T.C.
MUNZUR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

RASTGELE SAYI ÜRETEÇLERİNİN DONANIMSAL OLARAK GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

DİLAN BAKIR

Anabilim Dalı: Elektrik-Elektronik Mühendisliği

DANIŞMAN
Prof. Dr. Muzaffer AŞKIN

TUNCELİ – 2017

T.C.
MUNZUR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

RASTGELE SAYI ÜRETEÇLERİNİN DONANIMSAL OLARAK GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

DİLAN BAKIR
151103103

Anabilim Dalı: Elektrik-Elektronik Mühendisliği

Danışman
Prof. Dr. Muzaffer AŞKIN

TUNCELİ-2017

T.C.
MUNZUR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

RASTGELE SAYI ÜRETEÇLERİNİN DONANIMSAL OLARAK GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

DİLAN BAKIR

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez 14/07/2017 tarihinde aşağıdaki jüri üyeleri tarafından **oybirliği** ile kabul edilmiştir.

İmza:.....	**İmza:.....**	**İmza:.....**
Prof. Dr. Muzaffer AŞKIN (Munzur Üniversitesi)	Yrd. Doç. Dr. Bilgin ZENGİN (Munzur Üniversitesi)	Yrd. Doç. Dr. Mustafa SALTI (Mersin Üniversitesi)
DANIŞMAN	**ÜYE**	**ÜYE**

Bu tez, Enstitümüz Elektrik ve Elektronik Mühendisliği Anabilim Dalı'nda hazırlanmıştır.

Doç. Dr. Numan YILDIRIM
Enstitü Müdürü

NOT: Bu tezde kullanılan özgün ve başka kaynaktan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak gösterilmeden kullanımı, 5846 sayılı "Fikir ve Sanat Eserleri Kanunu"'ndaki hükümlere tabidir.

ÖZET

Bu tez çalışmasında, simetrik şifreleme algoritmalarından olan Trivium algoritmasının Alanda Programlanabilir Kapı Dizileri (FPGA - Field Programmable Gate Array) kullanılarak gerçekleştirilmesi incelenmiştir. Rastgele sayı üretimi ve şifreleme kavramları hakkında kısa bilgiler verildikten sonra öncelikle akış şifrelerin tasarım yapıları irdelenmiştir. Trivium algoritmasının gerçekleştirilmesi sürecinde VHDL (Very High Speed Integrated Curcuit Hardware Description Language) programlama dilinde simülasyon sonuçları elde edilerek 80 bitlik anahtar girdisi sonucu 128 bitlik rastgele sayı üretilmiştir. Daha sonra test kriterleri incelenerek üretilen 128 bitlik rastgele değerler kriterlere tabi tutulmuştur. Son aşamada ise üretilen bu rastgele sayı değerleri FPGA' da donanımsal olarak gerçekleştirilmiş ve elde edilen test sonuçları tartışılmıştır.

Anahtar Kelimeler: Rastgele Sayı Üreteçleri, Alanda Programlanabilir Kapı Dizileri, Şifreleme Algoritmaları.

ABSTRACT

Hardware Implementation of Random Number Generators

In this thesis, one of the symmetric cryptographic algorithms known as the Trivium algorithm is implemented by making use of its Field Programmable Gate Array (FPGA). After introducing random number generation and cryptographic concepts briefly, we focus on the design structures of stream ciphers. After using the VHDL (Very High Speed Integrated Curriculent Hardware Description Language) programming language in order to obtain some simulation results, a 128-bit random number is generated as a result of the 80-bit key input. Later, the test criteria have been mentioned briefly and the produced 128-bit random values are subjected to these criteria. Finally, these randomly generated numbers are implemented in the FPGA and corresponding test results are discussed.

Keywords: Random Number Generators, Field Programmable Gate Array, Encryption Algorithms.

TEŞEKKÜR

Yüksek Lisans çalışmamı yapabilmem için, çalışmalarımı destekleyen ve yönlendiren değerli hocam ve danışmanım Prof. Dr. Muzaffer AŞKIN hocama çok teşekkür ederim.

Tez çalışmam sırasında bana yardımlarından ve katkılarından dolayı değerli hocalarım Yrd. Doç. Dr. Bilgin ZENGİN, Yrd. Doç. Dr. Özal YILDIRIM ve Ali Murat GARİPCAN' na sonsuz teşekkürlerimi sunuyorum.

DİLAN BAKIR

TUNCELİ-2017

İÇİNDEKİLER **Sayfa No**

ŞEKİLLER LİSTESİ Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR LİSTESİ

ADD : Ayrık Dalgacık Dönüşümü
ASIC : Application Specific Integrated Circuit
CLB : Configurable Logic Blocks
EPROM : Erasable Programmable Read Only Memory
EEPROM : Electrically Erasable Programmable Read Only Memory
FPGA : Field Programmable Gate Array
FSM : Finite State Machine
GRSÜ : Gerçek Rastgele Sayı Üreteçleri
HDL : Hardware Description Language
IV : Initialization Vector
LFSR : Linear FeedBack Shift Register
NFSR : Nonlinear Feedback Shift Register
NIST : National Institute of Standards and Technology
PAL : Programmable Array Logic
PLA : Programmable Logic Array
PROM : Programmable Read Only Memory
RSA : Rivest , Shamir, Adlemon
RS : Rastsal Sayı
RSÜ : Rastgele Sayı Üreteçleri
SRSÜ : Sözde Rastgele Sayı Üreteçleri
VHDL : Very High Speed Integrated Circuit Hardware Description Language
TADD : Ters Ayrık Dalgacık Dönüşümü

1. GİRİŞ

Ünlü düşünür Francis Bacon yüzyıllar önce "bilginin bir güç" olduğunu ifade etmiştir. Bilgi, insanlar için her zaman kıymetli olmuştur ve çağlar boyunca toplumları etkilemiş, özellikle savaşları kazanmak, para kazanmak ve tarihi şekillendirmek için kullanılan önemli bir güç haline gelmiştir. Bazı bilgilerin hayati derecede önemli olmaya başlaması, onların gizli tutulması zorunluluğunu ortaya çıkarmıştır. Meraklı olan insanoğlu bu bilgilere ulaşmak istese de, geliştirilen çeşitli yöntemlerle bilgiyi çözülmesi güç bir gizem haline getirmeyi başarmıştır. Bunlardan biri olan kriptolojide bilgiyi yalnızca düşmandan gizlemek amacıyla kullanılmış olsa da günümüzde teknolojinin hayatımıza girmesiyle birlikte cep telefonlarından bankamatiklere, e-imzadan mail adreslerine kadar hayatımızın her anında yer alır vaziyete gelmiştir (Yeşilbaş, 2016).

Rastgele sayı (RS), üyeleri belli olan dizi içerisindeki elemanların matematiksel olarak düzgün bir şekilde dağılım yaparak ve daha önceki seçimlerden yeni seçimlerin tahmin de bulunulmayacak şekilde elde ettiğimiz sayıdır (Chaitin, 2001). Rastgele sayı da tahmin edilememezlik çok önemlidir. Bu tahmin edilememezlik sayıların karmaşıklığı ile alakalıdır. Çünkü rastgele sayı üreteçlerinde kullandığımız bu sayıların istenilmeyen kişiler tarafından tahmin edilip anahtar değere ulaşması bizim açımızdan istenilmeyecek durumdur. Rastgele sayılar deterministik olmayan fiziksel olaylar kullanılarak, matematiksel ifadeler, belli bir algoritma veya rastgele sayı üreteci (RSÜ) kullanılarak üretilebilir.

RSÜ, rastgele sayı üretmek için kullanılır. Fakat rastgele sayı üretirken sürekli aynı anahtarı kullanmak tekrarlanılabilen sayı dizilerinin üretilmesine sebep olabilir. Gerçek rastgele sayıyı fiziksel araçlar sağlamasına rağmen, bazı sorunların çözülebilmesi için kısa sürede binlerce rastgele sayıya ihtiyaç duyulabilmektedir. Fakat bu fiziksel araçlarla mümkün değildir. Bu sebepten ötürü yazılımsal algoritmaya dayalı rastgele sayı üreteçleri üzerinde çalışılmaktadır. Belli bir algoritma sonucunda üretilen rastgele sayılar düzenli matematik işlemleri sonucu üretildiği için tam olarak RS değildirler (Chaitin, 2001). RSÜ genel anlamda sözde rastgele sayı üreteci (SRSÜ) ve gerçek rastgele sayı üreteci (GRSÜ) olarak iki gruba ayırıyoruz. GRSÜ doğal bir entropi kaynağı kullanırken, SRSÜ yazılımsal algoritmaları kullanmaktadır.

FPGA, herhangi bir sayısal devreyi veya sistemi oluşturmak için elektriksel olarak programlanabilen ve çok sayıda mantık hücresinden meydana gelen silikon teknolojilerdir. FPGA' lar, sabit fonksiyonlu Application Specific Integrated Circuit (Uygulamaya Özgü Tümleşik Devre) (ASIC) teknolojisinden üstün olarak birçok avantaj sağlamaktadır. ASIC' ler tipik olarak istenilen bir devreyi elde etmek için aylarca sürebilen üretim aşamaları ve yüksek miktarda maliyetlere sahiptir. Buna karşın FPGA' lar bir saniyeden daha kısa bir sürede yapılandırılabilirlik ve tasarımda herhangi bir hata oluşması durumunda sürekli olarak yeniden programlanabilir bir yapı sunmaktadır. Ayrıca FPGA teknolojisine, birkaç dolardan başlayıp birkaç bin dolara kadar olan düşük maliyetlerle her yerde sahip olma imkanı bulunmaktadır (Kuon ve ark., 2007). FPGA teknolojisi iletişim, mobil telefonlar gibi pek çok yeni uygulamalarda en uygun mikro elektronik teknolojisi olmaya başlamıştır. Bunun nedenleri genel olarak bu cihazların oldukça yüksek kapasiteli ve düşük maliyetli olmaları ve elektronik tasarım araçları kullanılarak kısa tasarım süreçleri ile hızlı bir şekilde piyasaya sunulmaları olarak gösterilebilir. VLSI ASIC cihazlarının geliştirilmesindeki yüksek maliyet ve üretim sonrasında tasarım değişikliklerinin uygulanışının yetersiz kalmasından dolayı FPGA bu cihazlara göre büyük avantaj sağlamaktadır (Seals ve Whapshott, 1997).

FPGA' ların sahip olduğu paralel, hızlı işlem yapabilme yeteneği ve sahada birçok kez yeniden programlanabilir olması gibi özellikleri bu cihazların endüstride ve akademik araştırmalarda geniş bir şekilde kullanılmasına olanak tanımıştır. FPGA' lar, güç kalitesi ile ilgili çalışmalarda (Choong ve Reaz, 2005), gerçek zamanlı görüntü işlemede (Torres ve Arias, 2004), yapay sinir ağlarının donanımsal olarak gerçekleştirilmesinde (Omondi ve Rajapakse, 2006), Ayrık Dalgacık Dönüşümü (ADD) (Yılmaz, 2008), rastgele sayı üreteçlerinin donanımsal olarak gerçekleştirilmesinde ve Ters Ayrık Dalgacık Dönüşümü (TADD) gibi sinyal işleme algoritmalarının (Çavuşlu ve ark., 2006) uygulanmasında etkili bir şekilde kullanım alanı bulmuştur (Mahmoud, 2007).

Bu tez çalışmasında, rastgele sayı üreteçlerinden biri olan Trivium algoritmasının FPGA üzerinde gerçekleştirilmesi yapılmıştır.Gerçekleştirilen uygulamada, Xilinx firmasına ait olan Virtex5 FPGA cihazı kullanılmıştır.

Tez çalışmasının ikinci bölümünde şifreleme kavramları, kriptoloji üzerinde durulmuş, şifreleme ve şifre çözme işlemleri ile ilgili bilgiler verilmiştir. Aynı zamanda şifreleme sınıflarına ayırarak açıklanmalarına yer verilmiştir.

Üçüncü bölümde RSÜ hakkında detaylı bilgi verilip tarihsel gelişimine değinilip sınıflandırılması hakkında detaylı bilgi verilmiştir.

Dördüncü bölümde söz konusu olan rastgeleliğin tam olarak gerçeklendiğini gösteren istatistiksel testlerden National Institute of Standards and Technology (Ulusal Standartlar ve Teknoloji Enstitüsü) (NIST) test süitine ve bu süit içerisine dahil edilmiş olan algoritmalara yer verilmiştir.

Beşinci bölümde FPGA hakkında genel bilgi verilmiştir. FPGA' nın tarihsel gelişimine değinilip, genel yapısı, programlanması hakkında açıklama yapılmıştır. Ayrıca VHDL programlama dilinin özellikleri incelenip genel kod yapısı ortaya konmuştur.

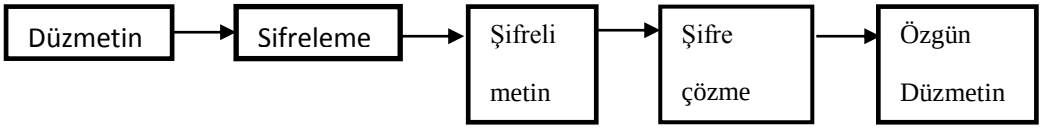
Altıncı bölümde ise gerçekleştirmiş olduğum Trivium Algoritmasının gerçeklenme adımları verilmiştir. Sözde kod yapısı incelenmiş olup Finite State Machine (Sonlu Durum Makinesi) (FSM) hakkında genel bilgi verilmiştir.

1.1. Tezin Amacı ve Kapsamı

Bu tez çalışmasında simetrik şifreleme tekniklerinden akış şifreler ile bu şifreleme teknikleriyle üretilebilen dizi şifreleme algoritmalardan bir tanesi olan Trivium algoritmasının donanımsal olarak FPGA' da gerçekleştirilmesi üzerinedir.Ayrıca kriptografik uygulamalarda kullanılan rastsal sayıların ve bu sayılar kullanılarak geliştirilen anahtarların güvenilirliğini sağlamada önemli kriter oluşturan istatistiksel testler incelenmiştir. Rastgele sayı üreteçlerinden incelenen algoritmalar sonucu gerçekleştirilen donanımlardan elde edilen rastgele sayı dizileri bu testlere tabi tutularak değerlendirmelere yer verilmiştir. NIST test paketinde bulunan çeşitli test teknikleri incelenerek seçilen bazı akış şifrelere bu testler uygulanmış ve sonuçlar değerlendirilmiştir.

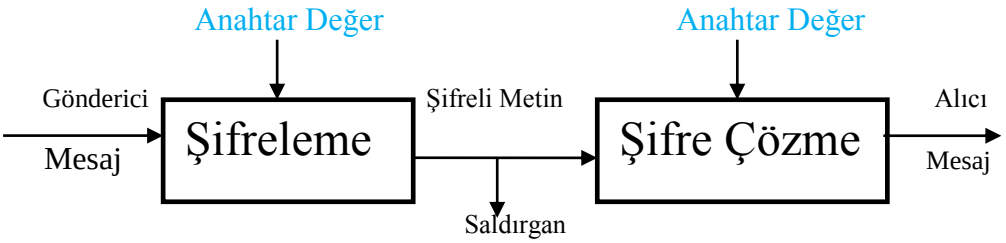
2. ŞİFRELEME KAVRAMLARI

Bulunduğumuz bilgi çağında bilginin giderek artması ile birlikte istenilmeyen kişilere karşı verinin korunması ve saklanılması artık bir sorun haline gelmiştir. Çünkü okul, hastahane, iş yeri, mobil iletişim vb. gibi alanlarda artık internet kavramı dünya çapında tahmin edilmeyecek hızda yayılmıştır. Bu yayılma sonucunda çeşitli kavramlar türemiştir. Kriptoloji iki gruba ayrılır kriptografi ve kriptanaliz dallarıdır. Kriptografinin amacı, verinin güvenliğini sağlamak ile ilgilidir, kriptanaliz ise var olan şifreleri analiz edip çözmektir. Kriptografi, gizlenecek veriyi okunamayacak hale getirmektir. Kriptografi bilgi güvenliğini sağlayacağı zaman matematiksel tekniklerden yararlanarak verinin bütünlüğünü bozmadan doğru ve güvenli bir şekilde çalışır ve bunu yaparken çeşitli algoritmalardan faydalanır. Veri ancak şifreleme algoritmaları ile çözülebilir. Şifreleme için önemli olan, verinin istenmeyen kişilere karşı çözülmesini engellemektir. Deşifreleme (şifre çözümü) ise şifreli verinin belli algoritmadan geçirilerek özgün veriye ulaşılmasıdır. (Soyalıç, 2005). Şifreleme ve deşifreleme Şekil 2.1' de gösterilmiştir.



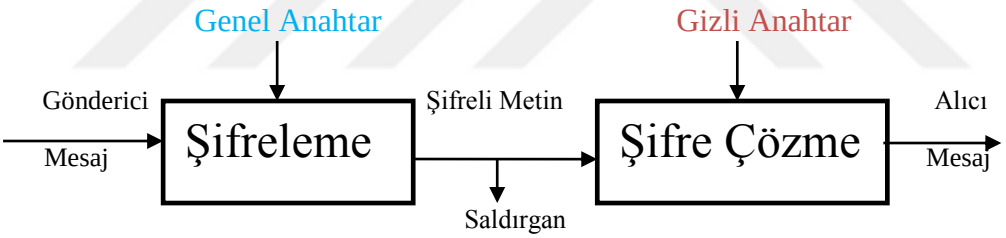
Şekil 2.1. Şifreleme ve şifreyi çözme işlemleri diyagramı

Güncel olarak kullanılan şifreleme algoritmaları iki gruba ayrılmaktadır. Bunlardan birincisi simetrik şifreleme algoritmalarıdır. Simetrik şifreleme algoritmaları en yaygın algoritmalardır. Şekil 2.2' de simetrik şifrelemenin basit mantığı verilmiştir. Blok şifreleme algoritmaları ve akış (stream) şifreler bu gruba girer. Simetrik şifreleme algoritmalarında veriyi şifrelerken ve çözerken aynı anahtarı kullanmaktayız. Bu anahtara gizli anahtar (key) denir ve K olarak ifade edilir. Bu gizli anahtarın değeri ne kadar büyük ve ne kadar karmaşıksa o kadar önemlidir. Anahtarın aldığı tüm değerlere anahtar uzayı denir. Şifreli verinin kalitesi anahtar değerin karmaşıklığına bağlıdır. Simetrik şifrelemede anahtar, sadece alıcı ve gönderici tarafından bilinir. Bu şifrelemeye örnek olarak Vigenere şifresini verebiliriz (Ekdahl ve Johansson, 2003).



Şekil 2.2. Simetrik şifreleme (Bellare ve Rogaway, 2017).

İkinci grup asimetrik şifreleme algoritmalarıdır. Şifreleme yaparken açık anahtar kullanırken şifreyi çözerken gizli anahtar kullanır. Asimetrik şifreleme algoritmaları, Diffie ve Hellman tarafından gelişmeye başlamıştır. Bu fikir anahtarın simetrik olmamasına dayanır. RSA şifreleme algoritması (Rivest ve ark., 1978), asimetrik şifrelemeye örnek olmuştur. Şifreleme yapılacağı zaman açık anahtar karşı tarafa gizlenilmeden verilir. Gizli anahtarı alan taraf bu anahtar sayesinde gizli anahtarı üretir ve veriyi özgün hale getirir (Yerlikaya, 2006). Şekil 2.3' te asimetrik şifreleme yapısı görülmektedir.



Şekil 2.3. Asimetrik şifreleme (Bellare ve Rogaway, 2017).

Şifreleme algoritmalarına örnek vermek gerekirse bunlardan blok şifreleme algoritmaları olarak AES, IDEA, Camellia verilebilir. Akış şifreleme algoritmalarına örnek vermek gerekirse HC-256, RC4, çalışmamda kullandığım Trivium algoritması verilebilir. Asimetrik şifrelemelerine örnek olarak ise ECC, RSA, ElGamal verilebilir (Sakallı, 2011; Aslan, 2008).

2.1. Sınıflandırma

2.1.1. Senkron Dizi Şifreleyiciler

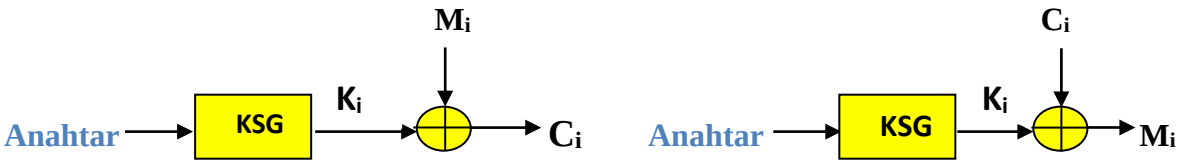
Akış şifreleyiciler şifrelenmemiş özgün metinden bağımsız olarak tohum değeri belirleyen senkronize sistemlerdir. Bu dizi şifreleyicileri şifreleme algoritması kullanarak tohum değer ile birlikte rastgele anahtar dizisi üretir. Senkron akış şifreleme algoritmalarının denklemsel aşamaları aşağıdaki gibi Denklem (2.1), (2.2) ve (2.3) ile verilebilir.

$$\Omega_{i+1} = f(\Omega_i; k); \quad (2.1)$$

$$z_i = g(\Omega_i; k); \quad (2.2)$$

$$c_i = h(z_i ; m_i); \quad (2.3)$$

Bu denklemlerde Ω_0 başlangıç durumdur. f fonksiyonu ise bir sonraki durumu ifade eder. f fonksiyonu güncel durum ile anahtar değerine bağlıdır. İfadedeki g fonksiyonu ise bir sonraki biti elde etmek için kullanılan fonksiyondur. Ürettiğimiz bit güncel duruma ve başlangıçta girilmiş olan anahtar değerine bağlı olarak değişir. Son olarak *h* fonksiyonu şifreli bitin elde edilmesinde kullanılanılır. Şifreleme işlemlerinde h fonksiyonu çoğu kez exor işleminden oluşur. Senkron olan akış şifreleyicilerine ait şifreleme ve şifre çözme süreçlerine Şekil 2.4' te verilmiştir.



Şekil 2. 4. Senkron dizi şifreleyicilerin genel modeli (Doğan, 2006).

Senkron dizi şifreleyicilerinin kullanımında şifre çözme işlemi yapılacaksa alıcı ve gönderen taraf tamamen eş zamanlı çalışmalıdır yani senkronize olmalıdır, aynı anahtar değer kullanmalı ve aynı durumlarda işlem yapmalıdırlar. Verinin gönderileceği zaman karşılaşılabilecek herhangi bir kayıp sırasında şifre çözümlenmesinde hata oluşur. Bu hatayı çözmek için sistemin tekrar senkronize edilmesi zorunluluğu vardır. Şifreli verinin

bazı bitleri iletim sırasında değişikliğe uğradığı zaman diğer bitler bu değişiklikten etkilenmeyecektir. Şifre çözümü doğru bir şekilde gerçekleşecektir. Verinin değişmesi veya kaybolması gibi hataları düzeltmek için ayrı mekanizmalar geliştirilmiştir. Bu mekanizmalar kullanılarak bu hatalar giderilebilir.

2.1.2. Asenkron Dizi Şifreleyiciler

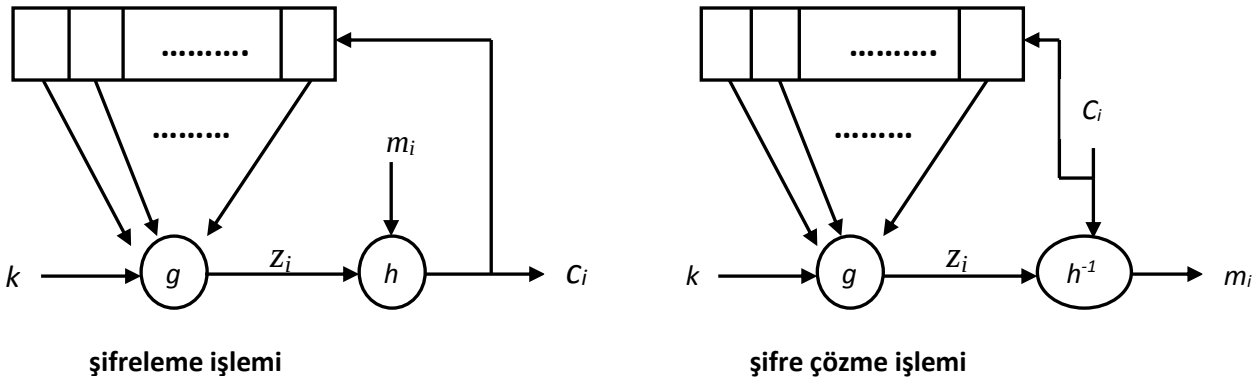
Asenkron dizi şifreleyicileri anahtar dizisini sisteme girilen anahtar tohum değeriyle birlikte şifrelenmiş verinin sabit bitlerini kullanarak oluştururlar. Bu türe ait şifreleme aşamaları Denklem (2.4), (2.5) ve (2.6) ile verilir.

$$\Omega_i = (c_{i-t}, c_{i-t+1}, c_{i-t+2},..., c_{i-1}); \quad (2.4)$$

$$z_i = g(\Omega_i; k); \quad (2.5)$$

$$c_i = h(z_i ; m_i); \quad (2.6)$$

Denklem (2.4)' te eşitlikte yer alan $\Omega_0 = (c_{-t}, c_{-t+1}, c_{-t+2},..., c_{-1})$ ifadesi başlangıç durumu belirtir. Eşitliklerde verilmiş olan g fonksiyonu anahtar biti üreten fonksiyon ve h ise şifrelenmiş biti üreten fonksiyonu belirtmektedir. Asenkron dizi şifreleyicilerine ait şifreleme ve şifre çözme aşamaları Şekil 2.5 'de verilmiştir.



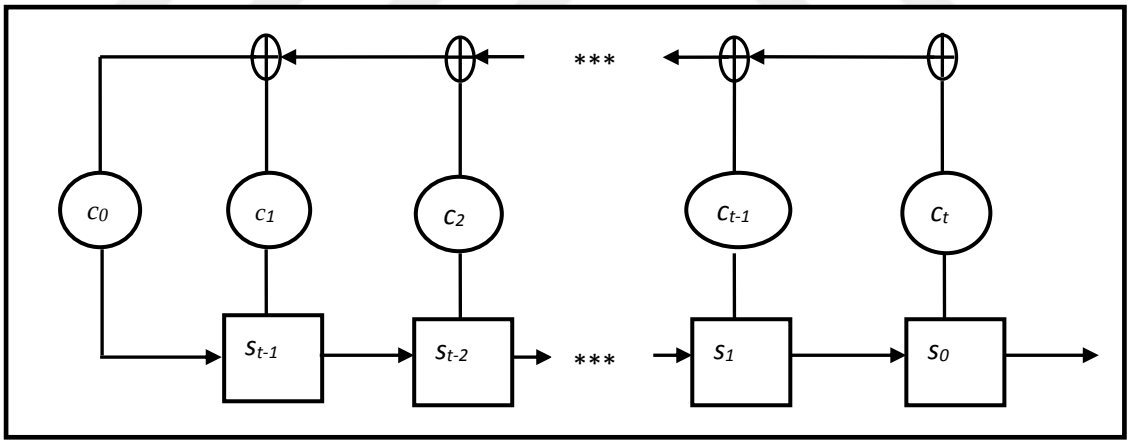
Şekil 2.5. Asenkron dizi şifreleyicilerin genel modeli (Doğan, 2006)

İletilen şifrelenmiş metinde hata veya kaybolma gibi sorunlar senkron sistemlerde şifreyi çözmek asla mümkün değilken asenkron veya kendinden senkronize olabilen dizi şifreleyicilerinin kullanıldığı şifreleme sistemlerinde şifre çözümü mümkündür. Şifre

çözme sırasında senkronize halin kaybolması halinde şifre çözme yolu sadece şifrelenmiş bit dizisine bağlı olduğundan sistem kendini yeniler ve kaybolan bitler dışında bir hata oluşmaz. Şifrelenmiş bitler üzerinde bir hata olduğu zaman tamamen doğru bitler oluşuncaya kadar elde ettiğimiz bitlerde hata meydana gelmiş olur. Hata olasılığını en aza indirmek amacıyla bu sistemlere ek olarak farklı mekanizmalar kullanılır.

2.2. Linear FeedBack Shift Register (LFSR)

Dizi şifreleme algoritmalarının ürettiği anahtarın deterministik bir dağılımına sahip olması ve tahmin edilememesi gerekmektedir. İstenilen özelliklere ait anahtar oluşturmada lineer geri beslemeli ötelemeli yazıcılar sıklıkla kullanılmaktadır. LFSR sisteminin çalışma şekli her birim zamanda lineer bir fonksiyon ile yazılması gereken giriş verisini kaydederken aynı esnada diğer geri kalan verileri bir öteler yani kaydırır. Bu sayede her birim zamanda bir veri saklanmış olur. LFSR' ye ait blok Şekil 2.6’ da verilmiştir (Doğan, 2006).

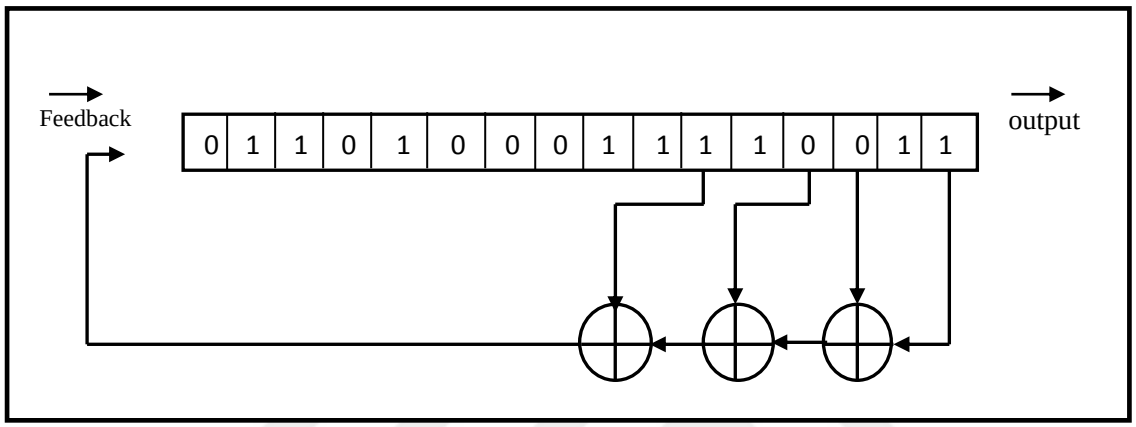


Şekil 2. 6. Genel halde LFSR yapısı (Doğan, 2006).

Şekil 2.6' da yer alan s harfleri LFSR 'ye ait hafıza birimlerini belirtirken *c* harfleri ise geri besleme ifadesine ait gerekli katsayıları belirtir. Yazıcı l bitlik olup c_0 katsayısı 1 alınır. Her $t \geq 0$ anında LFSR yapısı geri beslemeden gelen giriş uyarınca içeriğini günceller ve s_0 bitini üretir. LFSR' nin ilk başlangıç haline tohum denilir. LFSR' nin işlevi düzenli yani deterministik bir yapıda olduğundan üretilen bit dizisi bir önceki bit dizisine bağlıdır. LFSR yapısı FSM yani sonlu durum makinesine bağlı olduğundan kendini tekrarlayan bir

dizi üretecektir. Fakat kaliteli seçilmiş bir LFSR yapısı rastgelelik olasılığı yüksek olan ve kendini uzun süre sonra tekrarlayan bit dizisi üretebilir.

LFSR' nin C_i değerlerine sahip bitler yani bir sonraki durumu belirleyen bitler exor işlemlerine tabi tutularak LFSR' nin giriş biti belirlenir. LFSR yapısındaki yazmaçlarda kayıtlı değerler 2 tabanında modüler aritmetiğe sahip bir polinom olarak ifade edilebilir. Yani katsayıların sadece 0 ve 1 değerlerinden birine ait olabilecekleri açıktır.



Şekil 2. 7. LFSR' ye örnek (Doğan, 2006).

Şekil 2.7’ de verilen örnekte gösterildiği gibi LFSR yapısında G (geri besleme fonksiyonunu) fonksiyonunu oluşturan bitler 11., 13., 14., 16. bitlerdir. G fonksiyonunun eşitlik ifadesini $G(x) = x^{11} + x^{13} + x^{14} + x^{16} + 1 \pmod 2$ şeklinde ifade edebiliriz. G fonksiyonundaki x^n değerleri soldan kaçıncı biti alacağımı, belirtirken 1 değeri hiçbir şey ifade etmemektedir. LFSR sonucu çıkan rastgele bit dizisinin uzun periyotlu olması gerekir bunun için bazı durumlar gerekir. Bunlar; geri beslemeye ait polinomunun kendinden daha basit polinomlara bölünememesi, ayrıca bu polinomunun bitleri çift olmalıdır. Sonlu durum makinesi olarak LFSR' yi düşünebiliriz, n bitlik LFSR yapısı 2^n -1 duruma sahip olabilir.

3. RASTGELE SAYI ÜRETEÇLERİ

İlk olarak kullanılan rastgele sayı üreteçleri fiziksel rastgele sayı üreteçleridir. Zar atma, para fırlatma veya rulet tekerlekleri fiziksel üreteçlerin en yaygın örnekleridir. Bu örnekler rastgele sayı üretme hızları düşük olduğundan daha çok günlük hayatta oyunlarda veya kağıt oyunlarında kullanılır. Bunun için istatistiksel ve şifreleme gibi kısa sürede çok sayıda rastgele sayıya ihtiyaç olan sistemlerde kullanılması mantıklı değildir. Fiziksel sayı üreteçleri çeşitli alanları kullanarak rastgele sayı üretebilir. Bunlara atmosferik gürültülerin genliğini kullanarak rastgele sayı üreten uygulamalar, yine insan davranışları olan bir insanın bilgisayar başında bir yazı yazarken klavye tuşlarına basma hızı ve sertliği çok hassas cihazlar örnek verilebilir.

Rastgelelik olayı günlük olaylarda, hayatımızın içinde, işlerimizde ihtiyacımız olduğu bir kavramdır. Çeşitli uygulama alanlarında örnek verecek olursak oyun teorisi, eğlence, istatistik, nümerik analiz gibi birçok alanlarda kullanılan matematiksel bir olaydır. Kriptolojik uygulamaların büyük bir çoğunluğu da rastgele sayılara ihtiyaç duymaktadır. Oturum anahtarları, imza anahtarları ve parametreleri, kimlik doğrulama protokolleri, geçici anahtarlar, sıfır bilgi ispatı, blok şifreler için başlangıç vektörleri ve yan kanal saldırılarına karşı koruma önlemleri olan körleştirme ve maskeleme işlemleri rastgele sayılara gereksinim duyan kriptolojik uygulamalardan sadece birkaçıdır.

Literatürde genellikle kriptolojik uygulamaların gereksinim duyduğu rastgelelik ile diğer uygulamalar için sağlanması yeterli olan rastgelelik birbiri ile karıştırılmakta ve sonuç olarak çeşitli güvenlik kriterleri sağlanmamaktadır. Bu durum ise tüm sistemin güvenliğini etkilemektedir (Koç, 2009). Örneğin herhangi bir benzetim uygulamasında kullanılacak RSÜ iyi istatistiksel özellikler göstermesi yeterli iken bir şifreleme algoritmasında anahtar olarak kullanılacak rastgele sayıların tahmin edilemez olmasıdır. Diğer bir ifade ile n rastgele bit değerinden oluşan anahtarın tahmin edilmesi ortalama 2^n-1 deneme gerektirmelidir (Özkaynak, 2015).

3.1. Rastgele Sayı Üreteçlerinin Tarihsel Gelişimi

Rastgele sayıların kriptolojik uygulamalarda merkezi kullanımı; Vernam şifreleme sistemi olarak bilinen tek bir ayrıcalıklı veya (exclusive-OR) kapısı kullanılarak, açık metin (plaintext) olarak tanımlanan veri dizisinin anahtar olarak adlandırılan rastgele sayılar kullanılarak karıştırılması protokolüdür. Koşulsuz güvenli şifreleme protokolü olarak ifade edilen bu dizi şifreleme protokolünün pratik uygulamalarda da güvenli olabilmesi için her bir anahtarın (rastgele sayının) sadece tek bir defa kullanılması ve sayıların gerçek rastgele olması istenmektedir.

Bilgisayar tabanlı RSÜ' nün tasarım ve analizi için önemli kaynaklardan biri Knuth'un "The Art of Programming" isimli klasik eseri olmuştur. Ripley 1983 yılında yayınladığı çalışmasında; kişisel bilgisayar kullanıcılarının uzmanlar tarafından geliştirilen rastgele sayıları üretmek için kullandıkları kütüphanelere erişimindeki problemleri adreslemiştir. Ripley çalışmasında küçük kişisel bilgisayarlar tarafından sağlanan yetersiz RSÜ kullanıcı programları ile yer değiştirerek üstel, normal ve Poisson dağılımına sahip diziler üreten etkili yöntemler geliştirmiştir.

1990 yılında L'Ecuyer orta seviyedeki bir bilgisayar kullanım bilgisine sahip bir kullanıcı için düzgün rastgele değişkenler üretilmesi problemini çözmüştür. Yine 1990 yılında James tarafından yapılan bir çalışmada Monte Carlo hesaplamaları için sözde rastgele sayı üreteçlerinin pratiği ile ilgilenmiştir. Bu yıl yapılan bir diğer çalışmada ise Lagarias sözde RSÜ, tek yönlü fonksiyonlar ve gizli anahtarlı şifreleme sistemleri arasındaki bağlantıyı ortaya koyarak sayı teorisini temel alan sözde rastgele sayı üreteçlerini tanılamıştır. Lagarias çalışmasında ayrıca bu üreteçlerin kripto analizi üzerine elde ettiği sonuçları özetlenmiştir.

1991 yılında Zeng ve arkadaşları akan şifreleme sistemleri için temel problem olan kısa uzunluklu bir rastgele anahtardan tahmin edilemez uzunluklu ikili sinyallerin üretilmesi problemini ele almıştır.

RSÜ üzerine kapsamlı bir değerlendirme çalışması ise Ritter tarafından gerçekleştirilmiştir.

3.2. Rastgele Sayı Üreteçlerinin Sınıflandırılması

Gerçek dünya RSÜ' leri iki temel sınıfa ayrılmaktadır. Birinci sınıf gerekirci RSÜ' leri (sözde veya sahte rastgele sayı üreteçleri olarak da bilinmektedir) içermektedir. Gerekirci RSÜ bir tohum değerinden başlayıp algoritmik olarak rastgele sayıları üretmektedir. İkinci sınıf RSÜ' leri ise gerçek RSÜ' leridir ve kendi içerisinde fiziksel gerçek RSÜ ve fiziksel olmayan gerçek RSÜ olarak ikiye ayrılmaktadır. Fiziksel gerçek RSÜ' ler elektronik devrelerin gerekirci olmayan etkilerini (Zener diyottaki gürültü, yarı iletkendeki termal ısı, serbest çalışan osilator gibi) veya fiziksel deneyleri (radyo aktif bozulma arasında geçen zaman, kuantum rastgele süreçler gibi) kullanır. Fiziksel olmayan gerçek RSÜ'leri ise gerekirci olmayan olaylarda (sistem zamanı, hard disk arama zamanı, RAM içeriği, kullanıcı etkileşimleri gibi) ortaya çıkmaktadır.

3.2.1. Sözde RSÜ

Sözde rastgele sayı üreteçleri adından da anlaşıldığı gibi gerçek rastgele sayı üretmezler, amaçları gerçekliğe en yakın olasılıkla rastgele sayı dizileri oluşturmaktır. Sözde rastgele sayı üreteçleri tohum değeri (başlangıç değeri) alarak belli bir algoritmaya tabi tutarak rastgele sayıları tekrarlı olarak üretirler. Sözde rastgele sayı üreteçleri çoğu tohum değeri olarak zaman etmenini kullanmaktadır (Dutang ve Wuertz, 2009). Sözde RSÜ rastgele sayı üretme hızlarından ve tekrar üretilebildiklerinden dolayı önemlidirler.

Bir rastgele sayıda olması gereken özellikler şunlardır;

- Düzgün dağılım
- Birbirlerine bağımlı olmamaları

Sözde rastgele sayı üreteçlerinin özellikleri ise;

- Hızlı bir şekilde rastgele sayı üretme üretmeleridir
- Taşınılabilir özellikte olmalıdır
- Rastgele sayının taşıdığı düzgünlük ve bağımsızlığı bunlarda taşımalıdır

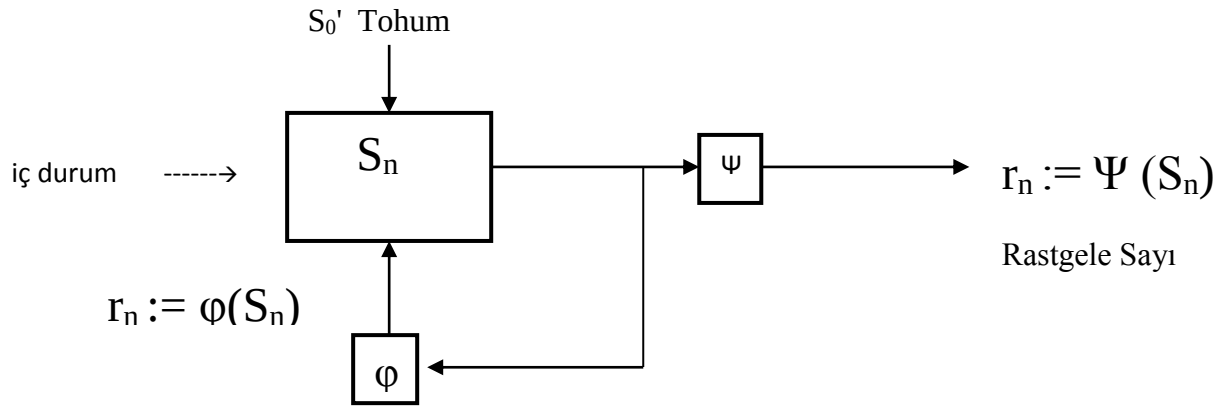
Sözde rastgele sayı üreteçlerinin seçtiği matematiksel fonksiyon eğer iyi bir özelliğe sahip olmazsa oluşacak sorunlar;

- Tohum değerler daha kısa periyotlu olabilir. Kısa periyotlu oluşu bazı değerlerin tekrar tekrar kullanılmasına sebep olabilir buda tahmin edilebilirlik olasılığını arttırır.

- Rastgele üretilen sayıların düzenliliği zayıf olabilir.
- Rastgele üretilen sayıların zayıf ölçülü dağılım olabilir.

Şekil 3.1' de sözde RSÜ'nin genel tasarım mimarisi gösterilmiştir. $r_1, r_2, r_3....r_n \in R$ rastgele sayıları $S_n \in S$ sözde RSÜ' nin iç durumlarını göstermektedir. Burada S ve R sonlu kümeleri sırasıyla sözde RSÜ' nin durum uzayı ve çıkış uzayı olarak adlandırılmaktadır. $\Psi: S \rightarrow R$ çıkış geçiş fonksiyonu, mevcut Sn iç durumundan Rn rastgele sayısını üretir. Ardından Sn durumu φ durum geçiş fonksiyonu kullanılarak $Sn+1$: $\varphi(S_n)$ biçiminde güncellenir. İlk iç durum değeri olan S_1 değeri tohum değeri olan S_0 kullanılarak $S_1: \varphi(S_0)$ biçiminde veya daha karmaşık tasarımlar kullanılarak güncellenir. S_0 tohum değerinden tüm $s_1, s_2,s_3....s_n$ iç durumlarının ve $r_1, r_2,r_3....r_n$ rastgele sayıların tahmin edilebileceği açıktır. Bu yüzden güvenlik öncelikli uygulamalarda tohum değeri rastgele seçilmeli ve durum geçiş fonksiyonu ile çıkış fonksiyonunun yeterince karmaşık olması gerekmektedir.

Sözde RSÜ' lerinin dezavantajı çıkış değerlerinin tohum değeri aracılığıyla tamamen belirlenebilmesi ve gelecek rastgele sayıların sadece mevcut iç duruma bağlı olmasıdır. Bu yüzden iç durum cihaz aktif olmasa bile korunmalıdır. Özellikle sözde RSÜ tasarımları bilgisayar veya akıllı kartlar üzerinde gerçekleştiriliyor ise mevcut iç durum sonraki oturum için tehlikeli olabilir.



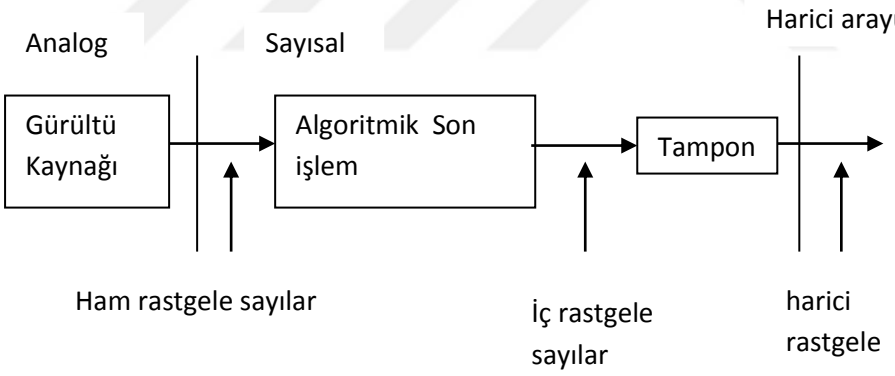
Şekil 3.1. Sözde RSÜ' nün genel tasarım mimarisi (Avaroğlu ve ark., 2014).

φ : durum geçiş fonksiyonu

Ψ: çıkış fonksiyon

3.2.2. Gerçek RSÜ

Şekil 3.2' de bir gerçek RSÜ' nin genel tasarım mimarisi gösterilmiştir. Tasarımın kalbini gürültü kaynağı oluşturmaktadır. Gürültü kaynağı tipik olarak elektronik devreler (gürültülü diyot veya serbest çalışan osilatör) veya fiziksel deneyler (radyoaktif bozulma veya ışığın kuantum etkisi) ile gerçekleştirilir. Gürültü kaynağı sürekli zamanlı analog sinyaller üretmekte ve aynı adımda bu değerler periyodik olarak sayısallaştırılarak ikili (binary) değerlere dönüştürülmektedir. Sayısal değerler, sayısallaştırılmış analog sinyaller olarak adlandırılırlar. Gerçek RSÜ' lerin içerebileceği potansiyel zayıflıkları indirgemek için sayılar değerlere çeşitli algoritmik son işlem (post-process) yöntemleri uygulanabilir. Ancak zayıflıkları indirgeme işlemi çoğu zaman basit bir dönüşümden daha fazlasına (bağımlılıkları eşiklemek veya veri sıkıştırma işlemi gibi) gereksinim duyduğundan bu işlemler RSÜ' nün çıkış hızını düşürmektedir. Bu yüzden gerçek RSÜ tasarımlarında güçlü gürültü kaynakları kullanılarak algoritmik son işleme ihtiyaç duymadan iç rastgele sayıları doğrudan çıkışa veren mimariler tercih edilmektedir.



Şekil 3.2. Gerçek RSÜ' nün genel tasarım mimarisi (Avaroğlu ve Türk, 2014).

4. RASTGELE SAYI ÜRETEÇLERİ İÇİN İSTATİSTİKSEL TESTLER

Bu testler, rastgele sayı üretecinin ürettiği rastgele sayıların gerçekliğe yakın olma durumunu ölçer. Bir sayı dizisinin rastgele olabilmesi için istatiksel testlerden başarılı olması gerekmektedir. Bu istatistiksel testlerden bir tanesi bile başarılı olmaz ise üretilen bu dizi rastgele sayılmaz.

Üretilen rastgele sayıların uygun koşullarda üretildiğini kontrol etmek ve rastgeleliğe uygunluğunu ölçmek için sadece bir test kullanmak uygun olmamaktadır. Bunun içindir ki bu alanda çeşitli test paketleri çıkartılmıştır. Bunlara örnek olarak NIST, FIBS 140, DieHard verilebilir. Özellikle rastgele sayıların kriptografik uygulamalarda kullanılabilmesi için bu rastgelellikleri belirleyen testlerden geçmesi gerekmektedir. Bu tez çalışmasında hipotez test tabanlı NIST istatistiksel test süiti açıklanacaktır. Bu hipotez testler üretilen 0 veya 1 sayısının rastgele olup olmadığını belirler. Bu amaç için NIST test süitinde iki önemli parametre vardır bunlar sırasıyla α ve P değeri değeridir. Önem seviyesi olarak bilinen α, 0.01 olarak seçilmesi test edilecek sayıların rastgeleliğinin 99% güven değerine sahip olduğunu belirtir. Diğer parametre P değeri rastgeleliğin ölçüsü olarak bilinir. Eğer P değeri 1'e eşit olursa sayılar mükemmel rastgeleliğe sahiptir denir. P değeri sıfıra eşit olursa sayıların rastgeliğinden söz edilemez. Kriptolojide üretilen rastgele sayıların önem durumunu α için uygun bir değer seçilmelidir. Daha sonra P değeri α' nın aldığı değerden eşit veya büyük olursa test başarılı olur değilse bu üretilen sayılara rastgele diyemeyiz. Tipik olarak önem seviyesi [0.001, 0.01] aralığında seçilir (Büyüksaraçoğlu, 2011).

4.1. FIPS 140-1 Testleri

FIPS 140-1 testi dört tane ayrı testten oluşur. RSÜ' nün çıkışından alınan ve 20000 tane bit içeren bir bit dizisi dört teste birden tabi tutulur ve dizinin rastsal olabilmesi için tüm testlerden geçmesi gerekir. Bu dört test aşağıda açıklanmıştır (Büyüksaraçoğlu, 2011).

4.1.1. Monobit Testi

Bu testin amacı, bit dizisindeki 0 ve 1 sayısının rastsal bir diziden beklendiği gibi olup olmadığını tespit etmektir. Testin başarılı olabilmesi için 20000 bitlik bir dizideki '1' sayısının 9654 < n < 10346 aralığında olması gerekir (Büyüksaraçoğlu, 2011).

4.1.2. Poker Testi

Bu testte, 'k' bit içeren dizi, m k >5.2^m olacak şekilde, üst üste çakışmayan m bitlik parçalara ayrılır ve i. parça n_i diye adlandırılır. Rastsal bir diziden beklenen, tüm m bitlik blokların k uzunluklu bir dizide aynı sayıda birbirini tekrar etmesidir. Testin başarılı olabilmesi için, $X = 2^m/k\ [\sum_{i=0}^{2^m} n_i^2] - k$ formülüyle hesaplanan X değerinin, k= 20000 ve m= 4 için, 1.03 < X < 57.4 aralığında olması gerekir (Büyüksaraçoğlu, 2011).

4.1.3. Blok Testi

Elimizdeki bit dizisinin bu testten başarıyla geçmesi için, dizide ardarda gelen '1' ve '0' lardan oluşan çeşitli uzunluktaki blokların (runs'ların) sayısının Tablo 4.1' de belirtildiği gibi olması beklenir. 6 bitten daha uzun bloklar 6 bitlik olarak kabul edilmektedir (Büyüksaraçoğlu, 2011).

Tablo 4.1. Run testi koşulları

Blok Uzunluğu	**Blok Sayısı Aralığı**
1	**2267-2733**
2	**1079-1421**
3	**502-748**
4	**223-402**
5	**90-223**
6+	**90-223**

4.2. NIST Testleri

NIST testi, FIPS 140-1 testine göre çok daha güçlü bir testtir. FIPS 140-1 testini geçen bir bit dizisi NIST 800-22 testinden kalabilir. Bu nedenle ciddi uygulamalarda NIST 800-22 tercih edilmektedir. NIST testleri uzun verilerden oluşan verileri test etmek için kullanılır. Diğer testlere göre daha kalitelidir. Şöyle demek istersek diğer testlerden geçen bu testten geçemeyebilir. NIST testleri genel olarak 15 ayrı testten oluşur ve teste girecek bit dizisi bu testlerden başarılı olmak zorundadır (Büyüksaraçoğlu, 2011). Yoksa bu bit dizisine rastgeledir diyemeyiz. Bu testlere kısaca değinelim;

4.2.1. Frekans Testi

Alınan dizide sıfır ve birlerin oranını test eder. Bu testte herhangi bir parametre kullanılmamaktadır. Bu testin başarılı olabilmesi için alınan dizinin uzunluğunun en alt sınırı 100 olmalıdır. Test denklemleri kullanılarak üretilen değer olan p > 0.01 ise dizi rastsal olarak kabul edilir (Büyüksaraçoğlu, 2011).

n : Bit dizisinin boyutu

ε : RNG veya PRNG ile üretilen bit dizisi

S_n: Bit dizisinin toplam değeri (Bit dizisindeki 0'lar (-1), 1' ler ise kendi değeri kabul edilerek toplama işlemi gerçekleştirilerek elde edilen sonuçtur).

$$S_{obs} = |S_n| / \sqrt{n} \tag{4.1}$$

erfc : Hata fonksiyonu $erfc(z) = 2/\sqrt{\pi} \int_z^{\infty} e^{-u^2} du$ (4.2)

$$P\text{-değeri} = erfc(S_{obs}/\sqrt{\pi}) \tag{4.3}$$

Üretilen bit dizisi;

$\varepsilon =$ 1011011010110110001010101010101110011100011011000101000110010011010101001101010010101100110011100110

$n = 100$ ve $S_{100} = 2$ olduğunda $S_{obs} = 0.2$ olur.

P değeri = *0.843053 > 0.01* olduğundan dizi rastgele kabul edilir (Büyüksaraçoğlu, Buluş, 2009).

4.2.2. Blok Frekans Testi

Alınan bir veride bulunan 0 ve 1' lerin oranını M bitlik bloklar içinde kontrol eder. Bu testte kullanılan tek parametre M (blok uzunluğu)' dir. Blok uzunluğu 1 olarak alındığında blok frekans testi, frekans testine dönüşür. Her bir bloktaki 1'lerin beklenen oranı M/2' dir. Bu test ki-kare dağılımını kullanır. Testin sonunda elde edilen p-değeri çok küçük çıkması, dizideki bloklarda 1' lerin ve 0' ların oranının 1/1' den fazlasıyla saptığını gösterir. Testin geçerli olabilmesi için blok uzunluğunun en az 20, dizi uzunluğunun da en az 100 olması gerekir (Büyüksaraçoğlu, 2011).

M **:** Blok uzunluğu

n **:** Dizi uzunluğu

ε: RNG veya PRNG ile üretilen bit dizisi

π : Bit dizisindeki 1' lerin sayısı

X²(obs) **:** Beklenen oran (1/2) ile karşılaştırılan verilmiş *M* bit blok içerisindeki 1' lerin gözlemlenen oranının nasıl olduğunun ölçüsüdür.

$$\pi_i = \sum_{j=1}^{M} \varepsilon_{(i-1)M+j} \tag{4.4}$$

$$\chi^2(obs) = 4M\sum_{i=1}^{N}(\pi_i - 1/2)^2 \tag{4.5}$$

- $N = \lfloor n/M \rfloor$ örtüşmeyen bloklar halinde girilen dizi bölünür. Kullanılmayan bitler atılır.

  n=10 , m=3 alınırsa ε=0110011010, $N = 10/3 \approx 3$

  011, 001 ve 101 bloklarından oluşur.

Gama Fonksiyonu: $r(z) = \int_0^{\infty} t^{z-1} e^{-t} dt$ (4.6)

Ters Gama Fonksiyonu: $1/r(z) \int_0^{x} t^{z-1} e^{-t} dt$ (4.7)

P değeri = igamc(ters gama fonksiyonu) $(N/2, x^2(obs)/2)$ (4.8)

Üretilen bit dizisi;

ε = 10110110101101100010101010101011100111000110110001010001100100110101

01001101010010101100110011100110

$\pi_1 = 3/5, \pi_2 = 1/2, \pi_3 = 1/2, \pi_4 = 3/5, \pi_5 = 1/2, \pi_6 = 2/5, \pi_7 = 1/2, \pi_8 = 2/5, \pi_9 = 3/5, \pi_{10} = 1/2$

$x^2(obs)=2$

P değeri = igamc ($N/2,x^2/2$) (4.9)

x^2 değerini yerine yazarsak => *P* değeri = *igamc* (5,1) = 0,9963015 sonucunu elde ederiz.

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir (Büyüksaraçoğlu ve Buluş, 2009).

4.2.3. Akış Testi

Bu test bit dizisindeki akışların toplam sayısıyla ilgilidir. Akış ardışık aynı bit sıralamasını ifade eder. Böylece 0' lar ve 1' ler arasındaki dalgalanmaların kontrolü sağlanarak üretilen bit dizisinin yavaş ya da hızlı olabileceği konusunda fikir verir (Büyüksaraçoğlu, 2011).

$V_n(obs)$ **:** Akışların sayısı

π **:** Bit dizisindeki 1'lerin sayısı

$$\pi = \sum j\varepsilon_j/\mathrm{n} \quad (4.10)$$

$$|\pi - 1/2| \geq \tau \quad (4.11)$$

$$\tau = 2/\sqrt{n} \quad (4.12)$$

$V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1\ \varepsilon_k = \varepsilon_{k+1}$ ise r(k) = 0 diğer durumlarda r(k) = 1 olarak alınır.

P değeri = $erfc(|V_n(obs)-2n\ \pi\ (1-\pi)|/2\sqrt{2\pi n}(1-\pi))$ (4.13)

Üretilen bit dizisi;

ε = 10110110101101100010101010101011100111000110110001010001100100110101010010011010100101011001100111001 10

$\pi = 0.51$ $\tau = 0.63246$ değerleri alındığında;

$r(1) = 1$ $r(2) = 1$ $r(3) = 0$ $r(4) = 1$ $r(5) = 1$

$r(6) = 0$ $r(7) = 1$ $r(8) = 1$ $r(9) = 1$ $r(10) = 1$

$r(11) = 0$ $r(12) = 1$ $r(13) = 1$ $r(14) = 0$ $r(15) = 1$

$r(16) = 0$ $r(17) = 0$ $r(18) = 1$ $r(19) = 1$ $r(20) = 1$

$r(21) = 1$ $r(22) = 1$ $r(23) = 1$ $r(24) = 1$ $r(25) = 1$

$r(26)=1 \quad r(27)=1 \quad r(28)=1 \quad r(29)=1 \quad r(30)=1$

$r(31)=0 \quad r(32)=0 \quad r(33)=1 \quad r(34)=0 \quad r(35)=1$

$r(36)=0 \quad r(37)=0 \quad r(38)=1 \quad r(39)=0 \quad r(40)=0$

$r(41)=1 \quad r(42)=0 \quad r(43)=1 \quad r(44)=1 \quad r(45)=0$

$r(46)=1 \quad r(47)=0 \quad r(48)=0 \quad r(49)=1 \quad r(50)=1$

$r(51)=1 \quad r(52)=1 \quad r(53)=0 \quad r(54)=0 \quad r(55)=1$

$r(56)=0 \quad r(57)=1 \quad r(58)=0 \quad r(59)=1 \quad r(60)=1$

$r(61)=0 \quad r(62)=1 \quad r(63)=0 \quad r(64)=1 \quad r(65)=1$

$r(66)=1 \quad r(67)=1 \quad r(68)=1 \quad r(69)=1 \quad r(70)=1$

$r(71)=0 \quad r(72)=1 \quad r(73)=0 \quad r(74)=1 \quad r(75)=1$

$r(76)=1 \quad r(77)=1 \quad r(78)=1 \quad r(79)=0 \quad r(80)=1$

$r(81)=1 \quad r(82)=1 \quad r(83)=1 \quad r(84)=1 \quad r(85)=0$

$r(86)=1 \quad r(87)=0 \quad r(88)=1 \quad r(89)=0 \quad r(90)=1$

$r(91)=0 \quad r(92)=1 \quad r(93)=0 \quad r(94)=0 \quad r(95)=1$

$r(96)=0 \quad r(97)=1 \quad r(98)=0 \quad r(99)=1$

$$V(obs) = 0.420184\, n \tag{4.14}$$

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.4. Bloktaki En Uzun Birler Testi

Bu testte blok uzunluğu M bit olan grupta en uzun birler üzerinde çalışır. Bloktaki en uzun birler testinin sadece bir parametresi vardır bu parametre ise M' dir. Alınan dizide M bloğu n tane bloğa bölünür. Bu bloklar içerisindeki en uzun birler grubuna bakılır. Beklenen değerlerle bu değerler kıyaslanır sapma miktarı ölçülür. Bu test için kullanılan dağılım ki-kare dağılımıdır.

n: bit dizisinin uzunluğu

ε: RSÜ veya SRSÜ ile üretilen bit dizisi

m: Her bir bloğun uzunluğu

N: Örtüşmeyen blokların sayısı

Tablo 4.2 Bloktaki en uzun birler test parametreleri

Minimum n	M
128	8
6272	128
750000	10^4

Dizi M bitlik bloklara bölünür ve kategori halinde her bir blok için en uzun birlerin akışının frekansı v_i hesaplanır. Bu değerler Tablo 4.3' de gösterilmiştir. Tablolardaki her bir hücre belirli uzunluktaki birlerin akışının sayısını içerir.

Tablo 4.3. Belirli uzunluktaki birlerin akış sayıları

v_i	M=8	M=128	M=10^4
V_0	≤1	≤4	<10
V_1	2	5	11
V_2	3	6	12
V_3	≥ 4	7	13
V_4		8	14
V_5		≥ 9	15
V_6			≥ 16

$$P(v \leq m) = \sum_{r=0}^{M} \binom{M}{r} P\left(v \leq \frac{m}{r}\right) 1/2^M \quad (4.15)$$

$$x^2(obs) = \sum_{i=0}^{K} (v_i - N\pi_i)^2 / N\pi_i \quad (4.16)$$

K=5 , M=128							
Sınıflar	*v* <= 1	*v* = 2	*v* = 3	*v* >= 4			
Olasılıklar	π_0= 0,2148	π_1=0,3672	π_2= 0,2305	π_3=0,1875			

K=5 , M=128							
Sınıflar	v <= 4	v = 5	v = 6	v = 7	v = 8	v >= 9	
Olasılıklar	π_0=0,1174	π_1=0,2430	π_2 = 0,2493	π_3=0,1752	π_4=0,1027	π_5=0,1124	
K=5 , M=512							
Sınıflar	v <= 6	v = 7	v = 8	v = 9	v = 10	v >=11	
Olasılıklar	π_0= 0,1174	π_1=0,2460	π_2 = 0,2523	π_3=0,1755	π_4=0,1015	π_5=0,1077	
K=5 , M=1000							
Sınıflar	v <=7	v = 8	v = 9	v = 10	v = 11	v >= 12	
Olasılıklar	π_0=0,1307	π_1=0,2437	π_2 = 0,2452	π_3=0,1714	π_4=0,1002	π_5=0,1088	
K=6 , M=10000							
Sınıflar	v <= 10	v = 11	v = 12	v = 13	v = 14	v = 15	v >=16
Olasılıklar	π_0= 0,0882	π_1=0,2092	π_2 = 0,2483	π_3=0,1933	π_4=0,1208	π_5=0,0675	π_5=0,0727

P degeri = igamc$(K/2, X^2(obs)/2)$ (4.17)

K = 3 ve M = 8 değerleri verildiğinde üretilen bit dizisi;

ε = 11001100000101010110110001001100111000000000001001001101010100010001001111010110100000001101011111001100111001101101100010110010

n = 128

Alt Blok	Max-Run	Alt Blok	Max-Run
11001100	(2)	00010101	(1)
01101100	(2)	01001100	(2)
11100000	(3)	00000010	(1)
01001101	(2)	01010001	(1)
00010011	(2)	11010110	(2)
10000000	(1)	11010111	(3)
11001100	(2)	11100110	(3)
11011000	(2)	10110010	(2)

$v_0 = 4$, $v_1 = 9$, $v_2 = 3$, $v_3 = 0$; $\chi^2 = 4.882457$

P değeri = *0.180609* ≥ 0.01 olduğunda dizi rastgele kabul edilir (Büyüksaraçoğlu ve Buluş, 2009).

4.2.5. Rank Testi

Rank testinde belli uzunluklu bit blokları kullanılır ve bu blokların her biri bir satırı belirtecek şekilde kullanılır ve daha sonra bir matris oluşturulur. Oluşturulan bu matrisin rankı hesaplanır daha sonra lineer bağımlılığı incelenir (Büyüksaraçoğlu, 2011).

n: Bit dizisinin boyutu

M: Her bir matristeki satır sayısı. Test için bu sayı 32 kabul edilir.

Q: Her bir matristeki sütun sayısı. Test için bu sayı 32 kabul edilir.

N: Matris Sayısı $N = [n/MQ]$ (4.18)

R_l **:** Her bir matrisin rankı

F_M **:** $R_l = M$ olan matris sayısı

F_{M-1} **:** $R_l = M - 1$ olan matris sayısı

$N - F_M - F_{M-1}$ **:** Arta kalan matrislerin sayısı

$$X^2(obs) = \frac{(F_M - 0.2888N)^2}{0.2888N} + \frac{(F_M - 0.2888N)^2}{0.5776N} + \frac{(N - F_M - F_{M-1} - 0.1336N)^2}{0.1336N}$$

P değeri : $e^{-x^2(obs)/2}$

P değeri = 0.180609 ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.6. Ayrık Fourier Dönüşümü Testi

Bu test dizinin hızlı Fourier dönüşümündeki tepeciklerin yüksekliklerini sınamaktadır. Bu test dizide rastsallığı engelleyici herhangi bir baskın harmoninin olup olmadığını sınamaktır. Bu testin amacı ise rastgelelik varsayımından bir sapma gösteren, test edilen sıradaki periyodik özellikleri (yani, birbirine yakın olan tekrarlı kalıpları) tespit etmektir. Tepe yüksekliklerinin %95 i aştığı durumlar rastgelelik için iyi sayılmaktadır (Büyüksaraçoğlu, 2011).

n: Bit dizisinin boyutu

d: %95 eşik değerinden fazla olan elemanların beklenen ve gerçekleşen sayısı arasındaki farkın normalize değeri.

T: %95 zayıflıktaki eşik değeri $T = \sqrt{\log(1/0.05)^n}$ (4.19)

M: Modül *(S´)* ≡ |*S*'|, *S´* bit dizisinin ilk n/2 elemanı içindeki alt dizileri temsil eder ve modül fonksiyonu zayıf eşik değerlerinin serisini üretir.

***N*:** Teorik olarak T değeri daha az sayıda beklenen eşik değeri sayısı

$N_0 = 0.95n/2$

***N_1*:** Gözlenen ve M'deki T değerinden daha az sayıda beklenen eşik değeri sayısı

$d = (N_1\text{-}N_0)/\sqrt{n(0.95)(0.05)/4}$

P değeri : erfc|d|/$\sqrt{2}$

N_0 : Teorik olarak T değerinden daha az sayıda beklenen eşik değeri sayısı

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.7. Çakışmayan Şablon Eşleşme Testi

m bitlik bir bloğun dizi içinde tekrarını inceler. Eşleşme bulunamazsa blok 1 bit kaydırılır. Tekrar edilmesi halinde, tekrar edilen bloktan itibaren m bit öteleme yapılarak yeni bir m bitlik blok oluşturulur (National Institute of Standard and Technology, 2017).

m : Her bir şablonun bit uzunluğu. Şablon hedef dizidir.

n: Test edilen bütün bit dizisi uzunluğu

ε: RSÜ veya SRSÜ ile üretilen bit dizisi

B: Eşleştirilecek m bit şablon; B test kodu içerisinde bulunan periyodik olmayan örnekler şablon kütüphanesinde tanımlı 0 ve 1' lerden oluşan dizidir.

M: Test edilecek ε alt dizilerinin bit uzunluğu

N: Bağımsız blokların sayısı

Wj (j = 1, …, N): j. blok içinde oluşan *B*' lerin sayısı.

ε = 10100100101110010110 için m= 3 ve B= 001 alınırsa W_1= 2 ve W_2= 1 olur. İ lgili adımlar aş ağıdaki tabloda gösterilmektedir.

Tablo 4.4. Çakışmayan şablon testinde blok içerisindeki B şablonlarının bulunma sayısı

Bit Pozisyonları	**Blok 1**		**Blok 2**	
	Bitler	**W_1**	**Bitler**	**W_2**

1-3	101	0	111	0
2-4	010	0	110	0
3-5	100	0	100	0
4-6	001 (bulundu)	1 arttırma	001 (bulundu)	1 arttırma
5-7	Test edilmedi		Test edilmedi	
6-8	Test edilmedi		Test edilmedi	
7-9	001	2 arttırma	011	1
8-10	010 (bulundu)	2	110	1

$\mu = (M-m+1) / 2^m \qquad \sigma^2 = M [(1/2^m) - (2m-1/2^{2m})]$ (4.20)

$x^2 (obs) = \sum_{j=1}^{N} (w_j - \mu)^2 / \sigma^2$ (4.21)

P değeri = igamc (N/2, $x^2(obs)/2$)

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.8. Çakışan Şablon Eşleşme Testi

m bitlik bir bloğun dizi içinde tekrarını inceler. Eşleşme bulunamazsa blok 1 bit kaydırılır. Tekrar edilmesi halinde, tekrar edilen bloktan itibaren m bit öteleme yapılarak yeni bir m bitlik blok oluşturulur (National Institute of Standard and Technology, 2001).

m : Her şablonun bit uzunluğu. Şablon amaçlanan koşuldur.

n : Testteki tüm bit dizisinin uzunluğudur.

ε : RNG veya test edilen PRNG tarafından üretilen bit dizisi

B: Eşleşecek m-bitlik şablon

M: Test edilecek ε' nın alt dizilerinin bit uzunluğu.

N: Bağımsız blokların sayısı. Test kodunda 8 sabit değerini alır.

Wj (j = 1, …, N) : j. blok içinde oluşan B' lerin sayısı.

ε = 10100100101110010110 için m= 3 ve B= 001 alınırsa W_1= 2 ve W_2= 1 olur. İlgili adımlar aşağıdaki Tablo 4.4' de gösterilmektedir.

Tablo 4.5. Çakışan şablon testinde blok içerisindeki *B* şablonlarının bulunma sayısı

	Blok 1		Blok 2	
Bit Pozisyonları	***Bitler***	***W1***	***Bitler***	***W2***
1-3	101	0	111	0
2-4	010	0	110	0
3-5	100	0	100	0
4-6	001 (eşleşti)	1 arttırma	001 (eşleşti)	1 arttırma
5-7	İşlem Yok		İşlem Yok	
6-8	İşlem Yok		İşlem Yok	
7-9	001	2 arttırma	011	1
8-10	010 (eşleşti)	2	110	1

μ = (M-m+1)/2^m (4.22)

σ^2 = M(1/2^m- (2m-1)/2^{2m}) (4.23)

$X^2(obs) = \sum_{j=1}^{N}(W_j - \mu)^2/\sigma^2$ (4.24)

P değeri = igamc(N/2, X^2(obs)/2) (4.25)

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.9. Evrensel Test

Dizinin veri kaybı olmadan ne kadar sıkıştırılabileceğini inceler (National Institute of Standard and Technology, 2001).

L : Her bir bloğun büyüklüğü.

Q : Başlangıç dizisindeki blokların sayısı.

n : Bit dizisinin uzunluğu.

***fn* :** Eşleşen L-bitlik şablonlar arasındaki mesafelerin *log2* tabanında toplamı

ε : Teste tabi tutulan bit dizisi

ε = $\varepsilon_1\varepsilon_2$…….. ε_n

K : Test edilen blokların sayısı

T_j : Her bir L-bitlik bloğun blok sayısını tutan j' ye bağlı tablo değeri

sum : K bloklarında tespit edilen farklılıkların log2 tabanında toplamı

σ : Standart sapma.

c : Sezgisel yaklaşım.

$n \geq (Q + K) \times L$

$6 \leq L \leq 16$

$Q = 10 \bullet 2^{L}$

$K = \overline{N/L} - Q \approx 1000 \times 2^{L}$

L, Q ve n değerleri aşağıdaki tabloya göre seçilir.

Tablo 4.6. Blok içerisindeki L, Q, n değerleri

N	L	$Q=10\bullet2^{L}$
≥387.840	6	640
≥904.960	7	1.280
≥2.068.480	8	2.560
≥4.654.080	9	5.120
≥1.342.400	10	10.240
≥22.753.280	11	20.480
≥49.643.520	12	40.960
≥107.560.960	13	81.920
≥231.669.760	14	163.840
≥496.435.200	15	327.680
≥1.059.061.760	16	655.360

$f_n = 1/K \sum_{i=Q+1}^{Q+K} log_2 (i - T_j) = sum/K$ (4.26)

P değeri = $erfc(|f_n\text{-BeklenenDeger(L)}|/\sqrt{2}\sigma)$ (4.27)

Beklenen değer (L) aşağıdaki Tablo 4.6' da elde edilir.

Tablo 4.7. Beklenen değer sonuçları

L	Beklenen Değer	Varyans
6	5.2177052	2.954
7	601962507	3.125
8	7.1836656	3.238
9	8.1764248	3.311
10	9.1723243	3.356
11	10.170032	3.384
12	11.168765	3.401
13	12.168070	3.410
14	13.167693	3.416
15	14.167488	3.419
16	15.167379	3.421

$\sigma = c\sqrt{Varyans(L)/K}$ (4.28)

$c = 0.7\text{-}0.8/L+(4+32/L)xK^{3/L}/15$ (4.29)

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.10. Doğrusal Karmaşıklık Testi

Bit dizisinin LFRS (linear feedback shift register) uzunluğuna bakarak kompleksliğini inceler. Test dizinin rassallık için yeterince karmaşık olup olmadığını kontrol eder. Diziler LFSR çıktıları olarak kabul edilir ve diziyi oluşturabilecek en küçük LFSR' ın boyu küçükse, dizinin rassal olmak için yeterince karmaşık olmadığına karar verilir. Testte dizi *M* bitlik bloklara ayrılır ve bloktaki bitlerin lineer karmaşıklıkları Berlekamp-Massey algoritması kullanılarak hesaplanır. Hesaplanan lineer karmaşıklıkların beklenen dağılıma uygun olup olmadıklarına bakılır. Testte kullanılan referans dağılım ki-kare dağılımıdır. Testin geçerli olabilmesi için dizinin boyu en az 1.000.000; blok uzunluğu da 500 ve 5000 arasında olmalıdır (Büyüksaraçoğlu, 2011).

M: Bloktaki bit uzunluğu.

N: M bitlik bağımsız blok sayısı

n: Bit dizisinin uzunluğu. n= M.N

K: Serbestlik derecesi. Test kodunda K = 6 olarak alınmıştır.

Ti : Alt dizi sayısı

$\mu = M/2 + (9+(-1)^{M+1})/36 - (M/3+2/9)/2^M$ (4.30)

$T_i = (-1)^M x(L_i - \mu) + 2/9$ (4.31)

Ti değerleri için v0 ,…,v6 değerleri şu şekilde hesaplanır:

Ti ≤ -2.5 olduğunda v_0 1 arttırılır.

-2.5 < Ti ≤ -1.5 olduğunda v_1 1 arttırılır.

-1.5 < Ti ≤ -0.5 olduğunda v_2 1 arttırılır.

-0.5 < Ti ≤ 0.5 olduğunda v_3 1 arttırılır.

0.5 < Ti ≤ 1.5 olduğunda v_4 1 arttırılır.

1.5 < Ti ≤ 2.5 olduğunda v_5 1 arttırılır.

Ti > 2.5 olduğunda v_6 1 arttırılır.

$X^2(obs) = \sum_{i=0}^{K}(v_i - N\pi_i)^2/N\pi_i$ (4.32)

P değeri = igamc (K/2, X^2(obs)/2) (4.33)

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.11. Seri Testi

Tekrar eden m bitlik 2m tane bloğun tekrar sayısının dağılımını inceler. m=1 için, birinci teste denktir. $\nabla\psi_m^2$ (*obs*) ve $\nabla^2\psi_m^2$ (*obs*) değerleri m-bitlik öbeklerin frekansını hesaplamak için kullanılır.

$\Psi_m^2 = 2^m/n \sum_{i_1 \ldots i_m} v_{i_1 \ldots i_m}^2 - n$ (4.34)

$\Psi_{m-1}^2 = 2^{m-1}/n \sum_{i_1 \ldots i_{m-1}} v_{i_1 \ldots i_{m-1}}^2 - n$ (4.35)

$\Psi_{m-2}^2 = 2^{m-2}/n \sum_{i_1 \ldots i_{m-2}} v_{i_1 \ldots i_{m-2}}^2 - n$ (4.36)

$\nabla\Psi_m^2 = \Psi_m^2 - \Psi_{m-1}^2 \quad \nabla^2\Psi_m^2 = \Psi_m^2 - 2\Psi_{m-1}^2 + \Psi_{m-2}^2$ (4.37)

P değeri 1= igamc (2^{m-2}, $\nabla \Psi_m^2$) (4.38)

P değeri 2 = igamc (2^{m-3}, $\nabla^2 \Psi_m^2$) (4.39)

Üretilen bit dizisi;

$\boldsymbol{\varepsilon}$ =10110110101101100010101010101011100111000110110001010001100100110101
01001101010010101100110011100110

$v_{000} = 4$ $v_{001} = 12$ $v_{010} = 18$ $v_{011} = 15$

$v_{100} = 12$ $v_{101} = 20$ $v_{110} = 15$ $v_{111} = 2$

$v_{00} = 16$ $v_{01} = 32$ $v_{10} = 33$ $v_{11} = 18$

$\Psi_3^2 = 2^3/100(16+144+324+225+144+400+225+4)-100 = 18.56$

$\Psi_2^2 = 2^2/100(256+1024+1089+324)-100 = 7.72$

$\Psi_1^2 = 2^1/100(2401+2601)-100 = 7.72$

$\nabla\Psi_3^2 = 10.84$ $\nabla^2\Psi_3^2 = 3.16$

P değeri 1 = igamc (2,10.84) = 0.0002319

P değeri 2 = igamc (1,3.16) = 0.04242574

P değeri 1 < 0.01 olduğundan dizi rastsal kabul edilemez.

P değeri 2 > 0.01 olduğundan dizi rastgele kabul edilebilir (Büyüksaraçoğlu, Buluş, 2009).

4.2.12. Yaklaşık Entropi Testi

Tekrar eden *m* ve *(m+1)* bitlik blokların entropisini inceler. Bu test, *m* bitlik kesişen blokların frekansları üzerine odaklaşır ve bu frekansları *m* ve *(m+1)* bitlik bloklar için beklenen değerler ile karşılaştırır. Testte kullanılan referans dağılımı ki-kare dağılımıdır. Diziden kesişen *n* tane *m*-bitlik blok üretilir. Bu blokların frekansları ve entropisi hesaplanır. Aynı işlemler blok uzunluğu *m+14* için tekrarlanır. *m* ve *m*+ 1 bit için hesaplanan değerlerin farkına bağlı test istatistiği hesaplanır. Bu farkın düşük olması rastgelelikten uzaklığı gösterir.
m: Her bir blok uzunluğu.

n: Tüm bit dizisinin uzunluğu.

$\boldsymbol{C_i^m}$ **:** Her bir i değeri için hesaplanan m bitlik blok sayısı

$ApEn(m) = \varphi^{(m)} - \varphi^{(m+1)}$ (4.40)

$C_i^m = \#i/n$ (4.41)

$\varphi^{(m)} = \sum_{i=0}^{(2^m-1)} \pi_i log \pi_i$ (4.42)

$\pi_i = C_j^3$ (4.43)

j = log_2^i (4.44)

(m+1) bitlik bloklar için de bir önceki denklem değerleri hesaplanır.

$X^2 = 2n\ [\log_2\text{-ApEn}(m)]$ (4.45)

P değeri = igamc (2^{m-1}, $X^2/2$) (4.46)

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.13. Birikimli Toplamlar Testi

Elimizdeki diziyi bloklara ayırıp bu blokların 1 ve 0 denge oranını belirler ve bloklar arasındaki dengesizlik farkına bakar. Blok içerisindeki kullanılması gereken mod formülleri Tablo 4.7' de verilmiştir.

n: Bit dizisinin uzunluğu

mod: Test uygulaması dizinin başından sonuna doğru yapılırsa mod= 0, sondan başa doğru yapılırsa mod = 1' dir.

S_i : Artarak büyüyen alt dizilerin toplamı

Tablo 4.8. Blok içerisindeki kullanılması gereken mod formülleri

***Mod* = 0**	***Mod* = 1**
$S_1 = X_1$	$S_1 = X_n$
$S_2 = X_1 + X_2$	$S_2 = X_n + X_{n-1}$
$S_3 = X_1 + X_2 + X_3$	$S_3 = X_n + X_{n-1} + X_{n-2}$
*	*
*	*
$S_k = X_1 + X_2 + \ldots + X_k$	$S_k = X_n + X_{n-1} + \ldots + X_{n-k+1}$
*	*
*	*
$S_n = X_1 + X_2 + \ldots + X_n$	$S_n = X_n + X_{n-1} + \ldots + X_{k-1} + \ldots + X_1$

$z = \max_{1 < k < n} |S|_k$

Φ = Normal birikimli olasılık dağılım fonksiyonu = $1/\sqrt{2\pi} \int_{-\infty}^{z} e^{-u^2/2}\, du$ (4.47)

P değeri = $1\sum_{k=(-\frac{n}{z}+1/4)}^{(\frac{n}{z}-1/4)}[\theta\left(\frac{(4k+1)z}{\sqrt{n}}\right)-\theta\left(\frac{(4k-1)z}{\sqrt{n}}\right)]+\sum_{k=(-\frac{n}{z}-3/4)}^{(\frac{n}{z}-1/4)}[\theta\left(\frac{(4k+3)z}{\sqrt{n}}\right)\theta\left(\frac{(4k+1)z}{\sqrt{n}}\right)]$

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

4.2.14. Rastgele Farklılık Testi

Elimizdeki diziyi bloklara ayırıp bu blokların 1 ve 0 denge oranını belirler ve daha sonra blokların dengesinin dağılımını inceler. Blok içerisindeki döngülerin bulunma sayısı Tablo 4.8' de verilmiştir.

n: Bit dizisinin uzunluğu

X: Bit dizisindeki 0 ve 1' lerin toplamı (0= -1, 1= 1 alınarak yapılan aritmetik toplama sonucu).

S_i : Her defasında X_1' den başlanarak adım adım arttırılan kısmi toplamlar.

$$
\begin{aligned}
&S_1 = X_1 \\
&S_2 = X_1 + X_2 \\
&. \\
&. \\
&S_k = X_1 + X_{2+\ldots+} X_k \\
&. \\
&S_n = X_1 + X_2 + \ldots + X_k + \ldots + X_n
\end{aligned}
$$

S' : Oluşturulan S dizisinin başına ve sonuna 0 eklenerek oluşturulan yeni alt dizi

J : S' dizisinde geçen sıfırların sayısıdır. Aynı zamanda S'' deki döngü sayısıdır. Döngü sayısı sıfır ile başlayan ve biten dizilerin sayısına bağlıdır.

x : Her bir döngü ve sıfır içermeyen durum sayısı $-4 \leq x \leq -1$ ve $1 \leq x \leq 4$

v (x) k : k' ya bağlı x koşulunun tüm döngülerdeki toplam sayısı. k = 0, 1, .., 5 için

$$\sum_{k=0}^{5} v_k(x) = J \qquad (4.48)$$

$$X^2(\text{obs}) = \sum_{k=0}^{5}(v_k(x) - J\pi_k(x))^2 / J\,\pi_k(x) \qquad (4.49)$$

$\pi_k(x)$: x koşulunun rastsal bir dağılımda k kez oluşma durumudur.

P değeri = igamc (5/2, $x^2/2$)

P değeri $\geq$ 0.01 olduğunda dizi rastgele kabul edilir.

ε = 0110110101

X= -1,1,1 -1,1,1 -1,1 -1,1

S= {-1,0,1,0,1,2,1,2,1,2}

S'= 0, -1,0,1,0,1,2,1,2,1,2,0

J = 3 {0,-1,0}, {0,1,0} , {0,1,2,1,2,1,2,0}

Tablo 4.9. Blok içerisindeki döngülerin bulunma sayısı

X	Döngü Sayıları					
	0	**1**	**2**	**3**	**4**	**5**
-4	3	0	0	0	0	0
-3	3	0	0	0	0	0
-2	3	0	0	0	0	0
-1	2	1	0	0	0	0
1	1	1	0	1	0	0
2	2	0	0	1	0	0
3	3	0	0	0	0	0
4	3	0	0	0	0	0

$v_0(-1)$ = 2 (-1 durumu 0. döngüde 2 kez gerçekleştiği için)

$v_1(-1)$ = 1 (-1 durumu 0. döngüde 1 kez gerçekleştiği için)

$v_2(-1) = v_3(-1) = v_4(-1) = v_5(-1) = 0$

4.2.15. Rastgele Farklılık Varyans Testi

Bit dizisini ardışık uzunluklu bloklara ayırıp blokların 1 ve 0 dengesini belirleyip ortalama değerden sapma miktarını belirler. Bir önceki rastgele farklılık testi ile aynı

stratejiyi kullanır.

ξ (x) : Tüm J döngülerindeki x durumlarının meydana gelme sayısı

P değeri = erfc ($|\xi(x)$ -J$|/\sqrt{2J(4|X|-2)}$) (4.50)

P değeri ≥ 0.01 olduğunda dizi rastgele kabul edilir.

5. FPGA VE PROGRAMLAMA TEKNİKLERİ

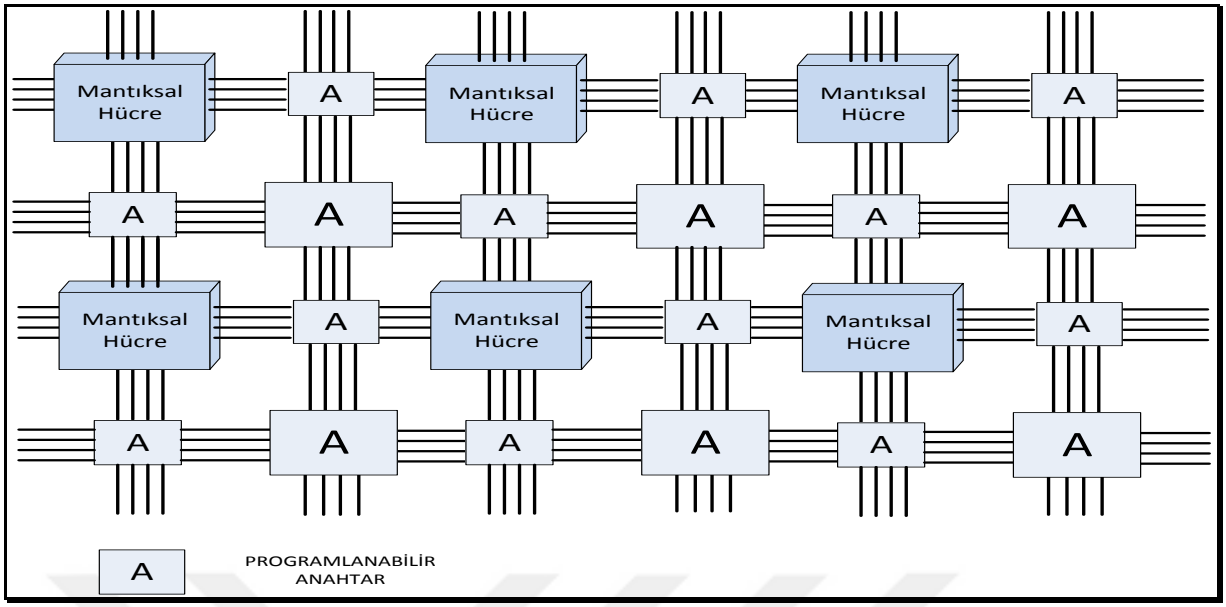
5.1. FPGA Hakkında Genel Bilgi ve Tarihsel Gelişim

FPGA programlanabilen elemanlardan, arabirimlerden oluşur. Üzerindeki programlanabilir eleman ve arabirimler lojik işlemlerini, dekoder, multiplekser gibi işlemleri uygulamak için tasarlanılabilir. FPGA yapısında yer alan ara birimler elektriksel olarak programlanır ve istenildiği kadar programlanabilme özelliğine sahiptir. Bu özelliği sayesinde tasarımlarda kolaylık sağlar.

Günümüze kadar gelen programlanabilen birimlerden bahsedecek olursak ilk olarak Programmable Read Only Memory (PROM)' ların ilk programlanabilir birim olduğu ve sadece bir kez üzerinde programlama işlemi yapılabildiği görülür. PROM' un ardından geliştirilen programlama birimleri Erasable Programmable Read Only Memory (EPROM) ve Electrically Erasable Programmable Read Only Memory (EEPROM) bunlardır. Bu birimler cok az yeteneğe sahiptirler ve basit lojik birimleri sadece bulunur yani ihtiyaca çok fazla cevap vermez. EEPROM' lardan sonra daha gelişmiş şekliyle Programmable Array Logic (PAL) ve Programmable Logic Array (PLA) yapıları geliştirilmiştir. Bu geliştirilen PAL ve PLA programlanabilen birimleri çarpım veya toplam şeklinde ifade edilebilir. PLA ve PAL birimlerinden sonra bu yapılar bir araya getirilip geliştirilerek CPLD yapısı oluşturulmuştur. CPLD' lerin avantaj ve dezavantajları vardır. FPGA' ye oranla avantajlarından biri olarak CPLD' ler FPGA' ye kıyasla programlara daha hızlı tepki verirken dezavantajına örnek olarak CPLD’ ler FPGA yapısına kıyasla daha küçük lojik birimlere sahip olması nedeniyle küçük tasarımlarda kullanılır.

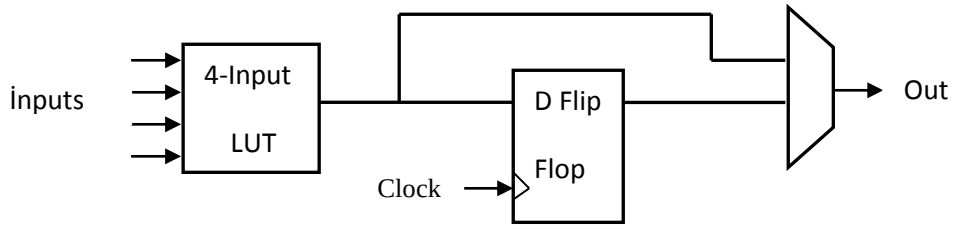
5.1.1. FPGA' nın Genel Yapısı

Tipik bir FPGA yapısı güncellenilebilen CLB, yönlendirilebilir kanallardan oluşur. FPGA genel yapısı Şekil 5. 1' de verilmiştir.

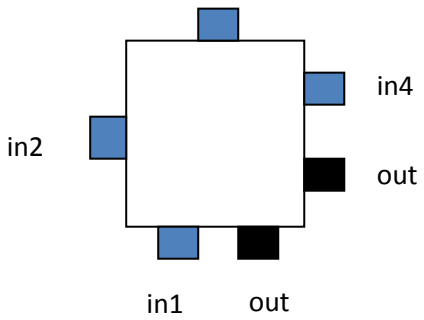


Şekil 5. 1. FPGA genel yapısı (Yıldırım, 2012).

Bu yapıları biraz inceleyelim. CLB yapısına değinecek olursak 5 girişten oluşurlar, 4 ü LUT (look-up table) girişleri iken diğer giriş ise D flip flop yapısının saat girişidir. CLB yapısı Şekil 5.2 ve 5.3' te verildiği gibidir.



Şekil 5.2. CLB genel yapısı (Doğan, 2006).



Şekil 5.3. CLB giriş/çıkışlarının donanım üzerindeki görünümü (Doğan, 2006).

CLB yapısının sahip olduğu anahtarlar ile yapılmasını istediğimiz tasarım çalışmalarını gerekli anahtarların bağlanması ile gerçekleştirilebiliriz.

5.1.2. FPGA Programlama

FPGA yapısının çalışması için öncelikli olarak programlanması gerekir. Programlanması için donanım dilleri veya tasarımlarını kullanır. Programlama dillerine örnek olarak VHDL veya Verilog-HDL dilleri verilebilir. FPGA yapısını programlarken kullandığımız FPGA' nın hangi üreticiye ait olduğu çok önemlidir. Çünkü üretici firma olarak farklılık gösterebilir. FPGA üzerinde bir uygulama gerçekleştireceğimiz zaman ilk olarak HDL dillerinden biri ile kodlama veya tasarım yapılır ve hata kontrolünü yapmak için gerekli simülasyonlar yapılır ve simülasyonlar sonucunda tasarım istediğimiz şekilde çalışıyorsa FPGA' ya bu programı aktarma kısmına geçeriz. Aktarma kısmında gerekli lojik birimlerin FPGA' da tanımlanması yapılırken kullanmış olduğumuz firma aracına bağlı bağlantı kurulur. Bağlantılar kurulduktan sonra tekrar simülasyondan geçirilir ve başarıyla sonuçlanılır ise devrede istediğimiz sonucu görürüz. Bu tez çalışmasında kullandığım rastgele sayı üreteci Trivium olup üretici firma Xilinx olup Virtex-5 ürünüdür.

5.2. VHDL Dili Yapısı ve Özellikleri

VHDL, donanımsal olarak gerçekleştirilecek sistemlerin nasıl çalışacağını tanımlayan, yönlendiren programlama dilidir. VHDL, VHSIC donanım tanımlama dilinin kısaltılmış şeklidir. VHSIC, 1980’ lerde Amerika Birleşik Devletleri Savunma Bakanlığı tarafından finanse edilen bir girişim olup VHDL dilinin ortaya çıkmasına öncülük etmiştir. VHDL, IEE1076 standardı ile IEEE tarafından standartlaştırılmış ilk ve orijinal donanım tanımlama dilidir.

VHDL dilinin tasarlanma amacı devre sentezi ve devrenin simülasyonu sağlamaktır. Bu dilde kulladığımız tüm kodlar içerisinde sentezlenemeyen kodlarda vardır. Bunların iyi bilinmesi gerek çünkü istediğimiz simülasyon sonucunu almamızı engeller. VHDL dilinin istenilen yere taşınılması ve tekrar tekrar kullanılabilmesi avantajlardan birkaçıdır. Programlanabilir Lojik Devreler (CPLD-FPGA) ve ASIC’ lerde VHDL dilinin iki farklı uygulaması mevcuttur. VHDL kodu yazıldıktan hemen sonra bu kodlar, Altera, Xilinx,

Atmel vb. gibi firmaların ürettiği programlanabilir cihazlar üzerinde tasarlanan devreyi gerçekleştirmek için kullanılır ya da ASIC üretimi için bu tür uygulamaya özgü üretimler yapan firmalara kendi tasarımlarını gerçekleştirmek üzere sunulur.

VHDL, C/C++, Pascal vb sıradan bilgisayar programlama dillerine benzemez. Kodlama sırasında çeşitli farklılıkları vardır bunlardan biri VHDL paralel çalışırken diğer bilgisayar programlama dilleri seri çalışır. VHDL sadece PROCESS, FUNCTION veya PROCEDURE deyimleri içerisindeki kodlar ardışıktır.

5.2.1. VHDL Tasarım Yapısı

VHDL tasarımını gerçekleştireceğimiz zaman bilmemiz gereken üç bölümü vardır bunlar entity (varlık), architecture (mimari) ve library (kütüphane) dir.

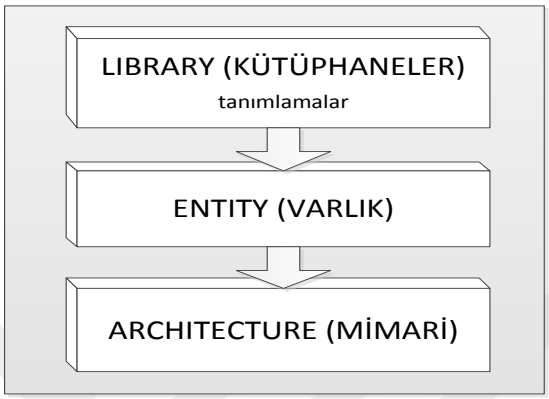
- ***Entity (varlık):*** Giriş/çıkış portlarını tanımladığımız kısımdır.
- ***Architecture (mimari):*** Kodlamayı yaptığımız, iç mimarisini tanımladığımız kısımdır.
- ***Library (kütüphane):*** VHDL içerisindeki kodların başarılı bir şekilde çalışılabilmesi için gerekli kütüphanelerin kullanılması zorunludur yoksa sistem hata verir ve istenilen sonuç alınamaz.

Genel bir VHDL dosyasının şablonu Şekil 5.4' de gösterilmiştir.

```
--Genel bir VHDL dosya yapısı
LIBRARY <Kütüphane adı>;
USE <Kütüphane_adı>.<alt_kütüphane_adı>.ALL
ENTITY <varlık_adı> IS
PORT (
<port_adı>: <port_yönü>  <port_tipi>;
<port_adı>: <port_yönü>  <port_tipi> );
END <varlık_adı>
ARCHITECTURE <mimari_adı> OF <varlık_adı> IS
SIGNAL <sinyal_adı> : <sinyal_tipi> := "<başlangıç değeri>";
```

Şekil 5.4. Genel bir VHDL dosya şablonu (Yıldırım, 2012).

VHDL tasarımında sadece bir tane varlık tanımlanması vardır bunun yanında çok sayıda mimari kullanabiliriz. Fakat en fazla tercih edilen kod yapısı hem basit hem de daha görsel olduğundan bir varlık bir tane de mimarinin bulunmasıdır. Basit anlamda VHDL kod yapısını gösteren hiyerarşi Şekil 5.5' de gösterildiği gibidir.



Şekil 5.5. VHDL kod tasarım bölümleri (Yıldırım, 2012).

6. TRIVIUM ALGORİTMASI VE GERÇEKLENMESİ

6.1. Trivium Dizi Şifreleme Algoritması

Trivium sözde rastgele sayı üreteci donanım bazlı senkron, akış şifreleme sistemidir. Tohum değerini özgün veriden bağımsız ürettiğimiz için senkron bir yapıya sahiptir.

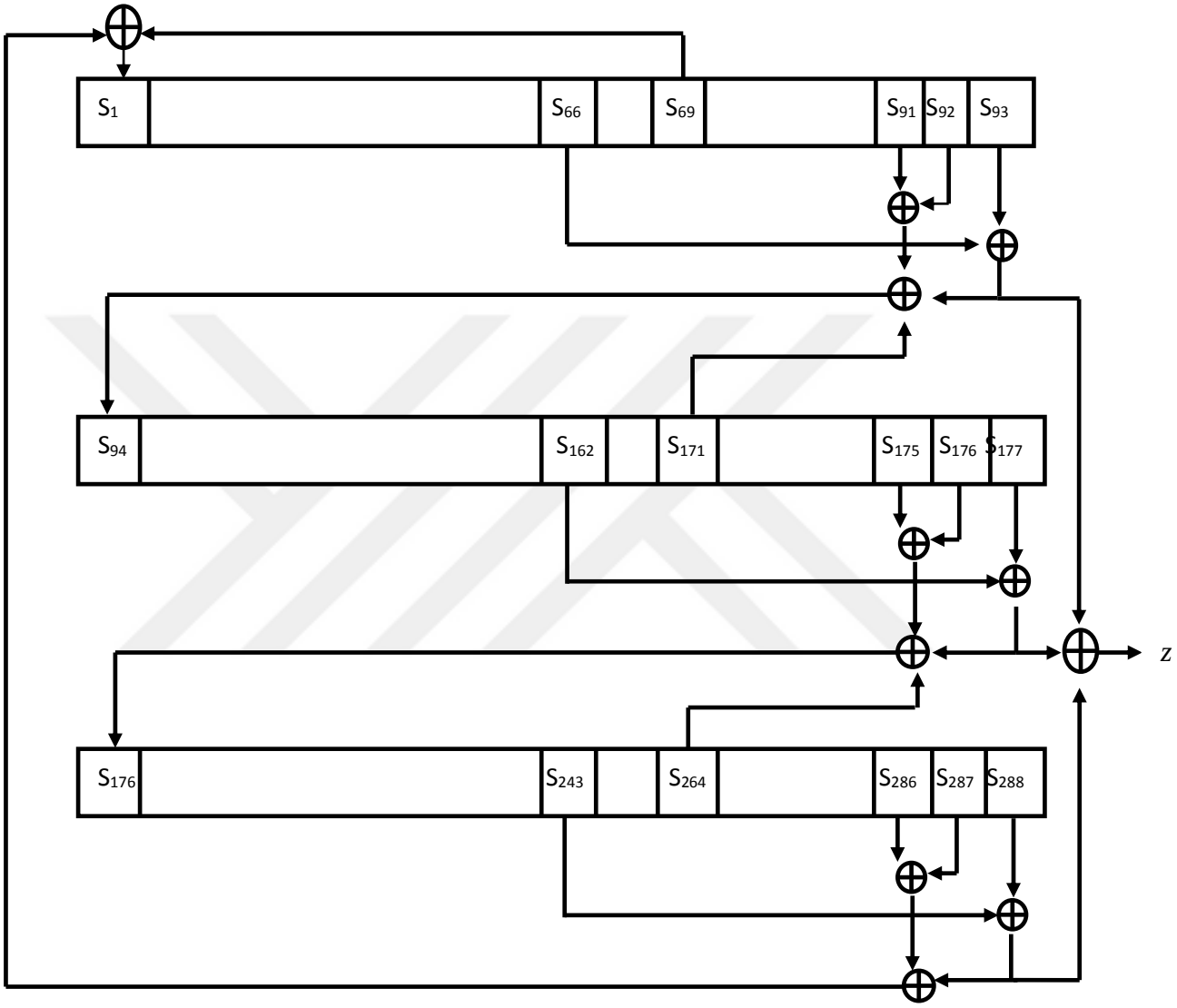
Trivium Dizi Şifreleme Algoritması' nın üretilmesini gerekliliklerinden bazıları hız ve rahat güncelleme işlemlerinin sağlanılmasıdır. Hız ve alan arasında esnek bir optimizasyon sağlamaktır. Trivium algoritması; güvenlik, hız ve esnekliğe önem vererek akış şifreleme sisteminin basitleştirilebileceğini araştırmak için üretilmiştir (De Canniere, 2005). Trivium rastgele sayı üreteci, 80 bitlik gizli anahtar (tohum değer) ve 80-bitlik başlangıç vektörü (IV) ile 2^{64} bite kadar anahtar dizisi üretebilen senkron bir dizi şifreleme sistemidir. Trivium rastgele sayı üretirken ilk olarak tohum değeri ve başlangıç vektörü seçilir tabiki tohum değeri özellikle seçerken karmaşıklığının kaliteli oluşuna dikkat etmeliyiz. Daha sonra başlangıç durumu oluşturma aşamasına geçilir. Daha sonra iç durum güncelleme aşamasına geçilir bu aşamada LFSR yapısı kullanılır. Anahtar bitleri kaydırılarak dört tam tur 1152 defa döndürülerek güncellenir. Güncel iç durum oluşturulduktan sonra ise istenilen sayıda anahtar bitlerini üretme aşamasına geçirilir (Kaplan, 2005). Tablo 6.1' de Trivium parametrelerinin aldığı uzunluk değerleri verilmiştir:

Tablo 6.1. Trivium parametrelerinin aldığı uzunluklar

PARAMETRELER	UZUNLUĞU
ANAHTAR UZUNLUĞU	80
İLK DEĞER UZUNLUĞU	80
İÇ DURUM UZUNLUĞU	288

Trivium dizi şifreleme üreteci başlangıç durumları yüklendikten sonra 288-bitlik iç durum (internal state) oluşturulur. Anahtar üretme aşamasında 15 özel bitin kullanılması bu

15 bitin her 3 bitin güncellenmesinde kullanılması ve son olarak bir bit anahtar dizisinin z_i' nin hesaplanması olaylarını kapsar. Algoritmanın iç yapısını gösteren Şekil 6.1' de iç durum işlemlerine karşılık kod yapısı Tablo 6.2' de verilmiştir.



Şekil 6.1. Trivium akış şifreleme algoritmasının donanımsal iç yapısı (URL-1, 2017)

Tablo 6.2. Algoritmanın iç durum işlemlerine karşılık sözde kod yapısı

$(s_1, s_2, \ldots \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots 0)$
$(s_{94}, s_{95}, \ldots \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots 0)$
$(s_{178}, s_{179}, \ldots \ldots, s_{288}) \leftarrow (0, \ldots .0, 1, 1, 1)$
for i = 1 to 4*288 do
$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$
$t_3 \leftarrow s_{243} + s_{288}$
$z_i \leftarrow t_1 + t_2 + t_3$
$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
end for

İç durum da yaptığımız işlemlerin aynısını anahtar biti üretmek için istenilen sayıda tekrarlı biçimde yapılmaya devam edinilerek üretme işlemi yapılır. Lojik düzeyde her bir anahtar bit çıkışı için iç durum güncellemesi ilk aşamada olduğu gibi tekrarlı bir biçimde istenen sayıda anahtar biti için yapılmaya devam edilir.

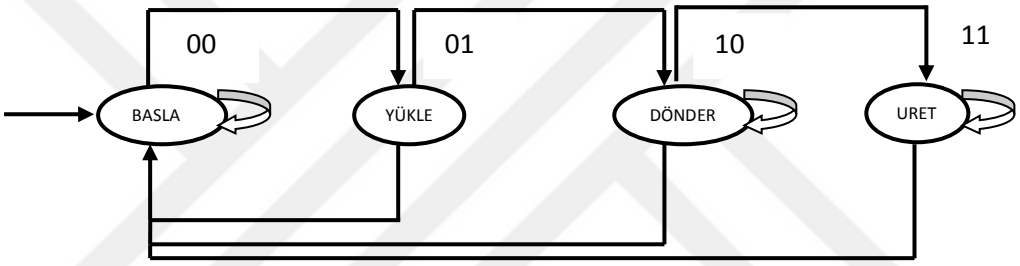
6.2. Gerçekleme Adımları

Trivium rastgele sayı üretecinin donanım üzerinde gerçeklenmesi için yapacağımız iş VHDL dilini öğrenmektir. VHDL dilini öğrenirken derslere katılıp çeşitli kitap ve sitelerden faydalanılarak çalışılmıştır. Daha sonra VHDL dili kullanılarak kodlanmış Trivium' a benzer rastgele sayı üreteçleri incelenilmiştir. Daha sonra bu algoritmalar hakkında geniş bilgi kazanılıp Trivium algoritmasının kodlanma aşamasına geçilmiştir. Trivium algoritmasının tasarım özeti Tablo 6.3' de verilmiştir.

Tablo 6.3. Hazırlanan sistemin tasarım özeti

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	426	12480	3%
Number of Slice LUTs	544	12480	4%
Number of fully used LUT-FF pairs	409	561	72%
Number of bonded IOBs	131	172	76%
Number of BUFG/BUFGCTRLs	1	32	3%

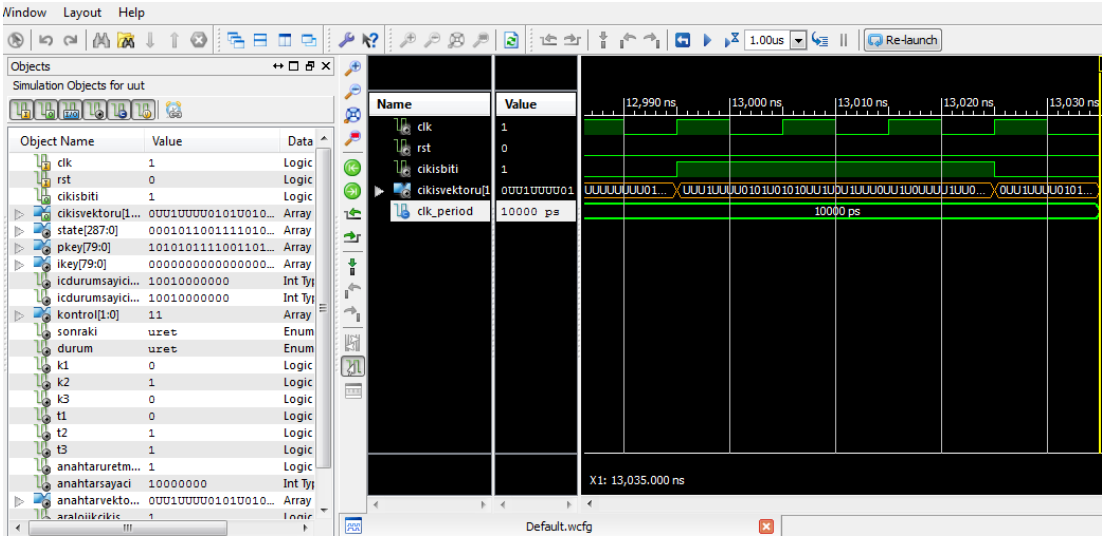
VHDL dili , Trivium rastgele sayı üreteci ve diğer rastgele sayı üreteçleri hakkında bilgi sahibi olunduktan sonra VHDL dilinin kullanılacağı programlar hakkında bilgi sahibi olunmuştur. Daha sonra bu programlardan Xilinx ISE programının kullanılmasına karar verilmiştir. Xilinx ISE programı üzerinde yapılan birçok çalışmalar incelenilip bu program hakkında daha geniş bilgi edinilmiştir. Bu program öğrenildikten sonra algoritmayı yazmaya uygun duruma gelinilmiştir. Trivium algoritmasında başlangıç olarak 128 bitlik LFSR yapısı tanımlanmıştır. VHDL kodu içerisinde FSM yapısı kullanılarak durumlar arası geçiş sağlanmıştır. FSM yapısı olarak iki bitlik kontrol değişkeni tanımlanmıştır. "00" değeri için "basla", "01" değeri için "yükle", "10" değeri için "dönder", "11" değeri için "üret" durumlarında olup bu kontrol değişkenleri sırası ile yapılacaktır. Şekil 6.2' te FSM yapısı gösterilmektedir.



Şekil 6.2. Dört durumlu FSM modeli

6.3. Algoritmanın Doğrulanması

FSM olarak modellenen Trivium akış şifreleme algoritması iki aşamada doğrulanmıştır. İlk aşamada FPGA bordu üzerine yüklenmeden önce mantıksal hataların önüne geçmek ve modelin doğru davranış sergilediğini görebilmek için Virtex 5 simülasyon aracı kullanılmıştır. Algoritma ilk olarak simülasyon ortamında 128 bitlik anahtar dizisi için çalıştırılmıştır. Kullanılan lojik seviyedeki giriş, çıkış ve sayaç ve kontrol sinyalleri ile durumların zamana bağlı değişimleri izlenerek mantıksal doğrulaması yapılmıştır. Simülasyon sonuçları Şekil 6.3' teki gibidir. Gerçekleme aşamasında, mantıksal doğrulaması yapılan algoritmaya ait kodlar lojik düzeyde sentezlenerek, yerleştirme ve yollandırma işlemlerinin ardından FPGA donanımı üzerine yüklenerek, anahtar dizilerine ait lojik çıkışlar gözlemlenmiştir.



Şekil 6.3. Trivium algoritmasının simülasyon sonucu

Çıkış Vektörü= 10111000010111010111011010011110000100100111111000000100010110010111001101010001010
01100011111001110000110011101010010100000011110

Doğrulamanın ikinci aşamasında rastgele sayılardan oluşan anahtar dizilerinin gerçek bir rastgele diziden beklenen yeterlilikleri karşılayıp karşılamadığına bakmak için teste tabi tutulmuştur. Bu konuda pek çok test (FIBS 140, DieHard, NIST vs.) bulunmasına rağmen bu çalışma kapsamında hipotez tabanlı NIST 800-22 istatistiksel test süiti kullanılmıştır. Diğer testlere oranla daha güçlü bir yapı içeren NIST 800-22 yazılım süiti rastgelelik ölçümü için kendi içerisinde 15 ayrı kritere ait testten oluşmaktadır. NIST testinden geçirilen bir dizinin geçerli olması için bu kriterlerden geçmesi gerekir. Bu çalışmada üretilen diziler NIST testinden geçirilmiş ve elde edilen sonuçların başarılı olduğu görülmüştür. Anahtar dizisine ait test sonuçları Tablo 6.4' de verilmiştir.

Tablo 6.4. Anahtar dizisine ait test sonuçları

Test	P Değer	Sonuç
Frekans Testi	0.576	Başarılı
Blok Frekans Testi	0.709	Başarılı
Akış Testi	0.084	Başarılı
Blok İçindeki En Uzun Birler Testi	0.946	Başarılı
İkili Matris Rankı Testi	0.327	Başarılı
Ayrık Fourier Dönüşümü Testi	0.720	Başarılı
Örtüşmeyen Şablon Eşleştirme Testi	0.532	Başarılı
Örtüşen Şablon Eşleştirme Testi	0.466	Başarılı

Maurer’in Evrensel İstatistik Testi	0.129	Başarılı
Doğrusal Karmaşıklık Testi	0.907	Başarılı
Seri Testi 1	0.193	Başarılı
Seri Testi 2	0.552	Başarılı
Yaklaşık Entropi Testi	0.387	Başarılı
Kümülatif Toplam Testi	0.616	Başarılı

7. SONUÇLAR

Yapılan bu tez çalışmasında rastgele sayı üreteçleri konusunda Avrupa Birliği Standardında olan Trivium algoritması kullanılmıştır. Trivium algoritmasının kodlanabilmesi için VHDL dili kullanılmış ve 128 bitlik rastgele sayılar üretilmiştir. Daha sonra Trivium rastgele sayı üretecinin donanımsal olarak sonucunu almak için Xilinx ISE programı kullanılmıştır ve simülasyon sonuçları alınarak donanım üzerinde gözlemlenmiştir.

Trivium rastgele sayı üreteci ile ürettiğimiz 128 bitlik rastgele dizinin NIST testine tabi tutulup sonuçları gözlemlenmiştir. Bu sonuçları incelediğimizde 2 testin 0.9' u geçip çok iyi sonuç verdiği, 7 testin 0.5 ile 0.9 arasında iyi sonuçlar verdiği, 6 testin ise 0.1 ile 0.5 arasında kalıp anlamlı sonuç verdiği ve başarılı olduğu gözlemlenmiştir. Bu sonuçların iyileştirilmesi için yani daha iyi sonuçlar vermesi için gizli anahtarı mümkün olduğunca karmaşık seçmemiz gerekmektedir yada bu alanda ek çalışmalar yapılabilir.

Rastgele sayı üreteçleri üzerine yapılan kriptografik çalışmalarının yeterli düzeyde olmaması bu alanda yapılan çalışmaları kıymetli kılmaktadır. Bu tez çalışması bu alanda çalışmak isteyen, ilgi duyan, bilgi almak isteyen her kişiye rehber niteliğindedir ve yalın halde, açıklayıcı bir şekilde hazırlanılmıştır.

KAYNAKLAR

Aslan, B., 2008**.** Boole fonksiyonları ve s-kutularının kriptografik özelliklerinin incelenmesi ve ters haritalama tabanlı cebirsel açıdan güçlendirilmiş bir s-kutusu önerisi. *Yüksek Lisans Tezi*, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne.

Avaroğlu, E., Tuncer, T., Özer, A., Türk, M., 2014. A new method for hybrid pseudo random number generator. *MIDEM Journal of Microelectronics, Electronic Components and Materials*, 4:303 – 311.

Avaroğlu, E., Türk, M., 2014. Son işlemin gerçek rastgele sayı üreteçleri üzerindeki etkisinin incelenmesi. *MIDEM Journal of Microelectronics, Electronic Components and Materials*, 4: 303 – 311.

Bellare, M., Rogaway, P., 2017. Introduction to modern cryptography. http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf, 12 Mart 2017

Büyüksaraçoğlu, F., 2011. Akış şifrelerin tasarım teknikleri ve güç analizi. *Doktora Tezi*, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne.

Büyüksaraçoğlu, F., Buluş, E., 2009. Sözde rastsal sayı üretiminin kriptografik açıdan incelenmesi. *İletişim Teknolojileri Ulusal Sempozyumu (İTUSEM)*, Adana.

Chaitin, G.J., 2001. Exploring randomness. IBM Research, United States of America.

Choong, F., Reaz, M.B.I., 2005. Implementation of power quality disturbance classifier in FPGA employing wavelet transform, ANN and fuzzy logic. *SETIT*, 27-31 Mart, Tunus.

Çavuşlu, M.A., Karakuzu, C., Şahin, S., 2006. Neural network hardware implementation using FPGA. *ISEECE 3rd International Symposium on Electrical, Electronic and Computer Engineering Symposium Proceedings*, Nicosia, TRNC, S. 287-290.

De Canniere, C., 2005. Trivium specifications. Belgium.

Diffie, W., Hellman, M.E., 1976. New directions in crytography. *IEEE Transactions on Information Theory,* 22: 644-654.

Doğan, A., 2006. SFINKS dizi şifreleme algoritmasının VHDL ile yazılımı ve FPGA üzerinde gerçeklenmesi. *Bitirme Tezi*, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Dutang, C., Wuertz, D., 2009. A note on random number generation. *Overview of Random Generation Algoritms,* September 2009.

Ekdahl, P., Johansson, T., 2003. A new version of the stream cipher SNOW. In *SAC'02*: *Revised papers from the 9th annual international workshop on selected areas in cryptography*, pp. 47-61, London, UK, Springer-Verlag.

Kaplan, T., 2005. Trivium dizi şifreleme algoritmasının VHDL ile FPGA üzerinde gerçeklenmesi. *Bitirme Tezi,* İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

Koç, Ç.K., 2009. Cryptographic engineering. Springer-Verlag.

Kuon, I., Tessier, R., Rose, J., 2007. FPGA Architecture: survey and challenges. *Foundations and trends in electronic design automation*, pp. 135-253, Canada.

Mahmoud, M.I., vd, 2007. Comparison between haar and daubechies wavelet transformions on FPGA technology. *World Academy of Science, Engineering and Technology*, 26:68-72.

National Institute of Standard and Technology., 2017. A statistical test suite for random and pseudo random number generators for cryptographic applications, NIST800-22. http://csrc.nist.gov/rng/SP800-22b.pdf, 26 Mart 2017.

Omondi, A.R., Rajapakse, J.C., 2006. FPGA implementations of neural networks. Springer, ISBN: 0387284850.

Özkaynak, F., 2015. Kriptolojik rasgele sayı üreteçleri. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği*, 8:37-45.

Rivest, R.L., Shamir, A., Adleman, L., 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of ACM,* 21:120-126.

Sakallı, B.F., 2011. Akış şifrelerin tasarım teknikleri ve güç analizi1. *Doktora Tezi*, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne.

Seals, R., Whapshott, G., 1997. Programmable logic: PLDs and FPGAs. UK, Macmillan.

Soyalıç, S., 2005. Kriptografik hash fonksiyonları ve uygulamaları. *Yüksek Lisans Tezi*, Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Kayseri.

Torres, H.C., Arias, E.M., 2004. Real-time image processing with a compact FPGA-based systolic architecture. *Real-Time Imaging*, 3:177-187.

URL-1, 2017. www.researchgate.net/figure/251857644_fig3_Fig-3-Trivium-stream-cipher. 20 Mayıs 2017.

Yerlikaya, T., 2006. Yeni şifreleme algoritmalarının analizi. *Doktora Tezi*, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne.

Yeşilbaş, E., 2016. Cebirsel kriptoloji yöntemleri ve bazı uygulamaları. *Yüksek Lisans Tezi*, Recep Tayip Erdoğan Üniversitesi Fen Bilimleri Enstitüsü, Rize.

Yıldırım, Ö., 2012. Gerçek zamanlı akım ve gerilim sinyallerinin izlenmesi için FPGA tabanlı bir sistem tasarımı. *Doktora Semineri*, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ.

Yılmaz, N., 2008. Alan programlamalı kapı dizileri (FPGA) üzerinde bir YSA' nın tasarlanması ve donanım olarak gerçekleştirilmesi. *Yüksek Lisans Tezi*, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya.

ÖZGEÇMİŞ

13.07.1991 yılında Elazığ' da doğdum. İlk ve ortaöğrenimini Elazığ’ da tamamladım. Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Bilgisayar Mühendisliği bölümünden 2015 yılında mezun oldum. 2015 yılından bu zamana Munzur Üniversitesi Elektrik-Elektronik Mühendisliği anabilim dalında yüksek lisans öğrenimimi sürdürmekteyim.