İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FPGA ÜZERİNDE HAFIZALI HÜCRESEL OTOMAT YAPISI İLE RASTGELE SAYI ÜRETECİ TASARIMI

YÜKSEK LİSANS TEZİ

Abdulkadir KOÇDOĞAN

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

OCAK 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FPGA ÜZERİNDE HAFIZALI HÜCRESEL OTOMAT YAPISI İLE RASTGELE SAYI ÜRETECİ TASARIMI

YÜKSEK LİSANS TEZİ

Abdulkadir KOÇDOĞAN
(504121351)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Müştak Erhan YALÇIN

OCAK 2015

İTÜ, Fen Bilimleri Enstitüsü'nün 504121351 numaralı Yüksek Lisans Öğrencisi **Abdulkadir KOÇDOĞAN**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"FPGA ÜZERİNDE HAFIZALI HÜCRESEL OTOMAT YAPISI İLE RASTGELE SAYI ÜRETECİ TASARIMI"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Müştak Erhan YALÇIN** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Yrd. Doç. Dr. Metin YAZGI** İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. Nerhun YILDIZ Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : **15 Aralık 2014**
Savunma Tarihi : **28 Ocak 2015**

Aileme,

ÖNSÖZ

Öncelikle bana bu tez çalışmasını yapma fırsatı veren ve çalışmalarım boyunca yol gösteren değerli hocam Prof. Dr. Müştak Erhan Yalçın'a, teşekkür ederim.
Tez çalışmamın her safhasında büyük yardımlarını gördüğüm, kıymetli Araş. Gör. Yük. Müh. Emre Göncü'ye teşekkür ederim.
Ayrıca, başta Yük. Müh. Ramazan Yeniçeri olmak üzere, İTÜ Gömülü Sistem Tasarımı Laboratuvarı ekibine de teşekkürlerimi sunarım.
Yüksek lisans hayatım boyunca beni maddi anlamda rahatlatan burs desteğini sağlayan TÜBİTAK Bilim İnsanı Destekleme Daire Başkanlığı'na teşekkürlerimi arz ederim.
Çalışmalarım boyunca maddi ve manevi desteklerini esirgemeyerek her zaman yanımda olduklarını hissettiğim anneme, babama ve kardeşlerime teşekkürü bir borç bilirim.

Ocak 2015

Abdulkadir Koçdoğan
(Elektronik Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

FPGA	**:** Field Programmable Gate Array
LUT	**:** Look-up Table
PC	**:** Personal Computer
DDR2 SDRAM	**:** Double Data Rate 2 Synchronous Dynamic Random Access Memory
NIST	**:** National Institute of Standards and Technology
FIPS	**:** Federal Information Processing Standard

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

FPGA ÜZERİNDE HAFIZALI HÜCRESEL OTOMAT YAPISI İLE RASTGELE SAYI ÜRETECİ TASARIMI

ÖZET

Hücresel otomatlar, farklı boyutlarda hücreler dizisinden oluşabilen yapılardır. Yapısındaki hücreler değerlerini sonlu bir durum kümesinden alır. Uzay ve zamanın ayrık olduğu sistemlerdir. Hücresel otomatlarda, hücreler komşularıyla belirli bir kurala göre etkileşerek durum değiştirir. Bu şekilde zamanda ilerleyerek bir model ortaya çıkarırlar.

Hücresel otomatlar, fiziksel, kimyasal ve biyolojik sistemlerin modellenmesinde kullanılmıştır. Cisimlerin çatlaması, bitki ve hayvanların gelişimi gibi bazı doğal olayların modellenmesini sağlamıştır. Ayrıca, hücresel otomatlar paralelliğinin de verdiği hızla görüntü işleme, görüntü tanıma gibi konularda da kullanılmıştır. Hücresel otomatlar, bu çalışmada da bahsedileceği gibi rastgele sayı üreteci tasarımında kullanılmıştır.

Bu çalışmada bir boyutlu hücresel otomat yapısıyla ilgilenilmiştir. Önce temel hücresel otomat yapıları incelenmiştir. Temel hücresel otomatta, hücre komşularıyla belirli bir kurala göre etkileşime girerek bir sonraki durum değerini almaktadır. Sonra hafızalı hücresel otomat yapısı incelenmiştir. Bu yapıda, hücrenin bir sonraki değeri sadece komşularının şimdiki durumlarına değil, kendisinin ve komşularının önceki değerlerine de bağlıdır. Daha sonra hafızaya rastgeleliğin katıldığı hücresel otomat yapısı incelenmiştir.

Buradan yola çıkılarak, hücrenin önceki değerlerine rastgele olarak bakılan yeni hücresel otomat yapısı tasarlanmıştır. Buna göre, hücrenin o anki değeri belirlenirken bir önceki ve şimdiki değerine rastgele olarak bakılır. Komşuları için de aynı işlem yapılır ve hücrelerin değerleri belirlenir. Böylece bu değerler belirli bir kurala göre etkileşerek hücrenin bir sonraki değeri belirlenir. Bu tasarımın FPGA üzerinde sayısal gerçeklemesi yapılmıştır. FPGA üzerinde çalıştırılan sistemin sonuçları bilgisayara alınarak hücresel otomatın yayılımı gözlenmiştir. Gerçekten rastgele sonuçlar alındığı görülmüştür. Sonra da bu sistem üzerinden rastgele sayı üreteci tasarımı yapılmıştır.

Rastgele sayı üreteçlerinin, rastgeleliklerinin ne kadar güvenilir olduğunu ölçen testler vardır. Rastgele sayı üreteçlerinin bu testlerdeki başarı durumuna göre güvenilirlikleri hakkında yorum yapılabilir. Bu çalışmada da, tasarlanan rastgele sayı üreteci bu testlerden geçirilmiş ve başarılı sonuçlar alınmıştır.

DESIGN OF RANDOM NUMBER GENERATOR USING CELLULAR AUTOMATA WITH MEMORY ON FPGA

SUMMARY

Cellular automata are structures consisting of array of cells which are different dimensions. Each cell gets its value from a finite state stack. Space and time are discreet in cellular automata. In cellular automata, state of the cell gives value of the cell at the moment. In fact, cellular automata is a finite state machine. A cell and its neighbours determine value of the cell at the next state. So, cellular automata progress with expanding in time.

Cellular automata concept was propounded firstly by Von Neumann and Ulam. Cellular automata were used in modelling physical, chemical and biologics systems. Wolfram described the cellular automata as a new science and studied on it. According to Wolfram, this science is a system that has basic rules and stairs effecting each other. Wolfram modelled some nature events, such as evolution of crystals, splitting of objects, evolution of plants and animals, with cellular automata.

When it was understood that cellular automata are not insufficient for modelling the problems in real world, different kind of cellular automata were propounded such as irregular cellular automata, asynchronous cellular automata. Cellular automata have been used in image processing, image reconition, cryptography areas because of their parallel design. Since, parallel design of cellular automata provide system with speed. Furthermore, cellular automata have been used in design of random number generator.

Cellular automata can be different dimensions such as one, two o three dimensions. In this work, one dimension cellular automata structure has been used. Next state of a cell obtains from the cell's and its neighbours' current state value. The value of the cell and its neighbours' values interact with each other according to a rule. So, the next state value of the cell is obtained. In this work, number of neighborhood is three. These are the cell in itself, left of the cell and right of the cell. Values of cells can be 0 or 1, so finite state stack has two elements in this work. Rules are expressed by decimal equivalent of binary value. In this work, rule 150 has been used.

In this work, firstly structure of elemantary cellular automata has been examined. In elemantary cellular automata, next state of a cell depends on current state of the cell and its neighbors.

Memory feature has been added to elemantary cellular automata later. So, cellular automata with memory have been obtained. In this structure, next state of a cell does not depend on only current state of the cell and its neighbors. Also, it depends on previous states of the cell and its neighbors. Firstly, value of a cell has been depended on all of previous states of the cell and its neighbors. Then R. Alonso-Sanz has proposed a new cellular automata with memory. He has been described a function named majority function. According to this, a cell's all previous states are examined and the value which is more often than the other, 0 or 1, is taken as the

cell's value. This value is used in applying rule. Then, a new majority function has been described named minimal majority function. According to this, a cell's current value, one previous value and two previous value are examined. The most often value determines the cell's value. This value is used in applying the rule.

Randomness has been wanted to be added to cellular automata with memory. Goncu has proposed a new cellular automata structure named cellular automata with random memory. In this strucuture, the minimal majority function has also been used. According to this new structure, a cell's current state, one previous state and two previous state are examined as random. So, the cell's value is determined as random. It has been seen from that this structure can be use as random number generator. A new celluar automata with random memory structure has been proposed to improve randomness. In new kind of cellular automata with random memory, the minimal majority function is not used. Furthermore, a cell's value depends on current state of the cell and one previous state of the sell as random. So, randomness has been increased.

These cellular automata structures which are cellular automata with memory, cellular automata with with random memory and a new cellular automata with random memory have been designed on FPGA. Design of cellular automata with memory contains two combinational circuits, a finite state machine and two flip-flops. One of the combinational circuits is verified the rule and the other is verified the minimal majority function. Finite state machine runs the cellular automata. Flip-flops keeps a cell's one previous and two previous values. Design of cellular automata with random memory contains same modules with cellular automata with memory except flip-flops. In this design, delay lines have been used instead of flip-flops to provide randomness. Delay lines have been acquired by array of LUTs which are FPGA's element. Design of a new cellular automata with random memory contains fewer modules. It has had a combinational circuit, finite state machine and a delay line. Similarly, combinational circuit verifies the rule. Finite state machine runs the cellular automata. Delay line provides with randomness between cell's current state and one previous state. These systems designed on FPGA ahave been run and acquired results have been sent to PC. Displays of cellular automata have been obtained on PC. Cellular automata with memory have regular shape. Cellular automata with random memory have irregular shape because of randomness. A new cellular automata with random memory have more irregular shape than the other. Since their randomness is better.

Random number genetors have been used in different areas such as cryptology. Long random number arrays are required in cryptology to provide data security. In this work, random number generator using cellular automata with random memory has been designed on FPGA. In this design, the last proposed cellular automata with random memory structure has been used. Two cells have been chosen from cellular automata with random memory which have 64 cells. EXOR operation has been applied between these two cells. Acquied results on all iterations have generated a random number array.

Some tests are applied to random number generators to measure reliability of randomness. Long random number arrays are required for these tests. It is required bigger memory. For this, DDR2 SDRAM on Atlys evolution board has been used.

FIPS 140-1 test family and NIST test family have been applied on designed random number generator. FIPS 140-1 test family consists of four different tests which are

frequency test, runs test, long run test and poker test. The designed random number generator passed succecfully on this tests. NIST test family consists of 16 tests. NIST test has also been applied on designed random number generator. NIST test has passed successfully substantially.

In conclusion, a random number generator using cellular automata with memory has been desgined on FPGA. Tests have been applied on the random number generator and successful results have been acquired.

1. GİRİŞ

Hücresel otomatlar, değerlerini sonlu bir durum kümesinin elemanlarından alan hücreler dizisinden oluşan, uzay ve zamanın ayrık olduğu sistemlerdir [1]. Hücrelerin zamanda birbirlerini etkilemesiyle yayılarak ilerleyen yapılardır.

Hücresel otomat kavramı ilk olarak Von Neumann ve Ulam tarafından ortaya atılmıştır. Daha sonraları fiziksel, kimyasal ve biyolojik sistemlerin matematiksel olarak modellenmesinde kullanılmıştır [2][3][4]. Wolfram, hücresel otomatları ayrı bir bilim dalı olarak görmüş ve üzerinde çalışmalar yapmıştır. Wolfram'a göre bu bilim, basit kuralları olan ve birbirlerini etkileyen basamaklardan oluşan bir sistemdir. Wolfram, kristallerin gelişimi, cisimlerin çatlaması, akışkan akışı, bitki ve hayvanların gelişimi gibi yaşamdan bazı olayları hücresel otomat ile modellemiştir [5]. İlerleyen süreçlerde hücresel otomatlar, rastgele sayı üreteçleri, görüntü işleme, görüntü şifreleme gibi birçok alanda kullanılmıştır.

Hücresel otomatlar, problemlere çözüm bulmada yetersiz kalmaya başlayınca, yapılarında değişiklikler yapılmaya başlanmıştır. Bu değişikliklerden biri de hücresel otomatlara hafıza eklenmesidir. Fredkin, hücresel otomatların hafızalı olarak çalışmasını sağlamıştır [6]. Böylece hafızalı hücresel otomatlar elde edilmiştir. Daha sonraları farklı hafızalı hücresel otomat yapıları elde edilmiştir [7]. R. Alonso-Sanz sıklık fonksiyonu adını verdiği bir fonksiyon tanımlayarak yeni bir hafızalı hücresel otomat yapısı önermiştir [8].

Göncü, hafızalı hücresel otomat yapısına rastgelelik katma üzerinde bir çalışma yapmıştır. Buna göre Göncü, R. Alonso-Sanz'ın önerdiği hafızalı hücresel otomat yapısını kullanarak hafızaya rastgelelik katmıştır [9]. Yani bir hücrenin önceki değerlerine bakılacağı zaman, hangisine bakılacağı konusunda rastgelelik sağlamıştır. Böylece elde edilen hücresel otomat yapısında rastgele sonuçlar alınmıştır.

Rastgele sayı üreteçleri, kriptoloji gibi rastgele sayı dizilerine ihtiyaç duyulan alanlarda kullanılan sistemlerdir [17]. Rastgele sayı üretme ile ilgili farklı yöntemler

kullanılmıştır [18]. Hücresel otomat yapıları da rastgele sayı üretme için kullanılan yapılardandır. Bu çalışmada, Göncü'nün önerdiği rastgele hafızalı hücresel otomat üzerinde değişiklikler yapılarak rastgelelik artırılmış ve buradan da rastgele sayı üreteci tasarımı yapılmıştır. Geliştirilen sistem FPGA üzerinde gerçeklenmiş ve sonuçlar alınmıştır.

İkinci bölümde, temel hücresel otomat yapısından bahsedilmiştir. Temel hücresel otomatın matematiksel modeli ve şekil olarak karşılığı verilmiştir. Hafızalı hücresel otomat ve rastgele hafızalı hücresel otomat yapıları ve bunların FPGA üzerinde gerçeklemeleri anlatılmıştır. Son olarak yeni tasarlanan rastgele hafızalı hücresel otomat yapısı ve onun FPGA üzerinde gerçeklemesi anlatılmıştır.

Üçüncü bölümde, rastgele sayı üreteçleri hakkında genel bir bilgi verilmiştir. Ardından bu çalışmada tasarlanan rastgele sayı üreteci anlatılmıştır. Ayrıca, bu tasarımda ihtiyaç duyulduğu için DDR2 SDRAM'den ve ona donanım ile nasıl erişildiğinden bahsedilmiştir.

Dördüncü bölümde, rastgele sayı üreteci üzerinde yapılan testler anlatılmış ve testlerden alınan sonuçlar verilmiştir. Sonuç kısmında, elde edilen sonuçlardan bahsedilmiş, bu çalışmayla ileride yapılabilecek çalışmalar hakkında önerilerde bulunulmuştur.

Tezde, Alt Bölüm 2.1, 2.2 ve 2.3'te daha önce yapılan çalışmalar anlatılmıştır. Alt Bölüm 2.4'ten itibaren bu tez kapsamında yapılan çalışmalar anlatılmıştır. Alt Bölüm 2.4'te bahsedilen yeni bir rastgele hafızalı hücresel otomat, Bölüm 3'te anlatılan rastgele sayı üreteci tasarımı ve Bölüm 4'te verilen rastgele sayı üreteci testleri bu tez kapsamında yapılan çalışmalardır.

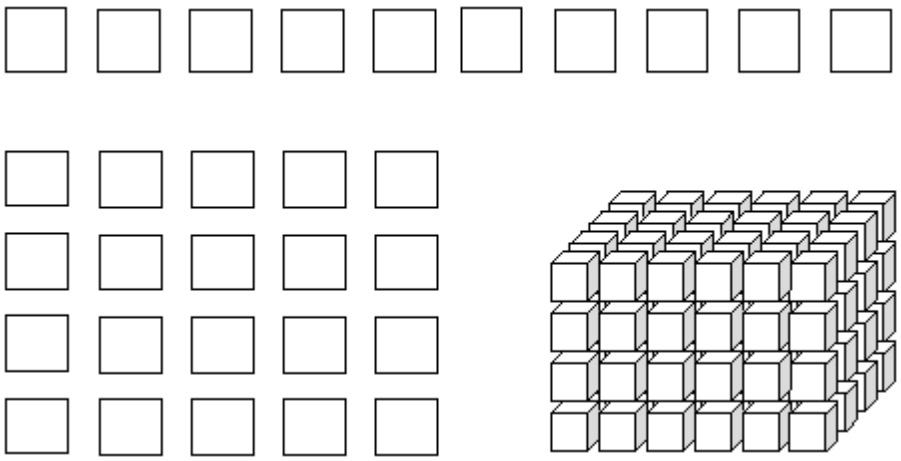
2. HÜCRESEL OTOMAT YAPILARI

Hücresel otomat, uzayda ve zamanda ayrık değerlere sahip hücrelerden oluşan bir latistir. Hücresel otomatta, her bir hücrenin bir anda aldığı değer, hücrenin o andaki durumunu gösterir. Böylece hücresel otomatın bir andaki durumu, hücrelerin o anda aldıkları değerlerdir. Bir hücrenin bir sonraki durumunu, hücrenin kendisinin ve komşularının şimdiki durumlarının bir fonksiyonu belirler. Hücresel otomatlarda bu fonksiyonlara kural denir.

Bütün hücrelerinin kuralları aynı olan hücresel otomatlara düzenli hücresel otomat, hücreleri farklı kurallara sahip hücresel otomatlara düzensiz hücresel otomat adı verilir.

Bütün hücrelerinin aynı anda durum değiştirdiği hücresel otomatlara senkron hücresel otomat, hücreleri farklı zamanlarda durum değiştiren hücresel otomatlara asenkron hücresel otomat adı verilir.

Hücresel otomat uzayları farklı boyutlarda olabilir. Şekil 2.1'de 1-boyutlu, 2-boyutlu ve 3-boyutlu hücresel otomat yapıları örnek olarak verilmiştir.



Şekil 2.1: 1, 2 ve 3 boyutlu hücresel otomat örnekleri [9].

d-boyutlu, düzenli bir hücresel otomat (Z^d,Q,V,f) dörtlüsü ile ifade edilebilir. Burada Z^d, hücresel otomatın bulunduğu uzayı ifade eder. Bu terimdeki d, hücresel otomatın boyutunu ifade eder. Q, durum kümesidir. Yani hücrelerin gidebileceği durumları gösterir. V, komşuluk vektörüdür.Yani hücresel otomatın kuralında, bir hücrenin bulunduğu durumdan bir sonraki duruma geçerken etkileştiği komşularını gösterir. f, bir hücrenin şimdiki durumundan bir sonraki durumuna geçiş fonksiyonudur. Yani hücresel otomatın kuralıdır.

$i \epsilon Z^d$, d boyutlu bir hücresel otomatın bir hücresinin uzaydaki yerini, α_i^T de hücrenin T anındaki durumunu göstermek üzere, hücrenin bir sonraki durumu (2.1)'deki gibi gösterilebilir.

$$\alpha_i^{T+1} = f(\alpha_{i-i1}^T, \ldots, \alpha_{i-is}^T),\ T = 0,1,2,\ldots \qquad \textbf{(2.1)}$$

Bu çalışmada bir boyutlu, düzenli ve senkron hücresel otomat yapısı kullanılmıştır. Bundan sonra kullanılan hücresel otomat ifadesi bir boyutlu, düzenli ve senkron hücresel otomat anlamında kullanılacaktır.

2.1 Temel Hücresel Otomat

Temel hücresel otomat; 1-boyutlu, 3 komşuluklu, 2 durumlu yapıdır. Her bir hücrenin alabileceği iki durum vardır. Bir hücrenin bir sonraki durumu, hücrenin kendisi ve en yakın iki komşusunun bir fonksiyonudur. Yani, her bir hücre belli bir kurala göre durum değişirir. Hücreler 2 durumlu olduklarından 0 veya 1 değerini alırlar. 3 komşuluklu bir yapıda yan yana bulunan 3 hücre ortadaki hücrenin bir sonraki durumunu belli bir kurala göre belirler. 3 komşu hücre 3 bitlik bir sayı gibi düşünüldüğünde $2^3 = 8$ farklı kombinasyon olabilir. Bu, Şekil 2.2'deki Wolfram notasyonu ile görülebilir.

111	110	101	100	011	010	001	000
0	1	0	1	1	0	1	0

Şekil 2.2: Hücresel otomatın kuralının Wolfram notasyonu ile gösterimi.

Burada 8 farklı kombinasyon için $2^8 = 256$ farklı sonuç elde edilir. Bu 256 farklı durum hüresel otomatın kuralı olarak adlandırılır. Bu kuralların isimleri 8 bitlik ikili değerin onluk sistemdeki karşılığı olarak verilir. Örneğin Şekil 2.2'deki kural 89 kuralıdır. Bir hücresel otomatın kuralı Şekil 2.3'teki gibi de gösterilebilir. Burada siyah kutucuklar 1 değerini, beyaz kutucuklar 0 değerini gösterir. Buna göre Şekil 2.3'teki kural 250 kuralıdır.

Şekil 2.3: 250 kuralının temsili gösterimi.

Aslında her bir kural lojik bir işlemle ifade edilir. Örneğin Şekil 2.4'te temsili gösterimi verilen Kural 150, (2.2)'deki gibi ifade edilir. ($(10010110)_2 = (150)_{10}$).

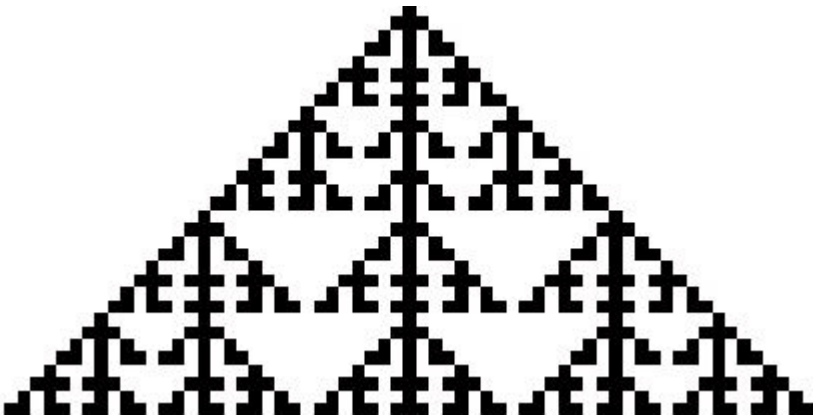
Şekil 2.4: Kural 150'nin temsili gösterimi.

$$\alpha_i^{T+1} = \alpha_{i-1}^{T} \; EXOR \; \alpha_i^{T} \; EXOR \; \alpha_{i+1}^{T} \qquad \textbf{(2.2)}$$

α_i^T**,** i hücresinin T anındaki durumunu, α_{i-1}^T ve α_{i+1}^T sırasıyla (i-1). hücrenin ve (i+1). hücrenin T anındaki durumlarını göstermektedir. α_i^{T+1}, i hücresinin T+1 anındaki, yani bir sonraki durumu ifade eder. Kural 150'nin doğruluk tablosu ve Karnough diyagramı yapıldığında (2.2)'deki ifade elde edilir. Benzer şekilde her kural için böyle lojik bir ifade elde edilebilir. Elde edilen bu lojik ifadeler sayısal gerçeklemeler esnasında kolaylıklar sağlar.

Hücresel otomatları grafiksel olarak göstermek de mümkündür. Örneğin Şekil 2.5'te Kural 150 grafiksel olarak gösterilmiştir. Şekil 2.5'te bir boyutlu, 64 hücreden oluşan bir hücresel otomat görülmektedir. Siyah renkli kareler 1 değerini alan hücreleri, beyaz renkli kareler 0 değerini alan hücreleri göstermektedir. Yatayda 64 adet kare bulunmaktadır. Bu kareler, 64 hücreden oluşan hücresel otomatı gösterir. Düşeydeki kareler ise hücresel otomatın zamandaki değişimini gösterir. Bu grafikte 64 hücreden

oluşan hücresel otomatın 32 iterasyon yaptığında zamandaki değişimi görülmektedir. Başlangıçta sadece 33. hücre 1 değerinde, diğer hücreler 0 değerindedir. Zamanla Kural 150'nin gerektirdiği şekilde hücreler değerlerini almıştır. Benzer şekilde diğer kurallar da grafiksel olarak gösterilebilir.



Şekil 2.5 : Üzerinde Kural 150 uygulanan 64 hücreden oluşan hücresel otomatın yaptığı 32 iterasyon sonucundaki durumunun grafiksel gösterimi.

Bu çalışmada, 64 hücreden oluşmuş bir boyutlu hücresel otomat yapısı kullanılmıştır. Kullanılan hücresel otomatlarda Kural 150 uygulanmıştır.

2.2 Hafızalı Hücresel Otomat

Temel hücresel otomatlarda, bir hücrenin bir sonraki durumu, hücrenin komşuluğundaki hücrelerin belli bir kurala göre işleme girmesiyle elde ediliyordu. Hücrelerin önceki değerlerine bakılmıyordu. Bu nedenle, bu hücresel otomatlara hafızasız hücresel otomat denilebilir.

Hücresel otomatlarda, bir hücrenin bir sonraki durumu hesaplanırken hücrelerin önceki değerlerine de bakılarak sistem hafızalı hale getirilmiştir. Böylece hafızalı hücresel otomatlar elde edilmiştir.

Ramon Alonso-Sanz ve Margarita Martin 2000'li yılların başlarında hafızalı temel hücresel otomat yapısını önermişlerdir. Hafızalı temel hücresel otomat, hafızasız temel hücresel otomat gibi üç komşuluklu ve iki durumludur. Ancak hücrelerin bir sonraki durumlarını elde ederken kullanılan fonkiyonda değişiklikler olmuştur.

Hafıza etkisini kullanabilmek için sıklık fonksiyonu adı verilen bir fonksiyon ortaya atılmıştır. (2.3)'te hücrelerin dinamik davranışı gösterilmiştir.

$$\alpha_i^{T+1} = f(s_{i-1}^T, s_i^T, s_{i+1}^T) \quad \textbf{(2.3)}$$

Burada s_i^T sıklık fonksiyonunu ifade etmektedir. Bu fonksiyon, i hücresinin T anına kadar almış olduğu bütün değerlerin en sık olanını gösterir. Yani hücrenin T anına kadar almış olduğu 1 değerlerinin sayısı, 0 değerlerinin sayısından fazla ise hücre 1 değerini alır. Tam tersi durumda hücre 0 değerini alır.

Hafızalı hücresel otomat en genel anlamda ifade edilecek olursa, sistemin dinamik davranışı (2.4) ve (2.5)'te gösterildiği gibi olur.

$$\alpha_i^{T+1} = f(m_{i-i1}^T, m_{i-i2}^T, \dots, m_{i-il}^T) \quad \textbf{(2.4)}$$

$$m_i^T = s^T(\alpha_i^{T-t1}, \dots, \alpha_i^{T-tj}) \quad \textbf{(2.5)}$$

(2.5)'teki s^T, i hücresinin j kadar önceki durumuna kadar aldığı tüm değerlerin sıklık fonksiyonunu göstermektedir. m_i^T bu sıklık fonksiyonunun aldığı değeri göstermektedir. (2.4)'teki f fonksiyonu i hücresinin komşuluğundaki tüm hücrelerin sıklık fonksiyonu sonucu aldıkları değerlerin, hücresel otomatın kuralına göre sonucunu verir. Bu sonuç da hücrenin bir sonraki durumunu gösterir.

(2.3) ve (2.5)'te verilen sıklık fonksiyonları hücrenin bütün önceki değerlerine bakarak hesaplanmıştı. Şimdi, yine Ramon Alonso-Sanz tarafından önerilen farklı bir sıklık fonksiyonu ele alınacaktır. Önerilen bu sıklık fonksiyonunda bütün önceki değerlere bakmak yerine, üç önceki duruma kadar olan değerlere bakılır. Yani, sıklık fonksiyonunun tanım kümesini, hücrenin şimdiki durumu, bir önceki durumu ve iki önceki durumu oluşturur. Bu fonksiyona minimal hafızalı sıklık fonksiyonu adı verilir. Minimal hafızalı sıklık fonksiyonun dinamiği (2.6) ve (2.7)'de ifade edildiği gibidir.

$$m_i^T = s^T(\alpha_i^{T-2}, \alpha_i^{T-1}, \alpha_i^T),\ T=2,3\ldots \quad \textbf{(2.6)}$$

$$m_i^T = \alpha_i^T,\ T=0,1 \quad \textbf{(2.7)}$$

Minimal hafızalı sıklık fonksiyonunu da bir kural ile ifade etmek mümkündür. Şekil 2.6'te görüldüğü gibi minimal hafızalı sıklık fonksiyonu Wolfram notasyonu ile gösteridiğinde Kural 232'e karşılık gelmektedir. ($(11101000)_2 = (232)_{10}$)

111	110	101	100	011	010	001	000
1	1	1	0	1	0	0	0

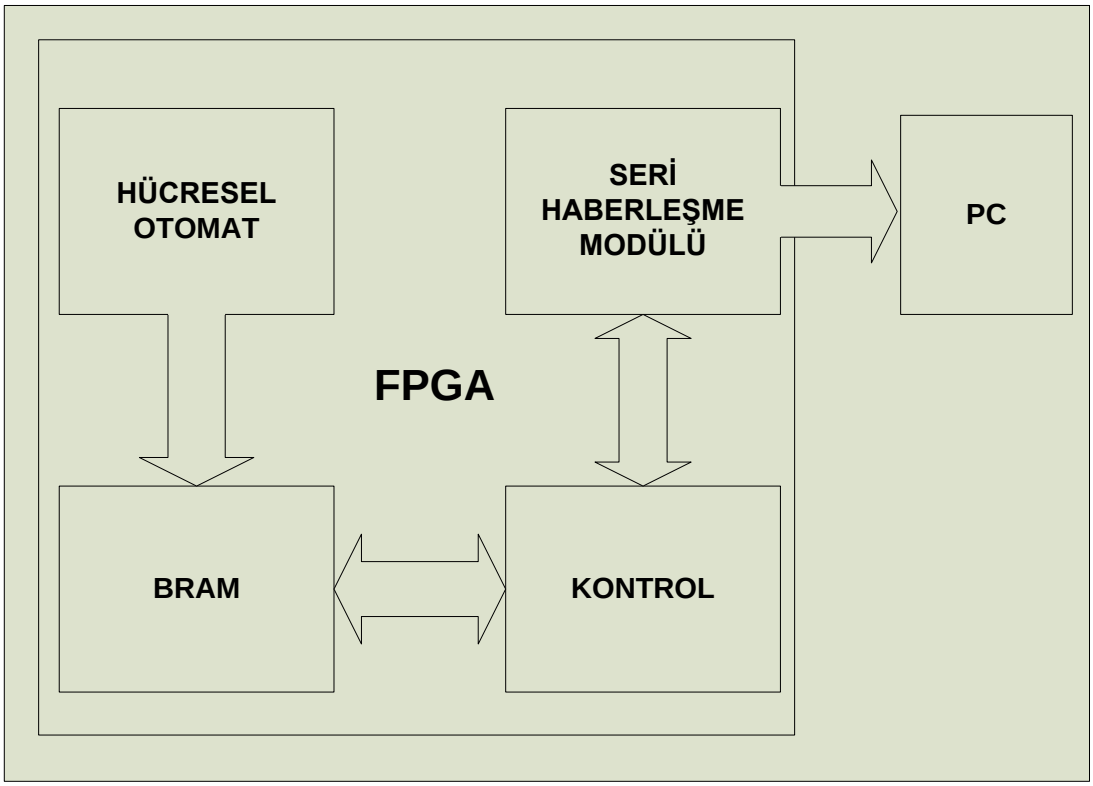
Şekil 2.6: Minimal sıklık fonksiyonunun Wolfram notasyonu ile gösterilimi.

Minimal hafızalı sıklık fonksiyonunda üç adım önceki duruma kadar bakıldığı için, ilk üç adımda hafızadan söz edilemez. Minimal hafızalı sıklık fonksiyonuna sahip hücresel otomatta ilk üç adım hafızasız hücresel otomat gibi çalışır. Üçüncü adımdan sonra hafızalı bir şekilde çalışmaya başlar.

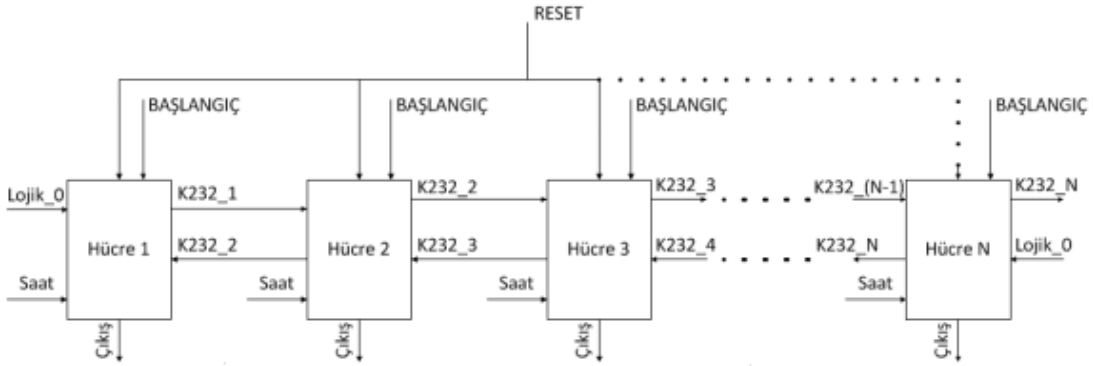
2.2.1 Hafızalı hücresel otomatın sayısal gerçeklemesi

Bir boyutlu, üç komşuluklu, iki durumlu, minimal sıklık fonksiyonuna ve Kural 150'ye sahip hafızalı hücresel otomat FPGA üzerinde sayısal olarak gerçeklenmiştir. Gerçeklenen sistemin genel blok diyagramı Şekil 2.7'de görülmektedir.

FPGA üzerinde tasarlanan sistem 4 ana modülden oluşmuştur. Hücresel Otomat modülünde, hücresel otomat yapısı tasarlanmıştır. Bu blokta başlangıç durumu verilen bir hücresel otomatın parametrik olarak girilen iterasyon sayısı kadar çalışması sağlanmıştır. Hücresel otomatın her durumu diyagramda BRAM olarak gözüken Blok RAM'e yazılmıştır. Kontrol modülü, Blok RAM ile Seri Haberleşme modülü arasındaki veri akışını kontrol eder. Seri Haberleşme modülü, hücresel otomatın aldığı tüm değerleri bilgisayar ortamına aktarır. Böylece hücresel otomatın örüntüsü bilgisayarda görülmüş olur.



Şekil 2.7: Hafızalı hücresel otomat sisteminin genel blok diyagramı.

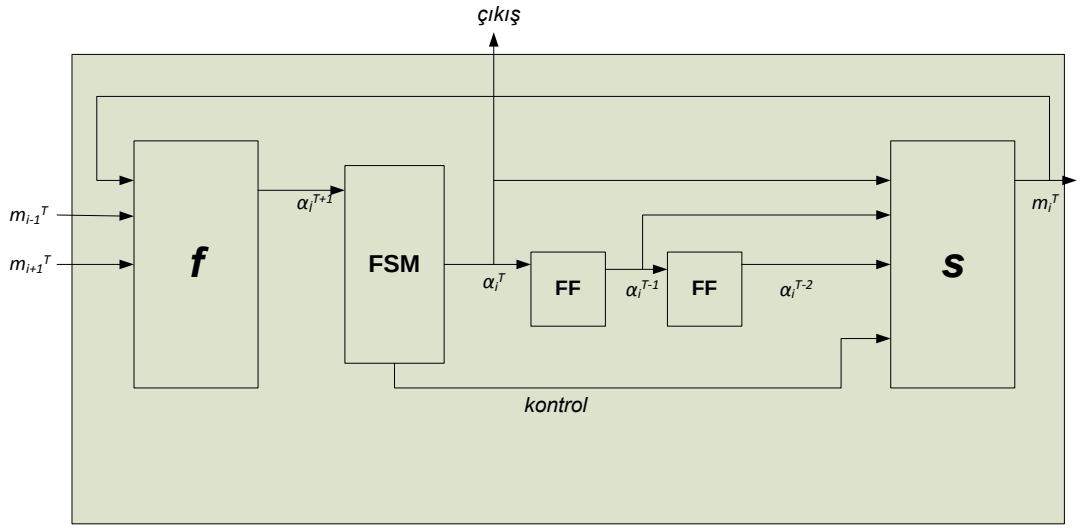


Şekil 2.8: Hücresel otomat modülünün iç yapısı [9].

Şekil 2.8'de hücresel otomat modülünün iç yapısı gösterilmiştir. Burada hücresel otomatı oluşturan n tane hücre ve bu hücreler arasındaki bağlantılar görülmektedir. Her bir hücrenin saat ve reset girişleri, başlangıç durumunu veren bir giriş, hücrenin sağındaki ve solundaki hücrelerden gelen, o hücrelerin sıklık fonksiyonları sonucu aldıkları değeri taşıyan girişleri bulunmaktadır. Çıkışlar, hücrenin kuralına göre

gittiği son durum ve sıklık fonksiyonu sonucu aldığı değeri sağındaki ve solundaki hücrelere taşıyan bir işaretlerdir.

Oluşturulan sistemde sabit sınır koşulları kullanılmıştır. Yani birinci hücrenin solunda ve sonuncu hücrenin sağında lojik 0 değerini taşıyan birer komşu hücre var gibi düşünülmüştür.



Şekil 2.9: Hafızalı hücresel otomatta bir hücrenin iç yapısı [9].

Şekil 2.9'da her bir hücrenin iç yapısı gösterilmiştir. Buna göre bir hücre iki tane kombinezonsal devre, bir sonlu durum makinesi ve iki flip-floptan oluşmaktadır. *f* olarak isimlendirilen kombinezonsal devre hücresel otomatın kuralını gerçekleyen devredir. *s* olarak isimlendirilen kombinezonsal devre ise sıklık fonksiyonunu gerçekleyen devredir. Sonlu durum makinesi (FSM) hücrenin o anda bulunduğu durumundaki değerini tutar. Ayrıca hücreye başlangıç değerinin verilmesini sağlar. Yine sistemde bulunan iki flip-flop ise hücrenin bir önceki ve iki önceki değerlerini tutar.

f devresine, *s* devresinden gelen hücrenin o andaki sıklık fonksiyonu sonucu olan değer (m_i^T), ve komşu hücrelerin o andaki sıklık fonksiyonları sonucu aldıkları değerler (m_{i-1}^T, m_{i+1}^T) girer. *f* devresinden hücresel otomatın kuralı sonucu alınan değer çıkış olarak verilir ve bu değer FSM'de tutulur. *s* devresine FSM'de tutulan hücrenin şimdiki değeri (α_i^T), flip-floplarda tutulan hücrenin bir önceki (α_i^{T-1}) ve iki önceki (α_i^{T-2}) değerleri ve FSM'den gelen kontrol işareti girer. Kontrol işareti,

hafızalı hücresel otomatın ilk üç adımında hafızasızmış gibi değer alabilmesini sağlar. (Hafızalı hücresel otomatın ilk üç adımında hafızadan bahsedilemeyeceği daha önce belirtilmişti). Çıkış olarak sıklık fonksiyonunun sonucu verilir ve bu değer *f* devresine gider. *f* ve *s* devreleri, kapı seviyesinde değil de FPGA üzerinde hazır bulunan, doğruluk tablosunu direk olarak gerçekleyebilen LUT'larla tasarlanmıştır. *f* ve *s* devrelerinin doğruluk tabloları Çizelge 2.1 ve Çizelge 2.2'de verilmiştir.

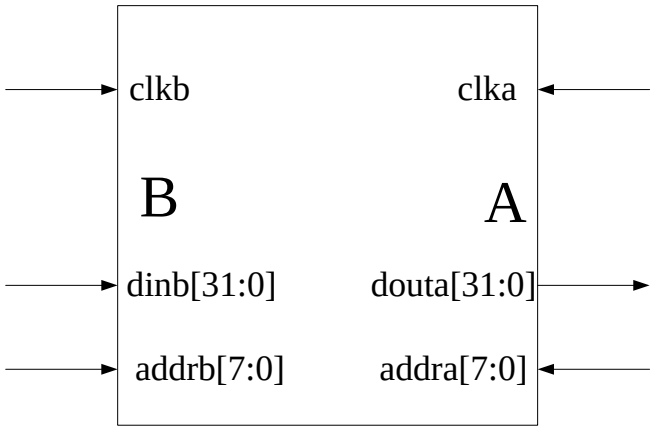
Çizelge 2.1: Kural 150'yi gerçekleyen *f* devresinin doğruluk tablosu [9].

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 2.2: Sıklık fonksiyonunu gerçekleyen *s* devresinin doğruluk tablosu [9].

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

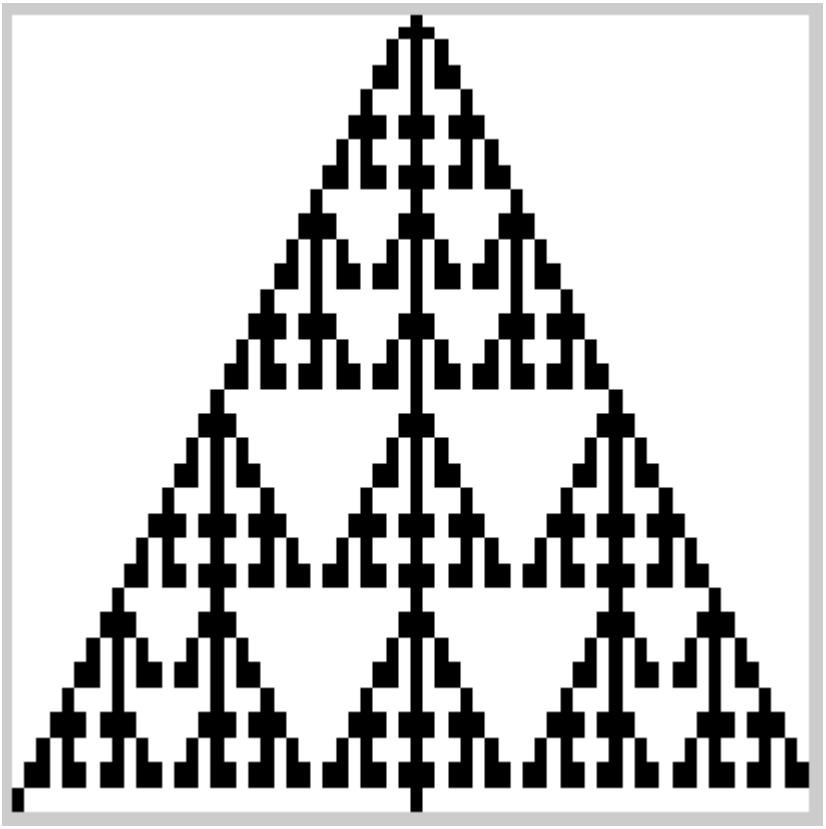
Hücresel otomat, her saat darbesinde durum değiştirir ve her durumda aldığı değerler Blok RAM'e yazılır. FPGA üzerinde hazır olarak bulunan Blok RAM'in genel yapısı Şekil 2.10'da görülmektedir.



Şekil 2.10: Blok RAM'in şematiği.

Kontrol modülü, Blok RAM'den UART'a veri akışındaki kontrolü sağlar. UART verileri bilgisayara 8 bitlik gruplar halinde yani bayt bayt ve 115200 bps hızında gönderir. Bilgisayara gönderilen veriler MATLAB ortamına alınır. MATLAB'da yazılan kodla hücresel otomatın örüntüsü bir şekil halinde bilgisayarda görülmüş olur.

Şekil 2.11'de hafızalı hücresel otomatın bilgisayarda elde edilen örüntüsü görülmektedir. Burada hafızalı hücresel otomat 64 hücreden oluşmaktadır. Yataydaki siyah beyaz kutucuklar bu 64 hücreyi göstermektedir. Siyah kutucuklar 1 değerini, beyaz kutucular 0 değerini ifade etmektedir. Başlangıç değeri olarak, sadece 33. hücreye 1 değeri verilmiş, diğer hücrelere 0 değeri verilmiştir. Düşeydeki kutucuklar hafızalı hücresel otomatın zamandaki değişimini göstermektedir. Burada 64 iterasyon yapılmıştır. İterasyon sayısı parametrik olup, BRAM'in izin verdiği ölçüde farklı değerler girilebilir.



Şekil 2.11: 64 hücreden oluşan hafızalı hücresel otomatın 64 iterasyon sonucunda aldığı değerleri gösteren bilgisayar çıktısı.

Bu sayısal sistem, Digilent firmasının ATLYS geliştirme kartı üzerinde bulunan, Xilinx firmasının ürettiği Spartan-6 ailesinden XC6SLX45-CSG324C FPGA üzerinde gerçeklenmiştir. Sistemin saat frekansı 100 MHz olarak seçilmiştir.

Hücresel otomat iterasyon sayısı kadar çalışır ve bütün değerleri Blok RAM'e kaydedilir. Sonra Blok RAM'deki değerlerin tamamının UART modülü ile bilgisayar ortamına aktarılmasını bekler. UART modülünden gelen bilgisayara iletimin bittiğini gösteren işareti aldığında tekrar çalışmaya başlar.

2.3 Rastgele Hafızalı Hücresel Otomat

Temel hücresel otomat ve hafızalı hücresel otomat sistemlerinde, daha çok, düzenli yapılar elde edilir. Hücresel otomatları rastgele sayı üreteci olarak kullanmak için sisteme rastgelelik katılması gerekmektedir. Bahsedilen bu rastgeleliği elde etmek

için Göncü bir sistem önermiştir. Göncü'nün önerdiği sistemde hafızalı hücresel otomat yapısına rastgelelik katılmıştır [9].

Rastgele hafızalı hücresel otomat da hafızalı hücresel otomatta olduğu gibi, bir boyut uzayına, komşuluk kümesine, sonlu durum kümesine, geçiş fonksiyonuna, sıklık fonksiyonuna sahiptir.

Rastgele hafızalı hücresel otomat da Alt bölüm 2.2'de tasarlanan hafızalı hücresel otomatta olduğu gibi bir boyutlu bir yapıya sahiptir. Üç komşuluklu ve iki durumludur. Yine geçiş fonksiyonu olarak Kural 150 kullanılmıştır. Sıklık fonksiyonu da yine kullanılmıştır. (2.8)'de verilen sıklık fonksiyonunda t_1 *ve* t_2 {0,1,2} kümesinin elemanı olup zamanla değişebilen rastgele değerler alabilmektedirler.

$$m_i^T = s^T(\alpha_i^{T-2}, \alpha_i^{T-1}, \alpha_i^T) \qquad \textbf{(2.8)}$$

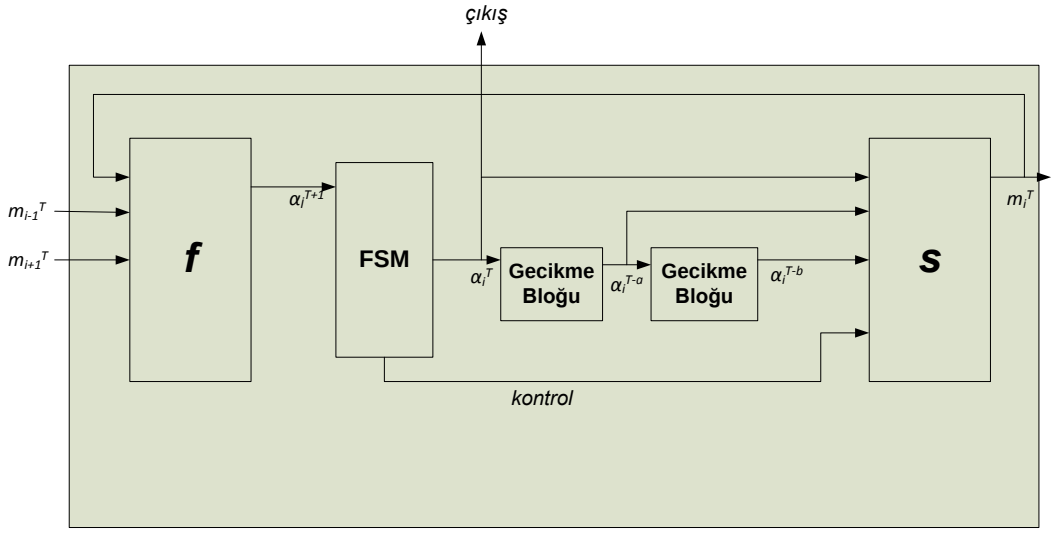
2.3.1 Rastgele hafızalı hücresel otomatın sayısal gerçeklemesi

Rastgele hafızalı hücresel otomatın FPGA üzerinde sayısal gerçeklemesi yapılırken, bir önceki bölümde anlatılan hafızalı hücresel otomatın FPGA üzerinde tasarlanan yapısı kullanılmıştır. Rastgele hafızalı hücresel otomatın sayısal tasarımının, hafızalı hücresel otomatın sayısal tasarımından farkı, Şekil 2.9'da bir hücrenin iç yapısını gösteren tasarımda hücrenin bir önceki ve iki önceki durumunu tutan flip-flopların yerine gecikme bloklarının kullanılmasıdır.

Şekil 2.12'de rastgele hafızalı hücresel otomatın bir hücresinin sayısal tasarımı görülmektedir. Bahsedildiği gibi flip-floplar yerine gecikme blokları kullanılmıştır.

Gecikme blokları FPGA üzerinde bulunan LUT'ların art arda konulmasıyla elde edilmiştir. Çevresel etkenlerden dolayı gecikme blokları farklı zamanlarda farklı gecikme değerleri alabilmektedirler. Böylece *s*, yani sıklık fonksiyonunu gerçekleyen kombinezonsal devrenin girişlerinde beklenen hücrenin bir önceki ve iki önceki durum değerleri yerine farklı zaman adımlarındaki değerler gelebilir. Şekil 2.12'de birinci gecikme bloğunun çıkışında görülen α_i^{T-a} olarak isimlendirilen işaretteki a

değişkeni 0 veya 1 değerini alabilir. Böylece gecikmenin süresine göre, birinci gecikme bloğunun çıkışı hücrenin şimdiki veya bir önceki değerini alabilir. İkinci gecikme bloğunun çıkışında görülen α_i^{T-b} olarak isimlendirilen işaretteki b değişkeni de 1 veya 2 değerini alabilir. Böylece gecikmenin süresine göre, ikinci gecikme bloğunun çıkışı hücrenin bir önceki veya iki önceki değerini alabilir. Böylece birinci gecikme bloğunun çıkışında hücrenin şimdiki ve bir önceki durumu arasında, ikinci gecikme bloğunun çıkışında ise bir önceki ve iki önceki durumu arasında rastgelelik sağlanmış olur.



Şekil 2.12: Rastgele hafızalı hücresel otomatın bir hücresinin yapısı [9].

Bahsedilen bu rastgeleliğin iyi bir şekilde sağlanabilmesi için gecikme bloklarının gecikme sürelerinin iyi ayarlanması gerekir. Gecikme süresi de, gecikme bloklarında art arda konulan LUT sayısıyla alakalıdır. Gecikme bloklarındaki LUT sayısı bir saat periyoduna yakın olacak şekilde ayarlanmalıdır. Böylece birinci gecikme bloğunun çıkışında bazen hücrenin şimdiki durumu, bazen de bir önceki durumu görülür. İkinici gecikme bloğunun çıkışında da bazen hücrenin bir önceki durumu, bazen de iki önceki durumu görülür. Böylece hafızada rastgelelik sağlanmış olur. Alt Bölüm 2.4'te yeni bir rastgele hafızalı hücresel otomat yapısı anlatılırken, flip-floplu tasarımla gecikme bloklu yapıdaki işaretler arasındaki ilişki, Şekil 2.16'daki zaman diyagramında gösterilecektir. Bu yapının mantığını anlamak için de bu zaman diyagramı incelenebilir.

Rastgele hafızalı hücresel otomatın sayısal tasarımı, hafızalı hücresel otomatın sayısal tasarımı üzerinden yapılmıştır. Aradaki tek fark hafızalı hücresel otomatta hücrenin bir önceki ve iki önceki değerlerini tutan flip floplar yerine, rastgele hafızalı hücresel otomatta gecikme bloklarının kullanılmış olmasıdır. Sistem yine aynı FPGA üzerinde gerçeklenmiştir. Hücresel otomatın zamanla aldığı tüm değerler seri haberleşme modülü ile bilgisayar ortamına alınmış ve MATLAB ortamında şekle dönüştürülmüştür. Şekil 2.13'te 64 hücreden oluşan rastgele hafızalı hücresel otomatın 64 adımda aldığı değerler görülmektedir. Şekil 2.11'deki hafızalı hücresel otomatın durumuyla karşılaştırıldığında, Şekil 2.13'teki rastgele hafızalı hücresel ototmatta rastgeleliğin sağlanmış olduğu görülmektedir.

Şekil 2.13: 64 hücreden oluşan hafızalı hücresel otomatın 64 iterasyon sonucunda aldığı değerleri gösteren bilgisayar çıktısı.

Şekil 2.14'te farklı bir zamanda çalıştırılan hücresel otomatın görüntüsü verilmektedir. Görüldüğü gibi Şekil 2.13'teki görüntüden farklı bir görüntü elde edilmiştir. Bu da gecikme bloklarının zamanla sıcaklık vs gibi çevre etkileriyle değiştiğini göstermektedir.

Şekil 2.14: Farklı bir zamanda çalıştırılan 64 hücreden oluşan hafızalı hücresel otomatın 64 iterasyon sonucunda aldığı değerleri gösteren bilgisayar çıktısı.

2.4 Yeni Bir Rastgele Hafızalı Hücresel Otomat

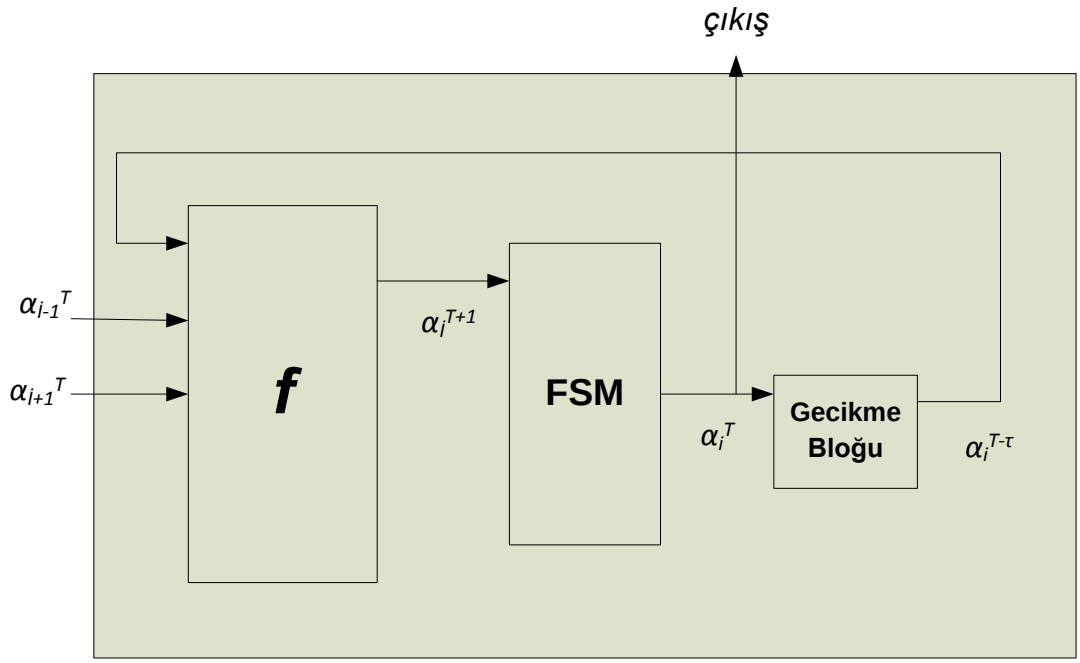
Bu bölümde yeni bir rastgele hafızalı hücresel otomat yapısı önerilmiştir. Bir önceki bölümde tasarlanan rastgele hafızalı hücresel otomat yapısında sıklık fonksiyonu kullanılmıştır. Bu sıklık fonksiyonunda da hücrenin iki önceki değerine kadar olan değerlere, yani hücrenin şimdiki, bir önceki ve iki önceki değerlerine bakılıp bu değerlerden en sık olanı hücrenin değeri olarak belirlenmiştir. Yeni önerilen rastgele hafızalı hücresel otomatta sıklık fonksiyonu kullanılmamıştır. Bunun yerine hücrenin değeri belirlenirken, şimdiki ve bir önceki durum değerlerine bakılır. Hücrenin şimdiki ve bir önceki değerleri arasında bir rastgelelik sağlanmışır.

2.4.1 Yeni bir rastgele hafızalı hücresel otomatın sayısal gerçeklemesi

Yeni bir rastgele hafızalı hücresel otomatın sayısal tasarımında, Alt bölüm 2.3'te tasarlanan rastgele hafızalı hücresel otomattan farklılıklar vardır. Öncelikle yeni

tasarımda sıklık fonksiyonu olmadığı için önceki tasarımda *s* diye isimlendirilen sıklık fonksiyonunu gerçekleyen kombinezonsal devre bulunmamaktadır. Ayrıca, hücrenin alacağı değer belirlenirken şimdiki ve bir önceki durumlarına bakılacağı için sadece bir tane gecikme bloğu kullanılmıştır.

FPGA'de tasarlanan sistem önceki tasarlanan sistemle aynı olup, hücrelerin iç yapısındaki tasarımda farklılıklar vardır. Buna göre yeni tasarlanan sistemde bir hücrenin iç yapısı Şekil 2.15'te görülmektedir. Görüldüğü gibi sistemde bir gecikme bloğu bulunmaktadır.

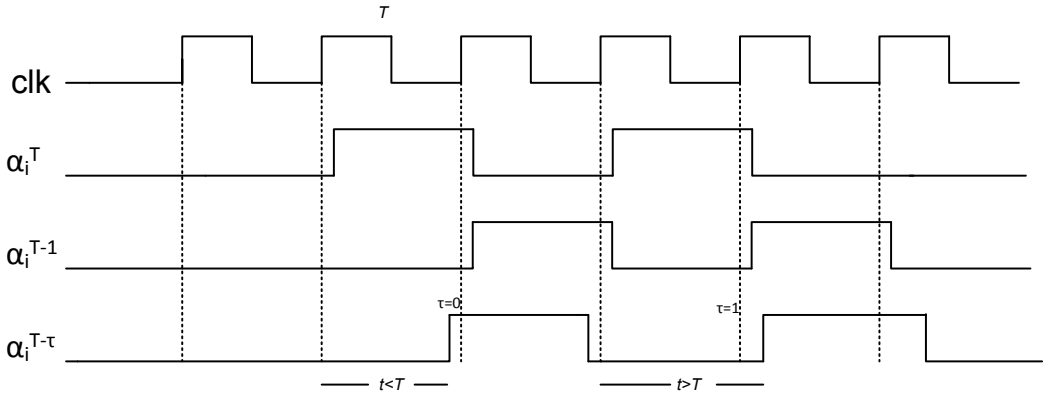


Şekil 2.15 : Yeni bir rastgele hafızalı hücresel otomatın bir hücresinin iç yapısı.

Gecikme bloğu iki farklı şekilde tasarlanmıştır. Bunların birincisinde, daha önceki tasarımda olduğu gibi FPGA içerisinde bulunan LUT'lar kullanılmıştır. LUT'ların içerisinde herhangi bir lojik işlem yapılmayıp LUT'lar sadece art arda gelecek şekilde bir dizi halinde birbirine bağlanmıştır. Bu sayede bir sinyal art arda gelen bu LUT dizisinden geçinceye kadar bir gecikme oluşmuş olur. Elde edilen bu gecikmenin değeri art arda kullanılan LUT sayısına göre değişmektedir. Hücrenin şimdiki durumu ile bir önceki durumu arasında rastgelelik sağlanması için gecikme bloğunun gecikme değeri saat periyoduna yakın olacak şekilde ayarlanmıştır. Art arda konulan LUT sayısı değiştirilerek en uygun gecikme elde edilmeye çalışılmıştır.

Buna göre en uygun LUT sayısı 22 olarak tespit edilmiştir. Elde edilen gecikme değeri çevresel faktörlerin de etkisiyle zamanla değişiklik gösterir. Böylece hücrenin şimdiki ve bir önceki değeri arasında rastgelelik sağlanmış olur.

Gecikme bloğunun tasarlandığı ikinci farklı tasarım ise NOT kapılarıyla yapılan tasarımdır. Buna göre gecikme bloğu elde edebilmek için NOT kapıları art arda bağlanarak bir NOT kapısı dizisi oluşturulmuştur. Burada önemli olan husus, art arda yerleştirilen NOT kapılarının sayısının çift sayı olmasıdır. Bu sayede NOT kapısı dizisinin girişine verilen işaret, değerini değiştirmeden çıkışa gelir. Ayrıca NOT kapısı dizisi oluşturulurken art arda bağlanan NOT kapısı sayısı önem arz etmektedir. NOT kapısı sayısı, bu NOT kapısı dizisinden elde edilecek gecikme değeri saat periyoduna yaklaşık olacak şekilde seçilmelidir. Buna göre NOT kapısı en uygun 22 olarak belirlenmiştir. Böylece çevresel faktörlerin de etkisiyle elde edilen gecikme değeri zamanla değişiklik gösterir. Sonuçta hücrenin şimdiki ve bir önceki değerleri arasında rastgelelik sağlanmış olur.



Şekil 2.16: Yeni bir rastgele hafızalı hücresel otomatın bir hücresindeki işaretlerin zaman diyagramında gösterilmesi.

Şekil 2.16'da, Şekil 2.15'te blok diyagramı verilen yeni bir rastgele hafızalı hücresel otomatın bir hücresindeki işaretlerin zaman diyagramında gösterimi verilmiştir. Burada *clk* saat işaretini, α_i^T i hücresinin şimdiki durumunu, α_i^{T-1} i hücresinin bir önceki durumunu, $\alpha_i^{T-\tau}$ gecikme bloğunun çıkışını göstermektedir. $\alpha_i^{T-\tau}$ ifadesindeki τ

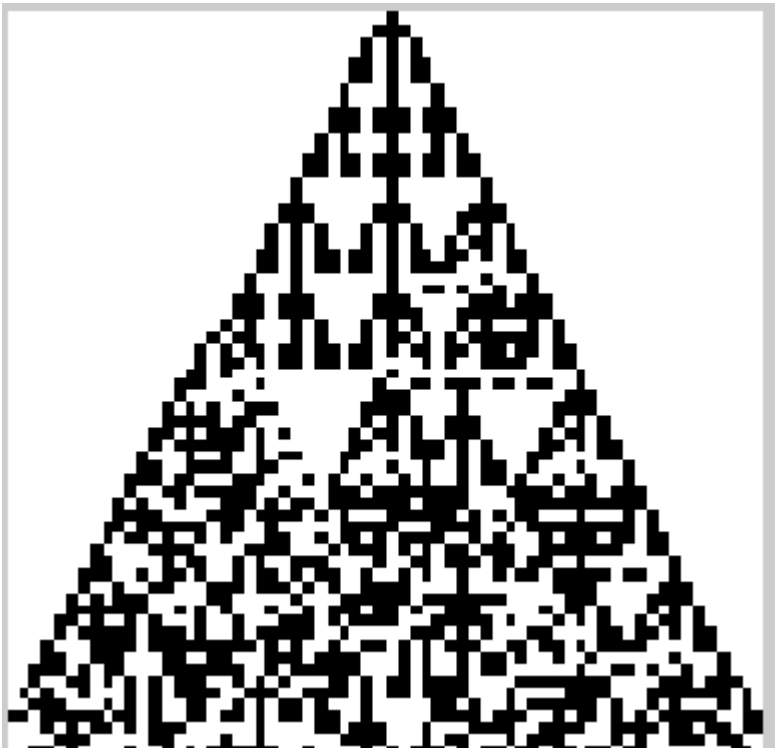
değeri 0 veya 1 değerini alabilmektedir. Yani gecikme bloğunun çıkışı α_i^T veya α_i^{T-1} olabilmektedir. Bu da gecikme bloğunun gecikme süresine bağlı olarak değişmektedir. Şekil 2.17'deki zaman diyagramında görüldüğü gibi gecikme süresi *t*, saat periyodu *T*'den küçük olursa ($t<T$), saatin yükselen kenarında gecikme bloğunun çıkışı $\alpha_i^{T-\tau}$, α_i^T değerini almış olur. Yani τ, 0 değerini almış olur. Eğer $t>T$ olursa saatin yükselen kenarında $\alpha_i^{T-\tau}$, α_i^{T-1} değerini almış olur. Yani τ, 1 değerini almış olur. Bu yüzden, daha önce de bahsedildiği gibi, tasarlanan gecikme bloklarında LUT ve NOT kapısı sayıları, gecikme değeri saat periyoduna yakın olacak şekilde seçilmiştir. Böylece gecikme bloğunun çıkışı bazen hücrenin şimdiki değerini bazen bir önceki değerini alabilmektedir.

Şekil 2.17'de FPGA üzerindeki LUT'lar kullanılarak yeni tasarlanan rastgele hafızalı hücresel otomatın FPGA üzerinde çalıştırılmasıyla elde edilen değerlerinin şekil olarak karşılığı görülmektedir. Hücresel otomat yine 64 hücreden oluşmuş olup 64 iterasyon çalıştırılmıştır. Elde edilen görüntü, önceki rastgele hafızalı hücresel otomattan elde edilen görüntüye göre daha karmaşıktır. Yani yeni sistemde daha iyi derecede bir rastgelelik elde edilmiştir.

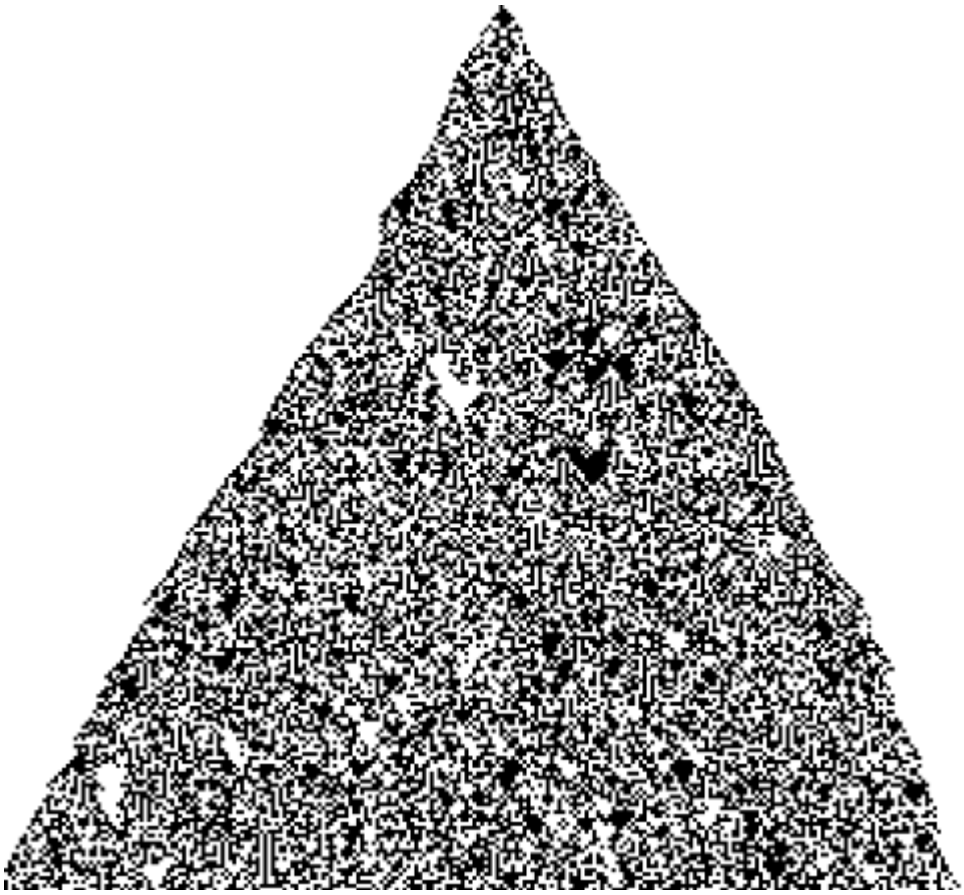
Şekil 2.18'de NOT kapıları kullanılarak yeni tasarlanan rastgele hafızalı hücresel otomatın FPGA üzerinde çalıştırılmasıyla elde edilen değerlerinin şekil olarak karşılığı görülmektedir. Hücresel otomat yine 64 hücreden oluşmuş olup 64 iterasyon çalıştırılmıştır. Elde edilen görüntü önceki rastgele hafızalı hücresel otomattan elde edilen görüntüye göre daha karmaşıktır.

Ayrıca, bigisayar üzerinde de rastgele hafızalı hücresel otomat yapısı çalıştırılmıştır. FPGA üzerinde tasarlanan sistemde, komşu üç hücrenin herbirinin şimdiki ve bir önceki değeri arasında rastgele değerler alınmıştı. Bilgisayar üzerinde çalıştırılan sistemde üç komşu hücrenin sadece birinin şimdiki ve bir önceki değeri arasında rastgele değerler alınmıştır. Bu rastgele değer alma da C'nin kendi rastgele sayı üretme komutuyla sağlanmıştır. Elde edilen sonuç Şekil 2.19'de görülmektedir.

Şekil 2.17: FPGA üzerindeki LUT'lar kullanılarak tasarlanan yeni bir rastgele hafızalı hücresel otomattan elde edilen görüntü.



Şekil 2.18: NOT kapıları kullanılarak tasarlanan yeni bir rastgele hafızalı hücresel otomattan elde edilen görüntü.



Şekil 2.19: Bilgisayar üzerinde çalıştırılan rastgele hafızalı hücresel otomatın sonuç çıktısı.

3. RASTGELE SAYI ÜRETECİ TASARIMI

Bu bölümde önce genel olarak rastgele sayı üreteçlerinden, daha sonra da bu çalışmada tasarlanan rastgele hafızalı hücresel otomat yapısı ile rastgele sayı üretecinden bahsedilecektir.

3.1 Rastgele Sayı Üreteçleri

Rastgele sayı üreteçleri günümüzde önemli bir yeri olan sistemlerdir. Özellikle kriptoloji alanında yaygın bir şekilde kullanılmaktadır. Kriptoloji alanında, bilgi güvenliğinin sağlanması için uzun rastgele sayı dizilerine ihtiyaç duyulabilmektedir. Dolayısıyla rastgele sayı üretme, yıllardır üzerinde çalışılan bir konudur.

Rastgele sayı üreteçleri, sözde rastgele sayı üreteçleri ve gerçek rastgele sayı üreteçleri olmak üzere ikiye ayrılır. Sözde rastgele sayı üreteçleri, belirli bir başlangıç koşulunu referans alarak rastgele sayı dizilerini üretir. Ancak bir müddet sonra kendini tekrar etmeye başlayarak periyodikleşir. Bu yüzden sözde rastgele sayı üreteçleri fazla güvenilir değildir. Gerçek rastgele sayı üreteçleri ise farklı yöntemlerle tasarlanabilir. Osilatör örnekleme, kaotik devreler gibi sistemlerle rastgele sayı dizileri üretirler [11][12][18].

Son zamanlarda hücresel otomat yapılarıyla da gerçek rastgele sayı üreteci tasarımı yapılmaktadır [10][15][16]. Bu çalışmada da farklı bir yöntem kullanılarak, hafızalı hücresel otomat yapısı ile rastgele sayı üreteci tasarımı yapılmıştır.

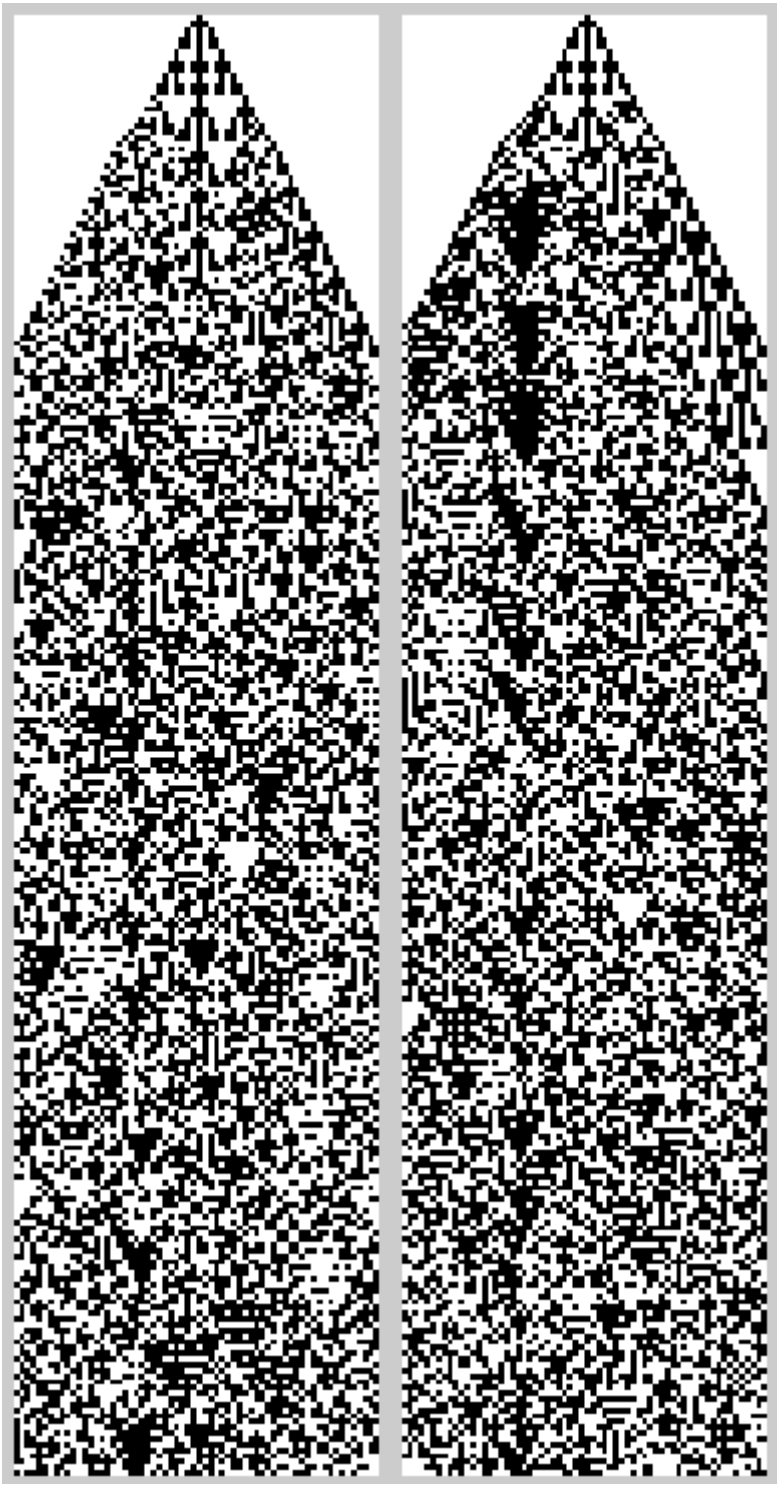
3.2 Hafızalı Hücresel Otomat Yapısı ile Rastgele Sayı Üreteci Tasarımı

Bölüm 2'de farklı hücresel otomat yapılarından bahsedilmişti. En son yeni bir rastgele hafızalı hücresel otomat yapısı tasarlanmış ve hücresel otomatın ilerleyen zaman adımlarında rastgele değerler aldığı görülmüştü. Bu sistem, iterasyon sayısı artırılarak tekrar çalıştırıldığında hücresel otomatın rastgele olarak yayılıma devam

ettiği görülmüştür. Örneğin, Şekil 3.1'de LUT'lu tasarımın iterasyon sayısı 256 olarak seçildiğinde elde edilen hücresel otomatın farklı zamanlarda çalıştırılan yayılımları görülmektedir. Aynı şekilde Şekil 3.2'de de NOT kapılı tasarımın iterasyon sayısı 256 olarak seçildiğinde elde edilen hücresel otomatın farklı zamanlarda çalıştırılan yayılımları görülmektedir.

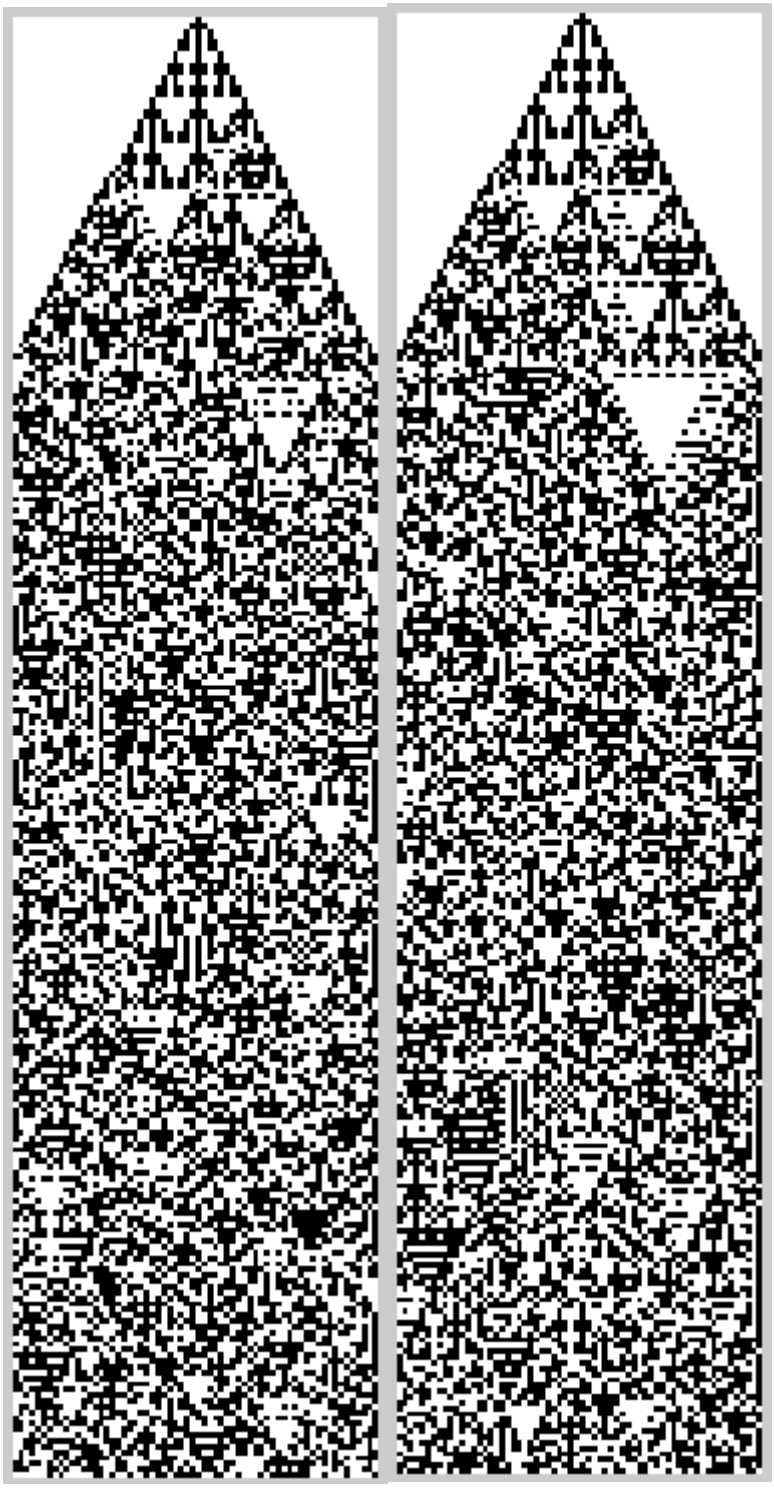
Şekil 3.1 ve Şekil 3.2'de iyi derecede rastgelelik sağlandığı görülmektedir. Böylece, her iki tasarımın da rastgele sayı üreteci tasarımında kullanılabileceği öngörülmüştür.

Bir rastgele sayı üreteci tasarlandığında, bunun gerçekten rastgele sayı üretip üretmediğinin anlaşılabilmesi için, üretilen rastgele sayı dizisinin belirli testlerden geçmesi gerekir. Bu testlerin yapılabilmesi için, teste girecek olan rastgele sayı dizisinin yüksek miktarda eleman içermesi gerekmektedir. Tasarlanan rastgele hafızalı hücresel otomat sisteminde, veriler Blok RAM'de saklanmaktadır. Hücresel otomat yüksek iterasyon sayılarında çalıştırılıp elde edilen yüksek miktardaki verilerin belleğe yazılması ve daha sonra da bellekten okunması gerekmektedir. Blok RAM'ler FPGA içerisinde hazır olarak bulunan belleklerdir. FPGA içerisindeki Blok RAM'ler testler için gerekli yüksek miktardaki verileri saklayacak kapasiteye sahip değildirler. Bu yüzden testlerin yapılabilmesi için bellek ihtiyacı doğmuştur. Bellek ihtiyacını karşılamak için sistemin tasarlandığı Spartan-6 FPGA'in üzerinde bulunduğu, Digilent'in Atlys geliştirme kartı üzerinde yer alan DDR2 SDRAM kullanılmıştır. Böylece, sistemin yeni blok diyagramı Şekil 3.3'teki gibi olmuştur.



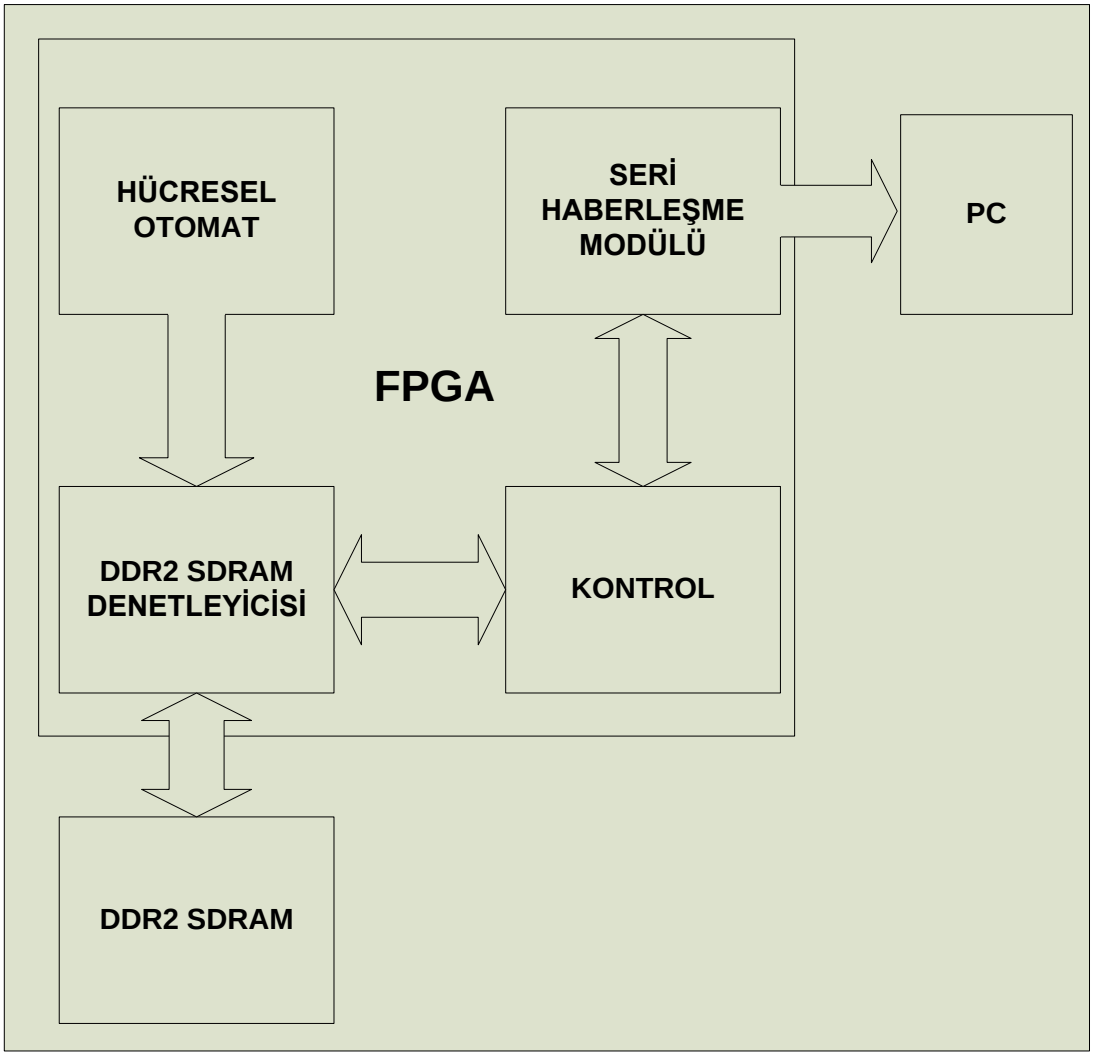
(a) (b)

Şekil 3.1: Farklı zamanlarda 256 iterasyon çalıştırılan, LUT'lu rastgele hafızalı hücresel otomat sonuçları (a), (b).



(a) (b)

Şekil 3.2: : Farklı zamanlarda 256 iterasyon çalıştırılan, NOT kapılı rastgele hafızalı hücresel otomat sonuçları (a), (b).



Şekil 3.3: Rastgele sayı üreteci genel blok diyagramı.

Şekil 3.3'te görülen sistemde FPGA içerisinde, Şekil 2.7'dekinden farklı olarak, BRAM yerine DDR2 SDRAM Denetleyicisi bulunmaktadır. DDR2 SDRAM, FPGA'nın dışında ayrı bir entegre olarak bulunan DDR2 SDRAM'e erişebilmeyi sağlayan bir devredir. DDR2 SDRAM ile ilgili ayrıntılı bilgi Alt Bölüm 3.2.1'de verilecektir.

Tasarlanan rastgele hafızalı hücresel otomat yapısından rastgele sayı dizisi üretmek için bir yol seçilmiştir. Buna göre, hücresel otomatın 64 hücresinden 2 tanesi seçilmiştir. Bu hücreler 10. ve 55. hücrelerdir. Seçilen bu iki hücrenin aynı zaman adımlarında aldıkları değerler EXOR işlemine sokulmuştur. Elde edilen değerler bir dizi şeklinde tutularak rastgele sayı dizisi üretilmiştir. Üretilen rastgele sayı dizisi,

gerçekten rastgele olup olmadığının anlaşılabilmesi için Bölüm 4'te anlatılacak olan testlere tabi tutulmuştur.

Rastgele sayı dizisi üretiminde, hücresel otomatın bütün hücrelerinin değerleri DDR2 SDRAM'e yazılmamıştır. Sadece seçilen iki hücrenin EXOR'lanmış değerleri DDR2 SDRAM'e yazılmıştır. Bu veriler 64 iterasyon boyunca, 64 bitlik bir yazmaca yazılır ve sonra bu 64 bitlik yazmaçtaki veriler DDR2 SDRAM'e gönderilir. Yani 64 iterasyonda bir DDR2 SDRAM'e yazma yapılmıştır.

3.2.1 DDR2 SDRAM

Oluşturulan sistemde DDR2 SDRAM'e ihtiyaç duyulduğunda, DDR2 SDRAM'e erişmek için ilk olarak Microblaze kullanmak düşünülmüştür. Microblaze, Xilinx tarafından geliştirilmiş soft bir mikroişlemcidir. Yani FPGA üzerinde hazır olarak bulunmayıp, istenildiği zaman FPGA üzerine yerleştirilebilen bir mikroişlemcidir. Microblaze, EDK program arayüzüyle FPGA üzerindeki sisteme eklenebilir. Microblaze sisteme eklenirken yazılımın üzerinde çalışacağı RAM olarak BRAM veya DDR2 SDRAM seçilebilir. Burada DDR2 SDRAM seçildiğinde, yazılımda tanımlanan bir dizi, direk DDR2 SDRAM üzerine kaydediliyor. Bu da DDR2 SDRAM'e erişimi bayağı kolaylaştırmış oluyor.

Öncelikle, DDR2 SDRAM'e erişebilmek için sisteme Microblaze eklenmiştir. Hücresel otomatta elde edilen değerler donanımdan yazılıma alınıp DDR2 SDRAM'e yazılmıştır. Fakat donanım ile yazılım arasındaki senkronizasyon sağlanamamış, arada veri kayıpları olmuştur. Çünkü, rastgele sayı dizisi üretirken sistemin sürekli çalışması gerekmektedir. Microblaze yazma ve okuma işlemlerini yaparken donanımın hızına yetişememektedir. Bu yüzden Microblaze kullanımının uygun olmadığı görülmüştür. Bunun üzerine DDR2 SDRAM'e donanım üzerinden erişilmeye çalışılmıştır.

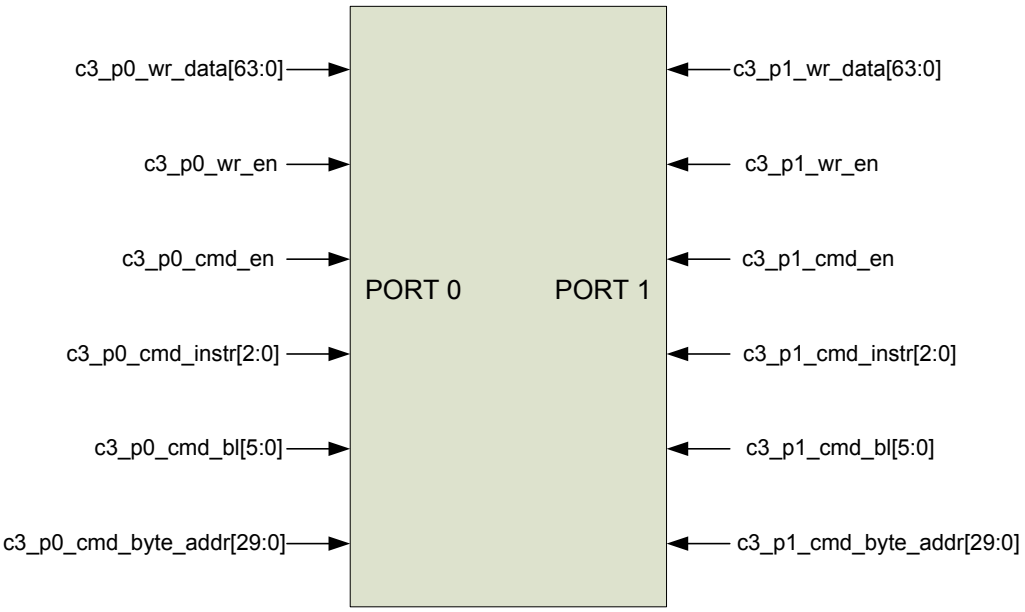
DDR2 SDRAM'e donanım üzerinden erişebilmek için FPGA üzerine DDR2 SDRAM denetleyici modülü yerleştirilmesi gerekmektedir. Şekil 3.3'te görüldüğü gibi FGPA dışında bulunan DDR2 SDRAM'e bu modül aracılığıyla erişilebilmektedir. Bu DDR2 SDRAM denetleyici, tasarımın yapıldığı ISE ortamında hazır olarak bulunan IP çekirdekleri arasından alınmıştır.

ISE programında “IP Core Generator” kısmında “MIG-Memory Interface Generator” sekmesi bulunur. Buradan FPGA’den tüm DDR SDRAM’lere erişmeyi sağlayan denetleyici modülü üretilir. Bu çalışmada kullanılan Atlys geliştirme kartı üzerinde DDR2 SDRAM bulunmaktadır. Bu yüzden burada erişilecek bellek olarak DDR2 SDRAM seçilmiş ve program ona uygun olan denetleyici modülü üretmiştir.

DDR2 SDRAM denetleyici modülü üretilirken DDR2 SDRAM iç çalışma frekansı 400 MHz seçilmiştir. Kullanıcı ile irtibat kurulan kısımda yani okuma yazma işlemlerinin yapıldığı kısımda sistemle senkronizasyon sağlanması için çalışma frekansı 100 MHz olarak seçilmiştir. Okuma ve yazmanın yapılacağı portlar olarak da iki adet 64 bitlik iki yönlü (bi-directional) port seçilmiştir. Her iki portta da okuma ve yazma yapılabilmektedir. Portların giriş ve çıkışlarında 64x64’lük FIFO bulunmaktadır. Yazma yapılacağı zaman kullanıcı tarafından veriler FIFO’ya yazılır oradan DDR2 SDRAM’e alınır. Benzer şekilde okuma yapılacağı zaman DDR2 SDRAM, verileri FIFO’ya yazar ve kullanıcı da verileri FIFO’dan okur. Yapılan tasarımda her iki port yazma için kullanılmıştır. Çünkü sistem devamlı çalışmaktadır ve devamlı veri üretmektedir. Arada veri kaybı yaşanmaması için her iki port da kullanılmıştır. Okuma işleminde ise yalnızca bir port kullanılmıştır.

DDR2 SDRAM denetleyici modülünün genel yapısında bir kalibrasyon kısmı bulunur. Bu kalibrasyon kısmı DDR2 SDRAM’in çalışmaya hazır hale gelmesini sağlar. Dolayısıyla tüm sistem kalibrasyonun yapıldığını belirten sinyalin gelmesiyle çalışmaya başlar. Komut ve data yolu kısımları bulunur. Data yolu kısmında yazma ve okuma işlemleri gerçekleşir. Komut yolu kısmında ise okuma ve yazma işlemlerinin kontrolü sağlanır [13][14].

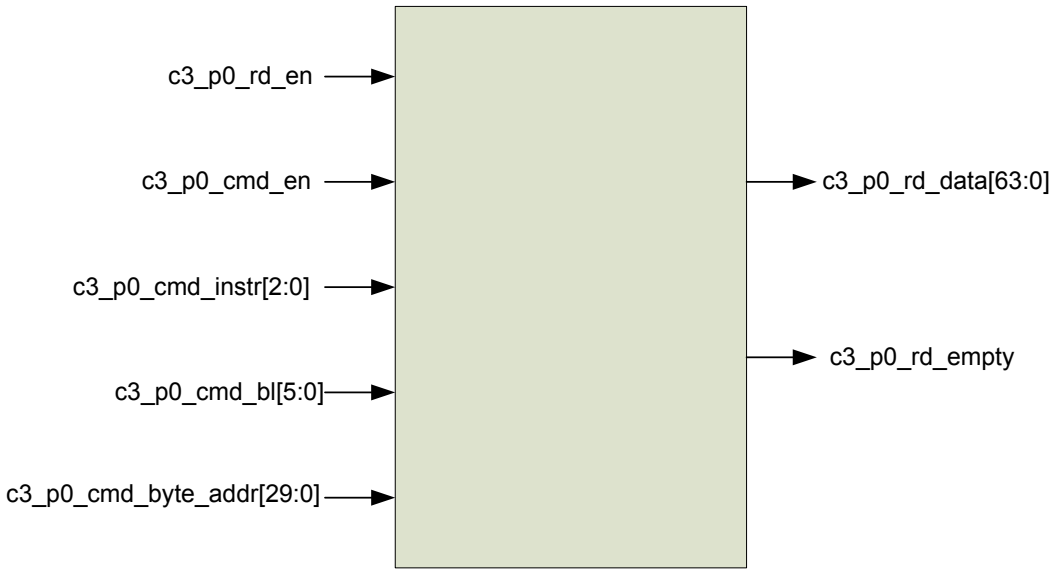
Sisteme eklenen DDR2 SDRAM denetleyici modülünün birçok giriş-çıkış işareti bulunmaktadır. Yazma ve okuma yapabilmek için, Verilog kodu yazarken bu işaretlerin hepsi kullanılmaz. Burada yazma ve okuma işlemleri, kullanılan işaretler gösterilerek anlatılacaktır. Şekil 3.4’te yazma işlemi için kullanılan işaretler bir blok şematiği halinde gösterilmiştir.



Şekil 3.4: DDR2 SDRAM'e veri yazabilmek için gerekli işaretler ve blok şematiği.

Daha önce de bahsedildiği gibi, yazma için iki adet port seçilmiştir. Bunlar port 0 ve port 1'dir. Önce, 64 iterasyon boyunca veriler 64 bitlik bir yazmaca yazılır. Sonra yazma izin girişi olan c3_p0_wr_en girişinin 1 verilmesiyle 64 bitlik bu veri c3_p0_wr_data yazmacına yazılır. c3_p0_cmd_instr, hangi işlemin yapılacağını belirtir. Bu işaret 0 değerindeyse yazma, 1 değerindeyse okuma yapılacağı anlaşılır. c3_p0_cmd_bl kaç tane 64 bitlik verinin FIFO'ya yazılacağını veya FIFO'dan okunacağını belirtir. c3_p0_cmd_byte_addr ise DDR2 SDRAM'e hangi adresten başlayarak yazma veya okuma yapılacağını belirtir. Verilerin c3_p0_wr_data yazmacına yazılmasıyla c3_p0_cmd_bl, c3_p0_cmd_instr ve c3_p0_cmd_byte_addr yazmaçlarına değerler girilir. Bu değerlerin aktif olabilmesi için c3_p0_cmd_en izin girişinin 1 girilmesi gerekir. c3_p0_cmd_en 1 girilir ve c3_p0_wr_en 0 girilir. Sonraki saat darbesinde c3_p0_cmd_en tekrar 0'a çekilir. Bu işlemlerin yapıldığı saat darbelerinde hücresel otomat çalışmaya devam eder. Burada veri kaybı olmaması için veriler bu sefer farklı bir 64 bitlik yazmaca yazılır ve aynı işlemler yapılır. Ve bu yazmaçtaki veriler de c3_p1_wr_data yazmacına yazılır. Böylece veriler bir port 0'dan, bir port 1'den DDR2 SDRAM'e yazılır. Yazılırken adresler birbirini takip edecek şekilde ayarlanır.

Şekil 3.5'da okuma işlemi için kullanılan işaretler bir blok şematiği halinde gösterilmiştir.



Şekil 3.5: DDR2 SDRAM'den veri okuyabilmek için gerekli işaretler ve blok şematiği.

DDR2 SDRAM'den veri okuyabilmek için önce c3_p0_cmd_bl, c3_p0_cmd_instr ve c3_p0_cmd_byte_addr yazmaçlarına değerler girilir. Sonra c3_p0_cmd_en yazmacına önce 1 sonra tekrar 0 yazılır. Böylece, c3_p0_cmd_bl, c3_p0_cmd_instr ve c3_p0_cmd_byte_addr yazmaçlarına girilen değerler aktif edilmiş olur. c3_p0_rd_en okuma izin girişine 1 verilir. Fakat okuma işlemi izin girişi geldikten hemen sonra başlamaz. Bir süre bekler. Sonra, okunan veri c3_p0_rd_data yazmacına yazılır. c3_p0_rd_empty çıkışı okuma işi bittiğinde 1 değerini alır. Böylece, okumanın ne zaman yapıldığı tespit edilmiş olur.

Çalışmada kullanılan Atlys geliştirme kartı üzerinde 128 MByte kapasiteli DDR2 SDRAM bulunmaktadır. Böylece rastgele sayı üreteci testleri için gerekli olan büyük miktardaki veriler rahatlıkla hafızaya kaydedilebilmektedir.

4. RASTGELE SAYI ÜRETECİ TESTLERİ

Rastgele sayı üreteçlerinin ürettiği rastgele sayı dizilerinin rastgeleliğini ölçmek için birtakım testler yapılır. Bu çalışmada, FIPS 140-1 test ailesinin testleri ve NIST testi uygulanmıştır.

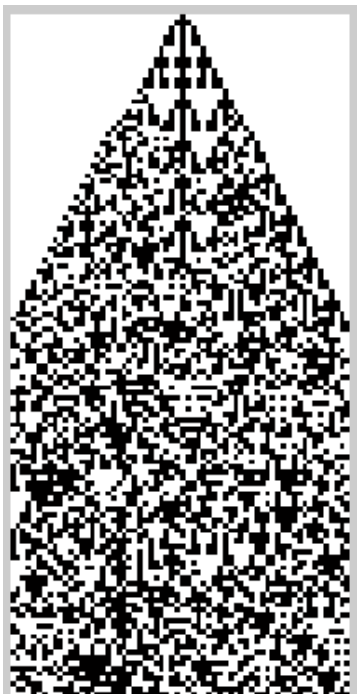
4.1 FIPS 140-1 Test Ailesinin Testlerinin Uygulanması

Tasarlanan rastgele sayı üreteci üzerinde , FIPS 140-1 test ailesinin içerdiği dört test uygulanmıştır. Bu testler; frekans(monobit) testi, runs testi, long run testi ve poker testidir [20].

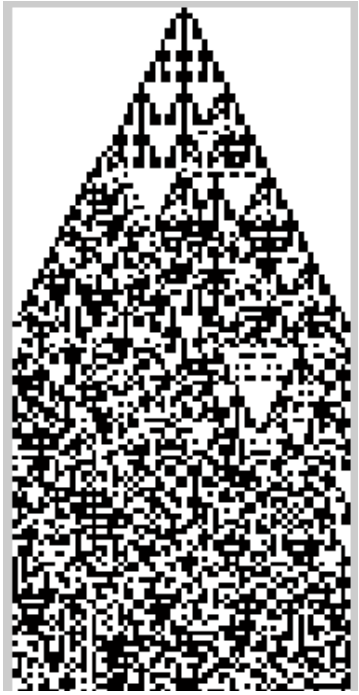
Bu testler, FIPS 140-1 test kurallarına göre rastgele sayı üretecinden alınan 20000 veri üzerinde yapılır. Tasarlanan rastgele sayı üretecinde, hücresel otomatın iki hücresi seçilip, bu hücrelerin değerleri EXOR işlemine sokulur. Bu işlem her iterasyon adımında yapılır. Her iterasyon adımında elde edilen bu değerler rastgele sayı dizisini oluşturmuş olur. 20000 tane rastgele değer elde edilebilmesi için hücresel otomat 20000 iterasyon çalıştırılmalıdır. Ancak Şekil 4.1 ve Şekil 4.2'de görüldüğü gibi, 64 hücreden oluşan hücresel otomatın başlangıçta sadece 33. hücresi 1 değerini, diğer hücreler 0 değerini almıştır. İlerleyen zaman adımlarında 0 ve 1 değerleri yavaş yavaş bütün hücrelere yayılmaktadır. 64. iterasyon adımına kadar hücresel otomatın zamanla değişimini gösteren şekil üçgen şeklindedir. 64. adımdan sonra 0 ve 1 değerleri tüm hücrelere dağılmaktadır. Bu yüzden seçilen iki hücrenin değerleri 128. adımdan itibaren alınmaya başlanmıştır. Böylece hücresel otomat 20128 adım çalıştırılmıştır.

20128 adım çalıştırılan hücresel otomatın bütün değerleri DDR SDRAM'e kaydedilir. Daha sonra bu veriler DDR SDRAM'den okunarak seri haberleşme modülü ile bilgisayara gönderilir. Bilgisayarda MATLAB ortamında yazılan kodlarla testler çalıştırılır. MATLAB ortamında öncelikle hücresel otomatın iki hücresi parametrik olarak seçilir. Daha sonra 129. adımdan 20128. adıma kadar 20000 iterasyon boyunca bu hücrelerin aldıkları değerler EXOR işlemine sokulur. Elde edilen 20000 veri rastgele sayı dizisini oluşturur. Bu 20000 tane ikili sayıdan oluşan

rastgele sayı dizisi, sırayla frekans (monobit), runs, long run ve poker testlerine sokulur. Bu testler de yine MATLAB ortamında yazılan kodlarla yapılmıştır. Yapılan bu testler ayrı başlıklar halinde incelenmiştir.



Şekil 4.1: LUT'lu rastgele hafızalı hücresel otomatın 128 iterasyon çalıştırılmış hali.



Şekil 4.2: NOT kapılı rastgele hafızalı hücresel otomatın 128 iterasyon çalıştırılmış hali.

4.1.1 Frekans (Monobit) testi

Bu test ile rastgele sayı dizisindeki 0 ve 1 değerlerinin sayısına bakılarak, rastgele sayı dizisinin rastgeleliği hakkında yorum yapılır. Gerçek rastgele sayı üreteçlerinde amaç 0 ve 1 değerlerinin sayılarının yaklaşık olarak eşit olmasıdır. Eğer rastgele sayı üretecinin ürettiği rastgele sayı dizisinde 0 ve 1 değerleri birbirine yakın olursa rastgele sayı üreteci frekans testinden geçmiş sayılır. 0 ve 1 değerlerinin sayılarının birbirine yakın olmasının ölçüsü FIPS 140-1 test ailesine göre rastgele sayı dizisindeki 1'lerin sayısının (X), $9654 < X < 10346$ aralığında olmasıdır. Böylece 0'ların sayısı da (Y), $9654 < Y < 10346$ aralığında olmalıdır.

MATLAB'de, 20000 verilik rastgele sayı dizisindeki değerlere tek tek bakılıp 0 ve 1 değerlerinin sayılarının tespit edildiği bir kod yazılmıştır. Böylece frekans testi rastgele sayı üreteci her çalıştırıldığında uygulanabilmektedir.

Çizelge 4.1'de ve Çizelge 4.2'de farklı zamanlarda çalıştırılan LUT'larla tasarlanan rastgele sayı üretecinin ve NOT kapılarıyla tasarlanan rastgele sayı üretecinin frekans testi sonuçları verilmektedir. Görüldüğü gibi rastgele sayı üreteçleri frekans testinden başarıyla geçmektedir.

4.1.2 Runs testi

Bu testte, üretilen rastgele sayı dizisinde kesintiye uğramadan gelen bit dizilerinin sayılarına bakılır. Bu şekilde, üretilen rastgele sayı dizisinin 0 ve 1 arasındaki osilasyonunun ne kadar hızlı ve yavaş olduğu tespit edilmektedir.

Rastgele sayı üretecinin bu testten geçebilmesi için, FIPS 140-1 test ailesine göre Çizelge 4.3'te verilen koşullar sağlanmalıdır. Test, 0 ve 1 değerleri için ayrı ayrı yapılmalıdır. Eğer Çizelge 4.3'te verilen bit uzunlukları 0 ve 1 için ayrı ayrı verilen değer aralıklarında olursa runs testinden başarıyla geçilmiş olur.

MATLAB'de 20000 verilik rastgele sayı dizisindeki 0 ve 1 değerlerinin Çizelge 4.3'te verilen bit uzunluklarının sayılarını tespit eden bir kod yazılmıştır. Böylece Runs testi, rastgele sayı üreteci her çalıştırıldığında uygulanabilmektedir.

Çizelge 4.1: LUT'lu tasarımın Frekans testi sonuçları.

Test No	1'lerin Sayısı	0'ların Sayısı	Başarı Durumu
1	10015	9985	Başarılı
2	10025	9975	Başarılı
3	9998	10002	Başarılı
4	9889	10111	Başarılı
5	10022	9978	Başarılı
6	10047	9953	Başarılı
7	9976	10024	Başarılı
8	9872	10128	Başarılı
9	9923	10077	Başarılı
10	10001	9999	Başarılı

Çizelge 4.2: NOT kapılı tasarımın Frekans testi sonuçları.

Test No	1'lerin Sayısı	0'ların Sayısı	Başarı Durumu
1	10089	9911	Başarılı
2	10039	9961	Başarılı
3	10023	9977	Başarılı
4	9991	10009	Başarılı
5	9944	10056	Başarılı
6	10023	9977	Başarılı
7	9981	10019	Başarılı
8	9992	10008	Başarılı
9	10073	9927	Başarılı
10	9926	10074	Başarılı

Çizelge 4.3: Runs testi koşulları.

Dizi Uzunluğu (Length of Run)	Gerekli Aralık (Required Interval)
1	2267 – 2733
2	1079-1421
3	502-748
4	223-402
5	90-223
6+	90-223

Çizelge 4.4 ve Çizelge 4.5'te farklı zamanlarda çalıştırılan LUT'larla tasarlanan rastgele sayı üretecinin ve NOT kapılarıyla tasarlanan rastgele sayı üretecinin frekans testi sonuçları verilmektedir. Görüldüğü gibi rastgele sayı üreteçleri frekans testinden başarıyla geçmektedir. Çizelge 4.4 ve Çizelge 4.5'te sadece 0 değerinden alınan sonuçlar verilmiştir. 1 değeri de yaklaşık sonuçlar vermiştir. Görüldüğü gibi rastgele sayı üreteçleri Runs testinden başarıyla geçmektedir.

Çizelge 4.4: LUT'lu tasarımın Runs testi sonuçları.

Test	Dizi Uzunlukları Sayısı						B. Durumu
	1	2	3	4	5	6+	
1	2490	1261	624	307	171	146	Başarılı
2	2508	1248	615	316	157	155	Başarılı
3	2560	1279	586	303	141	176	Başarılı
4	2500	1280	654	321	134	157	Başarılı
5	2469	1246	660	309	145	153	Başarılı
6	2474	1239	613	321	156	157	Başarılı
7	2509	1259	604	311	156	165	Başarılı
8	2423	1255	644	313	162	173	Başarılı
9	2533	1236	632	308	168	159	Başarılı
10	2518	1264	607	307	167	157	Başarılı

4.1.3 Long Run testi

Bir rastgele sayı üretecinin gerçekten rastgele olabilmesi için ürettiği rastgele sayı dizisinde uzun 0 veya 1 bit dizileri içermemesi gerekmektedir. Bu testte üretilen rastgele sayı dizisinde uzun 0 veya 1 bit dizilerinin olup olmadığı kontrol

edilmektedir. FIPS 140-1 test ailesine göre bir rastgele sayı üretecinin ürettiği rastgele sayı dizisinde en fazla 34 bit dizisi bulunabilir. Rastgele sayı dizisi, içerisinde 34'ten daha fazla bit dizisi içerirse, rastgele sayı üreteci Long Run testini geçemez.

MATLAB'de rastgele sayı üretecinin ürettiği rastgele sayı dizisinde 34'ten fazla bit dizisinin bulunup bulunmadığını tespit etmek için bir kod yazılmıştır. Böylece Long Run testi, rastgele sayı üreteci her çalıştırıldığında uygulanabilmektedir.

Çizelge 4.5: NOT kapılı tasarımın Runs testi sonuçları.

Test	Dizi Uzunlukları Sayısı						B. Durumu
	1	2	3	4	5	6+	
1	2556	1254	658	266	130	167	Başarılı
2	2500	1227	601	311	151	170	Başarılı
3	2521	1211	631	300	165	161	Başarılı
4	2559	1197	621	318	156	163	Başarılı
5	2450	1226	665	313	164	155	Başarılı
6	2491	1237	640	285	162	164	Başarılı
7	2497	1237	655	290	148	165	Başarılı
8	2506	1260	593	311	164	162	Başarılı
9	2517	1262	619	293	158	154	Başarılı
10	2402	1199	624	346	168	163	Başarılı

4.1.4 Poker testi

Bu testte, rastgele sayı dizisindeki 20000 bitlik veri dörder bitlik sayılara ayrılmaktadır. Böylece, dörder bitlik 5000 adet sayı elde edilmiş olur. 4 bit ile 16 farklı sayı ifade edilebilir. Rastgele sayı üretecinin rastgeleliğinin güvenilir olabilmesi için, elde edilen 5000 tane 4 bitlik sayı içerisindeki 16 farklı sayının her birinden yaklaşık eşit sayıda olması istenmektedir. Bu amaçla (4.1)'de X ile ifade edilen poker değeri hesaplanmaktadır. Buradaki i, 0'dan 15'e kadar olan dört bitlik sayıları ifade eder. f(i) ise i sayısından kaç tane olduğunu göstermektedir.

$$\mathbf{X = \frac{16}{5000} * (\sum_{i=0}^{15} [f(i)]^2) - 5000} \qquad \mathbf{(4.1)}$$

FIPS 140-1 test ailesinin kurallarına göre rastgele sayı üretecinin bu testten geçebilmesi için poker değerinin $1.03 < X < 57.4$ olması gerekmektedir.

MATLAB'de bu testin yapılabilmesi için gerekli olan kodlar yazılmıştır. Böylece, rastgele sayı üreteci her çalıştırıldığında bu test uygulanmaktadır.

4.2 NIST Testi

NIST, ABD Ulusal Standartlar ve Teknoloji Enstitüsü tarafından belirlenen, rastgele sayı üreteçlerinin güvenilirliğini ölçmek için kullanılan bir test ailesidir. NIST, kendi içerisinde 15 adet testten oluşur. Bu testler ve testler hakkındaki kısa bilgiler şu şekildedir:

Frekans testi, dizi içindeki 1 ve 0 oranlarını tespit etmektedir. İstenilen, belirli bir hata ile maksimum yaklaşıklık sağlamaktır. Farkın %1'den küçük olması istenmektedir.

Bir Blok için Frekans testi, M bit uzunluklu bir dizi parçasındaki 1'lerin oranlarına bakar. M bitlik bir bloktaki 1 sayısı yaklaşık M/2 olmalıdır. Belirli aralıklar için bakıldığından blok için frekans testi denilmektedir.

Runs testinin, odaklandığı nokta ardışık (kesilmeden) gelen bit değerlerinin uzunluğudur. FIPS testi ile benzerlik göstermektedir.

Bir Blok için Longest Run testi, FIPS 140-1 Long Run testinde anlatılan teste benzerlik göstermektedir. Bu testte ek olarak, M bitlik bir blok üzerinde test yapılmıştır. M bit için 1 sayılarının uzunluğuna bakılarak karesel bir ortalama alınır. Değer belli bir aralıkta çıkmalıdır.

İkili Matris Dizi testi, verilen bit dizisini kare matrislere ayırır. Her bir kare matris için lineer bağımsızlık koşulu aranır.

Ayrık Fourier Dönüşümü testinin amacı bit dizisinin Fourier Dönüşümü'nde en yüksek değeri verdiği frekansı bulmaktır. Frekans spektrumunda standart sapmasına göre test eder.

Örtüşmesiz Şablon Eşleme testinin amacı önceden belirlenmiş hedef dizilerinin sayısını test etmektir. M-bit ayrılmış bit dizisinin n-bit ayrılmış kısımlarında örtüşme olup olmadığına, varsa maksimum kaç tane olduğuna bakar.

Örtüşmeli Şablon Eşleme testinin amacı önceden belirlenmiş hedef dizilerinin sayısını test etmektir. Örtüşmesiz Şablon Eşleme testinden farkı hesaplanan parametrelerin farklı artırımıdır.

Maurer'in Evrensel İstatistik (Universal Statistical) testinin amacı bit dizisinin sıkıştırılabilir bir dizi olup olmadığını kontrol etmektir. Dizi eğer sıkıştırılabilir ise gerçek rastgele bir dizi değildir diye düşünülebilir.

Doğrusal Karmaşıklık testinin amacı doğrusal-geri-beslemeli-ötelemeli-kaydedicinin uzunluğunu kontrol etmektir. Böylece dizinin yeterince karmaşık olup olmadığını ölçer. Rastgele diziler daha uzun bir doğrusal-geri-beslemeli-ötelemeli-kaydedici (LFSR) tarafından karakterize edilir. LFSR'nin uzunluğu çok kısa ise rastgelelik yoktur diye düşünülmektedir.

Seri testin amacı m-bit ayrılan dizi parçacıklarında ne sıklıkta örtüşme olduğunu bulmaktır. m-bit ayrılan parçalar için dizinin içinde her bir olası parça durumundan ne kadar ve ne sıklıkta geldiğini kontrol eder. Buna göre matematiksel bir işleme tabi tutarak sonuç bildirir.

Yaklaşık Entropi testinde m-bit ayrılan dizi parçacıklarındaki örtüşme incelenir. Fakat bu testte birbiri ardına gelen parçacıkların rastgeleliği kontrol edilir.

Kümülatif Toplamlar (Cusum) testinde ayrılmış dizi parçacıklarında 0 etrafında sapma miktarı ölçülür. Sapma miktarı çok ise rastgelelik durumu yoktur diye

düşünülür. Hesaplamalar kümülatif toplam şeklinde ve değerlere sırasıyla 1,-1 verilerek gerçekleştirilir.

Rastgele Sapma testinin amacı, Cusum testinde kümülatif toplam olarak hesaplanan sapmanın, genel dizideki sıklığını bulmaktır. Bu test aslında 8 adet test dizisinden meydana gelmektedir.

Rastgele Sapma Varyansı testinin amacı kümülatif toplamda ziyaret edilen yerlerin beklenen olup olmadığını test etmektir. Bu test 18 adet test dizisinden meydana gelmektedir [19].

Şekil 4.3'te herhangi bir NIST testinin sonuç çıktısı görülmektedir. Burada 15 testin hepsi bulunmamaktadır. NIST testi uygulanırken içerdiği 15 testin hepsi yapılabildiği gibi sadece seçilenler de yapılabilir. Test şu şekilde yapılır: Örneğin, 2000000 bitlik bir sayı dizisi test edilecek olsun. Önce bu bit dizisi alt dizilere ayrılır. Burada 200000'er bitlik 10 tane alt diziye ayrılmıştır. Böylece bu dizilerin hepsi ayrı ayrı test edilir. 10 diziden kaç tanesi testten geçtiyse bu oran sonuç çıktısı ekranına yazılır. Testler p-değeri adı verilen ve rastgeleliğin derecesini belirleyen bir istatiksel değer hesaplar. Her bir p-değeri rastgele sayı üretecinin mükemmelliği hakkında bilgi verir. Testin başarılı olabilmesi için p-değerinin 0.01'den büyük olması gerekmektedir. C1-C10 arasında yazan değerler p-değerinin frekansını göstermektedir. Sonuç çıktısı sayfasının altında bir testin başarılı olması için gereken başarı oranı yazmaktadır. Burada testin başarılı olabilmesi için 8/10 başarı oranı olması gerektiği yazmaktadır. Başarısız olan testler * işareti ile gösterilir.

Bu çalışmada da tasarlanan rastgele sayı üreteçleri üzerinde NIST testi uygulanmıştır. NIST testine sokulmak üzere, tasarlanan sistemden 10000000 bitlik veri dizisi alınmıştır. Bunlar 400000 bitlik 25 alt diziye ayrılmış ve NIST testi bu diziler üzerinde ayrı ayrı uygulanmıştır.

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <data.txt>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
  2   0   0   1   0   2   2   2   1   0  0.534146      9/10      Frequency
  8   1   1   0   0   0   0   0   0   0  0.000000 *    2/10   *  BlockFrequency
  2   0   0   0   1   0   1   3   1   2  0.350485      9/10      CumulativeSums
  2   0   0   1   1   2   1   1   0   2  0.739918      9/10      CumulativeSums
  2   1   1   1   0   0   1   0   3   1  0.534146     10/10      Runs
  0   0   3   1   2   0   3   1   0   0  0.122325     10/10      LongestRun
  2   2   1   0   0   0   1   1   3   0  0.350485      9/10      Rank
  2   1   3   1   1   1   0   0   0   1  0.534146      9/10      FFT
  2   1   1   0   0   1   2   3   0   0  0.350485      9/10      OverlappingTemplate
  0   0   0   0   0   0   0  10   0   0  0.000000 *   10/10      Universal
 10   0   0   0   0   0   0   0   0   0  0.000000 *    0/10   *  ApproximateEntropy
  9   1   0   0   0   0   0   0   0   0  0.000000 *    5/10   *  Serial
  2   0   1   2   1   0   0   2   1   1  0.739918      9/10      Serial
  1   0   1   0   1   1   0   3   3   0  0.213309     10/10      LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 8 for a
sample size = 10 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Şekil 4.3 : NIST testinin genel sonuç çıktısı[17]

Öncelikle LUT'larla tasarlanan rastgele sayı üreteci NIST testine sokulmuş ve sonuçlar alınmıştır. Sonuçlar Çizelge 4.6'da görülmektedir. Görüldüğü gibi LUT'larla tasarlanan rastgele sayı üreteci bütün testlerden başarıyla geçmiştir. Bu testlerden Örtüşmesiz Şablon Eşleme (Non-Overlapping Template) testi 148 ayrı testten oluşmaktadır. Bu testlerin hepsinden geçilmiştir. Rastgele Sapma (Random Excursions) testi 8, Rastgele Sapma Varyansı (Random Excursions Variant) testi 18 ayrı testten oluşmaktadır. Bu testlerin de hepsinden geçilmiştir. Çizelgeye bu testlerden herhangi birinin sonuçları konulmuştur.

İkinci olarak NOT kapılarıyla tasarlanan rastgele sayı üreteci NIST testine sokulmuş ve sonuçlar alınmıştır. Sonuçlar Çizelge 4.7'de görülmektedir. Görüldüğü gibi NOT kapılarıyla tasarlanan rastgele sayı üreteci bütün testlerden başarıyla geçmiştir. Sadece 148 ayrı testten oluşan Örtüşmesiz Şablon Eşleme (Non-Overlapping Template) testinin sadece bir testi başarısız olmuştur. Onun da başarı oranı 22/25'tir.

Çizelge 4.6: LUT'larla tasarlanan rastgele sayı üretecinin NIST sonucu.

Test	**p-değeri**	**Başarı Oranı**	**Sonuç**
Frekans (Frequency)	0.311542	25/25	Geçti
Blok Frekans (Block Frequency)	0.311542	25/25	Geçti
Kümülatif Toplamlar (Cumulative Sums)	0.585209	25/25	Geçti
Kümülatif Toplamlar (Cumulative Sums)	0.788728	25/25	Geçti
Runs	0.788728	25/25	Geçti
Longest Run	0.788728	25/25	Geçti
İkili Matris Dizi (Rank)	0.242986	25/25	Geçti
FFT	0.078086	25/25	Geçti
Örtüşmesiz Şablon Eşleme (Non-Overlapping Template)	0.242986	25/25	Geçti
Örtüşmeli Şablon Eşleme (Overlapping Template)	0.186566	25/25	Geçti
Evrensel (Universal)	0.010606	25/25	Geçti
Yaklaşık Entropi (Approximate Entropy)	0.585209	25/25	Geçti
Rastgele Sapma (Random Excursions)	0.066882	11/12	Geçti
Rastgele Sapma Varyansı (Random Excursions Variant)	0.122325	12/12	Geçti
Seri (Serial)	0.484646	24/25	Geçti
Seri (Serial)	0.078086	25/25	Geçti
Doğrusal Karmaşıklık (LinearComplexity)	0.078086	25/25	Geçti

Çizelge 4.7: NOT kapılarıyla tasarlanan rastgele sayı üretecinin NIST sonucu.

Test	**p-değeri**	**Başarı Oranı**	**Sonuç**
Frekans (Frequency)	0.015065	25/25	Geçti
Blok Frekans (Block Frequency)	0.242986	25/25	Geçti
Kümülatif Toplamlar (Cumulative Sums)	0.057146	25/25	Geçti
Kümülatif Toplamlar (Cumulative Sums)	0.141256	25/25	Geçti
Runs	0.010606	25/25	Geçti
Longest Run	0.585209	24/25	Geçti
İkili Matris Dizi (Rank)	0.392456	25/25	Geçti
FFT	0.105618	25/25	Geçti
Örtüşmesiz Şablon Eşleme (Non-Overlapping Template)	0.000073	24/25	Geçti*
Örtüşmeli Şablon Eşleme (Overlapping Template)	0.392456	25/25	Geçti
Evrensel (Universal)	0.242986	25/25	Geçti
Yaklaşık Entropi (Approximate Entropy)	0.242986	25/25	Geçti
Rastgele Sapma (Random Excursions)	0.090936	11/11	Geçti
Rastgele Sapma Varyansı (Random Excursions Variant)	0.012650	11/11	Geçti
Seri (Serial)	0.689019	25/25	Geçti
Seri (Serial)	0.484646	25/25	Geçti
Doğrusal Karmaşıklık (LinearComplexity)	0.186566	25/25	Geçti

SONUÇ VE ÖNERİLER

Tez çalışmasında, literatürdeki farklı hücresel otomat yapıları incelenmiştir. Önce, temel hücresel otomat yapısı ve onun matematiksel modeli incelenmiştir. Daha sonra hafızalı hücresel otomat ve rastgele hafızalı hücresel otomat yapıları incelenmiştir. Bu hücresel otomat yapılarının FPGA üzerinde sayısal olarak gerçeklenmesiyle elde edilen sonuçlar gözlenmiştir. Buna göre, hafızalı hücresel otomatta zamanla düzgün bir yayılım olurken, rastgele hafızalı hücresel otomatta rastgele bir yayılım olduğu görülmüştür.

Hücresel otomat yapısının rastgele sayı üreteci olarak kullanılması düşünülmüştür. Buna göre, Göncü'nün önerdiği rastgele hafızalı hücresel otomat yapısından esinlenerek yeni bir rastgele hafızalı hücresel otomat yapısı önerilmiştir. Önerilen bu yapının matematiksel modeli verilmiş ve FPGA üzerinde sayısal gerçeklemesi iki farklı tasarımla yapılmıştır. FPGA üzerinde çalıştırılan bu iki tasarımdan alınan sonuçlar daha önceki sonuçlarla karşılaştırıldığında rastgeleliğin daha iyi olduğu görülmüştür. Bunun üzerine bu sistem kullanılarak rastgele sayı üreteci tasarımı yapılmıştır.

Rastgele sayı üreteçlerinin güvenilirliğini ölçen testler vardır. Bu çalışmada FIPS 140-1 ailesinin testleri ve NIST testi uygulanmıştır. Tasarlanan rastgele sayı üreteci, her iki tasarımda da FIPS 140-1 ailesinin testlerini başarıyla geçmiştir. NIST testi, kendi içerisinde birçok testten meydana gelmektedir. Tasarlanan rastgele sayı üreteci, tasarımlardan birinde NIST testinden tamamen başarıyla geçebilmiştir. Diğer tasarımda ise NIST'in testlerinden sadece bir tanesinin 148 alt testinden birinden geçememiştir. NIST'in diğer tüm testlerinden başarıyla geçmiştir.

Sonuç olarak, bu tez çalışmasında FPGA üzerinde hafızalı hücresel otomat yapısı ile rastgele sayı üreteci tasarımı yapılmıştır. Tasarlanan rastgele sayı üreteci testlerden geçirilerek büyük oranda güvenilir olduğu gösterilmiştir. Bir sonraki çalışmalarda, rastgele sayı üretimi yapılırken tasarlanan yapı üzerinde sadeleştirmeler yapılarak

daha az alan kaplayıp daha iyi sonuç verecek şekilde yeni tasarımlar yapılabilir. Tasarımda yer alan gecikme bloklarının çıkışlarındaki jitter etkisi ile 0 ve 1 gelme miktarları ölçülüp bunun rastgeleliğe etkisi gözlemlenebilir. Aynı anda farklı hücreler kullanılarak birden fazla farklı rastgele sayı üreteci üretilip bunlar arasında ilişki olup olmadığında bakılabilir. Sıcaklık gibi çevresel faktörlerin ve farklı cihaz kullanımının rastgelelik üzerindeki etkisi ölçülebilir.

KAYNAKLAR

[1] **Wolfram, S.** (1983).Statistical mechanics of cellular automata, *Rev. Mod. Phys., **55***, 601-644.

[2] **Von Neumann, J.** (1963). The general and logical theory of automata, *Collected Works, **5,*** 288.

[3] **Von Neumann, J.** (1966). *Theory of Self-Reproducing Automata,* University of Illinois Press, Champaign, IL, USA.

[4] **Ulam, S.M.** (1972). Some ideas and prospects in biomathematics, *Annual Review of Biophysics and Bioengineering,* 277-292.

[5] **Wolfram, S.** (2002). *A New Kind of Science,* Wolfram Media.

[6] **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,.

[7] **Alonso-Sanz, R. Ve Martin, M.** (2003). Elementary cellular automata with memory, *Complex Systems,* http://www.complex-systems.com/pdf/14-2-1.pdf,.

[8] **Alonso-Sanz, R. Ve Martin, M.** (2005). One-dimensional Cellular Automata with Memory in Cells of the Most Frequent Recent Value, ***15,*** 203-236.

[9] **Göncü, E.,** "Hafızalı Hücresel Otomat Sayısal Tasarımı", *İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi,* İstanbul, Haziran 2013.

[10] **Santoro, R.; Roy, S.; Sentieys, O.,** "Search for Optimal Five-Neighbor FPGA-Based Cellular Automata Random Number Generators," *Signals, Systems and Electronics, 2007. ISSSE '07. International Symposium on* , vol., no., pp.343,346, July 30 2007-Aug. 2 2007.

[11] **Tavas V.,** 2010. Tümleştirmeye Uygun Rastgele Sayı Üreteçleri, *Doktora Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[12] **Yalçın M.E., Suykens Johan A. K., Vandewalle J.,** 2005. Cellular neural networks, multi-scroll chaos and synchronization, World Scientific Series on Nonlinear Science, Series A, vol. 50, Singapore.

[13] **UG388,** 2010, Spartan-6 FPGA Memory Controller User Guide.

[14] **UG416,** 2011, Spartan-6 FPGA Memory Interface Solutions User Guide.

[15] **Shackleford, B.; Tanaka, M.; Carter, R.J.; Snider, G.,** "High-performance cellular automata random number generators for embedded probabilistic computing systems," *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on* , vol., no., pp.191,200, 2002.

[16] **Abdul Hameed, M.U.; Eldin, A.M.B.,** "A cellular automata random number generator for cryptographic applications," *Computer Engineering & Systems, 2007. ICCES '07. International Conference on* , vol., no., pp.101,105, 27-29 Nov. 2007

[17] **Ustaoğlu, B.,** "Gerçek Rastgele Sayı Üreteci Tasarımı, Testleri ve Gerçeklenmesi", *İstanbul Teknik Üniversitesi Elektrik Elektronik Fakültesi Lisans Tezi,* İstanbul, Haziran 2013.

[18] **Yeniçeri, R.; Yalçın, M.E.,** "True random bit generation with time-delay sampled-data feedback system," *Electronics Letters* , vol.49, no.8, pp.543,545, April 11 2013, doi: 10.1049/el.2012.3448.

[19] **National Institute of Standarts and Technology**, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST 800-22, April 2010.

[20] **Kalalı, E.,** "Rastgele Sayı Üreteçlerinin FPGA Tabanlı Bir Sistem Yardımıyla Gerçek Zamanlı Test Edilmesi", *İstanbul Teknik Üniversitesi Elektrik Elektronik Fakültesi Lisans Tezi,* İstanbul, Mayıs 2011.

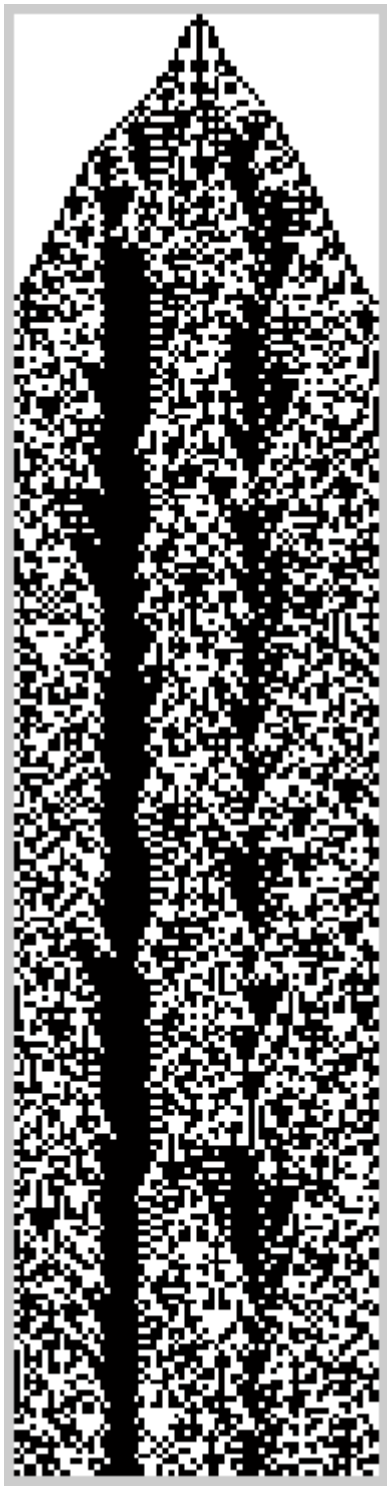
EKLER

EK A: LUT'lu tasarımda seçilen farklı LUT sayılarındaki alınan sonuçlardan örnekler.

EK B: NOT kapılı tasarımda seçilen farklı NOT kapısı sayılarındaki alınan sonuçlardan örnekler.

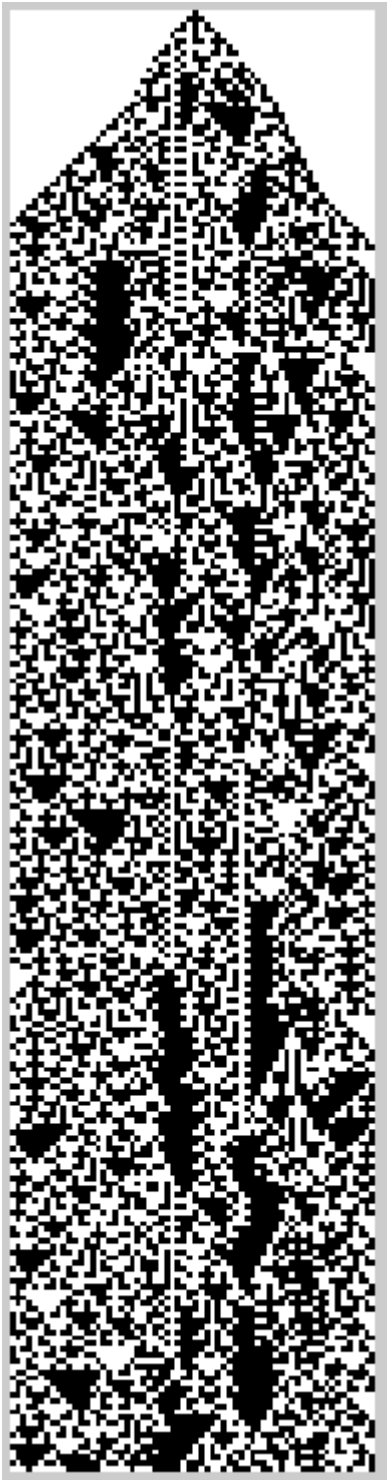
EK A

Şekil A.1: LUT sayısı 18 seçildiği zaman elde edilen sonuç.

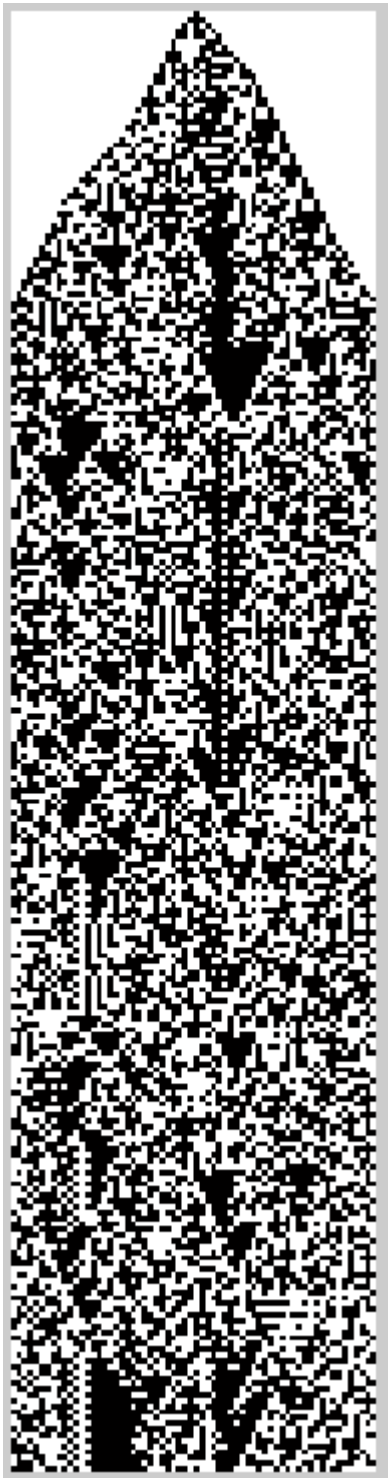


Şekil A.2: LUT sayısı 20 seçildiği zaman elde edilen sonuç.

EK B



Şekil B.1: NOT kapısı sayısı 18 seçildiği zaman elde edilen sonuç.



Şekil B.2 NOT kapısı sayısı 20 seçildiği zaman elde edilen sonuç.

ÖZGEÇMİŞ

Ad-Soyad **:** Abdulkadir KOÇDOĞAN

Doğum Tarihi ve Yeri **:** Kütahya, 1988

E-posta **:** akkocdogan88@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2012, İstanbul Teknik Üniversitesi, Elektrik-Elektronik Fakültesi, Elektronik Mühendisliği Bölümü

YAYIN :

- Koçdoğan A., Yeniçeri R., Yalçın M. E., "FPGA Üzerinde MicroBlaze Tabanlı Video İşlemci Tasarımı", *3. Gömülü Sistemler ve Uygulamaları Sempozyumu (GÖMSİS 2012) Bildiri Kitabı*, sayfa 99-100, İstanbul, 29-30 Kasım 2012.