T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DONANIM TABANLI RASGELE SAYI ÜRETECİNİN GERÇEKLEŞTİRİLMESİ

DOKTORA TEZİ

Erdinç AVAROĞLU

Anabilim Dalı: Elektrik Elektronik Mühendisliği

Programı: Telekomünikasyon

Danışman: Doç. Dr. Mustafa TÜRK

Tezin Enstitüye Verildiği Tarih: 20 Ocak 2014

ŞUBAT-2014

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DONANIM TABANLI RASGELE SAYI ÜRETECİNİN GERÇEKLEŞTİRİLMESİ

DOKTORA TEZİ

Erdinç AVAROĞLU

(08213201)

Tezin Enstitüye Verildiği Tarih : 20 Ocak 2014
Tezin Savunulduğu Tarih : 12 Şubat 2014

Tez Danışmanı :	**Doç. Dr. Mustafa TÜRK (F.Ü)**
Diğer Jüri Üyeleri :	**Doç. Dr. Ali KARCI (İnönü Ü)**
	Doç. Dr. A.Bedri ÖZER (F.Ü)
	Doç. Dr. Arif GÜLTEN (F.Ü)
	Yrd. Doç. Dr. Turgay KAYA (F.Ü)

ŞUBAT-2014

ÖNSÖZ

Tez çalışmam boyunca bilgi ve birikimini esirgemeyen, tezimin tüm aşamalarında yardımlarını gördüğüm değerli hocam Doç.Dr. Mustafa TÜRK'e içten teşekkürlerimi sunarım.

Ayrıca doktoraya başlamama vesile olan, doktoramı yaparken karşılaştığım zorluklarda yanımda olarak tüm desteğini veren ve bu çıktığım yolda sona gelmemi sağlayan, tüm bilgi ve birikimini benden esirgemeyen değerli hocam Doç.Dr. Ahmet Bedri ÖZER'e en içten teşekkürlerimi sunarım.

Doktora çalışmalarım sırasında bilgi ve birikimlerini benle paylaşan ve bana destek olan Arş.Gör.Dr. Fatih ÖZKAYNAK ve Yrd.Doç.Dr. Taner TUNCER hocama teşekkürlerimi sunarım.

Yaşamım ve eğitimim boyunca yanımda olan ve desteklerini eksiltmeyen doktora süresi boyunca tüm kahrımı çeken Eşim Nuray AVAROĞLU'na, canım oğullarım Çağan Halil ve Eymen'e, babam Halil AVAROĞLU ve annem Emine AVAROĞLU'na çok teşekkür ederim.

Erdinç AVAROĞLU
ELAZIĞ - 2014

İÇİNDEKİLER

Sayfa No

ÖZET

Bu tezde, kriptografik uygulamalarda kullanılan Gerçek Rasgele Sayı Üreteci (GRSÜ) ayrıntılı olarak incelenmiş olup literatürde bulunan tasarımların birçoğu tartışılmıştır.

Tez çalışmasında ilk olarak GRSÜ'de son işlemin önemi incelenmiştir. Son işlem, gerçek rasgele sayı üreteçlerinde kullanılan entropi kaynaklarının çevresel değişikliklerden etkilenmeleri nedeniyle oluşan zayıf istatistiksel özellikler gidermek amacıyla kullanılmaktadır. İkinci bir avantajı ise yan kanal analizi saldırısına karşı sistemi dirençli hale getirmesidir. Ancak kullanılan son işlemler GRSÜ'den elde edilen çıkış bit oranını düşürmektedir. Bu amaçla çalışmada, literatürdeki Von Neumann, XOR, H son işlem gibi çeşitli son işlem algoritmalarına alternatif olabilecek, GRSÜ'nün veri oranını düşürmeden istatistikî zayıflıkları gideren yeni lojistik haritaya dayalı son işlem algoritması önerilmiştir. Elde edilen sonuçlar mevcut sonuçlar ile karşılaştırıldığında daha iyi sonuçlar elde edilebileceği gösterilmiştir. Lojistik haritanın etkilerini gözlemleyebilmek amacıyla RO tabanlı TRNG yapısı dört farklı senaryo ile gerçekleştirilmiştir. Önerilen sistem EP4CE115F29C7 tabanlı Altera FPGA (Field Programmable Gate Array - Alan Programlanabilir Kapı Dizileri) bordu üzerinde gerçekleştirilmiştir. Elde edilen sonuçlara göre lojistik haritanın son işlem olarak kullanılabileceği gösterilmiştir. Gerçekleştirilen tasarımda çıkış hızı 20Mbit/s olarak elde edilmiştir.

Diğer çalışmada ise, Saf Sözde Rasgele Sayı Üreteçlerinin (SRSÜ) güvenliğini ve rasgeliliğini arttırmak amacıyla SRSÜ'nün geçiş ve çıkış fonksiyonuna gerçek rasgele sayı dizisi ek girdi olarak eklenmiştir. Bu amaçla, AES (Advanced Encryption System - Gelişmiş Şifreleme Sistemi) kullanarak tasarlanan saf SRSÜ'lere ek girdi olarak Xilinx FPGA'de Burke Shaw kaotik çekerden elde edilen rasgele bit dizisi eklenerek Hibrit SRSÜ sistemi geliştirilmiştir.

Gerçeklenen tasarımlardan elde edilen bit dizilerinin istatistiksel olarak rasgeleliğini test etmek amacıyla National Institute of Standards and Technology (NIST) tarafından yayınlanmış olan NIST 800-22 test paketinin programı C# ortamında yazılmıştır. Elde edilen sonuçlar, gerçekleştirilen tasarımlar için karşılaştırmalı olarak verilmiştir.

Anahtar Kelimeler: Gerçek Rasgele Sayı Üreteci, Son İşlem, Halka Osilatörler, Kaotik Sistemler, İstatistikî Testler

SUMMARY

Hardware Based Realization Of Random Number Generator

In this thesis, the True Random Number Generators (TRNG) used in cryptographic applications has been examined in detail, many of the designs in the literature have been discussed.

In this thesis study, the importance of TRNG in post processing were examined. Post processing has been used for; true random number generators used in the entropy sources affected by environmental changes in order to resolve poor statistical properties. A second advantage is, it makes the system resistant against to the side-channel analysis attacks. However, the used post processing decreases the output bit rate, obtained from TRNG. For this purpose in this study, in the literature Von Neumann, XOR, H last process as well as various post-processing algorithms for an alternative, without reducing the data-rate of TRNG, post processing is proposed which is based on a new logistic map and reducing the statistical weakness. When obtained results compared with the current results; it is shown that better results could be obtained. In order to observe the effects logistic map of RO-based TRNG structure was carried out with four different scenarios. The proposed system is set on EP4CE115F29C7-based Altera FPGA (Field Programmable Gate Array) board. According to obtained test results, it was shown that logistic map can be used as post processing. The output rate was get 20Mbit/s in the performed design.

In the other study, in order to increase the safety and randomness of pure pseudo random number generator (PRNG), true random number stream is added as additional input to the transition and output functions of PRNG. For this purpose, Hybrid PRNG was developed by adding random bit streams, obtained from Burke Shaw chaotic attractor on Xilinx FPGA, as additional input the pure PRNGs which are designed using AES (Advanced Encryption System)

To test, the statistical randomness of bit streams derived from the materialized design, the National Institute of Standards and Technology (NIST), who has published the program of NIST 800-22 test suite is written in C # environment. The obtained results was given for performed designs as comparative.

Keywords: True Random Number Generator, Post Processing, Ring Oscillators, Chaotic Systems, Statistical Tests.

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

SEMBOLLER LİSTESİ

KISALTMALAR

RSÜ : Rasgele Sayı Üreteçleri

SRSÜ : Sözde Rasgele Sayı Üreteçleri

GRSÜ : Gerçek Rasgele Sayı Üreteçleri

HRSÜ : Hibrit Rasgele Sayı Üreteçleri

FPGA : Field Programmable Gate Array (Alan Programlanabilir Kapı Dizileri)

LC : Bobin Kapasitör

XOR : Exclusive Or

FIPS : Federal Bilgi İşleme Standart

NIST : Uluslararası Standartlar ve Teknoloji Enstitüsü

IV : Başlangıç Vektörü

RAM : Rasgele Erişimli Bellek

FGRSÜ : Fiziksel Gerçek Rasgele Sayı Üreteçleri

DAS : Sayısallaştırılmış Analog Sinyal

FOGRSÜ : Fiziksel Olmayan Gerçek Rasgele Sayı Üreteçleri

ASIC : Application Specific Integrated Circuit (Uygulamaya Özel Tümleşik Devre)

FPAA : Field Programmable Analog Array (Alan Programlanabilir Analog Diziler)

PSOC : Programmable System on Chip (Çip Üzerinde Programlanabilir Sistem)

SHA : Secure Hashing Algorithm (Güvenli Özetleme Algoritması)

MD : Message Diggest

DGBKY : Doğrusal Geri Beslemeli Kayan Yazmaç

PLL : Phase Locked Loop (Faz Kilitli Döngü)

HOKY : Hücresel Otomat Kayan Yazmaç

GOH : Galois Osilatör Halkası

FOH : Fibonacci Osilatör Halkası

TERO : Transition Effect Ring Oscillator (Geçiş Etkili Halka Osilatör)

TFF : T Tipi Flip Flop

Rst : Reset

Ctrl : Control

BRAM : Blok Rasgele Erişimli Bellek

PUF : Fiziksel Klonlanamayan Fonksiyon

SRAM : Statik Rasgele Erişimli Bellek

VHDL : VHSIC Hardware Description Language

VHSIC : Very High Speed Integrated Circuit

ERFC : The Complementary Error Function

IGAMC : Incomplete Complementary Gamma Function

IEEE : The Institute of Electrical and Electronics Engineers

RO : Ring Oscillator

SEMBOLLER

$K_{ö}$: Özel Anahtar

a_1, a_2, K, a_n : Olasılık deneyinin olası sonuçları

p_1, p_2, p_n : Olasılık deneyinin olasılıkları

S : Durum uzayı

$H_n()$: Shannon Ölçütü

$H_{max}()$: Maksimum entropi

$H_{max}()$: Minimum Entropi

s_n : iç durum değeri

R : Çıkış uzayı

r_n : Rasgele çıktı

$\Psi_H()$: Çıkış fonksiyonu

$\phi_H()$: Geçiş fonksiyonu

p_s : Olasılık dağılımı

e_n : Ek girdi

σ : Sistemde bulunan durumlar

V : Sistemde bulunan bölgeler

c : Çift sarmallı yapı eşik değerleri

1. GİRİŞ

Son yıllarda sayısal haberleşmede kullanılan teknolojilerde hızlı gelişmeler olmuştur. Bunların başında özellikle internet ve cep telefonlarındaki gelişim gelmektedir. Bu gelişim, insanların hayatlarında önemli değişiklikler yapmasının yanı sıra gizli ve özel olan bilgiye erişmeye çalışan saldırganları da mümkün hale getirmiştir. Bu bilgiyi korumanın çözümlerinden biride kriptografiyi kullanmaktır. Kriptografi özellikle son yüzyılda askeri alanlarda gizli bir bilginin bir yerden diğer bir yere güvenli olarak taşınması amacıyla kullanıldı. Kriptografi, veriyi şifreleme ve şifre çözme işlemeleri ile ilgilenir. Veri şifreleme, gizli bir anahtar kullanımı ile açık metnin anlaşılmaz bir biçime (şifreli-metne) dönüştürme süreci olarak tanımlanır; şifreyi çözme ise tam tersidir. Burada kullanılan kriptografik anahtar gizli bir parametredir. Kriptografide anahtar olmadan şifrelerin çözülmesi çok düşük bir olasılıktır. Diğer bir deyişle, kriptografi bütün anlaşılmazlığını şifreden anahtara aktarmıştır. Bunun için bu anahtarların üretimi çok önemlidir [1-3].

Kriptografik anahtarın üretimi kriptografi ile rasgele sayılar arasında kuvvetli bir bağ oluşmasına neden olmuştur. Çünkü anahtar gizlidir, açığa çıkarılmaması ve kimse tarafından tahmin edilememesi gerekmektedir. Bu yüzden anahtar rasgele seçilir. Anahtarı oluşturan bit dizilerindeki bazı düzensizliklere veya örneklere bakılarak anahtar tahmin edilebilir, bu yüzden anahtarın *gerçekten* rasgele seçilmesi önemlidir. Sonuç olarak, iyi kriptografi iyi rasgele sayılara gereksinim duyar [4].

Rasgele sayıların üretilmesi rasgele sayı üreteçleri ile sağlanmaktadır. Rasgele sayı üreteçleri Sözde Rasgele Sayı Üreteci (SRSÜ), Gerçek Rasgele Sayı Üreteci (GRSÜ) ve Hibrit Rasgele Sayı Üreteci (HRSÜ) olmak üzere 3 sınıfa ayrılmaktadır [5].

Sözde rasgele sayı üreteci rasgele sayı özelliklerine benzeyen sayı dizileri üreten bir algoritmadır ve *tohum* olarak adlandırılan başlangıç vektörü ile deterministik olarak hesaplanırlar. Aslında tohum, bu sınıf üretecinde belirsizliği temsil etmektedir. Tüm sözde rasgele üreteçleri deterministik bir algoritma ile çalıştığından dolayı gerçek rasgele olamayacaktır. SRSÜ'ler periyodik olabilmektedir ve periyodun açığa çıkması ile üretilen rasgele sayıların tamamen tahmin edilebilir olmasını mümkün kılacaktır. Ayrıca tohum değerinin korunamaması da bu duruma neden olacaktır. Bu sebepten dolayı kriptografik uygulamalarda kullanımı uygun değildir [5].

GRSÜ'ler ile rasgele sayı üretmek için entropi kaynağı olarak deterministik olmayan fiziksel olaylar kullanılır. TRNG tasarımlarında entropi kaynağı tarafından elde edilen sinyaller sayısala dönüştürülerek örnekleme yapılır. Örnekleme sürecinden sonra üretilen sayıların istatistiksel olarak birbirinden bağımsız olması için son işleme tabi tutulur. Ancak, son işlem uygulamaları zayıflıkları giderirken bit oranında azalmaya sebep olmuştur. GRSÜ'ler yavaş, maliyetli ve donanıma bağımlı olması gibi dezavantajlara sahiptir. Ancak GRSÜ'ler kriptografik uygulamalar için zorunlu olan kestirilememe, tekrar üretilememe ve iyi istatistikî özellikleri sağlaması sebebiyle kriptolojide birçok uygulamada kullanılmıştır [5].

Gerçek rasgele sayı üreteçleri çeşitli yöntemler ile tasarlanarak rasgele sayı dizileri üretmişlerdir. Bunlar doğrudan kuvvetlendirme, çift osilatör yöntemi ve kaos tabanlı uygulamalardır. Kaotik işaretlerin gürültü benzeri işaretler üretmesi sebebiyle kaotik sistemler son yıllarda RSÜ sistemlerinde sıklıkla kullanılmaya başlanmıştır [6,7]. GRSÜ ve SRSÜ sayı üreteçleri tarafından üretilen sayılar bilgi güvenliği sistemlerinde anahtar olarak kullanılabilmektedir. Ancak anahtarların sistem dışında kontrolsüz ortamlarda üretimi sistemin güvenilirliliğini azaltmaktadır. Bu sebeple anahtarlar eğer bir entegre içerisinde gerçekleştirilirse sistem daha güvenli olmaktadır [8]. Bundan dolayı anahtarlar FPGA (Field Programmable Gate Array- Alan Programlanabilir Kapı Dizileri) gibi programlanabilir donanımlarda üretilmeye başlanmıştır.

Kaos kavramı bazen "karmaşa" ya da "rasgele" gibi yanlış yorumlanabilmektedir. Matematiksel olarak, karmaşık sistem rasgele olmayan bir sistemdir (çünkü denklemlerinin hepsinin çözümü mümkündür) ve iki özgün karakteri temsil eder: (a) Var olan düzensizliği, periyodik olmayan düzensizlikleri (b) Başlangıç şartlarına aşırı hassasiyeti. Başlangıç şartlarına karşı hassaslık kelebek etkisi olarak bilinmektedir; "kelebek etkisi" terimi Edward Lorenz'in çalışmalarıyla ilişkilidir [9].

Aynı iki başlangıç noktası ile başlayan iki özdeş sistem, çok küçük değişiklikler ile tamamen farklı iki sonuç verebilir. Bundan dolayıdır ki; sonuç olarak uzun soluklu karmaşık sistemleri tahmin etmek imkânsızdır, çünkü gerçek bir sistemi gözlemlerken, sistemin başlangıcını gözlemlemek ancak kısıtlı hassasiyette mümkündür [9].

Bu noktada, karmaşık tabanlı rasgele sayı üreteçleri ile sözde rasgele üreteçlerin benzer olduğu düşünülebilir, çünkü her ikisi de rasgele olmayan algoritmaya dayanır. Fakat iki durum onları farklı kılar. Birincisi, karmaşık devre analog bir makinedir, rasgele bit üretmesi için durumun nicelenmesi (analog sinyali alarak bunu sayısala dönüştürme işlemi)

gereklidir. Geri-dönüşü olamayan bir operasyonun nicelenmesi, sadece nicel değerlerin bilgisiyle mümkün olamaz. İkincisi; diğer analog elektronik devreler gibi, karmaşık devre gürültüden etkilenir. İşlem süresinde gürültüyü görmezden gelmek dahi, başlangıç ve değerlendirme durumlarını sürekli değiştirir. Bu açıdan, karmaşık tabanlı rasgele sayı üreteci gürültü benzeri olguya dayanan üreteçten farklı değildir [9].

Aslında, karmaşığın kullanımı, ayrık-zamanlı karmaşık devrenin rasgele sayı üreteçlerinde kullanımı uzun yıllardan beri biliniyor [10,11]. Ulam ve Von Neumann 1947'de, cebirsel dağılımı ve yinelenen değerlerinin istenen herhangi bir dağılıma kolayca transfer edilmesinden dolayı lojistik haritanın kullanımını önermişlerdir [12]. Karmaşık devrenin bir avantajı da istatistiksel oranları ile herhangi bir müdahaleyi teorik olarak engellemesidir. Karmaşık devrenin bozulması ve (teorik olarak) tüm entropi çıkış bit akımı için kullanılabilmesi karmaşık devre modeli için geçerlidir. Ayrıca, ayrık zamanda ulaşılabilir entropi oranı devre hızı ile doğrudan orantılıdır, bu durum Ulam ve Von Neumann tarafından çok öncesinden anlaşılmıştır. Devre ne kadar yüksek hızda çalışırsa, o kadar ulaşılabilir entropi oranı oluşur. Ancak ayrık sistemlerin fonksiyonlarının oluşturulması zor olduğundan ve devrenin dışarıdan bir saat ile kontrol edilmesinden dolayı düşük hızda kalmıştır.

Sürekli zamanlı sistemlerde ise, yüksek hızda çalışılabildiğinden ve çoğu sürekli zaman sistemlerinin kolay tasarlanabilmesi ile daha çok kullanılmaya başlanmıştır. Bunun sonucu olarak yeni sürekli zamanlı kaotik osilatörler tasarlanmaya başlanmıştır. Sürekli zaman kaotik bir osilatörün çıkışı, periyotları birbirinden farklı sonsuz sinüs bileşeni içerir. Çıkıştaki bileşen sayısı arttıkça işaret gürültüye benzemekte ve osilatörün de kalitesi artmaktadır. Bunun yanı sıra, her temel kaotik osilatör içinde bir sinüs osilatörü vardır. Bu sinüs osilatörünün denklemlerinde yeni parametreler oluşturmak için, devreye yeni elemanlar eklenir. Bu sayede sinüs osilatörü kaotik osilatöre dönüştürülür. Bu nedenle çıkışta ana bir sinüs bileşeni olduğu unutulmamalıdır. Bu iki özellik, frekans spektrumu incelenerek kolayca görülebilir. Kaotik osilatörün performansını belirleyen başka bir faktör de, osilatörün kontrol edilebilir ve birbirinden bağımsız parametrelerinin olması ve sistemi kaosta tutacak bu parametrelerin geniş bir aralığa sahip olmasıdır. Kaotik osilatörlerden beklenilen diğer bir önemli özellikte yüksek frekanslı çıkış sinyalleri üretebilmesidir.

1.1. Tezin Amacı

Son yıllarda rasgele sayı üreteçlerinde kaos yoğun bir şekilde kullanılmaya başlanmıştır [13,14], çünkü kaotik sayı dizilerinin üretilmelerinin ve depolanmalarının kolay ve hızlı olduğu ispatlanmıştır, uzun sayı dizilerinin depolanmalarına gerek yoktur. Sadece birkaç fonksiyon (kaotik harita) ve birkaç parametre (başlangıç şartı) çok uzun diziler için bile yeterlidir, ayrıca çok fazla sayıda farklı sayı dizisi, başlangıç şartı değiştirilerek çok kolay bir şekilde üretilebilir [14]. Bu avantajları ile kaos rasgele sayı üreteci olarak kullanılmaya başlanmıştır.

Bu tezde, GRSÜ ayrıntılı olarak incelenmiş hemen hemen literatürde bulunan bütün tasarımlar tartışılmıştır.

Tez çalışmasında ilk olarak GRSÜ'de son işlemin önemi incelenmiştir. Son işlem, gerçek rasgele sayı üreteçlerinde kullanılan entropi kaynaklarının çevresel değişikliklerden etkilenmeleri nedeniyle oluşan zayıf istatistiksel özellikler gidermek amacıyla kullanılmaktadır. İkinci bir avantajı ise yan kanal analizi saldırılarına karşı sistemi dirençli hale getirmesidir. Ancak kullanılan son işlemler GRSÜ'den elde edilen çıkış bit oranını düşürmektedir. Bu amaçla çalışmada, literatürdeki Von Neumann [15], XOR [16], H son işlem [17] gibi çeşitli son işlem algoritmalarına alternatif olabilecek, GRSÜ'nün çıkış oranını düşürmeden istatistikî zayıflıkları gideren yeni lojistik haritaya dayalı son işlem algoritması önerilmiştir. Gerçekleştirilen uygulamada entropi kaynağı halka osilatörler (Ring Oscillator- RO) tarafından üretilen faz seğirmesi kullanılmıştır. RO tabanlı bir GRSÜ sisteminde lojistik haritanın üretilen sayılara etkisini gözlemleyebilmek için 4 farklı senaryo geliştirilmiştir. GRSÜ sisteminde kullanılan RO sayıları ve RO'lardaki tümleyen sayıları sırasıyla (114,13) (25,3) (10,3) (5,3) seçilmiştir. Tüm senaryolarda son işlem olarak lojistik harita kullanılmış olup her bir sistem Altera'nın EP4CE115F29C7 tabanlı FPGA bordunda gerçekleştirilmiştir. Elde edilen sonuçlar mevcut sonuçlar ile karşılaştırıldığında daha iyi sonuçların elde edilebileceği gösterilmiştir.

Diğer çalışmada ise, Saf SRSÜ sistemleri, rasgele sayı üreteçlerinin sağlaması gereken dört gereksinimden ilk üçünü sağlamaktadır. R4 gereksinimini sağlayamadığı için saf SRSÜ sistemlerinin kriptografik uygulamalarda kullanılması uygun değildir. R4 gereksiniminin sağlanması için, saf SRSÜ'deki geçerli iç durum değerinin gerçek rasgele veri ile güncellenmesi gerekmektedir. Gerçek rasgele veri, Saf SRSÜ'deki çıkış ve giriş fonksiyonuna ek girdi olarak eklenecektir. Böylece Hibrit SRSÜ sistemi elde edilecektir.

Yapılan çalışmada Saf SRSÜ olarak AES, ek girdi olarak da Xilinx FPGA üzerinde gerçekleştirilen Burke Shaw kaotik çeker kullanılmıştır.

Elde edilen sonuçların rasgeleliğini test etmek amacıyla FIPS140, NIST, AIS 31, Diehard, Crypt-X gibi çeşitli test paketleri geliştirilmiştir [18]. GRSÜ'lerin en önemli test kriterlerinden biri istatistiksel analizdir. Literatürde en çok kullanılan istatistiksel analiz Ulusal Standartlar ve Teknoloji Enstitüsü tarafında kullanılan NIST 800-22 test paketidir. Bu test birçok yerde kullanılmasına rağmen birçok araştırmacı tarafından anlaşılamamıştır. Bu sorunu gidermek ve tasarladığımız GRSÜ'lerden elde ettiğimiz sonuçları test etmek amacıyla C# dilinde Windows ortamında çalışan bir program yazılmıştır. Ayrıca, NIST test paketi ayrıntılı bir biçimde anlatılmıştır.

1.2. Tezin İçeriği

Tezin içeriği yukarıda belirtilen amaç doğrultusunda 6 bölümden oluşmaktadır. Bölüm 2'de rasgelelilik ve rasgele sayı üreteçleri ile ilgili bilgi verilmiştir. Bölüm 3'te ise gerçek rasgele sayı üreteçlerinin temel yapısı ve literatürde ki gerçek rasgele sayı üreteçlerinin açıklamaları verilmiştir. Bölüm 4'te Uluslararası Standartlar ve Teknoloji Enstitüsü tarafından yayınlamış olan NIST testlerinin ayrıntılı olarak açıklamaları verilmiştir. Yazılımı gerçekleştirilen bu test paketinin program ekran çıktıları ve test işleminin nasıl yapıldığı açıklanmıştır. Bölüm 5'te ise halka osilatör kullanılarak FPGA üzerinde GRSÜ uygulanmasının gerçekleştirilmesi ve önerilen son işlem sonrası elde edilen sonuçlar verilmiştir. Ayrıca önerilen hibrit SRSÜ sistemi açıklanmıştır. Son bölümde ise çalışmayla ilgili sonuçlara ve bu sonuçlarla ilgili yorumlara yer verilmiştir.

2. RASGELELİLİK VE RASGELE SAYI ÜRETEÇLERİ

2.1. Rasgelelilik

Rasgelelilik en basit anlamda kesin olarak bilinememektir. Rasgele olmayan deterministiktir yani belirlidir. Kriptografide tahmin edilememe, istatistikte ise örnekler arasında ilişki bulunmaması anlamına gelmektedir [18,19].

Rasgele sayılar, belirli bir aralık için tanımlanmış, oluşma olasılıkları birbirine eşit ve bu sayılar arasında belirli bir ilişki olmayan sayılar olarak tanımlanabilir. Rasgele sayılar örneğin hilesiz madeni paranın atılması gibi rasgele süreçlerden elde edilir. Üretilen rasgele sayıların iyi istatistiksel özellikler göstermesi ve tahmin edilememesi gerekmektedir [19].

Rasgele sayıların kullanım alanları [19,20]:

- **Simülasyon:** Bilgisayar doğal olayları simüle etmeye başladığında rasgele sayılar gerçeğe uygun işler yapmak için gereklidir. Simülasyon hareket araştırmalarından (insanların rasgele aralıklar ile havaalanına gelmesi) nükleer fizik çalışmalarına (parçacıkların rasgele çarpışması konuları) kadar birçok alanı kapsar.
- **Örnekleme:** Genellikle olası her durumu incelemek zordur. Ancak rasgele bir örnek tipik bir davranışı neyin oluşturduğuna dair bir bilgi verecektir.
- **Nümerik analiz:** Karmaşık sayısal problemleri çözmek için yaratıcı teknikler rasgele sayılar kullanılarak geliştirilmiştir.
- **Bilgisayar programlama:** Rasgele değerler bilgisayar algoritmalarının etkinliğini test etmek amacıyla iyi bir veri kaynağı olur.
- **Karar verme:** Bazen tamamı ile tarafsız karar vermek önemlidir. Bu yetenek bazen bilgisayar algoritmalarında kullanılmaktadır. Örneğin belirli bir durumda karar vermek her zaman algoritmanın daha yavaş çalışmasına sebep olabilir. Rasgelelilik oyun teorilerinde iyi stratejilerin özel bir bölümüdür.
- **Eğlence:** Zar atma, kart destelerini karıştırma, rulet çarkını döndürme gibi eğlenceli oyunlar genel olarak rasgele sayıları kullanır.
- **Kriptografi:** Kriptografik sistemlerin işlevlerini doğru olarak yerine getirmesi için rasgele seçilmiş bit dizileri gereklidir. Bir kriptografik sistem içerisinde rasgele sayıların kullanıldığı yerler şu şekilde belirtilmiştir [21],

- Anahtar üretimi: Kriptografide herhangi bir verinin korunması amacıyla açık ve özel anahtar çifti kullanılmaktadır. Açık anahtar herkese açık iken özel anahtar gizli tutulmak zorundadır. Bu anahtarların üretimi sırasında rasgele sayılara ihtiyaç duyulmaktadır.
- Başlangıç vektörü (initilization vector-IV): Başlangıç vektörü, tipik olarak rasgele veya sözde rasgele sayılara ihtiyaç duyan kriptografik ilkellerin sabit boyutlu girişidir. Blok şifreler için, IV'nin kullanımı işlem modları ile tanımlanır. Rasgele sayılar özet fonksiyonları ve mesaj doğrulama kodu gibi diğer ilkellerde de gereklidir.
- Anahtar dağıtımında: Alıcı ve verici arasında gizli anahtarlar dağıtılırken, gizli anahtar içeren bir mesajı şifrelemek için bir oturum anahtarı üretilmelidir. Göndericinin anahtar dağıtım merkezine rasgele oluşturulmuş sayı içeren bir mesaj göndermesi gerekmektedir.
- Blok takviyesi (Block padding)
- Kimlik doğrulama protokolleri
- Sözde rasgele sayı üreteçleri için tohum üretimi
- Asal sayı üretimi
- Yan kanal ataklarına karşı koruma

2.2. Kriptografik Uygulamalar İçin Rasgele Sayı Üreteçleri

Rasgele sayılar çeşitli kriptografik uygulamalar için zorunluluktur. Çünkü kriptografi anahtarların üretimi ve dağıtımında, başlangıç vektörü oluşturulmasında, kimlik doğrulama protokollerinde, asal sayı ve şifre üretiminde rasgele sayılara ihtiyaç duyar. Bir kriptografik sistemin güvenliği elde edilen sayıların gerçek rasgeleliğine dayanmaktadır. Bu sebeple, rasgele sayı üreteçleri birçok güvenlik alanlarının bir bölümünü oluşturur. Uygun olmayan üreteçler ile üretilen rasgele sayılar güvenlik sistemlerini bütünüyle zayıflatabilir [21,22].

Bir saldırgan RSÜ'leri ile üretilmiş rasgele sayıların büyük bir kısmını biliyor olsa bile, rasgele sayıların tahmin edilemez olması gerekliliği açıktır. İdeal olarak, rasgele sayılar sıralanması tekdüze dağıtılmış ve bağımsız olmalıdır. Bu sebeplerden dolayı kriptografik sistemlerde kullanılan rasgele sayıların bazı sıkı gereksinimleri sağlamaları gerekmektedir. Bu gereksinimler şu şekilde açıklanabilir:

Kriptografik uygulamalarda güvenliğin tam olarak sağlanabilmesi için kullanılan rasgele sayıların sıkı gereksinimleri sağlaması gerekmektedir. Açık olarak görülmektedir ki, rasgele sayılar tüm değerleri eşit olasılıklı, önceki ve sonraki değerlerinin de birbirlerinden bağımsız olması gerekmektedir, ancak bu gereksinimler çok kısıtlayıcıdır ve ideal bir RSÜ'ni tanımlar.

Bir RSÜ'de olması gereken özellikler:

- İyi istatistiksel özellikler ile çıkış bitleri üretebilmelidir. Bu özellik tekrarlama saldırıları ile korelâsyona dayalı saldırıları engellemektedir.

Ancak bu özellik hassas uygulamalar için yetersizdir, çünkü bir saldırganın küçük ve bilinen bir alt diziden rasgele sayıları tahmin etmesine izin verir. Bir saldırganın bazı rasgele sayıları bilir varsayımı birçok uygulama için gerçekçidir. Örnek vermek gerekirse: Ali uygun bir anahtar değişim protokolü kullanarak rasgele seçilmiş oturum anahtarı $K_ö$ ile gizli bir mesajı şifreler ve yasal alıcıya $K_ö$'yü gönderir. Burada yasal alıcı sadece kendine gelen mesajın içeriğini görebilecek diğer mesajların içeriğini göremeyecektir. Bu durumda, bir mesajın yasal alıcısı en azından bir oturum anahtarını bildiğinden dolayı aslında imtiyazlı saldırgandır. Eğer Ali genel bir sunucuda sunarsa bir saldırgan milyonlarca rasgele sayıları öğrenebilir. Bu da bir sonraki özelliği gösterir.

- Rasgele sayı üreteçlerinden elde edilen çıkışlardan önceki veya sonraki çıkışların elde edilememesi gerekir (Tahmin edilememe).
- Düzgün dağılıma sahip olmalıdır.
- Yüksek entropi ile çıkış bitleri üretebilmelidir.
- Saldırılara karşı sağlam olmalıdır.
- Çevresel değişimlere karşı sağlam olmalıdır.
- Ürünün kullanım ömrü boyunca sabit özelliklere sahip olmalıdır.
- Oldukça yüksek veri ve bit hızına sahip olmalıdır.
- Basit bir uyguluma olmalıdır.

2.3. Entropi Kavramı

Bir sistemin entropisinin klasik kavramı termodinamik teoriden gelir ve bir sistemin niteliksel bozukluğunun ölçümü olarak tanımlanabilir. Entropi 19. y.y. sonlarında Boltzmann tarafından kapalı bir kaptaki ideal bir gazın düzensizliğinin ölçüsü olarak ortaya konulan bir kavramdır. İstatistiksel termodinamiğin ve kuantum teorisinin

gelişimiyle, entropi toplam enerjinin karışması ya da "dağılması" açısından tanımlanmaktadır.

Bilgi teorisinde ise, entropi bir rasgele değişkenin belirsizlik ölçüsü olarak tanımlanır. Genellikle Shannon entropi olarak kullanılmaktadır. Shannon'a göre bir olayın belirsizlik içermesi durumunda o olay hakkında bilgi edinilebilir. Buna göre oluşma olasılığı yüksek olan bir olayın ortaya çıkması fazla bilgi getirmemekte, oluşma olasılığı düşük olanlarda ise daha fazla bilgi taşımaktadır. Bu sebepten dolayı Shannon entropi kavramını bir olayın oluşabilecek olası durumların beklenen değerini matematiksel bağıntı ile tanımlamıştır. Bu şekilde bir rasgele sürecin entropisi hesaplanabilmektedir.

Belirsizliğin ölçütü olarak olayın gerçekleşme olasılığı dikkate alınır [23]. Olasılıkların yoğunlukları arttıkça ilgili olasılık uzayının belirsizliği azalır. Örneğin, kolayca görülebilir ki yazı ya da tura için (0.5, 0.5) olasılık dağılımının entropisi, bir piyangoda kazanmanın (0.00001,0.99999) olasılık dağılımının entropisinden çok daha fazladır [24].

Bu durum şöyle de ifade edilebilir. Bir olasılık deneyinin a1, a2, K, a_n olası sonuçları, p_1, p_2, K, p_n olasılıklarıyla

$$\sum_{i=1}^{n} p_i = 1 \text{ ve } p_i \geq 0 \; i = 1,2, \ldots, n \qquad (2.1)$$

koşullarını sağlamak suretiyle ortaya çıksın.

Bu olasılık deneyinde olasılık i. sonuçta yoğunlaşmış ise ve $p_j = 0$ ise ($\forall\, i \neq j$ için) bu durumda deneyin sonucu kesindir, belirsizlik yoktur ve yoğunluk en büyüktür. Tersi durumda, eğer deneyin tüm sonuçları eşit olasılıklara sahip ise ($p_j = \frac{1}{n}$, $i = 1,2 \ldots . n$) yoğunluk en küçük, deneyin sonuçlarının belirsizliği ise en büyüktür [23].

Shannon ölçütü;

$H_n(p) = -\sum_{i=1}^{n} p_i \ln p_i = 1$ olarak verilir.

Shannon fonksiyonu için özellikler şu şekildedir.

- Entropi negatif olamaz. $H_n(p) \geq 0$'dır.
- $H_n(p)$ yer değiştirme özelliğine sahiptir. Yani p_1, p_2, ..., p_n olasılıkları kendi aralarında yer değiştirse de entropi ölçümü değişmez.
- $H_n(p)$; 0 ile 1 arasındaki tüm pi'ler için p_1, p_2, ..., p_n'in sürekli fonksiyondur.
- Olasılığı sıfır olan bir olayın ifadeye eklenmesi entropiyi değiştirmez.

S sonlu olasılık örnek uzayı olarak varsayılır. Bir örnek uzayı S, rasgele uygulamaların mümkün olan bütün sonuçlarının kümesidir. Bir madeni para havaya atıldığında, bu durumun iki elemanı vardır, S={yazı, tura}. Bir zar atıldığında, bu durumun altı elemanı vardır, S={1,2,3,4,5,6}. S'nin belirsizliği veya entropisi aşağıdaki denklemde olduğu gibi tanımlanır:

$$H(S) = -\sum P(s)\log_2 P(s) \tag{2.2}$$

Burada s∈S uygulamaların olası sonucudur. P(s)=P{S=s}, s∈S 'tir. H(S), H(P) olarak gösterilmektedir.

- Logaritmanın tabanı 2 olarak seçilir ise bilgi miktarının birimi bit olmaktadır.
- Log(1)=0 olduğunda kesin olarak gerçekleşeceğini bildiğimiz bir olay bize 0 bitlik bilgi verir.
- Bütün olasılıklar eşit olduğu takdirde bilgi miktarı maksimum değerini alır.
- - işareti H(S)≥0 koşulunu sağlar.

Örneğin: madeni paranın havaya atıldığını varsayalım. Sonuç, her birinin olasılıkları 1/2 olan yazı ve turadır. Bunun anlamı:

$$H(P)= P(\text{yazı})*[\log_2 1/P(\text{yazı})]+ P(\text{tura})*[\log_2 1/P(\text{tura})] \tag{2.3}$$

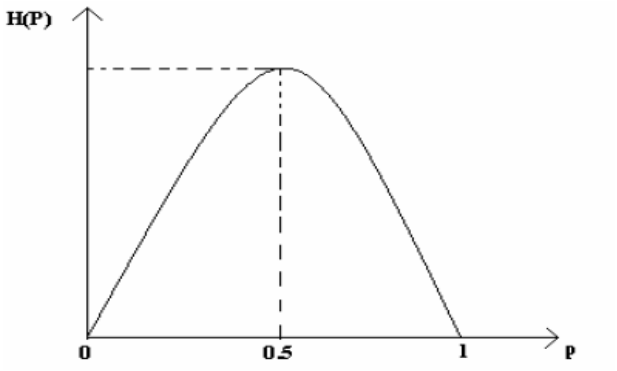
$$H(P)= (1/2)*[\log_2 1/(1/2) + (1/2) * [\log_2 1/(1/2)]=1 \text{ bit}$$

Örnek şunu gösterir: hilesiz madeni para atışı bize bilgiyi (belirsizliği) 1 bit verir.

Bir rassal değişkenin aldığı iki değerin eşit olmadığı, yani $P(S=s_1)=p$, $P(S=s_2)=1-p$ olduğu durumda,

$$H(P)=-p \log p - (1-p) \log (1-p) \tag{2.4}$$

H(P) fonksiyonun grafiği Şekil 2.1'de görülmektedir. Entropi fonksiyonu konkav bir fonksiyondur ve p=0 ya da p=1 iken H(P)=0 değerini alır. Zaten bilindiği gibi olasılığın sıfır ya da bir olması durumunda rassallık dolayısıyla da belirsizlik yoktur. Benzer olarak, entropi fonksiyonu maksimum değerini p=0.5'de alır ki bu durumda belirsizlik maksimumdur ve Şekil 2.1'den de görüleceği gibi entropi fonksiyonu S rassal değişkeninin aldığı değerlere değil sadece olasılıklarına bağlıdır [25].



Şekil 2.1. H(P) fonksiyonun grafiği

Eğer P(yazı)=1, P(tura)=0 olduğu takdirde bu uygulamada bilgi (belirsizlik) yoktur. Entropi sıfır olacaktır.

2.3.1. Maksimum Entropi

Maksimum entropi mümkün olan bütün olasılıkların eşit olduğu durumda başarılabilir (Çıkan tüm sonuçlar eşit olasılıklı). Bu durumda maksimum entropi:

$$H_{max}(P)=\log_2 n \text{ bits} \quad (2.5)$$

Yani, her olası örnek uzayının entropisi bu formül ile tanımlanan en üst sınırdır.

Örneğin:

Bir zar atılan uygulamanın entropisi: $H(P)=\log_2(6)=2.58$ bits

2.3.2. Minimum Entropi

Minimum entropi her zaman sonuçlardan birinin ortaya çıkması ile meydana gelir.

$$H_{min}(P)=0 \text{ bits} \quad (2.6)$$

Not: Olası örnek uzayının entropisi 0 bit ile $Log_2 n$ bit arasındadır. Burada, n olası sonuçların sayısıdır.

2.3.3. Entropinin Yorumu

Entropi sonuçları eşit olasılıklı olduğunda olası örnek uzayının her bir sonucu göstermek için gerekli bit sayısı olarak düşünülebilir. Örneğin, olası örnek uzayının sekiz

olası sonucu varsa, her bir sonuç 3 bit olarak (000'dan 111'e) gösterilir. Uygulamanın sonucu alındığında, bilginin 3 bit alındığı söylenebilir. Bu olası örnek uzayının entropisi 3 bittir ($\log_2 8 = 3$).

2.3.4. Üretecin Entropisinin Arttırılması

İyi sonuç vermeyen bir RSÜ'nin olduğunu düşünüldüğünde, bu üreteçten elde edilen bit dizisinin kalitesinin arttırılması gerekmektedir ki bu aşama mükemmel son işlem aşaması olacaktır. İyi sonuç vermeyen RSÜ'ne son işlem uygulandığı takdirde üretecin kalitesi artacaktır. Son işlem uygulandıktan sonra çıkış veri oranı giriş veri oranından küçük olmaktadır, diğer bir deyişle son işlem bir sıkıştırma algoritmasıdır.

Genellikle son işlem fonksiyonları, uzun diziler düşünürler ve oto korelasyonel bir şekilde "yayılırlar". Bağlantılı bit akışında, bağlantı yan yana iki bit katsayısı, uzak bitler arasındaki katsayı bağlantısından daha fazladır, bundan dolayı uzak bitler arasından ziyade yakın bitler arasında ilişki daha güçlüdür. Son işlem daha düzgün bir profil oluşturmak için, basit bir sıkıştırma yerine istatistiksel testlerden daha iyi sonuçlar almak için oto korelasyonu yeniden şekillendirir. Bu yolla, akış çok daha rastlantısal görünür veya daha kesin bir şekilde söylemek gerekirse, bitler arasında ilişkiyi bulmak daha da zor olur.

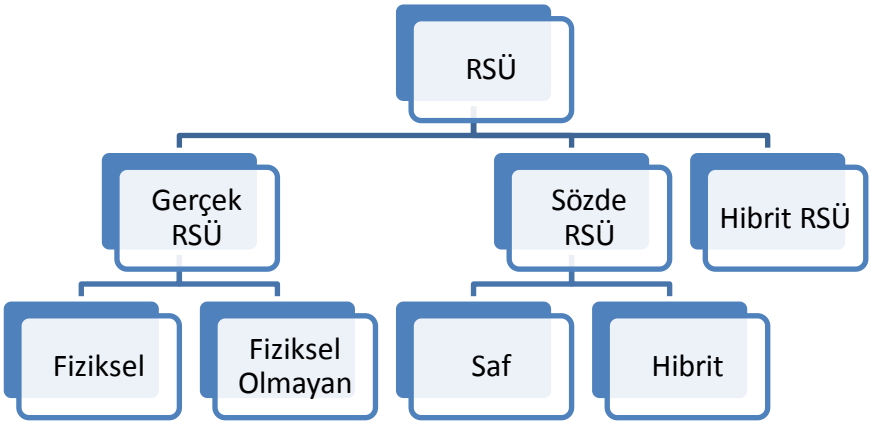
Fakat daha rasgele ve aha az rasgele şeklinde dizi açısından konuşmak gerekirse, kişi sadece entropiyi düşünmelidir: entropi yükseldikçe, dizi daha iyi olur.

Son işlem Bölüm 3.1.3'te ayrıntılı olarak açıklanacaktır.

2.4. Rasgele Sayı Üreteçlerini Sınıflandırma

Rasgele sayı üreteci tahmin edilemeyen sayı dizileri veya tekrar etmeyen bitler üreten bir sistem veya aygıttır. Rasgele sayı dizileri aralarında korelasyon bulunmayan ve istatistiksel olarak birbirinden bağımsız olan sayılardan oluşur. Rasgele Sayı Üreteçleri Sözde Rasgele Sayı Üreteçleri (SRSÜ), Gerçek Rasgele Sayı Üreteçleri (GRSÜ) ve Hibrit Rasgele Sayı Üreteçleri olmak üzere üç ana sınıfa ayrılır. SRSÜ'ler ile belirli bir başlangıç değerine (Tohum-Seed) bağlı olarak algoritmalar ile uzun sayı dizileri üretir. GRSÜ'ler ise elektronik devrelerdeki gürültü kaynaklarını, ısı gürültüsünü, osilatörler arasındaki faz farkını ve kaotik devrelerdeki gürültü kaynaklarını kullanarak gerçek diyebileceğimiz rasgele sayı dizileri oluşturur. Gerçek rasgele sayı üreteçleri fiziksel ve fiziksel olmayan (sistem zamanı, disk kafasının hareket zamanı, RAM içerikleri, kullanıcı etkileşimi) sayı

üreteçleri olarak iki alt sınıfa ayrılır. Şekil 2.2'de sınıflar ayrıntılı olarak gösterilmiştir. Hem SRSÜ hem de GRSÜ ile birlikte tasarlanan elemanlar hibrit RSÜ olarak adlandırılır. GRSÜ'lerin güvenliği çıkışlarının tahmin edilemezliğine bağlı iken SRSÜ'lerin güvenliği olası saldırılara karşı hesaplama karmaşıklığına bağlıdır. SRSÜ, uygulamaların birçoğu için talepleri karşılamada yeterlidir, ancak kriptografik uygulamalar için güçlü rasgele sayı üreteçlerine ihtiyaç vardır. Rasgele sayıların tamamının güvenilir bir kaynaktan sağlanması amacıyla GRSÜ kullanılmalıdır [5].



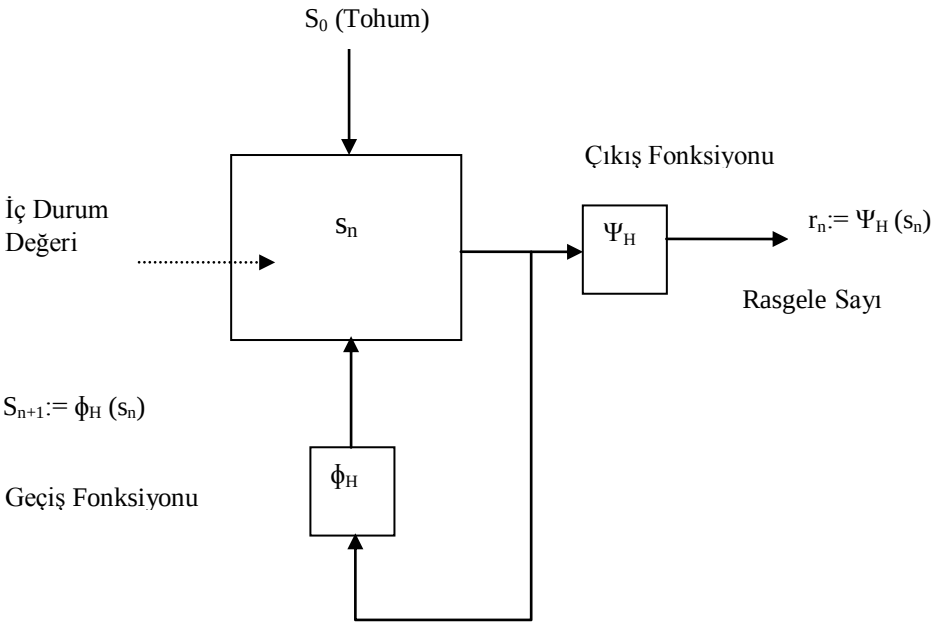
Şekil 2.2. RSÜ sınıflandırması

2.4.1. Deterministik (Sözde) Rasgele Sayı Üreteçleri (SRSÜ)

Sözde rasgele sayı üreteçleri saf ve hibrit olmak üzere iki sınıfa ayrılmaktadır. Bu bölümde saf ve hibrit SRSÜ'ler açıklanacaktır.

2.4.1.1. Saf Sözde Rasgele Sayı Üreteçleri

Saf sözde rasgele sayı üreteçleri, rasgele giriş olarak bazı entropi kaynağından elde edilen tohum değerini kullanır ve gerçek rasgele sayı üreteçlerinden hesapsal olarak ayırt edilmesi mümkün olmayan diziler üretir. Bu üreteçler deterministik olduğundan dolayı çıkış değeri girilen tohum değerini geçemez. Ayrıca diziler belli bir süre sonra kendini tekrar etmeye başlar. Yani sistem periyodiklik özelliği gösterir [5,26-30].



Şekil 2.3. Saf SRSÜ genel tasarımı [5].

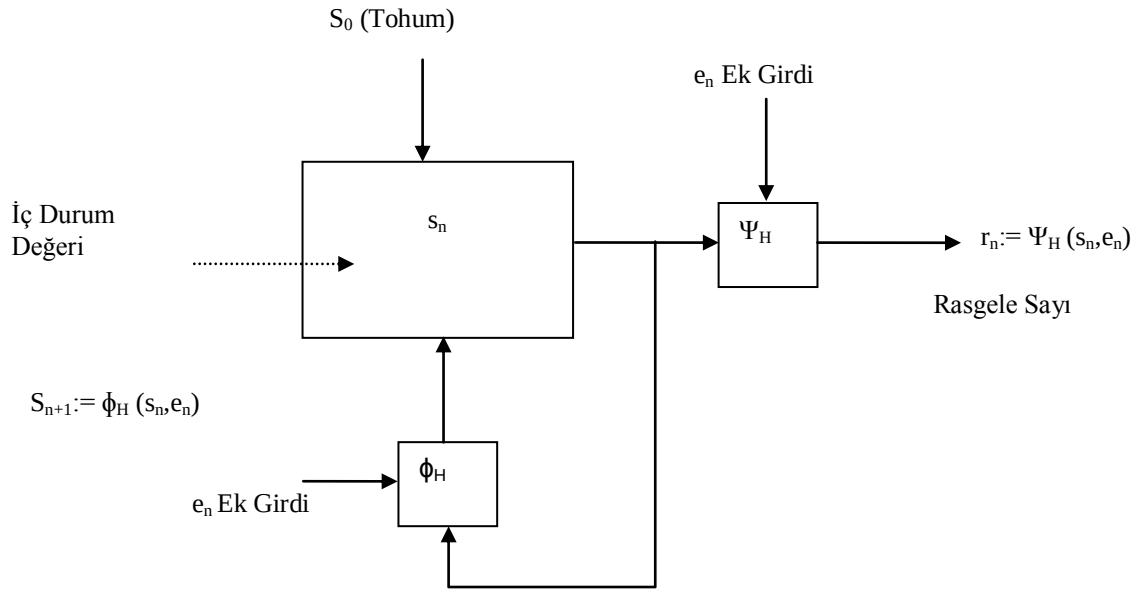
Şekil 2.3'de saf SRSÜ'nün genel tasarımı gösterilmiştir. n-1 rasgele sayılar üretildikten sonra, SRSÜ'nin iç durum değeri $s_n \in S$ elde edilir. Burada sonlu kümeler S durum uzayı ve R çıkış uzayı olarak adlandırılır. Şekil 2.3'den de görüldüğü üzere $\Psi:S\rightarrow R$ çıkış fonksiyonu geçerli iç durum değeri s_n 'den bir sonraki rasgele sayı r_n 'i hesaplar. Sonra s_n değeri geçiş fonksiyonu ϕ_H (ör. $S_{n+1}=\phi(s_n)$) ile s_{n+1} olarak değiştirilir. İlk durum değeri s_1 ilk girişte kullanılan tohum değeri s_0 'dan $s_1=\phi(s_0)$ şeklinde üretilir. Bu üretim kısmında daha karmaşık üretim yöntemleri kullanılabilir. Bu kısımda önemli olan durum, bütün durum değerleri $s_1,s_2,....$ ve bütün üretilen rasgele sayı $r_1,r_2,....$'lerin üretiminin tamamı tohum değeri s_0 'a bağlıdır. Bu durum s_0 değerinin bilinmesi ile tüm sistemin elde edilme riskini ortaya çıkarmaktadır. Bu sebepten dolayı tahmin edilememe özelliğinin sağlanması için s_0 tohum değeri rasgele seçilmek zorundadır. Saf SRSÜ (S, R, Ψ, ϕ, p_s) değişkenler dizisi ile tanımlanır. Burada p_s rasgele tohumun olasılık dağılımı olarak tanımlanır. Tohum üretimi saf SRSÜ sisteminin dışında gerçekleştirilir. Genellikle tohum, tahmin edilememe özelliği sağlaması amacıyla bir GRSÜ tarafından üretilir [5].

Saf SRSÜ'nin dezavantajı (GRSÜ ile karşılaştırıldığında) çıkışının tamamının tohum tarafından belirlenmesi ve bir sonraki rasgele sayının sadece geçerli iç durum değerine bağlı olmasıdır. Bundan dolayı, sistem aktif olmasa bile iç durum değeri korunmak zorundadır. SRSÜ'lerin avantajlı yanı, diğer üreteçlere göre ucuz olması ve herhangi bir

donanım ihtiyacı olmamasıdır. Tahmin edilememe özelliğinin sağlanması için tohum entropisinin büyük olması ile geçiş ve çıkış fonksiyonlarının yeterince karmaşık olması gerekmektedir.

2.4.1.2. Hibrit Sözde Rasgele Sayı Üreteçleri

Saf SRSÜ'den farklı olarak burada sisteme sonlu bir küme olan E_0'dan ek bir girdi sağlanacaktır. Geçiş fonksiyonu $E=E_0 \cup \{\infty\}$ ile $\phi_H:SxE \rightarrow S$ olur, burada ∞ ek girdi olmadığı anlamındadır. Biçimsel olarak, herhangi bir saf SRSÜ her bir adımda ∞ ek girdi ile Hibrit SRSÜ olarak Şekil 2.4'te görülebilir [5].



Şekil 2.4. Hibrit SRSÜ genel tasarımı [5].

Hibrit SRSÜ, yedi değişken kullanılarak tanımlanır. $(S, R, E, \phi_H, \psi_H, p_S, (q_n)_{n\in N})$ E ek girdi kümesi ve $(q_n)_{n\in N}$ ek verilen olasılık dağılımı olarak belirtilir. Eğer $E=\{\infty\}$ ve $q_n=\varepsilon_\infty$ ise saf SRSÜ' de yedi değişkenli açıklanabilir.

Sistemde ek giriş dizileri e_1, e_2 sabit veya bir saldırgan tarafından biliniyor olsa bile bu durum Hibrit SRSÜ'nün güvenliğini azaltamaz. Ek girişin sisteme tahmin edilememe ve rasgelelilik özellikleri katması nedeniyle güvenlik artar.

Bir saldırgan bu SRSÜ'de kullanıcı tarafından fark edilmeden SRSÜ deki geçerli iç durum değeri bilgisini elde ettiği zaman, bazı uygulamalar için aşağıdaki özellik arzu edilebilir ki daha fazla rasgele sayı üretir.

- İç durum değeri bilgisi dahi, pratik olarak bir sonraki rasgele sayılarının veya iç durum bilgisi olmadan göz ardı edilemeyecek kadar büyük olasılıkla bu değerlerin tahmin edilip hesaplanmasına izin verilmemelidir.

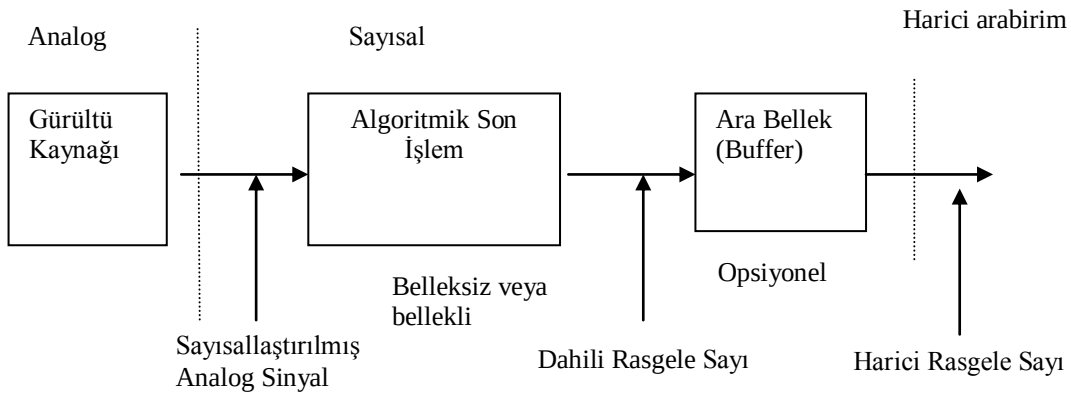
Tabi ki saf SRSÜ yukarıdaki özelliği yerine getiremez. Bu özellik ek girdinin rasgeleliğine bağlıdır. Güçlü bir GRSÜ' den devamlı ek girdi alınması sistemi güçlendirir.

2.4.2. Gerçek Rasgele Sayı Üreteçleri (GRSÜ)

Bu bölümde, GRSÜ'nün genel tasarımı ve önemli özellikleri açıklanmıştır. GRSÜ için daha kapsamlı değerlendirme bir sonraki bölümde yapılacaktır. GRSÜ'ler, Fiziksel Gerçek Rasgele Sayı Üreteci (FGRSÜ) ve Fiziksel Olmayan Gerçek Rasgele Sayı Üreteci (FOGRSÜ) olmak üzere iki sınıfa ayrılmıştır.

2.4.2.1. Fiziksel Gerçek Rasgele Sayı Üreteçleri (FGRSÜ)

FGRSÜ'nün Şekil 2.5'de genel tasarımı gösterilmiştir. Tez içerisinde GRSÜ olarak kullanılacaktır. GRSÜ'ler entropi kaynağı (tohum) olarak deterministik olmayan doğal fiziksel olaylar olarak tanımlanır ve rasgele sayı üretmede kullanılırlar, genellikle analog gürültünün sayısal sinyale dönüşümünü içerir. Entropi kaynağı olarak elektronik devreler (gürültülü diyotları veya serbest salınımlı osilatörleri kullanarak) veya fiziksel uygulamalar (radyoaktif çürüme, fotonların quantum etkisi) kullanılır. İyi tasarlanmış bir GRSÜ'de bu entropi kaynağı, son işlemden geçirilen istatistiksel bağımsız bit dizilerini elde etmek için örneklenir ve gerçek rasgele sayı dizileri ile sonuçlanır. Üretilen rasgele bitten bir sonraki tahmin edilememelidir ve aynı rasgele bitin üretilmesi engellenmelidir. GRSÜ'lerin gerçek rasgeleliği tamamıyla entropi kaynağına bağlıdır ve çıkışı etkileyebilecek herhangi deterministik giriş miktarı sistemi saldırı riski altında bırakır. Bununla birlikte entropisi kolay ayırt edilebilen sistemlerde deterministik girişleri çıkarmak oldukça zordur, onun için sağlam bir tasarım deterministik unsurları dikkate almalıdır [5,31-34].



Şekil 2.5. Fiziksel GRSÜ genel tasarımı

Burada sayısallaştırılan değerlere sayısallaştırılmış analog sinyaller veya kısaca DAS rasgele olarak adlandırılır. DAS rasgele sayılarının potansiyel zayıflıklarını azaltmak amacıyla algoritmik son işlem uygulanır. Ancak bu uygulama sırasında çıkış bit oranı azaltmakta ve çalışma hızını düşürmekte olduğuna dikkat etmek gerekir. Güçlü gürültü kaynaklarının algoritmik son işleme ihtiyacı yoktur. Algoritmik son işlemin en çok kullanılan uygulamaları Von Neumann, XOR ve Hash fonksiyonlarıdır. Örneğin Von Neumann uygulamasında dizide bulunan 11 ve 00 bit dizileri atılır. Diğer 01 (0) ve 10 (1) bit dizlerinin ilk başlangıç biti ne ise o alınır.

Saf SRSÜ'nün güvenliği iç durum değerinin ve giriş değeri olan tohum değerinin tahmin edilememesine bağlıdır. Güvenlik için diğer önemli bir unsurda geçiş ve çıkış fonksiyonlarının yeterince karmaşık olmasıdır. SRSÜ'lerin temel bileşenleri kriptografik ilkellerdir ve SRSÜ'lerin güvenliği genellikle bu ilkellerin iyi bilinen özelliklerine dayanmaktadır. SRSÜ'ler hesapsal güvenli olabilir, ancak bunların değerlendirmesi ilkellere saldırılar mümkün hale geldikçe zamanla değişir.

SRSÜ'nün aksine, GRSÜ'ler (fiziksel ve fiziksel olmayan) üretilen rasgele sayıların tahmin edilememesine bağlıdır. Elbette, entropi tahminlerinin doğruluğunu sağlamak şartıyla, böyle rasgele sayıları tahminde umulan iş yükü (yüklenme) zamanla sabit kalır. Sonuç olarak, uzun vadede gizliliği korunmak zorunda olan rasgele sayıların üretimi için GRSÜ'nün kullanılması daha uygundur. Rasgele sayıları bulmak için tahmin ile ilgili umulan sayıya göre hesaplanan teorik güvenlik sınırı ancak GRSÜ ile elde edilebilir. SRSÜ'den farklı olarak, değerlendirici rasgele bit başına düşen entropi tahmininde hata

yapmadıkça bu sayıda zamanla bir azalma olmaz. Tablo 2.1 ve 2.2'de GRSÜ ile SRSÜ karşılaştırılmaları verilmiştir.

Tablo 2.1. GRSÜ ve SRSÜ karşılaştırması [35].

	Sözde Rasgele Sayı Üreteçleri	**Gerçek Rasgele Sayı Üreteçleri**
Verimlilik	Mükemmel	Zayıf
Deterministik	Deterministik	Deterministik Değil
Periyodiklik	Periyodik	Periyodik Değil
Donanım Gereksinimi	İhtiyaç yok	Zorunlu

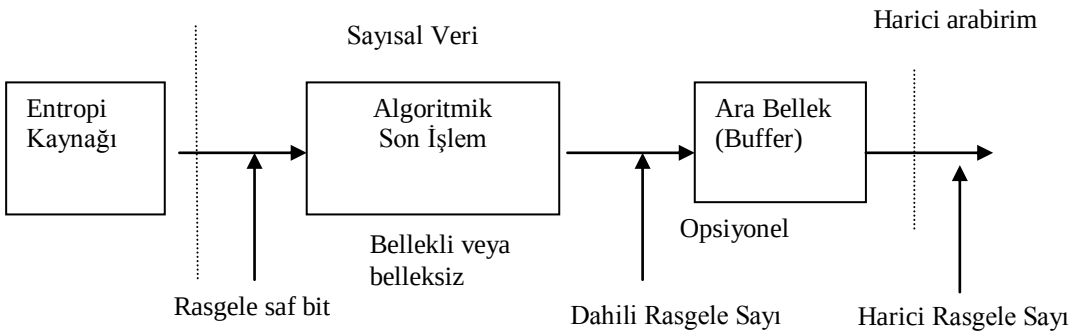
Tablo 2.2. Uygulamalarda GRSÜ ve SRSÜ karşılaştırması [35].

Uygulamalar	**En uygun Üreteç**
Şans Oyunları ve çekiliş, kura	GRSÜ
Oyunlar ve Kumar	GRSÜ
Rasgele Örnekleme	GRSÜ
Simülasyon ve Modelleme	SRSÜ
Güvenlik (veri şifreleme anahtarlarının üretimi)	GRSÜ

2.4.2.2. Fiziksel Olmayan Gerçek Rasgele Sayı Üreteçleri (FOGRSÜ)

Şekil 2.6'da genel tasarımı gösterilen FOGRSÜ genel olarak FGRSÜ tasarımı ile benzerdir. Ancak burada gürültü kaynağı için zorunlu donanım ihtiyacına gerek bulunmamaktadır. FOGRSÜ gürültü kaynağı olarak sistem verileri (PC zamanı, RAM verileri, işlem sayıları) veya kullanıcı etkileşimini (tuşlara basma, fare hareketi) kullanır. Kullanılan bu gürültü kaynaklarının genellikle entropisi düşüktür ve bu sebepten yüksek sıkıştırmalı son işlem algoritmasına ihtiyaç duyar. FGRSÜ'nü FOGRSÜ'den en önemli farkı entropi kaynağının kullanıcının kendisi veya kullanılan bilgisayardaki ayarlara bağlı olarak kullanıcının kontrolü altında olmamasıdır. Bu FGRSÜ ve FOGRSÜ'nin güvenlik değerlendirmesinde önemli farklılıklar göstermektedir. Ayrıca FGRSÜ den farklı olarak, FOGRSÜ'lerin çalıştırılma ortamlarının aynı olmasına gerek yoktur. Çok farklı durumlarda olabilir, bu durumda saf bitlerin entropisine etkisi olabilir [5].

FOGRSÜ'nün yazılım uygulamaları için kullanılmaktadır. FOGRSÜ' nün çıkışı direkt kullanılabilir veya saf SRSÜ için tohum Hibrit SRSÜ içinde ek girdi ihtiyacını karşılayabilir. Geçerli iç durum değerine saldırıyı (ör: arabellek aşımı saldırıları) etkisiz hale getirmek için, SRSÜ'nün iç durum değerinin, oturumlar süresince dahi periyodik olarak güncelleştirilmesi gerekir.

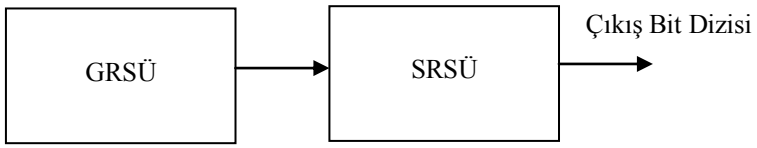


Şekil 2.6. FOGRSÜ genel tasarımı

Güvenlik değerlendirilmesi yönünden, saf bit dizisinin kısımları kullanıcı etkileşimi ile türetilirse durum daha kötü olabilir. Genellikle, FOGRSÜ için saldırganın bilgisi önemli rol oynar. Rasgele sayı üretildiğinde, bu işletim sistemi veya saldırı sistemlerinde kullanılan yapılandırma teknik sorunlarla ilgili olabilir veya bir e-posta zaman damgası rasgele sayının üretildiği zaman için kaba bir tahmin verebilir. FOGRSÜ'nün en iyi tasarımcısı veya değerlendiricisi mümkün olan bütün uygulamalarda ve çevrelerde en kötü durum senaryosu içinde bile geçerli olacak düşük entropi sınırına karar verebilmelidir. Saf bitler değiştiğinden dolayı, minimum entropi uygun tipte olmalıdır [5].

2.4.3. Hibrit Sözde Rasgele Sayı Üreteci

Hibrit Rasgele Sayı Üreteci, GRSÜ'den elde edilen rasgele sayının SRSÜ'de tohum değeri olarak kullanılmasıyla her iki sistemin birlikte çalıştığı Şekil 2.7'de gösterilen rasgele sayı üretecidir.



Şekil 2.7. Hibrit Rasgele Sayı Üreteci tasarımı

3. GERÇEK RASGELE SAYI ÜRETECİ

Rasgele sayılar kriptografik uygulamalar için çok önemlidir. Rasgele sayılar kriptografide başlangıç vektörü, özel ve gizli anahtarların oluşumunda, SRSÜ'de tohum olarak kullanılır. Bu kullanılan rasgele sayıların tahmin edilememesi, iyi istatistiksel özellikler göstermesi ve düzgün dağılımlı olması kriptografik açıdan çok önemlidir.

Bu bölümde daha çok genel ASIC silikon süreçleri veya tekrar ayarlanabilir mantıksal platformlarda (örneğin FPGA, FPAA (Field Programmable Analog Array-Alan Programlanabilir Analog Diziler), PSOC (Programmable System on Chip-Çip üzerinde Programlanabilir Sistem) gibi) üretimi için uygun GRSÜ tasarımlarına değineceğiz. Yapılmış olan tasarımların performansı, kullanılan yöntemleri ve test sonuçları ile ilgili bilgiler verilmiştir.

3.1. Gerçek Rasgele Sayı Üreteci Temel Blokları

Gerçek rasgele sayı üreteçleri rasgele bit dizisi üretmek için fiziksel süreçleri kullanan bir cihazdır. GRSÜ, üç temel bloktan oluşmaktadır.

3.1.1. Gürültü (Entropi) kaynağı

Isıl gürültüsü, saçma gürültüsü, atmosferik gürültü, yarı kararlı devreler, seğirme ve brown devinimi gibi fiziksel süreçlerden rasgelelilik elde etmeye çalışan çeşitli GRSÜ tasarımları önerilmiştir. Entropi kaynağı GRSÜ'nün en önemli bloğudur. GRSÜ'lerde kullanılan çeşitli tiplerde gürültü kaynakları bulunmaktadır. GRSÜ için entropi kaynağı fiziksel rasgelelilik kaynağıdır [20].

- **Elektriksel gürültü**

  Her elektriksel işarette, çeşitli sebeplerden kaynaklanıp rasgele değişen ve gürültü (noise) dediğimiz elektriksel işaretler de bulunmaktadır.

  - Isıl gürültü, termal veya johnson gürültüsü: İdeal olmayan iletim ortamlarında serbest elektronların ısıdan kaynaklı olarak hareketlerinin rasgeleliğinden kaynaklanır. Isıl gürültü dirençlerde gözlemlenebilir.

- Saçma gürültüsü (shot noise): Saçma gürültüsü, elektronik devrelerde elektron/foton salınımının olasılıksal olması, zamana göre düzenli olmaması nedeniyle oluşan gürültüdür.

- Titreme Gürültüsü (Flicker Shot): Elektron tüplerinde raslanılan ve spektral gücün frekansla ters orantıda değiştiği gürültüdür. Bu gürültü 1/f spektruma sahiptir. Gürültü kaynağı olarak özellikle FPGA'da bulunan lojik cihazlarda çeşitli entropi kaynakları aranmıştır. Genellikle FPGA içinde kullanılabilen rasgelelilik kaynağı olarak kullanılabilen üç doğal olay ve bunların birleşimi vardır. Bunlar lojik kapılarda bulunan gecikme değişimi, iki lojik seviye arasındaki lojik kapıların analog davranışlar ve cihazlar içinde üretilen ısıl gürültüdür.

- Yarıkararlılık, flip flop ayarlandığı ve bekleme koşulları değiştiğinde elde edilebilen diğer bir özelliktir. Kapılar kestirilemeyen lojik yükseklikler ve alçaklıklar üreten farklı osilasyonlar ve kestirilemeyen işlemlere yol açan çapraz bağlıdır.

- Seğirme yükselen kenar ve sinyalin düşme zamanları düzensiz olduğunda gürültü kaynağı olarak kullanılabilir. Seğirme deterministik ve deterministik olmayan olarak da sınıflandırılır. Tepeden tepeye seğirme, sinüsoidal seğirme, veri bağımlı seğirme ve ilişkisiz seğirme olarak bir kaç tipi vardır.

- **Fotonların kuantum mekanik özellikleri**

  Fotonlar rasgelelilik kaynağı olarak kullanılmaktadır. Fotonlar belirli durumlarda rasgele davranışlar sergiler. Örnek olarak, fotonların yarı geçirgen bir aynadan geçişleri verilebilir. Bu geçiş tamamen rasgeledir ve herhangi bir parametreden etkilenmemektedir.

- **Radyoaktivite**

  Kararsız izotopların bozulması rasgele davranış olduğu düşünülmektedir.

- **İnsan kaynaklı Etkileşimler**

  Mouse hareketleri ve klavyeden herhangi bir tuşa basılması gibi insan kaynaklı etkileşimlerde rasgelelilik kaynağı olarak kullanılmaktadır.

- **Mekanik Sistemler**

  Bir sabit sürücüdeki hava akış bozukluklarına bağlı olarak bir sabit sürücüdeki rasgele bazı verileri okuyarak üretilen zamanlama bilgilerinin incelenmesi ile ortaya çıkarılır [36].

3.1.2. Örnekleyici (Sayısallaştırıcı)

Örnekleyici, gürültü sinyalinin gerekli örneklemesini yapar ve fiziksel gürültü kaynakları için üretim mekanizması (Harvesting Mechanism) olarak adlandırılabilir. Analog sinyalden sayısallaştırılmış sinyal elde edilir. Genellikle D-tipi flip flop örnekleyici olarak kullanılır. Gerilim kontrollü osilatörde bazı gürültü kaynakları için kullanılır.

3.1.3. Son İşlem

Son işlem genellikle sinyaldeki rasgeleliliği arttırmak için kullanılır. Son işlem uygulanmış sinyal değerleri saf sinyal ile karşılaştırıldığında düzenli (uniform) dağılımlıdır. DAS rasgele sayı başına entropi artacaktır. Son işlemin aktif hata ve yan kanal analizi saldırılarından dolayı daha da önem kazanan ikinci amacı, saldırgan kurcalamaları ve çevresel değişikliklere karşı dirençli hale getirmesidir. Son işlem algoritmasına bağlı olarak üretecin güvenliği artacaktır. XOR doğrulama, Von Neumann doğrulama, extractor fonksiyon, özet fonksiyonu ve resilient fonksiyonlar gibi çeşitli son işlem algoritmaları uygulanmıştır [15,16,37-39].

3.1.3.1. Kriptografik Özet Fonksiyonu

Kriptografik özet fonksiyonu çeşitli güvenlik özelliklerini sağlayan bir özet fonksiyonudur. Veriyi belirli uzunlukta bir bit dizisine, (kriptografik) özet değerine, dönüştürür. Bu dönüşüm öyle olmalıdır ki verideki herhangi bir değişiklik özet değerini değiştirmelidir.

İdeal bir kriptografik özet fonksiyonu şu dört özelliği sağlamalıdır:

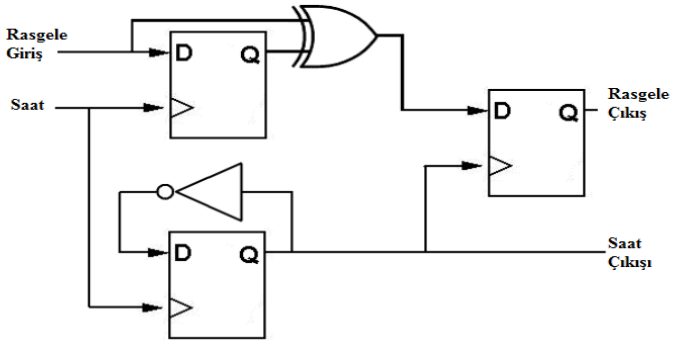
- Herhangi bir mesaj için özet hesaplamak kolay olmalıdır.
- Bir özete karşılık gelecek mesajı oluşturmak zor olmalıdır.
- Özeti değişmeyecek şekilde mesajı değiştirmek zor olmalıdır.
- Aynı özete sahip iki farklı mesaj bulmak zor olmalıdır.

Özet fonksiyonlarının karıştırma ve parçalama özeliklerinden dolayı, etkili bir biçimde son işlem fonksiyonu olarak kullanılır. Sha-1 (160 bit) veya Md5 (128 bit) gibi

kriptografik olarak güçlü özet fonksiyonlarının GRSÜ çıkışında çalıştırılmasıyla uygulanmış en popüler ve en sağlam son işleme tekniğidir. Intel RSÜ'de Sha-1 kullanmıştır. Bu algoritmaların asıl problemi hesaplamasal olarak çok ağır olmalarıdır.

3.1.3.2. XOR Doğrulama

Şekil 3.1'de gösterilen XOR doğrulama, bir çıkış biti üretmek amacıyla, elde edilen bit dizisi n bit (n=2) bloklara ayrılmıştır. Ayrılan bloklar kendi içinde XOR işlemine tabii tutulmuştur. Çıkış bitindeki bias'ı giderirken bir yandan da çıkış bit verimini 1/n kez azalmasına neden olacaktır, Ancak çıkış bit dizisindeki bias giriş bitlerinin bağımsız olması şartıyla düşecektir. Bu doğrulamanın avantajı, basitliği ve sabit çıkış bit hızını sağlama olasılığıdır [16].



Şekil 3.1. XOR son işlemi (n=2) [37].

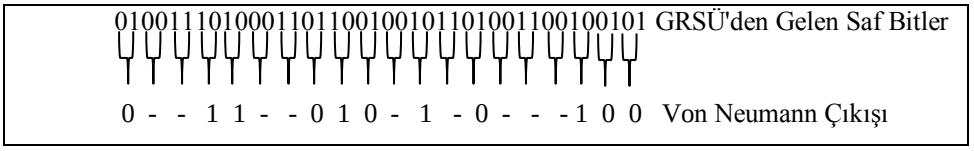
3.1.3.3. Von Neumann Doğrulama

En eski ve en basit son işleme yöntemidir. Bit dizisindeki düzensizlikleri giderir. Tablo 3.1'de gösterildiği gibi Von Neumann düzgün dağılımlı 0 ve 1 bitleri üretir. GRSÜ'den gelen eşzamanlı çiftleri dikkate alır. Eğer bit dizisi (1, 0) ise 1 biti üretir, eğer bit dizisi (0, 1) ise 0 biti üretir. (0, 0) ve (1, 1) bit dizileri atılır. Tablo 3.2'de bit oranındaki değişim gösterilmiştir. Bu doğrulama entropiyi ideal değer 1'e yaklaştırarak ürettiği bitler ile entropinin iyileştirilmesine katkıda bulunur, ancak GRSÜ'den gelen bazı bit dizilerinin atılmasından dolayı Von Neumann çıkış bit hızı GRSÜ'nün çıkışına bağlıdır ve bundan dolayı sabit değildir. Bit hızı giriş bit hızının 1/4'ü kadar azaltılır [15,38].

Tablo 3.1. Von Neuman doğrulama

Girilen Bit Çiftleri	Von Neumann çıkışı
00	Çıkış yok
01	0
10	1
11	Çıkış yok

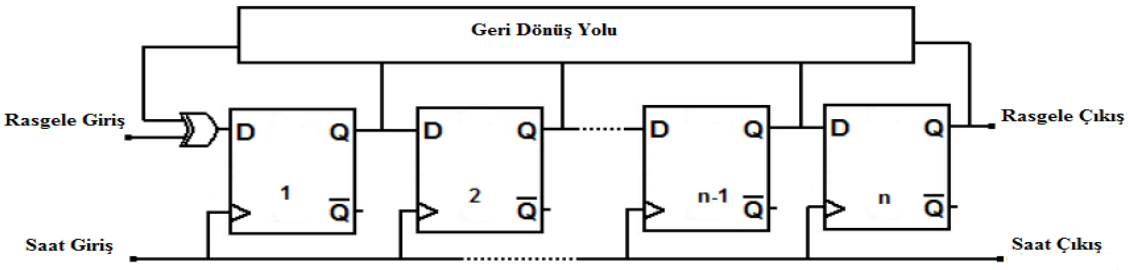
Tablo 3.2. Von Neumann'dan dolayı bit oranındaki değişim



3.1.3.4. Doğrusal Geri Beslemeli Kayan Yazmaç

Şekil 3.2'de gösterilen Doğrusal Geri Beslemeli Kayan Yazmaç **(**DGBKY) birçok rasgele bit dizisi üretmede kullanılmıştır [39]. Bunun için çeşitli sebepler vardır:

1. DGBKY'nın donanım üzerinde uygulanmaları kolaydır.
2. Uzun periyotlu ve iyi istatistiksel özelliklere sahip diziler üretebilirler.
3. Cebirsel teknikler kullanılarak kolayca analiz edilebilirler.



Şekil 3.2. Doğrusal geri beslemeli kayan yazmaç [37].

Bir L uzunluğundaki DGBKY verinin hareketini kontrol eden bir saat, bir giriş ve bir çıkışa sahip ve bit bit saklama yeteneğine sahip L gecikme elemanlarından oluşur. Her bir zaman biriminde aşağıdaki işlemleri gerçekleştirir:

- İlk gecikme elemanın içeriği çıkış dizisinin bir bölümünü oluşturur.
- i. Elemanın içeriği her bir i $1 \leq i \leq L-1$ için i-1 aşama hareket ettirilir.
- Son gecikme elemanın içeriği elemanlarının sabit altkümesinin bir önceki içeriği mod 2 ile birlikte eklenerek hesaplanan geribildirim bitidir.

3.1.3.5. Extractor Fonksiyon

Extractor fonksiyonu, değişen çevre koşullarına karşı GRSÜ daha sağlam olması için Barak, Shaltiel ve Tomer tarafından önerilmiştir [40]. Bu fonksiyon ölçülebilir özellikleri ile güçlü bir fonksiyondur. Extractor fonksiyon başlangıçta karmaşıklık teorisi için bir araç olarak geliştirilen ölçülebilir özelliklere sahip güçlü bir stateless fonksiyondur. [40]'da, rasgelelilik kaynağı üzerinde bir saldırganın etkilerini yakalamak için matematiksel bir model önerildi ve giriş kayağında sebebi belli olmayan korelasyon varsa bile onun çıkış özellikleri için kanıtlanmış özet fonksiyonlarına dayalı açık bir yapı vermektedir.

3.1.3.6. Resilient (esnek-direnç) Fonksiyon

Deterministik bitlerin filtrelenmesidir. Sunar, Martin ve Stinson tarafından yapılan bir çalışmada, etkilenen deterministik bitler resilient fonksiyonunun tolerans özelliklerini incelemek için kullanılmıştır [41]. Resilient fonksiyonunun yüksek esneklik derecesi ve örneklemeden beklenen deterministik bitlerin sayısı aktif düşman için GRSÜ'nün tolerans miktarını belirler.

3.1.3.7. H Son İşlem Fonksiyon

Şekil 3.3'te gösterilen H son işlem fonksiyonu, Dichtl tarafından GRSÜ'deki biası önlemek amacıyla önerilmiştir [17]. Quasigroup dizi dönüşümüne dayalıdır. Son işlem algoritması 8 bit çıkış elde etmek amacıyla fiziksel rasgelelilik kaynağından 16 bitini kullanır.

$a_0,a_1,.....,a_{15}$ son işlem için giriş bitleridir. $b_i=a_i \oplus a_{(i+1) \bmod 8}$ ile $b_0,b_1,.....,b_7$ 8 bit tanımlanır. $b_0,b_1,.....,b_7$ için sadece 128 farklı değer mümkündür. Giriş mükemmel bir rasgele sayı üreteci olduğunda, entropinin bir biti yok edilir. Son işlem fonksiyonunun $c_0,c_1,.....,c_7$ çıkışı $c_i=b_i \oplus a_{i+8}$ ile tanımlanır. Bu fonksiyon H fonksiyonu olarak adlandırılmıştır. Dichtl b dizisini elde etmek için sistemi geliştirmiştir.

Girişteki 16 bitlik dizi a1 ve a2 byte olarak bölünmüştür. a1 byte RL(a1,1) ile 1 bit sola kaydırılmıştır.

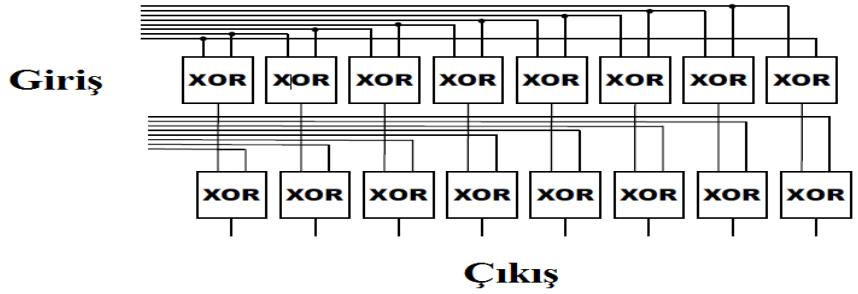
$$H(a1,a2)=a1 \oplus RL(a1,1) \oplus a2 \quad (3.1)$$

Dichtl aşağıdaki aşamaları takip ederek H fonksiyonunu daha da geliştirmiştir.

$$H_2(a1,a2)=a1 \oplus RL(a1,1) \oplus RL(a1,2) \oplus a2 \quad (3.2)$$

ve

$$H_4(a1,a2)=a1 \oplus RL(a1,1) \oplus RL(a1,2) \oplus RL(a1,4) \oplus a2 \quad (3.3)$$



Şekil 3.3. H son işlem fonksiyon [17].

3.1.3.8. Önerilen Son İşlem

Von Neumann tarafından önerilen sistemde atılan 00 ve 11 bitleri yerine 00 bitleri 0, 11 bitleri 1 olarak alınmıştır. Tablo 3.3'te son işlem çıkışı ve Tablo 3.4'te bit oranındaki değişim verilmiştir. Bu şekilde alınması ile Von Neumann son işleminde bit dizisi 1/4 oranında azalırken önerdiğimiz son işlemde azalma 1/2 oranında meydana gelmiştir. Bu işlem sonrasında yapılan testlerde başarılı sonuçlar elde edilmiştir.

Tablo 3.3. Önerilen son işlem çıkışı

Girilen bit çiftleri	Önerilen son işlem çıkışı
00	0
01	0
10	1
11	1

Tablo 3.4. Önerilen son işlemden dolayı bit oranındaki değişim

0100111010001101100100101101001100100101 GRSÜ'den Gelen Saf Bitler
0 0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 Önerilen son işlem çıkışı Çıkışı

3.2. Bir GRSÜ'den Beklenen Özellikler

Birçok GRSÜ tasarımı önerilmiştir [7,34,39]. Bu tasarımlar entropi kaynakları ve üretim tekniklerine göre önemli ölçüde değişiklikler gösterebilirler. Her tasarımın güçlü ve zayıf yanları vardır. Bu özelliklerin bazıları performans ile ilgili bazıları ise güvenlik ve sağlamlıkla ilgilidir. Bir GRSÜ'den beklenen özellikler özetle aşağıdakiler söylenebilir [41]:

- Pratik açıdan bakıldığında, GRSÜ'ler genelde mevcut ucuz silikon yöntemleri kullanılarak kurulmuştur. Dahası sadece sayısal tasarım teknikleri kullanılarak GRSÜ uygulaması gerçekleştirilmesi istenir. Bu sayısal mikroişlemciler ile kolay entegrasyon için izin verir ve popüler ayarlanabilir platformlarda (FPGA vs.) GRSÜ uygulamaları mümkün olmaktadır.
- Üretim mekanizması (örnekleyici) basit olmalı (analizi kolay), korunmalı ve entropi kaynağını en iyi şekilde örneklemelidir. Bir başka deyişle, GRSÜ'nün tahmin edilmemesi üretim mekanizmasının karmaşıklığına dayalı olmamalıdır, ancak sadece entropi kaynağının tahmin edilememesine bağlıdır.
- Enerji tüketimi ve alan başına yüksek çıkış hızı ile etkili ve verimli tasarım sağlanmalıdır. Mümkünse kuvvetlendiriciler ve diğer analog bileşenlerin kullanımından kaçınılmalıdır. Analog bileşenler fazla enerji harcamaktadırlar ve analizleri zor olmuştur. Dikkat edilmelidir ki, analog bileşenler kullanılmadığından bu yana, gerilim seviyelerindeki değişikliklerden ziyade zaman domainindeki değişikler örneklenmek zorunda kalınmıştır. Takip edildiği takdirde, bu uygulama karışık son işlem uygulamasından veya en azından bir yazılım uygulamasından kaçınmak anlamına gelir.
- Tüm varsayımların deneysel doğrulanması ile entropi toplama mekanizmasının matematiksel doğrulamasının olması istenilen bir durumdur. Tasarım iyi bir şekilde analizini sağlamak için yeterince basit olmalıdır. GRSÜ'nün çıkışını onaylamak için Diehard ve NIST test paketleri çoğunlukla kullanılmaktadır.

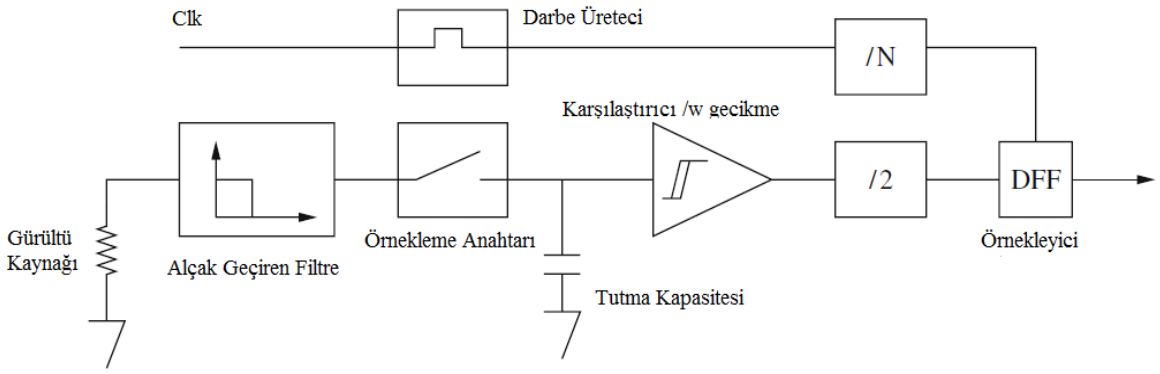
3.3. Literatürdeki GRSÜ Tasarımları

Şekil 2.5'de görüldüğü gibi, GRSÜ tipik uygulanması üç bölümden oluşur. İlk olarak, Bir FPGA'nın tipik olarak seğirme veya yarı kararlılık durumunda fiziksel rasgele gürültü kaynağıdır. İkinci olarak, sayısallaştırıcı, entropi kaynağından rasgelelilik toplar ve ikili bit dizisi üretir. Üçüncü olarak son işlem, korelasyonu azaltarak ve veri sıkıştırması yaparak, rasgele bit dizisinin istatistiksel özelliklerini geliştirir.

Birçok GRSÜ uygulamaları önerilmiştir [5] ve bu bölümde, uygulamalarda kullanılan gürültü kaynağı ve sayısallaştırıcı devre üzerine odaklanılmıştır.

3.3.1. Bagini ve Bucci Tasarımı

Şekil 3.4'te görüldüğü gibi, Baggini ve Bucci [42] tarafından ilk olarak önerilen tasarım beyaz gürültü örnekleme ve kuvvetlendirme için analog ve sayısal bileşenlerin birleşimini kullanır. Tasarım çalışma koşulları ve bileşen davranışlarındaki değişimlere karşı dirençlidir. [42]'de çıkış bit hızı ile maksimum bit korelasyonu arasındaki ilişkiyi yakalayan GRSÜ için analitik model verilmiştir. Referansta uygulama sonucu ile ilgili bilgi bulunmamaktadır [5].

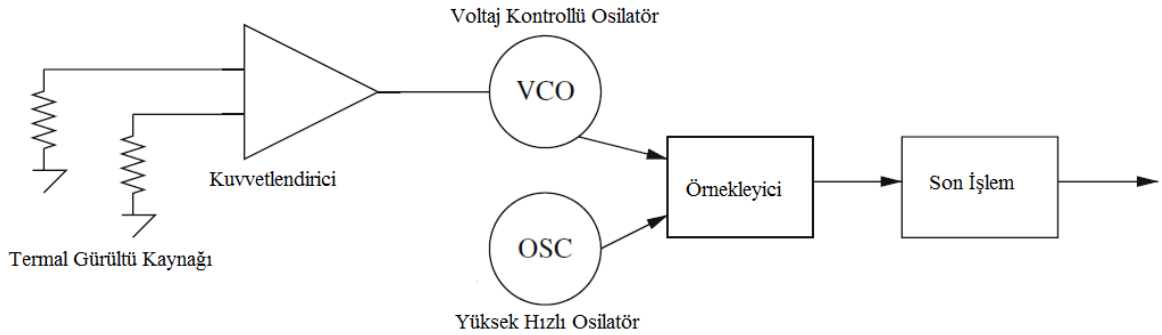


Şekil 3.4. Baggini ve Bucci GRSÜ tasarımı [42].

3.3.2. Intel GRSÜ Tasarımı

Intel GRSÜ [43] tasarımı Şekil 3.5'de gösterilmiştir. Tasarımın entropi kaynağı bir bağlantı (birleşim-jonksiyon) yerindeki ısıl gürültüdür. Tasarım güç kaynağı ve çevresel değişikliklere karşı sağlam olmak amacıyla farklı konfigürasyonlarda iki direnç kullanır.

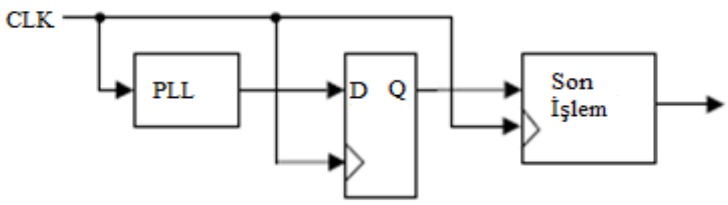
Farklı ısıl gürültü kuvvetlendirilmiş ve gerilim kontrollü osilatörün girişinde kullanılmıştır. Daha sonra hızlı bir osilatör, gerilim kontrollü osilatörün çıkışında oluşan yavaş osilatör ile örneklenmiştir. Örnekleme sonucu elde edilen çıkış bit dizisi tam olarak rasgelelilik özelliliği gösteremediğinden dolayı Von Neumann kullanılarak son işleme tabi tutulduktan sonra Sha-1 kullanılarak karıştırılmıştır. Güvenlik ölçüsünü arttırmak amacıyla FIPS 140-1 testine tabi tutulmuştur. Bu test sonucunda herhangi bir zayıflık tespit edilmemiştir.



Şekil 3.5. Intel GRSÜ tasarımı [43].

3.3.3. The Fischer–Drutarovsk´y Tasarımı (PLL'e Dayalı)

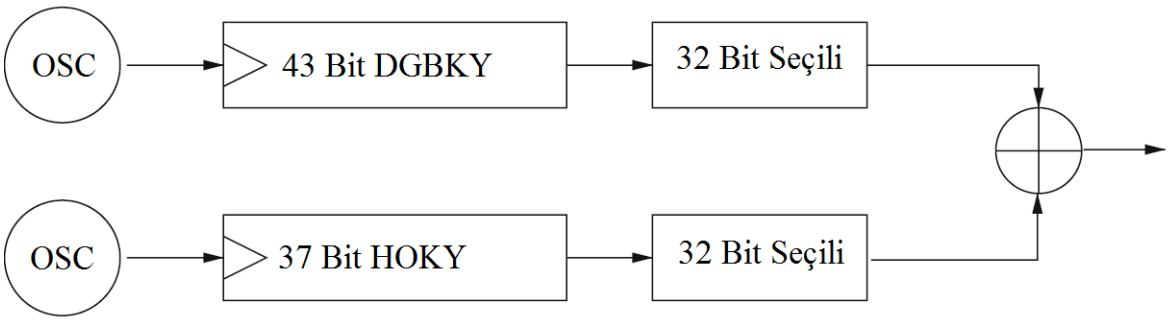
Şekil 3.6'da gösterilen tasarım 2002'de Fischer ve arkadaşları [44] tarafından, Altera FPGA'lerde PLL'ler analog entropi kaynağı olarak GRSÜ'de kullanılmıştır. PLL'den gelen sinyal seğirir ve saat girişi ile clocked edilen D tipi flip flop'ta örneklenmiştir. Saat girişinin frekansı ve PLL'nin çıkışı aralarında asal olarak seçilmiştir. Decimator (son işlem) devresi rasgele sinyalin deterministik bölümlerini ayırmak için kullanılır. Bit oranını artırmak için; bu GRSÜ'den elde edilmiş örneklenmiş çıkışlar XOR (Exclusive OR) yaparak uzatılabilir. Kolay tasarım ve uygulamaları, sentezin bütünüyle FPGA araçları içinde yapılabiliyor olması, sabit çıkış hızına sahip olması ve düşük güç tüketimi bu tasarımın avantajları olarak gösterilebilir. Düşük çıkış hızı ve bu GRSÜ tasarımının, analog PLL içeren FPGA'lerde sınırlandırılmış olması dezavantajlı yanıdır. Ayrıca Resilient doğrulayıcı kullanılmasından dolayı üretece yapılacak saldırılar tespit edilemeyecektir. Bu tasarımda elde edilen çıkış hızı 70 Kbits/s'dir. Tasarımda elde edilen bit dizilerinin istatistiksel davranışı NIST testi kullanılarak doğrulanmıştır.



Şekil 3.6. Fisher'in GRSÜ prensibi [44]

3.3.4. Tkacik GRSÜ Tasarımı (Durum Makinelerine Dayalı)

2002'de, Tkacik [45] iki bağımsız serbest çalışan osilatör halkaları tarafından zamanlanan, doğrusal geri beslemeli kaydıran yazmaç (DGBKY-LFSR) ve hücresel otomat kaydıran yazmaç (HOKY- CASR) kullanarak bir GRSÜ önermiştir. Kullanılan durum makineleri farklı uzunluklara sahiptir ve bu makinelerden çıkış üretimi için sadece 32 biti kullanılmıştır. Kullanılan DGBKY (2^{43}-1) döngü uzunluğu sağlayan 43bit uzunluğunda polinoma dayalıdır. HOKY ise (2^{37}-1) döngü uzunluğu sağlayan 37bit uzunluğundadır. Şekil 3.7'de görüldüğü gibi 32 bit rasgele sayı üretimi için DGBKY ve HOKY'ın 32 biti seçilmiştir ve bit bit mod iki XOR işlemi ile birleştirilmiştir. Durum makinelerinin uzunluğu birleştirilen üretecin döngü uzunluğunu $2^{\text{toplamuzunluk}}$-1(2^{80}) yaklaştırmak amacıyla aralarında asal seçilmiştir [37]. HOKY/DGBKY yapıları hafıza elemanları içerdiği için, çıktı bitleri korelâsyonludur. Bu nedenle bağımsız rasgele bit elde etmek için her çıkış biti örnekleri arasındaki DGBKY kaydının iki katından daha fazla uzunlukta bir gecikme olmamalıdır. XOR'dan önce permutasyon, HOKY ve DGBKY'nin geri besleme polinomlarının bilindiğin varsayımı altında bu GRSÜ'ye saldırı Dichtl [46] tarafından sunulmuştur.

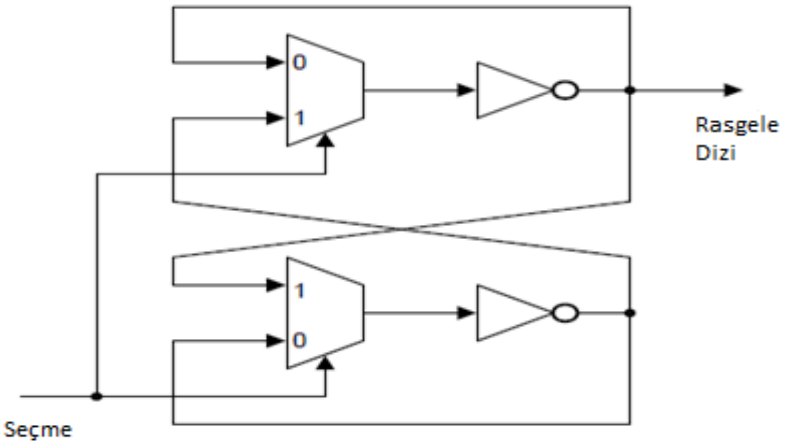


Şekil 3.7. Tkacik'in GRSÜ prensibi [45].

Tasarımın, kolay uygulanması, sentezin bütünüyle FPGA araçları içinde yapılabiliyor olması, sabit ve yüksek çıkış hızına sahip olması ve çıkış bitlerine herhangi bir son işlem uygulanmıyor olması avantajlı yanıdır. Ancak sadece iki osilatör kullanılıyor olması sebebiyle entropi kaynağı sınırlıdır. Bunun sonucu olarak, DGBKY ve HOKY sadece iki düşük entropili osilatör ile tohumlanmış bir SRSÜ gibi davranmaktadır. Tasarımın doğrusal bileşenler kullanması nedeniyle bir saldırgan doğrusal bir model yapabilir ve çözümleyebilir.

3.3.5. Epstein GRSÜ Tasarımı

2003'te Epstein ve diğerleri Şekil 3.8'de gösterilen, iki durumlu bellek elemanlarına dayanan, iki çoklayıcı ve tümleyenden (değil kapısından) oluşan bir GRSÜ sunmuştur. Seçili sinyal lojik sıfır durumundayken, devre yarı kararlı hale çevrilen iki tek tümleyen osilatör halkalarına dönüşür. Seçme sinyali yükseldiğinde, devre iki osilatör halkasın çıktısını çözer ve lojik sıfır ya da lojik birde kararlı hale getirir ve böylece rasgelelilik yaratır. Çoğu hafıza elemanlarının çıkışı, XOR kullanılarak üretilmiş son çıkış bit dizisi ile birbirine bağlanır [47].

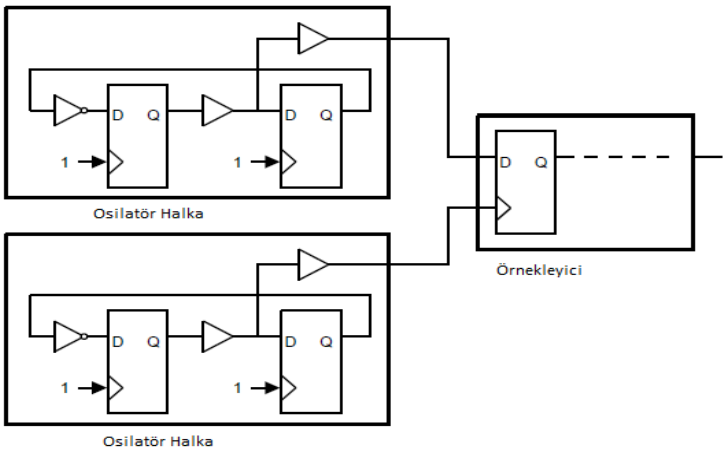


Şekil 3.8. Epstein'in GRSÜ tasarımı

Tüm çıkış dizileri Von Neumann son işlemi uygulandıktan sonra Diehard testini geçmiştir. Sadece sayısal bileşenlerle kurulmuştur, yani tekrar kontrol edilebilir lojik üzerinde uygulanabilir.

3.3.6. Kohlbrenner GRSÜ Tasarımı

Şekil 3.9'da gösterildiği gibi 2004'te Kohlbrenner ve arkadaşları [48] GRSÜ tasarımında her biri iki açık tutucu, bir tampon ve bir tümleyen oluşan osilatör halkası kullanılmıştır. Entropi kaynağı olarak osilatör halkadan üretilen saat sinyalindeki seğirme kullanılmıştır. Halka osilatör saatlerindeki seğirme halka osilatör döngüsü içeren mantık kapılarının kararsız yayılma gecikmelerinden ileri gelmektedir. Örnekleyici devre, diğer halka çıkışı saat girişi ile bağlı iken osilatör halkalardan birinin çıkışı veri girişine bağlı olan D flip-flop'tur. Rasgele bitin çıkışına örnekleyici devre karar verir. Bu kurulumun doğru çalışması için; iki osilatör halkasının sıklığı, tamamen aynı veya doğru bir şekilde eşleşmiş olmalıdır. Bu GRSÜ'nün çıkışı XOR son işlemine tabi tutulmuştur. Çıkış oranı düşüktür, elde edilen bit 1 Mbit/s den daha azdır. Çıkış dizisi istatistiksel olarak NIST testine tabi tutularak doğrulanmıştır.

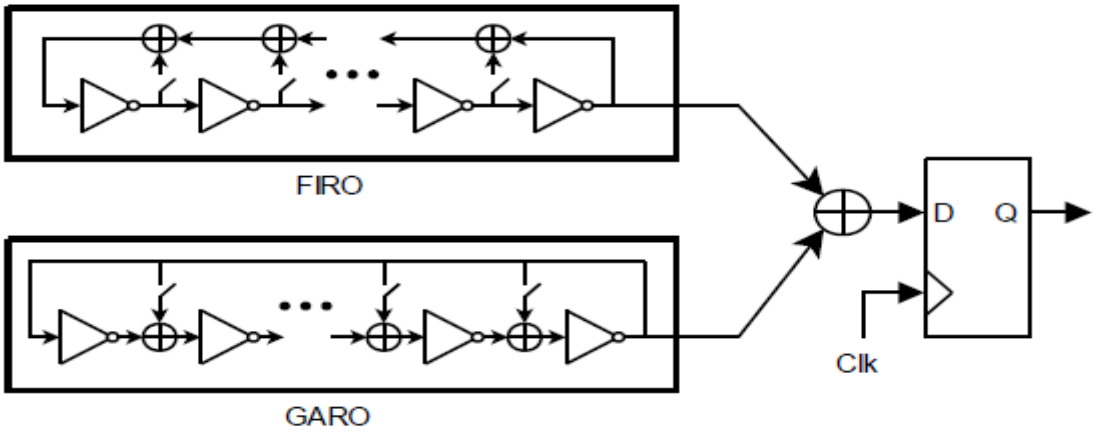


Şekil 3.9. Kohlbrenner GRSÜ tasarımı [48].

Bu tasarımın avantajı, bütün FPGA'lerin ortak kaynaklarını kullanması, matematiksel model uygulanabilmesi, çok az lojik kaynaklar kullanması ve düşük güç tüketimidir. Ancak bütün durumlarda üretecin doğru çalışmasını garanti etmek için elle yerleştirme gereklidir ve son işlem uygulanması dezavantajlı yanıdır.

3.3.7. Figaro GRSÜ Tasarımı

2006'da Goli´c [49] Galois osilatör halkası (GOH) ve Fibonacci osilatör halkası (FOH)'nı kullanarak GRSÜ önermiş, ilgili tasarım Şekil 3.10'da gösterilmektedir. Bu doğrusal geri beslemeli kayan yazmaç (DGBKY) benzeri yapı, kayıt elemanları yerine gecikme elemanları olarak tümleyenler kullanır ve anahtarlar geri besleme polinomu uygulanırken ya kapalıdır ya da açıktır. Eğer, geri besleme belirli gereksinimleri sağlarsa [49], bu yapıda sürekli osilatörler elde edilir. FOH ve GOH'dan çıkışlar XOR ile birleştirilir ve D flip-flop ile XOR'un çıkışı örneklenerek rasgele diziler üretilir. FOH / GOH yapıları gürültü sinyaline benzer ve anolog davranış gösterir ve bundan dolayı bu yapılar FPGA'nın içindeki diğer sinyallere karşı duyarlıdır.



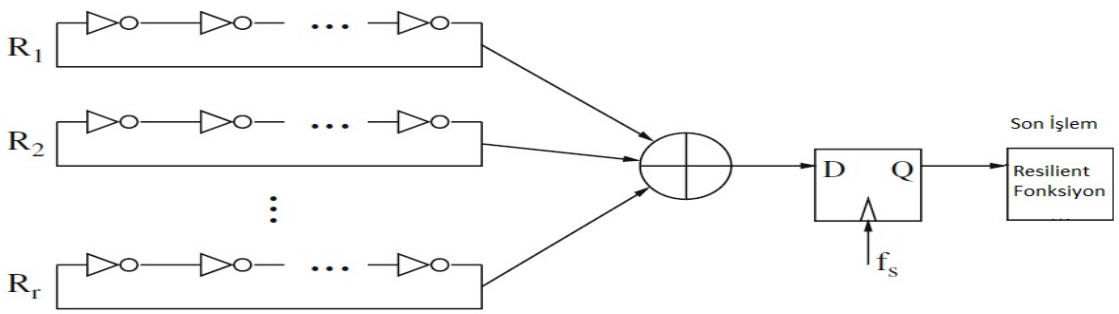
Şekil 3.10. Golic GRSÜ tasarımı [49].

Çıkış bitlerinde meydana gelebilecek olan korelâsyonu önlemek amacıyla nispeten daha düşük seçilen örnekleme frekansı kullanılmıştır. Golic saat kontrollü DGBKY'a dayalı rasgele veri son işlemi için bir yöntem önermiştir [50]. Önerilen son işlem rasgelelilik çıkarımı için ve çıkış hızının daha verimli, güvenli elde edilmesi için kullanıldı. Bu tasarımda elde edilen bit hızı 12,5 Mbit/s olmuştur.

Figaro tasarımının, bütün FPGA'lerde kolay tasarım ve uygulamaları, sentezin bütünüyle FPGA araçları içinde yapılabiliyor olması, sabit ve yüksek çıkış hızına sahip olması, az miktarda lojik kaynak kullanması ve düşük güç tüketimi avantajlı yanıdır. Ancak matematiksel bir model olarak rasgelelilik tanımı zordur. Çıkışın düzensizliğinin çok olması ve ataklara karşı üretecin sağlamlığının düşük olması dezavantajıdır.

3.3.8. Sunar GRSÜ Tasarımı (Halka Osilatöre Dayalı)

Şekil 3.11'de gösterilen tasarım Sunar ve arkadaşları [41] tarafından 2007 yılında önerilmiştir. Temel olarak serbest salınımlı halka osilatörlerin çıkışları XOR işlemi ile birlikte toplanır ve sonra örneklenir. Rasgelelilik kaynağı faz seğirmesidir. Halka osilatör tek sayıda tümleyen içeren gecikmelerin birleşimsel döngüsüdür. Bu döngüdeki salınımı sağlar. Kapalı döngü içinde her bir mantık kapısının yayılım gecikmesindeki kararsızlık halka osilatör saatte seğirme üretir. Bu tasarımda her biri 13 tümleyenden oluşan büyük sayılarda halka osilatör (seçilen Xilinx Virtex-2 FPGA'de 114 adet) kullanılır. Kullanılan halka osilatörlerin sayısı, ölçülür seğirmelere göre belirlenir. Tüm halka osilatörlerinin çıkışları yüksek frekanslı rasgele sinyal almak için XOR işlemine girişi yapılır. XOR işleminin çıkışı da düşük frekanslı saat frekansı ile örneklenir. Daha sonra resilient fonksiyon son işlemi kullanılarak, rasgele sinyaldeki birler ve sıfır arasındaki orantısızlıklar düzeltilebilir. Bu tasarımın bir uygulaması Schellenkens ve arkadaşları [51] tarafından gerçekleştirilmiştir. Bu GRSÜ'de son işlemi de içererek başarılan bit oranı 2.5Mbit/s'dir. Çıkış bit dizileri Diehard ve NIST testleri kullanılarak doğrulanmıştır. Bu tasarım Dichtl ve Golic [52] tarafından eleştirilmiştir. Bu eleştiriler arasında örnekleme oranı ve halka osilatörlerin bağımsızlık varsayımı vardır. Örnekleme hızı kolay düşürülmüş olsa da, çok sayıda halka kullanıldığında halka osilatörlerin bağımsızlığını doğrulamak oldukça zordur. Küçük sayıda halka için, dikkatli yerleşim ve yönlendirme ile diğerleri ile etkileşimden halkalar yeterince izole edilebilir. 2008'de, Wold ve arkadaşları [53] Sunar tasarımına her bir osilatör halkasına [52]'de belirtildiği gibi D flip-flop ekleyerek ve XOR'lar arasındaki hız sorununu gidererek bir donanım önerdi. Burada önerilen GRSÜ'de sadece 25 halka osilatör kullanılarak istatistiksel testleri geçmiştir. Elde edilen çıkış hızı 100 Mbit/s olmuştur.

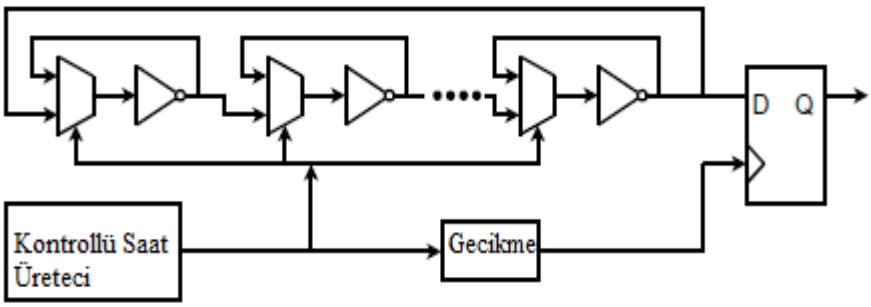


Şekil 3.11. Sunar GRSÜ tasarımı [41].

Halka osilatör tabanlı GRSÜ'lerin avantajı, bağımsız teknoloji (bütün FPGA'ler için uygundur), kolay tasarım ve uygulamaları, sentezin bütünüyle FPGA araçları içinde yapılabiliyor olması, nispeten sabit ve yüksek çıkış hızına sahip olmalarıdır. Ancak halka osilatör sayısının çokluğundan dolayı yüksek güç tüketimi, halka osilatörlerin bağımsız olmaması rasgelelilik kalitesinin düşmesine ve çıkışın korelasyonlu olması, resilient fonksiyonundan dolayı saldırıların tespit edilemeyecek olması, güç tüketimi dolayısıyla aşırı lokal ısınmalar tasarımın dezavantajlarıdır.

3.3.9. Vasyltsov GRSÜ Tasarımı (Yarı Kararlı Flip Flop Dayalı)

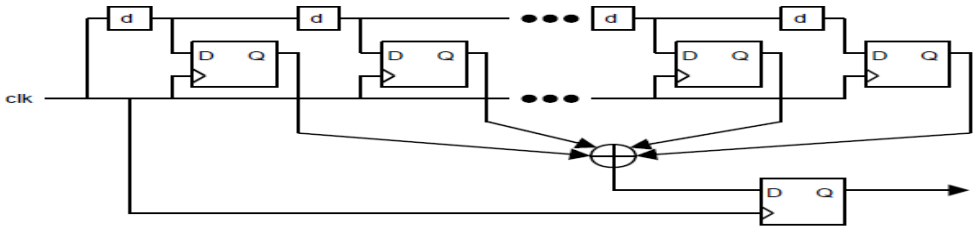
Vasyltsov ve arkadaşları tarafından 2008'de [54], Şekil 3.12'de görüldüğü gibi her aşamada yalnızca bir çoklayıcı ve bir tümleyen (DEĞİL kapısı) içeren, 5 aşamalı yarı kararlı halkaya dayanan bir GRSÜ önermişlerdir. Çoklayıcılar şu şekilde kontrol edilir; ilk aşamada beş tümleyenden her biri, bir döngü içerisine bağlanır (Giriş- çıkış) böylelikle yarı kararlı duruma sokulurlar. İkinci aşamada; beş yarı kararlı tümleyen, osilatör halkasına dayanan başlangıç değerleri ile bağlanmıştır. Yarı kararlı devre konfigürasyonları arasında geçiş fikri ve salınan yapılandırmalar Epstein ve diğerlerine dayanmaktadır [47]. Rasgele bit ikinci aşamanın sonunda, halka çıkışında örneklenir. Sonuç olarak GRSÜ'nin FPGA veya Uygulamaya Özgü Tümleşik Devre'de (UÖBD-ASIC) küçük ve hızlı uygulanmasıdır, fakat sentez işlemindeki optimizasyon FPGA kurulumunda zorluklara yol açar ve sayısal mantığın FPGA'de içerisindeki yeri önemlidir. Bu kurulum, FPGA içerisindeki kaynakların çok azını kullanır ve rapor edilen bit oranı 140 Mbit /s'dir.



Şekil 3.12. Vasyltsov GRSÜ tasarımı [54].

3.3.10. Danger GRSÜ Tasarımı

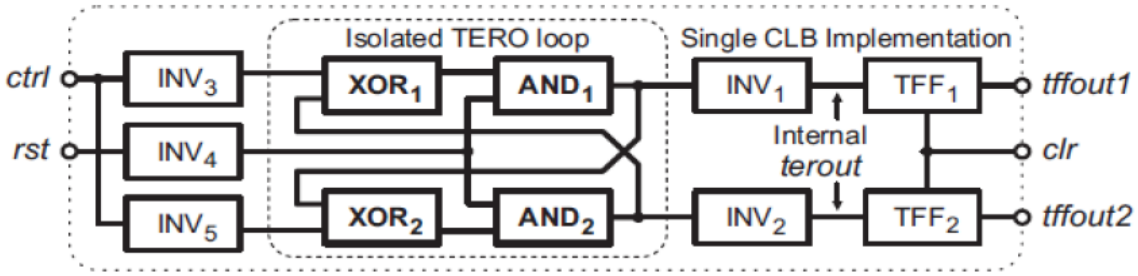
2009'da Danger ve arkadaşları [55], FPGA'de açık döngü yarı kararlılığına dayanan bir GRSÜ önermişlerdir. Bu tasarımın prensibi gecikme dizisidir, dizideki birkaç kademe noktasından örneklenmiş sinyaller birlikte XOR edilir ve ardından rasgele bir sinyal oluşturmak için örneklenir, bu durum Şekil 3.13'te gösterilmiştir. Elemanlar arasındaki gecikme önemli ve bu duruma karşı özel özen gösterilmelidir. Bu GRSÜ'nün rapor edilen bit oranı 20 Mbit/s'dir.



Şekil 3.13. Danger ve diğerlerinin GRSÜ prensibi [55].

3.3.11. Geçiş Etkili Halka Osilatöre Dayalı GRSÜ Tasarımı

Bu tasarım [56] FPGA'de gerçekleştirilebilen geçiş etkili halka osilatör adında yeni bir yüksek entropili sayısal devre içerir. Geçiş Etkili Halka Osilatör (Transition Effect Ring Oscillator-TERO) bilerek uzatılmış geri bildirim yolları ile iki kararlı flip flop çeşididir. TERO yarı kararlılık olayını çözümlüyor iken üreteç rasgele bit üretir.



Şekil 3.14. Tero GRSÜ tasarımı [56].

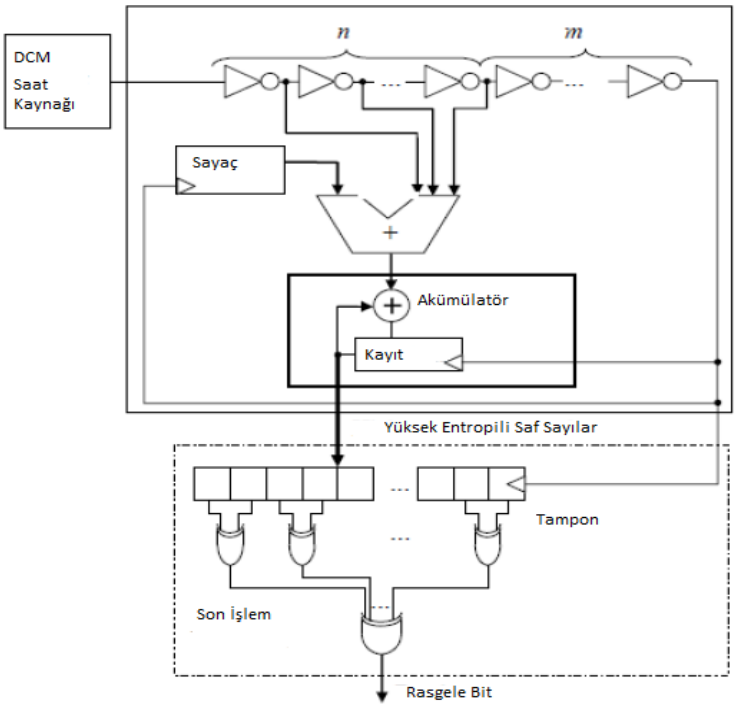
TERO tasarımı ctrl sinyaline göre tampon (buffers) veya tümleyen gibi davranan iki XOR kapısı içerir. Eğer ctrl=1 ise her iki XOR kapısı da tümleyen gibi davranır ve aksi durumda iki XOR kapısı da tampon gibi davranır. Bu iki şart için burada döngü üzerinde

hiç bir salınım olmamalıdır. Ancak ctrl sinyalinin yükselen kenarında iki XOR kapısının her ikisi de tümleyen olarak davranır ve onların çıkışlarını değiştirmeye çalışır. Bu hareket döngünün kararlı hal durumunu rahatsız eder ve sinyal döngüde yükselir. Bu sinyal küçük geçici periyotlarda yok olur. T tipi flip flop (TFF) kullanılan döngü TERO'nun bu geçici etkisinden rasgelelilik çıkarır. Döngüde salınımların sayısı çift ya da tek yapıldı ise TFF çözülebilir. Rst ve ctrl sinyalleri, iki ardışık bit arasındaki korelâsyondan korunmak amacıyla her bir üretim için sıfır tero'yu başlatmak için kullanıldı. Çıkış hızı 250 Kbit/s olmuştur. Üretilen bit dizisi herhangi bir son işlem uygulanmadan NIST testinden başarı ile geçmiştir. Bu tasarımın avantajı, bütün FPGA'lerin ortak kaynaklarını kullanması, çok az lojik kaynaklar kullanması ve düşük güç tüketimidir, ancak bütün durumlarda üretecin doğru çalışmasını garanti etmek için elle yerleştirme gereklidir [37].

3.3.12. Parazitlenme Etkisine (crosstalk effect based design) Dayalı GRSÜ Tasarımı

Cret ve arkadaşları [57,58] tarafından FPGA'de parazitlenme etkisine (crosstalk effect based design) dayalı GRSÜ uygulanması için yeni bir yöntem sunmuşlardır. Yazarlar, her biri Xilinx FPGA'de bulunan özel elde zincirlerinde parazitlenmenin neden olduğu ağır yük işleminin çeşidini denediler. Elde zinciri, verimli aritmetik fonksiyonlar oluşturmaya izin veren komşu lojik hücreler arasındaki hızlı özel hatların en yaygın türüdür. Parazitlenme, elde zincir uzunluğunun bir bölümü olan eşik aşıldıktan sonra ortaya çıkmaya başlar. Tasarımda entropi kaynağı olarak parazitlenmenin bu türlerini kullanmaya başlamışlardır.

Tasarımın mimarisi Şekil 3.15'de gösterildiği gibi sistem saati tarafından ileri sürülen tümleyenin bir zincirinden oluşur. Tümleyenin çıkışları (n+m) hızlı zincir hattı üzerinden akan akımı temsil eder. Tümleyen çıkış verileri bir sayaç değerine ilave edilir ve böylece akümülatör içinde toplanır. m'in artması ile aşılan eşik'ten sonra akümülatörde elde edilen sonuç parazitlenme ve bağlantı ağı gibi görünen diğer elektrostatik veya manyetik müdahalelerden dolayı saat çevrimlerinin sabit bir miktarının her bir çalışmasının sonunda farklı olur. NIST testlerini başarıyla geçmiştir. Son işlem uygulandıktan sonra çıkış hızı 0,7 Gbit/s olarak Spartan 3E100 FPGA üzerinde elde edilmiştir [37].



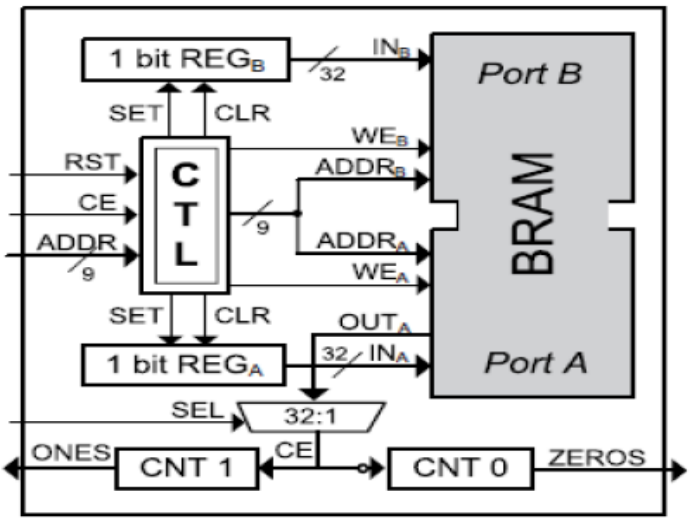
Şekil 3.15. Cret GRSÜ tasarımı [57].

Bu tasarımın avantajı, yüksek çıkış hızına sahip olması ve FPGA'yi etkileyen sıcaklık veya değer aşırı koşullar ile entropi düzeyini artmasıdır. Ancak Xilinx FPGA özel tasarlanmıştır. Aşırı güç tüketimi, elle yerleştirme uygun ara bağlantılı kullanımlar için gerekli olması ve FPGA'lerin büyük kapasiteli seviyelerde kullanılması tasarımın dezavantajlarıdır.

3.3.13. Hafıza Bloklarının Yazma Çarpışmasına Dayalı GRSÜ Tasarımı

FPGA'deki GRSÜ uygulamaları için alternatif bir yöntem olarak Şekil 3.16'da gösterilen çift port hafızaların yazma çarpışmalarına dayalı tasarım [59,60] önerilmiştir. Yazar, GRSÜ ve aygıt tanımlamaları gibi diğer güvenlik sorunları için veya kriptolojide kullanılmak üzere verimli entropi sağlayan blok ram'lerin (BRAM) yazma çarpışmalarını rapor etmiştir. Yazma çarpışması hafıza hücresinin bir alanında meydana geldiğinde hücre tekrar kararlı duruma geçmeden önce yarı kararlı durumda kalması mümkündür. Hücrenin kararlı durumu ilgili sürücüler, komşu bileşenler, üretim ve süreç anormallikleri, materyallerin ısıl titreşimleri ve diğer küçük etkenler tarafından etkilenmektedir. FPGA'deki bu harici ve dâhili etkenler açıkçası hafıza hücresinin bazı bitlerini etkiler. Bu özel hafıza hücreleri 100 Mbit/s 'den daha fazla çıkış hızı ile hızlı ve sağlam GRSÜ

olmasını sağlayan bu tasarım için entropi kaynağı olarak kullanılmıştır. Son işlem uygulanmış ve Diehard testi başarıyla geçilmiştir. Tasarımın değerlendirilmesinde aynı zamanda ister sıfır veya bir, hafızanın tüm giriş hatları kullanılabilir. Ayrıca sonlu durum makineleri her bir hafıza adresinde bütün bitler üzerinde tekrarlayan sorgular gerçekleştirir. Bu tasarım kullanılarak GRSÜ entropi üretimi için uygun olan hafıza bloklarının her bir bitini belirler. Bu belirleme, çarpışma ölçümlerinin 2^{15} denenmesi ile monobit testi tarafından yapılır. Bu değerlendirme sürecinden sonra GRSÜ önerilen tasarımın entropi kaynağı olan yazma çarpışması için hafıza bloğunun seçilmiş bitlerini kullanır [37].



Şekil 3.16. BRAM dayalı tasarım [60].

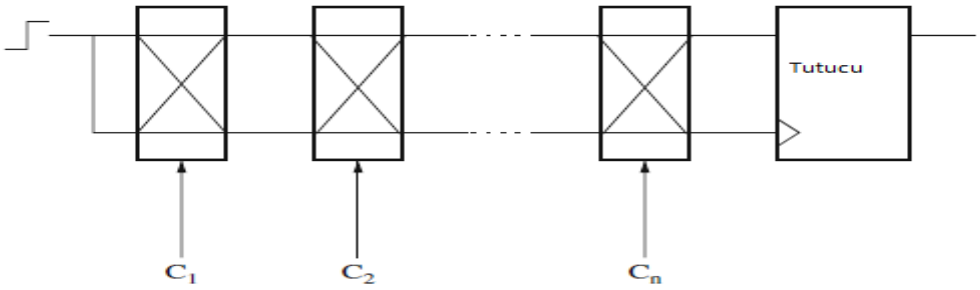
Bu tasarımın avantajı, yüksek çıkış hızı, çıkış hızının aynı FPGA'de çoklu giriş kullanılarak arttırılabilmesi, düşük kaynak kullanılması ve düşük güç tüketimidir. Dezavantajı ise yazma çarpışmasından dolayı kullanılan BRAM'lerin kusurları potansiyel risktir. Entropi kaynağı için matematiksel bir model yoktur.

3.3.14. Fiziksel Klonlanamayan Fonksiyona Dayalı GRSÜ Tasarımı

Bir Fiziksel Klonlanamayan Fonksiyon (Physical Unclonable Function-PUF) kolay değerlendirilebilen ancak çok zor tahmin edilebilen bazı fiziksel aygıttaki bir fonksiyondur. Pek çok açıdan matematiksel tek yönlü fonksiyona eşdeğer donanımdadır. Her soruyu belirli bir cevap ile eşleştirmek gerekir, ancak bu eşleştirme her fiziksel aygıt için farklı ve tahmini mümkün olmamalıdır [61].

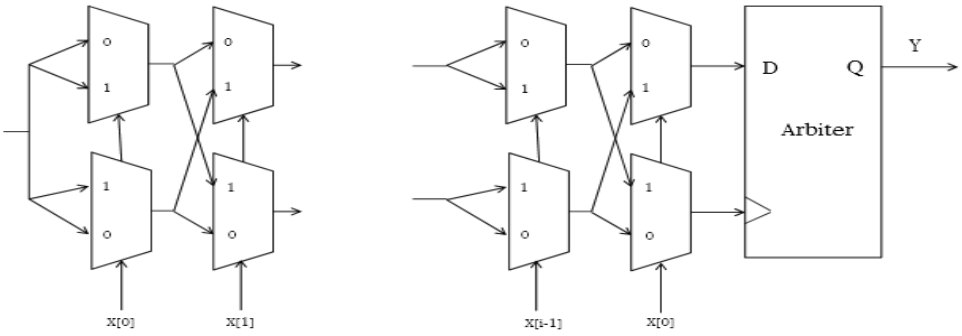
Çip üzerinde bir PUF bazı sebeplerden ötürü oldukça kullanışlıdır. İlk olarak, verilen aygıtı benzersiz tanımlamak için bir yol sağlar. Bu güvenli çip kimlik doğrulaması için kullanıldı. İkinci olarak, bir FPGA'deki PUF gerçek rasgelelilik temelli olduğundan dolayı, rasgele sayı üreteci olarak kullanılabilir. Son olarak en önemlisi, bir PUF çip üzerinde herhangi bir gizli bilgi olmadan tüm fonksiyonelliliğini sağlar.

PUF'a dayalı bir RSÜ tasarımı O'Donnel tarafından önerilmiştir. RSÜ tasarımı Şekil 3.17'de gösterildiği gibi bir PUF devresi etrafında kurulur. Normal şartlar altında PUF devresinin çıkışına verilen soru değeri ile birlikte üretim işlemi süresince yaratılan gecikme yollarının kesin olmaması ile karar verilir. Alternatif olarak, sorunların belirli bir kümesi için gecikme uyumlu olacak ve örnekleme devresi yarı kararlı duruma girecektir. Bundan dolayı, aygıtın çıkışı tahmin edilemeyecektir. Bu iyi bir haber iken, bir soru sıcaklık ve gerilim değişikliklerinden dolayı geçici olarak yarı kararlılığa neden olur. Bu yüzden eğer yeterince kararsız çıkış alınırsa PUF-RSÜ tasarımı sürekli bir soru uygulayarak ve denetleyerek yarı kararlı sorular bulmaya çalışır. Bir başka deyişle, bir pencere içersinde PUF çıkışının düzgün dağılım olması umuduyla, aynı soru sabit pencere uzunluğu ile sabit sayıda PUF devresi ile beslenir. Eğer bu sabit sayıda pencere denedikten sonra başarılamadı ise yeni bir soru SRSÜ yardımıyla üretilir. Yarı kararlı soru bulunduğunda, Von Neumann kullanılarak son işleme tabi tutulan çıkış bit dizisi elde edildi [5].



Şekil 3.17. Gecikme temelli PUF tasarımı [61].

FPGA için ilk PUF tasarımı [62]'de önerilmiştir. Önerilen "arbiter PUF" tasarımı Şekil 3.18'de verilmiştir. İki yol, dizilerde anahtar gecikme elemanlarının sayısı ile ilgili olarak oluşturulmuştur. Her eleman girişler tarafından elde edilen yolu kontrol etmek için ikiye bir çoklayıcı kullanır. X soru bitleri çoklayıcı için seçilmiş bitler olarak kullanılmış ve PUF'un cevabı daha hızlı sinyal yayılımına neden olan yola dayalıdır.



Şekil 3.18. Arbiter PUF tasarımı [62].

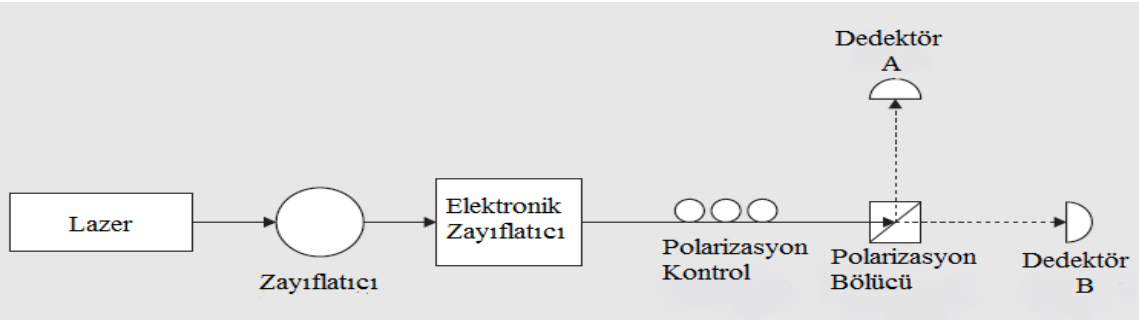
Ancak FPGA üzerinde bu tasarımın kullanımı ile ilgili bazı problemler mevcuttur. Yönlendirme nedeniyle gecikmeler araçların tamamen küçük gecikmelerin süreç değişikliği nedeniyle gölgelenmesi ile otomatik olarak yapılır. PUF'un temeli süreç değişikliğindeki asıl rasgelelilik olduğundan dolayı bu tasarım fonksiyonel değildir.

Diğer tasarımlar:

- Belirli bir frekansta gecikme döngülerinden oluşan halka osilatör PUF'tur [63].
- SRAM (CMOS transistörlerden oluşan yarıiletken bellek türüdür) PUF [64].
- Kararsız çapraz bağlı devrelere dayalı kelebek PUF [65].

3.3.15. Kuantum GRSÜ Tasarımı

Kuantum GRSÜ tasarımı [39], çoğu süreçleri ve denklemleri olasılığa dayalı olan kuantum fizikten rasgelelilik elde eder. Bu tasarım UP optik laboratuarı ve Çek Bilim Akademisinin fizik bölümleri tarafından ortaklaşa olarak geliştirilmiştir. Bu GRSÜ'nün dünyadaki yüksek dereceli kriptografik GRSÜ olduğu iddia edilmektedir.

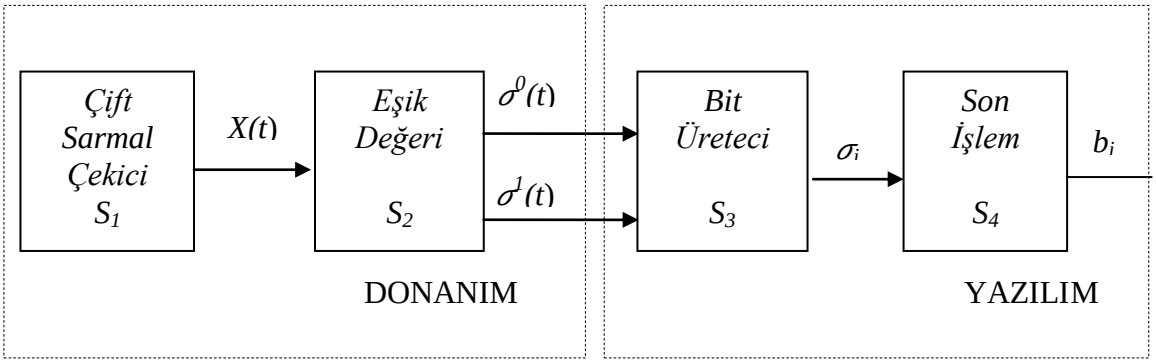


Şekil 3.19. Kuantum mekaniğine dayalı GRSÜ prensibi [39].

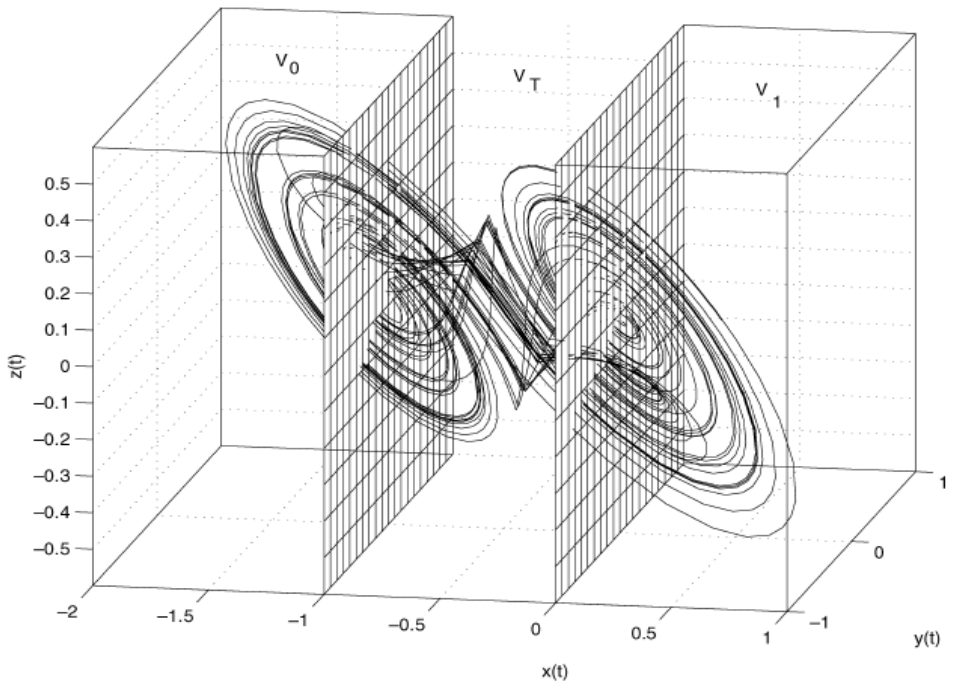
Şekil 3.19'da uygulamanın optik bölümünün şematik diyagramı verilmiştir. Yarı iletken lazer ışık çarpmaları (darbeleri) kaynağı olarak kullanılmıştır. Onun çalışma dalga boyu 830nm'dir. Bir çarpma süresi 100 Hz ile 1 Mhz tekrarlayan frekans ile 400ps-4ns arasından seçmek mümkündür. Her çarpma yaklaşık olarak 108 foton içerir. Çarpmalar ilk olarak makine üzerinde zayıflatılır ve sonra, fotonların ortalama miktarı çarpma için 1.38 fotona sahip olmak için elektronik zayıflatıcı ile belirlenir. Seçilebilir bölme oranı ile ışık bölücü polarizasyon kontrol ve polarizasyon bölücü olarak iki parçadan oluşmaktadır. Polarizasyon kontrol ile isteğe bağlı olarak bölücü oranı ayarlanması mümkündür. Işık bölücünün çıkışı iki çığ etkili foto diyot tarafından yakalanır. Eğer foton A veya B detektörleri arasında hareket edecekse, bunun tamamen rasgele olduğuna dikkat etmek gerekir. Detektörlerden gelen sinyaller elektronik yollar ile işlenir. Burada son işlem için iki yol seçilmiştir: XOR ve Von Neumann doğrulayıcı. Bu üretecin XOR son işlemi sonrası hızı 22.1 Kbit/s olmuştur. Von Neumann ise daha çok bias'a neden olmuştur.

3.3.16. Çift Sarmal Çekicilerden Gerçek Rasgele Bit Üretimi

Şekil 3.20'de tasarımı gösterilen çift sarmal çekiciler kullanılarak GRSÜ önerilmiştir [7]. Uygulamada tasarım kolaylığı ve yüksek frekans işlemleri gibi kaotik osilatör uygulamaları için akım geri beslemeli op-amp kullanılmıştır. Sistem iyi istatistiksel özellikler göstermiş ancak tahmin edilemezlik yeterli olmamıştır. Bunun için daha karmaşık çekiciler kullanılması gerekmektedir. Son işlem uygulanmıştır. FIPS 140-1, NIST ve Diehard testleri başarıyla geçilmiştir.



Şekil 3.20. Çift sarmal çekicilerden gerçek rasgele bit üretimi [7].



Şekil 3.21. Çift sarmallı yapı [7].

Şekil 3.21'de görüldüğü üzere kaotik işaret iki düzlem (s_1 ve s_2) ile üç farklı bölgeye (V_0, V_1 ve V_T) ayrılmıştır. Denklem (3.4)'e göre sistem, V_T bölgesinden V_1 bölgesine geçtiğinde 1 üretmekte, V_T bölgesinden V_0 bölgesine geçtiğinde ise 0 üretmektedir.

$$\sigma_c(x) = \begin{cases} 0, & x < c \\ 1, & x \geq c \end{cases} \tag{3.4}$$

$V_0 = \{(x,y,z)\ x \leq c_2\}, V_T = \{(x,y,z)\ c_2 \leq x \leq c_1\}$ ve $V_1 = \{(x,y,z)\ x \geq c_1\}$. Şekil 3.21. c_1=0 ve c_2= -1 için alt uzaylarını gösterir. S_2, iki eşik fonksiyonunu ifade eder ve Denklem 3.5'te gösterilmiştir:

$$S_2: \quad \sigma^1(x(t)) = \begin{cases} 0, & x(t) < c_1 \\ 1, & x(t) \geq c_1 \end{cases} \qquad \sigma^0(x(t)) = \begin{cases} 0, & x(t) > c_2 \\ 1, & x(t) \leq c_2 \end{cases} \tag{3.5}$$

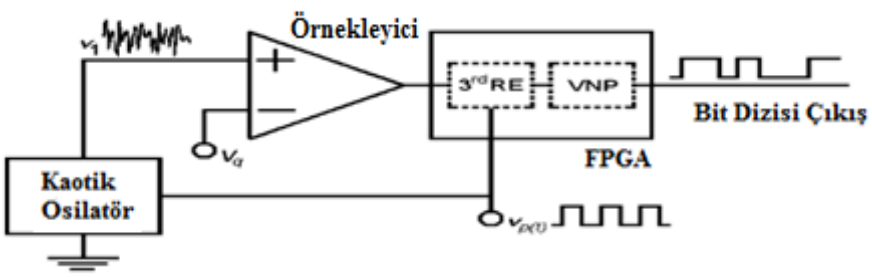
$$\sigma_i(\sigma^0, \sigma^1) = \begin{cases} 0, \sigma^0 = 0, \sigma^1 : 0 \uparrow 1 \\ 1, \sigma^1 = 0, \sigma^0 : 0 \uparrow 1 \end{cases}$$

Sistemde iki durum bulunmaktadır (σ_1 ve σ_0) ve bunlar yukarıdaki denklemlerde gösterilmiştir. σ_1 durumunda sistem V_T bölgesinden V_1 bölgesine veya tersi durumda 1 veya 0 üretmektedir. σ_0 durumunda sistem V_T bölgesinden V_0 bölgesine veya tersi durumda 1 veya 0 üretmektedir. Bu iki durum sonucunda σ_i bit dizisi elde edilmektedir, ancak burada bölgelerin ayrılmasında seçilen sınırlar rasgele sayı üretimini doğrudan etkilemektedir buda sistemde zayıflıklar ortaya çıkarmaktadır. Üretilen bit dizisindeki 1-0 dağılımının kötü olmasıdır ve bu sebeple uygulanan Von Neumann son işleminin sonucunda çıkış oranının dörtte bir oranında düşmesi sistemin eksik yönleridir.

3.3.17. Otonom Olmayan Kaotik Osilatörlere Dayalı GRSÜ

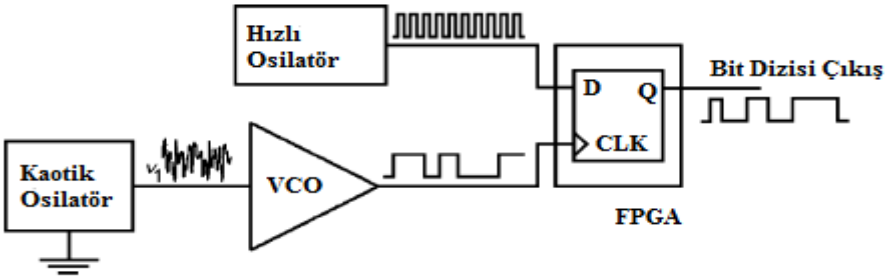
Entegre devrelerde yüksek performans için kullanımı uygun olan otonom olmayan kaotik osilatör kullanılarak GRSÜ önerilmiştir. Ayrıca daha iyi istatistiksel özellikler ve iyi çıkış hızı elde etmek için çift osilatör mimarisi kullanılmıştır. Şekil 3.22'de gösterilen tasarımda ise sadece sürekli zaman kaotik osilatörler kullanıldığında 488 Bit/s, Şekil 3.23'te tasarımı gösterilen sürekli zamanlı kaotik osilatörler ile çift osilatör mimarisi bir arada kullanıldığında ise çıkış hızı 830 Kbit/s olmuştur. Bu çıkış hızları herhangi bir son işlem uygulaması uygulanmadan elde edilmiştir [66].

Literatürde var olan bir çok kaotik osilatör olmasına rağmen, osilatörlerin bazıları düşük güç tüketimi, yüksek frekanslı işlem ve alçak gerilim seviyelerinde çalışma imkanı gibi yüksek performanslı entegre devre gerçekleştirimi için tasarlanmıştır. Bu sebepten dolayı bu çalışmada yüksek performanslı entegre devre gerçekleştirimi için uygun olan otonom olmayan kaotik osilatörlerin kullanımı tercih edilmiştir. Önerilen kaotik sistemin stroboscopic poincare haritalarında rasgele veriler elde edilmiş ve bu veriler FIPS 140-2 testine tabi tutularak doğrulanmıştır.



Şekil 3.22. Sadece kaotik osilatör kullanılan RSÜ [66].

Ancak sadece kaotik osilatör kullanılan yapı NIST testlerinden dört âdetini geçememiştir ve hız 488 Bit/s olmuştur. Bu yüzden daha iyi çıkış hızı ve istatistiksel özelliklerinin iyi olması amacıyla çift osilatör mimarisi önerilmiştir. Bu uygulama sonucunda çıkış hızı 830 Kbit/s olmuş ve son işlem uygulanmadan NIST testlerinin başarıyla geçmiştir.

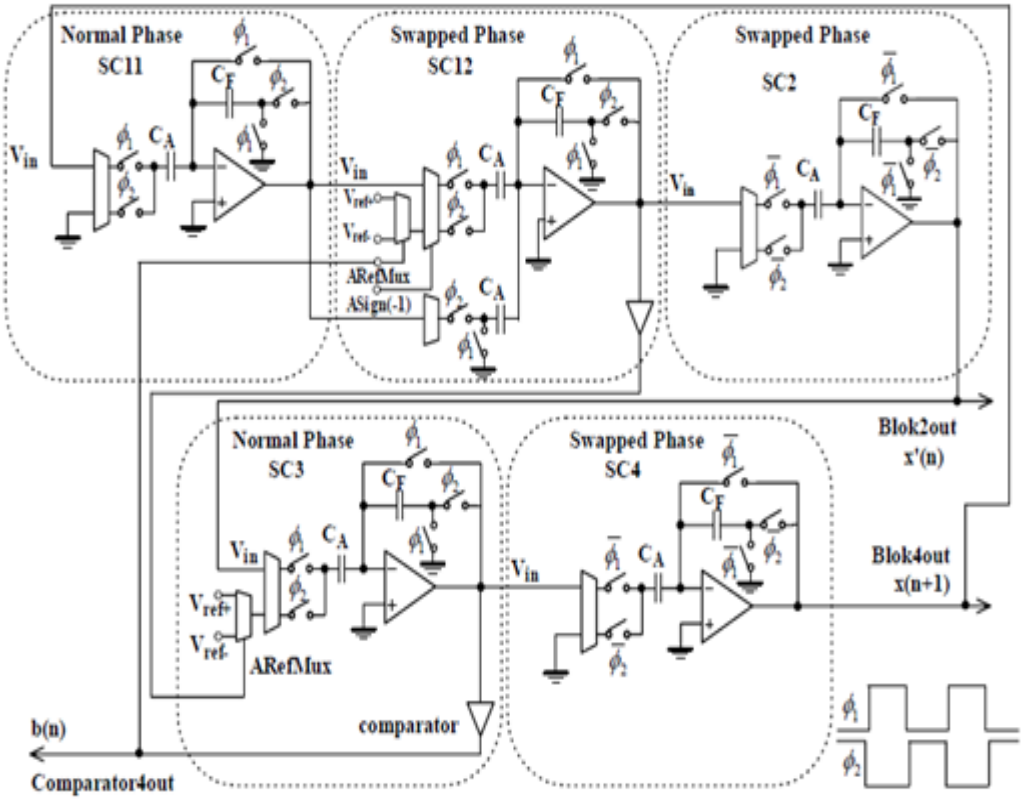


Şekil 3.23. Çift osilatör mimarisi kullanılan RSÜ [66].

3.3.18. Anahtarlamalı Kapasitör Donanımda Gömülü Kaos Tabanlı Güçlü GRSÜ

Şekil 3.24'te tasarımı gösterilen gerilim besleme duyarlılığının azaltılması ile yeni bir kaos tabanlı gerçek rasgele sayı üreteci önerilmiştir. Geleneksel olarak kullanılan rasgele kaynakların aksine kaos özellik gösteren iyi tanımlanmış deterministik anahtarlamalı kapasitör devre kullanılmıştır. Bu tasarımda herhangi bir harici bileşen olmadan yeniden yapılandırılabilir PSOC cihaz içine yerleştirilmiştir [67].

Önerilen tasarım üretilen rasgele bit dizisinin kalitesine besleme geriliminin etkisini azaltmak için optimize edilmiştir. Bu etki önerilen XOR doğrulayıcı ve devre topolojisinin optimizesi ile önemli oranda azaltılır. 60 Kbit/s veri hızı elde edilmiştir. XOR son işlemine tabi tutulmuş ve FIPS testini başarıyla geçmiştir. Bu tasarımda FPAA'lere göre daha düşük hızlar vermesine rağmen PSOC donanımı ve piece wise affine (Markov) kaotik haritaları kullanılmıştır. Bu kaotik haritaların avantajı analog uygulama olması ile sistemin sağlamlığı artacaktır.



Şekil 3.24. 5-SC blok GRSÜ PSOC uygulaması [67].

3.3.19. Sayısal Diferansiyel Kaosa Dayalı Rasgele Sayı Üreteci

Tamamen sayısal diferansiyel kaosa dayalı rasgele sayı üreteci [68]'de önerilmiştir. Sayısal devrenin çıkışı maksimum Lyapunov üslerinin çıkış zaman serilerini hesaplayarak kaotik olduğunu kanıtlanmıştır. Sistemin çözümünde dördüncü dereceden Range Kutta, orta nokta ve euler tekniği olmak üzere üç faklı metot kullanılmıştır. Sistem, VHDL'de yazılmıştır ve Xilinx Virtex 4 FPGA üzerinde gerçekleştirilmiştir. Devre oldukça küçük bir alan kaplamış ve çıkış hızı 2.1 Gbit/s olarak elde edilmiştir. Yeni önerdikleri son işlemden geçirildikten sonra NIST testine tabi tutulmuştur ve başarıyla geçmiştir. Yeni önerdikleri son işlem adımları şu şekildedir:

1 adım: Göz ardı edilen en yüksek değerlikli bit bulundu ve sadece düşük olanları korunarak kısa sürede tahmin edilme ortadan kaldırılır.

2.Adım: İstatistiksel gözlemlere dayanarak tek ve çift sayıların dağılımı bulundu ve böylece birler ve sıfırlar dengelenmemektedir. Çıkış dağılımı düşük değerlikli bitlerin göz ardı edilmesi ile dengelenmiştir.

Bu metodun sorunu; tarafsız (yansız) bir sonuç veren, RAM içerisinde hafıza yeri bulmaktır. Onların önerisi; düzenli olarak, uygun bir yer arayan devre kullanmaktır. Bu tasarımın dezavantajı; Xilinx cihazlarından FPGA'lere taşınabilir olmamasıdır. GRSÜ'nün istatistiksel oranlarını iyileştirmek için son işlem gerekmektedir.

Tablo 3.5'te literatürde bulunan GRSÜ tasarımlarının karşılaştırmaları verilmiştir.

Tablo 3.5. GRSÜ tasarımlarının karşılaştırılması

GRSÜ Tasarımları	**Entropi Kaynağı**	**Uygulanan Son işlem**	**Uygulanan İstatistiki Test**	**Çıkış Hızı**
Bagini ve Bucci	Direnç	-	-	-
Intel GRSÜ	Termal ısıl gürültü	Von Neumann	FIPS-140	-
Fischer Drutovsky	PLL	Resilient	NIST	70 Kbit/s
Tkacik GRSÜ	RO, LFSR, CASR	-	Diehard	-
Epstein GRSÜ	İki çoklayıcı, iki durumlu bellek elemanları, tümleyen	Von Neumann	Diehard	80 Mbit/s
Kohlbranner GRSÜ	RO	XOR	NIST	< 1 Mbit/s
Figaro GRSÜ	Galois RO, Fibonacci RO	LFSR	Diehard	12.5 Mbit/s
Sunar GRSÜ	RO	Resilient	NIST, Diehard	2.5 Mbit/s
Vasyltsov GRSÜ	Yarı kararlı halka	-	AIS 31,FIPS 140-1/2	
Danger GRSÜ	FPGA açık döngü yarı kararlılığı	Von Neumann	NIST	20 Mbit/s
Geçiş Etkili Halka Osilatör Dayalı GRSÜ	Geçiş Etkili RO	-	NIST	250 Kbit/s
Parazitlenme Etkisine Dayalı GRSÜ	FPGA'de pazitlenme etkisi	-	NIST	0.7 Gbit/s
Hafıza Bloklarının Yazma Çarpışmasına Dayalı GRSÜ	Çift port hafızalarının (BRAM) yazma çarpışmaları	Von Neumann, H Fonksiyon	AIS, Diehard	100 Mbit/s
PUF GRSÜ	PUF	VonNeumann	-	-
Kuantum GRSÜ	Kuantum Fizik	XOR, Von Neumann	-	22.1 Kbit/s
Çift Sarmal Çekicilere Dayalı GRSÜ	Kaotik Osilatör	Von Neumann	NIST, FIPS-I, Diehard	-
Otonom Olmayan Kaotik Osilatöre Dayalı GRSÜ	Kaotik Osilatör	-	NIST	830 Kbit/s
Anahtarlamalı Kapasitör Donanıma Gömülü Kaos Tabanlı GRSÜ	Deterministik anahtarlamalı kapasitör devre	XOR	FIPS-140	60 Kbit/s
Sayısal Differansiyel Kaosa Dayalı GRSÜ	Sayısal differansiyel kaos	Son işlem önerilmiş	NIST	2.1 Gbit/s

4. RASGELE SAYI ÜRETEÇLERİNİN TEST EDİLMESİ

Rasgele ve sözde rasgele sayılara olan ihtiyaç birçok kriptografik uygulamada ortaya çıkmaktadır. Örneğin, genel olarak kriptografide kullanılan anahtarların rasgele sayılar ile üretilmesi gerekmektedir. Çoğu kriptografik protokoller çeşitli noktalarda rasgele ve sözde rasgele sayı girişleri gerektirir.

Bu kısımda kriptografi, modelleme ve simülasyon uygulamalarını içeren amaçlar için kullanılabilen rasgele ve sözde rasgele sayı üreteçlerinin rasgelelilik testleri incelenecektir. Ancak bu kısmın odak noktası, rasgeleliğin kriptografik amaçlar için gerektiği uygulamalardır. Ulusal Standartlar ve Teknoloji Enstitüsü (NIST), bu prosedürlerin, ikili bir dizinin rasgeleliğinde bir sapmanın olup olmadığının tespit edilmesinde yararlı olduğuna inanmaktadır. Bununla beraber, testi yapan, ikili dizide görülecek büyük sapmaların, hem kötü tasarlanmış üreteçlerden hem de anormalliklerden kaynaklanabileceğine dikkat etmelidir (Örneğin, belli bir sayıdaki başarısızlık belirli bir üreteç tarafından üretilen rasgele dizilerde beklenebilir). Bu bölüm için referans [18] kabul edilmiştir.

4.1. Rasgelelilik

Bir rasgele bit dizisi bir madeni paranın yüzeylerinin 0 ve 1 ile etiketlenerek havaya atılmasının sonucu olarak açıklanabilir. Her atılma olayında 0 ve 1'in gelme olasılığı tam olarak 1/2'dir. Ayrıca atılma olayının her biri birbirinden bağımsızdır: Sonuçta bir önceki para atılma olayı gelecekteki para atılma olayını etkilemeyecektir. 0 ve 1 değerleri rasgele dağıtılmış ([0,1] tekdüze dağıtılmış) olacağından dolayı, hilesiz madeni para mükemmel bit dizisi üretecedir. Dizideki tüm elemanların her biri diğerlerinden bağımsız üretilmiştir ve dizinin bir sonraki elemanın değeri tahmin edilemez. Ancak kriptografik amaçlar için hilesiz madeni paralar kullanılması uygun değildir.

4.2. Tahmin Edilemezlik

Kriptografik uygulamalar için üretilen rasgele ve sözde rasgele sayılar tahmin edilemez olmalıdır. SRSÜ'leri için, eğer tohum bilinmiyor ise, dizinin bir sonraki çıktısı, dizinin

önceki rasgele sayılarının herhangi biri biliniyor olsa bile tahmin edilememelidir. Ayrıca tohum değerini üretilen değerlerin bilgisinden elde etmek mümkün olmamalıdır. Tohum değeri ile üretilen değerler arasında herhangi bir ilişki olmamalıdır, dizideki her eleman olasılığı 1/2 olan bağımsız rasgele olayların sonucu olarak ortaya çıkmalıdır. Sonraki değerlerin tahmin edilemezliğini sağlamak için tohum elde etmeye önem gösterilmelidir. SRSÜ'lerde üretim algoritması ve tohum değeri bilinir ise, elde edilen bütün değerler elde edilebilir. Üretim algoritması açık olabilir ancak tohum mutlaka gizli olmak zorundadır, tohum değerinden üretilen değerler tekrar etmemelidir ve tohum değeri de tahmin edilemez olmalıdır.

4.3. Test Etme

Çeşitli istatistiksel testler, gerçek rasgele dizilerin rasgeleliğinin değerlendirmesi ve karşılaştırılmasının yapılması amacıyla diziye uygulanabilir. Rasgelelilik olasılıksal özelliktir öyle ki, rasgele dizinin özellikleri olasılıksal olarak tanımlanabilir. İstatistiksel testlerin olası sonuçları, gerçek bir diziye uygulandığında, testten önce bilinmektedir ve olasılık terimleri ile ifade edilebilir. Sonsuz sayıda istatistiksel test vardır. Bu testler dizinin rasgele olmadığını anlamamızı sağlayacak örnekleri tespit etmemizi sağlarlar. Bir dizinin rasgele olup olmadığına karar vermek için o kadar çok test olması sebebiyle, testlerin belirli sonlu kümeleri tamamlandı diyemeyiz. Ayrıca üreteçler hakkında yanlış kararlar vermekten kaçınmak ve dikkatli karar vermek amacıyla istatistiksel testler iyi yorumlanmalıdır.

İstatistiksel bir test, belirli bir geçersiz hipotezi (H_0) test etmek için formüle edilmiştir. Burada amaç, test altındaki geçersiz hipotez test edilen dizinin rasgele olduğudur. Geçersiz hipotez ile ilgili olarak alternatif hipotez (H_a) dizinin rasgele olmadığıdır. Her uygulanan test için, geçersiz hipotezin kabul veya ret edilmesi yönünde bir karar veya sonuç ortaya çıkar.

Her bir test için uygun bir rasgelelilik istatistiği seçilmiş olmalı ve geçersiz hipotezin kabul edilmesi veya ret edilmesin belirlenmesinde kullanılmalıdır. Rasgelelilik varsayımı altında, böyle bir istatistiğin olası değerlerinin dağılımı vardır. Geçersiz hipotez altında bu istatistiğin teorik referans dağılımı matematiksel yöntemler ile belirlenebilir. Bu referans dağılımından kritik bir değer belirlenir. Test süresince, test istatistiksel değeri verileri hesaplanır (dizi test edilmekte). Test istatistiksel değeri kritik değer ile karşılaştırılır. Eğer

test istatistiksel değeri kritik değeri geçiyor ise, rasgelelilik için geçersiz hipotez ret edilir. Aksi takdirde, geçersiz hipotezi ret edilemez (örneğin, hipotez kabul edilir).

Uygulamada, istatistiksel hipotez testinin çalışma sebebi referans dağılımının ve kritik değerin rasgeleliğe bağımlı olması ve rasgeleliğin kesin olmayan varsayımı altında üretilmiş olmasıdır. Aslında rasgelelilik varsayımı eldeki veriler için doğruysa, veride hesaplanan test istatistik değerinin sonucu kritik değeri geçmesi çok düşük olasılıkta olacaktır (%0.01).

Diğer taraftan, hesaplanan test istatistik değeri kritik değeri geçmesi, düşük olasılık olayının meydana gelmediğini gösterir. Bu nedenle, hesaplanmış test istatistik değeri kritik değeri geçtiğinde, rasgelelilik varsayımının şüpheli veya hatalı olduğu sonucu elde edilir. Bu durumda istatistiksel hipotez testi aşağıdaki sonuçlar verir: H_0 (rasgele) ret edilir ve H_a (rasgele değil) kabul edilir.

Tablo 4.1'de istatistiksel hipotez testinin genel sonuçları verilmiştir.

Tablo 4.1. Hipotez testi genel sonuçları

Gerçek Durum	**Sonuç**	
	H_0 Kabul edilir	H_a Kabul edilir
Veri rasgeledir(H_0 doğrudur)	Hata yok	Tip I hata
Veri rasgele değildir(H_a doğrudur)	Tip II hata	Hata yok

Eğer veri gerçekten rasgele ise, geçersiz hipotezin ret sonucu çok kısa sürede meydana gelecektir (verinin rasgele olmadığına karar verilir). Bu sonuç tip I hatası olarak adlandırılır. Eğer veri gerçekten rasgele değil ise, geçersiz hipotezin (verinin rasgele olduğuna karar verilir) kabul sonucu tip II hatası olarak adlandırılır. Veri gerçekten rasgele iken H_0 kabul eden ve veri gerçekten rasgele değil iken H_0'ı ret eden sonuçlar doğrudur.

Tip I hatasının olasılığı testin yanılgı düzeyi (doğru sonucun güven aralığı dışında ortaya çıkma olasılığı) olarak adlandırılır. Bu olasılık testte öncelikle ayarlanır ve α olarak gösterilir. Test için α dizi gerçekte rasgele iken dizinin rasgele olmama olasılığını gösterecektir. Bu, dizi iyi bir üreteçle üretildiğinde bile dizinin rasgele olmayan özelliklerinin tespit edilebileceğini gösterir. Kriptografide α'nın genel değeri yaklaşık olarak 0.01'dir.

Tip II hatasının olasılığı β olarak gösterilir. Test için β dizi gerçekte rasgele değil iken dizinin rasgele olma olasılığını gösterecektir. Bu, dizi kötü bir üreteçle üretildiğinde bile dizinin rasgele olma özelliklerinin tespit edilebileceğini gösterir. α'dan farklı olarak β sabit bir sayı değildir. Bir veri dizisinin rasgele olmaması için sonsuz yol olduğundan dolayı β

birçok farklı değerler alabilir ve her farklı yol farklı β üretir. Birçok rasgele olamama durumu olduğundan dolayı tip II hatası β'nın hesaplanması tip hatası α'nın hesaplanmasından zordur.

Aşağıdaki testlerin en önemli amacı, tip II hatasının olasılığını en aza indirmektedir, örneğin kötü bir üreteç ile üretilen dizinin iyi bir üreteçle üretilmiş gibi kabul edilme olasılığını en aza indirmektir. α ve β olasılıkları birbirleriyle ve test edilen dizinin boyutuyla ilişkilidir. Burada, iki değer tanımlanmış ise üçüncü değer otomatik olarak belirlenebilir. Uygulayıcılar genellikle örnek dizi boyutu n ve tip I hata olasılığı α'yı belirlerler. Sonra en küçük β değerini üretecek olan kritik nokta seçilir. Bu, gerçekten rasgele bir dizi üreten kötü bir üretecin kabul edilme olasılığını belirleyen uygun bir örnek dizi boyutu seçmeyi ifade etmektedir. Sonra kabul edilebilirlik için kestirim noktası, bir dizinin yanlışlıkla rasgele olarak kabul edilme olasılığının en küçük olası değerinin verecek şekilde seçilmelidir.

Her test verinin bir fonksiyonu olan hesaplanmış bir test istatistik değerine bağlıdır. Eğer test istatistik değeri S ve kritik değer t ise, tip I hata olasılığı $P(S>t)||H_0$ doğru)=$P(H_0$ ret | H_0 doğru), ve $P(S\leq t)||H_0$ yanlış)=$P(H_0$ kabul | H_0 yanlış)'dır. Test istatistiği geçersiz hipoteze karşı kanıt kuvvetinin özeti olan P değerini bulmada kullanılır. Bu testler için, her p değeri test edilen diziden daha düşük rasgelelilikte dizi üretebilen mükemmel rasgele sayı üretecinin olasılığıdır ve test tarafından değerlendirilirken rasgele olmamanın çeşidini verir. Eğer bir test için p değeri 1'e eşit elde edilirse, dizi mükemmel rasgeleliğe sahip olduğu görünür. 0 olduğu takdirde tamamen rasgele olmadığı görünür. Bir önem seviyesi (α) testler için seçilmiş olabilir. Eğer p_değeri ≥α ise geçersiz hipotez kabul edilir; örneğin dizi rasgeledir. Eğer p_değeri <α ise geçersiz hipotez red edilir; örneğin dizi rasgele değildir. α değeri genellikle [0.001, 0.01] aralığında seçilir.

- α değerinin 0.001 olması, eğer bir dizi rasgele ise test tarafından her 1000 diziden 1 tanesinin reddedileceğinin beklendiğini gösterir. p_değeri ≥ 0.001 için, dizi %99.9 doğrulukta rasgele olması beklenir. p_değeri< 0.001 ise %99.9 doğrulukta rasgele olmaması beklenir.
- α değerinin 0.01 olması, eğer bir dizi rasgele ise test tarafından her 100 diziden 1 tanesinin reddedileceğinin beklendiğini gösterir. p_değeri ≥ 0.01 için, dizi %99 doğrulukta rasgele olması beklenir. p_değeri< 0.01 ise %99 doğrulukta rasgele olmaması beklenir.

Buradaki örneklerde α değeri 0.01 olarak seçilmiştir.

Rasgelelilik, tahmin edilemezlik ve test etme için dikkat edilecekler:

Aşağıdaki varsayımlar test edilen ikili diziler ile ilgili olarak yapılmıştır.

- **Tekdüzelik:** Rasgele veya sözde rasgele bit dizilerinin üretimindeki her bir noktada 0 ve 1'lerin oluşumu neredeyse eşit şekildedir. Örneğin, her birinin olasılığı 1/2'dir. 0 ve 1'lerin beklenen sayısı n/2'dir, burada n dizi uzunluğudur.
- **Ölçeklenebilirlik:** Bir dizi için kabl edilebilen bir test rasgele çıkarılmış alt dizilere de uygulanabilir. Eğer dizi rasgele ise, çıkarılmış alt dizilerde rasgeledir. Bundan dolayı, çıkarılan alt dizlerde rasgelelilik testini geçmelidir.
- **Tutarlılık:** Bir üretecin davranışı başlangıç değerlerine (tohum) karşı dayanıklı olmalıdır. Tek bir tohumdan elde edilen çıktıya dayana SRSÜ veya tek bir fiziksel çıktıdan üretilen çıktıya dayanan RSÜ'yü test etmek için yetersizdir.

4.4. RSÜ'lerine Uygulanan NIST İstatistiksel Testleri

Rasgele sayı üreteçleri tarafından üretilen uzun ikili bit dizilerinin rasgeleliğini ölçmek için geliştirilmiştir. 16 test içeren istatistiksel bir pakettir. Bu testler dizi içerisindeki rasgele olmayan durumlara odaklanır. Bu testler aşağıda verilmiştir:

1. Frekans (monobit) testi
2. Bir blok içinde frekans testi
3. Akış (Runs) testi
4. Bir blok içinde en-uzun-birlerin akış (longest-run-of-ones) testi
5. İkili matris rankı testi
6. Ayrık Fourier dönüşümü (spektral) testi
7. Örtüşmeyen şablon eşleştirme (Non-overlapping template matching) testi
8. Örtüşen şablon eşleştirme (Overlapping template matching) testi
9. Maurer'in "Evrensel İstatistik" testi
10. Lempel-Ziv sıkıştırma testi
11. Doğrusal karmaşıklık (linear complexity) testi
12. Seri (serial) test
13. Yaklaşık entropi (approximate entropy) testi
14. Kümülatif toplamlar (cumulative sums – cusums) testi

15. Rasgele gezinimler (random excursions) testi
16. Rasgele gezinimler değişken (random excursions variant) testi

Bu 16 testin uygulanma sırası isteğe bağlı olan bir durumdur. Ancak ilk olarak frekans testinin uygulanması tavsiye edilmektedir. Eğer frekans testi başarısız sonuçlanırsa diğer testlerinde başarısız olma olasılıkları yüksektir.

Test paketindeki bazı testler, referans dağılımı olarak standart normal ve ki-kare (X^2) dağılımlarını kullanır. Bu dağılımların açıklaması aşağıda yapılmıştır:

a. **Normal Dağılım:** İstatistiksel yorumlamanın temelini oluşturan normal dağılım, birçok rassal süreçlerin dağılımı olarak karşımıza çıkar. Sürekli bir x rasgele değişkeninin olasılık yoğunluk fonksiyonu:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad , -\infty < x < \infty, -\infty < \mu < \infty, \sigma^2 > 0 \tag{4.1}$$

biçiminde olduğunda x rasgele değişkenine normal dağılıma sahip denir ve $x \sim N(\mu, \sigma^2)$ biçiminde gösterilir.

Burada $\mu \in R$ ve $\sigma^2 \in (0, \infty)$ dağılımın parametreleridir.

μ=kitle ortalaması

σ^2=kitle varyansı

e=2.71828

π=3.14159...

Normal dağılımın özellikleri:

- Normal dağılım olasılık yoğunluk fonksiyonunun yani f(x) altında kalan alan 1'dir.

$$\int_{-\infty}^{\infty} f(x)dx = 1 \tag{4.2}$$

- Simetrik bir dağılım gösterir.

$$\int_{-\infty}^{\mu} f(x)dx = \int_{\mu}^{\infty} f(x)dx = 1/2 \tag{4.3}$$

- Normal dağılım için dağılım fonksiyonu

$$f(x) = \Phi(x) = P(X \leq x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{4.4}$$

Bu fonksiyonun belli bir x değerinden daha küçük olma olasılıklarını verir.

$\sigma^2 = 1$ ve μ=0 olur ise normal dağılıma standart normal dağılım denir. Standart normal dağılıma sahip rasgele değişken genellikle z harfi ile gösterilir. z~N(0,1) rasgele değişkeninin olasılık yoğunluk fonksiyonu

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} \quad -\infty < z < \infty \tag{4.5}$$

x~N(μ,σ^2) normal dağılımına sahip bir rasgele değişken olmak üzere $z = \frac{x-\mu}{\sigma}$ rasgele değişkeni standart normal dağılıma sahip bir rasgele değişken olur. Bu durumda:

$$P(X \leq x) = \phi\left(\frac{x-\mu}{\sigma}\right) = \phi(z) = P(Z \leq z) \tag{4.6}$$

Buna standartlaşma işlemi denilir. Simetri özelliğinden: $\phi(z) = 1 - \phi(z) = 1 - P(Z \leq z)$ olur.

Örnek: P(=0.42<z<2.53)=P(z<2.53)-P(z<0.42) =0.4943-0.1628=0.3315

2.53 değeri Tablo 4.2'den okunurken 2.5 satırdan 0.03 sütun kısmından alınarak ikisinin birleşimindeki değer alınır.

Örnek: P(Z>1)=1-P(Z<1)=1-0.8413=0.1587

Örnek : P(-1,56<Z<2.53)= P(z<2.53)-P(z<-1.56)=

P(z<2.53)-(1- P(z<1.56))=0.4943-(1-0.4406)=0.4349

Tablo 4.2. Z Standart dağılım tablosu

z	**.00**	**.01**	**.02**	**.03**	**.04**	**.05**	**.06**	**.07**	**.08**	**.09**
.0	.0000	.0040	.0080	.0120	.0160	.0199	.0239	.0279	.0319	.0359
.1	.0398	.0438	.0578	.0517	.0557	.0596	.0636	.0675	.0714	.0753
.2	.0793	.0832	.0871	.0910	.0948	.0987	.1026	.1064	.1103	.1141
.3	.1179	.1217	.1255	.1293	.1331	.1368	.1406	.1443	.1480	.1517
.4	.1554	.1591	.1628	.1664	.1700	.1736	.1772	.1808	.1844	.1879
.5	.1915	.1950	.1985	.2019	.2054	.2088	.2123	.2157	.2190	.2224
.6	.2257	.2291	.2224	.2357	.2389	.2422	.2454	.2486	.2517	.2549
.7	.2580	.2611	.2642	.2673	.2704	.2734	.2764	.2794	.2823	.2852
.8	.2881	.2910	.2939	.2967	.2995	.3023	.3051	.3078	.3106	.3133
.9	.3159	.3186	.3212	.3238	.3264	.3289	.3315	.3340	.3365	.3389
1.0	.3413	.3438	.3461	.3485	.3508	.3531	.3554	.3577	.3599	.3621
1.1	.3643	.3665	.3686	.3708	.3729	.3749	.3770	.3790	.3810	.3830
1.2	.3849	.3869	.3888	.3907	.3925	.3944	.3962	.3980	.3997	.4015
1.3	.4032	.4049	.4066	.4082	.4099	.4115	.4131	.4147	.4162	.4177
1.4	.4192	.4207	.4222	.4236	.4251	.4265	.4279	.4292	.4306	.4319
1.5	.4332	.4345	.4357	.4370	.4382	.4394	.4406	.4418	.4429	.4441
1.6	.4452	.4463	.4474	.4484	.4495	.4505	.4515	.4525	.4535	.4545
1.7	.4554	.4564	.4573	.4582	.4591	.4599	.4608	.4616	.4625	.4633
1.8	.4641	.4649	.4656	.4664	.4671	.4678	.4686	.4693	.4699	.4706
1.9	.4713	.4719	.4726	.4732	.4738	.4744	.4750	.4756	.4761	.4767
2.0	.4772	.4778	.4783	.4788	.4793	.4798	.4803	.4808	.4812	.4817
2.1	.4821	.4826	.4830	.4834	.4838	.4842	.4846	.4850	.4854	.4857
2.2	.4861	.4864	.4868	.4871	.4875	.4878	.4881	.4884	.4887	.4890
2.3	.4893	.4896	.4898	.4901	.4904	.4906	.4909	.4911	.4913	.4916
2.4	.4918	.4920	.4922	.4925	.4927	.4929	.4931	.4932	.4934	.4936
2.5	.4938	.4940	.4941	.4943	.4945	.4946	.4948	.4949	.4951	.4952
2.6	.4953	.4955	.4956	.4957	.4959	.4960	.4961	.4962	.4963	.4964
2.7	.4965	.4966	.4967	.4968	.4969	.4970	.4971	.4972	.4973	.4974
2.8	.4974	.4975	.4976	.4977	.4977	.4978	.4979	.4979	.4980	.4981
2.9	.4981	.4982	.4982	.4983	.4984	.4984	.4985	.4985	.4986	.4986
3.0	.4987	.4987	.4987	.4988	.4988	.4989	.4989	.4989	.4990	.4990
3.1	.4990	.4991	.4991	.4991	.4992	.4992	.4992	.4992	.4993	.4993
3.2	.4993	.4993	.4994	.4994	.4994	.4994	.4994	.4995	.4995	.4995
3.3	.4995	.4995	.4995	.4996	.4996	.4996	.4996	.4996	.4996	.4997
3.4	.4997	.4997	.4997	.4997	.4997	.4997	.4997	.4997	.4997	.4998

b. Ki-Kare Testi (X^2 Testi)

Bilinen en iyi testtir. Ampirik dağılım fonksiyonu ve teorik olarak beklenen dağılım arasındaki karşılaştırılması temellidir.

Ampirik dağılım rasgele işlem sonuçlarına dayanmaktadır. Rasgele değerleri ölçülen n $i_1,i_2,.....,i_k$ k sınıfa bölünmelidir.Sınıflar $N_1+N_2+.....+N_k=N$ değerlerini içerir. Her bir sınıf için, değerlerin beklenilen sayıları verilen $P_i(p_i=p(i))$ için beklenilen dağılım fonksiyonu $N_i^* = Np_i$ile hesaplamak zorundadır.

Ölçülen değerler ile beklenilen değerler arasındaki farklılığın kareleri göz önüne alınırsa X^2 değerini verir.

$$X^2 = \sum_{i=1}^{k} \frac{(N_i - NP_i)^2}{NP_i} = \frac{1}{N} \sum_{i=1}^{k} (\frac{N_i}{P_i})^2 \text{-n} \qquad (4.7)$$

K sınıfları için v=k-1, X^2 dağılımının serbestlik derecesidir.

Örnek: zar atma

Değer:	1	2	3	4	5	6
Gözlenen:	15	19	22	21	17	26

Değerlerin sayısı n=120

Her bir atılan zar için beklenen olasılık $P_i = \frac{1}{6}$

Değerlerin beklenen sayısı $N_i^* = Np_i = 120 * \frac{1}{6} = 20$

$$X^2 = \sum_{i=1}^{6} \frac{(N_i\text{-}NP_i)^2}{NP_i} = \frac{(15\text{-}20)^2}{20} + \frac{(19\text{-}20)^2}{20} + \frac{(22\text{-}20)^2}{20} + \frac{(21\text{-}20)^2}{20} + \frac{(17\text{-}20)^2}{20} + \frac{(26\text{-}20)^2}{20} = 3.8$$

4.4.1. Frekans Testi

Bu testin amacı, gerçek bir RSÜ'de olması beklenen bir özellik olan bir dizideki "0" ve "1"sayılarının yaklaşık olarak aynı olup olmadığının ölçülmesidir. Testte kullanılan referans dağılım yarı normal dağılımdır. Parametre yoktur. Dizi uzunluğu en az 100 bit olmalıdır. P<0.01 olursa dizi rasgele değildir.

Denklem (4.8)'de gösterilen ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,.....,\varepsilon_n$), n bit dizisinin uzunluğunu, S_{obs}, X_i'lerin mutlak değer toplamı ve S_n rasgele değişkenlerin toplamını temsil etmektedir.

$S_n=X_1+X_2+.....+X_n$ $X_i=2\varepsilon_i-1$ (4.8)

Örnek:

$\varepsilon=$ 1011010101 $X_1=2\ \varepsilon_1 -1 = 2.1-1=1$

$\varepsilon_1,\varepsilon_2,....., \varepsilon_{10}$ $X_2=2\ \varepsilon_2 -1 = 2.0-1=-1$

n=10 ve $S_n=1+(-1)+1+1+(-1)+1+(-1)+1+(-1)+1=2$ olur.

$$S_{obs} = \frac{|S_n|}{\sqrt{n}} = \frac{2}{\sqrt{10}} = 0.6324.. \quad (4.9)$$

erfc, hata fonksiyonu göstermek üzere,

$P_{değeri} = \text{erfc}\left(\frac{S_{obs}}{\sqrt{2}}\right) = 0.527089 \geq 0.01$ ise dizi rasgeledir.

4.4.2. Blok Frekans Testi

Dizide bulunan 0 ve 1'lerin oranını m bitlik bloklar içinde kontrol eder. Parametre m'dir. Blok uzunluğu 1 bit olursa test frekans testine dönüşür. Her bir bloktaki 1'lerin beklenen oranı m/2'dir. Kullanılan referans dağılımı ki-kare dağılımıdır. Testin iyi sonuç vermesi için dizi uzunluğu en az 100 bit blok uzunluğunda olmalıdır.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,....., \varepsilon_n$), n bit dizisinin uzunluğunu, M blok uzunluğu ve X^2(obs) beklenen oran (1/2) ile karşılaştırılan verilmiş m bit blok içerisindeki 1'lerin gözlemlenen oranının nasıl olduğunun ölçüsünü temsil etmektedir.

a) $N = \frac{n}{m}$ örtüşmeyen bloklar halinde girilen dizi bölünür. Kullanılmayan bitler atılır.

   n=10 m=3 ε=0110011010 $N = \frac{10}{3} = 3$

   011, 001 ve 101 bloklarından oluşur.

b) $\pi_i = \frac{\sum_{j=1}^{m} \varepsilon_{(i-1)m+1}}{m}$ denklemi kullanılarak her bir m bit bloktaki 1'lerin oranı π_i'ye karar verilir. $1\leq i\leq N$, $\pi_1 = \frac{2}{3}, \pi_2 = \frac{1}{3}, \pi_3 = \frac{2}{3}$

c) X^2 istatistiği, $X^2(obs) = 4\ m \sum_{i=1}^{N}(\pi_i - \frac{1}{2})^2$ denklemi ile hesaplanır.

$$X^2(\text{obs}) = 4 * 3 * \left(\left(\frac{2}{3} - \frac{1}{2}\right)^2 + \left(\frac{1}{3} - \frac{1}{2}\right)^2 + \left(\frac{2}{3} - \frac{1}{2}\right)^2\right) = 1$$

d) $P_{\text{değeri}} = \text{igamc}\left(\frac{N}{2}, \frac{X^2(\text{obs})}{2}\right) = \text{igamc}\left(\frac{3}{2}, \frac{1}{2}\right) = 0.801252 \geq 0.01$
ise dizi rasgeledir.

4.4.3. Akış Testi

Bu test dizideki akışların toplam sayısı ile ilgilidir, burada akış, sürekli (devamlı) aynı (özdeş) bit dizisi olarak tanımlanır.

K uzunluğundaki akış tamamen k özdeş bitler içerir ve bu bit dizisi başında ve sonunda kendinden farklı bir bit ile sınırlıdır. Akış testinin amacı rasgele olması beklenilen 0 ve 1'lerin akış sayılarının beklendiği gibi mi yoksa beklenenden farklı mı olduğunun araştırılmasıdır. Özellikle, 0'lar ve 1'ler arasındaki osilasyonun hızlı veya yavaş olup olmaması ile ilgili bilgiler verir.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon = \varepsilon_1, \varepsilon_2, \ldots\ldots, \varepsilon_n$), n bit dizisinin uzunluğunu ve $V_n(\text{obs})$akışların toplam sayısını (Tüm n bitlerin arasında toplam tekrar sayısı (yani, sıfırların toplam tekrarı + birlerin toplam tekrarı)) temsil etmektedir.

Bu test için referans dağılımı X^2 dağılımıdır.

a) $\pi = \frac{\sum_j \varepsilon_j}{n}$ dizideki 1'lerin sayısını hesaplar.
Örneğin: ε=1001101011 n=10 ve $\pi = \frac{6}{10} = \frac{3}{5}$

b) Eğer ön şart olarak frekans testi geçildi ise karar verilir:
$\left|\pi - \frac{1}{2}\right| \geq \tau$ gösterilir. Eğer bu durum gerçekleşmez ise akış testi uygulanmayabilir.
Test uygulanmaz ise p değeri 0.000 olur. Bu test için, $\tau = \frac{2}{\sqrt{n}}$ test kodu önceden tanımlanmıştır.
Bu bölümdeki örnek için: $\tau = \frac{2}{\sqrt{10}} = 0.63246$ olduğundan dolayı,
$\left|\pi - \frac{1}{2}\right| = \left|\frac{3}{5} - \frac{1}{2}\right| = 0.1 < \tau$ olur ve test çalışmaz.

c) $V_n(\text{obs}) = \sum_{k=1}^{n-1} r(k) + 1$, burada $\varepsilon_k = \varepsilon_{k+1}$ ise r(k)=0 değilse r(k)=1 olur.
Örneğin: ε=1001101011 V_10 (obs)=(1+0+1+0+1+1+1+1+0)+1=7

d) P değeri hesapla= $\text{erfc}\left(\frac{|V_n(obs)-2n\pi(1-\pi)|}{2\sqrt{2n}\pi(1-\pi)}\right)$= $\text{erfc}\left(\frac{\left|7-2\ 10\ \frac{3}{5}\left(1-\frac{3}{5}\right)\right|}{2\sqrt{210}\ \frac{3}{5}\left(1-\frac{3}{5}\right)}\right) \geq 0.01$

ise dizi rasgeledir.

Bu testin uygulanabilmesi için en az 100 bit dizi uzunluğu olmalıdır. $V_n(obs)$'un büyük değerleri için dizide osilasyon hızlı, küçük değerleri için yavaştır (bir osilasyon 1'den 0'a veya tersine bir değişimdir). Değişim ne kadar çok gerçekleşiyor ise osilasyon o kadar hızlı gerçekleşiyordur. Örneğin 010101010 her bit de dalgalıdır. Yavaş dalgalı bir akış, rasgele bir sırada beklenenden daha az çalışabilir. 100 tane 1,devamında 73 tane 0 ve devamında da 127 tane 1'i kapsayan bir sıra (toplam 300 bit) sadece 3 defa çalışabilir, hâlbuki 150 defa çalışması beklenir.

4.4.4. Bloktaki En Uzun Birlerin Akış Testi

Bu test, bir dizi içerisindeki en uzun birlerin akışının gözlemlenmesine odaklanır. Testin tek parametresi blok uzunluğudur (m). Dizi m bitlik n tane bloğa bölünür ve her blok içerisindeki en uzun birlerin akışına bakılır. Elde edilen değerlerin frekansları beklenen değerler ile karşılaştırılır ve sapma olup olmadığına bakılır.
Dizi uzunluğuna bakılarak blok uzunluğu ve blok sayısına karar verilir.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,\ldots\ldots, \varepsilon_n$), n bit dizisinin uzunluğunu, m her bir bloğun ve N örtüşmeyen blokların sayısını temsil etmektedir. Testte alınacak olan n ve m değerleri Tablo 4.3'te verilmiştir.

Tablo 4.3. Minimum n ve m sayıları

Minumum n	m
128	8
6272	128
750000	10^4

a) M bitlik bloklar halinde dizi bölünür.
b) Kategori halinde her bir blok için en uzun birlerin akışının frekansı v_i Tablo 4.4'te gösterildiği gibidir. Burada her bir hücre belirli uzunluktaki birlerin akışının sayısını içerir.

Tablo 4.4. Her bir blok için en uzun birlerin akış frekans değerleri

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

c) $X^2(obs) = \sum_{i=0}^{k} \frac{(v_i - N\pi_i)^2}{N\pi_i}$ (4.10)

Tablo 4.5'te verilen K ve N değerlerine göre m değeri belirlenir. Tablo 4.6'da ise verilen dizi için 1 akış gösteren blokların gözlenen sayısı verilmiştir.

Tablo 4.5. K ve N değerlerine göre belirlenecek m değerleri

M	K	N
8	3	16
128	5	49
10^4	6	75

N değeri toplam dizi sayısının blok sayısına bölümünden bulunmaktadır.

$$X^2(obs) = \frac{(4-16(0.2148))^2}{16(0.2148)} + \frac{(9-16(0.3672))^2}{16(0.3672)} + \frac{(3-16(0.2305))^2}{16(0.2305)} + \frac{(0-16(0.1875))^2}{16(0.1875)} = 4.882605$$

d) $P_{değeri} = igamc\left(\frac{K}{2}, \frac{X^2(obs)}{2}\right) = igamc\left(\frac{3}{2}, \frac{4.882605}{2}\right) = 0.180 \geq 0.01$

ise dizi rasgeledir.

Örnek: K=3, M=8 ve n=128

ε: 11001100000101010110110001001100111000000000001001
00110101010001000100111101011010000000110101111100
11001110011011011000010110010

Tablo 4.6. v_0 alt blok olarak ayrılan ve tüm dizi içinde 1 akış gösteren blokların gözlenen sayısı

Alt Blok	Mak.Akış	Alt Blok	Mak.Akış
11001100	2	00010011	2
00010101	1	11010110	2
01101100	2	10000000	1
01001100	2	11010111	3
11100000	3	11001100	2
00000010	1	11100110	3
01001101	2	11011000	2
01010001	1	10110010	2

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

v_1 tüm dizide 2 tane arka arkaya 1'lerin akışını gösteriri, v_2 tüm dizi içinde 3 akış gösteren blokların gözlenen sayısıdır. $P_{\text{değeri}}$=0.180609≥0.01 ise dizi rasgele olarak kabul edilir.

4.4.5. İkili Matris Rankı Testi

Bu test, bütün dizinin ayrık alt matrislerinin ranklarına odaklanır. Testin amacı, orijinal dizinin alt dizileri arasında doğrusal bir bağıntı olup olmadığını araştırmaktır. Test içerisinde sıra M x M-bitlik matrisler halinde parçalanır ve oluşturulan her bir matrisin rankı hesaplanır. Sıra ile oluşturulan matrislerin ranklarının frekansları hesaplanır, beklenen frekansla karşılaştırılır ve ciddi bir sapma olup olmadığı kontrol edilir. Bu test Diehard test suitinde de bulunmaktadır.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi (ε= $\varepsilon_1,\varepsilon_2,\ldots\ldots,\varepsilon_n$), n bit dizisinin uzunluğunu, M her bir matrisin satır sayısını, Q her bir matrisin sütun sayısını, $X^2(\text{obs})$ gözlenen farklı sınıf sıralarının(ranklarının) sayısı ile bir rasgelelik varsayımı altındaki sıraların (rankların) beklenen sayısının ne kadar uyumlu olduğunun bir ölçümünü temsil etmektedir.

Bu testte M ve Q değerleri 32 olarak alınmıştır. Kullanılan diğer M ve Q değerleri için, yeni yaklaşımların hesaplanması gerekir.

a) Bit dizisi M x Q bit ayrık matrislere bölünür. Burada matris sayısı: N=[n/MQ]. Bu matrisler oluşturulduktan sonra geri kalan bitler atılır.
   Örneğin: n=20, M=Q=3 için ve ε= 01011001001010101101 alınırsa N=2 olur. M*Q=9 ve 2 matris için 18 bit alınır son 2 bit atılır.
   Oluşacak olan matrisler:

$$1.\text{matris:}\begin{vmatrix}0 & 1 & 0\\ 1 & 1 & 0\\ 0 & 1 & 0\end{vmatrix}\quad 2.\text{matris:}\begin{vmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 1\end{vmatrix}$$

   İlk 3 bit ilk satırda, sonraki 3 bit 2 satırda ve diğer 3 bitte 3.satıra gelecek şekilde matris oluşturulur. Aynı şekilde 2. matriste oluşturulur.

b) Her bir matrisin ikili rankına (R_L) karar verilir. Karar vermede kullanılan yöntem aşağıda adım adım gösterilmiştir.

mXm matris a_{ij} olarak tasarlanmıştır.

1. i=1
2. Eğer $a_{i,j}$=0 ise (diagonal(köşegen,çapraz)≠1), bir sonraki satırda i.sütununda1 içeren satırın tüm elemanları i.satırın tüm elemanlar ile yer değiştirir. Eğer 1 içermiyorsa adım 4'e gidilir.
3. Eğer $a_{i,j}$=1 İse,sonra gelen satırların ilk sütununda 1 içeren satırlarla ilk satır XOR işlemine tabi tutulur.
   I. row=i+1
   II. col=i
   III. eğer $a_{row,col}$=0 ise 3 adımın VII. Aşamasına git
   IV. $a_{row,col}= a_{row,col} \oplus a_{i,col}$
   V. eğer col=m ise 3 adımın VII. Aşamasına git
   VI. col=col+1; 3 adımın IV. aşamasına git
   VII. eğer row=m ise 4.adıma git
   VIII. row=row+1; 3 adımın II. aşamasına git
4. eğer i<m-1 ise i=i+1 ve 2 adıma git.
5. Geriye satır işlemi tamamlanır.
6. Matrisin rankı satırı sıfır olmayanların sayısıdır.

c) F_M= (R_L=M) ile matrislerinin sayısıdır. Yani rankı M değerine eşit olanların sayısıdır.

F_{M-1}= (R_L=M-1) ile matrislerinin sayısıdır. Yani rankı M-1 değerine eşit olanların sayısıdır.

N-F_M-F_{M-1}= geriye kalan matrislerin sayısıdır.

d) Ki kare değeri hesaplanır.

$$X^2(\text{obs}) = \frac{(F_M - 0.2888N)^2}{0.2888N} + \frac{(F_{M-1} - 0.5776N)^2}{0.5776N} + \frac{(N - F_M - F_{M-1} - 0.1336N)^2}{0.1336N} \quad (4.11)$$

e) $\text{pdegeri} = e^{-X^2(\text{obs})/2}$ hesaplanır. Burada örnekte üç sınıf olduğundan dolayı, pdeğeri, igamc(1,X^2(obs)/2) eşittir.

4.4.6. Ayrık Fourier Dönüşümü

Bu test Spektral test olarak da adlandırılır ve dizinin ayrık fourier dönüşümünün tepe yüksekliklerine odaklı bir testtir. Bu testin amacı rasgelelilik varsayımından bir sapma gösteren, test edilen sıradaki periyodik özellikleri (yani, birbirine yakın olan tekrarlı kalıpları) tespit etmektir. Tepe yüksekliklerinin %95 i aştığı durumlar rasgelelilik için iyi sayılmaktadır.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,\ldots\ldots, \varepsilon_n$), n bit dizisinin uzunluğunu, d gözlenen ve beklenen %95 eşik değerinin üstündeki frekans bileşenlerinin gözlenen ve beklenen sayıları arasındaki standart farklılığı temsil etmektedir

a) Bit dizisindeki 0 ve 1'ler -1 ve +1 değerlerine dönüştürülerek $X=x_1,x_2,\ldots,x_i$ dizisi yaratılır. Burada $xi=2\ \varepsilon_i-1$
   Örneğin n=10 ve ε=1001010011 sonra X=1,-1,-1,1,-1,1,-1,-1,1,1 olur.
b) X'e ayrık fourier dönüşümü (AFD) uygulanarak S=AFD(X) üretilir. Karmaşık değişkenler dizisi farklı frekanslardaki bit dizisinin periyodik bileşenlerinden üretilmiştir.

$$S_j = \sum_{k=1}^{n} X_k \exp^{\left(\frac{2\pi i(k-1)j}{n}\right)} \tag{4.12}$$

Burada

$$\exp^{\left(\frac{2\pi i(k-1)j}{n}\right)} = \cos\left(\frac{2\pi kj}{n}\right) + i \sin\left(\frac{2\pi kj}{n}\right) \tag{4.13}$$

j=0,1,.....,n-1 ve $i\equiv\sqrt{-1}$

c) M=modulus(S_j')≡$|S_j'|$ burada S', S'deki ilk n/2 elemandan oluşan alt dizidir ve modulus fonksiyonu tepe yüksekliklerinin bir dizisini üretir.
d) $T = \sqrt{3n}$ = %95 tepe yüksekliği eşik değeri hesaplanır. Rasgelelilik varsayımı altında, bu testte elde edilen %95 değerleri T 'yi geçmemelidir.
e) N_0=0.95*n/2 hesaplanır. N_0 T'den daha küçük tepe yüksekliklerinin beklenen teorik sayısıdır.

Bu örnek için N_0=4.75.

f) T den daha küçük kaç tepe değeri olduğu kolayca sayılır, N_1 olarak ifade edilir.

g) Test istatistiği :

$$d = \frac{(N_1 - N_0)}{\sqrt{\frac{0.95*0.05*n}{2}}} \quad (4.14)$$

bu kısımdaki örnek için : $d = \frac{(4-4.75)}{\sqrt{\frac{0.95*0.05*10}{2}}} = -1.538$

h) P değeri=erfc(|d| / 2) hesaplanır. Eğer p değeri > 0.01 olursa dizi testi geçer.

4.4.7. Örtüşmeyen Şablon Eşleştirme Testi

Bu test, önceden belirlenmiş hedef dizisinin bulunma sıklığının gözlenmesine dayanır. Bu testin amacı, üretecin oluşturduğu periyodik olmayan örneklerin tespit edilmesidir. Bu test ve bir sonraki test olan örtüşen şablon eşleştirme testlerinde, m bitlik bir örneği aramak için m bitlik bir pencere kullanılır. Eğer bu örnek bulunmaz ise pencere bir bit kaydırılarak arama işlemine devam edilir. Eğer örnek bulunur ise, pencere bulunan örnekten sonraki ilk bite yeniden konumlanır ve arama işlemi devem eder.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon = \varepsilon_1, \varepsilon_2, ..., \varepsilon_n$), n bit dizisinin uzunluğunu, m her bir şablonun bit uzunluğu, 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 sayısını temsil etmektedir.

Örnek:

a) ε = 10100100101010010010 n=20, N=2 ve M=n/N=20/2=10
20 bitlik dizi 10 bitlik 1010010010 ve 1010010010 iki bloğa ayrılır.

b) W_j (j=1,2,.....,N) j blok içindeki B (şablonun) kaç kez bulunduğunun sayısıdır. Tablo 4.7'de b şablonun kaç kez bulunduğu gösterilmiştir.
Eğer m=3 ve şablon B=001 ise;

Tablo 4.7. b şablonunun kaç kez bulunduğunun gösterimi

Bit Pozisyonları	Blok 1		Blok 2	
	Bits	W_1	Bits	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	001 (bulundu)	1
5-7	Test edilmedi		Test edilmedi	
6-8	Test edilmedi		Test edilmedi	
7-9	001	2	011	1
8-10	010	2	110	1

Böylece W_1=2 ve W_2=1 olur.

c) Rasgelelilik varsayımları altında, ortalama μ ve varyans σ^2 hesaplanır.

$$\mu = \frac{(M-m+1)}{2^n} \qquad \sigma^2 = M\left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right) \tag{4.15}$$

$\mu=(10-3+1)/2^3=1$ ve $\sigma^2=10(\ (1/2^3\ )-\ (2*3-1)/2^{2*3})=0.46875$

d) Ki kare değeri hesaplanır.

$$X^2(obs) = \sum_{j=1}^{N} \frac{(W_j-\mu)^2}{\sigma^2} \tag{4.16}$$

Örnekteki değerlere göre: $X^2(obs) = \frac{(2-1)^2-(1-1)^2}{0.46875} = 2.13333$

e) P değeri hesaplanır.

$$P_{degeri} = \text{igamc}\left(\frac{N,}{2}\ \frac{X^2(obs)}{2}\right) = \left(\frac{2}{2}, \frac{2.13333}{2}\right) = 0.344154 \tag{4.17}$$

M değeri 2,3,....10 arasında alınabilir. Ancak tavsiye edilen m=9 ve m=10 da daha olumlu sonuçlar alınmıştır. N genellikle 8 alınmaktadır.

4.4.8. Örtüşen Şablon Eşleştirme Testi

Bu test, önceden belirlenmiş hedef dizisinin bulunma sıklığının gözlenmesine dayanır. Testin amacı, bir önceki testte tanıtılan üretecin oluşturduğu birçok periyodik olmayan

örneğin aranmasıdır. Bu ve bir önceki testte tanıtılan Örtüşmeyen Şablon Eşleştirme testlerinde, m bitlik bir örneği aramak için m bitlik bir pencere kullanılır. Eğer aranan örnek bulunmazsa, pencere bir bit kaydırılarak tarama işlemine devam edilir. Bu testin bir önceki testten tek farkı, eğer örnek bulunursa pencere bulunan örüntüden sonraki ilk bit yerine sadece o an bulunan pozisyondan bir sonraki bite yeniden konumlanır ve aramaya devam edilir.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,.....,\varepsilon_n$), n bit dizisinin uzunluğunu, m her bir şablonun bit uzunluğu, B eşleştirilecek m bit şablon (B test kodu içerisinde bulunan periyodik olmayan örnekler, şablon kütüphanesinde tanımlı 0 ve 1'lerden oluşan dizidir), K bağımsızlık derecesi (K=5 olarak sabitlenmiştir), M test edilecek ε alt dizilerinin bit uzunluğu (M=1032 alınmıştır), N bağımsız blokların sayısını (N=968 alınmıştır) temsil etmektedir.

Örnek:

a) ε = 10111011110010110100011100101110111110000101101001 n=50 K=2 M=10 ve N=5 alınmıştır. Sonra dizi 1011101111, 0010110100, 0111001011, 1011111000 ve 0101101001 şeklinde 5 bloğa bölünmüştür.

b) Her bir N bloktaki B şablonunun sayısı Tablo 4.8'de gösterildiği gibi hesaplanır. m=2 için B=11 alınırsa ve ilk blokta 1011101111 aranırsa;

Tablo 4.8. B=11 için bulunma sayısı

Bit Pozisyonu	Bitler	B = 11 in bulunma sayısı
1-2	10	0
2-3	01	0
3-4	11 (bulundu)	1
4-5	11 (bulundu)	2
5-6	10	2
6-7	01	2
7-8	11 (bulundu)	3
8-9	11 (bulundu)	4
9-10	11 (bulundu)	5

İlk blokta arama yapılır. 11'e beş defa rastlanmıştır. v_5 arttırılır ve $v_1=0$, $v_2=0$, $v_3=0$, $v_4=0$ ve $v_5=1$.

c) λ ve η değerleri hesaplanır.

$$\lambda =(M-m+1)/2^m \quad \eta= \lambda /2 \tag{4.18}$$

Bu örnek için $\lambda = (10-2+1)/2^2 = 2.25$ ve $\eta = 2.25/2 = 1.125$.

d) Tablo 4.9'da gösterilen V_0 sınıflarına bağlı olasılıklara göre ki kare değeri hesaplanır.

Tablo 4.9. V_0 sınıflarına bağlı olasılıklar

$\pi 0$	0.364091
$\pi 1$	0.185659
$\pi 2$	0.139381
$\pi 3$	0.100571
$\pi 4$	0.0704323
$\pi 5$	0.139865

$$X^2(\text{obs}) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i} \tag{4.19}$$

$$X^2(\text{obs}) = \frac{(0-5*0.324652)^2}{5*0.324652} + \ldots + \frac{(0-5*0.166269)^2}{5*0.166269} = 3.167729$$

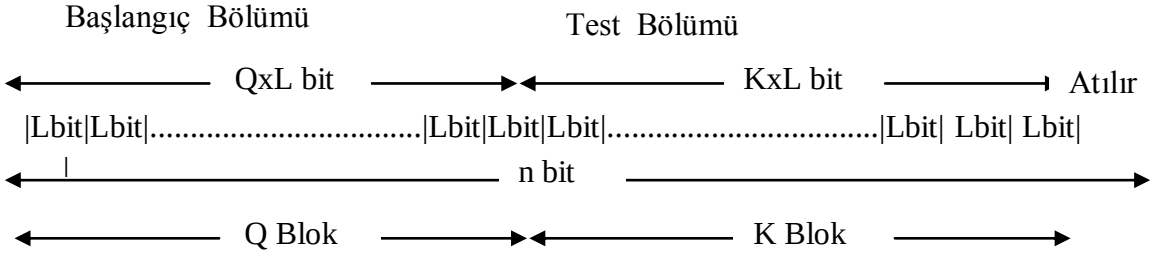
e) P değeri hesaplanır.

$$P_{\text{degeri}} = \text{igamc}\left(\frac{K}{2}, \frac{X^2(\text{obs})}{2}\right) = \left(\frac{5}{2}, \frac{3.167729}{2}\right) = 0.274932 \tag{4.20}$$

4.4.9. Maurer Evrensel İstatistiksel Testi

Bu test, dizide eşleşen örneklerin bit sayılarının gözlemlenmesine dayanır. Testin amacı, dizinin bilgi kaybı olmadan sıkıştırılıp sıkıştırılamayacağının tespit edilmesidir. Anlamlı bir şekilde sıkıştırılan bir dizi rasgele olarak kabul edilmez.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$), n bit dizisinin uzunluğunu, L her bloğun uzunluğu ve $f_n \log_2$ toplamı gözlenen L bit şablon arasındaki uzaklığı temsil etmektedir.



Örnek: ε = 01011010011101010111 n=20,L=2 Q=4 K=[n/L]-Q=[20/2]-4=6

Başlangıç bölümü 01011010; test bölümü 011101010111 olur. Mümkün olan L değerleri Tablo 4.10'da belirtilmiştir.

Tablo 4.10. Mümkün olan L değerleri

	Mümkün olan L değerleri			
	00 (T_0'da kayıtlı)	**01 (T_1'de kayıtlı)**	**10 (T_2'de kayıtlı)**	**11 (T_3'de kayıtlı)**
Başlangıç	0	2	4	0

1 den Q ya kadar T_j=i oluşturulur. J burada i. bloğun ondalık sayısıdır. Oluşturulan T_j dizilerinden kaç tane olduğu tespit edilir.

5. blok için (ilk test bloğu): 5 Tablo 4.11'deki satırda 01 olarak belirtilmiştir (ör, T_1), ve sum=$\log_2$(5-2) = 1.584962501. ve T_1'in yeni değeri blok numarası olmuştur (yani 5).

6. blok için : 6 Tablo 4.11'deaki satırda 11 olarak belirtilmiştir (ör., T_3), ve sum = 1.584962501+$\log_2$(6-0) = 1.584962501 + 2.584962501 = 4.169925002 ve T_3'ün yeni değeri blok numarası olmuştur (yani 6).

7.blok için : 7 Tablo 4.11'deki satırda 01 olarak belirtilmiştir (ör., T_1), ve sum = 4.169925002 + $\log_2$(7-5) = 4.169925002 + 1 = 5.169925002. ve T_1'in yeni değeri 7 olmuştur.

8.blok için : 8 Tablo 4.11'deki satırda 01 olarak belirtilmiştir (T_1) ve sum = 5.169925002 + $\log_2$(8-7) = 5.169925002 + 0 = 5.169925002. ve T_1'in yeni değeri 8 olmuştur.

9.blok için: 9 Tablo 4.11'deki satırda 01 olarak belirtilmiştir (ör., T_1), ve sum = 5.169925002 + $\log_2$(9-8) = 5.169925002 + 0 = 5.169925002.

10.blok için: 10 Tablo 4.11'deki satırda 11 olarak belirtilmiştir (ör.,T_3), ve sum = 5.169925002 + $\log_2$(10-6)= 5.169925002 + 2 = 7.169925002.

Tablo 4.11. Maurer evrensel istatistiksel test bölümleri

Blok	**Tip**	**İçerik**
1	Başlangıç Bölümü	01
2		01
3		10
4		10
5	Test bölümü	01
6		11
7		01
8		01
9		01
10		11

Tekrar eden bloklar ve L değerleri Tablo 4.12'de verilmiştir.

Tablo 4.12. Tekrar bloğu ve L değerleri

Tekrar Bloğu	Mümkün olan L bit değerleri			
	00	**01**	**10**	**11**
4	0	2	4	0
5	0	5	4	0
6	0	5	4	6
7	0	7	4	6
8	0	8	4	6
9	0	9	4	6
10	0	9	4	10

$f_n = \frac{1}{K}\sum_{i=Q+1}^{Q+K} \log_2(i - T_j)$ hesaplanır.

Örnek için $f_n = \frac{7.169925002}{6} = 1.1949875$

Pdegeri $= \left(\left|\frac{f_n - beklenendeger(L)}{\sqrt{2}\,\sigma}\right|\right)$ hesaplanır. L değerleri için beklenen değer ve varyans Tablo 4.13'te verilmiştir.

Tablo 4.13. L değerleri için beklenen değer ve varyans

L	Beklenen Değer	Varyans
6	5.2177052	2.954
7	6.1962507	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

Örnek için: Pdegeri $= \left(\left|\frac{1.1949875 - 1.5374383}{\sqrt{2}\ 1.338}\right|\right) = 0.767189$ hesaplanır.

4.4.10. Lempel Ziv

Bu test, bir dizide bulunan kümülatif olarak ayrık örneklerin (kelimelerin) sayısına odaklanır. Amaç, test edilen dizinin ne kadar sıkıştırılabildiğinin belirlenmesidir. Eğer dizi anlamlı bir şekilde sıkıştırılamazsa dizi rasgele olarak kabul edilir.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,.....,\varepsilon_n$), n bit dizisinin uzunluğunu ve W_{obs} dizideki birbirinden ayrılmış ve kümülatif farklı örneklerin (kelimelerin) sayısını temsil etmektedir.

a) Tablo 4.14'te gösterilen kurala göre sözlük oluşturulur.

ε= 010110010

Tablo 4.14. Lempel Ziv sözlük oluşumu

Sözlük	Son Durum	Geçerli Durum	Giriş	Kod Çıkışı	
0 = 1		ε_0=0			Giriş Kısmında elde edilen değer sözlükte yoksa Son Durumdaki Değerlerin sözlükteki karşılık gelen değerleri kod çıkışına yazılır.
1 = 2	0	ε_1=1	01=3 (Sözlükte yok ekle)	1	
01 = 3	1	ε_2=0	10=4 (Sözlükte yok ekle)	2	
10 = 4	0	ε_3=1	01 (Sözlükte Mevcut)		
011 = 5	01	ε_4=1	011=5 (Sözlükte yok ekle)	3	
100 = 6	1	ε_5=0	10 (Sözlükte Mevcut)		
010 = 7	10	ε_6=0	100=6 (Sözlükte yok ekle)	4	
	0	ε_7=1	01 (Sözlükte Mevcut)		
	01	ε_7=0	010=7 (Sözlükte yok ekle)	3	
	0	Son		1	

Tablo 4.14'te gösterilen kurala göre oluşturulan Lempel Ziv kod çıkış örneği Tablo 4.15'te gösterilmiştir.

Tablo 4.15. Lempel Ziv kod çıkış örneği

Kod Çıkışı	1	2	3	4	3	1
Sözlük Değeri	0	1	01	10	01	0
Örnek Dizi	010110010					

W_{obs}=5 (Sözlüğe yeni eklenen değerlerin sayısı)

b) Pdegeri = $^1/_2$ erfc($\frac{\mu-W_{obs}}{\sqrt{2\sigma^2}}$) burada n=$10^6$ için μ=69586.25 ve $\sigma = \sqrt{70.448718}$ dir. N'in diğer değerleri için Sha-1 kullanılarak ortalama(μ) ve varyans (σ) değerleri hesaplanabilir.

4.4.11. Doğrusal Karmaşıklık Testi

Bu test genel olarak, Doğrusal Geri beslemeli Kayan Yazmaç (DGBKY) uzunluğuna odaklıdır. Testin amacı, rasgele olduğu iddia edilen dizinin yeterince karmaşık olup olmadığının belirlenmesidir. Rasgele diziler daha uzun DGBKY ile karakterize edilir. Çok kısa DGBKY'ler büyük olasılıkla rasgele olmayan dizileri gösterir.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,\ldots\ldots, \varepsilon_n$), n bit dizisinin uzunluğunu ve M blok uzunluğunu etmektedir.

İlk olarak dizi her biri M uzunlukta N tane bloğa bölünür. Sonra her blokta Tablo 4.16'da verilmiş olan Berlekamp Massey algoritması uygulanır. Tablo 4.17'de örneği verilen Berlekamp Massey algoritması belirli ikili dizinin DGBKY' ın geri beslemeli polinomunun en düşük derecesini hesaplar. I. bloğun geri besleme polinomunun L en düşük derecesi kaydedilir. Her bir blok için T_i hesaplanır.

$$T_i = (-1)^M * (L - \mu) + {}^2/_9 \tag{4.21}$$

μ teorik ortalama:

$$\mu = \frac{M}{2} + \frac{(9+(-1)^{M+1})}{36} + \frac{({}^M/_3 + {}^9/_2)}{2^M} \tag{4.22}$$

T_i'nin dağılımı -2.5 ile 2.5 arasında yoğunlaşır.

$T_i \leq -2.5\ v_0 = v_0 + 1\ \ -2.5 < T_i \leq -1.5\ v_1 = v_1 + 1\ -1.5 < T_i \leq -0.5\ v_2 = v_2 + 1$
$-0.5 < T_i \leq 0.5\ v_3 = v_3 + 1\ \ 0.5 < T_i \leq 1.5\ v_4 = v_4 + 1\ 1.5 < T_i \leq 2.5\ v_5 = v_5 + 1$
$2.5 < T_i\ v_6 = v_6 + 1$

$\pi_0, \pi_1, \ldots\ldots \pi_6$ olasılık değerleri önceden hesaplanmıştır.

$\pi_0 = 0.010417\ \pi_1 = 0.3125\ \pi_2 = 0.125\ \pi_3 = 0.5\ \pi_4 = 0.25\ \pi_5 = 0.625\ \pi_6 = 0.2083$

Ki kare değeri:

$$X^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)}{N\pi_i} \tag{4.23}$$

$$\text{pdegeri} = \text{igamc}(\frac{K}{2}, \frac{X^2(obs)}{2}) \quad (4.24)$$

Tablo 4.16. Berlekamp Massey Algoritması

Alınan ikili dizi s=(s_0,s_1,.....,s_{n-1})
Doğrusal karmaşıklık L ve bağlantı polinomu C(x) aranır.
BM(s)
C(x)=B(x)=1
L=N=0
m=-1
While N<n n giriş dizisinin uzunluğudur.
$d=s_N \oplus c_1 s_{N-1} \oplus c_2 s_{N-2} \oplus ..... \oplus c_L s_{L-1}$
if (d==1)
T(x)=C(x)
$C(x)=C(x)+B(x)\, x^{N-m}$
if (L<N/2)
L=N+1-L
m=N
B(x)=T(x)
End if
End if
N=N+1
End while
Return(L)
End BM

Tablo 4.17. Berlekamp Massey Algoritması Örnek [69].

N	S_N	D	L	C(x)	M	B(x)
			0	1	-1	1
0	0	0	0	1	-1	1
1	1	1	2	$1+x^2$	1	1
2	1	1	2	$1+x+x^2$	1	1
3	1	1	2	1+x	1	1
4	1	0	2	1+x	1	1
5	0	1	4	$1+x+x^4$	5	1+x
6	0	0	4	$1+x+x^4$	5	1+x

4.4.12. Seri Testi

Bu test, tüm dizideki m-bitlik örtüşen olası örneklerin frekansına odaklanır. Testin amacı, 2^m adet m-bitlik örtüşen örneklerin sayısının, rasgele bir diziden beklenen sayıya ne kadar yakın olduğunun incelenmesidir. Rasgele dizilerde tek-biçimlilik (uniformity)

önemli bir özelliktir. Bu tek-biçimlilikte her m-bitlik örneğin diğer m-bitlik örnekler gibi ortaya çıkma olasılığının aynı olması beklenir. m=1 için seri testi 1. testte açıklanan frekans testine denk düşer.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi (ε= $\varepsilon_1,\varepsilon_2,\ldots\ldots,\varepsilon_n$), n bit dizisinin uzunluğunu, m her bir bloğun uzunluğu, $\nabla\psi^2(\text{obs})$ m bit örneğin frekansının gözlemlenmesi ve $\nabla^2\psi^2(\text{obs})$ m bit örneğin beklenen frekansını temsil etmektedir.

a) ε': dizininin sonuna m-1 bit kadar ekleme yapılır.
   Örneğin verilen dizi n=10 ve ε=0011011101 olsun. Eğer m=3 ise ε'=001101110100. m=2 ise ε'=00110111010. m=1 ise dizinin orjinali ε'=0011011101 olur.
b) Bütün mümkün olan örtüşen m,m-1 ve m-2 bitlik blokların frekansına karar vermek. $i_1,\ldots,i_m$ 'e kadar olan m bitlik örneğin frekansı $v_{i_1\ldots i_m}$, $i_1,\ldots,i_{m-1}$ 'e kadar olan m-1 bitlik örneğin frekansı $v_{i_1\ldots i_{m-1}}$ ve $i_1,\ldots,i_{m-2}$ 'e kadar olan m -2 bitlik örneğin frekansı $v_{i_1\ldots i_{m-2}}$olarak ifade edilir.
   Bu bölümdeki örnek için, m=3 , (m-1)=2 ve (m-2)=1. Bütün 3 bitlik blokların frekansı: v_{000} =0, v_{001} =1, v_{010} =1, v_{011} =2, v_{100} =1, v_{101} =2, v_{110} =2, v_{111} =0. 2 bitlik bloklar için: v_{00} =1, v_{01} =3, v_{10} =3, v_{11} =3. 1 bitlik bloklar için: v_0 =4, v_1 =6.
c) Hesaplanır:

$$\psi_m^2 = \frac{2^m}{n}\sum_{i_1..,i_m} v_{i_1..,i_m}^2 - n \qquad (4.25)$$

$$\psi_{m-1}^2 = \frac{2^{m-1}}{n}\sum_{i_1..,i_{m-1}} v_{i_1..,i_{m-1}}^2 - n \qquad (4.26)$$

$$\psi_{m-2}^2 = \frac{2^{m-2}}{n}\sum_{i_1..,i_{m-2}} v_{i_1..,i_{m-2}}^2 - n \qquad (4.27)$$

Örnek için:

$\psi_3^2=\frac{2^3}{10}(0+1+1+4+1+4+4+1)-10=2.8$

$\psi_2^2=\frac{2^2}{10}(1+9+9+9)-10=1.2$

$\psi_1^2=\frac{2^1}{10}(16+36)-10=0.4$

d) Hesaplanır :

$$\nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2 \text{ ve } \nabla^2\psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2 \quad (4.28)$$

Örnek için: $\nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2 = 2.8 - 1.2 = 1.6$

$\nabla^2\psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2 = 2.8 - 2(1.2) + 0.4 = 0.8$

e) Pdegeri=igamc(2^{m-2},$\nabla\psi_m^2/2$) ve Pdegeri1=igamc(2^{m-3},$\nabla^2\psi_m^2/2$) hesaplanır.

$\text{Pdegeri} = \left(2, \frac{1.6}{2}\right) = 0.9057$

$\text{Pdegeri1} = \left(1, \frac{0.8}{2}\right) = 0.8805$

4.4.13. Yaklaşık Entropi Testi

Seri testinde ki gibi, bu testte bütün dizideki m-bitlik örtüşen olası örneklerin frekansına odaklanır. Burada amaç, iki ardışık veya komşu uzunluktaki (m ve m+1 bit) örtüşen bloğun frekansını, rasgele bir dizinin beklenen frekansı ile karşılaştırılmasıdır.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon = \varepsilon_1, \varepsilon_2, \ldots\ldots, \varepsilon_n$), n bit dizisinin uzunluğunu, m her bir bloğun uzunluğu (Testte kullanılan ilk blok uzunluğudur), m+1 kullanılan 2. blok uzunluğudur ve X2 kullanılan referans dağılımını temsil etmektedir.

a) m-bitlik örtüşen diziler yaratırken m-1 bit dizi başlangıcı dizi sonuna eklenerek n-bit dizi çoğaltılır.

Örneğin, eğer ε=0100110101 ve m=3 ise, n=10 dur. Dizinin başlangıcındaki 0 ve 1 (m-1 bit) dizinin sonuna eklenir. Test edilecek dizi 010011010101 haline gelir.

b) Örtüşen blokların frekans hesabı yapılır. M bit bloklara bölünerek bu blokların diziden kaçar tane olduğu hesaplanır.

Bu kısımdaki örnekte; 010,100,001,011,110,101,010,101,010 ve 101. $2^m = 2^3 = 8$ olası m bit dizi için hesaplanan değer:

#000=0, #001=1, #010=3, #011=1, #100=1, #101=3, #110=1, #111=0

c) Her bir i değeri için $C_i^m = \frac{\#i}{n}$ hesaplanır.

Bu örnek için: $C_{000}^3 = 0$, $C_{001}^3 = 0.1$, $C_{010}^3 = 0.3$, $C_{011}^3 = 0.1$, $C_{100}^3 = 0.1$, $C_{101}^3 = 0.3$, $C_{110}^3 = 0.1$, $C_{111}^3 = 0$

d) $\varphi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i \log \pi_i$ hesaplanır. Burada $\pi_i = C_j^3$ ve j=$\log_2$ i.

$\varphi^{(3)}$=0(log0)+ 0.1(log0.1)+ 0.3(log0.3)+ 0.1(log0.1)+ 0.1(log0.1)+ 0.3(log0.3)+ 0.1(log0.1)+ 0(log0)

e) İlk 4 adım m+1 (4) bloklar içinde gerçekleştirilir.

f) Ki kare hesaplanır.

$$X^2 = 2n[\log 2 - \mathrm{ApEn}(m)] \tag{4.29}$$

burada

$$\mathrm{ApEn}(m) = \varphi^{(m)} - \varphi^{(m+1)} \tag{4.30}$$

g) Pdeğeri= igamc(2^{m-1}, $X^2/2$) (4.31)

4.4.14. Kümülatif Toplam Testi

Bu test, bir dizide düzenlenmiş -1,+1 dijitlerinin kümülatif toplamı olarak tanımlanmış, rasgele yürüyüş maksimal gezinimlerinin araştırılmasına odaklanır, amacı test edilen dizide bulunan kısmi alt dizilerin kümülatif toplamlarının, rasgele olduğu bilinen bir dizinin beklenen değerine göre çok küçük veya çok büyük olup olmadığının belirlenmesidir. Kümülatif toplamda rasgele yürüyüş olarak kabul edilebilir. Rasgele dizilerde rasgele yürüyüş gezinimleri sıfır civarındadır.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi (ε= $\varepsilon_1,\varepsilon_2,\ldots\ldots,\varepsilon_n$), n bit dizisinin uzunluğunu, z dizide kümülatif toplamların en büyük gezinimini temsil etmektedir. Mode ise, 0 seçilirse giriş dizisi testte ileri doğru uygulanır 1 seçilirse geriye doğru uygulanır.

Burada kullanılan referans dağılımı normal dağılımdır.

a) Dizi normalize edilir. Giriş dizisindeki (ε) sıfırlar ve birler X_i=2 ε_i-1 kullanılarak X_i değerleri -1 ve +1 dönüştürülür.

Örneğin: ε= 1011010111, X=1,(-1),1,1,(-1),1,(-1),1,1,1 olur.

b) S_i kısmi toplamları hesaplanır. Mode 0 ise X_i den mode 1 ise X_n den başlanır.

Mode =0 (İleri)	Mode=1 (Geriye)
$S_1=X_1$	$S_1=X_n$
$S_2=X_1+ X_2$	$S_2=X_n+ X_{n-1}$
$S_3=X_1+ X_2+ X_3$	$S_3= X_n+ X_{n-1}+ X_{n-2}$
.	.
.	.
$S_k= X_1+ X_2+ X_3+.....+X_k$	$S_k= X_n+ X_{n-1}+ X_{n-2}+.....+X_{n-k+1}$
.	.
.	.
$S_n= X_1+ X_2+ X_3+.....+X_k+....+X_n$	$S_n= X_n+ X_{n-1}+ X_{n-2}+.....+X_{k-1}+....+X_1$

Bu bölümdeki örnek için, mode 0 alındığında X=1,(-1),1,1,(-1),1,(-1),1,1,1

$S_1=1$
$S_2=1 + (-1) = 0$
$S_3=1 + (-1) + 1 = 1$
$S_4=1 + (-1) + 1 + 1 = 2$
$S_5=1 + (-1) + 1 + 1 + (-1) = 1$
$S_6=1 + (-1) + 1 + 1 + (-1) + 1 = 2$
$S_7=1 + (-1) + 1 + 1 + (-1) + 1 + (-1) = 1$
$S_8=1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 = 2$
$S_9=1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + 1 = 3$
$S_{10}=1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + 1 + 1 = 4$

c) Elde edilen S değerlerini mutlak değerli en büyük sayısı tespit edilir.
Bu örnek için $S_k=4$ tür, o halde z=4 olur.

d) P değeri hesaplanır.

$$p_{degeri} = 1 - \sum_{k=(\frac{-n}{z}+1)/4}^{(\frac{n}{z}-1)/4}\left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right] + \sum_{k=(\frac{-n}{z}-3)/4}^{(\frac{n}{z}-1)/4}\left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right] \quad (4.32)$$

Burada Φ standart normal kümülatif olasılık dağılım fonksiyonudur.
Bu örnek için p_degeri=0.4116588.

4.4.15. Rasgele Yürüyüş Testi

Bu test, kümülatif toplam rasgele yürüyüşündeki tam olarak K ziyaretlik döngünün sayısına odaklanır. Kümülatif toplam rasgele yürüyüşü, (0,1)'lerden oluşan dizinin (-1,+1) olarak düzenledikten sonra kısmi toplamlarından elde edilir. Bir rasgele yürüyüş döngüsü, rasgele kabul edilen bir yerden başlayıp tam bir döngü olana kadar belli bir uzunluktaki adım dizisinden oluşur. Bu testin amacı, bu döngü sırasında rasgele dizide beklenen sapmadan kaynaklanan belirli bir durumun ziyaretlerinin sayısının belirlenmesidir. Bu test aslında 8 testten ve çıkarımlarından oluşan bir seridir.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,\ldots\ldots, \varepsilon_n$), n bit dizisinin uzunluğunu temsil etmektedir. Burada kullanılan referans dağılımı ki kare testidir.

a) Dizi normalize edilir. Giriş dizisindeki (ε) sıfırlar ve birler $X_i=2\ \varepsilon_i-1$ kullanılarak X_i değerleri -1 ve +1 dönüştürülür.
Örneğin: ε= 0110110101, X=(-1),1,1,(-1),1,1,(-1),1,(-1),1 olur.

b) S_i kısmi toplamları hesaplanır. Mode 0 ise X_i den mode 1 ise X_n den başlanır.

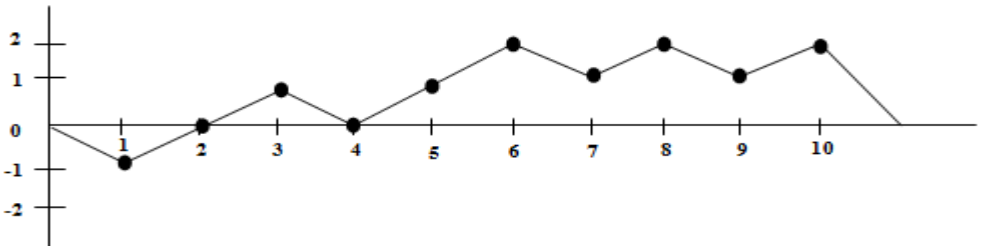
$S_1=X_1$
$S_2=X_1+ X_2$
$S_3=X_1+ X_2+ X_3$
.
.
$S_k= X_1+ X_2+ X_3+.....+X_k$
.
.
$S_n= X_1+ X_2+ X_3+.....+X_k+....+X_n$

Bu bölümdeki örnek için,

$S_1= -1$	$S_6= 2$
$S_2= 0$	$S_7= 1$
$S_3= 1$	$S_8= 2$
$S_4= 0$	$S_9= 1$
$S_5= 1$	$S_{10}= 2$

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

c) S kümesinin başına ve sonuna 0 eklenerek S' kümesi elde edilir. S'=0,s_1,s_2,....s_n,0.Bu kısımdaki örnek için: S'={0,-1,0,1,0,1,2,1,2,1,2,0}. Rasgele yürüyüş sonuçları Şekil 4.1'de gösterilmiştir.



Şekil 4.1. Rasgele yürüyüş sonuçları

d) J, S' deki sıfır geçişlerinin toplam sayısıdır, burada sıfır geçişleri başlangıçta sıfırdan sonra görülen sıfırların sayısıdır. Başlangıçtaki sıfır ile sonraki sıfır arası ilk döngüyü oluşturur. Sonra ilk döngünün sonu ikinci döngünün başlangıcı olacak ve sonraki sıfır sonu olarak diğer döngü oluşturulur. Eğer J<500 olursa test devam etmeyecektir.

Bu kısımdaki örnek için, S'={0,-1,0,1,0,1,2,1,2,1,2,0} için, j=3 (S'deki sıfırların pozisyonu 3,5 ve 12'dir). Sıfır geçişleri yukarıdaki grafikte kolaylıkla gözükmektedir. J=3 olduğunda, burada {0,-1,0}, {0,1,0} ve {0,1,2,1,2,1,2,0} içeren 3 döngü vardır.

e) Tablo 4.18'de gösterildiği gibi tüm döngüler içindeki x durum değerlerinin frekansı hesaplanır. -4≤x≤-1 ve 1≤x≤4 için. Buradaki örnek için, ilk döngü 1 tane -1 in oluşumuna sahiptir, ikinci döngüde 1 tane1'in oluşumuna sahiptir ve 3. Döngüde 3'er tane 1 ve 2'nin oluşumuna sahiptir.

Tablo 4.18. Rasgele yürüyüş döngüleri

Durum x	**Döngüler**		
	Döngü 1 {0,-1,0}	**Döngü 2 {0,1,0}**	**Döngü 3 {0,1,2,1,2,1,2,0}**
-4	0	0	0
-3	0	0	0
-2	0	0	0
-1	1	0	0
1	0	1	3
2	0	0	3
3	0	0	0
4	0	0	0

f) X'in sekiz durum değerinin her biri için, k=0,1,....,5 için, tüm döngüler arasında k kez oluşan x durumunun döngülerdeki toplam sayısı $v_k(x)$ hesaplanır.

$\sum_{k=0}^{5} v_k(x) = j$ olduğuna dikkat edin.

Bu kısımdaki örnek için:

- $v_0(-1) = 2$ (-1 durumu 2 döngüde de 0 kez meydana gelir)
  $v_1(-1) = 1$ (-1 durumu 1 döngüde de 1 kez meydana gelir)
  $v_2(-1) = v_3(-1) = v_4(-1) = v_5(-1) = 0$ (-1 durumu sıfır döngüde {2,3,4,≥5} meydana gelir.)
- $v_0(1) = 1$ (1 durumu 1 döngüde de 0 kez meydana gelir)
  $v_1(1) = 1$ (1 durumu 1 döngüde de 1 kez meydana gelir)
  $v_3(1) = 1$ (1 durumu 1 döngüde de 3 kez meydana gelir)
  $v_2(1) = v_4(1) = v_5(1) = 0$ (1 durumu sıfır döngüde {2,4,≥5} meydana gelir.)
- $v_0(2) = 2$ (2 durumu 2 döngüde de 0 kez meydana gelir)
  $v_3(2) = 1$ (2 durumu 1 döngüde de 3 kez meydana gelir)
  $v_1(2) = v_2(2) = v_4(2) = v_5(2) = 0$ (2 durumu sıfır döngüde {1,2,4,≥5} meydana gelir.)
- $v_0(-4) = 3$ (-4 durumu 2 döngüde de 0 kez meydana gelir)
  $v_1(-4) = v_2(-4) = v_3(-4) = v_4(-4) = v_5(-4) = 0$ (-1 durumu sıfır döngüde {1,2,3,4,≥5} meydana gelir.)

g) X'in sekiz durum değerinin her biri için $X^2(\text{obs}) = \sum_{k=0}^{5} \frac{(v_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$ hesaplanır. Burada $\pi_k(x)$, x durum değerinin rasgele dağılımda k kez oluşumunun olasılığıdır.

Tablo 4.19. x durum değişkeninin oluşma olasılığı

	$\pi_0(x)$	$\pi_1(x)$	$\pi_2(x)$	$\pi_3(x)$	$\pi_4(x)$	$\pi_5(x)$
X=0	0.0	0.0	0.0	0.0	0.0	0.0
X=1	0.5000	0.2500	0.1250	0.0625	0.0312	0.0312
X=2	0.7500	0.0625	0.0469	0.0352	0.0264	0.0791
X=3	0.8333	0.0278	0.0231	0.0193	0.0161	0.0804
X=4	0.8750	0.0156	0.0137	0.0120	0.0105	0.0733
X=5	0.9000	0.0100	0.0090	0.0081	0.0073	0.0656
X=6	0.9167	0.0069	0.0064	0.0058	0.0053	0.0588
X=7	0.9286	0.0051	0.0047	0.0044	0.0041	0.0531

h) Her bir x değeri için, pdeğeri=igamc(5/2 , X^2(obs)/2) hesaplanır. 8 tane p değeri üretilir.

4.4.16. Rasgele Gezinimler Değişken Testi

Bu testte, bir kümülatif toplam rasgele yürüyüşünde, özel durumların ziyaret edilme sayısı ölçülür. Testin amacı, bir rasgele yürüyüşte özel durumların beklenen ziyaret sayısındaki sapmalarının gözlenmesidir. Bu test aslında 18 durumun (ve sonuçlarının) serisidir. Durumlar -9,-8,.....,-1 ve 1,2,.....9'dur.

Hesaplamalarda kullanılan ε, RSÜ veya SRSÜ ile üretilen bit dizisi ($\varepsilon= \varepsilon_1,\varepsilon_2,.....,\varepsilon_n$), n bit dizisinin uzunluğunu ve ξ 4.4.15 nolu testte adım4'te karar verilen bütün rasgele yürüyüşler süresince ziyaret edilen durumların toplam sayısını temsil etmektedir.

a) Dizi normalize edilir. Giriş dizisindeki (ε) sıfırlar ve birler $X_i=2\varepsilon_i-1$ kullanılarak X_i değerleri -1 ve +1 dönüştürülür.
   Örneğin: ε= 0110110101, X=(-1),1,1,(-1),1,1,(-1),1,(-1),1 olur.
b) Si kısmi toplamları hesaplanır. Mode 0 ise X_i 'den mode 1 ise X_n 'den başlanır.

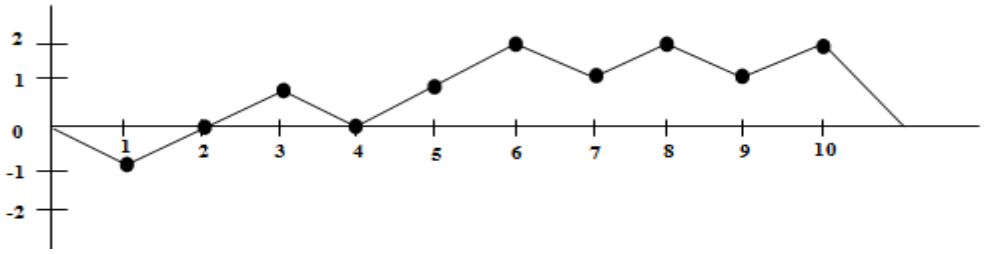
$S_1=X_1$
$S_2=X_1+ X_2$
$S_3=X_1+ X_2+ X_3$
.
.
$S_k= X_1+ X_2+ X_3+.....+X_k$
.
.
$S_n= X_1+ X_2+ X_3+.....+X_k+....+X_n$

Bu bölümdeki örnek için,

$S_1= -1$ $S_6= 2$
$S_2= 0$ $S_7= 1$
$S_3= 1$ $S_8= 2$
$S_4= 0$ $S_9= 1$
$S_5= 1$ $S_{10}= 2$

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

c) S kümesinin başına ve sonuna 0 eklenerek S' kümesi elde edilir. $S'=0,s_1,s_2,....s_n,0$.
   Bu kısımdaki örnek için: S'={0,-1,0,1,0,1,2,1,2,1,2,0}. Rasgele yürüyüş sonuçları aşağıda gösterilmiştir.



Şekil 4.2. Rasgele gezinim değişken yürüyüş sonuçları

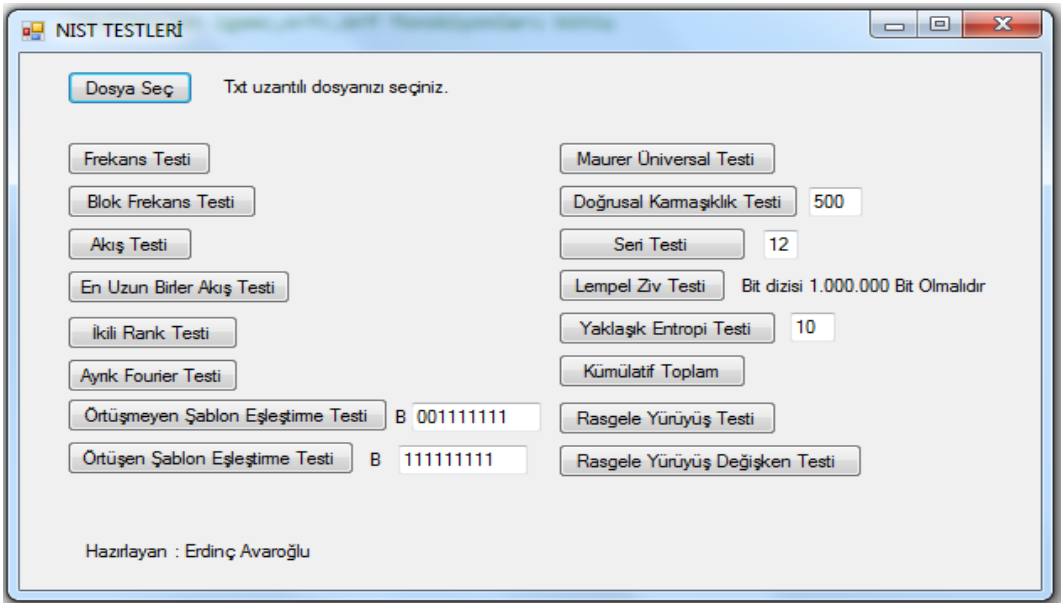
d) 18 farklı x durumu için, bütün j döngüleri içinde oluşan x durum değerlerini toplam sayısı ξ(x) hesaplanır.

Bu kısımdaki örnek için: ξ(-1)=1, ξ(1)=4, ξ(2)=3 ve diğerleri ξ(x)=0 olur.

e) Her bir ξ(x) için, pdegeri = $\text{erfc}(\frac{|\xi(x) - J|}{\sqrt{2J(4|x|-2)}})$ hesaplanır.

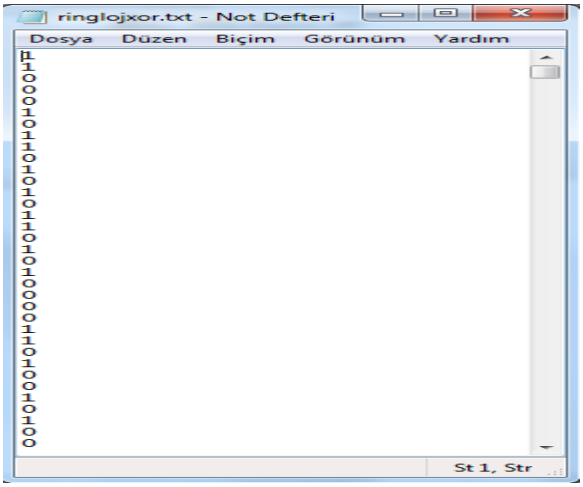
4.5. NIST Yazılımı

Elde ettiğimiz sayı dizilerini test etmek amacıyla NIST istatistikî testlerinin programı C# programlama dilinde yazılmış ve Şekil 4.3'te gösterilmiştir. Yazılmış olan programın çalışma düzeni, ekran görüntüsü ve yapılan test ile ilgili örnek sonuç çıktıları aşağıda verilmiştir.

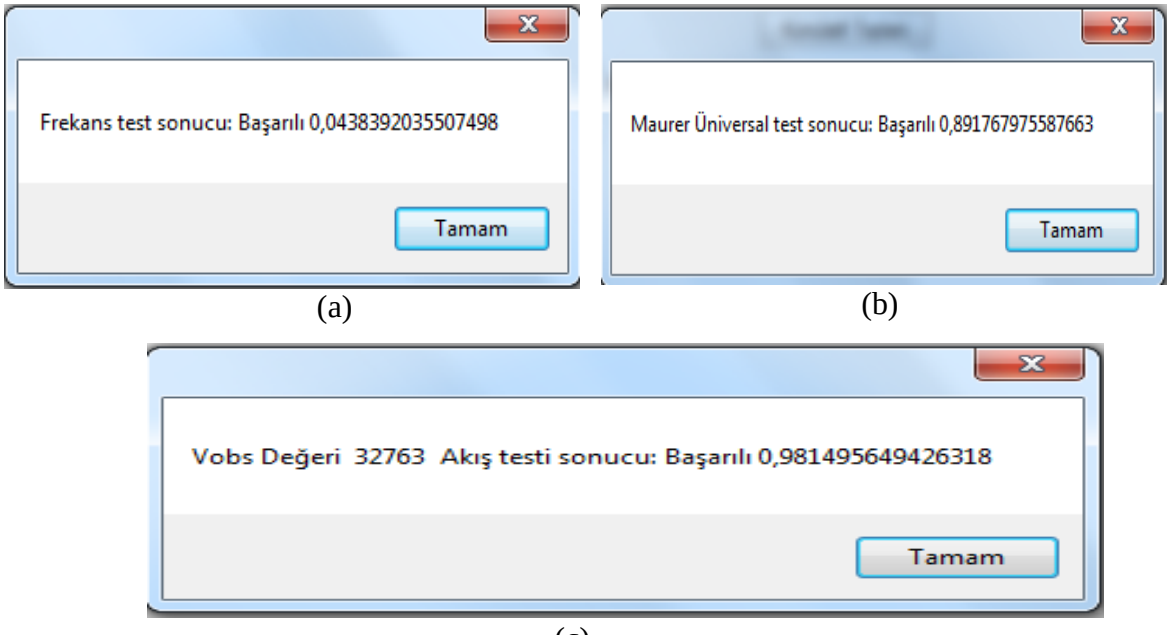


Şekil 4.3. NIST Programının Ekran Çıktısı

Elde edilen rasgele bit dizisi Şekil 4.4'te gösterildiği gibi text dosyasının içerisine her bir satırda 1 karakter olmak üzere (tek sütunlu bir matris formatında) veriler girilerek kayıt edilmelidir. Yazılım üzerinden dosya seç menüsü kullanılarak test edilmek istenilen dosya seçimi yapılmalıdır. Bit dizisi uzunluğu otomatik olarak alınmaktadır. Dosya seçimi yapıldıktan sonra diğer menüler kullanılarak test işlemi yapılmaktadır. Bazı test sonuçları Şekil 4.5'te gösterilmiştir.



Şekil 4.4. Seçilen Dosya İçeriği



(a) (b)

(c)

Şekil 4.5. NIST Programı Test Çıktısı Örneği (a) Frekans Testi Sonucu (b) Maurer Üniversal Test Sonucu (c) Akış Test Sonucu

5. GRSÜ TASARIMLARI İÇİN SON İŞLEM ÖNERİSİ VE SAF SRSÜ İÇİN HİBRİT SRSÜ ÖNERİSİ

Bu bölümde yapmış olduğumuz iki farklı çalışma açıklanmıştır. İlk olarak gerçekleştirilen çalışmada, GRSÜ tasarımları için yeni bir son işlem önerilmiştir. Bu amaçla kullanılan tasarımda entropi kaynağı olarak halka osilatör kullanılmıştır. Önerilen son işlem ise lojistik haritaya dayalıdır.

Diğer çalışmada ise, Saf SRSÜ sistemleri, rasgele sayı üreteçlerinin sağlaması gereken dört gereksinimden ilk üçünü sağlamaktadır. R4 gereksinimini sağlayamadığı için saf SRSÜ sistemlerinin kriptografik uygulamalarda kullanılması uygun değildir. R4 gereksiniminin sağlanması için, saf SRSÜ'deki geçerli iç durum değerinin gerçek rasgele veri ile güncellenemesi gerekmektedir. Gerçek rasgele veri, Saf SRSÜ'deki çıkış ve giriş fonksiyonuna ek girdi olarak eklenecektir ve böylece Hibrit SRSÜ sistemi elde edilecektir. Yapılan çalışmada Saf SRSÜ olarak AES, ek girdi olarak da Xilinx FPGA üzerinde gerçekleştirilen Burke Shaw kaotik çeker kullanılmıştır.

5.1. GRSÜ İçin Kaotik Tabanlı Yeni Son işlem Önerisi

GRSÜ ve SRSÜ sayı üreteçleri tarafından üretilen sayılar bilgi güvenliği sistemlerinde anahtar olarak kullanılabilmektedir. Ancak anahtarların sistem dışında kontrolsüz ortamlarda üretimi sistemin güvenirliliğini azaltmaktadır. Bu sebeple anahtarlar eğer bir entegre, içerisinde gerçekleştirilirse sistem daha güvenli olmaktadır [8]. Bundan dolayı anahtarların FPGA gibi programlanabilir donanımlarda üretilmesi gittikçe popüler olmaktadır. Rasgele sayıların donanımsal geliştirilmesi durumunda güç tüketimi ve kullanılan mantık elemanlarının sayısı önemli kriterlerdir. GRSÜ tabanlı rasgele sayı üretiminde sayıların rasgelelik kalitesi kullanılan fiziksel gürültüye bağlıdır. Fiziksel gürültü olarak literatürde bilinen radyoaktif bozulma, termal gürültü veya serbest salınımlı osilatör kullanılmaktadır. Şekil 2.5'de gösterildiği gibi GRSÜ fiziksel gürültü kaynağı, sayısallaştırma ve son işlem biriminden oluşmaktadır.

Son iki birim sayısallaştırma ve son işlem genellikle sayısallaştırılmış analog sinyal (Digizited Analog Sinyal (DAS)) ve dâhili rasgele sayılar olarak adlandırılır. Son işlem birimi genellikle GRSÜ'lerde gerekli olup DAS'ın eksikliğini (kusurunu) gidermek için

kullanılmaktadır. Bir diğer sistemi yan kanal analizi saldırıları ve çevresel değişikliklere karşı dirençli hale getirmesidir. Son işlem algoritmasına bağlı olarak üretecin güvenliği artar. Ancak son işlem bit dizisinde ki zayıflıkları giderirken üretilen sayıların çıkış oranını azaltır. Bu nedenle GRSÜ'nün çıkış hızını düşürmeden, istatistikî zayıflıkları gideren yeni bir son işlem algoritması önerilmiştir. Bu uygulamada entropi kaynağı olarak halka osilatörler (Ring Oscillator-RO) tarafından üretilen faz seğirmesi kullanılarak GRSÜ, FPGA ortamında donanımsal olarak gerçekleştirilmiştir. Seğirme, elektronik veya termal gürültüden dolayı teorik olarak doğru konumdan zamansal sapma olarak tanımlanır. Aslında seğirme birçok sistemde istenmeyen bir özelliktir ancak seğirmenin rasgele olması rasgele sayı üretmek için GRSÜ gibi sistemlerde kullanılmasını ideal kılmaktadır. Ancak halka osilator tarafından üretilen seğirme kaliteli değildir [69]. Bu eksikliği gidermek için GRSÜ sistemlerinde son işlem kullanılır. Önerilen son işlem kaotik davranış sergileyen lojistik haritadır. Lojistik harita 1. dereceden bir denklem olup bir başlangıç koşulu ve bir kontrol parametresine gerek duymaktadır. Bu sebepten dolayı donanım üzerinde gerçekleştirilmesi kolaydır. Gerçekleştirilen RO tabanlı bir GRSÜ sisteminde lojistik haritanın üretilen sayılara etkisini gözlemleyebilmek için dört farklı senaryo geliştirilmiştir. GRSÜ sisteminde kullanılan RO sayıları ve RO'lardaki tümleyen sayıları sırasıyla (114,13) (25,3) (10,3) (5,3) seçilmiştir. Tüm senaryolarda son işlem olarak lojistik harita kullanılmış olup her bir sistem Altera'nın EP4CE115F29C7 tabanlı FPGA bordunda (Ek1) gerçekleştirilmiştir. Her bir sistem tarafından elde edilen sayıların istatistiksel testleri NIST 800.22 test suitine göre elde edilmiştir. Test sonuçlarına göre lojistik haritanın son işlem olarak kullanılabileceği gösterilmiştir.

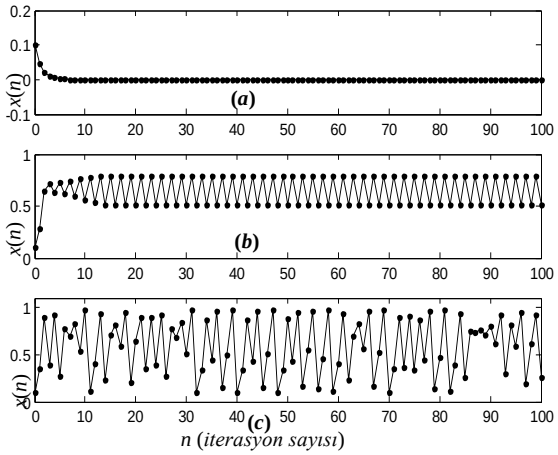
5.1.1. Önerilen Son İşlem

Kaotik davranış gösteren en basit yapı lojistik harita aşağıda verilen Denklem (5.1) ile tanımlanmaktadır.

$$x(n+1) = rx(n)(1-x(n)) \tag{5.1}$$

Burada, $0 \leq x(n) \leq 1$ olmalıdır. Ayrıca r sistem parametresi ve n iterasyon sayısıdır. x(n), bir tohum (başlangıç değeri) x(0) verilerek hesaplanır. r parametresine bağlı olarak sistem farklı davranışlar sergileyebilmektedir.

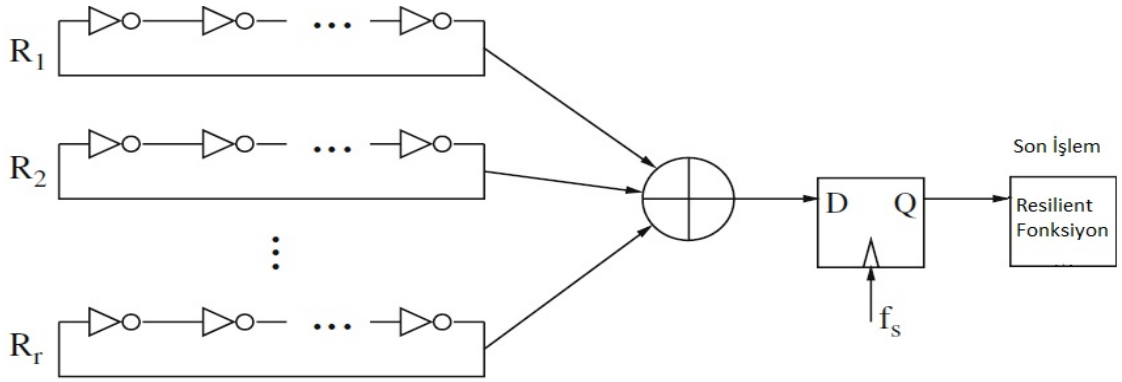
Şekil 5.1'de r=0.5, r=3.2 ve r=3.9 değerleri için iterasyon sayısına göre hesaplanan x(n) değerinin değişimleri verilmiştir. Şekil 5.1'e bakıldığında, r=0.5 için sistem kararlı bir davranış sergilerken, r=3.2 için sistemin çıkışı osilasyonludur. r=3.9 seçildiğinde ise sistemde kaotik davranış oluşmaktadır.



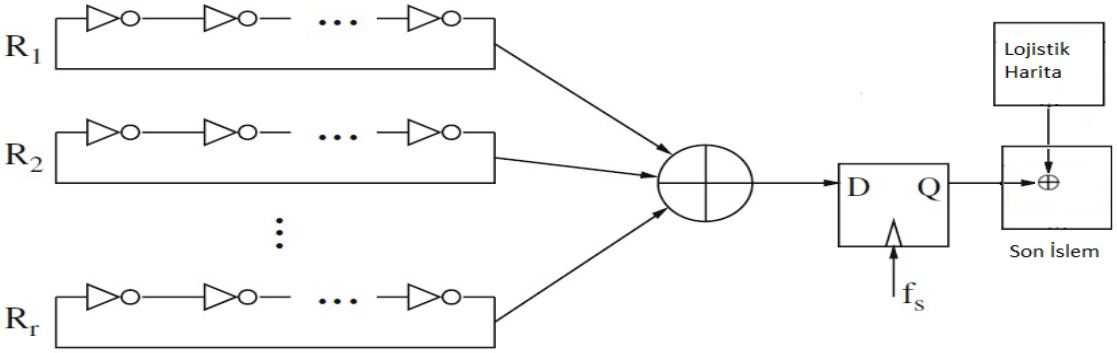
Şekil 5.1. a) r=0.5 b) 3.2 c) 3.9

Lojistik haritada, başlangıç şartı değiştirilerek farklı sayı dizileri üretilebilmektedir [14]. GRSÜ tarafından üretilen sayılara lojistik harita son işlem olarak kullanıldığında sayılar lojistik haritanın yapısına bağlı olarak değişim gösterecektir. Bu değişim diğer son işlem yöntemlerine göre daha etkilidir. Çünkü lojistik harita, XOR veya Resilent fonksiyonları gibi lineer yapıya sahip değildir. Bu avantajı sayesinde lojistik harita son işlem olarak kullanılabilir. Lojistik haritanın kullanıldığı sistemlerde başlangıç şartına bağlı olarak farklı sayılar üretmesi sistemi saldırılara karşı güçlü yapmaktadır. Bir başka ifade ile son işlem sistemi daha karmaşık bir yapıya sahip olacaktır.

Burada lojistik haritanın son işlem olarak kullanımını gösterebilmek için Sunar tarafından önerilen RO tabanlı GRSÜ sistemi kullanılmıştır. Sunar modeli Şekil 5.2.a'da, önerdiğimiz yapı ise Şekil 5.2.b'de gösterilmiştir.



(a)



(b)

Şekil 5.2. a) Sunar GRSÜ Modeli [41]. b) Lojistik haritaya dayalı önerilen son işlem modeli

Şekil 5.2.a'da gösterilen sistem 114 RO ve her bir RO içinde 13 tümleyenden meydana gelmektedir. RO tarafından elde edilen rasgele sinyaller XOR işleminde birleştirilmiştir. XOR işleminden sonra elde edilen sinyaller D flip flop kullanılarak örneklenmiştir. Sunar bu tasarımında bulunan istatistikî zayıflıkları gidermek amacıyla son işlem kısmında, resillent fonksiyonu kullanmıştır [41].

Şekil 5.2.b'de önerilen sistemin sunar modelinden farkı resilient fonksiyon son işlemi yerine lojistik haritaya dayalı son işlemin kullanımıdır. Lojistik haritaya dayalı son işlemin sisteme etkisini göstermek için önerilen sistem dört farklı senaryo ile FPGA ortamında gerçekleştirilmiştir. Her bir sistem tarafından üretilen sayıların istatistiksel testleri NIST 800.22 test suitine göre yapılmıştır.

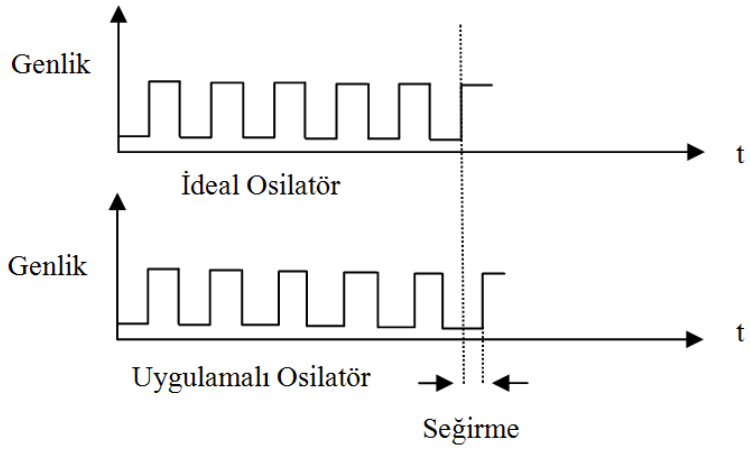
5.1.2. Gerçekleştirilen Uygulama

Önerilen sistem FPGA ortamında gerçekleştirilmiştir. Sistemde kullanılan RO ve Lojistik harita tasarımları bu kısımda ayrıntılı olarak açıklanmıştır.

Gerçekleştirilen uygulamada entropi kaynağı olarak kullanılan RO tarafından üretilen rasgele sayıların kalitesini aşağıdaki üç parametre belirler. Bunlar;

- Sistemdeki osilatörlerin sayısı
- Örnekleme frekansı
- Osilatörlerdeki tümleyenlerin sayısıdır.

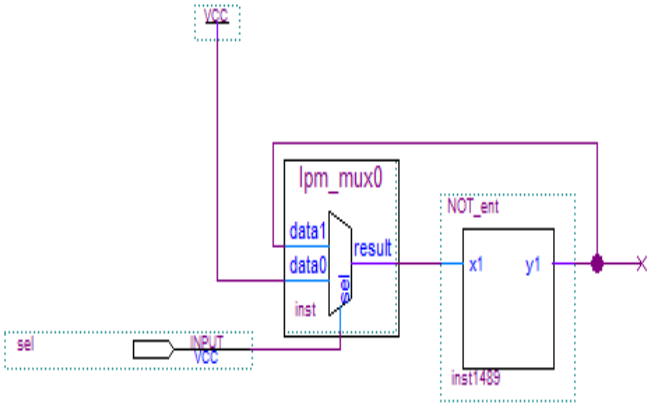
Gürültü kaynağından elde edilen saat sinyallerin doğru pozisyonlarından sapmaları değişkenlik gösterir. Bu değişkenlik seğirme olarak adlandırılır. Genellikle seğirme bir sistemde istenmeyen bir özelliktir. Seğirmenin rasgele olmasından dolayı TRNG sistemlerinde kullanılması idealdir. Şekil 5.3'de seğirme oluşumu gösterilmiştir.



Şekil 5.3. Seğirme oluşumu

Denklem (5.2)'de kullanılan n tümleyen sayısını t_{inv} ise kullanılan tümleyen elemanlarının (propagation delay) yayılma gecikmesidir. Her bir RO yayılma gecikmeleri farklılık göstermektedir. RO'lar tarafından elde edilen rasgele işaretlerin XOR devresine verilmesi ile rasgele işaretler üretilebilmektedir. Şekil 5.4'de FPGA ortamında tasarlanan RO yapısı verilmiştir. Sistemde kullanılan her bir RO VHDL dili ile data flow ve şematik tasarım yöntemleri kullanılarak gerçekleştirilmiştir.

$$F_s = 1/2 * n * t_{inv} \tag{5.2}$$



Şekil 5.4. Halka osilatör

Şekil 5.4'de tümleyen çıkışı sürekli olarak lojik 0 seviyesindedir. RO tarafından rasgele işaretin elde edilmesi için fiziki ortamdan bir uyartım işaretinin (sel=1) verilmesi ile RO çıkışı elde edilmektedir. Elde edilen rasgele değişimin örneklenmesi ile rasgele sayı üretimi tamamlanır, ancak entropi kaynağı olarak RO kullanımı ile kaliteli sayılar üretilemez. Bu dezavantajın ortadan kaldırılması için son işlem kullanılır. Son işlem için önerilen lojistik haritanın FPGA ortamında gerçekleştirilmesi Şekil 5.5'de verilmiştir.

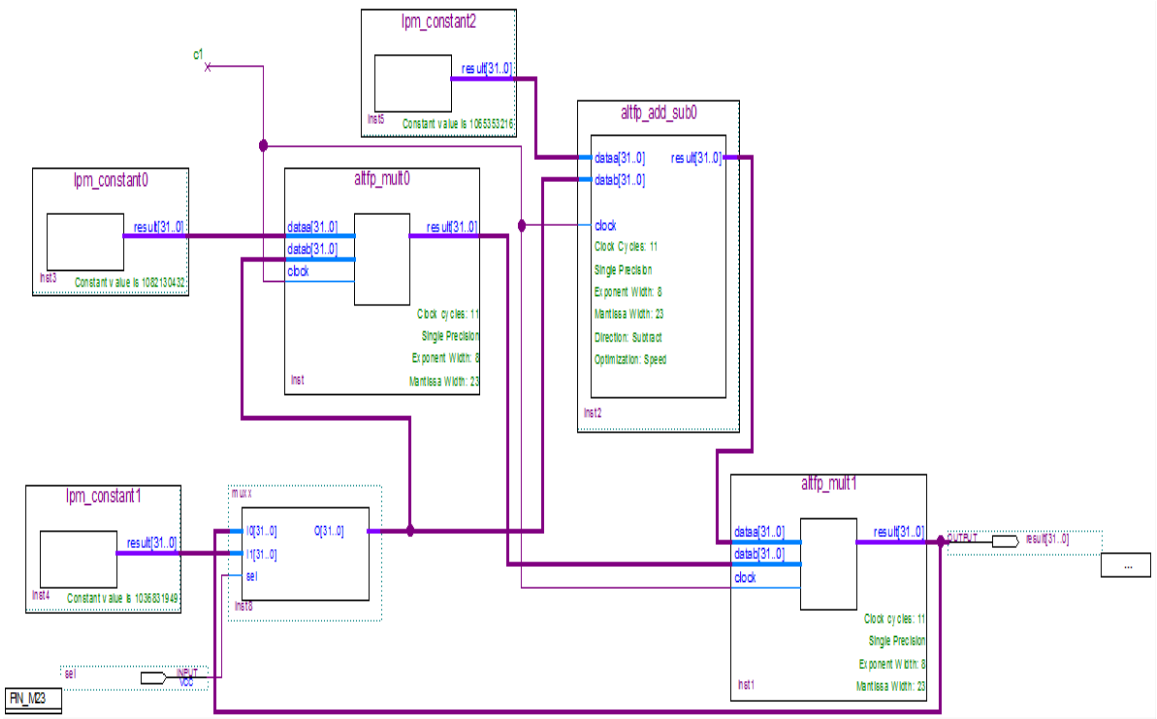
Lojistik haritanın gerçekleştirilmesi için Altera tarafından geliştirilen hazır modüller kullanılmıştır [70]. Bunlar, floating point sayılar üzerinde işlemler yapabilen Lmp_constant, Altfp_mult ve Altfp_add_sub modülleridir. Bu modüller sırasıyla sabit floating point sayı gösterimi, çarpma ve toplama işlemlerini yerine getirebilmektedir. Ayrıca, sistemde kullanılan mux VHDL (VHSIC hardware description language) dilinde data flow mimarisi kullanılarak tasarlanmıştır. Sistemin sürekli zamanda rasgele sayı üretebilmesi için çıkıştan üretilen değerin, x_n 'in bir sonraki iterasyonda sistemin girişini sağlayabilmesi için mux kullanılmıştır. İlk olarak sistemin başlangıç değeri, diğer bir ifade ile tohum değeri yüklenmiştir. Örnek olarak Lmp_constant1 modülünün değeri 0.1 ya da floating point gösterimde $(3DCCCCCD)_H$ ile yüklenmiştir. Şekil 5.6 sistemin 0.1 tohum değerine karşılık çıkış değerini, Şekil 5.7 ise devam eden iterasyonlar için lojistik harita tarafından üretilen sayıların değerini göstermektedir. Elde edilen sayılar basit bir karşılaştırma devresine verilerek üretilen sayı 0.5'ten büyük ise 1 değil ise 0 olarak elde edilmiştir. Sistemde kullanılan Altfp_mult0, Altfp_mult1 ve Altfp_add_sub modüllerinin her biri 11 saat darbesi süresinde çıkış vermektedirler. Altfp_mult0 ve Altfp_add_sub modülleri eş zamanlı olarak çalıştıklarından dolayı her bir rasgele sayı 22 saat darbesinde

bir üretilmektedir. Sistemde kullanılan saat frekansı 450 Mhz olduğundan dolayı her bir sayı 0.048888 µsn'de üretilmektedir.

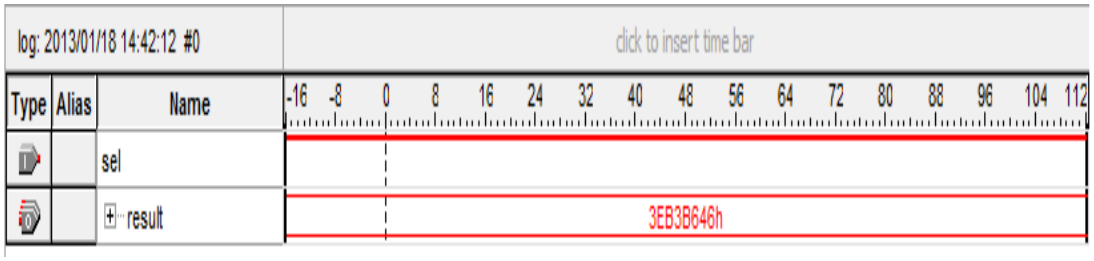
Sistem tarafından üretilen sayılar 32 bittir. IEEE 754 floating point formatı günümüzde gerçek sayıları temsil etmek üzere kullanılan en yaygın gösterim şeklidir. Bu formatta floating point sayılar üç bölüm ile gösterilir. Bu bölümler işaret, üs ve ondalık olarak ifade edilir. İşaret (S) biti pozitif sayılar için 0, negatif sayılar için 1 değerini alır. Üs(E) alanı hem negatif hem de pozitif üsleri temsil edebilmektedir. Bunu gerçekleştirmek için bir bias değeri gerçek üs değeriyle toplanıp üs kısmı oluşturulur. M, ondalık sayıyı ifade eden bitleri gösterir. Bu, sayının tam ve kesir kısımlarını gösteren bitlerden oluşur. Floating point sayı formatı Denklem (5.3)'deki gibi gösterilir.

$$\text{Sayı Değeri}=(-1)^{S}x2^{E-127}x(1.M) \qquad (5.3)$$

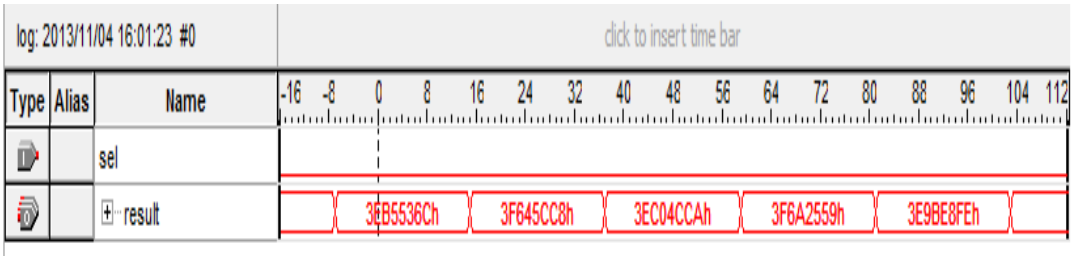
16'lık sayı sistemine göre sistem tarafından üretilen, ilk değer $(3EB3B646)_H$'e karşılık floating point karşılığı 00111110101100111011011001000110=0.351 olacaktır. Elde edilen bu değer 0.5'ten küçük olduğundan dolayı lojistik haritanın ürettiği ilk değer 0 olacaktır.



Şekil 5.5. Lojistik haritanın FPGA'da gerçekleştirilmesi



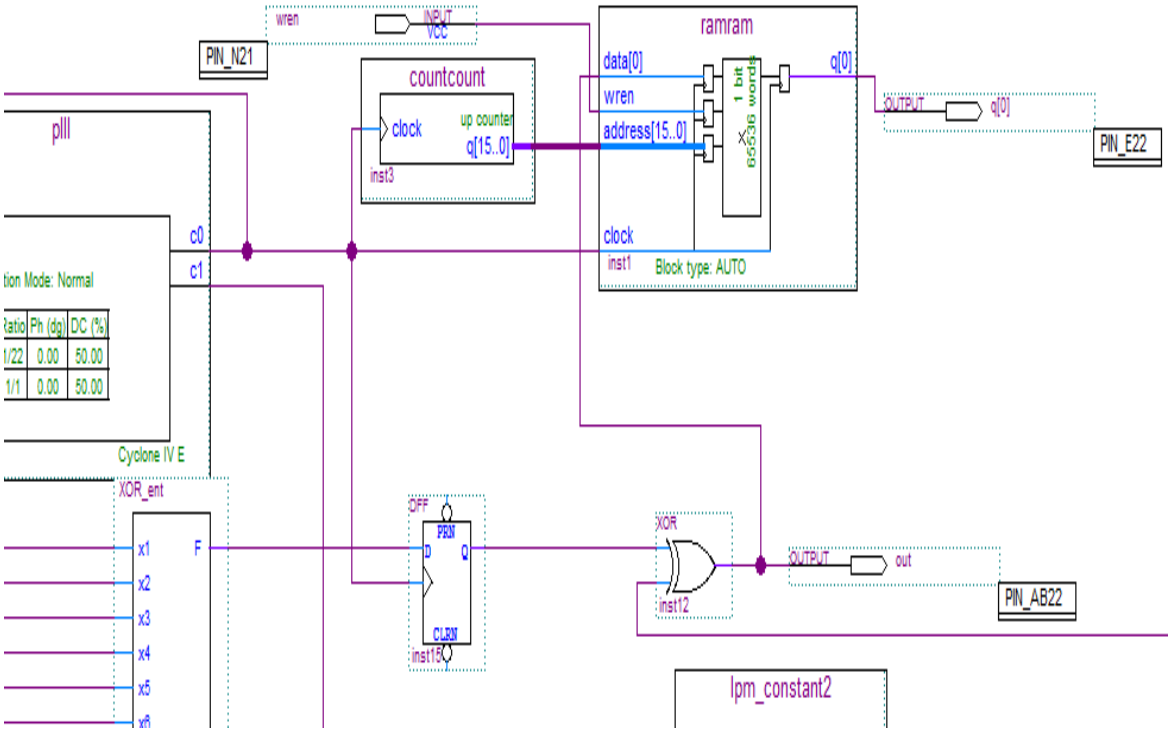
Şekil 5.6. Sistemin başlangıç durumu



Şekil 5.7. Üretilen rasgele sayılar

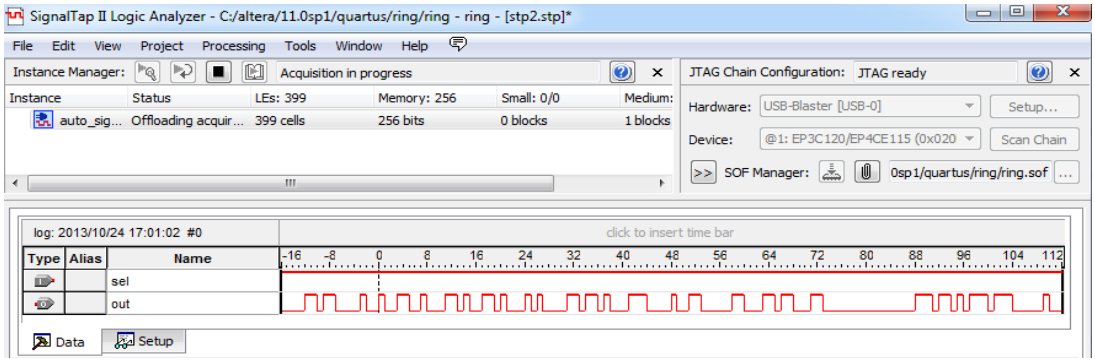
Önerilen GRSÜ sistemi tarafından üretilen sayılar ve lojistik harita tarafından üretilen sayılar (lojik 1 ve lojik 0) iki kapılı XOR ünitesine giriş verilmektedir. 450 MHz frekansa sahip lojistik harita 22 saat darbesinde bir sayı üretmektedir. Bir başka ifade ile sistemden her 0.04888 µsn' de bir gerçek sayı üretilmelidir. Bundan dolayı GRSÜ sistemi tarafından üretilen sayılarında frekansı 20.45Mhz olmak zorundadır. Sistemde kullanılan 450 ve 20.45 Mhz frekansları elde etmek için PLL kullanılmıştır.

XOR ünitesi çıkışından elde edilen sayıların istatistiksel testlerinin yapılması için sayılar bir hafıza elemanında saklanmıştır. Bu amaç için gerçekleştirilen donanım Şekil 5.8'deki gibidir. Hafıza adreslerine kaydedilecek sayıların adresleri sistemde kullanılan counter yardımıyla belirlenmiştir.



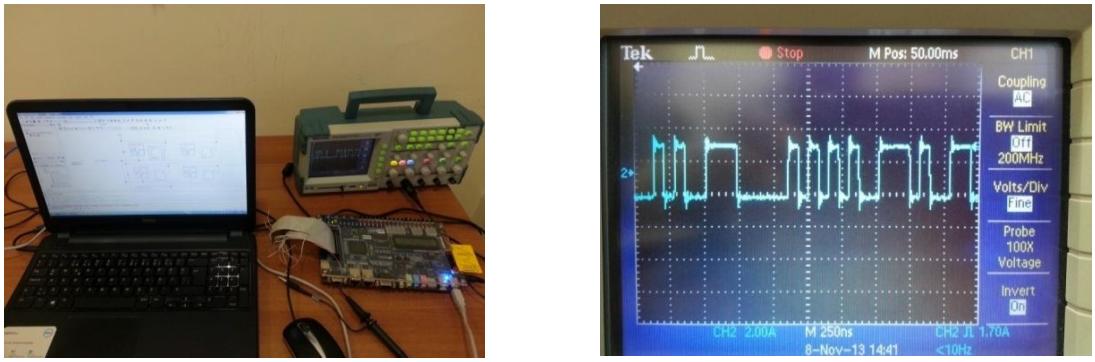
Şekil 5.8. Örnekleme ünitesi ve üretilen sayıların hafıza birimine depolanması için geliştirilen donanımsal yapı

Sistemden elde edilen simulasyon sonuçlarına göre rasgele sayıların değişimini Şekil 5.9'da gösterilmektedir.



Şekil 5.9. Gerçekleştilen sistemden elde edilen rasgele sayı

Bu sistemin gerçek zamanda çalıştırılması ve testlerinin gerçekleştirilmesi için set Şekil 5.10.a'da verilmiştir. Elde edilen bit değişimleri ise Şekil 5.10.b'de verilmiştir.



(a) (b)

Şekil 5.10. (a) Deney seti görünümü (b) Gerçek zamanda üretilen bitler.

5.1.3. Sonuçlar

GRSÜ üretim yöntemlerinden olan RO tarafından üretilen seğirme kullanılarak FPGA ortamında donanımsal olarak gerçekleştirilmiştir. Bu sistem Sunar ve diğerleri tarafından önerilmiş bir sistemdir. Sunar tarafından üretilen sistemde entropi kaynağından üretilen rasgele bit dizilerinde korelasyon olması nedeniyle son işlemsiz olarak testleri geçememiştir. Bunun sonucunda resilient fonksiyonu olarak adlandırılan son işleme tabi tutulmuş olup çıkış hızı ve oranı 1/16 oranında azalmıştır. Bu amaçla sistemin entropi kaynağında bulunan olası eksikliği gidermek için lojistik harita son işlem olarak önerilmiştir. Lojistik haritanın rasgeleliğinin tahmin edilememe ve rasgelelilik özellikleri katması nedeniyle GRSÜ sisteminin güvenliği artacaktır. Sunar sisteminde çıkış hızı 2.5 Mbit/s iken geliştiren sistemde yaklaşık 20 Mbit/s olmuştur.

Geliştirilen sistemin kriptografik uygulamalarda güvenli olarak kullanılabilmesi için rasgelelik testlerine tabi tutulması gerekmektedir. Üretilen sayıların rasgeleliğini analiz etmek için birçok test suiti geliştirilmiştir. Bu testlerden biri de istatistiksel tabanlı NIST 800.22 test suitidir. Sistem tarafından üretilen sayıların NIST 800.22 test suitine göre istatistiksel testleri geliştirilen yazılım ile elde edilmiştir. NIST 800.22 test suitine göre istatistiki test sonuçları aşağıdaki Tablo 5.1 ve Tablo 5.2'de verilmiştir. Tablo 5.1, Şekil 5.2.a ve Şekil 5.2.b'de verilen sistem sonuçlarını göstermektedir. Bu testlerde RO'sayıları 114 ve tümleyen sayıları 13'tür. Tablo 5.1'e göre önerdiğimiz sistem tüm NIST testlerinden başarılı olmuştur. Bu başarıdan dolayı önerdiğimiz GRSÜ sistemi basitleştirilmiş, RO sayıları ve her bir RO'daki tümleyen sayı çiftleri sırasıyla (25,3), (10,3) ve (5,3) seçilmiştir. Tablo 5.2'de bu tasarımlar için NIST 800.22 test suitine göre elde edilen sonuçları göstermektedir.

Tüm tasarımlarda başarılı sonuçlar elde edilmiştir. Sonuçlar göstermiştir ki, lojistik harita son işlem olarak kullanılabilir. Çünkü istatistiksel olarak kaliteli sayılar üretilebilmiştir.

Tablo 5.1. Son işlemsiz Sunar sistemi ile geliştiren sistemin istatistiki test sonuçları

	Son İşlemsiz Sunar GRSÜ tasarımı		**Önerilen Son İşlemli Sunar GRSÜ tasarımı**	
Test	**P- Değeri**	**Sonuç**	**P-Değeri**	**Sonuç**
Frekans Testi	-	Başarısız	0.043	Başarılı
Blok Frekans Testi	-	Başarısız	0.326	Başarılı
Akış Testi	-	Başarısız	0.981	Başarılı
Blok İçindeki En Uzun Birler Akışının Testi	-	Başarısız	0.128	Başarılı
İkili Matris Rankı Testi	0.424	Başarılı	0.785	Başarılı
Ayrık Fourier Dönüşümü Testi	-	Başarısız	0.089	Başarılı
Örtüşmeyen Şablon Eşleştirme Testi	-	Başarısız	0.754	Başarılı
Örtüşen Şablon Eşleştirme Testi	-	Başarısız	0.473	Başarılı
Maurer'in Evrensel İstatistik Testi	-	Başarısız	0.891	Başarılı
Doğrusal Karmaşıklık Testi	0.615	Başarılı	0.073	Başarılı
Seri Testi	-	Başarısız	0.908	Başarılı
	0.879	Başarılı	0.846	Başarılı
Yaklaşık Entropi Testi	-	Başarısız	0.810	Başarılı
Kümülatif Toplam Testi	-	Başarısız	0.066	Başarılı

Tablo 5.2. (25,3), (10,3) ve (5,3) RO ve tümleyenlerin sayıları için önerilen sistemin lojistik harita son işleme tabi tutulması ile elde edilen istatistiki test sonuçları

Test	**25 RO 3 Inverter P- Değeri**	**10 RO 3 Inverter P- Değeri**	**5 RO 3 Inverter P- Değeri**	**Sonuç**
Frekans Testi	0.814	0.360	0.222	Başarılı
Blok Frekans Testi	0.329	0.901	0.673	Başarılı
Akış Testi	0.950	0.647	0.099	Başarılı
Blok İçindeki En Uzun Birler Akışının Testi	0.702	0.920	0.879	Başarılı
İkili Matris Rankı Testi	0.210	0.628	0.171	Başarılı
Ayrık Fourier Dönüşümü Testi	0.504	0.757	0.478	Başarılı
Örtüşmeyen Şablon Eşleştirme Testi	0.815	0.087	0.670	Başarılı
Örtüşen Şablon Eşleştirme Testi	0.241	0.186	0.375	Başarılı
Maurer'in Evrensel İstatistik Testi	0.088	0.526	0.721	Başarılı
Doğrusal Karmaşıklık Testi	0.745	0.859	0.150	Başarılı
Seri Testi	0.353 0.024	0.650 0.387	0.625 0.496	Başarılı Başarılı
Yaklaşık Entropi Testi	0.921	0.682	0.622	Başarılı
Kümülatif Toplam Testi	0.934	0.622	0.236	Başarılı

5.2. Hibrit SRSÜ İçin Yeni Bir Metod ve İstatiski Analizi

Güçlü kriptografik sistemler kaliteli rasgele sayılara gereksinim duyar. Bu amaçla kullanılan rasgele sayı üreteçlerinden biri olan Saf SRSÜ ile üretilen sayılar, tohum değeri tespit edildiğinde, sistemde kullanılan fonksiyonlar yeterince karmaşık olmadığı taktirde tahmin edilebilmiştir. Belirtilen bu eksiklikler nedeniyle saf SRSÜ'ler kriptografik uygulamalar için uygun görülmemiştir. Ancak bu eksikliklerin giderilmesi amacıyla sisteme ek girdi eklenmiştir ve bu ek girdinin tahmin edilememe özelliğini katması amacıyla GRSÜ tasarımı olması gerekmektedir.

Rasgele sayı üretmek amacıyla kullanılan rasgele sayı üreteçlerinin belirli güvenlik gereksinimlerini sağlamaları gerekmektedir [5]. Bunlar;

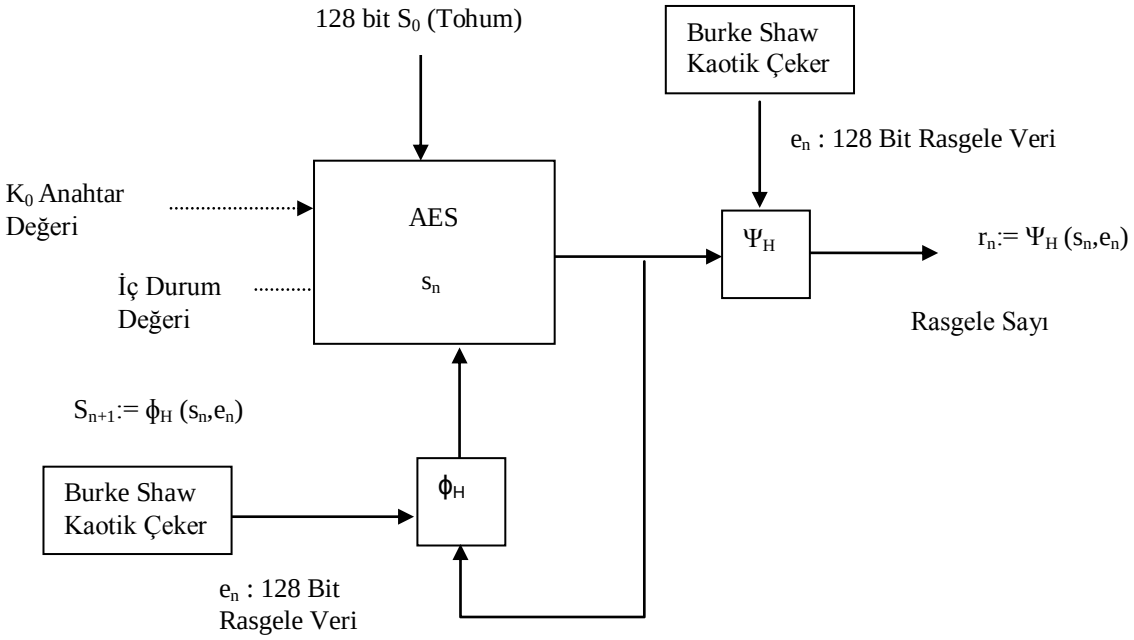
- R1: Rasgele sayılar herhangi bir istatistiksel zayıflık göstermemelidir.
- R2: Rasgele sayıların alt dizilerini (bir kısmını) bilmek (saldırgan için) öncül (predecessor) ve ardıl (successor) rasgele sayıların hesaplanmasına veya tahminine izi vermemelidir.
- R3: İç durum değerinin bilinmesi durumunda veya iç durum değeri bilinmese bile tahmin edilebilme olasılığı ile ardıl rasgele sayıları hesaplayabilmek mümkün olmamalıdır.
- R4: İç durum değerinin bilinmesi durumunda veya iç durum değeri bilinmese bile tahmin edilebilme olasılığı ile gelecek rasgele sayıları hesaplayabilmek mümkün olmamalıdır.

Güçlü kriptografik sistemler için bu dört gereksinim sağlanmalıdır. Ancak, Saf SRSÜ'ler R4 gereksinimini sağlamamaktadır. Bu durum ancak Hibrit SRSÜ'de iç durum değerinin rasgele bir veri ile güncellendikten sonra üretilen rasgele sayılar ile R4 gereksinimi sağlanmaktadır.

Bu amaçla çalışmada, R1-R3 güvenlik gereksinimlerini sağlayan saf SRSÜ olarak AES (advanced enctyption system) kullanılmıştır. İç durum değerinin sürekli olarak rasgele bir veri ile güncellenmesi için sisteme ek girdi olarak kaos tabanlı GRSÜ ünitesi (Burke Shaw kaotik çeker) eklenmiştir. Bu sayede R4 gereksimide sağlanarak Hibrit SRSÜ sistemi geliştirilmiştir. Hibrit SRSÜ sisteminden üretilen rasgele bit dizilerinin rasgeleliliği NIST istatistikî testine tabi tutularak elde edilen sonuçlar verilmiştir.

5.2.1. Geliştirilen Hibrit SRSÜ sistemi

Şekil 5.11'de geliştirilen hibrit SRSÜ sisteminin şekli gösterilmiştir. Bu amaçla saf SRSÜ AES kullanılarak oluşturulmuştur. Sisteme ek girdi olarak Xilinx FPGA üzerinde gerçekleştirilen Burke Shaw kaotik çeker kullanılarak elde edilen rasgele bit dizisi kullanılmıştır. Sistemde her bir adımda rasgele üretilen 128 bit tohum değeri ve anahtar değeri AES'e girilmiştir. Şifreleme işlemi sonucunda elde edilen 128 bit şifreli tohum değeri Burke Shaw kaotik çekerden elde edilen 128 bit rasgele veri ile XOR'lanmıştır ve çıkışa verilmiştir. Yeni iç durum değeri değeri olarak, şifreleme sonucu elde edilen veri ile kaotik çekerden elde edilen verinin XOR yapılarak güncellenmesi ile elde edilerek sisteme girilmiştir.

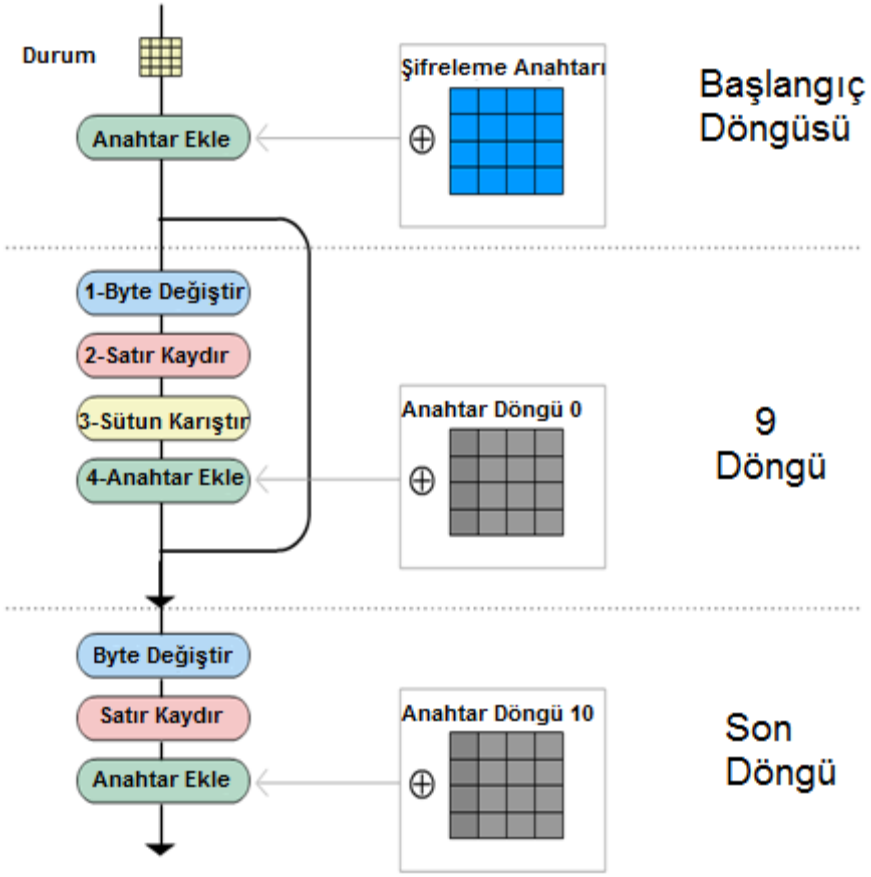


Şekil 5.11. Önerilen Hibrit SRSÜ genel tasarım [5].

AES blok şifreleme algoritması, geliştirilen sistemde algoritmik kısmı için kullanılmıştır. AES'in algoritmik kısmı rasgele sayı üreteçleri için gerekli olan R1-R3 gereksinimlerini sağlamaktadır [71]. AES, 1997 yılında J. Daemen ve V. Rijmen tarafından tasarlanmış ve 2000 yılında bir standart olarak seçilmiş güncel blok şifreleme standardıdır. DES ve 3DES gibi şifreleme algoritmalarına göre daha güvenli ve verimlidir.

AES, değiştirme (subsitution) ve karıştırma (permutation) yapısına dayalı tekrarlı bir blok şifredir. AES, durum (state) denilen 4x4 sütun-öncelikli bayt matrisi üzerinde çalışır. Matristeki işlemler de özel bir sonlu cisim (finite field) üzerinde yapılmaktadır. AES, 128, 192, 256 bit anahtar uzunluklarını desteklemektedir. Döngülerin tekrar sayıları 128-bit, 192-bit ve 256-bit anahtar uzunlukları için sırası ile 10, 12 ve 14'tür [71].

AES Şekil 5.12'de gösterildiği gibi herbiri 128 bit veri girişi ile işlenen katmanlardan oluşur. Bu katmanlar anahtar ekle (key addition), byte değiştir (byte substitution), satır kaydır (ShiftRow) ve sütun karıştırdır (MixColumn). Son döngüde ise sütun karıştırma işlemi kullanılmamaktadır. Byte değiştir katmanı 16 adet S-kutusundan oluşmaktadır. Bu S kutuları özdeş ve AES'in tek doğrusal olmayan elemanlarıdır. Satır kaydır bir byte düzeyinde verinin permutasyonudur. Sütun karıştır ise dört bayt'lık blokları birleştiren matris işlemidir.

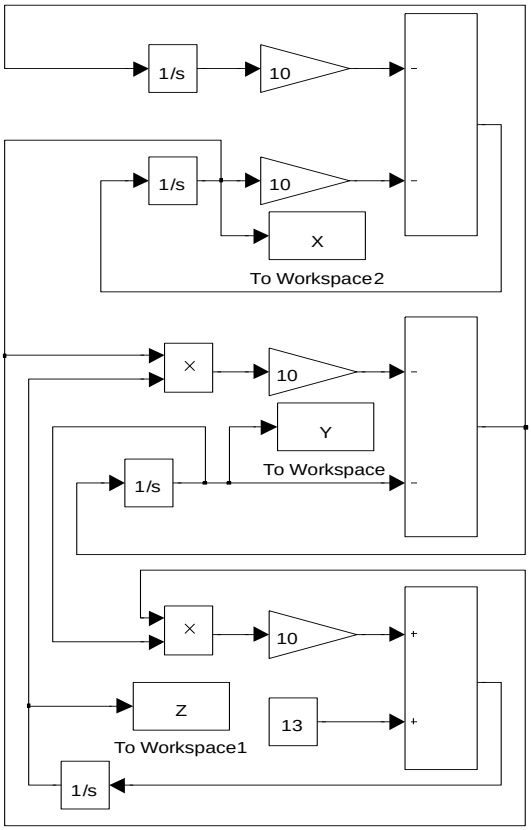


Şekil 5.12. AES Yapısı [72].

Önerilen sistemde kullanılan Burke-Shaw, nonlineer otonom kaotik sistemi için denklem takımları aşağıda Denklem (5.4)'de verilmiştir. Bu denklem takımlarındaki α ve β parametrelerinin değişimi kaotik sistemin dinamik davranışını değiştirmektedir. Bu nedenle bu parametrelerin değeri oldukça önemlidir. Yapılan çalışmada α parametresi 10.0 ve β parametresi ise 13.0 olarak seçilmiştir. Kaotik sistem başlangıç koşulları x_0=0.6, y_0=0.0 ve z_0=0.0' tir [73].

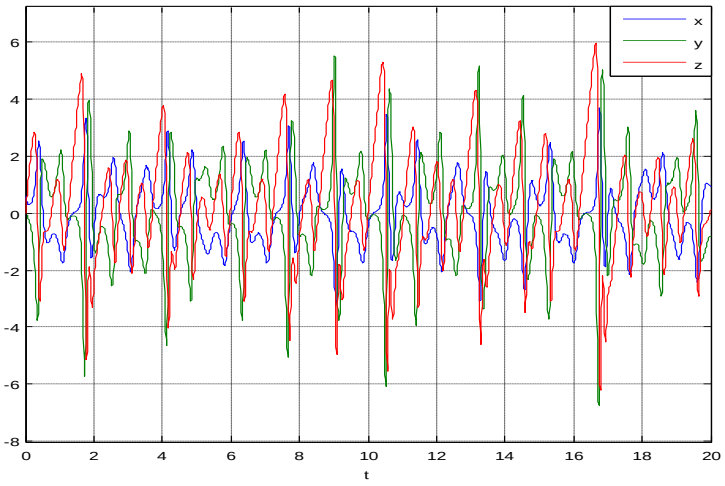
$$\begin{aligned} dx/dt &= -\alpha \cdot x - \alpha \cdot y \\ dy/dt &= -\alpha \cdot x \cdot z - y \\ dz/dt &= \alpha \cdot x \cdot y + \beta \end{aligned} \tag{5.4}$$

Öncelikli olarak yukarıda verilen Burke Shaw kaotik sistemi, yapılan elektronik devre tasarımı sonuçlarını doğrulamak amacıyla Matlab-Simulink programında modellenmiştir. Şekil 5.13'de Burke Shaw kaotik sisteminin Matlab-Simulink'de yapılan modellemesi görülmektedir [73].

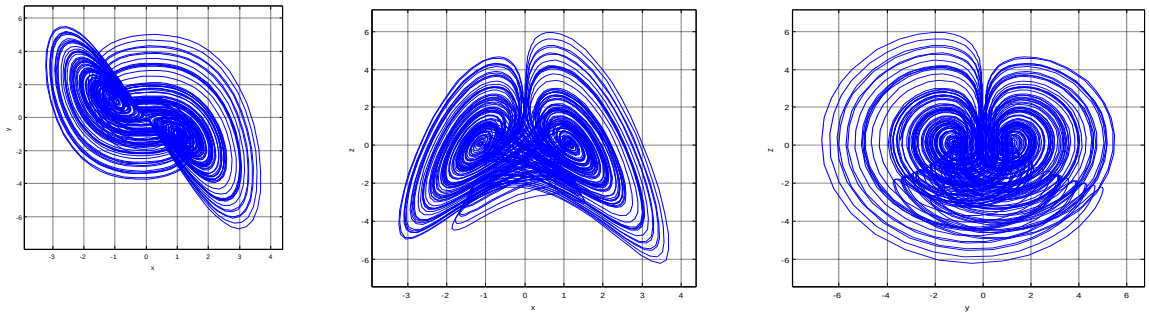


Şekil 5.13. Burke Shaw kaotik sistemi Matlab-Simulink modeli [73].

Şekil 5.14'de ve Şekil 5.15 (a), (b) ve (c)'de Burke Shaw kaotik sistemi Matlab-Simulink programı modelleme sonuçlarından elde edilen çıkış grafikleri verilmiştir. Şekil 5.15.a Burke Shaw kaotik sisteminin x çıkışının y çıkışına göre, Şekil 5.15.b x çıkışının z çıkışına göre ve Şekil 5.15.c y çıkışının z çıkışına göre değişimini göstermektedir.



Şekil 5.15. Burke Shaw kaotik sistemi x, y, ve z çıkış grafikleri [73].



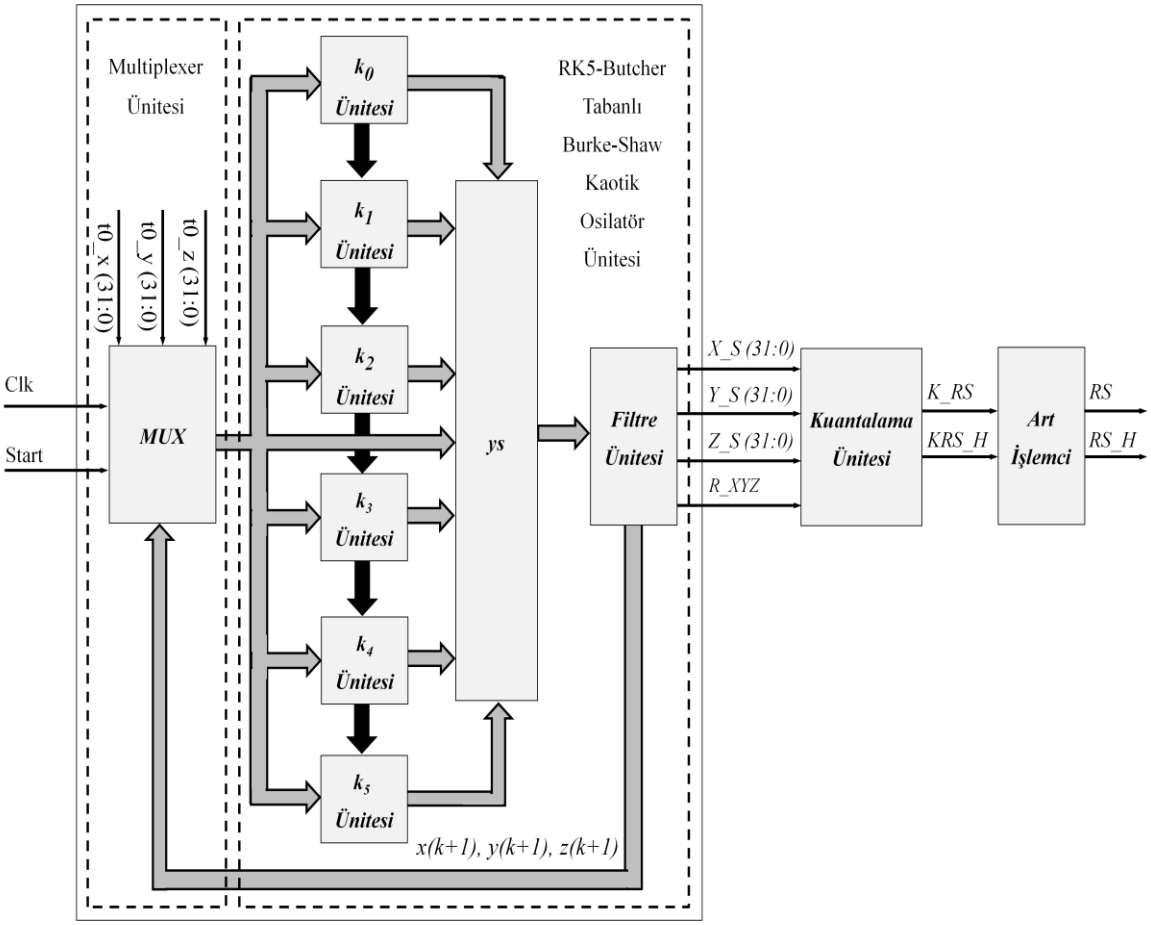
(a) (b) (c)

Şekil 5.15. Matlab-Simulink programı modelleme sonuçlarından elde edilen çıkış grafikleri (a) x sinyalinin y sinyaline gore değişimi (b) x sinyalinin z sinyaline gore değişimi (c) y sinyalinin z sinyaline gore değişimi [73].

Önerilen sistemde saf SRSÜ'nün R4 güvenlik gereksinimini sağlaması amacıyla çıkış fonksiyonuna ve geçiş fonksiyonuna kaos tabanlı GRSÜ ünitesi eklenmiştir. GRSÜ ünitesi, 32-bit IEEE 754-1985 floating-point number standardına uygun olarak FPGA çiplerinde çalışmak üzere donanımsal olarak tasarlanmıştır. Tasarlanan kaotik osilatör ünitesi bir donanım tanımlama dili olan VHDL'de kodlanmış ve Xilinx ISE 14.1 tasarım platformu kullanılarak Virtex-6 FPGA çipi için sentezlenmiştir. Tasarım aşamasında kullanılan Adder, Divider, Multiplier ve diğer modüller Xilinx'in IP CORE Generator Tools'u

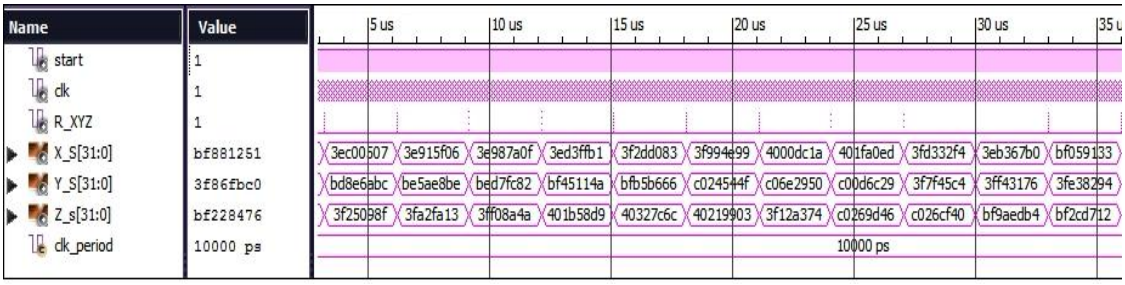
kullanılarak oluşturulmuştur. Şekil 5.16'de RK5-Butcher algoritması ile tasarımı yapılan kaotik GRSÜ ünitesi blok diyagramı görülmektedir [74]. Kaotik TRNG ünitesinin girişinde bulunan 32-bit τ sinyali algoritmanın adım sayısını ifade etmektedir. 1-bitlik Clk ve Start sinyalleri ise kaotik osilatörün bağlı bulunduğu sistem ile arasındaki senkronizasyonu sağlamak için kullanılmaktadır. Osilatörde MUX(Çoklayıcı-Multiplexer), k1, k2, k3, k4, ys üniteleri RK4 algoritmasının katsayılarını hesaplamaktadır. ys ünitesi ise k1, k2, k3, k4 değerlerini algoritmadaki sabit değerler ile çarparak osilatör çıkış sinyallerini elde etmekte ve bu değerleri filter ünitesine göndermektedir. Filter ünitesi kaotik osilatörün çıkışında oluşabilecek olası hata sinyallerini filtrelemek amacıyla kullanılmıştır. Filtre ünitesinden çıkan 32-bitlik X_S, Y_S ve Z_S sinyalleri kaotik osilatörün ürettiği sinyallerdir. Bununla birlikte bu sinyaller osilatörün t+1 anında üreteceği çıkış sinyalleri için başlangıç şartlarını oluşturmakta kullanılacağınan dolayı bu sinyaller x(k+1), y(k+1) ve z(k+1) sinyalleri olarak tekrar MUX ünitesine gönderilmektedir. MUX ünitesinin görevi Start sinyali geldiğinde sistemin ihtiyaç duyduğu başlangıç şartlarının atanmasını sağlamaktır. Diğer bir ifade ile sisteme kullanıcı tarafından atanan başlangıç şartları (t0_x, t0_y ve t0_z) ile sistemin çıkışından elde edilen başlangıç şartları (x(k+1), y(k+1) ve z(k+1)) arasında seçim yaparak bu sinyalleri sisteme göndermektir. 1-bitlik R_XYZ sinyali ise kaotik osilatörün ürettiği sinyallerin hazır olduğunu göstermektedir. Kaotik osilatör çıkış sinyalleri ürettiğinde R_XYZ sinyali '1' olmakta, diğer durumlarda ise '0' değerini çıkışa göndermektedir. Kuantalama ünitesi Filtre ünitesinden aldığı değerleri kullanarak karşılaştırma işlemine tabi tutmaktadır. Burke-Shaw kaotik osilatörü üretmiş olduğu değerleri Kuantalama ünitesine göndermektedir. Bu ünite dışarıdan gelen 32-bitlik sinyali Denklem (5.5)'de verilen işlemi gerçekleştirmek amacıyla bir karşılaştırma işlemi yapmaktadır. Burada σ değeri eşik değeri olarak adlandırılmakta ve bu değer kullanılan kaotik sistemin karakteristiğine göre değişiklik göstermektedir. Bu ünite çıkışından elde edilen K_RS sinyali rasgele sayıların taşındığı sinyalleridir. KRS_H sinyali ise ünite çıkışından rasgele sinyallerin alındığını göstermektedir. Son aşamada ise üretilen K_RS ve KRS_H sinyalleri ART ünitesine gönderilerek burada XOR işlemine tabii tutulmakta ve buradan rasgele sayılar çıkışa aktarılmaktadır [74].

$$\mathrm{RS}(x,y,z)=\begin{cases}0 & x,y,z<\sigma\\ 1 & x,y,z>=\sigma\end{cases} \tag{5.5}$$

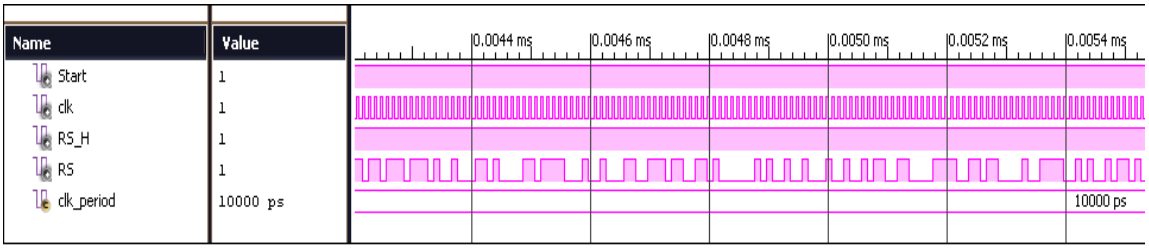


Şekil 5.16. Kaos tabanlı GRSÜ ünitesi blok diyagramı [74].

RK-5 Butcher algoritması kullanılarak FPGA tabanlı olarak tasarımı yapılan Burke Shaw kaotik osilator ünitesi, Xilinx Virtex-6 ailesinin en küçük çiplerinden olan XC6VLX550T çipi (Ek2) için sentezlenmiştir. Tasarımı yapılan ünitenin çalışma frekansı, Xilinx ISE Design Tools 14.2 simülasyon programı kullanılarak 373 MHz olarak elde edilmiştir. FPGA tabanlı kaotik osilatörün Xilinx ISE Simülatöründen elde edilen sonuçlar Şekil 5.17 ve Şekil 5.18'de verilmiştir. Şekilde kaotik osilatörün ISE Design Tools kullanılarak FPGA'de gerçeklenmesinden elde edilen x, y ve z sinyallerinin ayrıklaştırılmış karşılıkları olan x1out, x2out ve x3out çıkışlarının zaman serilerine ait değerler görülmektedir. Xilinx ISE simülatöründe sonuçların daha kolay incelenebilmesini sağlamak amacıyla simülasyon sonuçları, hexadecimal formatta gösterilmiştir [74].



Şekil 5.18. FPGA tabanlı Burke-Shaw kaotik osilatörü Xilinx ISE Simulatörü tarafından üretilen zaman serileri.



Şekil 5.19. FPGA tabanlı kaotik GRSÜ ünitesi Xilinx ISE Simulatörü sonuçları.

Tablo 5.3'de FPGA tabanlı kaotik GRSÜ ünitesi için sentezleme işleminin ardından gerçekleştirilen Place-Route işleminden elde edilen FPGA çip istatistikleri verilmiştir.

Tablo 5.3. FPGA-tabanlı kaotik GRSÜ ünitesi FPGA çip istatistikleri

FPGA Çipi	**Slice Regs. Sayısı / %**	**LUTs Sayısı/ %**	**Occupied Slices Sayısı/ %**	**IOBs Sayısı/ %**	**Minumum Periyot (ns)**
Virtex-6	80.274 / 12	80.039 / 23	24.281 / 28	4 / 1	2.680

5.2.2. Sonuçlar

Bu bölümde saf sözde rasgele sayı üretecinden elde edilen bit dizilerine eksikliklerinin giderilmesi ve kullanılan fonksiyonların karmaşıklığını arttırmak amacıyla sisteme ek girdi eklenmesi ve sonuçları üzerinde durulmuştur. Saf SRSÜ'ler R1-R4 güvenlik gereksinimlerinden sadece ilk üçünü sağlamaktadır. Bu amaçla sistemin rasgeleliğinin ve güvenliğinin arttırılması amacıyla saf SRSÜ geçiş ve çıkış fonksiyonlarına ek girdi eklenerek Hibrit SRSÜ sistemi geliştirilmiştir. Geliştirilen sistemde iç durum değeri her adımda gerçek rasgele veri ile güncellendiği için herhangi bir saldırgan tohum değerini veya iç durum değerini bilse dahi üretilen rasgele sayıları tahmin edemeyecektir. Bu sayede R4 güvenlik gereksinimi de sağlamıştır. Sistemden elde edilen rasgele 1000000 bit dizisinin NIST test sonuçları Tablo 5.4'te gösterilmiştir.

Tablo 5.4. Önerilen Hibrit SRSÜ NIST testi sonuçları

Test Adı	**P değeri**		**Sonuç**
Frekans Testi	0.310		Başarılı
Blok Frekans Testi	0.228		Başarılı
Akış Testi	0.044		Başarılı
En Uzun Birler Akış Testi	0.260		Başarılı
İkili Matris Rankı Testi	0.898		Başarılı
Ayrık Fourier Dönüşümü Testi	0.532		Başarılı
Örtüşmeyen Şablon Eşleştirme Testi	0.746		Başarılı
Örtüşen Şablon Eşleştirme Testi	0.123		Başarılı
Maurer'in Evrensel İstatistik Testi	0.628		Başarılı
Doğrusal Karmaşıklık Testi	0.201		Başarılı
Seri Testi	0.712		Başarılı
	0.688		
Lempel Ziv Testi	1		Başarılı
Yaklaşık Entropi Testi	0.603		Başarılı
Kümülatif Toplam Testi	0.257		Başarılı
Rasgele Yürüyüş Testi	-4	0.348	Başarılı
	-3	0.969	
	-2	0.211	
	-1	0.417	
	1	0.751	
	2	0.667	
	3	0.995	
	4	0.023	
Rasgele Yürüyüş Değişken Testi	-9	0.914	Başarılı
	-8	0.814	
	-7	0.952	
	-6	0.889	
	-5	0.986	
	-4	0.822	
	-3	0.545	
	-2	0.909	
	-1	0.597	
	1	0.448	
	2	0.804	
	3	0.970	
	4	0.965	
	5	0.741	
	6	0.705	
	7	0.901	
	8	0.969	
	9	0.860	

6. SONUÇ VE ÖNERİLER

Bu tezde enropi kaynağı olarak, kaotik sistemler ve halka osilatörler kullanılarak rasgele bit dizileri elde edilmiştir. Elde edilen rasgele bit dizileri üzerinde son işlemin etkisi incelenmiş ve yeni son işlemler önerilmiştir. Gerçekleştirilen çalışmaların sonucu ve daha sonra yapılması planlanan çalışmalar aşağıda ayrıntılarıyla açıklanmıştır:

- GRSÜ üretim yöntemlerinden olan halka osilatör tarafından üretilen faz seğirmesi kullanılarak FPGA ortamında donanımsal olarak gerçekleştirilmiştir. Bu sistem Sunar ve diğerleri tarafından önerilmiş bir sistemdir. Sunar tarafından önerilen sistemde entropi kaynağından üretilen rasgele bit dizilerinde korelasyon olması nedeniyle son işlemsiz olarak testleri geçememiştir. Bunun sonucunda resilient fonksiyonu olarak adlandırılan son işleme tabi tutulmuştur. Kullanılan son işlem çıkış oranını 1/16 oranına düşmektedir. Ayrıca literatürde kullanılan birçok son işlemde çıkış oranı azalmaktadır. Bu amaçla GRSÜ'nün çıkış oranını düşürmeden, istatistikî zayıflıkları gideren yeni bir son işlem algoritması önerilmiştir. Bu amaçla önerilen son işlemde lojistik harita kullanılmıştır. Lojistik harita kullanımı ile sağlam ve kaliteli sayılar üretilebilmiştir. Lojistik harita kullanımına bağlı olarak RO ve tümleyen sayısı azaltılabilmiştir. Sistemde güç tüketiminin yüksek olması bir dezavantajdır. Ancak sistemde lojistik haritanın son işlem olarak kullanımı sistemi saldırılara karşı güvenilir yapmaktadır. Literatürde bulunan birçok son işlem çıkış oranını düşürürken önerilen son işlem ile çıkış oranı düşürülmemiş olup sistemden yüksek çıkış oranı ile sayı üretilebilmiştir. Bu sistemin bir diğer avantajı ise lojistik harita kullanımı ile rasgelelik kaynağının korelesyonu giderilmiştir. Test sonuçlarına göre GRSÜ yapısı daha basit bir yapıya dönüştürülebilmiştir. Bu sonuçlar istatistiksel testler ile de gösterilmiştir.
- Sunar için resilient fonksiyon son işlemi sonrası 2.5 Mbit/s iken önerilen yeni son işlem sonucu üretilen bit hızı yaklaşık 20 Mbit/s olarak hesaplanmıştır.
- GRSÜ tasarımlarından elde edilen bit dizilerinin, istatistikî eksikliklerinin giderilmesi amacıyla Von Neumann son işlem değiştirilerek yeni bir son işlem algoritması önerilmiştir. Von Neumann son işlem çıkış bit oranını yaklaşık olarak 1/4 oranına düşürürken önerilen son işlem 1/2 oranında azaltmaktadır.

- Rasgele sayı üreteçlerinden elde edilen bit dizilerinin kriptografik uygulamalarda kullanılması için dört gereksinimi (R1, R2, R3, R4) sağlaması gerekmektedir. Ancak saf SRSÜ olarak kullanılan sistemlerin geneli bu dört gereksinimden üç tanesini (R1, R2, R3) sağlamaktadır. Bu sistemlerin kriptografik uygulamalarda kullanılabilmesi için R4 gereksiniminin de sağlanması gerekir. Bu amaçla Saf SRSÜ sistemlerine ek girdi eklenerek Hibrit SRSÜ sistemleri geliştirilmiştir. Bu çalışmada R4 gereksinimini sağlamak için yeni bir Hibrit SRSÜ sistemi geliştirilmiştir. Saf SRSÜ sistemi için AES ve ek girdi için de kaos tabanlı (Burke Shaw kaotik çeker) GRSÜ kullanılmıştır. AES ile üretilen rasgele bit dizileri R1, R2, R3 gereksinimlerini sağlamaktadır. R4 gereksinimi kaos tabanlı GRSÜ tasarımı ile sağlanmıştır. Önerilen tasarımda, iç durum değerinin ve çıkış fonksiyonun GRSÜ sistemi ile sürekli olarak güncellenmesi sayesinde sistemin rasgeleliği ve güvenliği arttırılmıştır. Elde edilen sonuçlar NIST testini başarılı olarak geçmiştir.
- Elde edilen bit dizileri istatistiksel testlere tabi tutulması NIST 800-22 test paketi C# dilinde yazılmıştır. Bu program sayesinde, birçok kullanıcı elde ettikleri rasgele sayıları kolaylıkla test edebileceklerdir. NIST 800-22'de tanımlı tüm testler ayrıntılarıyla açıklanmıştır.

Öneriler:

- Tasarımı yapılmış diğer gerçek rasgele sayı üreteçlerine ek girdi eklenerek sonuçlar karşılaştırılacaktır.
- Lojistik harita yerine daha az lojik eleman ile gerçekleştirilebilecek, daha kolay kaotik sistemlerin uygulaması yapılacaktır.
- NIST testleri gerçekleştirilmiştir, ancak 2011 yılında Werner Schindler tarafından önerilen AIS 31 testlerinin de yazılımı gerçekleştirilecektir. Literatürdeki GRSÜ'ler bu testten geçirilerek elde edilen sonuçlar karşılaştırılacaktır.
- NIST 800-22 test paketi için yazılan yazılım kolay erişim amacıyla web ortamına taşınacaktır.

KAYNAKLAR

[1]. **Katz, J. Lindell, Y.,** 2008. Introduction to modern cryptography : principles and protocols, Chapman & Hall

[2]. **Uygulamalı Matematik Enstitüsü**, 2004. Kriptolojiye Giriş Ders Notları, Kriptorafi Bölümü-ODTÜ, Türkiye

[3]. **Delfs, H. and Knebl, H.,** 2007. Introduction to Cryptography Principles and Applications Springer-Verlag

[4]. **Wold, K.,** 2011. Security Properties of a Class of True Random Number Generators in Programmable Logic, Thesis submitted to Gjøvik University College for the degree of Doctor of Philosophy in Information Security

[5]. **Koç, Ç.,** 2009. Cryptographic Engineering, Springer-Verlag

[6]. **Santoro, R., Sentieys, O. and Roy, S.,** 2009. On-the fly evaluation of FPGA-based True Random Number Generator, *2009 IEEE Computer Society Annual Symposium on VLSI*, 55-60

[7]. **Yalcın, M. E., Suykens, J. A. K. and Vandewalle, J.,** 2004. True random bit generation from a double scroll attractor, *IEEE Trans. Circuits and Systems-I,* **51(7)**, 1395–1404

[8]. **Kilmann, W., and Schindler, W.,** 2001. Ais 31: Functionality classes and evaluation methodology for physical random number generators. version 1 (25.09.2001), *Federal Office for Information Security Technical Report*, Germany

[9]. **Pareschi, P.,** 2006. Chaos-based random number generators:monolithic implementation, testing and applications, *PHD. Thesis*, Alma Mater Studiorum Universita Di Bologna

[10]. **Bernstein, G. M. and Lieberman, M. A.,** 2000. Secure Random Number Generation using Chaotic Circuit, *in IEEE Transaction on Circuit and Systems I: Fundamental Theory and Applications*, vol. 47, no. 9, pp. 1157–1164

[11]. **Stojanovski, T. and Kocarev, L.,** 2001. Chaos-Based Random Number Generators - Part I: Analysis, *in IEEE Transaction on Circuit and Systems I: Fundamental Theory andApplications,* vol. 38, no. 3, pp. 281-288

[12]. **Ulam, S.M. and Neumann, J. V.,** 1947. On combination of stochastic and deterministic process, *in Bulletin of American Mathematical Society*, no. 53, pp 1120–1132

[13]. **Drutarowsky, M. and Galajda, P.,** . 2006. Chaos-Based True Random Generator Embedded In a Mixed-Signal Reconfigurable Hardware, Journal of Electirical Engineering,Vol.57,No. 4, 218-225

[14]. **Alatas, B., Akin, E. ve Ozer, A.B.,** 2007. Kaotik Haritalı Parçacık Sürü Optimizasyon Algoritmaları, *XII. Elektrik Elektronik Bilgisayar Biyomedikal Mühendisliği Ulusal Kongresi*, Eskişehir Osmangazi Üniversitesi

[15]. **Suresh,V.B. and Burleson, W.P.,** 2010. Entropy Extraction in Metastability-based TRNG, Hardware-Oriented Security and Trust (HOST), *2010 IEEE International Symposium on*, 135-140

[16]. **Davies, R.B.,** 2002. Exclusive OR (XOR) and Hardware Random Number Generators, http://webnz.com/robert

[17]. **Dichtl, M.,** 2007. Bad and Good Ways of Post-Processing Biased Physical Random Numbers, *Fast Software Encryption Lecture Notes in Computer Science*,Volume 4593, 137-152

[18]. **Rukhin, A., Soto, J., Nechvatal, J., Smid, M. and Banks, D.,** 2001. A statistical test suite for random and pseudorandom number generators for statistical applications, *NIST Special Publication in Computer Security*

[19]. **Kenny, C., and Mosurski, K.,** 2005. Random number Generators, *The distributed systems group*, Trinity college dublin Management Science and Information Systems Studies Project Report

[20]. **Kohlbrenner, P.W.,** 2003. The Design and Analysis of a True Random Number Generator in a Field Programmable Gate Array, *A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University*

[21]. **Márton, K., Suciu, A., Săcărea, C. and Creţ, O.,** 2012. Generation and testing of random numbers for cryptographic applications, *Proceedings Of The Romanian Academy*, Series A, Volume 13, Number 4/2012, 368–377

[22]. **Krhovjak, J.,** 2009. Crytographic random and pseudorandom data generators, *Dissertation Thesis*, Fakulty of İnformatics Masaryk University

[23]. **Kapur, J. N. and Kesavan, H.K.,** 1992. Entropy Optimization Principles With Applications, *Academic Press*

[24]. **Papoulis, A. and Pillai, S.U.**, 2002. Probability, Random Variables and Stochastic Processes, 4th edition, McGraw Hill

[25]. **Cover, T. M. and Thomas, J. A.,** 2006. Elements of Information Theory, *2nd Edition, ISBN: 0-471-24195-4*, 776 pages

[26]. **Abumuala, M., Khalifa, O. and Hashim, A. A.,** 2010. A New Method for Generating Cryptographically Strong Sequences of Pseudo Random Bits for Stream Cipher, *International Conference on Computer and Communication Engineering (ICCCE 2010)*, 11-13 May, 1-4, Kuala Lumpur, Malaysia

[27]. **Özkaynak, F. ve Özer, A.B.,** 2006. Lojistik Harita ile Rasgele Sayı Üretilmesi ve İstatistiki Yöntemlerle Sınanması, *Journal of İstanbul Kültür University*, 2006/3, 129-133

[28]. **Chan, J. J. M., Sharma, B., L. J., Thomas, G., Thulasirami R. and Thulasiraman, P.,** 2011. True random number generator using GPUs and Histogram equalization techniques, *2011 IEEE International Conference on High Performance Computing and Communications*,161-170

[29]. **Xingyuan, W., Xue, Q., and Lin, T.,** 2012. A Novel True Random Number Generator Based on Mouse Movement and a One-Dimensional Chaotic Map, *Hindawi Publishing Corporation Mathematical Problems in Engineering Volume 2012*, Article ID 931802, 9 pages

[30]. **Wang, A.L. and Xu, Z.,** 2011. Asymptotic Statistical Analysis of Pseudo Random Numbers, 978-1-4244-8728-8111, *IEEE*

[31]. **Hoţoleanu, D., Creţ, O., Suciu, A., Gyorfi, T. and Văcariu, L.,** 2010. Real-time Testing of True Random Number Generators through Dynamic Reconfiguration, *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, 247-250

[32]. **Guler, U. and Ergun S.,** 2012. A high speed, fully digital IC random number generator, *Int. J. Electron. Commun. (AEU) 66 (2012)*, 143– 149

[33]. **Murphy, J.P.,** 2012. Field-programmable true random number generator, *Electronics Letters*, 10th May 2012, Vol. 48 No. 10, 565-566

[34]. **Fischer, V., Aubert, A., Bernard, F., Valtchanov, B., Danger, J.-L. and Bochard, N.,** 2009. True Random Number Generators in Configurable Logic Devices, February 12, *Project ANR – ICTeR*

[35]. **Random**, 2008. http://www.random.org/randomness/

[36]. **Davis, D., Ihaka, R. and Fenstermacher, P. P.,** 1994. Cryptographic randomness from air turbulence in disk drives, In Y. Desmedt editor, *Advances in Cryptology (Crypto 94)*, vol. 839, 114–120, Heidelberg, Germany: Springer-Verlag

[37]. **Yıldırım, S.,** 2012. A True Random Number Generator In Fpga For Cryptographıc Applıcatıons, *A Thesıs Submıtted To The Graduate School Of Natural And Applıed Scıences Of Mıddle East Technıcal Unıversıty*

[38]. **Neumann, J.V.,** 1951. Various techniques used in connection with random digits, *J. Research Nat. Bur. Stand., Appl. Math. Series*, vol. 12, pp. 36-38

[39]. **Varchola, M.,** 2008. FPGA Based True Random NumberGenerators for Embedded Cryptographic Applications, *Phd Thesis*, Faculty of Electrical Engineering and Informatics Department of Electronics and Multimedia Communications, Technical University of Kosice

[40]. **Barak, B. , Shaltiel, R. and Zomer, E.,** 2003. True Random Number Generators Secure in a Changing Environment*, In C¸ . K. Koc¸ and C. Paar, editors, Workshop on Cryptographic Hardware and Embedded Systems—CHES 2003*, 166– 180, Berlin, Germany, Lecture Notes in Computer Science, Vol. 2779, Springer-Verlag

[41]. **Sunar, B., Martin, W. J. and Stinson, D. R.,** 2007. A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks, *IEEE Transactions on Computers*, vol. 58, no 1, 109–119

[42]. **Bagini, V. and Bucci, M.,** 1999. A Design of Reliable True Random Number Generator for Cryptographic Applications*. In C¸ .K.Koc¸ and C. Paar, editors, Workshop on Cryptographic Hardware and Embedded Systems—CHES 1999*, 204–218, Berlin, Germany, Lecture Notes in Computer Science, Vol. 1717. Springer-Verlag

[43]. **Jun, B. and Kocher, P.,** 1999. The Intel random number generator, *White Paper Prepared for Intel Corporation,* April 1999V.

[44]. **Fischer, V. and M. Drutarovsk´y.**, 2002. True Random Number Generator Embedded in Reconfigurable Hardware, *In B. S. Kaliski Jr., C¸ . K. Koc¸, C. Paar, editors, Workshop on Cryptographic Hardware and Embedded Systems—CHES 2002,* pp. 415–430, Berlin, Germany, Lecture Notes in Computer Science, Vol. 2523. Springer-Verlag Berlin Heidelberg

[45]. **Tkacik, T. E.,** 2003. A Hardware Random Number Generator, *In Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'02) (2003)*, vol. 2523 of Lecture Notes in Computer Science, Springer, 450–453

[46]. **Dichtl, M.,** 2003. How to Predict the Output of a Hardware Random Number Generator, *In Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'03) (2003),* vol. 2779 of Lecture Notes in Computer Science, Springer, 181–188

[47]. **Epstein, M., Hars, L., Krasinski, R., Rosner, M. and Zheng, H.,** 2003. Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts, *In C. D.Walter, C¸ . K. Koc¸, C. Paar, editors, Workshop on Cryptographic Hardware and Embedded Systems—CHES 2003*, Lecture Notes in Computer Science, Vol. 2779, 152–165. Springer-Verlag Berlin Heidelberg

[48]. **Kohlbrenner, P. and Gaj, K.,** 2004. An embedded true random number generator for FPGAs, International Symposiumon Field Programmable Gate Arrays. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, 71–78, ACM Press, New York, NY

[49]. **Golic, J. D.,** 2006. New Methods for Digital Generation and Postprocessing of Random Data, *IEEE Transactions on Computers*, 55, 10, 1217–1229.

[50]. **Golic, J.D.,** 2004. New paradigms for digital generation and post-processing of random data, *Technical report, Cryptology ePrint Archive, Report 2004/254, 2004.* Online. Available: http://eprint.iacr.org/2004/254.ps

[51]. **Schellekens, D., Preneel, B. and Verbauwhede,** 2006. FPGA Vendor Agnostic True Random Number Generator, *In Proceedings of the 16th International Conference on Field-Programmable Logic and Applications (FPL'06)*, IEEE, 1–6

[52]. **Dichtl, M. and Golic, J.D.,** 2007 High-Speed True Random Number Generation with Logic Gates Only, Pascal Paillier, Ingrid verbauwhede, editors, *Proceedings of the Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop*, Vienna, Austria, September 10–13, 2007. Lecture Notes in Computer Science, vol. 4727, 45-62, Springer Verlag

[53]. **Wold, K., And Tan, C. H.,** 2008. Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings, *In Proceedings of the 4th IEEE InternationalConference on ReConFigurable Computing and FPGAs (ReConFig'08)*, 385–390

[54]. **Vasyltsov, I., Hambardzumyan, E., Kım, Y.-S. and Karpinskyy, B.,** 2008. Fast Digital TRNG Based on Metastable Ring Oscillator, *In Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'08)*, vol. 5154 of Lecture Notes in Computer Science, Springer, 164–180

[55]. **Danger, J.-L., Guilley, S. and Hoogvorst, P.,** 2009. High Speed True Random Number Generator Based on Open Loop Structures in FPGAs, *Microelectronics Journal 40*, 11 (November 2009), 1650–1656

[56]. **Varchola, M. and Drutarovsky, M.**, 2010. New High Entropy Element for FPGA based True Random Number Generators, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop*, Santa Barbara, USA, Springer, 351-365

[57]. **Tudoran, R. , Cret, O., Banescu, S. and Suciu, A.,** 2009. Implementing true random number generators by generating crosstalk effects in FPGA chips, *Proceedings of the 6th FPGA world Conference*, 25-31

[58]. **Cret, O., Tudoran, R., Suciu, A. and Gyorfi, T.,** 2009. Implementing True Random Number Generators in FPGAs by Chip Filling, *In Proceedings of the International Conference on Security and Cryptography, SECRYPT'09*, 62-67

[59]. **Güneysu, T.,** 2010. True Random Number Generation in Block Memories of Reconfigurable Devices" , *Field-Programmable Technology (FPT),*200 – 207

[60]. **Güneysu, T.,** 2012. Using Data Contention in Dual-ported Memories for Security Applications , *Journal of Signal Processing Systems,* 15-29

[61]. http://class.ece.iastate.edu/cpre583/project_presentations/PUFs_report.pdf

[62]. **Gassend, B., Clarke, D., Dijk, M.V. and Devadas, S.,** 2003. Delay-based circuit authentication and applications, *In Proceedings of the 2003 ACM symposium on Applied computing , SAC '03*, 294–301, New York, NY, USA

[63]. **Suh, G. E. and Devadas, S.,** 2007. Physical unclonable functions for device authentication and secret key generation, *In Proceed- ings of the 44th annual Design Automation Conference , DAC '07*, 9–14, New York, NY, USA

[64]. **Guajardo, J., Kumar, S., Schrijen, G.-J. and Tuyls, P**., 2007. Physical unclonable functions and public-key crypto for fpga ip protection, *In Field Programmable Logic and Applications*, International Conference on,189 –195

[65]. **Kumar, S., Guajardo, J., Maes, R., Schrijen, G.-J., and Tuyls, P.,** 2008. Extended abstract: The butterfly PUF protecting ip on every FPGA, *In Hardware-Oriented Security and Trust, IEEE International Workshop on* , 67 –70

[66]. **Ergün, S. and Özoğuz, S.,** 2007. Truly random number generators based on a non-autonomous chaotic oscillator, *AEU - International Journal of Electronics and Communications*, Volume 61, Issue 4, 235–242

[67]. **Drutarovsky, M.,** 2007. A Robust Chaos-Based True Random Number Generator Embedded in Reconfigurable Switched-Capacitor Hardware, *Radioelektronika*,17th International Conference, 1-6

[68]. **Zidan, M.A., Radwan, A.G. and Salama, K.N.,** 2011. Random number generation based on digital differential chaos, *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, 1-4

[69]. **Yoo, S.-K., Karakoyunlu, D., Birand, B. and Sunar, B.,** 2010. Improving the Robustness of Ring Oscillator TRNGs, *TRETS 3(2):* 9

[70]. **Tuncer, T. and Celik, V., 2013.** Hybrid PRNG based on Logistic Map , *Signal Processing and Communications Applications Conference (SIU)*, 2013 21st ,pp.1-4

[71]. **Daemen, J. and Rijmen, V.,** 1998. AES Proposal: Rijndael, First Advanced Encryption Conference, California

[72]. http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf

[73]. **Koyuncu, I., Uyaroğlu Y. ve Pehlivan, I.,** 2012. " Burke Shaw Kaotik Sisteminin Güvenli Haberleşme İçin Elektronik Devre Gerçeklemesi ve Senkronizasyon Uygulaması", 5. Uluslararası Bilgi Güvenliği ve Kriptoloji Konferansı, 279-284

[74]. **Koyuncu, I., Ozcerit A.T. ve Pehlivan, I.,** 2013. "An analog circuit design and FPGA-based implementation of the Burke-Shaw chaotic system", Optoelectronics and Advanced Materials– Rapid Communations, 7(9-10), 635-638

EKLER

EK1 Altera Cyclone FPGA EP4CE115F29C7 Datasheet

1. Cyclone IV FPGA Device Family Overview

CYIV-51001-1.5

Altera's new Cyclone® IV FPGA device family extends the Cyclone FPGA series leadership in providing the market's lowest-cost, lowest-power FPGAs, now with a transceiver variant. Cyclone IV devices are targeted to high-volume, cost-sensitive applications, enabling system designers to meet increasing bandwidth requirements while lowering costs.

Built on an optimized low-power process, the Cyclone IV device family offers the following two variants:

- Cyclone IV E—lowest power, high functionality with the lowest cost
- Cyclone IV GX—lowest power and lowest cost FPGAs with 3.125 Gbps transceivers

Cyclone IV E devices are offered in core voltage of 1.0 V and 1.2 V.

For more information, refer to the *Power Requirements for Cyclone IV Devices* chapter.

Providing power and cost savings without sacrificing performance, along with a low-cost integrated transceiver option, Cyclone IV devices are ideal for low-cost, small-form-factor applications in the wireless, wireline, broadcast, industrial, consumer, and communications industries.

Cyclone IV Device Family Features

The Cyclone IV device family offers the following features:

- Low-cost, low-power FPGA fabric:
  - 6K to 150K logic elements
  - Up to 6.3 Mb of embedded memory
  - Up to 360 18 × 18 multipliers for DSP processing intensive applications
  - Protocol bridging applications for under 1.5 W total power

Cyclone IV Device Handbook,
Volume 1
November 2011

Subscribe

- Cyclone IV GX devices offer up to eight high-speed transceivers that provide:
  - Data rates up to 3.125 Gbps
  - 8B/10B encoder/decoder
  - 8-bit or 10-bit physical media attachment (PMA) to physical coding sublayer (PCS) interface
  - Byte serializer/deserializer (SERDES)
  - Word aligner
  - Rate matching FIFO
  - TX bit slipper for Common Public Radio Interface (CPRI)
  - Electrical idle
  - Dynamic channel reconfiguration allowing you to change data rates and protocols on-the-fly
  - Static equalization and pre-emphasis for superior signal integrity
  - 150 mW per channel power consumption
  - Flexible clocking structure to support multiple protocols in a single transceiver block
- Cyclone IV GX devices offer dedicated hard IP for PCI Express (PIPE) (PCIe) Gen 1:
  - ×1, ×2, and ×4 lane configurations
  - End-point and root-port configurations
  - Up to 256-byte payload
  - One virtual channel
  - 2 KB retry buffer
  - 4 KB receiver (Rx) buffer
- Cyclone IV GX devices offer a wide range of protocol support:
  - PCIe (PIPE) Gen 1 ×1, ×2, and ×4 (2.5 Gbps)
  - Gigabit Ethernet (1.25 Gbps)
  - CPRI (up to 3.072 Gbps)
  - XAUI (3.125 Gbps)
  - Triple rate serial digital interface (SDI) (up to 2.97 Gbps)
  - Serial RapidIO (3.125 Gbps)
  - Basic mode (up to 3.125 Gbps)
  - V-by-One (up to 3.0 Gbps)
  - DisplayPort (2.7 Gbps)
  - Serial Advanced Technology Attachment (SATA) (up to 3.0 Gbps)
  - OBSAI (up to 3.072 Gbps)

- Up to 532 user I/Os
  - LVDS interfaces up to 840 Mbps transmitter (Tx), 875 Mbps Rx
  - Support for DDR2 SDRAM interfaces up to 200 MHz
  - Support for QDRII SRAM and DDR SDRAM up to 167 MHz
- Up to eight phase-locked loops (PLLs) per device
- Offered in commercial and industrial temperature grades

Device Resources

Table 1–1 lists Cyclone IV E device resources.

Table 1–1. Resources for the Cyclone IV E Device Family

Resources	EP4CE6	EP4CE10	EP4CE15	EP4CE22	EP4CE30	EP4CE40	EP4CE55	EP4CE75	EP4CE115
Logic elements (LEs)	6,272	10,320	15,408	22,320	28,848	39,600	55,856	75,408	114,480
Embedded memory (Kbits)	270	414	504	594	594	1,134	2,340	2,745	3,888
Embedded 18 × 18 multipliers	15	23	56	66	66	116	154	200	266
General-purpose PLLs	2	2	4	4	4	4	4	4	4
Global Clock Networks	10	10	20	20	20	20	20	20	20
User I/O Banks	8	8	8	8	8	8	8	8	8
Maximum user I/O (1)	179	179	343	153	532	532	374	426	528

Note to Table 1–1:

(1) The user I/Os count from pin-out files includes all general purpose I/O, dedicated clock pins, and dual purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the pin count.

Table 1–2 lists Cyclone IV GX device resources.

Table 1–2. Resources for the Cyclone IV GX Device Family

Resources	EP4CGX15	EP4CGX22	EP4CGX30 (1)	EP4CGX30 (2)	EP4CGX50 (3)	EP4CGX75 (3)	EP4CGX110 (3)	EP4CGX150 (3)
Logic elements (LEs)	14,400	21,280	29,440	29,440	49,888	73,920	109,424	149,760
Embedded memory (Kbits)	540	756	1,080	1,080	2,502	4,158	5,490	6,480
Embedded 18 × 18 multipliers	0	40	80	80	140	198	280	360
General purpose PLLs	1	2	2	4 (4)	4 (4)	4 (4)	4 (4)	4 (4)
Multipurpose PLLs	2 (5)	2 (5)	2 (5)	2 (5)	4 (5)	4 (5)	4 (5)	4 (5)
Global clock networks	20	20	20	30	30	30	30	30
High-speed transceivers (6)	2	4	4	4	8	8	8	8
Transceiver maximum data rate (Gbps)	2.5	2.5	2.5	3.125	3.125	3.125	3.125	3.125
PCIe (PIPE) hard IP blocks	1	1	1	1	1	1	1	1
User I/O banks	9 (7)	9 (7)	9 (7)	11 (8)	11 (8)	11 (8)	11 (8)	11 (8)
Maximum user I/O (9)	72	150	150	290	310	310	475	475

Notes to Table 1–2:

(1) Applicable for the F169 and F324 packages.
(2) Applicable for the F484 package.
(3) Only two multipurpose PLLs for F484 package.
(4) Two of the general purpose PLLs are able to support transceiver clocking. For more information, refer to the *Clock Networks and PLLs in Cyclone IV Devices* chapter.
(5) You can use the multipurpose PLLs for general purpose clocking when they are not used to clock the transceivers. For more information, refer to the *Clock Networks and PLLs in Cyclone IV Devices* chapter.
(6) If PCIe ×1, you can use the remaining transceivers in a quad for other protocols at the same or different data rates.
(7) Including one configuration I/O bank and two dedicated clock input I/O banks for HSSI reference clock input.
(8) Including one configuration I/O bank and four dedicated clock input I/O banks for HSSI reference clock input.
(9) The user I/Os count from pin-out files includes all general purpose I/O, dedicated clock pins, and dual purpose configuration pins. Transceiver pins and dedicated configuration pins are not included in the pin count.

Package Matrix

Table 1–3 lists Cyclone IV E device package offerings.

Table 1–3. Package Offerings for the Cyclone IV E Device Family (1)

Package	E144		M164		U256		F256		U484		F484		F780	
Size (mm)	22 × 22		8 × 8		14 × 14		17 × 17		19 × 19		23 × 23		29 × 29	
Pitch (mm)	0.5		0.5		0.8		1.0		0.8		1.0		1.0	
Device	User I/O	LVDS (2)	User I/O	LVDS (2)	User I/O	LVDS (2)	User I/O	LVDS (2)	User I/O	LVDS (2)	User I/O	LVDS (2)	User I/O	LVDS (2)
EP4CE6	91	21	—	—	179	66	179	66	—	—	—	—	—	—
EP4CE10	91	21	—	—	179	66	179	66	—	—	—	—	—	—
EP4CE15	81	18	89	21	165	53	165	53	—	—	343	137	—	—
EP4CE22	79	17	—	—	153	52	153	52	—	—	—	—	—	—
EP4CE30	—	—	—	—	—	—	—	—	—	—	328	124	532	224
EP4CE40	—	—	—	—	—	—	—	—	328	124	328	124	532	224
EP4CE55	—	—	—	—	—	—	—	—	324	132	324	132	374	160
EP4CE75	—	—	—	—	—	—	—	—	292	110	292	110	426	178
EP4CE115	—	—	—	—	—	—	—	—	—	—	280	103	528	230

Notes to Table 1–3:

(1) The E144 package has an exposed pad at the bottom of the package. This exposed pad is a ground pad that must be connected to the ground plane of your PCB. Use this exposed pad for electrical connectivity and not for thermal purposes.

(2) This includes both dedicated and emulated LVDS pairs. For more information, refer to the *I/O Features in Cyclone IV Devices* chapter.

Table 1–4 lists Cyclone IV GX device package offerings, including I/O and transceiver counts.

Table 1–4. Package Offerings for the Cyclone IV GX Device Family

Package	N148			F169			F324			F484			F672			F896		
Size (mm)	11 × 11			14 × 14			19 × 19			23 × 23			27 × 27			31 × 31		
Pitch (mm)	0.5			1.0			1.0			1.0			1.0			1.0		
Device	User I/O	LVDS (1)	XCVRs	User I/O	LVDS (1)	XCVRs	User I/O	LVDS (1)	XCVRs	User I/O	LVDS (1)	XCVRs	User I/O	LVDS (1)	XCVRs	User I/O	LVDS (1)	XCVRs
EP4CGX15	72	25	2	72	25	2	—	—	—	—	—	—	—	—	—	—	—	—
EP4CGX22	—	—	—	72	25	2	150	64	4	—	—	—	—	—	—	—	—	—
EP4CGX30	—	—	—	72	25	2	150	64	4	290	130	4	—	—	—	—	—	—
EP4CGX50	—	—	—	—	—	—	—	—	—	290	130	4	310	140	8	—	—	—
EP4CGX75	—	—	—	—	—	—	—	—	—	290	130	4	310	140	8	—	—	—
EP4CGX110	—	—	—	—	—	—	—	—	—	270	120	4	393	181	8	475	220	8
EP4CGX150	—	—	—	—	—	—	—	—	—	270	120	4	393	181	8	475	220	8

Note to Table 1–4:

(1) This includes both dedicated and emulated LVDS pairs. For more information, refer to the *I/O Features in Cyclone IV Devices* chapter.

Cyclone IV Device Family Speed Grades

Table 1–5 lists the Cyclone IV GX devices speed grades.

Table 1–5. Speed Grades for the Cyclone IV GX Device Family

Device	M148	F169	F324	F484	F672	F896
EP4CGX15	C7, C8, I7	C6, C7, C8, I7	—	—	—	—
EP4CGX22	—	C6, C7, C8, I7	C6, C7, C8, I7	—	—	—
EP4CGX30	—	C6, C7, C8, I7	C6, C7, C8, I7	C6, C7, C8, I7	—	—
EP4CGX50	—	—	—	C6, C7, C8, I7	C6, C7, C8, I7	—
EP4CGX75	—	—	—	C6, C7, C8, I7	C6, C7, C8, I7	—
EP4CGX110	—	—	—	C7, C8, I7	C7, C8, I7	C7, C8, I7
EP4CGX150	—	—	—	C7, C8, I7	C7, C8, I7	C7, C8, I7

Table 1–6 lists the Cyclone IV E devices speed grades.

Table 1–6. Speed Grades for the Cyclone IV E Device Family (1), (2)

Device	E144	M164	U256	F256	U484	F484	F780
EP4CE6	C8L, C9L, I8L C6, C7, C8, I7, A7	—	I7N	C8L, C9L, I8L C6, C7, C8, I7, A7	—	—	—
EP4CE10	C8L, C9L, I8L C6, C7, C8, I7, A7	—	I7N	C8L, C9L, I8L C6, C7, C8, I7, A7	—	—	—
EP4CE15	C8L, C9L, I8L C6, C7, C8, I7	I7N	I7N	C8L, C9L, I8L C6, C7, C8, I7, A7	—	C8L, C9L, I8L C6, C7, C8, I7, A7	—
EP4CE22	C8L, C9L, I8L C6, C7, C8, I7, A7	—	I7N	C8L, C9L, I8L C6, C7, C8, I7, A7	—	—	—
EP4CE30	—	—	—	—	—	C8L, C9L, I8L C6, C7, C8, I7, A7	C8L, C9L, I8L C6, C7, C8, I7
EP4CE40	—	—	—	—	I7N	C8L, C9L, I8L C6, C7, C8, I7, A7	C8L, C9L, I8L C6, C7, C8, I7
EP4CE55	—	—	—	—	I7N	C8L, C9L, I8L C6, C7, C8, I7	C8L, C9L, I8L C6, C7, C8, I7
EP4CE75	—	—	—	—	I7N	C8L, C9L, I8L C6, C7, C8, I7	C8L, C9L, I8L C6, C7, C8, I7
EP4CE115	—	—	—	—	—	C8L, C9L, I8L C7, C8, I7	C8L, C9L, I8L C7, C8, I7

Notes to Table 1–6:

(1) C8L, C9L, and I8L speed grades are applicable for the 1.0-V core voltage.
(2) C6, C7, C8, I7, and A7 speed grades are applicable for the 1.2-V core voltage.

Cyclone IV Device Family Architecture

This section describes Cyclone IV device architecture and contains the following topics:

- "FPGA Core Fabric"
- "I/O Features"
- "Clock Management"
- "External Memory Interfaces"
- "Configuration"
- "High-Speed Transceivers (Cyclone IV GX Devices Only)"
- "Hard IP for PCI Express (Cyclone IV GX Devices Only)"

FPGA Core Fabric

Cyclone IV devices leverage the same core fabric as the very successful Cyclone series devices. The fabric consists of LEs, made of 4-input look up tables (LUTs), memory blocks, and multipliers.

Each Cyclone IV device M9K memory block provides 9 Kbits of embedded SRAM memory. You can configure the M9K blocks as single port, simple dual port, or true dual port RAM, as well as FIFO buffers or ROM. They can also be configured to implement any of the data widths in Table 1–7.

Table 1–7. M9K Block Data Widths for Cyclone IV Device Family

Mode	Data Width Configurations
Single port or simple dual port	×1, ×2, ×4, ×8/9, ×16/18, and ×32/36
True dual port	×1, ×2, ×4, ×8/9, and ×16/18

The multiplier architecture in Cyclone IV devices is the same as in the existing Cyclone series devices. The embedded multiplier blocks can implement an 18 × 18 or two 9 × 9 multipliers in a single block. Altera offers a complete suite of DSP IP including finite impulse response (FIR), fast Fourier transform (FFT), and numerically controlled oscillator (NCO) functions for use with the multiplier blocks. The Quartus® II design software's DSP Builder tool integrates MathWorks Simulink and MATLAB design environments for a streamlined DSP design flow.

For more information, refer to the *Logic Elements and Logic Array Blocks in Cyclone IV Devices*, *Memory Blocks in Cyclone IV Devices*, and *Embedded Multipliers in Cyclone IV Devices* chapters.

I/O Features

Cyclone IV device I/O supports programmable bus hold, programmable pull-up resistors, programmable delay, programmable drive strength, programmable slew-rate control to optimize signal integrity, and hot socketing. Cyclone IV devices support calibrated on-chip series termination (Rs OCT) or driver impedance matching (Rs) for single-ended I/O standards. In Cyclone IV GX devices, the high-speed transceiver I/Os are located on the left side of the device. The top, bottom, and right sides can implement general-purpose user I/Os.

Table 1–8 lists the I/O standards that Cyclone IV devices support.

Table 1–8. I/O Standards Support for the Cyclone IV Device Family

Type	I/O Standard
Single-Ended I/O	LVTTL, LVCMOS, SSTL, HSTL, PCI, and PCI-X
Differential I/O	SSTL, HSTL, LVPECL, BLVDS, LVDS, mini-LVDS, RSDS, and PPDS

The LVDS SERDES is implemented in the core of the device using logic elements.

For more information, refer to the *I/O Features in Cyclone IV Devices* chapter.

Clock Management

Cyclone IV devices include up to 30 global clock (GCLK) networks and up to eight PLLs with five outputs per PLL to provide robust clock management and synthesis. You can dynamically reconfigure Cyclone IV device PLLs in user mode to change the clock frequency or phase.

Cyclone IV GX devices support two types of PLLs: multipurpose PLLs and general-purpose PLLs:

- Use multipurpose PLLs for clocking the transceiver blocks. You can also use them for general-purpose clocking when they are not used for transceiver clocking.
- Use general purpose PLLs for general-purpose applications in the fabric and periphery, such as external memory interfaces. Some of the general purpose PLLs can support transceiver clocking.

For more information, refer to the *Clock Networks and PLLs in Cyclone IV Devices* chapter.

External Memory Interfaces

Cyclone IV devices support SDR, DDR, DDR2 SDRAM, and QDRII SRAM interfaces on the top, bottom, and right sides of the device. Cyclone IV E devices also support these interfaces on the left side of the device. Interfaces may span two or more sides of the device to allow more flexible board design. The Altera® DDR SDRAM memory interface solution consists of a PHY interface and a memory controller. Altera supplies the PHY IP and you can use it in conjunction with your own custom memory controller or an Altera-provided memory controller. Cyclone IV devices support the use of error correction coding (ECC) bits on DDR and DDR2 SDRAM interfaces.

For more information, refer to the *External Memory Interfaces in Cyclone IV Devices* chapter.

Configuration

Cyclone IV devices use SRAM cells to store configuration data. Configuration data is downloaded to the Cyclone IV device each time the device powers up. Low-cost configuration options include the Altera EPCS family serial flash devices and commodity parallel flash configuration options. These options provide the flexibility for general-purpose applications and the ability to meet specific configuration and wake-up time requirements of the applications.

Table 1–9 lists which configuration schemes are supported by Cyclone IV devices.

Table 1–9. Configuration Schemes for Cyclone IV Device Family

Devices	Supported Configuration Scheme
Cyclone IV GX	AS, PS, JTAG, and FPP (1)
Cyclone IV E	AS, AP, PS, FPP, and JTAG

Note to Table 1–9:

(1) The FPP configuration scheme is only supported by the EP4CGX30F484 and EP4CGX50/75/110/150 devices.

IEEE 1149.6 (AC JTAG) is supported on all transceiver I/O pins. All other pins support IEEE 1149.1 (JTAG) for boundary scan testing.

For more information, refer to the *JTAG Boundary-Scan Testing for Cyclone IV Devices* chapter.

For Cyclone IV GX devices to meet the PCIe 100 ms wake-up time requirement, you must use passive serial (PS) configuration mode for the EP4CGX15/22/30 devices and use fast passive parallel (FPP) configuration mode for the EP4CGX30F484 and EP4CGX50/75/110/150 devices.

For more information, refer to the *Configuration and Remote System Upgrades in Cyclone IV Devices* chapter.

The cyclical redundancy check (CRC) error detection feature during user mode is supported in all Cyclone IV GX devices. For Cyclone IV E devices, this feature is only supported for the devices with the core voltage of 1.2 V.

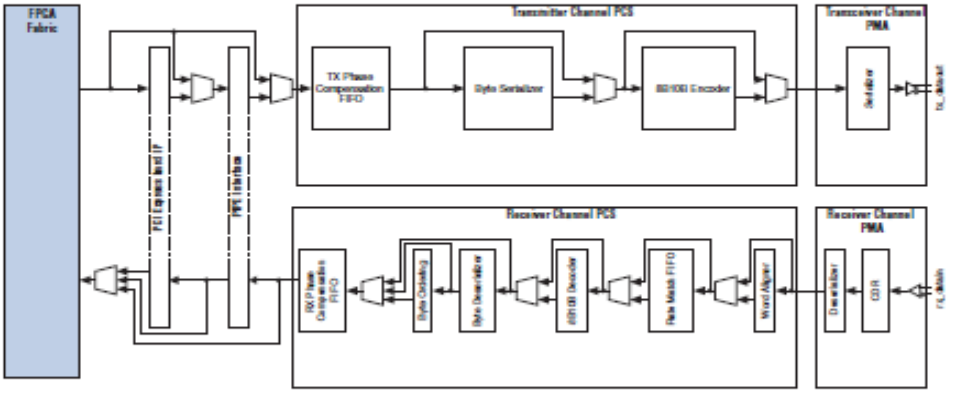
For more information about CRC error detection, refer to the *SEU Mitigation in Cyclone IV Devices* chapter.

High-Speed Transceivers (Cyclone IV GX Devices Only)

Cyclone IV GX devices contain up to eight full duplex high-speed transceivers that can operate independently. These blocks support multiple industry-standard communication protocols, as well as Basic mode, which you can use to implement your own proprietary protocols. Each transceiver channel has its own pre-emphasis and equalization circuitry, which you can set at compile time to optimize signal integrity and reduce bit error rates. Transceiver blocks also support dynamic reconfiguration, allowing you to change data rates and protocols on-the-fly.

Figure 1–1 shows the structure of the Cyclone IV GX transceiver.

Figure 1–1. Transceiver Channel for the Cyclone IV GX Device



For more information, refer to the *Cyclone IV Transceivers Architecture* chapter.

Hard IP for PCI Express (Cyclone IV GX Devices Only)

Cyclone IV GX devices incorporate a single hard IP block for ×1, ×2, or ×4 PCIe (PIPE) in each device. This hard IP block is a complete PCIe (PIPE) protocol solution that implements the PHY-MAC layer, Data Link Layer, and Transaction Layer functionality. The hard IP for the PCIe (PIPE) block supports root-port and end-point configurations. This pre-verified hard IP block reduces risk, design time, timing closure, and verification. You can configure the block with the Quartus II software's PCI Express Compiler, which guides you through the process step by step.

For more information, refer to the *PCI Express Compiler User Guide*.

Reference and Ordering Information

Figure 1–2 shows the ordering codes for Cyclone IV GX devices.

Figure 1–2. Packaging Ordering Information for the Cyclone IV GX Device

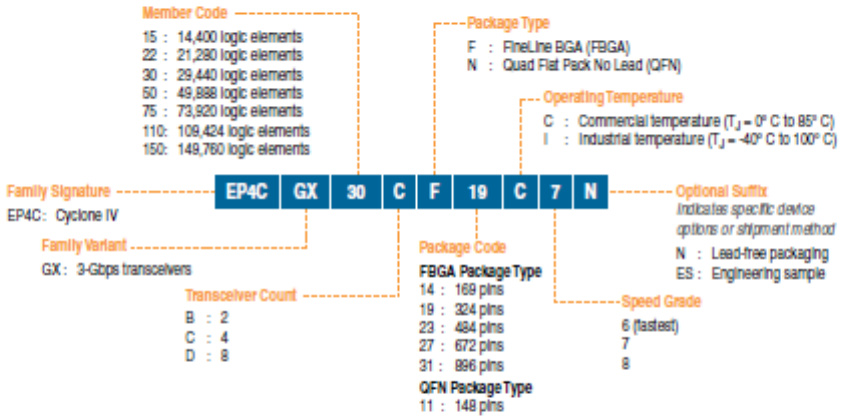
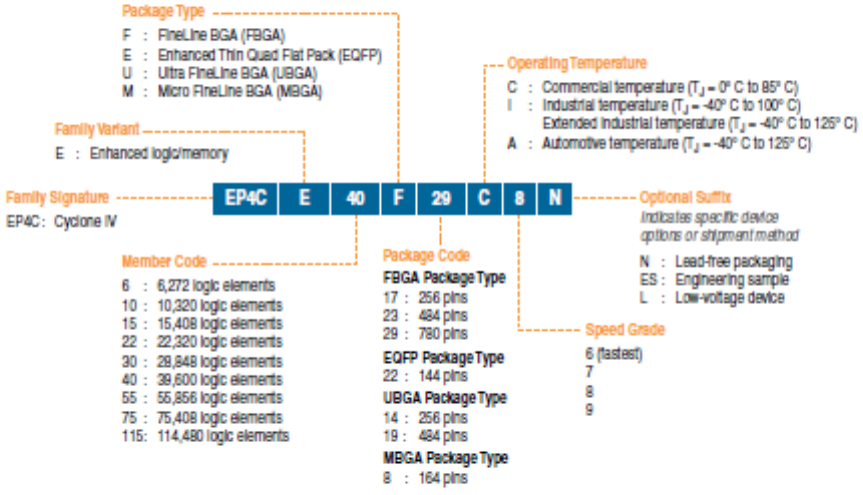


Figure 1–3 shows the ordering codes for Cyclone IV E devices.

Figure 1–3. Packaging Ordering Information for the Cyclone IV E Device



EK2 Virtex 6 FPGA XC6VLX550T Datasheet

Virtex-6 Family Overview

DS150 (v2.4) January 19, 2012 **Product Specification**

General Description

The Virtex®-6 family provides the newest, most advanced features in the FPGA market. Virtex-6 FPGAs are the programmable silicon foundation for Targeted Design Platforms that deliver integrated software and hardware components to enable designers to focus on innovation as soon as their development cycle begins. Using the third-generation ASMBL™ (Advanced Silicon Modular Block) column-based architecture, the Virtex-6 family contains multiple distinct sub-families. This overview covers the devices in the LXT, SXT, and HXT sub-families. Each sub-family contains a different ratio of features to most efficiently address the needs of a wide variety of advanced logic designs. In addition to the high-performance logic fabric, Virtex-6 FPGAs contain many built-in system-level blocks. These features allow logic designers to build the highest levels of performance and functionality into their FPGA-based systems. Built on a 40 nm state-of-the-art copper process technology, Virtex-6 FPGAs are a programmable alternative to custom ASIC technology. Virtex-6 FPGAs offer the best solution for addressing the needs of high-performance logic designers, high-performance DSP designers, and high-performance embedded systems designers with unprecedented logic, DSP, connectivity, and soft microprocessor capabilities.

Summary of Virtex-6 FPGA Features

- Three sub-families:
  - Virtex-6 LXT FPGAs: High-performance logic with advanced serial connectivity
  - Virtex-6 SXT FPGAs: Highest signal processing capability with advanced serial connectivity
  - Virtex-6 HXT FPGAs: Highest bandwidth serial connectivity
- Compatibility across sub-families
  - LXT and SXT devices are footprint compatible in the same package
- Advanced, high-performance FPGA Logic
  - Real 6-input look-up table (LUT) technology
  - Dual LUT5 (5-input LUT) option
  - LUT/dual flip-flop pair for applications requiring rich register mix
  - Improved routing efficiency
  - 64-bit (or two 32-bit) distributed LUT RAM option per 6-input LUT
  - SRL32/dual SRL16 with registered outputs option
- Powerful mixed-mode clock managers (MMCM)
  - MMCM blocks provide zero-delay buffering, frequency synthesis, clock-phase shifting, input-jitter filtering, and phase-matched clock division
- 36-Kb block RAM/FIFOs
  - Dual-port RAM blocks
  - Programmable
    - Dual-port widths up to 36 bits
    - Simple dual-port widths up to 72 bits
  - Enhanced programmable FIFO logic
  - Built-in optional error-correction circuitry
  - Optionally use each block as two independent 18 Kb blocks
- High-performance parallel SelectIO™ technology
  - 1.2 to 2.5V I/O operation
  - Source-synchronous interfacing using ChipSync™ technology
  - Digitally controlled impedance (DCI) active termination
  - Flexible fine-grained I/O banking
  - High-speed memory interface support with integrated write-leveling capability
- Advanced DSP48E1 slices
  - 25 x 18, two's complement multiplier/accumulator
  - Optional pipelining
  - New optional pre-adder to assist filtering applications
  - Optional bitwise logic functionality
  - Dedicated cascade connections
- Flexible configuration options
  - SPI and Parallel Flash interface
  - Multi-bitstream support with dedicated fallback reconfiguration logic
  - Automatic bus width detection
- System Monitor capability on all devices
  - On-chip/off-chip thermal and supply voltage monitoring
  - JTAG access to all monitored quantities
- Integrated interface blocks for PCI Express® designs
  - Compliant to the PCI Express Base Specification 2.0
  - Gen1 (2.5 Gb/s) and Gen2 (5 Gb/s) support with GTX transceivers
  - Endpoint and Root Port capable
  - x1, x2, x4, or x8 lane support per block
- GTX transceivers: up to 6.6 Gb/s
  - Data rates below 480 Mb/s supported by oversampling in FPGA logic.
- GTH transceivers: 2.488 Gb/s to beyond 11 Gb/s
- Integrated 10/100/1000 Mb/s Ethernet MAC block
  - Supports 1000BASE-X PCS/PMA and SGMII using GTX transceivers
  - Supports MII, GMII, and RGMII using SelectIO technology resources
  - 2500Mb/s support available
- 40 nm copper CMOS process technology
- 1.0V core voltage (-1, -2, -3 speed grades only)
- Lower-power 0.9V core voltage option (-1L speed grade only)
- High signal-integrity flip-chip packaging available in standard or Pb-free package options

Virtex-6 FPGA Feature Summary

Table 1: **Virtex-6 FPGA Feature Summary by Device**

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices[2]	Block RAM Blocks			MMCMs[4]	Interface Blocks for PCI Express	Ethernet MACs[5]	Maximum Transceivers		Total I/O Banks[6]	Max User I/O[7]
		Slices[1]	Max Distributed RAM (Kb)		18 Kb[3]	36 Kb	Max (Kb)				GTX	GTH		
XC6VLX75T	74,496	11,640	1,045	288	312	156	5,616	6	1	4	12	0	9	360
XC6VLX130T	128,000	20,000	1,740	480	528	264	9,504	10	2	4	20	0	15	600
XC6VLX195T	199,680	31,200	3,040	640	688	344	12,384	10	2	4	20	0	15	600
XC6VLX240T	241,152	37,680	3,650	768	832	416	14,976	12	2	4	24	0	18	720
XC6VLX365T	364,032	56,880	4,130	576	832	416	14,976	12	2	4	24	0	18	720
XC6VLX550T	549,888	85,920	6,200	864	1,264	632	22,752	18	2	4	36	0	30	1200
XC6VLX760	758,784	118,560	8,280	864	1,440	720	25,920	18	0	0	0	0	30	1200
XC6VSX315T	314,880	49,200	5,090	1,344	1,408	704	25,344	12	2	4	24	0	18	720
XC6VSX475T	476,160	74,400	7,640	2,016	2,128	1,064	38,304	18	2	4	36	0	21	840
XC6VHX250T	251,904	39,360	3,040	576	1,008	504	18,144	12	4	4	48	0	8	320
XC6VHX255T	253,440	39,600	3,050	576	1,032	516	18,576	12	2	2	24	24	12	480
XC6VHX380T	382,464	59,760	4,570	864	1,536	768	27,648	18	4	4	48	24	18	720
XC6VHX565T	566,784	88,560	6,370	864	1,824	912	32,832	18	4	4	48	24	18	720

Notes:

1. Each Virtex-6 FPGA slice contains four LUTs and eight flip-flops; only some slices can use their LUTs as distributed RAM or SRLs.
2. Each DSP48E1 slice contains a 25 x 18 multiplier, an adder, and an accumulator.
3. Block RAMs are fundamentally 36 Kbits in size. Each block can also be used as two independent 18 Kb blocks.
4. Each CMT contains two mixed-mode clock managers (MMCM).
5. This table lists individual Ethernet MACs per device.
6. Does not include configuration Bank 0.
7. This number does not include GTX or GTH transceivers.

Virtex-6 FPGA Device-Package Combinations and Maximum I/Os

Virtex-6 LXT and SXT FPGA package combinations with the maximum available I/Os per package are shown in Table 2.

Table 2: **Virtex-6 LXT and SXT FPGA Device-Package Combinations and Maximum Available I/Os**

Package	FF484 FFG484		FF784 FFG784		FF1156 FFG1156		FF1759 FFG1759		FF1760 FFG1760	
Size (mm)	23 x 23		29 x 29		35 x 35		42.5 x 42.5		42.5 x 42.5	
Device	GTXs	I/O	GTXs	I/O	GTXs	I/O	GTXs	I/O	GTXs	I/O
XC6VLX75T	8	240	12	360						
XC6VLX130T	8	240	12	400	20	600				
XC6VLX195T			12	400	20	600				
XC6VLX240T			12	400	20	600	24	720		
XC6VLX365T					20	600	24	720		
XC6VLX550T							36	840	0	1200
XC6VLX760									0	1200
XC6VSX315T					20	600	24	720		
XC6VSX475T					20	600	36	840		

Notes:
1. Flip-chip packages are also available in Pb-Free versions (FFG).

Virtex-6 HXT FPGA package combinations with the maximum available I/Os per package are shown in Table 3.

Table 3: **Virtex-6 HXT FPGA Device-Package Combinations and Maximum Available I/Os**

Package	FF1154 FFG1154			FF1155 FFG1155			FF1923 FFG1923			FF1924 FFG1924		
Size (mm)	35 x 35			35 x 35			45 x 45			45 x 45		
Device	GTXs	GTHs	I/O	GTXs	GTHs	I/O	GTXs	GTHs	I/O	GTXs	GTHs	I/O
XC6VHX250T	48	0	320									
XC6VHX255T				24	12	440	24	24	480			
XC6VHX380T	48	0	320	24	12	440	40	24	720	48	24	640
XC6VHX565T							40	24	720	48	24	640

Notes:
1. Flip-chip packages are also available in Pb-Free versions (FFG).

Configuration

Virtex-6 FPGAs store their customized configuration in SRAM-type internal latches. The number of configuration bits is between 26 Mb and 177 Mb, depending on device size but independent of the specific user-design implementation, unless compression mode is used. The configuration storage is volatile and must be reloaded whenever the FPGA is powered up. This storage can also be reloaded at any time by pulling the PROGRAM_B pin Low. Several methods and data formats for loading configuration are available, determined by the three mode pins.

Bit-serial configurations can be either master serial mode where the FPGA generates the configuration clock (CCLK) signal, or slave serial mode where the external configuration data source also clocks the FPGA. For byte- and word-wide configurations, master SelectMAP mode generates the CCLK signal while slave SelectMAP mode receives the CCLK signal for the 8-, 16-, or 32-bit-wide transfer. Alternatively, serial-peripheral interface (SPI) and byte-peripheral interface (BPI) modes are used with industry-standard flash memories and are clocked by the CCLK output of the FPGA. JTAG mode uses boundary-scan protocols to load bit-serial configuration data.

The bitstream configuration information is generated by the ISE® software using a program called BitGen. The configuration process typically executes the following sequence:

- Detects power-up (power-on reset) or PROGRAM_B when Low.
- Clears the whole configuration memory.
- Samples the mode pins to determine the configuration mode: master or slave, bit-serial or parallel, or bus width.
- Loads the configuration data starting with the bus-width detection pattern followed by a synchronization word, checks for the proper device code, and ends with a cyclic redundancy check (CRC) of the complete bitstream.
- Start-up executes a user-defined sequence of events: releasing the internal reset (or preset) of flip-flops, optionally waiting for the phase-locked loops (PLLs) to lock and/or the DCI to match, activating the output drivers, and transitions the DONE pin High.

Dynamic Reconfiguration Port

The dynamic reconfiguration port (DRP) gives the system designer easy access to configuration bits and status registers for three block types: 32 locations for each clock tile, 128 locations for the System Monitor, and 128 locations for each serial GTX or GTH transceiver.

The DRP behaves like memory-mapped registers, and can access and modify block-specific configuration bits as well as status and control registers.

Encryption, Readback, and Partial Reconfiguration

As a special option, the bitstream can be AES-encrypted to prevent unauthorized copying of the design. The Virtex-6 FPGA performs the decryption using the internally stored 256-bit key that can use battery backup or alternative non-volatile storage.

Most configuration data can be read back without affecting the system's operation. Typically, configuration is an all-or-nothing operation, but the Virtex-6 FPGA also supports partial reconfiguration. When applicable in certain designs, partial reconfiguration can greatly improve the versatility of the FPGA. It is even possible to reconfigure a portion of the FPGA while the rest of the logic remains active i.e., active partial reconfiguration.

CLBs, Slices, and LUTs

The look-up tables (LUTs) in Virtex-6 FPGAs can be configured as either one 6-input LUT (64-bit ROMs) with one output, or as two 5-input LUTs (32-bit ROMs) with separate outputs but common addresses or logic inputs. Each LUT output can optionally be registered in a flip-flop. Four such LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a configurable logic block (CLB). Four flip-flops per slice (one per LUT) can optionally be configured as latches. In that case, the remaining four flip-flops in that slice must remain unused.

Between 25–50% of all slices can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers (SRL32) or as two SRL16s. Modern synthesis tools take advantage of these highly efficient logic, arithmetic, and memory features. Expert designers can also instantiate them.

Clock Management

Each Virtex-6 FPGA has up to nine clock management tiles (CMTs), each consisting of two mixed-mode clock managers (MMCMs), which are PLL based.

Phase-Locked Loop

The MMCM can serve as a frequency synthesizer for a wider range of frequencies and as a jitter filter for incoming clocks. The heart of the MMCM is a voltage-controlled oscillator (VCO) with a frequency from 600 MHz up to 1600 MHz, spanning more than one octave. There are three sets of programmable frequency dividers (D, M, and O).

The pre-divider D (programmable by configuration) reduces the input frequency and feeds one input of the traditional PLL phase/frequency comparator. The feedback divider (programmable by configuration) acts as a multiplier because it divides the VCO output frequency before feeding the other input of the phase comparator. D and M must be chosen appropriately to keep the VCO within its specified frequency range.

The VCO has eight equally-spaced output phases (0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°). Each can be selected to drive one of the seven output dividers, O0 to O6 (each programmable by configuration to divide by any integer from 1 to 128).

MMCM Programmable Features

The MMCM has three input-jitter filter options: low bandwidth, high bandwidth, or optimized mode. Low-bandwidth mode has the best jitter attenuation but not the smallest phase offset. High-bandwidth mode has the best phase offset, but not the best jitter attenuation. Optimized mode allows the tools to find the best setting.

The MMCM can have a fractional counter in either the feedback path (acting as a multiplier) or in one output path. Fractional counters allow non-integer increments of 1/8 and can thus increase frequency synthesis capabilities by a factor of 8.

The MMCM can also provide fixed or dynamic phase shift in small increments that depend on the VCO frequency. At 600 MHz the phase-shift timing increment is 30 ps; at 1600 MHz, it is 11.5 ps.

Clock Distribution

Each Virtex-6 FPGA provides five different types of clock lines (BUFG, BUFR, BUFIO, BUFH, and the high-performance clock) to address the different clocking requirements of high fanout, short propagation delay, and extremely low skew.

Global Clock Lines

In each Virtex-6 FPGA, 32 global-clock lines have the highest fanout and can reach every flip-flop clock, clock enable, set/reset, as well as many logic inputs. There are 12 global clock lines within any region. Global clock lines can be driven by global clock buffers, which can also perform glitchless clock multiplexing and the clock enable function. Global clocks are often driven from the CMT, which can completely eliminate the basic clock distribution delay.

Regional Clocks

Regional clocks can drive all clock destinations in their region as well as the region above and below. A region is defined as any area that is 40 I/O and 40 CLB high and half the chip wide. Virtex-6 FPGAs have between 6 and 18 regions. There are 6 regional clock tracks in every region. Each regional clock buffer can be driven from either of four clock-capable input pins, and its frequency can optionally be divided by any integer from 1 to 8.

I/O Clocks

I/O clocks are especially fast and serve only I/O logic and serializer/deserializer (SerDes) circuits, as described in the I/O Logic section. Virtex-6 devices have a high-performance direct connection from the MMCM to the I/O directly for low-jitter, high-performance interfaces.

Block RAM

Every Virtex-6 FPGA has between 156 and 1064 dual-port block RAMs, each storing 36 Kbits. Each block RAM has two completely independent ports that share nothing but the stored data.

Synchronous Operation

Each memory access, read and write, is controlled by the clock. All inputs, data, address, clock enables, and write enables are registered. *Nothing happens without a clock*. The input address is always clocked, retaining data until the next operation. An optional output data pipeline register allows higher clock rates at the cost of an extra cycle of latency.

During a write operation, the data output can reflect either the previously stored data, the newly written data, or remain unchanged.

Programmable Data Width

- Each port can be configured as 32K × 1, 16K × 2, 8K × 4, 4K × 9 (or 8), 2K × 18 (or 16), 1K × 36 (or 32), or 512 x 72 (or 64). The two ports can have different aspect ratios, without any constraints.
- Each block RAM can be divided into two completely independent 18 Kb block RAMs that can each be configured to any aspect ratio from 16K x 1 to 512 x 36. Everything described previously for the full 36 Kb block RAM also applies to each of the smaller 18 Kb block RAMs.
- In 18 Kb block RAMs, only simple dual-port mode can provide data width of >36 bits. In this mode, one port is dedicated to read and the other port is dedicated to write operation. In SDP mode one side (read or write) can be variable while the other is fixed to 32/36 or 64/72. There is no read output during write. The dual-port 36 Kb RAM both sides can be of variable width.
- Two adjacent 36 Kb block RAMs can be configured as one cascaded 64K × 1 dual-port RAM without any additional logic.

Error Detection and Correction

Each 64 bit-wide block RAM can generate, store, and utilize eight additional Hamming-code bits, and perform single-bit error correction and double-bit error detection (ECC) during the read process. The ECC logic can also be used when writing to, or reading from external 64/72-wide memories. This works in simple dual-port mode and does not support read-during-write.

FIFO Controller

The built-in FIFO controller for single-clock (synchronous) or dual-clock (asynchronous or multirate) operation increments the internal addresses and provides four handshaking flags: full, empty, almost full, and almost empty. The almost full and almost empty flags are freely programmable. Similar to the block RAM, the FIFO width and depth are programmable, but the write and read ports always have identical width. First-word fall-through mode presents the first-written word on the data output even before the first read operation. After the first word has been read, there is no difference between this mode and the standard mode.

Digital Signal Processing—DSP48E1 Slice

DSP applications use many binary multipliers and accumulators, best implemented in dedicated DSP slices. All Virtex-6 FPGAs have many dedicated, full-custom, low-power DSP slices combining high speed with small size, while retaining system design flexibility.

Each DSP48E1 slice fundamentally consists of a dedicated 25 × 18 bit two's complement multiplier and a 48-bit accumulator, both capable of operating at 600 MHz. The multiplier can be dynamically bypassed, and two 48-bit inputs can feed a single-instruction-multiple-data (SIMD) arithmetic unit (dual 24-bit add/subtract/accumulate or quad 12-bit add/subtract/accumulate), or a logic unit that can generate any one of 10 different logic functions of the two operands.

The DSP48E1 includes an additional pre-adder, typically used in symmetrical filters. This new pre-adder improves performance in densely packed designs and reduces the logic slice count by up to 50%.

The DSP48E1 slice provides extensive pipelining and extension capabilities that enhance speed and efficiency of many applications, even beyond digital signal processing, such as wide dynamic bus shifters, memory address generators, wide bus multiplexers, and memory-mapped I/O register files. The accumulator can also be used as a synchronous up/down counter. The multiplier can perform logic functions (AND, OR) and barrel shifting.

Input/Output

The number of I/O pins varies from 240 to 1200 depending on device and package size. Each I/O pin is configurable and can comply with a large number of standards, using up to 2.5V. The *Virtex-6 FPGA SelectIO Resources User Guide* describes the I/O compatibilities of the various I/O options. With the exception of supply pins and a few dedicated configuration pins, all other package pins have the same I/O capabilities, constrained only by certain banking rules.

All I/O pins are organized in banks, with 40 pins per bank. Each bank has one common V_{CCO} output supply-voltage pin, which also powers certain input buffers. Some single-ended input buffers require an externally applied reference voltage (V_{REF}). There are two V_{REF} pins per bank (except configuration bank 0). A single bank can have only one V_{REF} voltage value.

I/O Electrical Characteristics

Single-ended outputs use a conventional CMOS push/pull output structure driving High towards V_{CCO} or Low towards ground, and can be put into high-Z state. The system designer can specify the slew rate and the output strength. The input is always active but is usually ignored while the output is active. Each pin can optionally have a weak pull-up or a weak pull-down resistor.

Any signal pin pair can be configured as differential input pair or output pair. Differential input pin pairs can optionally be terminated with a 100Ωinternal resistor. All Virtex-6 devices support differential standards beyond LVDS: HT, RSDS, BLVDS, differential SSTL, and differential HSTL.

Digitally Controlled Impedance

Digitally controlled impedance (DCI) can control the output drive impedance (series termination) or can provide parallel termination of input signals to V_{CCO}, or split (Thevenin) termination to $V_{CCO}/2$. DCI uses two pins per bank as reference pins, but one such pair can also control multiple banks. VRN must be resistively pulled to V_{CCO}, while VRP must be resistively connected to ground. The resistor must be either 1× or 2× the characteristic trace impedance, typically close to 50Ω

I/O Logic

Input and Output Delay

This section describes the available logic resources connected to the I/O interfaces. All inputs and outputs can be configured as either combinatorial or registered. Double data rate (DDR) is supported by all inputs and outputs. Any input or output can be individually delayed by up to 32 increments of ~78 ps each. This is implemented as IODELAY. The number of delay steps can be set by configuration and can also be incremented or decremented while in use.

For using either IODELAY, the system designer must instantiate the IODELAY control block and clock it with a frequency close to 200 MHz. Each 32-tap total IODELAY is controlled by that frequency, thus unaffected by temperature, supply voltage, and processing variations.

ISERDES and OSERDES

Many applications combine high-speed bit-serial I/O with slower parallel operation inside the device. This requires a serializer and deserializer (SerDes) inside the I/O structure. Each input has access to its own deserializer (serial-to-parallel converter) with programmable parallel width of 2, 3, 4, 5, 6, 7, 8, or 10 bits. Each output has access to its own serializer (parallel-to-serial converter) with programmable parallel width of up to 8 bits wide for single data rate (SDR), or up to 10 bits wide for double data rate (DDR).

System Monitor

Every Virtex-6 FPGA contains a System Monitor circuit providing thermal and power supply status information. Sensor outputs are digitized by a 10-bit 200kSPS analog-to-digital converter (ADC). This fully tested and specified ADC can also be used to digitize up to 17 external analog input channels. The System Monitor ADC utilizes an on-chip reference circuit thereby eliminating the need for any external active components. On-chip temperature and power supplies are monitored with a measurement accuracy of ±4°C and ±1% respectively.

By default the System Monitor continuously digitizes the output of all on-chip sensors. The most recent measurement results together with maximum and minimum readings are stored in dedicated registers for access at any time through the DRP or JTAG interfaces. User defined alarm thresholds can automatically indicate over temperature events and unacceptable power supply variation. A specified limit (for example: 125°C) can be used to initiate an automatic power down.

The System Monitor does not require explicit instantiation in a design. Once the appropriate power supply connections are made, measurement data can be accessed at any time, even pre-configuration or during power down, through the JTAG test access port (TAP).

Low-Power Gigabit Transceivers

Ultra-fast serial data transmission between ICs, over the backplane, or over longer distances is becoming increasingly popular and important. It requires specialized dedicated on-chip circuitry and differential I/O capable of coping with the signal integrity issues at these high data rates.

All but one Virtex-6 device has between 8 to 72 gigabit transceiver circuits. Each GTX transceiver is a combined transmitter and receiver capable of operating at a data rate between 480 Mb/s and 6.6 Gb/s. Lower data rates can be achieved using FPGA logic-based oversampling. Each GTH transceiver is a combined transmitter and receiver capable of operating at a rate between 2.488 Gb/s and 11.18 Gb/s. The GTX transmitter and receiver are independent circuits that use separate PLLs to multiply the reference frequency input by certain programmable numbers between 4 and 25, to become the bit-serial data clock. The GTH transceiver is a purpose-built design for 10 Gb/s rates and shares a single high-performance PLL between four transmitter and receiver circuits. Each GTX and GTH transceiver has a large number of user-definable features and parameters. All of these can be defined during device configuration, and many can also be modified during operation.

Transmitter

The GTX transmitter is fundamentally a parallel-to-serial converter with a conversion ratio of 8, 10, 16, 20, 32, or 40. The GTH transmitter offers bit widths of 16, 20, 32, 40, 64, or 80 to allow additional timing margin for high-performance designs. These transmitter outputs drive the PC board with a single-channel differential current-mode logic (CML) output signal.

TXOUTCLK is the appropriately divided serial data clock and can be used directly to register the parallel data coming from the internal logic. The incoming parallel data is fed through a small FIFO and can optionally be modified with the 8B/10B, 64B/66B, or the 64B/67B (GTX only) algorithm to guarantee a sufficient number of transitions. The bit-serial output signal drives two package pins with complementary CML signals. This output signal pair has programmable signal swing as well as programmable pre-emphasis to compensate for PC board losses and other interconnect characteristics.

Receiver

The receiver is fundamentally a serial-to-parallel converter, changing the incoming bit serial differential signal into a parallel stream of words, each 8, 10, 16, 20, 32, or 40 bits wide. The GTH transceiver offers 16, 20, 32, 40, 64, and 80 bit widths to allow greater timing margin. The receiver takes the incoming differential data stream, feeds it through a programmable equalizer (to compensate for PC board and other interconnect characteristics), and uses the F_{REF} input to initiate clock recognition. There is no need for a separate clock line. The data pattern uses non-return-to-zero (NRZ) encoding and optionally guarantees sufficient data transitions by using the selected encoding scheme. Parallel data is then transferred into the FPGA logic using the RXUSRCLK clock. The serial-to-parallel conversion ratio for GTX transceivers can be 8, 10, 16, 20, 32, or 40. The serial-to-parallel conversion ratio for GTH transceivers can be 16, 20, 32, 40, 64, or 80 for GTH.

Out-of-Band Signaling

The GTX transceivers provide Out-of-Band (OOB) signaling, often used to send low-speed signals from the transmitter to the receiver, while high-speed serial data transmission is not active, typically when the link is in a power-down state or has not been initialized. This benefits PCI Express and SATA/SAS applications.

Integrated Interface Blocks for PCI Express Designs

The PCI Express standard is a packet-based, point-to-point serial interface standard. The differential signal transmission uses an embedded clock, which eliminates the clock-to-data skew problems of traditional wide parallel buses.

The PCI Express Base Specification Revision 2.0 is backwards compatible with Revision 1.1 and defines a configurable raw data rate of 2.5 Gb/s, or 5.0 Gb/s per lane in each direction. To scale bandwidth, the specification allows multiple lanes to be joined to form a larger link between PCI Express devices.

All Virtex-6 devices (except the XC6VLX760) include at least one integrated interface block for PCI Express technology that can be configured as an Endpoint or Root Port, compliant to the PCI Express Base Specification Revision 2.0. The Root Port can be used to build the basis for a compatible Root Complex, to allow custom FPGA-FPGA communication via the PCI Express protocol, and to attach ASSP Endpoint devices such as Fibre Channel HBAs to the FPGA.

This block is highly configurable to system design requirements and can operate 1, 2, 4, or 8 lanes at the 2.5 Gb/s data rate and the 5.0 Gb/s data rate. For high-performance applications, advanced buffering techniques of the block offer a flexible maximum payload size of up to 1024 bytes. The integrated block interfaces to the GTX transceivers for serial connectivity, and to block RAMs for data buffering. Combined, these elements implement the Physical Layer, Data Link Layer, and Transaction Layer of the PCI Express protocol.

Xilinx provides a light-weight, configurable, easy-to-use LogiCORE™ wrapper that ties the various building blocks (the integrated block for PCI Express, the GTX transceivers, block RAM, and clocking resources) into an Endpoint or Root Port solution. The system designer has control over many configurable parameters: lane width, maximum payload size, FPGA logic interface speeds, reference clock frequency, and base address register decoding and filtering.

More information and documentation on solutions for PCI Express designs can be found at:

http://www.xilinx.com/technology/protocols/pciexpress.htm

10/100/1000 Mb/s Ethernet Controller (2,500 Mb/s Supported)

An integrated Tri-mode Ethernet MAC (TEMAC) block is easily connected to the FPGA logic, the GTX transceivers, and the SelectIO resources. This TEMAC block saves logic resources and design effort. All of the Virtex-6 devices (except the XC6VLX760) have four TEMAC blocks, implementing the link layer of the OSI protocol stack. The CORE Generator™ software GUI helps to configure flexible interfaces to GTX transceiver or SelectIO technology, to the FPGA logic, and to a microprocessor (when required). The TEMAC is designed to the IEEE Std 802.3-2005 specification. 2,500 Mb/s support is also available.

Virtex-6 FPGA Ordering Information

Table 4 shows the speed and temperature grades available in the different Virtex-6 devices. Some devices might not be available in every speed and temperature grade.

Table 4: **Virtex-6 FPGAs Speed Grade and Temperature Ranges**

Device Family	Speed Grade and Temperature Range		
	Commercial (C) 0°C to +85°C	Extended (E) 0°C to +100°C	Industrial (I) -40°C to +100°C
Virtex-6 LXT	-3, -2, -1, -1L	-2	-2, -1, -1L
Virtex-6 SXT	-3, -2, -1, -1L	-2	-2, -1, -1L
Virtex-6 HXT	-3, -2, -1	-2	-2, -1

The Virtex-6 FPGA ordering information shown in Figure 1 applies to all packages including Pb-Free.

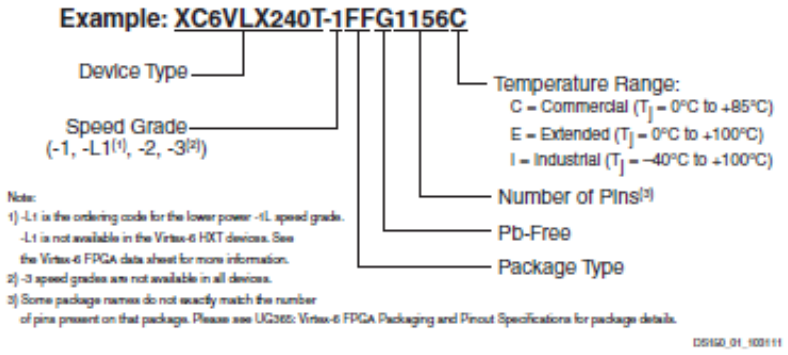


Figure 1: **Virtex-6 FPGA Ordering Information**

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
02/02/09	1.0	Initial Xilinx release.
05/05/09	1.1	Added the FF1156 package for both the XC6VSX315T and XC6VSX475T devices in Table 2, page 3. Updated the PCI Express design discussion on page 9 to remove the LogiCORE wrapper (<100 LUT) description and clarify 8 lanes at the 5.0 Gb/s data rate. Clerical edits to Global Clock Lines and 10/100/1000 Mb/s Ethernet Controller (2,500 Mb/s Supported) sections. Overall clarifications made in text.
06/24/09	1.2	Added ordering information and FPGA documentation sections.
09/16/09	2.0	Added Virtex-6 HXT family information. Updated number to 26 Mb in Configuration section.
11/06/09	2.1	Clarified distributed RAM features on page 1. Updated CLB slice number for the XC6VHX565T in Table 1. Updated compliance to the PCI Express Base Specification Revision 2.0. Updated Integrated Interface Blocks for PCI Express Designs section with link to documentation.
01/28/10	2.2	In Table 1, there are two Ethernet MACs in the XC6VHX255T. Under Clock Management, page 5, revised the VCO frequency minimum to 600 MHz which also revised the phase-shift timing increment. Updated GTX transceivers operating data rate range to 6.6 Gb/s. Changed GTX PLL input reference clock frequency divider.
03/24/11	2.3	Changed document classification to Preliminary Product Specification from Advance Product Specification. Updated Figure 1.
01/19/12	2.4	Changed document classification to Product Specification from Preliminary Product Specification. Updated Configuration, CLBs, Slices, and LUTs, Low-Power Gigabit Transceivers, and Virtex-6 FPGA Ordering Information (including Figure 1).

Notice of Disclaimer

ÖZGEÇMİŞ

Erdinç Avaroğlu, 1979 yılında Adana'da doğdu. Lise öğrenimini Adana Çağrı Bey Lisesinde, lisans öğrenimide 1997-2001 yılları arasında Mersin Üniversitesi Bilgisayar Mühendisliği Bölümü'nde tamamladı. 2005-2007 yıllarında İnönü Üniversitesi Elektrik Elektronik Mühendisliğinde Elektronik İmza konulu yüksek lisans eğitimini tamamladı. 2004 yılında İnönü Üniversitesinde Mühendis olarak göreve başlamıştır ve halen bu görevine devam etmektedir.