresel otomat için yaptıkları örnek tasarım [22].	16
Şekil 2.12 : Shackleford ve arkadaşlarının 2-boyutlu ve 3-boyutlu hücresel otomatlar için yaptıkları örnek tasarımlar [22].	17
Şekil 2.13 : Corsonello ve arkadaşlarının 2-boyutlu hücresel otomatlar için geliştirdikleri "CAREM" sisteminin blok diyagramı [25].	17
Şekil 3.1 : Kural 254 [17].	21
Şekil 3.2 : Kural 254 için hafızalı ve hafızasız hücresel otomatların karşılaştırılması [27].	21
Şekil 3.3 : Kural 90 ve Kural 150'nin hafızalı hücresel otomattaki durumları [27].	21
Şekil 3.4 : Kural 254'ün yeni sıklık fonksiyonuna sahip hafızalı hücresel otomattaki durumu [28].	23
Şekil 3.5 : Son üç duruma bakan sıklık fonksiyonunun legal temel hücresel otomat kurallarına etkisi [28].	23
Şekil 4.1 : Wolfram notasyonuna göre minimal hafızalı sıklık fonksiyonu.	25
Şekil 4.2 : Sayısal sistemin en üst seviyeden görünümü.	26
Şekil 4.3 : Bir hücre modülünün detaylı tasarımı.	27

Şekil 4.4	: Sonlu durum makinesi.....	29
Şekil 4.5	: Bir hücre modülünün LUT'lar ile tasarımı.	29
Şekil 4.6	: Tasarlanan sayısal sistemin Kural 150 için (a) davranışsal benzetiminin birinci kısmı, (b) davranışsal benzetiminin ikinci kısmı, (c) davranışsal benzetiminin üçüncü kısmı, (d) yerleştir ve bağla sonrası benzetiminin birinci kısmı, (e) yerleştir ve bağla sonrası benzetiminin ikinci kısmı, (f) yerleştir ve bağla sonrası benzetiminin üçüncü kısmı.	31
Şekil 4.7	: Kural 150'ye sahip hafızalı hücresel otomatın zamanda değişimi [29].	32
Şekil 4.8	: Gerçeklenen sistemin blok diyagramı.	32
Şekil 4.9	: Kontrol bloğunu gerçekleyen algoritmik durum makinesi.....	33
Şekil 4.10	: Gerçekleme sonucu elde edilen 256 hücreli hafızalı hücresel otomatın 256 zaman adımı boyunca zamanda değişimi.	34
Şekil 5.1	: Kelebek PUF'ların yapısı [33].	36
Şekil 5.2	: Keyfi PUF'ların yapısı [31].	37
Şekil 5.3	: Halka osilatör tabanlı PUF'ların yapısı [35].	38
Şekil 5.4	: FKHO sisteminin bir hücre modülünün blok diyagramı.....	39
Şekil 5.5	: FKHO sistemindeki gecikme bloklarının değerlerine göre bir hücrenin durumları.....	40
Şekil 5.6	: 9 hücreli 1-boyutlu 150 kuralına sahip hafızalı hücresel otomatın dördüncü hücresinin, FKHO sistemindeki değerlerinin, hafızalı hücresel otomat sayısal sisteminden farklılığının yüzde grafiği.	42
Şekil 5.7	: $T_{GECIKME} \geq T_{SAAT}$ için 150 kuralına sahip FKHO sisteminin zamanda değişimi.....	42
Şekil 5.8	: $T_{GECIKME} \leq T_{SAAT}$ için 150 kuralına sahip FKHO sisteminin zamanda değişimi.....	43
Şekil 5.9	: Şekil 5.8'de örüntüsü gösterilen aynı sistemin kısa zaman sonra (1 dakika sonra) tekrar çalıştırıldığında elde edilen örüntüsü.	44
Şekil 5.10	: Farklı tümdevreden elde edilen örüntü.....	44
Şekil 5.11	: Şekil 5.8'deki ve Şekil 5.10'daki örüntülerin farkları.	45
Şekil A.1	: Üç canlı hücreden oluşan hayat formlarının dinamik davranışı [17]. ..	57
Şekil A.2	: Dört canlı hücreden oluşan hayat formlarının dinamik davranışı [17].	58
Şekil A.3	: Zaman adımları boyunca değişmeyen hayat formları [17].	58
Şekil A.4	: Bir osilatörün dinamik davranışına örnek [17].	58
Şekil A.5	: Bir planörün dört zaman adımı boyunca dinamik davranışı [17].	59
Şekil B.1	: ZYNQ-7000 tümdevresinin blok diyagramı [36].	62

HAFIZALI HÜCRESEL OTOMAT SAYISAL TASARIMI

ÖZET

Uzay ve zamanın ayrık olduğu hücresel otomatlarda, fiziksel büyüklükler ayrık değerlerden oluşmuş sonlu bir kümenin elemanlarıdır. Bir hücresel otomat, her bir hücresi ayrık değerlerden meydana gelen tek tip düzenli bir latisten oluşur. Hücresel otomatın durumu her bir hücrenin o andaki değerleri ile belirlenir ve ayrık zaman adımlarında değişir. Bu değişim hücresel otomattaki her bir hücrenin aynı anda, komşuluğundaki hücrelerin değerleriyle belli bir kurala göre etkileşip bir sonraki adımda yeni bir değer almasıyla gerçekleşir.

Hücresel otomat kavramı ilk olarak Von Neumann ve Ulam tarafından ortaya atılmıştır. Hücresel otomat, daha sonra fiziksel, kimyasal ve biyolojik sistemlerin modellenmesinde birçok kez kullanılmıştır. Klasik hücresel otomatın gerçek dünyadaki problemlerin modellenebilmesi için yeterli olmadığı anlaşıldığında hücresel otomatın yapısında değişiklikler yapıp, düzensiz hücresel otomat, asenkron hücresel otomat gibi farklı türlerde hücresel otomatlar elde edilmiş ve kullanılmıştır.

Hücresel otomatın modelleme dışında farklı kullanım alanları da mevcuttur. Bunlardan birisi de rastgele sayı üreticileridir. Hücresel otomat, paralelliliğinin getirdiği hız sebebiyle görüntü işleme, tanıma ve şifreleme uygulamalarında kullanılmaya da oldukça uygundur.

1990'da Fredkin, hücresel otomatlara hafızalı kurallar ekleyerek, hafızalı hücresel otomatlar oluşturmuştur. Daha sonra farklı hafızalı hücresel otomat yapıları ortaya atılmıştır. Bunlardan birisi de bu çalışmada kullanılan, R. Alonso-Sanz'ın önerdiği hafızalı hücresel otomatlardır. Bu hafızalı hücresel otomatlarda hücrelerin önceki ve şimdiki zamanlarda aldığı durum değerlerinin sıklığına göre hücrelerin yeni durumları elde edilir. Bu çalışmada ileride bahsedilecek olan minimal hafızalı sıklık fonksiyonu kullanılarak oluşturulan hafızalı hücresel otomat sayısal olarak tasarlanmış ve FPGA(Sahada Programlanabilir Kapı Dizisi) tümdevresinde gerçekleştirilmiştir.

Hücresel otomatlar, deterministik ve ayrık olduğundan sistemden rastgele sonuçlar elde etmek mümkün değildir. Bu çalışmada bahsedilen rastgeleliği elde edebilmek için yeni bir sistem önerilmiştir. Hafızalı hücresel otomatlardan esinlenerek, oluşturulan bu sisteme "Fiziksel Olarak Klonlanamayan Hücresel Otomat" (FKHO) adı verilmiştir. Bu sistemde hafıza fonksiyonları oluşturulurken rastgele değerler kullanılmıştır. Böylece hafıza fonksiyonunun hücrelerin rastgele bir zaman adımı önceki değerine bakarak sonuç üretmesi sağlanmıştır. Önerilen bu sistem FPGA tümdevresinde gecikme hatlarının rastgeleliği kullanılarak gerçekleştirilmiştir. Daha sonra yapılan testler sonucunda aynı aileden olan beş ayrı FPGA tümdevresi FKHO sayesinde kimliklendirilmiştir.

DIGITAL DESIGN OF CELLULAR AUTOMATA WITH MEMORY

SUMMARY

Cellular automata (CA) are a kind of mathematical systems . In these systems space and time are discrete. Physical quantities are element of a finite set which composed of discrete elements, in CA. A cellular automaton (CA) consists of a uniform lattice which composed of discrete cells. State of a CA is determined by current state of cells and updated in every time step. The state of CA is updating by a transition rule.

Cellular means that system is composed of cells. Every cell in cellular automata has an automaton which is actually a finite state machine. Finally, the structure is composed of cells sited at space and finite state machines in cells. When CA is in a state, actually states of cells are in a state of finite state machines. Dimension of cellular spaces can be any size.

Concept of CA was firstly used by Von Neumann and Ulam. von Neumann proposed CA while he was trying to modelling biological self-producing. 2-dimension CA proposed by von Neumann has 29 states. In this CA, cells change its states by a local transition rule regarding to the state of the nearest cells. According to von Neumann Cellular spaces have to have two properties: Universal Computation and Universal Construction.

Universal Computation means that Cellular Space can compute any computable function. Universal Construction means that CA can be create other constructions.

After that, Konrad Zuse showed that actually all spaces can be defined by von Neumann's Cellular Space. Also he determine how that can be performed. This idea is used by Edward Fredkin and Stephen Wolfram, later.

English mathematician John Horton Conway proposed a new Cellular Space named "Game of Life". This Cellular Space has 2-dimension. Conway's Game of life is an important invention. Because it satisfied two properties: Universal Computation and Universal Construction. Martin Gardner published an article about Conway's Game of life in Scientific American.

Wolfram worked with simplest structure of CA which has a two states, 1-dimension and 3-neighborhood. He named that CA Elementary Cellular Automata. He classified that CA regarding to its dinamical behaviours.

Then, CA have been used for modelling many physical, chemical and biological systems. After it had been understood that classical CA was not enough for solving real-world problems, the structures of CA were changed and new kind of CA were generated like inhomogeneous, asynchronous CA. Cellular Automata with memory is an structure that be changed, as well.

There are also many area of usage related CA beside modelling. One of these is random number generators. CA is very useful to image processing and image encryption, as

well because of its parallelism. Also due to this parallelism, CA are appropriate to be implemented by FPGA. In this thesis different designs of CA implemented by FPGA in literature have been given.

Classic CA have no memory. So next state of a cell is rely on only the current states of the neighborhood cells. In 1990, Fredkin has added some rules with memory to CA and generated CA with memory. In Fredkin's CA with memory, next state of a cell rely on not only the current states of the neighborhood cells but also the previous state of the cell.

Then, different kind of CA with memory have been propound. One of these structures is R.Alonso-Sanz's CA which used in our work. In this CA, the new state of CA cells is obtain by regarding frequency of cells state at current time and previous times. So these CA have actually two functions: local transition function and memory function.

Local transition function named local rule as well, is used classical CA, too. Memory function means that calculating new state of the cell regarding to frequency of the cell states.

CA with memory which proposed by Ramon Alonso-Sanz were analysed many times. Its dynamical behaviours were compared with classical CA's dynamic behaviours.

In our work, mathematical definition of CA given in literature is arranged in order to define Ramon Alonso-Sanz's CA with memory according to mathematical form given in literature. Because it is required for Physically Unclonable Cellular Automata (PUCA) that proposed in our work.

Also it has been realized that there is no hardware implementation of Ramon Alonso-Sanz's CA with memory in literature. In our work digital design of R.Alonso-Sanz's CA has been proposed and implemented on FPGA (Field Programmable Gate Array). Implementation results have been compared with results in theory and it has been realized that results are succeed.

Since CA are deterministic, it is not possible to obtain true random results. In our work, to obtain the randomness, a new system has been proposed.

Basic idea of that system is choosing memory of CA randomly at every time step. For example, while deciding next state of a cell relies on previous state of the cell at a time step, it can rely on two previous state of the cell at a different time step.

The proposed System which called Physical Unclonable Cellular Automata (PUCA) is inspired by CA with memory. In this system, different kind of memory functions in which time variables are random are used. Therefore memory function produces output respect to the prior states randomly.

Mathematical definition of the PUCA system have been given appropriate to mathematical forms of classical CA and CA with memory.

The proposed system has been implemented by using delay lines in FPGA. Randomness of that delay lines relies on fabrication process of FPGA. That randomness can be used for designing PUFs (Physical Unclonable Function) and true random number generators.

PUF is a function that transforms physical inputs applying a physical device to physical outputs. Actual feature of this function is producing different outputs for same inputs at different device. In this thesis, slicon PUFs are used.

Silicon PUFs use uncontrollable circuit characteristics arising from the fabrication process. ICs (Integrated Circuit) even from the same lot or wafer have delay variations. Device delays are different due to mask variations. Also environment factors like temperature, circuit aging impact the delay variations.

PUFs are usually used for IC authentication and IP (Intellectual Property) security. IP designers prevent that vicious users have their design by using PUFs. Because PUFs provide a secret key specified to IC.

In our work, a PUCA application has been developed as an alternative to PUFs. A digital design of the PUCA has been proposed. Then the proposed design has been implemented on FPGA. Results of the implementation have been given in thesis. Respect to the results, PUCA produces different patterns when it started at different times. Also it has been realized that PUCA produces different outputs for different FPGAs (same families). So PUCA can be a very suitable system for IC authentication.

A digital design of PUCA has been proposed and implemented on FPGA. Then, this PUCA system has been executed one thousand times for 63 time steps. Configurations of PUCA at time step 63 have been converted to decimal numbers. After that, average of those numbers has been calculated. That process has been performed for five FPGAs which are the same IC. It has been realized that, measurement of the averages are different for each FPGA. Those averages have been used for authentication of FPGAs.

1. GİRİŞ

İlk olarak 1966'da ortaya atılan hücresele otomatlar sayesinde, günümüze kadar birçok mühendislik problemine çözüm bulunmuştur. İlk başlarda fiziksel, kimyasal ve biyolojik sistemlerin modellenmesinde kullanılan hücresele otomatlar, daha sonra rastgele sayı üreticileri, görüntü işleme, görüntü şifreleme gibi birçok farklı alanda da kullanılmıştır.

Hücresele otomatlar, fiziksel sistemlerin matematiksel olarak idealleştirilmesidir. Uzay ve zamanın ayrık oduğu bu sistemlerde, fiziksel büyüklükler ise ayrık değerlerden oluşmuş sonlu bir kümenin elemanlarıdır [1]. Bu tez çalışmasında, hücresele otomatın literatürde verilen matematiksel tanımları araştırılarak tezin asıl konusu olan, oluşturulan yeni sisteme en uygun olanı seçilmiştir.

Klasik hücresele otomatın gerçek dünyadaki problemlere çözüm bulmakta yeterli olmadığı anlaşıldığında hücresele otomatın yapısında değişiklikler yapılarak, düzensiz, asenkron gibi farklı türlerde hücresele otomatlar elde edilmiş ve kullanılmıştır. Buna benzer şekilde Fredkin, hücresele otomatlara hafızalı kurallar ekleyerek, hafızalı hücresele otomatlar oluşturmuştur. Daha sonra farklı hafızalı hücresele otomat yapıları ortaya atılmıştır. Bunlardan birisi de bu çalışmada kullanılan, R. Alonso-Sanz'ın önerdiği hafızalı hücresele otomatlardır.

Hafızalı hücresele otomatlar, günümüze kadar pek çok kez incelenmiş, dinamik davranışları, klasik hücresele otomatların dinamik davranışlarıyla karşılaştırılmıştır. Bu tez çalışmasında literatürde verilen klasik hücresele otomatın matematiksel tanımına uygun olarak, hafızalı hücresele otomat için yeni bir matematiksel tanım önerilmiştir. Aynı zamanda literatürde R. Alonso-Sanz'ın önerdiği hafızalı hücresele otomatın donanımsal olarak gerçeklemesinin olmadığı farkedilmiş ve bu sistem FPGA tümdevresi kullanılarak gerçekleştirilmiş ve sonuçları verilmiştir.

Sayısal sistemlerle çalışırken, hayat deterministik ve ayrık olduğundan sistemden rastgele sonuçlar elde etmek mümkün değildir. Bu çalışmada bahsedilen rastgeleliği

elde edebilmek için yeni bir sistem önerilmiştir. Bu sistemi oluştururken kullanılan temel düşünce hafızalı hücresel otomatta her bir zaman adımında hafızanın rastgele olarak seçilmesidir. Örneğin bir zaman adımında hücrenin bir önceki zaman adımındaki değerine bakılırken, sonraki zaman adımında hücrenin iki önceki zaman adımındaki değerine bakılabilir. Hafızalı hücresel otomatlardan esinlenerek, oluşturulan bu sisteme "Fiziksel Olarak Klonlanamayan Hücresel Otomat" (FKHO) adı verilmiştir. Bu sistemde hafıza fonksiyonları oluşturulurken rastgele değerler kullanılmıştır. Böylece hafıza fonksiyonunun hücrelerin rastgele bir zaman adımı önceki değerine bakarak sonuç üretmesi sağlanmıştır. Sistemin matematiksel tanımı klasik hücresel otomat ve hafızalı hücresel otomat için kullanılan matematiksel tanımlara uygun olarak verilmiştir. Önerilen bu sistem FPGA tümdevresinde gecikme hatlarının rastgeleliği kullanılarak gerçekleştirilmiştir. FPGA tümdevresinde üretim işlemlerinden kaynaklanan bu rastgelelik PUF (Fiziksel Olarak klonlanamayan fonksiyon) yapılarında ve rastgele sayı üreteçlerinde kullanılmaktadır.

PUF, fiziksel bir cihaza uygulanan girişleri yine fiziksel çıkışlara dönüştüren bir fonksiyondur. Bu fonksiyonun özelliği aynı türden cihazlara uygulan aynı girişlerin her cihazda farklı sonuç üretmesidir. Bu tez çalışmasında önerilen FKHO sistemi PUF'lara alternatif olarak geliştirilmiştir. Elde edilen sonuçlara göre FKHO sisteminin aynı girişlere karşılık aynı türden farklı cihazlarda farklı çıkışlar verdiği görülmüştür. Burada aynı türden farklı cihazlara örnek olarak aynı üretim firmasının ürettiği aynı model tümdevreler verilebilir.

Bu tez çalışmasında yapılan testler sonucunda aynı aileden olan beş ayrı FPGA tümdevresine FKHO sistemi gömülmüştür. Bu tümdevreler FKHO sistemi kullanılarak ileride detayları verilecek çeşitli işlemler sonucunda kimliklendirilmiştir.

Tez çalışmasının ikinci bölümünde hücresel otomatlar ile ilgili genel bilgi verilip temel hücresel otomatlar ve bunların özelliklerinden bahsedilmiş ve literatürde yer alan hücresel otomatın sayısal gerçeklemelerine değinilmiştir.

Üçüncü bölümde, hafızalı hücresel otomatlar tanımlanmış ve özelliklerinden bahsedilmiştir. Daha sonra standart (hafızasız) hücresel otomat ve hafızalı hücresel otomat arasındaki farklara değinilmiştir.

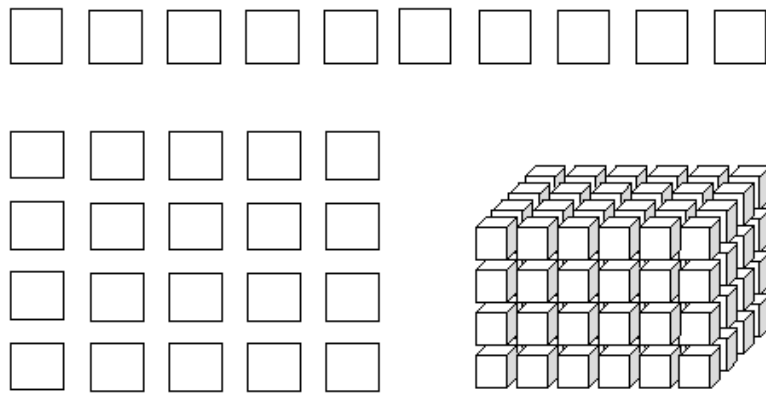
Dördüncü bölümde, üçüncü bölümde anlatılan hafızalı hücreli otomatın sayısal olarak tasarlanarak FPGA tümdevresinde gerçekleştirilmesi ve sonuçları anlatılmıştır.

Beşinci bölümde öncelikle fiziksel olarak klonlanamayan fonksiyonlar (PUF) anlatılmıştır. Daha sonra FKHO sisteminin tanımı verilmiştir. Bunun yanında FKHO tanımına uygun bir şekilde sistemin, sayısal tasarımı verilmiştir. Daha sonra FPGA tümdevresinde gerçekleştirilen bu sayısal tasarımın farklı tümdevrelerde gerçekleştirilerek elde edilen sonuçları verilmiştir.

2. HÜCRESEL OTOMAT

Hücresel otomatlar, fiziksel sistemlerin matematiksel olarak idealleştirilmesidir. Uzay ve zamanın ayrık oduğu bu sistemlerde, fiziksel büyüklükler ise ayrık değerlerden oluşmuş sonlu bir kümenin elemanlarıdır. Bir hücresel otomat, her bir hücresi ayrık değerlerden meydana gelen tek tip düzenli bir latisten oluşur. Hücresel otomatın durumu her bir hücrenin o andaki değerleri ile belirlenir ve ayrık zaman adımlarında değişir. Bu değişim hücresel otomattaki her bir hücrenin aynı anda, komşuluğundaki hücrelerin değerleriyle belli bir kurala göre etkileşip bir sonraki adımda yeni bir değer almasıyla gerçekleşir [1].

Hücresel sözcüğü, sistemin hücrelerden oluştuğunu belirtmek için kullanılır. Bu sistemdeki her bir hücre aslında sonlu durum makinesi olarak ifade edilebilecek bir otomat içerir. Sonuçta ortaya çıkan yapı uzaya yerleştirilmiş hücreler ve bu hücrelerin içerisinde bulunan sonlu durum makineleridir. Hücre bir durumda denildiğinde, o hücrenin sonlu durum makinesinin bir durumunda olduğu kastedilmektedir. Hücresel uzaylar herhangi bir boyutta olabilir. Şekil 2.1’te bir, iki ve üç boyutlu hücresel otomatların temsilleri verilmiştir [2].



Şekil 2.1: 1, 2 ve 3-boyutlu hücresel otomatların temsilleri [2].

Hücresel otomat kavramı ilk kez John von Neumann tarafından önerilmiştir [3] [4]. John von Neumann 1950’lerde kendi kendine çoğalabilen organizmaların modellenenebilmesi için en basit makineyi bulmaya çalışıyordu. Fakat bu makinenin

kullanışlı olabilmesi için evrensel bir hesaplayıcı olması gerekiyordu. Ulam'ın [5] önerdiği farklı bir modele kadar, von Neumann differansiyel denklemler ile bu problemi çözmeye çalışmıştır. 2-boyutlu modelin yeterli olduğunu farkedenden sonra Ulam'ın önerdiği modelle kendi modelini oldukça basitleştirmiştir. von Neumann'ın 2-boyutlu modelinde hücreler, 29 olası durumda olabilecek hücrelerden oluşmaktadır. Bu modelde her hücre durumunu kendisine en yakın hücrelerin durumuna göre hesaplanan bir yerel kural göre değiştirir. von Neumann oluşturduğu uzaya hücreler, uzaydaki yapılar ise hücreler otomat ismini vermiştir. Kendi kendine çoğalma von Neumann'ın ortaya koyduğu problemin sadece bir kısmıdır. Aslında bu sistemde iki özelliğin sağlanmasına dikkat edilmiştir: Birincisi oluşturulan yapının herhangi hesaplanabilen bir fonksiyonu gerçekleyebilmesi, ikincisi ise herhangi bir otomatın başka otomatlar oluşturabilmesidir. von Neumann kendi kendine çoğalabilen makineler için bir çekirdek oluşturabilmesine rağmen işine tam olarak bitiremeden hayatını kaybetmiştir. Modelin tamamı 1966'da Athur W. Burks tarafından yayınlanmıştır. Daha sonra Codd [6], von Neumann'ın önerdiği komşuluk ile evrensellik özelliklerini sağlayan iki durumlu hücrelerden oluşan bir hücreler uzayın olamayacağını göstermiş ve von Neumann'ın hücreler uzayından daha basit kurallara sahip bir hücreler uzay önermiştir. Bu uzayın yukarıda verilen iki evrensellik özelliğini de sağladığını göstermiştir. 1967'de Konrad Zuse [7] aslında bütün uzayın von Neumann'ın tanımladığı hücreler uzayla nasıl tanımlanabileceğini göstermiştir. Daha sonra bu fikir Edward Fredkin ve Stephen Wolfram tarafından kullanılmıştır. Codd'ın deneylerinden sonra İngiliz Matematikçi John Horton Conway [8] "hayat oyunu" adını verdiği yeni bir hücreler uzay geliştirmiştir. Hayat oyunu büyük bir buluştur. Çünkü yayınlandığı zaman bilinmemesine rağmen, evrensel hesaplama ve evrensel yapı özelliklerinin ikisine de sergileyen hücreler uzaylardan biridir. Martin Gardner [9], Hayat oyunu ile alakalı Scientific American dergisinde bir makale yazmıştır. O zamanın bilgisayarları ile Hayat Oyunu'nun simülasyonunu yapmak oldukça popülerdi. Daha sonra Wolfram [1], hücreler otomatının en basit hali (1-boyutlu, 3 komşuluklu, 2 durumlu) ile ilgilenmiştir. Temel hücreler otomat adını verdiği bu hücreler otomatının davranışsal sınıflandırmasını yapmıştır.

Harold Niesche [2], Hücresel otomatın günümüze kadar olan gelişme sürecini detaylı olarak incelemiştir.

2.1 Uygulamalar

Hücresel otomat, günümüze kadar birçok uygulamada kullanılmıştır. Burstedde [10], yayaların hareketlerini modellemek için bir hücresel otomat geliştirmiştir. Hücresel otomatın modelleme dışında farklı kullanım alanları da mevcuttur. Bunlardan birisi de rastgele sayı üreticileridir [11]. Hücresel otomat, paralelliliğinin getirdiği hız sebebiyle görüntü işleme, tanıma ve şifreleme uygulamalarında kullanılmaya da oldukça uygundur [12] [13] [14].

Hücresel otomat ile ilgili en çok bilinen çalışmalardan biri de CAM-6 CA makinesidir. CAM-6 CA makinesi, hücresel otomat simülasyonları için özel olarak geliştirilmiş bir bilgisayardır. Büyük çaplı hücresel otomat deneylerinde birçok kez kullanılmıştır [15].

2.2 Hücresel Otomatın Temel Kavramları

d -boyutlu düzenli bir hücresel otomat $(\mathbb{Z}^d, V, Q, f)$ 4-lüsü ile tanımlanabilir [16]. Burada $\mathbb{Z}^d$, hücresel otomatı oluşturan hücrelerin bulunduğu uzayı ifade eder. V , komşuluk vektörüdür:

$$V = (i_1, i_2, \dots, i_s), i_m \in \mathbb{Z}^d, m = 1, 2, \dots, s$$

Burada s , V komşuluk kümesinin kardinalitesi, $s = |V|$ 'dir. Kardinalitesi s olan komşuluk kümesine sahip hücresel otomatlar s -komşulukludur denir.

Q , sonlu durum kümesidir. Hücrelerin alabileceği değerler bu kümenin elemanlarıdır.

f , yerel geçiş fonksiyonudur. Hücresel otomatın kuralı olarak da isimlendirilir. Hücrelerin komşuluklarında bulunan hücrelerin durumlarına göre bir sonraki durumuna geçişini sağlar:

$$f : Q^s \rightarrow Q$$

$i \in \mathbb{Z}^d$, hücresel otomatın bir hücresinin uzaydaki yerini göstermek üzere α_i^T , o hücrenin T anındaki durumunu belirtsin. Buna göre senkron düzenli bir hücresel otomatın dinamiği (2.1)'deki gibi verilir.

$$\alpha_i^{T+1} = f(\alpha_{i-i_1}^T, \dots, \alpha_{i-i_s}^T), \quad T = 0, 1, 2, \dots \quad (2.1)$$

α_i^T , $T = 0$ için i hücresinin başlangıç durumunu belirtir.

Bütün hücrelerinin kuralları aynı olan hücresel otomatlar düzenli hücresel otomatlar, kuralları hücreden hücreye veya zamanla değişen hücresel otomatlar ise düzensiz hücresel otomatlar olarak isimlendirilir.

Senkron hücresel otomat, otomattaki her bir hücrenin aynı anda güncellendiği hücresel otomatlardır. Asenkron hücresel otomattaki hücreler ise birbirinden bağımsız bir şekilde farklı zamanlarda güncellenirler. Yani asenkron hücresel otomatta bir hücrenin sonraki durumu, o hücrenin komşusu olan hücrelerin durumunu etkileyebilir.

1-boyutlu hücresel otomatın sınır koşullarını iki şekilde tanımlayabiliriz:

Periyodik sınır koşulunda hücresel otomatın en solundaki hücrenin solunda en sağdaki hücrenin var olduğunu kabul ederiz. Terside en sağdaki hücre için geçerlidir.

Sabit sınır koşulunda ise, hücresel otomatın en solundaki hücrenin solunda ve en sağdaki hücrenin sağında sabit bir değere sahip hücreler olduğu kabul edilir.

Çalışmanın bundan sonraki bölümlerinde, hücresel otomat ifadesi kullanıldığında düzenli ve senkron hücresel otomat kastedilmiş olacaktır.

2.3 Temel Hücresel Otomat

Temel hücresel otomatların hücrelerinin bulunduğu uzay $\mathbb{Z}$ 'dir (1-boyutlu). Komşuluk vektörü $V = (1, 0, -1)$ 'dir. Durum kümesi $Q = \{0, 1\}$ 'dir. [1] de temel hücresel otomat detaylı bir şekilde anlatılmıştır:

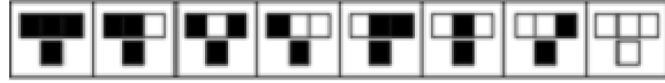
Şekil 2.1'de bir temel hücresel otomatın bir kuralının Wolfram notasyonu ile gösterimi verilmiştir. Hücrenin komşularının alabileceği değerlere karşılık kendisinin bir sonraki adımında alacağı değerler belirtilmiştir. Şekil 2.3'de kuralların bir gösterimi daha verilmiştir. Burada siyah hücreler 1 durumunu, beyaz hücreler ise 0 durumunu

ifade etmektedir. Kuralların isimleri ikili değerin onluk sistemdeki karşılığı olarak verilir. Dolayısıyla 3-komşuluklu temel hücresel otomatta toplam $2^8 = 256$ yerel kural mevcuttur.

Hücresel otomatta bu 256 kuralla alakalı iki kısıtlama yapılır. Birincisi, başlangıç durumu bütün hücrelerin 0 olduğu hücresel otomatların hücreleri daha sonraki adımlarda değişiyorsa uygulanan kural illegaldir. Bu yasak kurallar son biti 1 olan kurallardır. Çünkü bu kurallar 0 ve 1 değerlerini alan hücrelerin işleyişindeki simetriyi ortadan kaldırır. İkincisi, kurallar yansımali ve simetrik olmalıdır. Yani 100, 001, 110, 011 aynı sonuçları vermelidir. Bu iki kısıtlama sayesinde 32 tane $\alpha_1 \alpha_2 \alpha_3 \alpha_4 \alpha_5 \alpha_6 \alpha_7 0$ formunda legal kural kalmış olur.

1 1 1	1 1 0	1 0 1	1 0 0	0 1 1	0 1 0	0 0 1	0 0 0
0	1	0	1	1	0	1	0

Şekil 2.2: Temel hücresel otomat kural örneği [1]. Bu gösterim Wolfram gösterimi olarak anılır. Çizgilerin üstünde bir sonraki durumu bulunacak hücrenin komşuluğundaki hücrelerin değerleri görülmektedir. Çizgilerin altında ise komşuluğundaki hücrelerin değerlerine göre hücrenin alacağı bir sonraki durum görülmektedir.



Kural 254

Şekil 2.3: Temel hücresel otomatta kural gösterimi. Kural 254 [17].

Hücresel otomatın kuralları, Boolean fonksiyonu gibi düşünülebilir. α_i^T , i hücresinin (uzayda i noktasında bulunan) T zaman adımıdaki değerini gösterebilir. Örneğin Kural 90'ı ele alırsak, bu kurala göre aslında hücrenin sağındaki ve solundaki hücrelerle mod 2'de toplama işlemi veya mantıksal xor (exclusive or) işlemi yapılmaktadır (2.2).

$$\alpha_i^{T+1} = \alpha_{i-1}^T \oplus \alpha_{i+1}^T \quad (2.2)$$

Benzer şekilde Kural 18, $\alpha_i^{T+1} = \alpha_i^T \vee (\alpha_{i-1}^T \oplus \alpha_{i+1}^T)$ işlemine karşılık gelmektedir. Bu eşitlikte " $\vee$ ", mantıksal veya işlemini belirtmektedir. Kural 150, $\alpha_i^{T+1} =$

$\alpha_{i-1}^T \oplus \alpha_i^T \oplus \alpha_{i+1}^T$ eşitliğiyle ifade edilebilir. Hücresel otomat kurallarının Boolean fonksiyonları ile gösterimi pratik gerçeklemelerin yapılabilmesi için oldukça uygundur.

Hücresel otomatlar, kolay anlaşılabilir bir grafiksel gösterime sahiptir. Şekil 2.4'te Kural 30'a sahip bir temel hücresel otomatın zamandaki değişimi gösterilmiştir. Başlangıç durumu ilk satırda gösterilen hücresel otomat, bir siyah hücreden (1 durumu) meydana gelmektedir. Geri kalan hücreler beyazdır (0 durumu). İkinci satırda Kural 30'a göre ikinci zaman adımıdaki hücrelerin durumu gösterilmiştir. Aynı şekilde üçüncü satırda hücrelerin üçüncü zaman adımıdaki durumu gösterilmektedir. Grafikte aşağı doğru iniş zamanda ilerlemeye karşılık düşmektedir. Her bir satır ise hücrelerin uzaydaki değerlerini göstermektedir.



Şekil 2.4: Kural 30'un zamana ve konuma bağlı dinamik davranışı [17]. x eksenı hücrelerin uzayda bulunduğu yeri y eksenı ise hücrelerin durumlarının zamandaki değişimini göstermektedir.

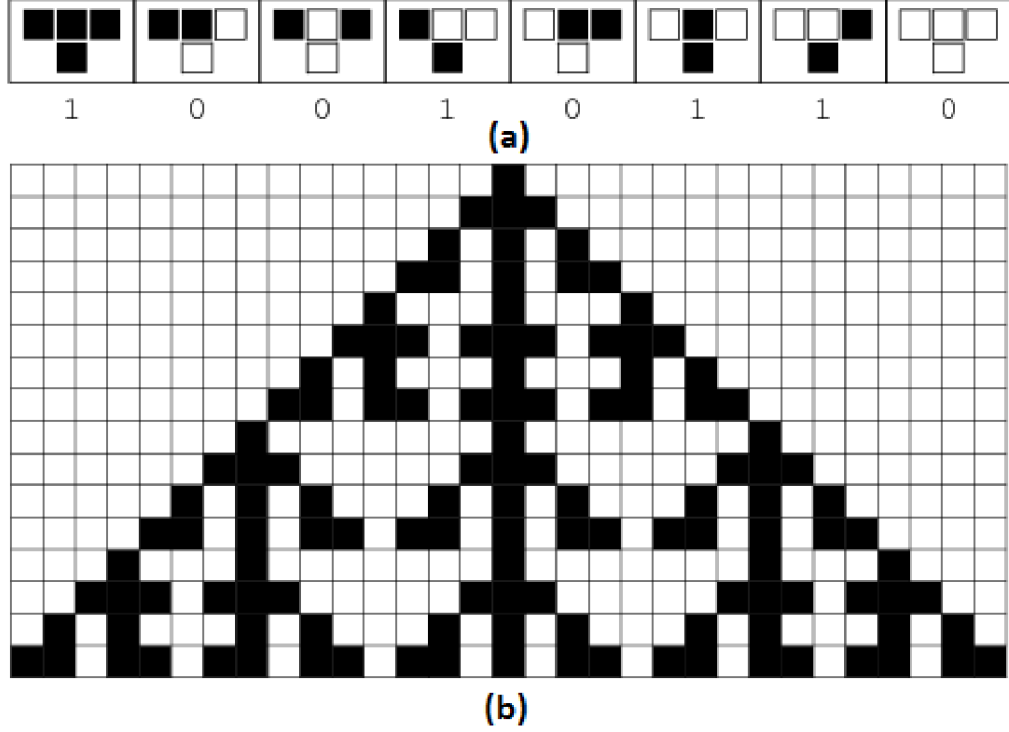
2.3.1 Totalistik kurallar

Komşulukta bulunan hücrelerin değerlerinin toplamının bir fonksiyonundan oluşan kurallara totalistik kurallar denir(2.3) [17].

$$\alpha_i^{T+1} = \phi [\alpha_{i-1}^T + \alpha_i^T + \alpha_{i+1}^T] \quad (2.3)$$

Örneğin, $\alpha_i^{T+1} = \alpha_{i-1}^T \oplus \alpha_i^T \oplus \alpha_{i+1}^T$ eşitliğiyle verilen Kural 150, totalistik bir kuraldır. Çünkü $\oplus \text{ mod } 2$ 'de toplama işlemidir. Yani toplamın bir fonksiyonudur. Temel hücresel otomatta böyle yazılabilecek toplam 16 totalistik kural mevcuttur. Çünkü temel hücresel otomatta komşulukta bulunan hücrelerin toplamı 4 farklı değer olabilir:

0, 1, 2, 3 [17]. Şekil 2.5'te totalistik kurallardan birisi olan Kural 150'ye sahip bir temel hücresel otomatın zamanda değişimi verilmiştir.



Şekil 2.5: (a) Kural 150'nin Wolfram gösterimi.(b) Kural 150'nin zamana ve konuma bağlı dinamik davranışı [18].

2.3.2 Eklenebilirlik

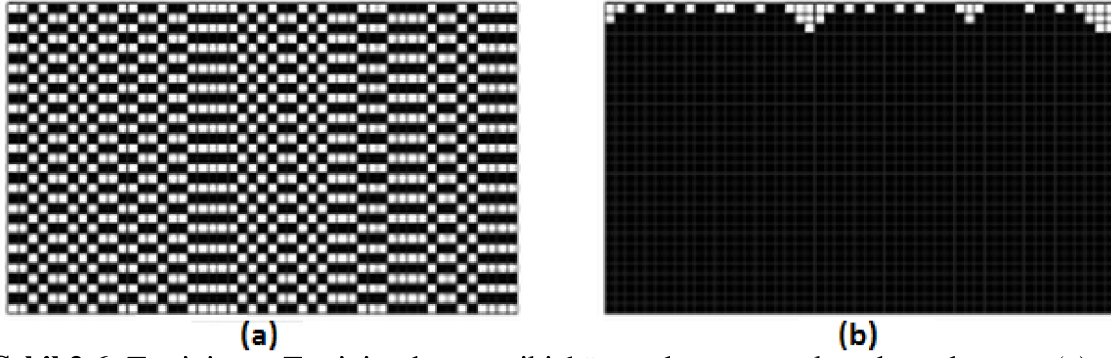
$\alpha_i^{T+1} = x\alpha_{i-1}^T \oplus y\alpha_i^T \oplus z\alpha_{i+1}^T$ formuna sahip kurallara eklenebilir özelliğine sahip kurallar denir. Burada x, y ve z, 0 veya 1 olabilen katsayılardır. x, y ve z katsayılarının kombinasyonlarına göre temel hücresel otomatın 256 kuralının yalnızca 8 tanesi eklenebilir özelliğine sahiptir. Bu kurallar, Kural 0 ($x = 0, y = 0, z = 0$), Kural 60 ($x = 1, y = 1, z = 0$), Kural 90 ($x = 1, y = 0, z = 1$), Kural 102 ($x = 0, y = 1, z = 1$), Kural 150 ($x = 1, y = 1, z = 1$), Kural 170 ($x = 0, y = 0, z = 1$), Kural 204 ($x = 0, y = 1, z = 0$), Kural 240 ($x = 1, y = 0, z = 0$)'dır [17].

2.3.3 Tersinirlik

Günümüze kadar gelen bilgi birikimimize göre temel fizik yasaları tersinirdir. Yani bir fiziksel sistemin şimdiki zamandaki durumuyla alakalı yeterince bilgiye sahipsek,

söz konusu sistemin sadece gelecekteki durumunu değil aynı zamanda geçmişteki durumunu da bilebiliriz [19].

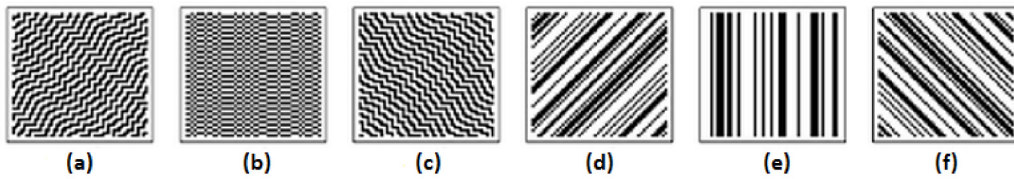
Kural 51 için konuşursak, her hücre bir önceki adımdaki durumunun tersi durumundadır (Şekil2.6). Yani hangi zaman adımında olursak olalım hücresel otomatın geçmişteki değerlerini hesaplayabiliriz. Kural 254'te ise herhangi bir adımdaki hücresel otomatın durumuna bakarak, hücresel otomatın geçmişi hakkında bilgi elde edemeyiz (Şekil2.6).



Şekil 2.6: Tersinir ve Tersinir olmayan iki hücresel otomatın karşılaştırılması. (a) Kural 51, (b) Kural 254 [19].

Örneklerden de anlaşıldığı gibi bir hücresel otomatın tersinir olabilmesi için hücresel otomatın her adımındaki durumuna neden olan önceki adımdaki durumunun tek olması gerekir.

Temel hücresel otomatın 256 kuralının sadece 6 kuralı tersinirdir (Şekil 2.7).



Şekil 2.7: Tersinir hücresel otomatlar, (a) Kural 15, (b) Kural 51, (c) Kural 85, (d) Kural 170, (e) Kural 204, (f) Kural 240 [19].

Edward Fredkin, 1990'da tersinir olmayan kurallardan tersinir kurallar elde edebilmek için yeni bir yöntem bulmuştur [20]. Bu yöntemle göre $f(\alpha_{i-1}^T, \alpha_i^T, \alpha_{i+1}^T)$ formundaki bir kurala mod2'de α_i^{T-1} terimi eklenir. Yeni kural (2.4) eşitliğindeki gibi olur. Böylece yeni kuralımızın tek bir tersi vardır(2.5) [17].

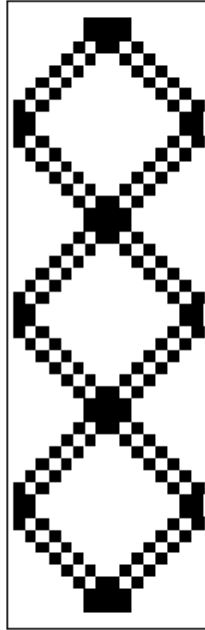
$$\alpha_i^{T+1} = f(\alpha_{i-1}^T, \alpha_i^T, \alpha_{i+1}^T) \oplus \alpha_i^{T-1} \quad (2.4)$$

$$\alpha_i^{T-1} = f(\alpha_{i-1}^T, \alpha_i^T, \alpha_{i+1}^T) \oplus \alpha_i^{T+1} \quad (2.5)$$

Tersinir hale getirilen kurallar, isimlerinin sonlarına R harfi getirilerek adlandırılır. Kural 90'nın tersinir hali Kural 90R'nin ve Kural 90R'nin tersinin eşitlikleri sırasıyla eşitlik (2.6) ve eşitlik (2.7)'de verilmiştir. Şekil 2.8'de periyodik sınır koşullarına ve Kural 90R'ye sahip bir temel hücresel otomatın zamanda değişimi verilmiştir.

$$\alpha_i^{T+1} = \alpha_{i-1}^T \oplus \alpha_{i+1}^T \oplus \alpha_i^{T-1} \quad (2.6)$$

$$\alpha_i^{T-1} = \alpha_{i-1}^T \oplus \alpha_{i+1}^T \oplus \alpha_i^{T+1} \quad (2.7)$$

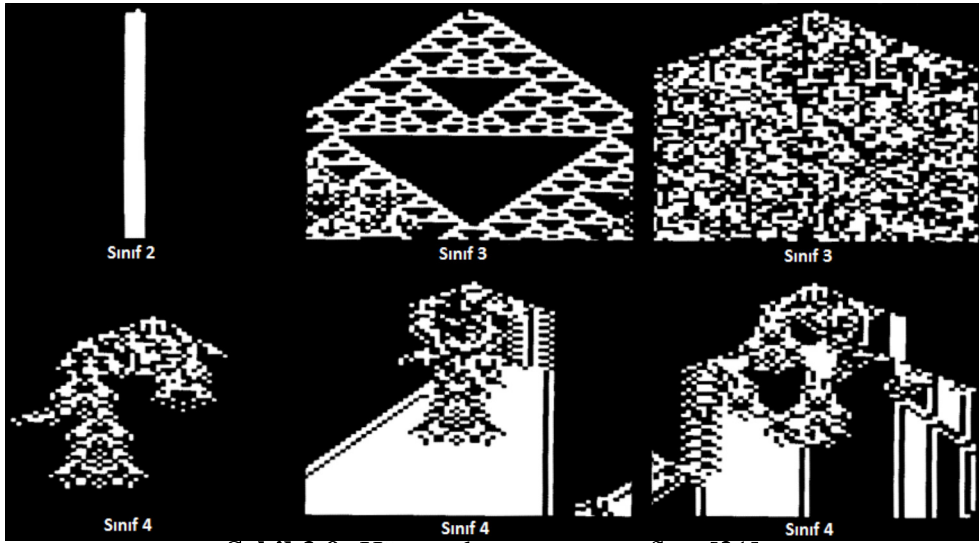


Şekil 2.8: Kural 90R'nin zamanda değişimi [17].

2.3.4 Hücresel otomatın sınıflandırılması

Stephen Wolfram, 80'li yılların başlarında hücresel otomatları dinamik davranışlarına göre sınıflandırmıştır. Bu sınıflandırmayı başlangıç durumu verilen bir hücresel otomatın zamana bağlı değişiminin tahmin edilebilme derecesine göre yapmıştır. Sınıf 1'deki hücresel otomatlar için tam bir tahmin yapabilmek kolaydır. Bu sınıfa ait hücresel otomatlar başlangıç durumlarından bağımsız olarak zaman adımları boyunca

tek bir homojen duruma giderler. Sınıf 2'ye ait hücresel otomatlarda, belli hücrelerde yapılan değişiklik sonlu sayıda hücreye etki eder. Yani sadece bir hücrenin başlangıç durumunda yapılan bir değişiklik, sonsuz zaman adımıyla dahi, sadece bu hücrenin etrafındaki sonlu sayıdaki hücreye etki eder. Şekil 2.9'de sınıf 2'ye ait bir hücresel otomatın davranışı görülmektedir. Böylece bu sınıfa ait bir hücresel otomatın belli hücrelerinin son durumunu tahmin edebilmek için sadece sonlu sayıda hücrenin başlangıç durumlarını bilmek yeterlidir. Sınıf 3'teki hücresel otomat, Şekil 2.9'de görüldüğü gibi, sonsuza kadar neredeyse aynı hızla yayılır, böylece zaman geçtikçe daha fazla hücreye etki eder. Yani zaman adımı arttıkça belli bir hücrenin durumunu tahmin edebilmek için başlangıç durumları bilinmesi gereken hücre sayısı artar. Böylece bu sınıftaki bir hücresel otomatta bir hücrenin sonsuz adım sonraki durumunu tahmin edebilmek için sonsuz sayıda hücrenin başlangıç durumunu bilmemiz gerekir. Sınıf 4'teki hücresel otomatların ise Şekil 2.9'de de görüldüğü gibi zamanla yayılma hızı da değişmektedir. Bu sınıfın tahmin edilebilirlik derecesi en düşüktür [21].



Şekil 2.9: Hücresel otomatın sınıfları [21].

Sınıf 1'e ait hücresel otomatlar, değeri 1 olan her bir hücresi çekici sabit nokta olan dinamik sistemlere karşılık gelir. Sınıf 2'ye ait hücresel otomatlar, periyodik davranış sergileyen dinamik sistemlere karşılık gelir. Sınıf 3'e ait hücresel otomatlar kaotik dinamik sistemlere benzerler. Rastgele bir davranış sergilerler. Sınıf 4'e ait hücresel otomatlar ise kendisine benzeyen bir dinamik sisteme sahip değildir. Aslında Sınıf 3 ile Sınıf 4 arasında gibidir. Yani periyodiklik ile rastgelelik arasındadır [17].

2.4 Hücresel Otomatın Sayısal Gerçeklemeleri

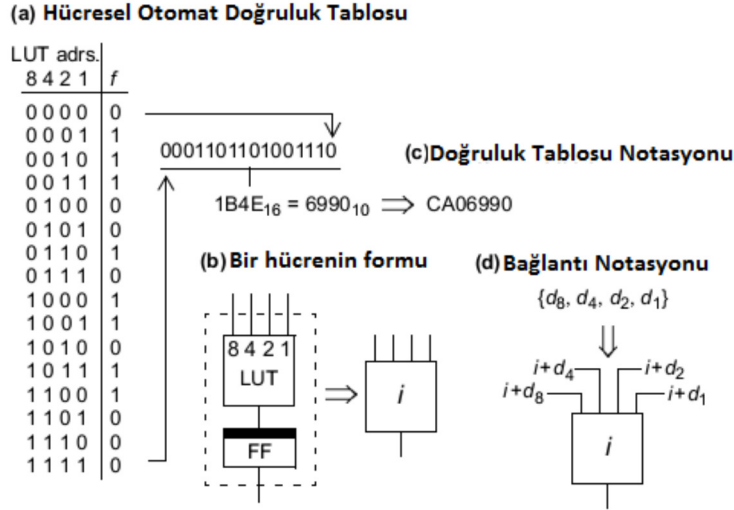
İki durumlu hücresel otomatların yapısı FPGA’da(Sahada programlanabilir kapı dizisi) gerçekleştirme için oldukça uygundur. Günümüze kadar hücresel otomatların bir çok kez FPGA gerçeklemeleri yapılmıştır [22] [23] [24] [25] [26].

Kobori ve arkadaşlarının [24], hücresel otomat için tasarladıkları sistem bir masaüstü bilgisayardan ve içinde FPGA olan bir PCI kartından oluşmaktadır. Bu sistemde kullanıcı C programlama dilinden oluşturulmuş bir özel bir dille hücresel otomatın kuralını belirler. Daha sonra kural derlenip özel bir metotla tasarlanmış donanıma bağlanır. Bu metotla çok fazla hücreye (2048x1024 gibi) sahip hücresel otomatlar paralel yapılarla ve pipeline yapılarıyla etkin bir şekilde oluşturulur. Kullanılan FPGA’nın giriş/çıkış pinlerinin sayısının hücresel otomatın hücre sayısından daha az olmasına ve daha fazla olmasına göre değişik tasarımlar yapılmıştır.

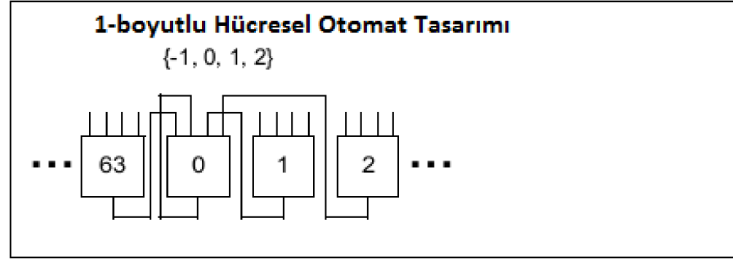
Shackleford ve arkadaşları [22], 4-komşuluklu 1-boyutlu, 2-boyutlu, 3-boyutlu hücresel otomat için her bir hücrenin, onların zamanındaki popüler FPGA’ların bir lojik diliminde bulunan dört girişli bir LUT’tan (Look Up Table) ve 1-bitlik bir yazmaçtan oluştuğu bir tasarım ortaya koymuşlardır. Şekil 2.10’da bu tasarımın detayları özetlenmiştir. Şekil 2.10-a’da bir hücrenin kuralının LUT ile nasıl temsil edileceği gösterilmiştir. Şekil 2.10-b’de bir hücrenin formu gösterilmiştir. Doğruluk tablosundan (LUT) elde edilen kuralın notasyonu Şekil 2.10-c’de verilmiştir. Bu notasyona göre doğruluk tablosunun çıkış sütündeki değerler aşağıdan yukarıya doğru sırayla yazıldığında elde edilen ikili sayının onluk karşılığı kuralın ismini belirler. Şekil 2.10-d’de verilen bağlantı notasyonu ise hücrenin hangi hücrelerle komşu olduğunu belirler.

Şekil 2.11’da $\{-1, 0, 1, 2\}$ notasyonuna göre bağlanmış 1-boyutlu hücresel otomat için bir örnek verilmiştir. Şekil 2.12’de ise 2-boyutlu ve 3-boyutlu hücresel otomatlar için örnekler verilmiştir.

Corsonello ve arkadaşları [25], "CAREM" (Cellular Automata Reconfigurable Machine) adını verdikleri bir sistem geliştirmişlerdir. Bu sistem iki boyutlu hücresel otomatların FPGA’da etkin bir şekilde gerçekleştirilmesini sağlar. Sistemin blok diyagramı Şekil 2.13’de gösterilmiştir. Sistemin çekirdeği "CAREM" kartıdır. Veri girişi ve çıkışların görüntülenmesi için "CAREM" kartı bir arayüz sistemi ile

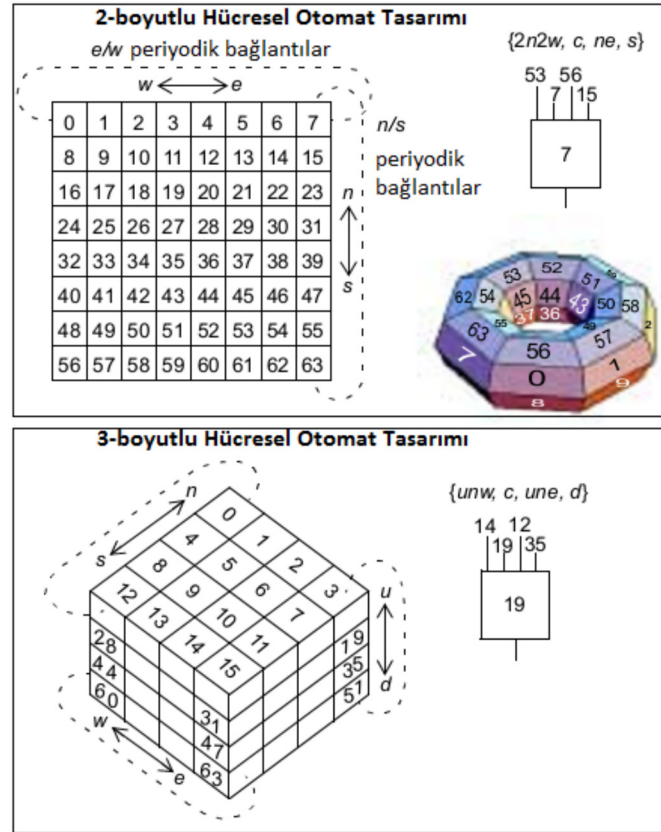


Şekil 2.10: Shackleford ve arkadaşlarının FPGA’da hücresel otomat gerçeklemeleri [22].

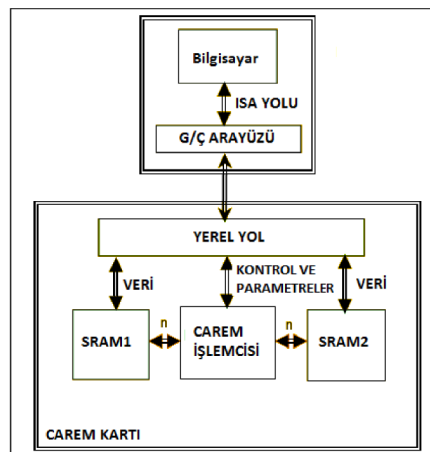


Şekil 2.11: Shackleford ve arkadaşlarının 1-boyutlu hücresel otomat için yaptıkları örnek tasarım [22].

haberleşir. Bu arayüz sistemi bir bilgisayar ve özel olarak tasarlanmış bir giriş/çıkış arayüzünden oluşmaktadır. Sistemin çalışması üç ana fazdan oluşmaktadır. Birinci fazda, hücresel otomatın başlangıç durumu bilgisayarın hafızasından karttaki SRAM’e (Static RAM) transfer edilir. Bunun dışında, hesaplamalar için gereken, hücre sayısı, zaman adımı sayısı, kural parametreleri gibi parametreler "CAREM" işlemcisine gönderilir. İkinci fazda, "CAREM" işlemcisi hücresel otomatın değişimini belirlenen kurala göre hesaplar. Bütün değişim hesaplandıktan sonra üçüncü fazda, karttaki SRAM’deki çıkış verileri bilgisayara gönderilir ve ekranda gösterilir.



Şekil 2.12: Shackleford ve arkadaşlarının 2-boyutlu ve 3-boyutlu hücresel otomatlar için yaptıkları örnek tasarımlar [22].



Şekil 2.13: Corsonello ve arkadaşlarının 2-boyutlu hücresel otomatlar için geliştirdikleri "CAREM" sisteminin blok diyagramı [25].

3. HAFIZALI HÜCRESEL OTOMAT

Standart hücresel otomatların, hücrelerinin bir önceki durumlarının değerine bağlı olarak yeni durumları belirlenir. İkinci bölümde bahsettiğimiz gibi Edward Fredkin tersinir olmayan kurallar için yeni bir yöntem ile tersinir kurallar elde etmeyi bulmuştur. Bu sistem $(\mathbb{Z}, V, Q, f)$ 4-lüsü ile tanımlanabilir. Burada komşuluk vektörü $V = (1, 0, -1)$ 'dir. Durum kümesi $Q = \{0, 1\}$ 'dir. Sistemin dinamik davranışı ise 3.1 eşitliği ile gösterebilir.

$$\alpha_i^{T+1} = f(\alpha_{i-1}^T, \alpha_i^T, \alpha_{i+1}^T) \oplus \alpha_i^{T-1} \quad (3.1)$$

Elde edilen yeni kural tersinirdir. Bu hücresel otomatlara ikinci dereceden hücresel otomatlar denir. Bu kurallara başka bir açıdan bakarsak aslında hücresel otomatın hücrelerinin bir sonraki durumu, sadece şu andaki durumunlarına değil aynı zamanda bir önceki durumunlarına da bağlıdır. Böylece standart hücresel otomat kurallarına hafıza da eklenmiş olur.

Ramon Alonso-Sanz ve Margarita Martin [27], 2000'li yılların başlarında Fredkin'in sunduğu hücresel otomattan farklı bir hafızalı temel hücresel otomat önermişlerdir. Bu hafızalı hücresel otomatın komşuluk kümesi ve durum kümesi aynıdır. Dinamik davranışı ise (3.2) ile gösterilir. Burada s_i^T fonksiyonu sıklık fonksiyonudur ve örnek olarak α_i hücresinin T zaman adımına kadar aldığı bütün değerlerin (0 veya 1) en sık olanı olarak tanımlanmıştır.

$$\alpha_i^{T+1} = f(s_{i-1}^T, s_i^T, s_{i+1}^T) \quad (3.2)$$

3.1 R. Alonso-Sanz'ın Hafızalı Temel Hücresel Otomatı

Hafızalı hücresel otomatı en genel anlamda $(\mathbb{Z}^d, V, Q, f, S, M)$ 6-lısı ile tanımlanabilir. Burada komşuluk vektörü $V = (i_1, i_2, \dots, i_l)$ 'dir. Q elemanları, hücrelerin alabileceği değerler olan sonlu durum kümesidir. f yerel geçiş fonksiyonu $f : Q^l \rightarrow Q$ şeklinde

tanımlıdır. Burada l , V komşuluk vektörünün eleman sayısı ($l = |V|$). M ise hafıza komşuluk vektörlerinden oluşan bir kümedir.

$$H_n = \{t_1, t_2, \dots, t_j\} \in M, t_k \in \{0, 1, \dots\}, k = 1, 2, \dots, j$$

S ise hafıza fonksiyonlarının oluşturduğu bir kümedir. $s_n \in S$ hafıza fonksiyonu,

$$s : Q^j \rightarrow Q$$

şeklinde tanımlıdır. Burada j , H_n hafıza komşuluk vektörünün eleman sayısıdır ($j = |H_n|$). Sistemin dinamik davranışı ise (3.3) ve 3.4 eşitliklerinde verilmiştir.

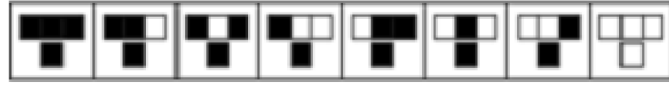
$$\alpha_i^{T+1} = f(m_{i-i_1}^T, m_{i-i_2}^T, \dots, m_{i-i_l}^T) \quad (3.3)$$

$$m_i^T = s_T(\alpha_i^{T-t_1}, \dots, \alpha_i^{T-t_j}), t_k \in H_T, k = 1, \dots, j \quad (3.4)$$

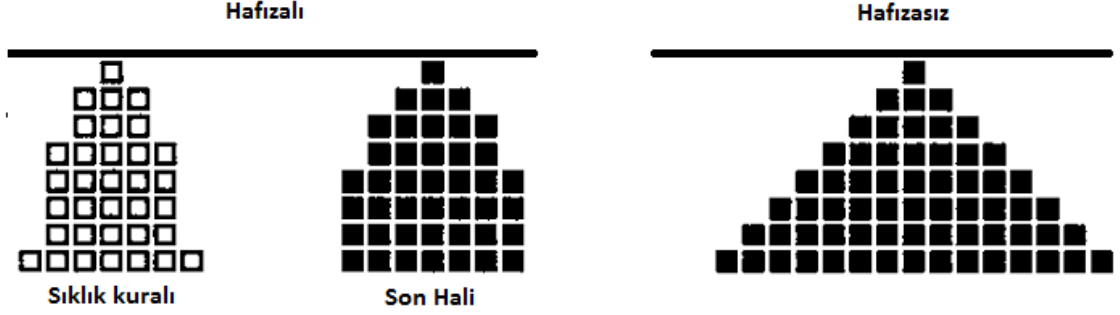
Bu tanımlamardan anlaşılacağı gibi hafıza komşuluğu ve hafıza fonksiyonu her zaman adımında farklı olabilir.

Bir önceki bölümde örnek olarak verilen sıklık fonksiyonunda T zaman adımı için hafıza komşuluk vektörleri $H_T = (0, \dots, T) \in M^*$ dir. Her zaman adımında hafıza komşuluğu dolayısıyla da hafıza fonksiyonu değişmektedir. Böylece her zaman adımında hücrelerin önceki bütün zaman adımındaki durumlarına bakılır.

Standart hücresel otomatta (hafızasız) Kural 254'ü şu şekilde tanımlayabiliriz: Bir hücrenin komşuluğundaki hücrelerin en az bir tanesinin değeri 1 ise hücrenin sonraki adımdaki değeri 1'dir, komşuluğundaki bütün değerler sıfır ise sonraki adımdaki değeri 0'dır (Şekil 3.1). Kural 254'e sahip bir standart hücresel otomatta her zaman adımında değeri 1 olan hücrelerin sayısının 2 arttığını görürüz. Kural 254'e sahip hafızalı hücresel otomatta ise dinamik standart hücresel otomata göre daha yavaştır. Çünkü hücrenin sonraki değerinin 1 olabilmesi, hücrenin önceki değerlerinin en az yarısının 1 olmasına bağlıdır (Şekil 3.2). Kural 254 için 3. zaman adımına kadar standart hücresel otomatın ve hafızalı hücresel otomatın durumları aynıdır. 3. adımdan sonra durumları değişmektedir [27].

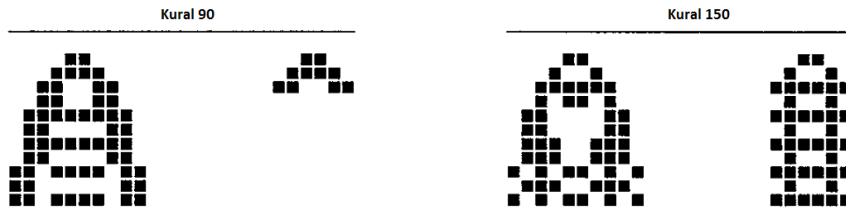


Şekil 3.1: Kural 254 [17].



Şekil 3.2: Kural 254 için hafızalı ve hafızasız hücresel otomatların karşılaştırılması [27].

Standart hücresel otomatta eklenebilir özelliğine sahip kurallardan olan Kural 90 ve Kural 150, hafızalı hücresel otomatta eklenebilir değildir. Eklenebilir özelliğine sahip olup olmadığı hücresel otomatın başlangıç durumundaki 1 değerine sahip olan hücrelerinin, tek 1 değerine sahip olan başlangıç durumlarına ayırdırdıktan sonra ve bu ayırştılan başlangıç durumlarına sahip hücresel otomatların örüntülerinin süperpozisyonunun ayırştırılmamış başlangıç koşuluna sahip hücresel otomatın örüntüsü ile aynı olup olmamasına bakılarak kontrol edilebilir. Şekil 3.3'ten görüldüğü gibi Kural 90 ve Kural 150 hafızalı hücresel otomatta eklenebilirlik özelliklerini kaybetmişlerdir. Şekil 3.3'te soldaki örüntüler iki tane 1 değeri olan başlangıç durumuna sahip hafızalı hücresel otomatların zamandaki değişimini gösterirken, sağdaki örüntüler bu iki 1 değerine sahip hücreden ayrı ayrı tek 1 değerine sahip başlangıç durumu elde edilmiş hücresel otomatların örüntülerinin süperpozisyonu sonucunu göstermektedir [27].



Şekil 3.3: Kural 90 ve Kural 150'nin hafızalı hücresel otomattaki durumları [27].

Bahsedilen sıklık fonksiyonunun (s_i^T) kullanıldığı hafızalı hücresel otomatlar, standart hücresel otomatlara göre daha korunumlu bir yapıya sahiptir. Sınıf 3'e ait kaotik hücresel otomatlar, hafızalı hücresel otomatta sınıf 2 davranışı sergilerler.

3.2 Minimal Hafızalı Sıklık Fonksiyonu

Bundan önce bahsedilen sıklık fonksiyonu, hücrenin bütün önceki değerlerine bakarak en sık alınan değeri çıkış olarak veren bir fonksiyondur (3.5). Hafıza komşulukları $H_T = (0, 1, 2, \dots, T)$ 'dir.

$$m_i^T = s_T(\alpha_i^1, \alpha_i^2, \alpha_i^3, \dots, \alpha_i^T) \quad (3.5)$$

Bu bölümde ise farklı bir sıklık fonksiyonundan bahsedeceğiz. Bu sıklık fonksiyonu yine Ramon Alonso-Sanz tarafından 2005 yılında önerilmiştir. Bu fonksiyonda hücrenin bütün önceki değerleri yerine, son üç zaman adımındaki değerlerine bakılır. Bu fonksiyona minimal hafızalı sıklık fonksiyonu adı verilir (3.6) [28].

Burada M kümesinin elemanı olarak sadece bir hafıza komşuluk kümesi vardır. $H = \{0, 1, 2\}$ 'dir. S hafıza fonksiyonları kümesinin de bir elemanı bulunur. Minimal hafızalı sıklık fonksiyonu adı verilen bu hafıza fonksiyonu

$$s : Q^3 \rightarrow Q$$

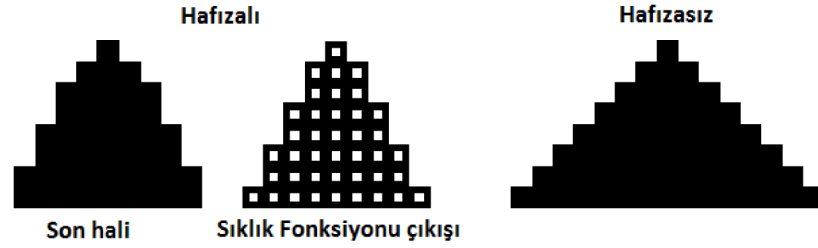
şeklinde tanımlıdır. Dinamiğini oluşturan eşitlikler ise (3.6) ve (3.7)'de verilmiştir.

$$m_i^T = s(\alpha_i^{T-2}, \alpha_i^{T-1}, \alpha_i^T), T = 2, 3, \dots \quad (3.6)$$

$$m_i^T = \alpha_i^T, T = 0, 1 \quad (3.7)$$

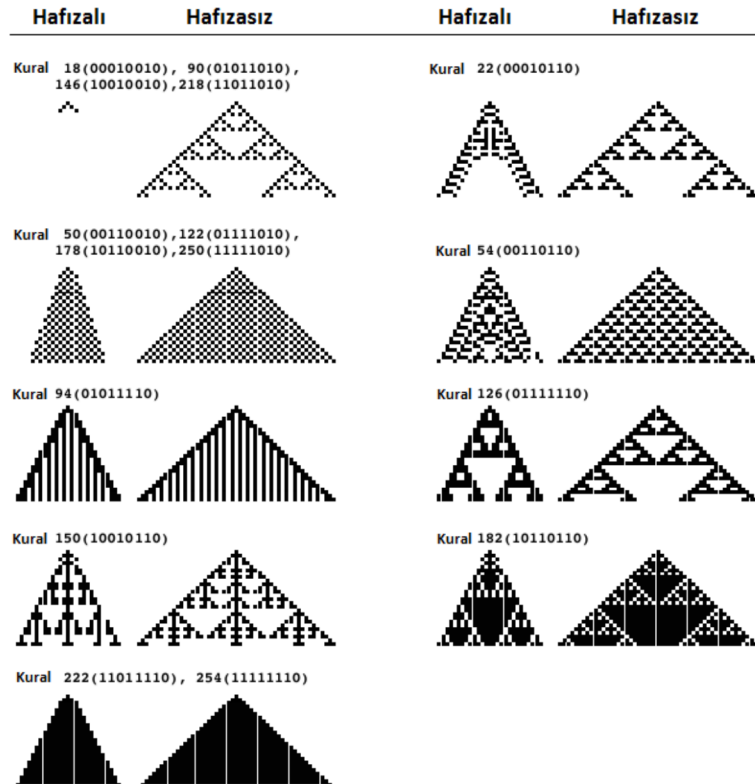
Hafıza fonksiyonu olarak minimal hafızalı sıklık fonksiyonuna sahip hücresel otomatta ilk üç zaman adımı için hafızadan söz edemeyiz. Hafıza, üçüncü adımdan sonra etkili olur. Üçüncü adıma kadar hafızasız hücresel otomat gibi çalışır. Yani sadece sahip olduğu kurala göre hücrelerini günceller.

Şekil 3.4'te Kural 254'e sahip, yeni sıklık fonksiyonunu kullanan hafızalı hücresel otomatın zamanda değişimi verilmiştir. Bir önceki sıklık fonksiyonuna sahip hafızalı hücresel otomat gibi standart hücresel otomata göre daha yavaştır.



Şekil 3.4: Kural 254'ün yeni sıklık fonksiyonuna sahip hafızalı hücresel otomattaki durumu [28].

Şekil 3.5 yeni sıklık fonksiyonunun legal temel hücresel otomat kurallarına etkisini göstermektedir. Hafızanın, Kural 18, 90, 146 ve 218'e etkisi açık bir şekilde görülmektedir. Dördüncü zaman adımından sonraki adımlarda bütün hücreler 0 değerini almaktadır. Diğer legal kuralların hafızalı hücresel otomattaki durumu, standart hücresel otomattakinin sıkıştırılmışı gibidir. Genel olarak kuralların büyük bir çoğunluğunda etkinin bu yönde olduğunu söyleyebiliriz.



Şekil 3.5: Son üç duruma bakan sıklık fonksiyonunun legal temel hücresel otomat kurallarına etkisi [28].

4. HAFIZALI HÜCRESEL OTOMATIN SAYISAL GERÇEKLEMESİ

Bu çalışmada bir elemanlı hafıza fonksiyonları kümesine sahip ve o elemanı da minimal hafızalı sıklık fonksiyonu olan temel hafızalı hücresel otomat sisteminin sayısal gerçekleştirilmesi yapılmıştır. Bu fonksiyon, hücrenin son üç zaman adımındaki değerlerine bakar ve sonuç olarak hücrenin son üç adımda en çok aldığı değeri verir.

Minimal hafızalı sıklık fonksiyonunu, Wolfram notasyonuna göre yazarsak, bu fonksiyonun Kural 232'ye karşılık geldiğini görürüz(Şekil 4.1).

$$\begin{array}{cccccccc} \underline{111} & \underline{110} & \underline{101} & \underline{100} & \underline{011} & \underline{010} & \underline{001} & \underline{000} \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array} = (232)_{10}$$

Şekil 4.1: Wolfram notasyonuna göre minimal hafızalı sıklık fonksiyonu.

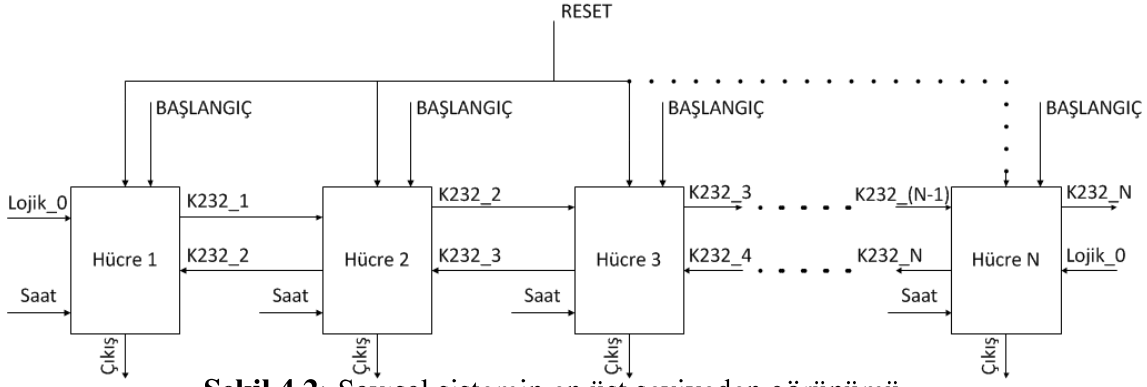
Bu bilgiler ışığında minimal hafızalı sıklık fonksiyonuna sahip hafızalı hücresel otomatın çalışmasını şu şekilde özetleyebiliriz:

- İlk üç adımda hücresel otomat aynen hafızasız hücresel otomat gibi çalışır. Hücresel otomatın sahip olduğu kurala göre hücreler güncellenir.
- Üçüncü adımdan sonra Kural 232'ye göre hücrelerin son üç durumundaki değerlerinden en çok aldığı değer sonuç olarak verilir.
- Kural 232'den elde edilen sonuçlar sanki hücrenin son değerleriymiş gibi kabul edilip hafızasız hücresel otomat gibi bu değerlere ve hücresel otomatın sahip olduğu kurala göre hücreler güncellenir.

4.1 Sayısal Sistemin Oluşturulması

Şekil 4.2'de sistemin en üst seviyeden tasarımı gösterilmiştir. Görüldüğü gibi sistem N tane hücre modülünden oluşmaktadır. Hücre modülleri bir saat işaretiyle senkron olarak çalışmaktadırlar.

Hücre modüllerinin, dört girişi ve iki çıkışı (Şekil 4.2'de hücrelerin üç çıkışı var gibi görülmesine karşın bu çıkışlardan K232_N ile gösterilen ikisi aynıdır) bulunmaktadır. Hücre modüllerinin iki girişi sırasıyla solundaki ve sağındaki hücrelerin Kural 232'yi



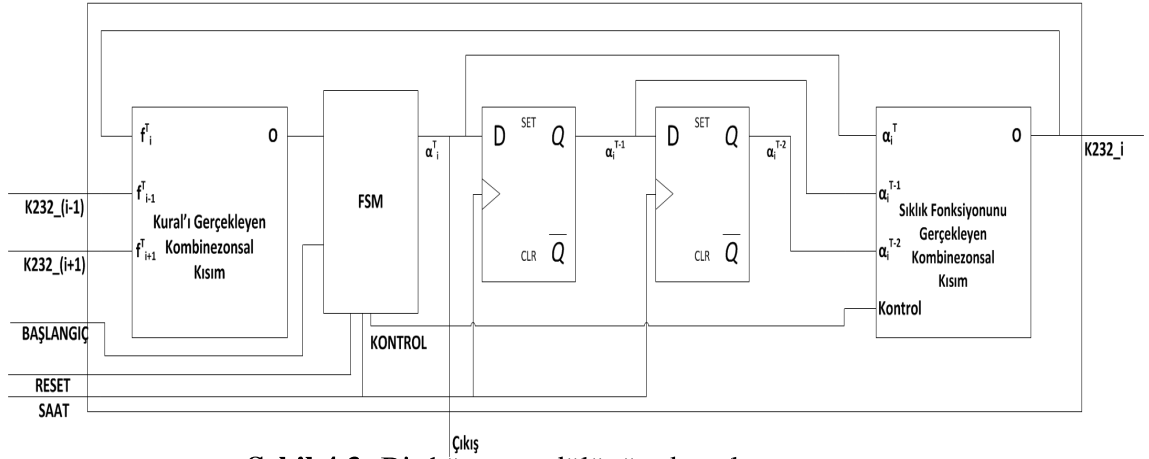
Şekil 4.2: Sayısal sistemin en üst seviyeden görünümü.

gerçekleyen kombinezonsal kısımlarının çıkışlarından gelir. Hücre modüllerinin diğer girişleri ise saat işareti ve sonlu durum makinesinden gelen "KONTROL" işaretidir. Her hücre sağdaki ve solundaki hücrelere, kendisinin Kural 232'yi gerçekleyen kombinezonsal kısmının çıkışını verir. K232_N, N.hücrenin Kural 232'yi gerçekleyen kombinezonsal kısmının çıkışını ifade etmektedir. Hücre modüllerinin diğer çıkışı ise şekilde "Çıkış" olarak belirtilmiş olan, hücrenin son adımdaki değeridir.

Oluşturulan sistemde sabit sınır koşulları uygulanmıştır. En soldaki hücre modülüne soldaki hücreden gelmesi gereken giriş lojik 0 ile sağlanmıştır. Aynı şekilde en sağdaki hücreye sağdaki hücreden gelmesi gereken giriş lojik 0 ile sağlanmıştır.

Şekil 4.3 'te bir hücre modülünün detaylı tasarımı gösterilmiştir. Modülde iki kombinezonsal kısım, bir sonlu durum makinesi ve ayrıca iki flip-flop bulunmaktadır. Sağdaki kombinezonsal kısım hücresel otomatın kuralının gerçekleşmesini sağlar. Bu kombinezonsal kısmının üç girişi ve bir çıkışı bulunmaktadır. Girişlerin birisi (s_i^T) sağdaki kombinezonsal kısımdan, diğer ikisi (s_{i-1}^T, s_{i+1}^T) ise modülün solundaki ve sağdaki hücre modüllerinin sağdaki kombinezonsal kısımlarının çıkışlarından gelmektedir. Soldaki kombinezonsal kısmın çıkışı ise "O" ile belirtilmiştir. Soldaki kombinezonsal kısmın çıkışı sonlu durum makinesine bağlıdır. Sonlu durum makinesi içinde hücrenin o anki değerini tutar. Başlangıç değerlerinin hücreye yazılması sonlu durum makinesi sayesinde yapılır. Başlangıç değerleri hücreye yazıldıktan sonra bir sonraki yükselen kenarda sonlu durum makinesinin içinden tutulan hücrenin bir önceki değeri birinci flip-flopa soldaki kombinezonsal kısmın bir saat darbesi önceki değeri olarak (α_i^{T-1}) yazılır. Aynı saat darbesinin yükselen kenarında ikinci flip-flopa ise birinci flip-flopta bir önceki saat darbesinde tutulan değer yani hücrenin

iki zaman adımı önceki değeri (α_i^{T-2}) yazılır. Sağdaki kombinezonsal kısım ise minimal hafızalı sıklık fonsiyonunun gerçekleştirilmesini sağlar. Girişleri iki flip-flobun çıkışlarına ($\alpha_i^T, \alpha_i^{T-1}, \alpha_i^{T-2}$) ve sonlu durum makinesinin "CTRL" çıkışına ve hücrenin o anki değerini belirten çıkışına bağlanmıştır. Çıkışı ise daha önceden belirttiğimiz gibi soldaki kombinezonsal kısmın bir girişine bağlanmıştır. Aynı zamanda bu çıkış modülün sağındaki ve solundaki hücre modüllerinin soldaki kombinezonsal kısımlarının girişlerine bağlıdır.



Şekil 4.3: Bir hücre modülünün detaylı tasarımı.

Sonlu durum makinesi, RESET sinyali geldikten sonra ilk iki zaman adımında "CTRL" çıkışını lojik 0, sonraki adımlarda lojik 1 yapmak için oluşturulmuştur. Şekil 4.4'de sonlu durum makinesi verilmiştir. "STATE 0" ilk durumu göstermektedir. RESET sinyali gelene kadar bu durumda kalmaktadır. "STATE 0" durumunda "RESET" sinyali geldiğinde "STATE 1" durumuna geçilir. Görüldüğü gibi "CTRL" çıkışı "STATE 3" durumuna kadar lojik 0, "STATE 3" durumunda ise lojik 1 yapılmıştır.

Çizelge 4.1'de Kural 150'yi gerçekleyen kombinezonsal kısım için doğruluk tablosu verilmiştir.

Çizelge 4.2'de de minimal sıklık fonksiyonunu gerçekleyen kombinezonsal kısım için doğruluk tablosu verilmiştir. "KONTROL" girişinin lojik 0 olduğu değerler için çıkış f_i^T 'nin değerine eşittir. "KONTROL" girişinin lojik 1 olduğu değerler için ise Kural 232, gerçekleştirilmektedir.

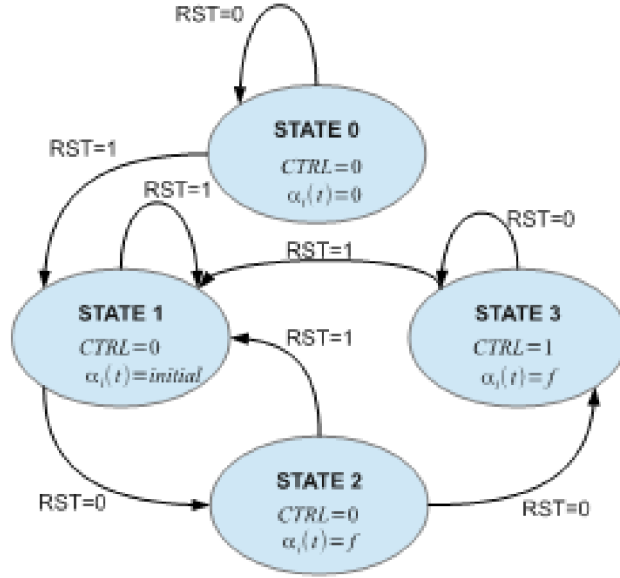
Sayısal sistem, FPGA'da (Field Programmable Gate Array) "Verilog" donanım tanımlama dili kullanılarak gerçekleştirilmiştir. Kombinezonsal kısımlar için FPGA'larda bulunan doğruluk tablosu gibi çalışan LUT'lar(Look Up Table) kullanılmıştır.

Çizelge 4.1: Kural 150'yi gerçekleyen kombinezonsal kısım için doğruluk tablosu.

m_i^T	m_{i-1}^T	m_{i+1}^T	O
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

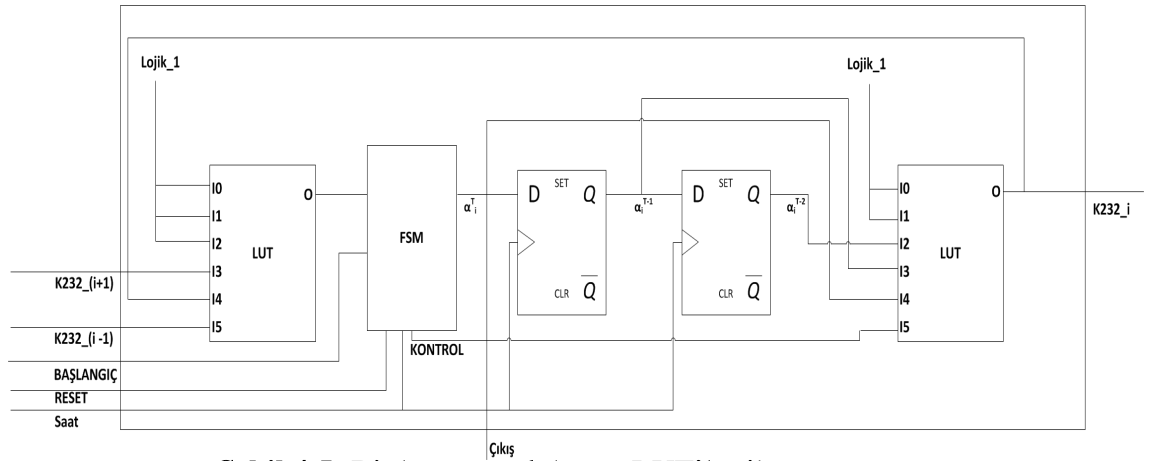
Çizelge 4.2: Minimal sıklık fonksiyonunu gerçekleyen kombinezonsal kısım için doğruluk tablosu.

KONTROL	α_i^T	α_i^{T-1}	α_i^{T-2}	O
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



Şekil 4.4: Sonlu durum makinesi.

Dolayısıyla kapı seviyesinde bir gerçekleştirme yapılmamıştır. Kullanılan FPGA ailesinin(Spartan 6) LUT'ları 6 girişlidir. Sayısal sistemin kombinezonsal kısımlarının girişlerinin sayısı sırası ile üç ve dört olduğu için LUT'ların boş kalan girişleri lojik 1'e bağlanmıştır. Şekil 4.5'te sayısal sistemin bir hücre modülünün 6 girişli LUT'lar ile oluşturulmuş tasarımı verilmiştir.



Şekil 4.5: Bir hücre modülünün LUT'lar ile tasarımı.

4.2 Benzetim Sonuçları

Tasarlanan sayısal sistemde bir hücre modülünün soldaki kombinezonsal kısmı Kural 150'yi gerçekleyecek oluşturulmuştur. Şekil 4.6-a, Şekil 4.6-b ve Şekil 4.6-c'de

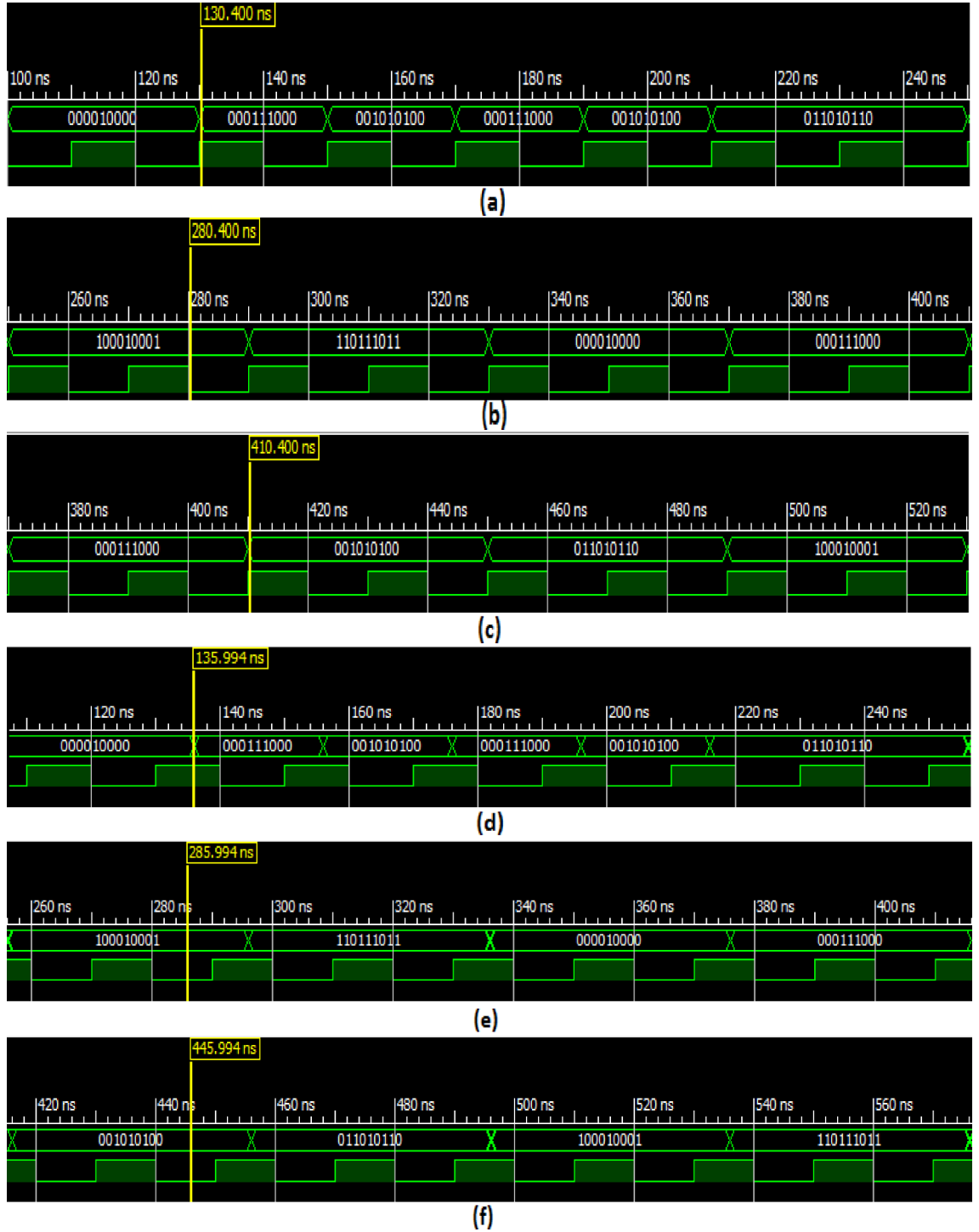
sistemin kapı gecikmeleri göz önüne alınmadan (Davranışsal Benzetim) "ISIM" programı ile yapılan benzetimin sonuçları gösterilmektedir. Sonuçların, Şekil 4.7'da verilen minifal hafızalı sıklık fonksiyonuna ve Kural 150'ye sahip hafızalı hücresel otomatın dinamik davranışı ile karşılaştırıldığında doğru olduğu görülmüştür. Şekil 4.6-d, Şekil 4.6-e ve Şekil 4.6-f'de ise kapı gecikmelerinin ve yol gecikmelerinin de göz önünde bulundurarak yapılan benzetim (Yerleştir ve Bağla işlemi sonrası benzetim) gösterilmiştir. Bu benzetim sonuçlarının da Şekil 4.7 ile karşılaştırıldığında, doğru olduğu görülmüştür.

Benzetimlerde saat ve "Çıkışlar" olmak üzere iki sonuç birlikte gösterilmiştir. Buradaki "Çıkışlar" değeri bütün hücre modüllerinin "Çıkış" değerlerinin birleşiminden oluşan ikili bir sayıdır. Davranışsal benzetimde, görüldüğü gibi "Çıkışlar" değerinin değişimi, saat işaretlerinin yükselen kenarlarında gerçekleşmektedir. Yerleştir ve bağla işlemi sonrası benzetimde ise kapı gecikmeleri de göz önünde bulundurulduğundan "Çıkışlar" değerlerinin değişimi saat işaretlerinin yükselen kenarlarından daha sonra gerçekleşmektedir.

4.3 Gerçekleme Sonuçları

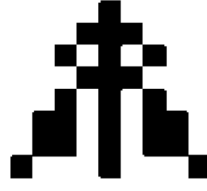
Şekil 4.8'te gerçekleştirilen sistemin blok diyagramı verilmiştir. Diyagramdan görüldüğü gibi daha önceden anlatılan hücresel otomat sisteminin çıkışları, M zaman adımı boyunca, FPGA'da oluşturulmuş M satır, N sütuna sahip RAM'e yazılır. Saat üretici bütün alt blokların kullandığı saat işaretini üretir. Kontrol bloğu RAM'e yazma, RAM'den okuma sinyallerini, RAM'in adresini ve seri gönderme bloğu için gereken sinyalleri ve yazıcıları kontrol eder. Seri gönderme bloğu ise RAM'e yazılan değerleri 8'er bitlik paketlerle bilgisayara gönderir. Bilgisayara gönderilen veriler MATLAB programı ile alınarak işlenir ve gerçekleştirilen hafızalı hücresel otomatın örüntüsü oluşturulur.

Şekil 4.9'te Kontrol bloğunu gerçekleyen algoritmik durum makinesi verilmiştir. Görüldüğü gibi Kontrol bloğu ayrı iki algoritmik durum makinesi ile tasarlanmıştır. Soldaki algoritmik durum makinesi tasarlanan hücresel otomatın çıkışlarını RAM'e yazar. Sağdaki algoritmik durum makinesi ise RAM'e yazma işlemi bittikten sonra RAM'den okuma işlemi yapar ve seri gönderme bloğunun girişine gelecek olan "GIDECEK_SEKIZLI" değerini ve "GONDER" sinyalini ayarlar.

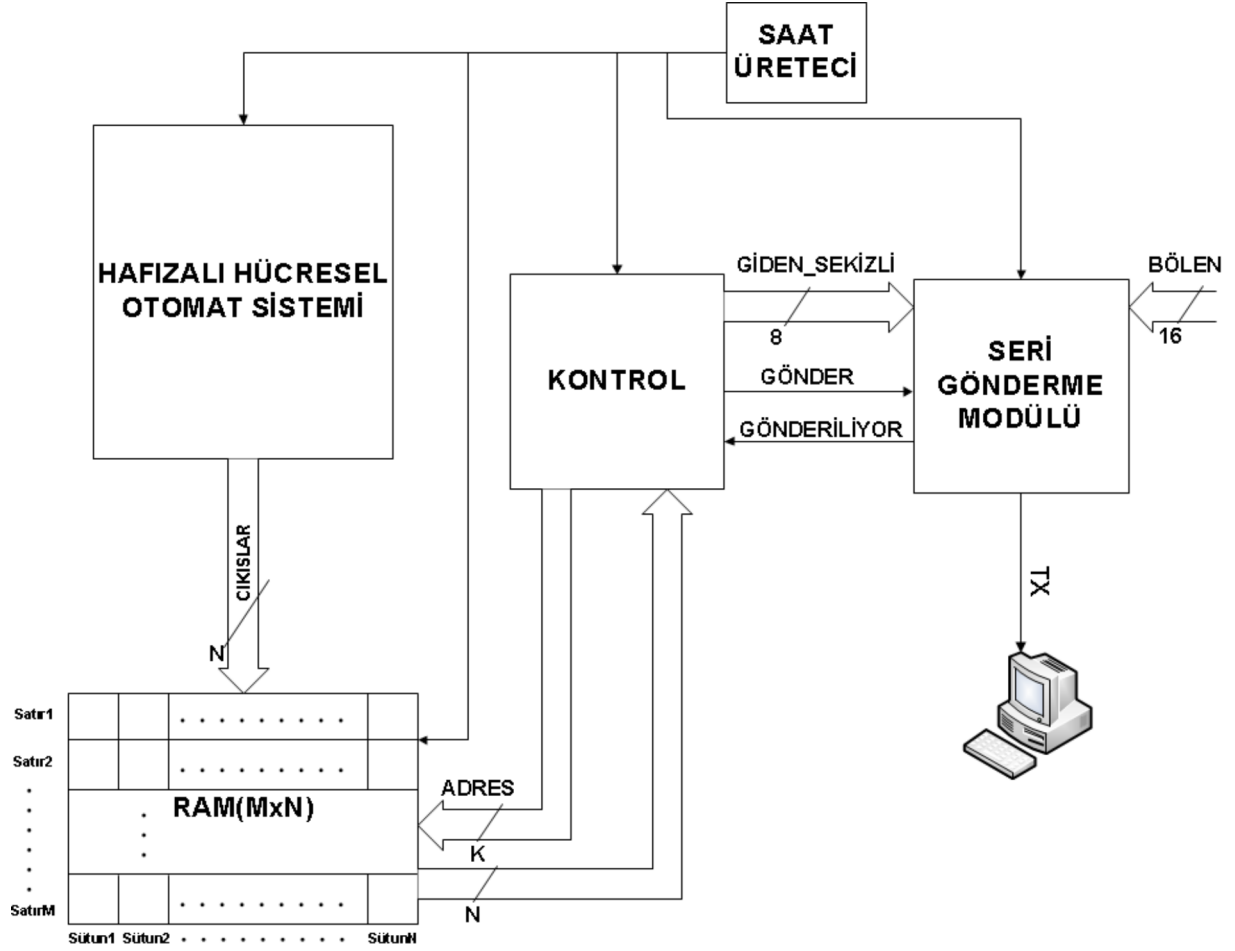


Şekil 4.6: Tasarlanan sayısal sistemin Kural 150 için (a) davranışsal benzetiminin birinci kısmı, (b) davranışsal benzetiminin ikinci kısmı, (c) davranışsal benzetiminin üçüncü kısmı, (d) yerleştir ve bağla sonrası benzetiminin birinci kısmı, (e) yerleştir ve bağla sonrası benzetiminin ikinci kısmı, (f) yerleştir ve bağla sonrası benzetiminin üçüncü kısmı.

"GIDECEK_SEKIZLI" değeri algoritmik durum makinesinde belirtilen TX_BUFFER yazmacının en düşük anlamlı sekiz bitinden oluşmaktadır. RAM'den okunan veriler



Şekil 4.7: Kural 150'ye sahip hafızalı hücresel otomatın zamanda değişimi [29].

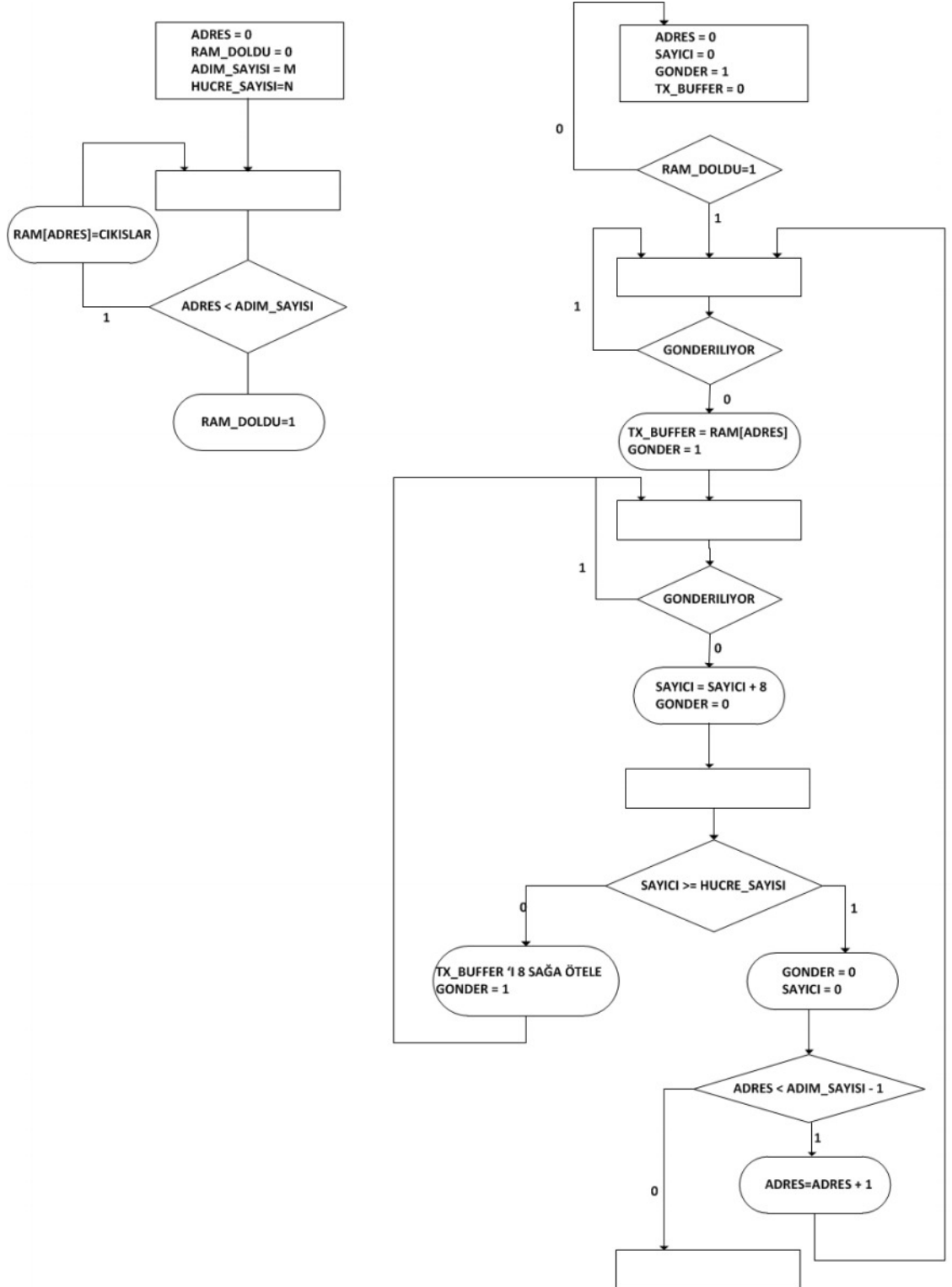


Şekil 4.8: Gerçeklenen sistemin blok diyagramı.

TX_BUFFER yazmacına yazılır ve TX_BUFFER yazmacı sekiz bit sağa ötelenerek "GIDECEK_SEKİZLİ" değeri değiştirilir.

Seri gönderme modülünün "GIDEN_SEKİZLİ", "GONDER" ve "BOLEN" olmak üzere üç girişi, "GONDERİLİYOR" ve "TX" olmak üzere iki çıkışı bulunmaktadır. "GIDEN_SEKİZLİ" girişi sekiz bitlik bir giriştir ve bilgisayara gönderilecek olan veriyi ifade eder. "GONDER" girişi verinin göndermeye hazır olduğunu belirtir. "GONDER" girişi lojik 1 ise modül, veriyi göndermeye başlar, lojik 0 ise lojik 1 olana kadar bekler. "BOLEN" girişi ise seri gönderme modülünün içerisinde

bulunan frekans bölücü modülün saat üreticinden gelen işaretin frekansını ne kadar boleceğini belirten girişidir. Bilgisayar ile haberleşme hızını ayarlamak için kullanılır.



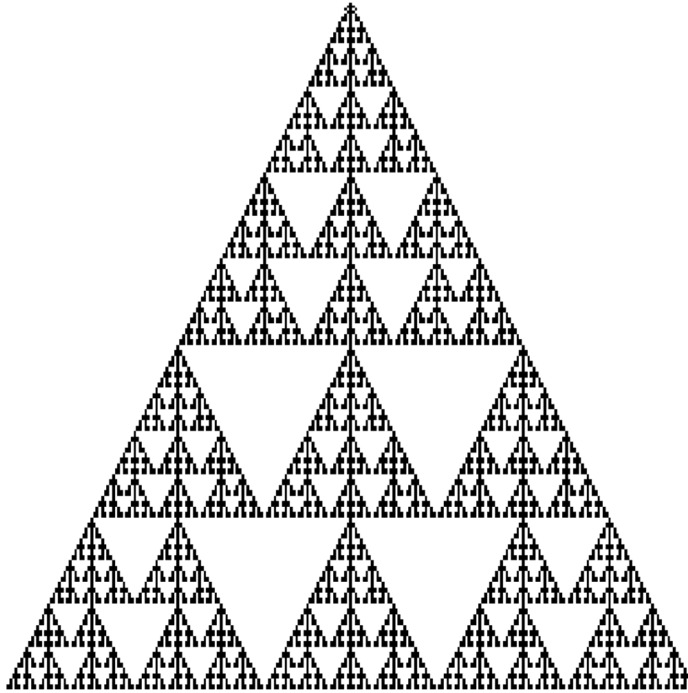
Şekil 4.9: Kontrol bloğunu gerçekleyen algoritmik durum makinesi.

"GONDERILİYOR" çıkışı modülün o anda veri gönderip göndermediğini belirten çıkışıdır. "GONDERILİYOR" çıkışı lojik 1 ise modül o anda veri gönderiyor, lojik

0 ise göndermiyor demektir. "TX" çıkışı ise verinin gönderildiği bir bitlik çıkıştır. Bilgisayarın seri portunun "RX" girişine bağlıdır.

Yapılan gerçektelemede hücre sayısı, $N=256$, zaman adımı sayısı, $M=256$ olarak alınmıştır. Hafızalı hücresel otomatın başlangıç durumu sadece 127.hücrenin değeri 1, diğer hücrelerin değeri 0 olarak alınmasıyla oluşturulmuştur. Çalışmada Digilent firmasının ATLYS geliştirme kartı kullanılmıştır. Tasarım bu kartta bulunan Xilinx firmasının Spartan-6 ailesinden olan XC6SLX45-CSG324C FPGA entegresine gömülmüştür. Saat üretici olarak kartta hazır olarak bulunan 100 MHZ lik osilatör kullanılmıştır. Yani her bir zaman adımı arasında 10 ns lik bir zaman farkı vardır. Seri gönderme modülünde frekans bölücüye "BOLEN" değeri olarak 5208 değeri girilmiştir. Böylece seri porttan bilgisayara 9600 bps hızında veri gönderilmiştir. Bilgisayara gönderilen veri MATLAB programında yazılan kısa bir kodla alınıp 256×256 'lık bir matrise dönüştürüldükten sonra resime çevrilmiştir.

Bu işlemler sonunda elde edilen örüntü Şekil 4.10'da verilmiştir. Elde edilen örüntü, beklenildiği gibi Şekil 4.7'de verilen örüntüye uygun olarak çıkmıştır.



Şekil 4.10: Gerçekteleme sonucu elde edilen 256 hücreli hafızalı hücresel otomatın 256 zaman adımı boyunca zamanda değişimi.

5. FİZİKSEL OLARAK KLONLANAMAYAN HÜCRESEL OTOMAT

5.1 Fiziksel Olarak Klonlanamayan fonksiyonlar

Fiziksel olarak klonlanamayan fonksiyonlar (PUF), fiziksel bir cihaza uygulanan girişleri, fiziksel cihazın çıkışlarına dönüştüren ve aşağıdaki özellikleri sağlayan fonksiyonlardır:

- 1- Fiziksel cihaz fonksiyonu çok kısa zaman dilimlerinde gerçekleyebilmelidir.
- 2- Fiziksel cihaza yalnızca yalnızca dışarıdan müdahale edebilen bir kullanıcı, rastgele uygulanmış girişlere karşılık düşen çıkışlardan kayda değer bir bilgi elde edemez [30].

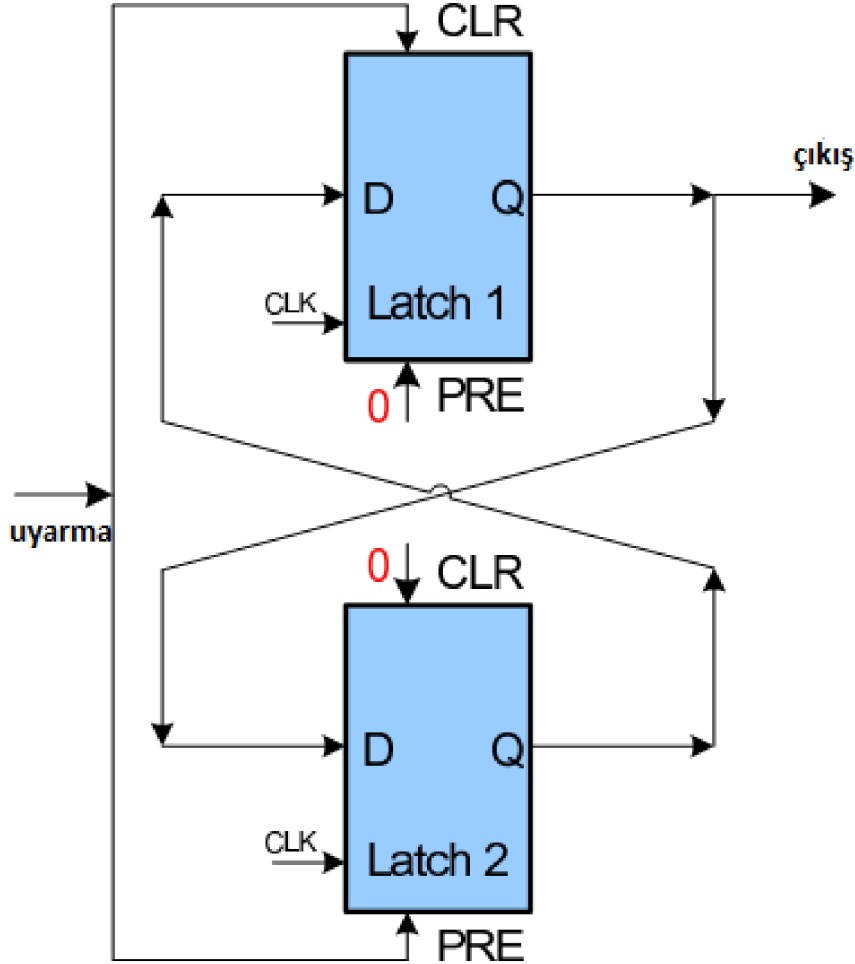
Gassend ve arkadaşlarının [30] ortaya koyduğu silikon PUF'lar, tümdevre üretim işlemlerinde kontrol edilemeyen devre karakteristiklerini kullanırlar. Bundan sonra sadece silikon PUF'lardan bahsedileceği için silikon PUF'lar çalışmanın geri kalan kısmında kısaca PUF olarak ifade edilmiştir.

Aynı silisyum puldan üretilen tümdevreler dahi gecikme sapmalarına sahiptir. Maskeleme işleminin sapmalarından dolayı, bir kalıp boyuncaki bütün tümdevrelerin gecikmeleri değişir. İşlem sıcaklığı ve basınç sapmaları, farklı gecikmelere neden olan üretim sapmalarına örnek olarak verilebilir. Bu gecikme sapmaları %5 veya daha fazla olabilir. Farklı kalıplardan üretilmiş, aynı yolların veya cihazların gecikme sapmaları, Gaus dağılımı ve diğer olasılık dağılım fonksiyonlarıyla modellenir. Bu gecikme sapmaları tümdevre boyutları küçüldükçe artmaktadır [30].

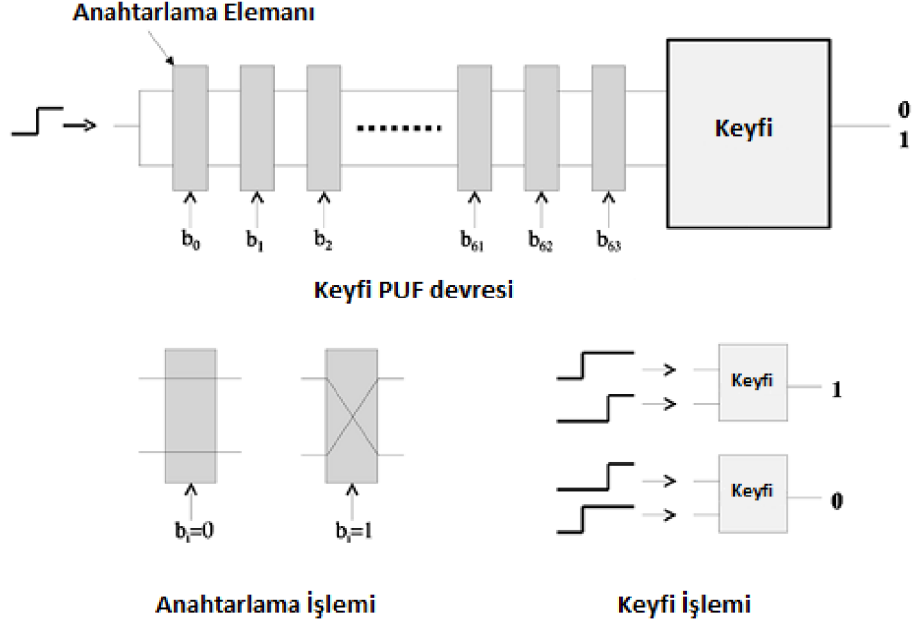
Gecikme sapmalarına, sıcaklık ve yaşlanma gibi çevresel faktörler de etki eder. Bunlardan ortam sıcaklığı en çok etkili olanıdır [30].

PUF'lar tümdevre kimlik doğrulama ve IP (intellectual property) tasarım korumalarında özellikle kullanılmaktadır. IP donanım tasarımcıları tasarımlarının kötü amaçlı kullanıcıların ellerine geçmesini önlemek için tasarımın yükleneceği tümdevre dışında başka bir tümdevrede çalışmasını PUF tasarımlarını kullanarak engellerler. Çünkü PUF'lar tümdevreye özel bir gizli anahtar oluşturulmasını sağlarlar [31].

PUF tasarımının en ilginç olanları SRAM(Statik RAM), kelebek, keyfi ve halka osilatörü tabanlı PUF'lardır [32]. SRAM PUF'larda herhangi bir tümdevreye güç verildiği anda içindeki SRAM'ın hücrelerinin değerlerinin rastgele olması (farklı tümdevreler için farklı değerler) özelliği kullanılır [33]. Kelebek PUF'ta ise FPGA'da başlangıç anında (güç verilme anı) SRAM gibi davranan yapılar oluşturulmuştur [34]. Şekil 5.1'de kelebek PUF'un yapısı gösterilmiştir. Keyfi PUF, sonu keyfi olarak belirlenen gecikme yollarından oluşur [31]. Şekil refseki52'de Keyfi PUF'ın yapısı gösterilmiştir. Halka osilatörü tabanlı PUF ise birbirine eş halka osilatörlerinden oluşur. Bu tip PUF'taki ana fikir, halka osilatörlerinin herbirinin üretim varyasyonlarından dolayı farklı bir frekansa sahip olmasıdır [35]. Şekil sekil53'te halka osilatör tabanlı PUF'ların yapısı verilmiştir.



Şekil 5.1: Kelebek PUF'ların yapısı [33].

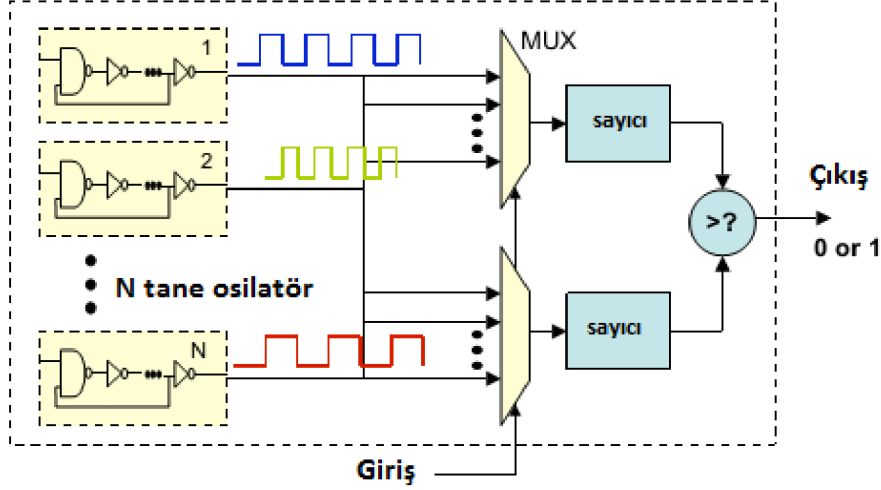


Şekil 5.2: Keyfi PUF'ların yapısı [31].

5.2 Fiziksel Olarak Klonlanamayan Hücresel Otomat

Sayısal sistemlerle çalışırken, hayat deterministik ve ayrık olduğundan sistemden rastgele sonuçlar elde etmek mümkün değildir. Sayısal sistemlerle tasarlanan rastgele sayı üreteçleri de aslında deterministik algoritmalarla oluşturulan ve istatistiksel olarak gerçek rastgele sayılara yaklaşık olan sahte rastgele sayı üreteçleridir. Ravasz ve arkadaşları gerçek rastgeleliliği elde edebilmek için "Hücresel Sinir Ağları Evrensel Makinesi" (CNN-UM) ile gerçekledikleri hücresel otomat sisteminin çıkışlarına CNN-UM'nin doğal gürültüsünü XOR işlemi ile uygulamış ve başarılı olmuşlardır [37].

Bu çalışmada, bahsedilen rastgeleliliği elde edebilmek için "Fiziksel Olarak Klonlanamayan Hücresel Otomat" (FKHO) adı verilen yeni bir sistem önerilmiştir. d -boyutlu bir FKHO sistemi, üçüncü bölümde hafızalı hücresel otomat sistemi tanımlanırken kullanılan $(\mathbb{Z}^d, V, Q, f, S, M)$ 6-lısı ile tanımlanabilir. Burada komşuluk $V = (i_1, i_2, \dots, i_l)$ 'dir. Q elemanları, hücrelerin alabileceği değerler olan sonlu durum kümesidir. f yerel geçiş fonksiyonu $f : Q^l \rightarrow Q$ şeklinde tanımlıdır. Burada l, V



Şekil 5.3: Halka osilatör tabanlı PUF’ların yapısı [35].

komşuluk vektörünün eleman sayısıdır ($l = |V|$). M ise elemanları hafıza komşuluk vektörleri olan bir kümedir.

$$H_n = (t_1, t_2, \dots, t_{j-1}) \in M$$

olmak üzere $t_k \in \{0, 1, 2, 3, \dots\}$, $k = 1, 2, \dots, j-1$ ’dir ve t_k ’lar rastgele sayılardır.

S kümesinin elemanları hafıza fonksiyonlarıdır. Bu elemanlardan birisi olan s_n hafıza fonksiyonu,

$$s_n : Q^j \rightarrow Q$$

şeklinde tanımlıdır. Burada j , H_n hafıza komşuluk vektörlerinin eleman sayısının bir fazlasıdır ($j = |H_n| + 1$). Sistemin dinamik davranışı ise (5.1) ve (5.2) eşitliklerinde verilmiştir.

$$\alpha_i^{T+1} = f(m_{i-i_1}^T, m_{i-i_2}^T, \dots, m_{i-i_m}^T) \quad (5.1)$$

$$m_i^T = s_T(\alpha_i^T, \alpha_i^{T-t_1}, \alpha_i^{T-t_2}, \dots, \alpha_i^{T-t_{j-1}}), \quad s_T \in S, \quad t_k \in H_T, \quad k = 1, \dots, j-1 \quad (5.2)$$

Bu çalışmada 1-boyutlu, 3-komşuluklu, komşuluk kümesi $V = \{1, 0, -1\}$ olan, iki durumlu, durum kümesi $Q = \{0, 1\}$, hafıza komşuluk vektörleri $H_T = \{0, t_1, t_2\}$, $t_1, t_2 \in$

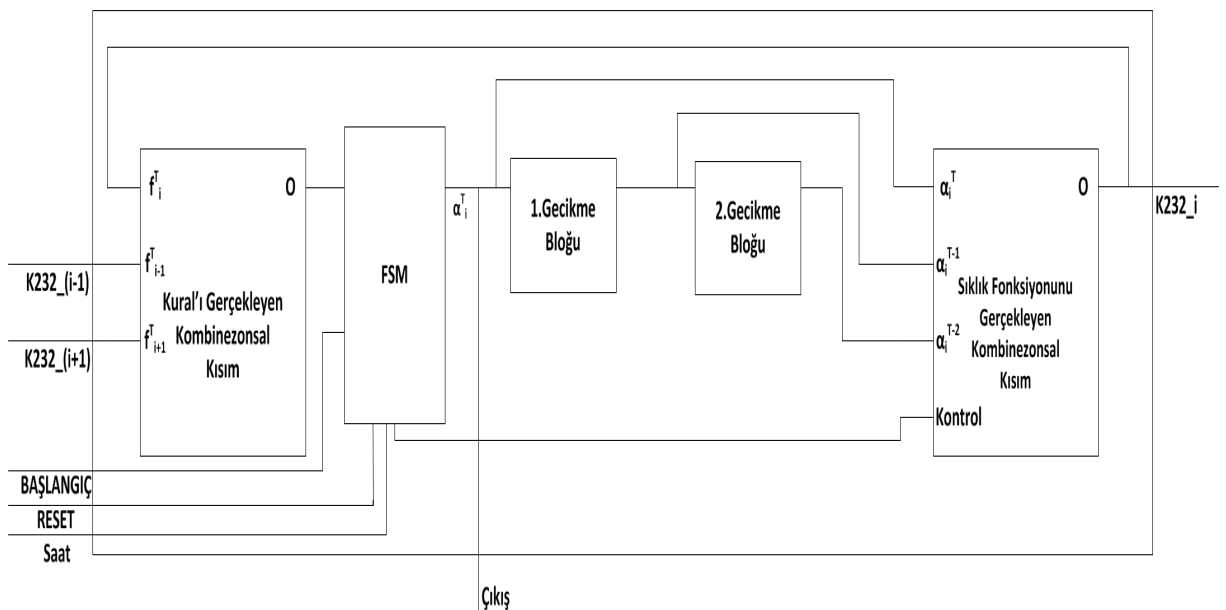
$\{1,2\}$ hafıza fonksiyonları minimal hafızalı sıklık fonksiyonları olan ve Kural 150 yerel geçiş fonksiyonuna sahip bir FKHO sistemi oluşturulmuştur. Sistemi oluşturan hafıza fonksiyonları (5.3) eşitliğinde verilmiştir.

$$m_i^T = s_T(\alpha_i^T, \alpha_i^{T-t_1}, \alpha_i^{T-t_2}) \quad (5.3)$$

Burada $t_1, t_2 \in \{0, 1, 2\}$ olmak üzere zamanla değişebilen rastgele sayılardır.

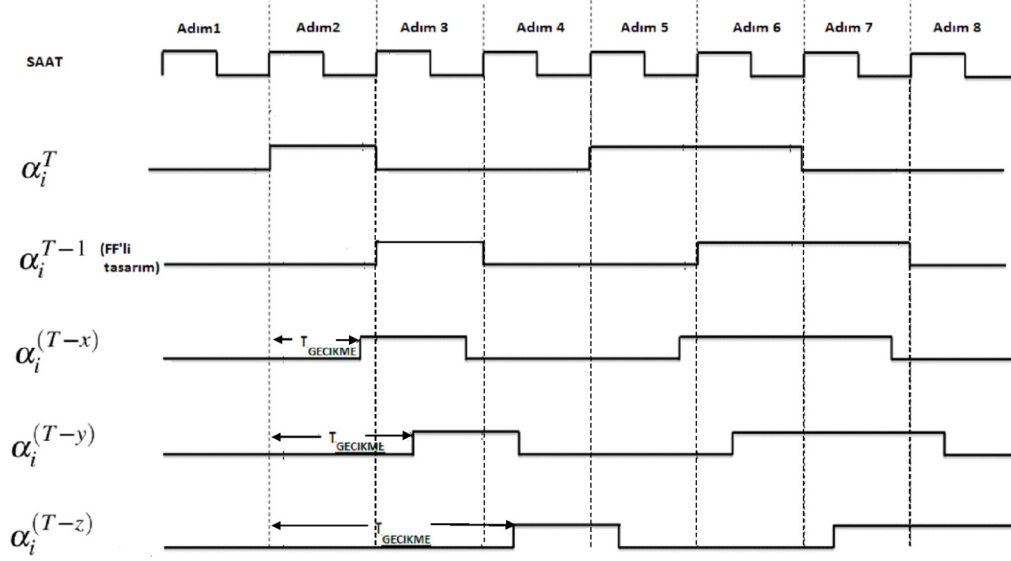
Bu sistemin oluşturulması için, dördüncü bölümde anlatılan hafızalı hücresel otomat sayısal sisteminde hücrelerin bir zaman adımı önceki ve iki zaman adımı önceki değerlerini ($\alpha_i^{T-1}, \alpha_i^{T-2}$) tutmak için kullanılan flip-floplar yerine, gecikme blokları kullanılmaktadır. Çevresel etkenlerden dolayı gecikme değerleri sürekli aynı değerleri vermeyip farklı gecikme değerleri verebilir. Dolayısıyla gecikme bloklarının çıkışlarında, minimal sıklık fonksiyonunda kullanılan bir zaman adımı önceki veya iki zaman adımı önceki değerler beklenirken, farklı zaman adımıdaki değerler görülebilir.

Şekil 5.4'te sistemin bir hücre modülünün blok diyagramı verilmiştir. Görüldüğü gibi daha önceden gerçekleştirilen hafızalı hücresel otomat sisteminden tek farkı iki flip-flop yerine gecikme bloklarının kullanılmasıdır. Bloğun girişine gelen "KONTROL" işareti yine sonlu durum makinesinden gelmektedir. Sonlu durum makinesi de hafızalı hücresel otomat sayısal sisteminin sonlu durum makinesi ile aynıdır.



Şekil 5.4: FKHO sisteminin bir hücre modülünün blok diyagramı.

Şekil 5.5'te verilen zaman diyagramında FKHO'nun gecikme bloklarının gecikme sürelerine bağlı olarak bir hücrenin durumları gösterilmiştir. Burada T_{SAAT} saat periyodunu, $T_{GECIKME}$ hücrenin bir önceki değerini elde etmek için oluşturulan 1.gecikme bloğundan elde edilen gecikme değerini ifade eder. Hücrenin bir önceki durumu ile oluşturulan sistemdeki 1. gecikme bloğunun çıkışındaki değerleri karşılaştırmak için saat işaretinin yükselen kenarındaki durumlara bakılır.



Şekil 5.5: FKHO sistemindeki gecikme bloklarının değerlerine göre bir hücrenin durumları.

Burada $x, y, z \in \{0, 1, 2\}$ 'dir. $T_{GECIKME}$ değerine göre bu değişkenlerin değerleri belirlenir.

$\alpha_i^{(T-x)}$ ile belirtilen değer, hücrenin $T_{GECIKME} < T_{SAAT}$ koşulu için geçerli olan 1.gecikme bloğunun çıkışındaki değerini belirtmektedir. Saatin yükselen kenarlarında bu değer, hücrenin bir önceki zaman adımındaki değerlerden farklı olduğu görülür. Bu koşul altında minimal hafızalı sıklık fonksiyonuna göre hücrenin bir önceki zaman adımındaki değer beklenirken, 1.gecikme bloğunun çıkışında hücrenin o anki değeri elde edilir.

$\alpha_i^{(T-y)}$ ile belirtilen değer, hücrenin $T_{SAAT} < T_{GECIKME} < 2T_{SAAT}$ koşulu için geçerli olan 1.gecikme bloğunun çıkışındaki değerini belirtmektedir. Saatin yükselen kenarlarında bu değer, hücrenin bir önceki zaman adımındaki değerler ile aynı

olduğu görülür. Bu koşul altında elde edilen sistemin çıkışlarının bir önceki bölümde bahsedilen hafızalı hücresel otomat sisteminin çıkışlarından bir farkı yoktur.

$\alpha_i^{(T-z)}$ ile belirtilen değer, hücrenin $T_{GECIKME} > 2T_{SAAT}$ koşulu için geçerli olan 1.gecikme bloğunun çıkışındaki değerini belirtmektedir. Saatin yükselen kenarlarında bu değer, hücrenin bir önceki zaman adımındaki değerlerden farklı olduğu görülür. Bu koşul altında minimal hafızalı sıklık fonksiyonuna göre hücrenin bir önceki zaman adımındaki değer beklenirken, 1.gecikme bloğunun çıkışında, hücrenin iki zaman adımı önceki değeri elde edilir.

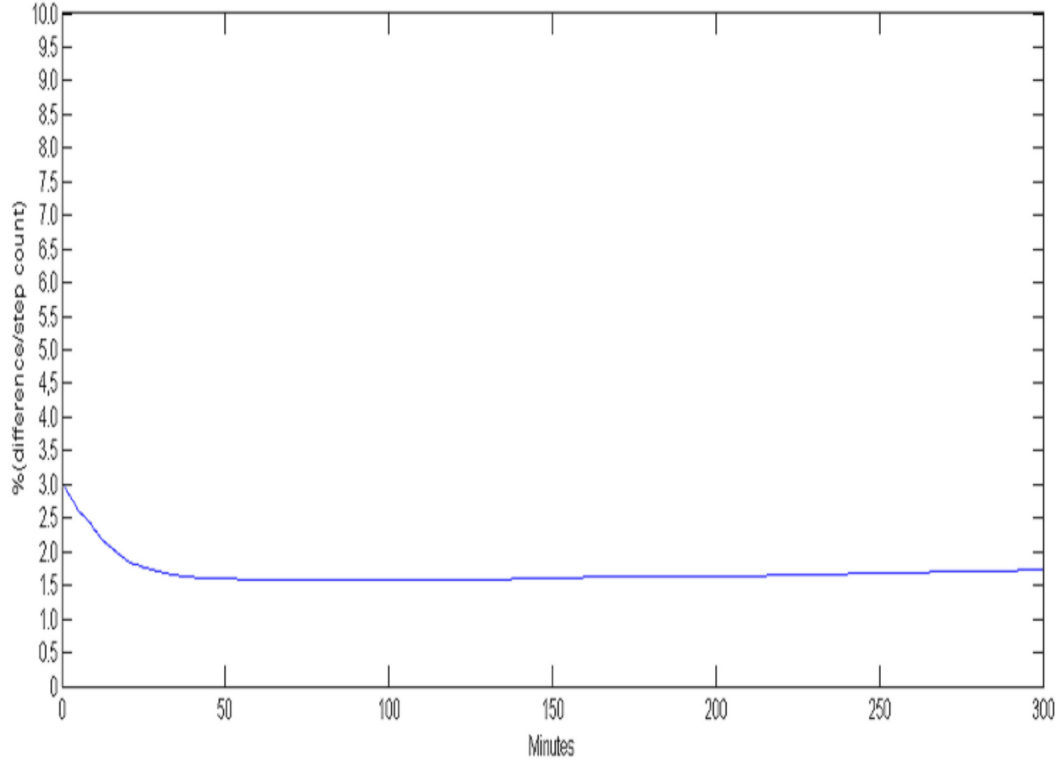
Böylece FKHO sisteminin gerçekten klonlanamaz olması için $T_{GECIKME} < T_{SAAT}$ veya $T_{GECIKME} > 2T_{SAAT}$ koşulları sağlanmalıdır.

5.3 Deneysel Sonuçlar

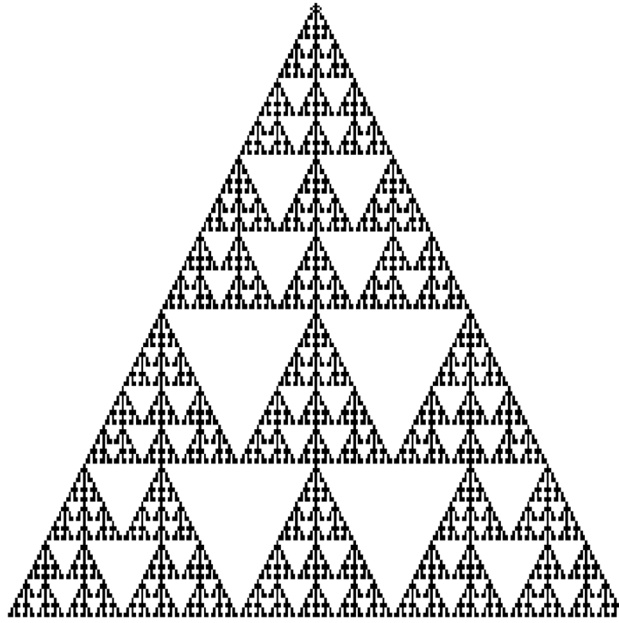
Şekil 5.6'da, 9 hücreli 1-boyutlu 150 kuralına sahip hücresel otomatın dördüncü hücresinin, aynı başlangıç koşullarında, hafızalı hücresel otomat sayısal sisteminde iki zaman adımı önceki değerinin ($\alpha_3^{(T-2)}$) ve FKHO sistemindeki ikinci gecikme bloğunun çıkışındaki değerinden ($\alpha_3^{(T-t_2)}$) farklı olduğu değerlerin sayısının toplam zaman adımı sayısına (18.10^{10}) oranının yüzde grafiği verilmiştir. Ölçüm beş saat boyunca 30° sıcaklıkta yapılmıştır. Kullanılan entegre, Xilinx'in Spartan-6 ailesinin XC6SLX45 entegresidir. Gecikme blokları her bir hücre modülünde, fonksiyonunun çıkışı, bir girişinin değerine eşit olan LUT'ları ard arda bağlayarak oluşturulmuştur. LUT'ların sayısı, gecikme bloklarının gecikme değerlerinin, sistemin saat frekansına yaklaşık olarak eşit olacağı şekilde seçilmiştir.

Şekil 5.6'da görüldüğü gibi yaklaşık 50 dakika sonra farklılık oranı %1.5 gibi sabit bir değerde kalmaktadır.

Şekil 5.7'de 256 hücreli 1-boyutlu, 1.gecikme bloğunun gecikme değeri $T_{GECIKME} \geq T_{SAAT}$ olacak şekilde seçildiğinde elde edilen 150 kuralına sahip, başlangıç durumunda yalnızca bir hücrenin 1(siyah) olduğu, Spartan-6 ailesinden XC6SLX45 entegresinde gerçekleştirilen FKHO sisteminin 256 zaman adımı boyunca zamanda değişimi gösterilmiştir. Beklenildiği gibi örüntü, aynı başlangıç koşullarında, hafızalı hücresel otomat sayısal sisteminden elde edilen örüntü (Şekil 4.16) ile aynıdır. Burada ve bundan sonra oluşturulan bütün sistemlerde $T_{SAAT} = 10ns$ 'dir.



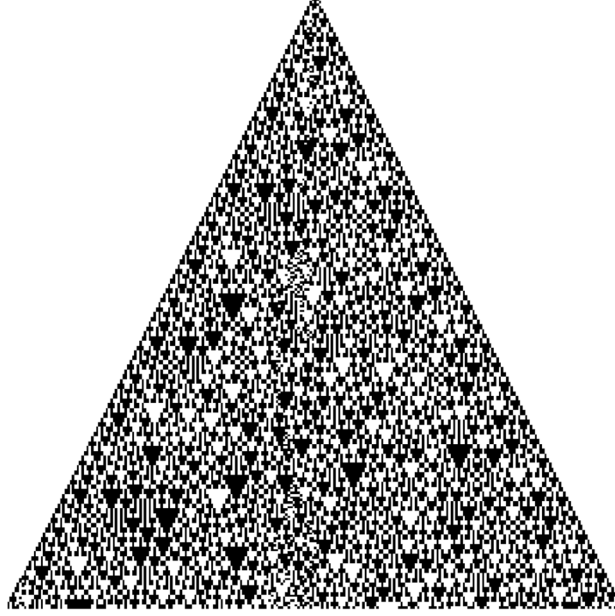
Şekil 5.6: 9 hücreli 1-boyutlu 150 kuralına sahip hafızalı hücresel otomatın dördüncü hücresinin, FKHO sistemindeki değerlerinin, hafızalı hücresel otomat sayısal sisteminden farklılığının yüzde grafiği.



Şekil 5.7: $T_{GECIKME} \geq T_{SAAT}$ için 150 kuralına sahip FKHO sisteminin zamanda değişimi.

Şekil 5.8’de 256 hücreli 1-boyutlu, 1.gecikme bloğunun gecikme değeri $T_{GECIKME} \leq T_{SAAT}$ olacak şekilde seçildiğinde elde edilen 150 kuralına sahip, başlangıç

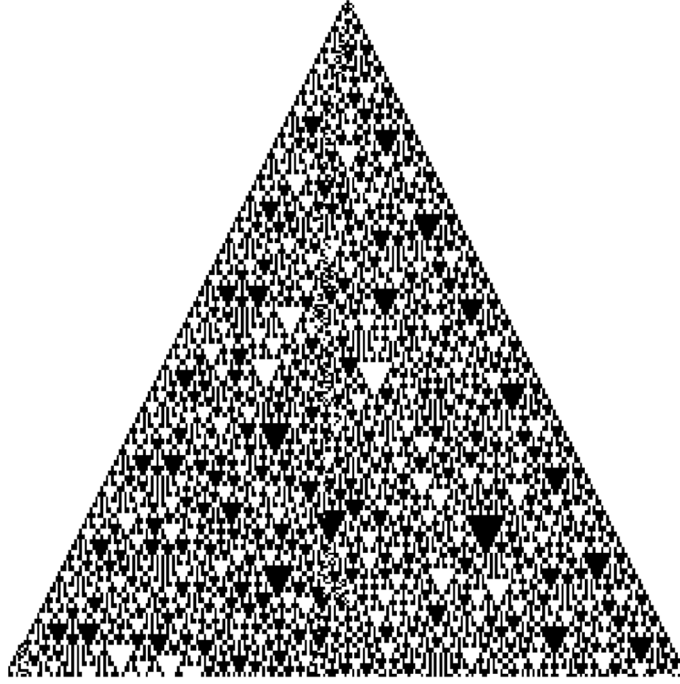
durumunda yalnızca bir hücrenin 1(siyah) olduğu, Spartan-6 ailesinden XC6SLX45 entegresinde gerçekleşen FKHO sisteminin 256 zaman adımı boyunca zamanda değişimi gösterilmiştir. Burada $T_{GECIKME}$ değeri T_{SAAT} değerine çok yakın olacak şekilde ayarlanmıştır. Böylece sistemde 1.gecikme bloğunun çıkışında, hücrenin bazen bir zaman adımı önceki bazen de o anki değeri görülmektedir. Böylece rastgele bir örüntü elde edilmiştir.



Şekil 5.8: $T_{GECIKME} \leq T_{SAAT}$ için 150 kuralına sahip FKHO sisteminin zamanda değişimi.

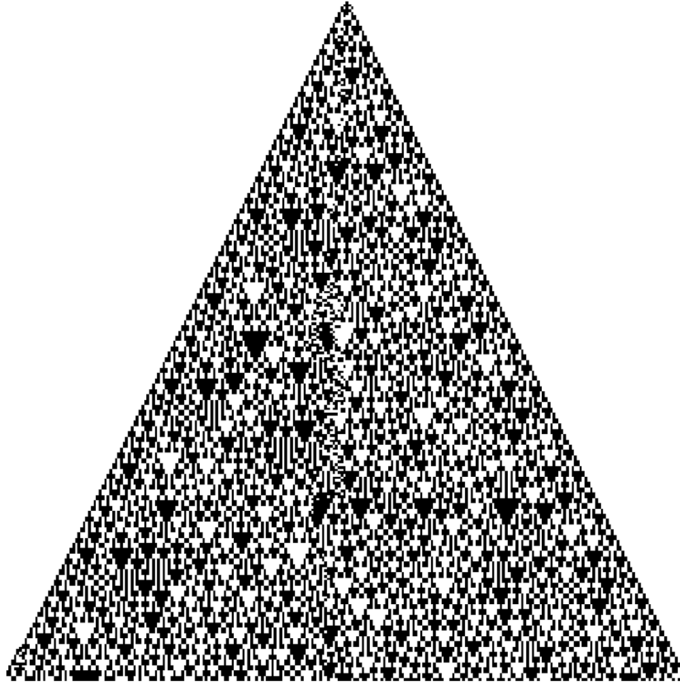
Şekil 5.9’da Şekil 5.8’de örüntüsü gösterilen aynı sistemin kısa zaman sonra (1 dakika sonra) aynı entegre üzerinde tekrar başlatıldığında elde edilen örüntüsü verilmiştir. Görüldüğü gibi örüntü Şekil 5.8’dekinden (sistem aynı olmasına rağmen) farklıdır.

Şekil 5.10’da, Şekil 5.8 ve Şekil 5.9’da örüntüleri verilen FKHO sisteminin bu sefer aynı ailenin aynı türü olan fakat farklı bir entegrede gerçekleştirildiğinde elde edilen örüntüsü verilmiştir. Bu örüntü de Şekil 5.8 ve Şekil 5.9’da verilen örüntülerden farklıdır. Bu örüntülerin birbirinden farklarının anlaşılabilmesi için de Şekil 5.11’de bu örüntülerin birbirileri ile xor lojik işlemi gerçekleştirildiğinde elde edilen örüntü verilmiştir. Görüldüğü gibi yaklaşık 10 zaman adımı sonra örüntüler birbirinden farklılaşmaktadır.

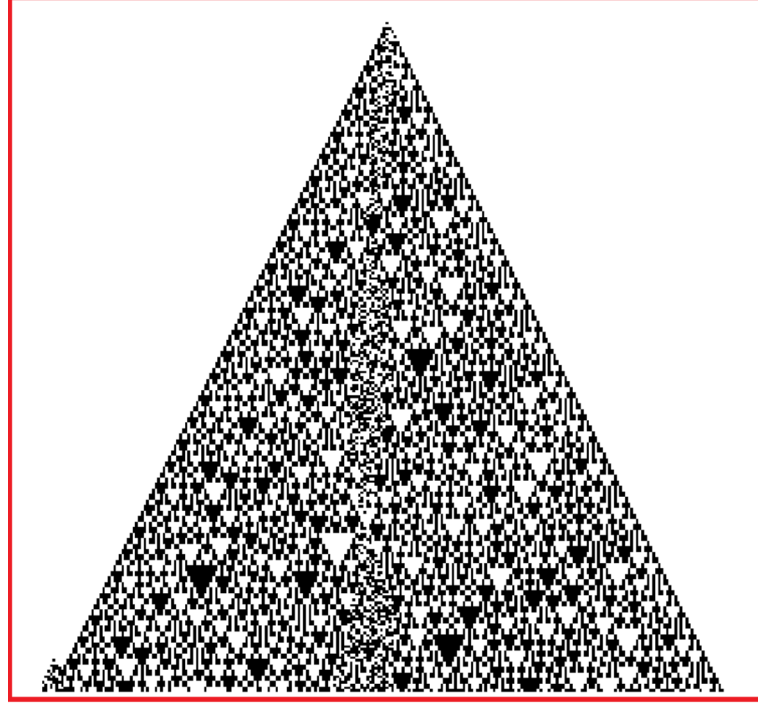


Şekil 5.9: Şekil 5.8’de örüntüsü gösterilen aynı sistemin kısa zaman sonra (1 dakika sonra) tekrar çalıştırıldığında elde edilen örüntüsü.

Bu deneylerden elde edilen sonuçlara göre FKHO sistemi farklı entegrelerde farklı çıkışlar vermektedir. Aynı entegre için dahi farklı zamanlarda farklı çıkışlar vermektedir.



Şekil 5.10: Farklı tümdevreden elde edilen örüntü.



Şekil 5.11: Şekil 5.8’deki ve Şekil 5.10’daki örünütlülerin farkları.

5.4 FKHO Sistemi Kullanılarak Tümdevre Kimliklendirmesi

Bir önceki bölümdeki ile aynı fakat 64 hücreli bir FKHO sistemi oluşturulmuştur. Bu sistem, beş ayrı FPGA içine gömülmüş ve aynı başlangıç koşullarında 63 zaman adımı boyunca, 24° sıcaklıkta, 10000 kere tekrar çalıştırılmıştır. Her çalıştırmanın sonunda 63. zaman adımındaki hücrelerin değerlerinden oluşturulan 64 bitlik sayıların ortalamaları alınarak FPGA’lar birbirinden ayrılmaya çalışılmıştır. Çizelge 5.1’de bu ölçümlerin sonuçları verilmiştir.

Çizelge 5.1: Tümdevre kimliklendirme için yapılan ölçümlerin sonuçları.

	FPGA A	FPGA B	FPGA C	FPGA D	FPGA E
ORTALAMA	$1,01 \cdot 10^{19}$	$0,87 \cdot 10^{19}$	$0,957 \cdot 10^{19}$	$0,765 \cdot 10^{19}$	$0,724 \cdot 10^{19}$

6. SONUÇ VE ÖNERİLER

Tez çalışmasında, hücresel otomatın literatürde verilen matematiksel tanımları araştırılarak oluşturulan yeni sisteme en uygun olanı seçilmiştir. Bu seçilen tanım FKHO sisteminin tanımlamasında temel teşkil edeceğinden detaylı olarak anlatılmıştır.

Bu matematiksel forma uygun olarak hafızalı hücresel otomat için yeni bir matematiksel tanım önerilmiştir. Ramon Alonso-Sanz'ın önerdiği hafızalı temel hücresel otomat, literatüre katılan bu matematiksel tanıma uygun olarak yeniden tanımlanmıştır. Daha sonra bu sistem sayısal olarak tasarlanmış ve bu tasarım FPGA tümdevresi kullanılarak gerçekleştirilmiştir. Gerçekleme sonucunda elde edilen verilerin literatürdeki veriler ile karşılaştırılması yapılmış ve sonuçların aynı olduğu görülmüştür.

Hafızalı hücresel otomat ile ilgili çalışmalardan sonra yeni bir sistem önerilmiştir. Hafıza komşuluklarının rastgele seçilmesi ile elde edilen, FKHO (Fiziksel Olarak Klonlanamayan Hücresel Otomat) adı verilen bu sistemin matematiksel tanımı verilmiştir. Daha sonra bu matematiksel tanıma uygun olarak örnek bir FKHO sistemi sayısal olarak tasarlanmıştır. Aslında tasarım, hafızalı hücresel otomat için oluşturulan tasarımdaki flip-floplar yerine FPGA içersindeki gecikme hatlarının kullanılmasıyla elde edilmiştir. Bu tasarıma göre hücrelerin bir sonraki durumları belirlenirken hafızalı hücresel otomatta bir önceki ve iki önceki durumlara bakılırken, FKHO sisteminde bazen bir önceki, bazen şimdiki bazen de iki önceki durumlara rastgele olarak bakılır. Oluşturulan bu tasarım FPGA tümdevresi kullanılarak gerçekleştirilmiştir. Hafızalı hücresel otomatta verilen örneğin (Kural 150 ve minimal hafızalı) aynısı bu sistemde de denenmiş ve sonuçlar karşılaştırılmıştır. Sonuçlar, hafızalı hücresel otomatın sonuçlarından oldukça farklı çıkmıştır. Aynı sistemin aynı tümdevrede birden fazla çalıştırılması durumunda yine sonuçların farklı olduğu görülmüştür. Daha sonra aynı sistem aynı firmanın aynı ürünü olan iki ayrı tümdevrede çalıştırılmış ve iki ayrı tümdevreden elde edilen sonuçların yine birbirinden farklı olduğu görülmüştür.

Çalışmanın sonunda önerilen FKHO sisteminin ayrı tümdevrelerde aynı girişlere karşılık farklı çıkışlar üretmesi kullanılarak aynı firmanın aynı ürünü olan beş ayrı tümdevrenin kimliklendirilmesi yapılmıştır. Aynı aileden olan beş ayrı FPGA tümdevresinde gerçekleştirilen aynı başlangıç koşullarında, aynı özelliklere sahip FKHO sistemi sayesinde her tümdevreden farklı değerler elde edilmiş ve böylece tümdevreler kimliklendirilmiştir. Elde edilen bu sonuçlara göre üzerinde çalışılan FKHO sisteminin, tümdevre kimliklendirilmesi ve IP korumalarında kullanılan PUF yapılarına alternatif olarak kullanılmaları mümkün hale gelecektir.

Tümdevre kimliklendirilmesi deneyinde daha önceden belirtildiği gibi beş FPGA tümdevresi kullanılmıştır. Elde edilen sonuçların bütün tümdevrelerde geçerli olup olamayacağını belirleyebilmek için daha fazla tümdevre üzerinde testler yapılmalıdır.

FKHO sistemi kullanılarak tümdevre kimliklendirmesi yapılırken basit bir yöntem kullanılmıştır. Bu yöntemle göre aynı parametreler ve aynı başlangıç değerlere verilerek sistem, belli bir sayıda tekrar çalıştırılıp belli bir zaman adımı sonucunda çıkan örüntüler sayılara çevrilip ortalaması alınmıştır. Bu ortalamaların ayrı tümdevrelerde farklı olduğu görülmüş ve tümdevre kimliklendirilmesi için ayırt edici özellik olarak kullanılmasına karar verilmiştir. Bu basit yöntem yerine tümdevre kimliklendirilmesinde kullanılabilecek başka ayırt edici özellikler de bulunabilir.

FKHO sistemi, daha önceden belirtildiği gibi aynı tümdevrelerde aynı başlangıç değerleri verilerek farklı zamanlarda tekrar çalıştırıldığında farklı sonuçlar üretmektedir. Ayrıca ortam sıcaklığı değiştiğinde gecikme hatlarının hızlarının bu sıcaklık değişiminden etkilenmesinden sonuçlarda büyük değişiklikler görülmektedir. Bunun gibi sistemin güvenilirliğini olumsuz etkileyen faktörler için testler yapılmalı ve sistemin güvenilirliği artırılması için tasarımda iyileştirmeler yapılmalıdır.

Tez çalışmasında FKHO sisteminin uygulaması olarak tümdevre kimliklendirilmesi yapılması ile birlikte sistemin farklı zamanlarda farklı sonuçlar vermesi kullanılarak gerçek rastgele sayı üreteçleri geliştirmek de mümkün olabilir.

FKHO sisteminin dinamiksel davranışı incelenmeli ve klasik hücresel otomatların ve hafızalı hücresel otomatların dinamiksel davranışı ile karşılaştırılmalıdır. Bu inceleme sonucunda FKHO sisteminin kullanılabileceği başka alanlar belirlenebilir.

Sonu olarak tez alıřması boyunca elde edilen tecrbelere gre literatre katılan FKHO sisteminin ileride rastgele sayı reteleri ve tmdevre kimliklendirmesinden bařka insanlıėa katkıda bulunacak birok bilimsel alıřmalarda da kullanılması beklenmektedir.

KAYNAKLAR

- [1] **Wolfram, S.** (1983). Statistical mechanics of cellular automata, *Rev. Mod. Phys.*, **55**, 601–644.
- [2] **Niesche, H.** Introduction to Cellular Automata Seminar “Organic Computing”.
- [3] **von Neumann, J.** (1963). The general and logical theory of automata, *Collected Works*, **5**, 288.
- [4] **von Neumann, J.** (1966). *Theory of Self-Reproducing Automata*, University of Illinois Press, Champaign, IL, USA.
- [5] **Ulam, S.M.** (1972). Some ideas and prospects in biomathematics, *Annual Review of Biophysics and Bioengineering*, 277–292.
- [6] **Codd, E.** (1968). *Cellular automata*, ACM monograph series, <http://books.google.com.tr/books?id=GdpQAAAAMAAJ>, .
- [7] **Zuse, K.** (1967). *Elektronische Datenverarbeitung (Electronic Data processing)*, cilt 8.
- [8] **Conway, J.H.** (1970). *unpublished*.
- [9] **Gardner, M.** (1970). Mathematical Games: The fantastic combinations of John Conway’s new solitaire game “life”, *Scientific American*, 120–123.
- [10] **Burstedde, C., Klauck, K., Schadschneider, A. ve Zittartz, J.** (2001). Simulation of pedestrian dynamics using a two-dimensional cellular automaton, *Physica A: Statistical Mechanics and its Applications*, <http://www.sciencedirect.com/science/article/pii/S0378437101001418>, .
- [11] **Wolfram, S.** (1986). Random sequence generation by cellular automata, *Advances in Applied Mathematics*, **7**(2), 123–169.
- [12] **Viher, B., Dobnikar, a. ve Zazula, D.** (1998). Cellular automata and follicle recognition problem and possibilities of using cellular automata for image recognition purposes, *International journal of medical informatics*, **49**(2), 231–41.
- [13] **Rosin, P.L.** (2006). Training cellular automata for image processing, *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, **15**(7), 2076–87.
- [14] **Jin, J.** (2012). An image encryption based on elementary cellular automata, *Optics and Lasers in Engineering*, **50**(12), 1836–1843.

- [15] **Toffoli, T. ve Margolus, N.** (1987). *Cellular automata machines: a new environment for modeling*, MIT Press, Cambridge, MA, USA.
- [16] **Goles, E.,** (1989). Cellular Automata, Dynamics and Complexity, **P. Manneville, N. Boccara, G. Vichniac ve R. Bidaux,** (düzenleyenler), Cellular Automata and Modeling of Complex Physical Systems, cilt 46 of *Springer Proceedings in Physics*, Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-75259-9_2,.
- [17] **Schiff, J.L.** (2007). *Cellular Automata: A Discrete View of the World (Wiley Series in Discrete Mathematics & Optimization)*.
- [18] **Url-1,** <http://mathworld.wolfram.com/Rule150.html>, alındığı tarih: 16.02.2013.
- [19] **Wolfram, S.** (2002). *A New Kind of Science*, Wolfram Media.
- [20] **Fredkin, E.** (1990). Digital mechanics: an informational process based on reversible universal cellular automata, *Phys. D*, [http://dx.doi.org/10.1016/0167-2789\(90\)90186-S](http://dx.doi.org/10.1016/0167-2789(90)90186-S),.
- [21] **Wolfram, S.** (1983). Cellular Automata, *Los Alamos Sci*, 2–21.
- [22] **Shackleford, B., Tanaka, M., Carter, R.J. ve Snider, G.** (2002). High-Performance Cellular Automata Random Number Generators for Embedded Probabilistic Computing Systems, *Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH'02)*, EH '02, IEEE Computer Society, Washington, DC, USA, <http://dl.acm.org/citation.cfm?id=787264.787871>,.
- [23] **Shackleford, B., Tanaka, M., Carter, R.J. ve Snider, G.** (2002). FPGA implementation of neighborhood-of-four cellular automata random number generators, *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays - FPGA '02*, ACM Press, New York, New York, USA, <http://portal.acm.org/citation.cfm?doid=503048.503064>,.
- [24] **Kobori, T., Maruyama, T. ve Hoshino, T.** (2001). A cellular automata system with FPGA, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1420908,.
- [25] **Corsonello, P., Spezzano, G., Staino, G. ve Talia, D.** (2002). Efficient implementation of cellular algorithms on reconfigurable hardware, *Proceedings of the 10th Euromicro conference on Parallel, distributed and network-based processing, EUROMICRO-PDP'02*, IEEE Computer Society, Washington, DC, USA, <http://dl.acm.org/citation.cfm?id=1895489.1895518>,.
- [26] **Chuanwu, Z.** (2008). Performance Analysis of the CPLD/FPGA Implementation of Cellular Automata, <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4627177>,.

- [27] **Alonso-Sanz, R., Martín, M.** (2003). Elementary cellular automata with memory, *Complex Systems*, <http://www.complex-systems.com/pdf/14-2-1.pdf>, .
- [28] **Alonso-sanz, R. ve Martín, M.** (2005). One-dimensional Cellular Automata with Memory in Cells of the Most Frequent Recent Value, *IS*, 203–236.
- [29] **Alonso-Sanz, R. ve Martín, M.** (2006). Elementary Cellular Automata with Elementary Memory Rules in Cells: the case of linear rules, *Journal of Cellular Automata*, <http://www.cs.sun.ac.za/~lvzijl/courses/rw778/autappl/exam-article.pdf>, .
- [30] **Gassend, B., Clarke, D., van Dijk, M. ve Devadas, S.** (2002). Silicon physical random functions, <http://portal.acm.org/citation.cfm?doid=586110.586132>, .
- [31] **Lim, D., Lee, J., Gassend, B., Suh, G., van Dijk, M. ve Devadas, S.** (2005). Extracting secret keys from integrated circuits, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, http://ieeexplore.ieee.org/xpls/abs/_all.jsp?arnumber=1561249<http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1561249>, .
- [32] **Kaya, G., Saldamli, G., Yalcin, M. ve Goncu, E.** Improving the reliability of ring oscillator based physical unclonable functions by a labeling technique, *The 2012 FTRA International Symposium on Advances in Cryptography, Security and Applications for future computing*.
- [33] **Maes, R., Tuyls, P. ve Verbauwhede, I.** (2008). Intrinsic PUFs from flip-flops on reconfigurable devices, *3rd Benelux workshop on information and System Security (WISSec 2008)*, <https://www.cosic.esat.kuleuven.be/publications/article-1173.pdf>, .
- [34] **Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.j. ve Tuyls, P.** (2008). Extended abstract: The butterfly PUF protecting IP on every FPGA, *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 71369, IEEE, <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4559053>, .
- [35] **Suh, G.E. ve Devadas, S.** (2007). Physical Unclonable Functions for Device Authentication and Secret Key Generation, http://www.crossref.org/deleted/_DOI.html, .
- [36] **Url-2**, <http://www.xilinx.com/content/xilinx/en/products/slicon-devices/soc/zynq-7000.html>, alındığı tarih: 20.03.2013.
- [37] **Ercsey-Ravasz, M., Roska, T. ve Néda, Z.** (2006). Perspectives for Monte Carlo Simulations on the CNN Universal Machine, *International Journal of Modern Physics C*, <http://arxiv.org/abs/physics/0603121><http://www.worldscientific.com/doi/abs/10.1142/S0129183106009230>, , 0603121.

EKLER

EK A : Hayat Oyunu

EK B : Sahada Programlanabilir Kapı Dizileri

EK A

Hayat oyunu, İngiliz matematikçi İngiliz John Horton Conway'in [8] geliştirdiği iki boyutlu bir hücresel otomattır. İlk olarak 1970 yılında Scientific American dergisinde Martin Gardner'ın sütununda yayınlanmıştır [9].

Oyunun kuralları oldukça basittir. Her hücre iki durumdan (canlı, ölü) birindedir. Yani hücresel otomat iki durumludur. Komşuluk hücrenin kendisine en yakın olan iki hücre ile tanımlanmıştır. Hücrenin bir sonraki durumu komşuluğundaki hücrelerin durumuna göre değişir:

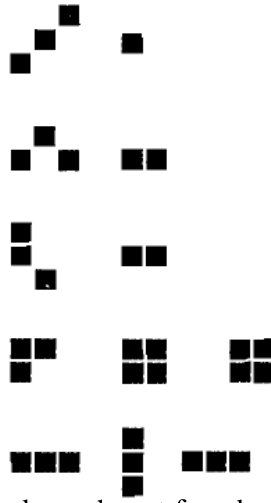
1- Hücre eğer ölü ise ve komşuluğundaki hücrelerin üçü canlı ise bir sonraki adımda hücre canlıdır. 2- Hücre eğer canlı ise ve komşuluğundaki hücrelerin ikisi veya üçü canlı ise bir sonraki adımda hücre canlı kalır. Ters durum söz konusu ise hücre bir sonraki adımda ölür.

Aslında ikinci kural, tek başına bir hücre canlı ise bir sonraki zaman adımında o hücrenin yalnızlıktan öleceğini söylemektedir. Üçten fazla canlı komşu olduğunda ise hücre fazla kalabalıktan dolayı ölecektir.

Hayat oyunu hücresel otomatında periyodik sınır koşulları geçerlidir. Dolayısıyla iki boyutlu olan Hayat Oyunu torus şeklinde düşünülebilir.

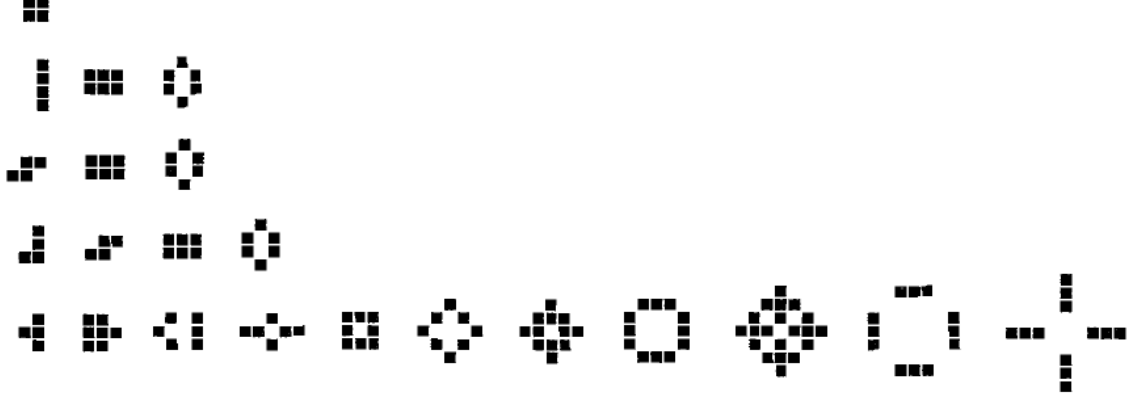
Hayat Oyunu, Wolfram'ın sınıflandırmasına göre Sınıf 4'e aittir. Buna göre ne periyodik ne de kaotik bir davranış sergilediği söylenemez.

Daha önceden belirttiğimiz gibi Hayat Oyunu'nda üçten daha az olan canlı hücre bir sonraki zaman adımında ölecektir. Olası üçü canlı hücreden oluşan formlar için sonraki zaman adımlarında ya hızlı bir şekilde yayılmanın olacağından ya da bazı şekillerin periyodik bir şekilde tekrarlanacağından söz etmek mümkündür. Şekil A.1'de üç canlı hücreden oluşan formların dinamik davranışlarına örnekler verilmiştir. Bu örnekte zaman soldan sağa doğru ilerlemektedir.



Şekil A.1: Üç canlı hücreden oluşan hayat formlarının dinamik davranışı [17].

Dört canlı hücreden oluşan yaşam formlarında ise sonraki zaman adımlarında kararlı formlar oluşabileceği gibi uzun zaman adımları boyunca farklı yaşam formlarının da oluşması mümkündür. Şekil A.2 'de dört canlı hücreden oluşan yaşam formlarına örnekler verilmiştir.



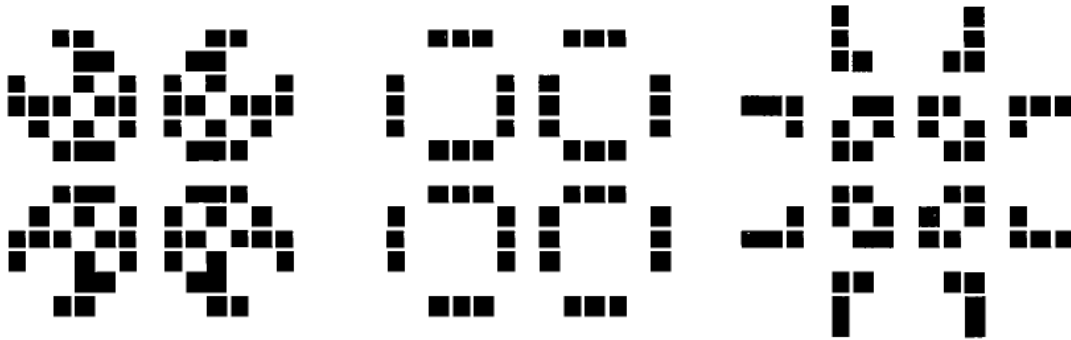
Şekil A.2: Dört canlı hücreden oluşan hayat formlarının dinamik davranışı [17].

Bazı formlar ise zaman adımları boyunca değişmeden kalır. Bu formlardan bazıları Şekil A.3 'te verilmiştir.



Şekil A.3: Zaman adımları boyunca değişmeyen hayat formları [17].

Bazı yaşam formları ise belli periyotlarda kendisini tekrarlar. Osilatör adı verilen bu yaşam formlarının dinamik davranışına Şekil A.4'te bir örnek verilmiştir. Bu örnekte dördüncü zaman adımından sonra hücresel otomat ilk haline geri döner.



Şekil A.4: Bir osilatörün dinamik davranışına örnek [17].

Yaşam formlarından en çok bilinenlerinden birisi de planörlerdir. Planörler beş canlı hücreden oluşur. Dördüncü zaman adımında bu yaşam formundaki hücrelerden birisi köşegen üzerinde hareket eder. Bu davranış her dört zaman adımında bir tekrarlanır.



Şekil A.5: Bir planörün dört zaman adımı boyuncaki dinamik davranışı [17].

Böylece planör zaman adımları boyunca, dönerek hareket ediyormuş gibi görünür. Şekil A.5’ te bir planör dört zaman adımı boyuncaki dinamik davranışı gösterilmiştir.

Hayat Oyunu daha detaylı ve geniş bilgiye [17]’dan ulaşılabilir.

EK B

FPGA (Sahada Programlanabilir Kapı Dizileri), üretildikten sonra yapılandırılabilen bir tümdevredir. Yapılandırma genellikle donanım tanımlama dilleri ile yapılır. FPGA'lar içerisinde çok sayıda sayısal bloklar ve RAM (Rastgele Erişimli Hafıza) blokları içerir. İçerisindeki bu kaynaklar sayesinde karmaşık sayısal tasarımlar oluşturmak mümkündür. Genellikle hızın önemli olduğu uygulamalarda tercih edilir. Çünkü mikro denetleyiciler gibi kodları sırası ile çalıştırmak yerine, FPGA'da sayısal tasarım donanım olarak oluşturulduğundan, genellikle çalışma frekansları düşük olmasına rağmen aynı amaç için FPGA ve mikro denetleyicilerle geliştirilen tasarımlar karşılaştırıldığında FPGA ile geliştirilen tasarımların çok daha hızlı olduğu görülür.

FPGA'lar içerisinde programlanabilir lojik blokları bulunur. Bu lojik bloklar karmaşık kombinezonsal fonksiyonları gerçekleyebilir. Ayrıca lojik bloklar flip-flop gibi hafıza elemanları da içerir. Hafıza elemanlarını da kullanarak ardışıl devreler gerçeklemek de mümkündür.

FPGA'lardan önce programlanabilir lojik cihazlar bulunuyordu ve FPGA'nın icat edilmesine kadar geçen süreç şu şekilde özetlenebilir:

1-) 1970'te ilk kez Texas Instruments PLA (Programlanabilir lojik dizisi)'ni geliştirmişlerdir. Bu cihaz kombinezonsal lojik dizileri gerçeklemek için kullanılıyordu. İçerisinde AND kapı dizileri ve OR kapı dizileri bunların arabağlantıları bulunurdu. Bu sayede istenen tasarım çarpımların toplamı haline getirilip gerçekleştirilebiliyordu.

2-) 1978 'de MMI (Monolithic Memories Inc.) yeni bir programlanabilir lojik cihaz geliştirdi. PAL (Programlanabilir Dizi Lojik) adını verdikleri bu ürün içerisinde bir PROM (Programlanabilir sadece okunabilir hafıza ve çıkış lojik birimleri) içermekteydi. Bu cihazlar da bir kez programlanabiliyordu.

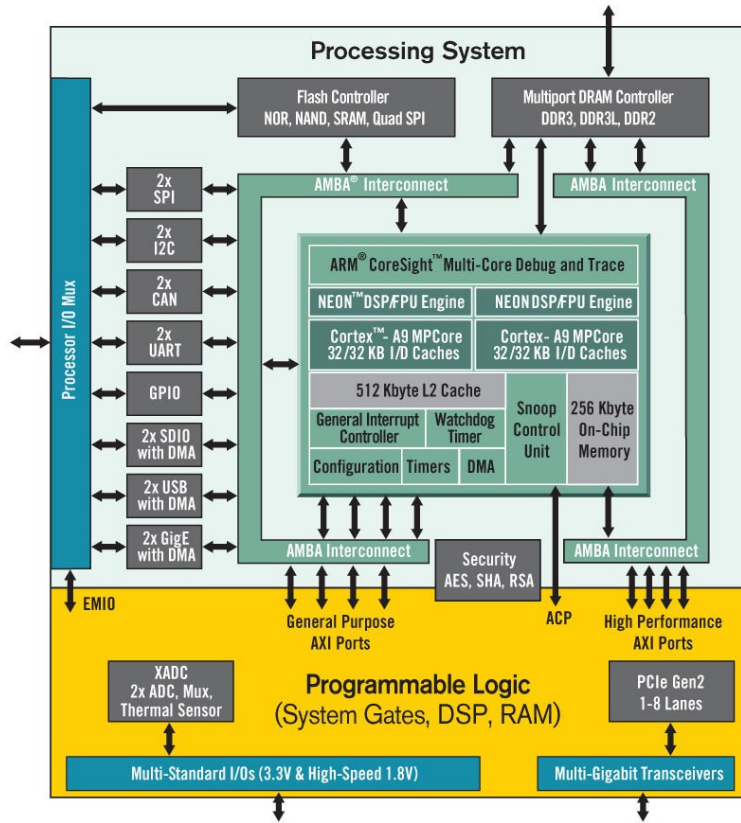
3-) 1985'te Lattice Semiconductor GAL (Genelleştirilmiş Dizi Lojiği) adını verdikleri bir ürün çıkardılar. GAL'ların getirdiği en önemli yenilik yeniden programlanabilir olmalarıdır.

Bütün bu gelişmelerden sonra Xilinx şirketinin ortakları Ross Freeman ve Bernard Vonderschmitt ilk ticari FPGA'yı (XC2064) 1985'te icat etmişlerdir. XC2064 programlanabilir lojik kapılar ve programlanabilir lojik arabağlantılar içermekteydi. Freeman bu icadından yirmi sene sonra Amerika'nın Ulusal Mucitler Onur listesine (NIHF) girebilmiştir.

Günümüze kadar FPGA'lar kaynak açısından (lojik, RAM ve I/O blokları) oldukça genişleyerek gelmiştir. Bugün aynı tümdevrede hem mikroişlemci hem de FPGA bulunduran tümdevreler mevcuttur. Şekil B.1'de bu tümdevrelerden biri olan Xilinx ZYNQ-7000'in blok diyagramı verilmiştir.

Genellikle bir FPGA mimarisi, programlanabilen lojik blok, I/O (Giriş/Çıkış) tamponlarından (PAD) ve yönlendirme kanallarından oluşur.

Lojik blok, birkaç lojik hücreden (dilim) meydana gelir. Tipik bir dilim dört girişli bir LUT (Look Up Table), bir FA (Tam toplayıcı) ve bir D tipi flip-floptan oluşur.



Şekil B.1: ZYNQ-7000 tümdevresinin blok diyramı [36].

Yolları yönlendirme ve bağlama işlemi yine FPGA içinde bulunan programlanabilir şalter kutuları (switch box) ile yapılır.

FPGA ile bir sayısal devre tasarlamak için donanım tanımlama dilleri veya şematik programları kullanılır. Büyük tasarımlarda şematik programları kullanmak pek mümkün olmayacağı için çoğunlukla donanım tanımlama dilleri kullanılır. FPGA ile bir sayısal tasarım yapmak için aşağıdaki adımlar sırasıyla FPGA üreticilerinin sağladığı programlar (Xilinx ISE gibi) kullanılarak gerçekleştirilir:

- 1-) Sentez: Donanım tanımlama dili ile yazılmış kod ile sentezleyici aracının türüne göre yeni bir kod sentezlenir. Bu kodda sentezleyicinin kütüphanelerinde bulunan lojik elemanlar vardır.
- 2-) Çevirme: Sentez sonrası oluşturulan kod FPGA üreticisinin kütüphanesine uygun olacak şekilde çevirilir.
- 3-) Eşleme : Çevirme sonrasında elde edilen kod kullanılacak FPGA tümdevresinin içinde gerçekten bulunan lojik bloklar, RAM blokları ve I/O blokları ile eşleştirilecek şekilde yeni bir kod üretilir.
- 4-) Yerleştir ve Bağla : Bu işlem sonrasında tasarımın FPGA tümdevresinin tam olarak neresine yerleştirileceğini hangi yollar kullanılacağını ve hangi I/O PAD'lerinin kullanılacağını belirten bir kod oluşturulur.
- 5-) Bit Dizisi Üretme: Yerleştir ve bağla işlemi sonrasında elde edilen kod .bit uzantılı bir dosyaya dönüştürülür. Bu dosya daha sonra FPGA'yı programlamak için kullanılır.

Bütün bu bilgilerden sonra FPGA'nın kullanım alanlarından da bahsetmek gerekir: Otomobillerde ses işleme, video işleme, resim işleme, ASIC (Uygulamaya yönelik Tümdevre) protipini oluşturmada, gerçek zamanlı video işlemede, tüketici elektroniğinde sayısal kameralarda ve daha birçok alanda kullanılmaktadır.

PHOTO

ÖZGEÇMİŞ

Ad Soyad: Emre GÖNCÜ

Doğum Yeri ve Tarihi: Keçiören, 1987

Adres: İstanbul Teknik Üniversitesi Ayazağa Kampüsü Elektrik-Elektronik Fakültesi
Oda:1109 Şişli/İSTANBUL

E-Posta: egoncu@itu.edu.tr

Lisans: Uludağ Üniversitesi Elektronik Mühendisliği

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- Kaya G., Saldamli G., Yalcin M., **Göncü E.**, 2012: Improving the reliability of ring oscillator based physical unclonable functions by a labeling technique. *The 2012 FTRA International Symposium on Advances in Cryptography, Security and Applications for future computing.*, November, 2012 Jeju, Korea.