Int. J. Adv. Eng. Pure Sci. 2020, 32(3): 219-228
DOI: 10.7240/jeps.614156

RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

Yüksek Hızlı FPGA ile LFSR Tabanlı 32-Bit Kayan Noktalı Yeni Bir Sözde Rastgele Sayı Üreteci Tasarımı

A New Pseudo Random Number Generator Design with LFSR Based 32-Bit Floating Point with High-Speed FPGA

Serkan DERELİ[1]

[1] *Sakarya Uygulamalı Bilimler Üniversitesi, Bilgisayar Teknolojileri ve Programlama Bölümü, Adapazarı, Sakarya*

Öz
Bu çalışmada FPGA temelli IEEE 754 kayan noktalı sayı standardına uygun sözde rasgele sayı üreteci tasarımı gerçekleştirilmiştir. Gerçekleştirilen tasarım doğrusal geri beslemeli kayan yazmaç (LFSR) yöntemini kullanarak 32-bit uzunluğunda ve [0, 1] arasında ondalık sayılar üretmektedir. 32-bitlik bu sayılara bakıldığında en değerli 4-bitin (28-31) tamamında aynı değeri alması nedeniyle işlemler 28-bit üzerinden gerçekleştirilmiştir. Bu çalışmada bahsi geçen tasarımın en önemli özelliği üretilen rasgele sayının doğrudan kayan noktalı bir değer olmasıdır. Bu nedenle üretilen rasgele sayının [0, 1] aralığında olmaması durumunda sayı üretme işlemi tekrar baştan başlatıldığından dolayı her sayının işlem zamanı farklı olabilmektedir. VHDL tasarım dili ile oluşturulan sayısal devre Vivado arabiriminde simülasyon ile test edildikten sonra Xilinx Nexys 4 DDR FPGA aygıtı ile gerçeklenmiştir. Sonuçlar üretilen rasgele sayıların dağılımı ve üretilme süreleri bakımından analiz edilmiştir.
Anahtar kelimeler: Rastgele Sayı Üreteci, Doğrusal Geri Beslemeli Öteleyen Kaydedici, VHDL, FPGA, Kayan Noktalı Sayı

Abstract
In this study, pseudo random number generator design which is based on FPGA based IEEE 754 floating point number standard has been realized. The realized design generates floating-point numbers of 32-bit length and between 0 to 1 using the linear feedback floating register (LFSR) method. Looking at these 32-bit numbers, operations are performed on 28-bits since the most valuable 4-bits (28-31) all have the same value. The most important feature of the design mentioned in this study is that the generated random number is a direct floating point value. Therefore, if the generated random number is not in the range of 0 to 1, the computation time of each number may be different since the number generation process is restarted from the beginning. The digital circuit created by VHDL design language was tested by simulation on Vivado interface and implemented with Xilinx Nexys 4 DDR FPGA device. The results were analyzed in terms of distribution and generation times of random numbers generated.
Keywords: Random Number Generator, Lineer Feedback Shift Register, VHDL, FPGA, Floating Point Number

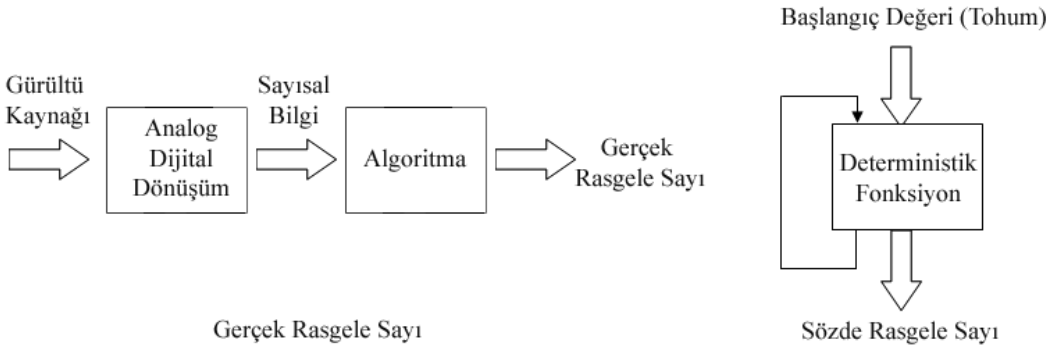
I. GİRİŞ

Oyun programlama [1], kriptoloji [2] ve yapay zeka optimizasyon teknikleri [3] başta olmak üzere pek çok hesaplamalı bilim dalında yaygın bir şekilde kullanılan rasgele sayılar araştırma dünyasında büyük bir öneme sahiptir. Rasgele sayılar, aralarında belli bir ilişki bulunmayan ve tahmin edilmesi zor olan sayılar olduklarından dolayı kriptolojide anahtar olarak, oyun programlamada ekrana gelecek sahnenin oluşturulması için ve yapay zekâda çözüm uzayından rasgele başlangıç çözümü seçmek için kullanılmaktadır.

Rasgele sayı üreteçlerinin tarihi gelişimine bakıldığında ilk olarak Knuth'un yazdığı "The Art of Programming" isimli eseri ile karşılaşılacaktır [4]. 1983 yılına gelindiğinde Ripley küçük kişisel bilgisayar kullanıcıları için yetersiz olan rasgele sayı üreteçlerinin daha etkili hale gelebilmesi için üstel ve normal dağılıma sahip diziler üretmiştir [5]. 1990 yılında James, Monte Carlo hesaplamaları için çeşitli rasgele sayı üreteçleri üzerine çalışmalar gerçekleştirmiştir [6]. Aynı yıl Lagarias sayı teorisini esas alan çalışmaları neticesinde rasgele sayı üreteçlerini kriptolojide yaygınca kullanılan gizli anahtarlar üretmek için kullanmıştır [7].

Sorumlu yazar: Serkan DERELİ, Tel: 0 264 6160492, e-posta: dereli@subu.edu.tr
Gönderilme: 02.09.2019, **Düzenleme:** 27.02.2020, **Kabul:** 04.06.2020

Literatürde "gerçek rasgele sayı üreteci (True Random Number Generator)" ve "sözde rasgele sayı üreteci (Pseudo Random Number Generator)" olmak üzere rasgele sayı üreteçleri iki gruba ayrılmıştır. Belli bir matematiksel fonksiyon yardımıyla yani deterministik olarak rasgele sayı üreteçlerine "sözde rasgele sayı üreteci" adı verilmektedir. Bu üreteçlerde bir başlangıç değeri vardır ve bu değer üzerinden rasgele sayılar üretilir. Başlangıç değeri değiştirilerek farklı rasgele sayı dizileri elde edilmektedir [8]. Gerçek rasgele sayı üreteçleri ise gerçek fiziksel değerler baz alınarak sayı üretme işlemidir ve bunun için gerçek yaşamda var olan gürültü kaynakları rasgele sayılar üretmek için fiziksel değer kaynağı olarak kullanılmaktadırlar. Termal ve atmosferik gürültü, nükleer bozulma, ses, video, EEG (Elektroensefalografi), ECG (Electrocardiogram) gibi gürültü kaynakları rasgele sayı üretiminde fiziksel değer olarak kullanılmaktadır [9].



Şekil 1. Gerçek ve Sözde rastgele sayı üreteçleri blok şeması

Genç ve Tuncer yaptıkları çalışmalarında insan hareketlerinden elde ettikleri gürültü değerleriyle gerçek rasgele sayı üretici tasarımı gerçekleştirmişlerdir. Bunun için GPS ve ivme sensörünü kullanarak elde ettikleri verileri fiziksel büyüklük olarak tasarladıkları sisteme aktarmışlar ve bu sayede normal dağılıma sahip rasgele sayılar üretmişlerdir [10]. Khalique ve arkadaşları sistem saati ve iki büyük asal sayı parametrelerini kullanarak sözde rasgele sayı üreteci tasarımı gerçekleştirmişlerdir. Sistem saati ile başlangıç değerini gerçek bir değerden elde etmek suretiyle üretilen rasgele sayının tahmin edilemez bir şekle kavuşmasını, iki büyük asal sayı parametresi ile de üretilen sayıların rassallığını sağlamışlardır [11]. Özkaynak yaptığı çalışmada rasgele sayı üreteçlerine genel bakış yaparak tarihçesi ve hali hazırda kullanım şekilleri üzerine araştırma gerçekleştirmiştir. Ayrıca kriptolojik uygulamalarda rasgele sayı üreteçlerinin ihtiyaç duyduğu gereksinimler ile bu gereksinimleri karşılayacak mimariler üzerine bir araştırma yapmıştır [12]. Aydın ve Dalkılıç yaptıkları çalışmada nesnelerin interneti gibi cihazların ihtiyaç duyduğu rasgele sayıları üretmek için bir algoritma geliştirmişlerdir. Algoritmada öncelikle C dilinde rasgele beş adet 16-bitlik sayılar üreterek başlamaktadır. Devamında bu sayıların ilk dördü ikişerli olarak gruplandırılıp doğrusal geri beslemeli kayan yazmaçtan geçirildikten sonra birleştirilerek 32-bitlik sayılar elde edilmiştir. Sonrasında ise elde edilen bu 32-bitlik iki adet sayı ile ilk başta elde edilip kullanılmayan 16-bitlik sayılar bir fonksiyon dâhilinde birleştirilerek 32-bitlik yeni bir sayı üretilmektedir [13]. Falih yaptığı çalışmada basit bir metot olarak isimlendirdiği sözde rasgele sayı üreteci tasarlamıştır. Bu üreteç, rasgele sayı üretmek için hem doğrusal geri beslemeli öteleyen kaydediciyi hem de kaos teorisini kullanmaktadır. Kaos teorisi üretilen sayıların doğrusallığını ve tekrarlanabilirliğini ortadan kaldırmak için kullanılmıştır [14]. Masoodi ve arkadaşları, akış şifreleme yönteminde kullanılan doğrusal geri beslemeli öteleyen kaydedici (LFSR) tabanlı rasgele sayı üretme mekanizmalarını ve onların uygulamalarını kapsamlı bir şekilde analiz etmiştir [15].

Panda ve arkadaşları doğrusal geri beslemeli öteleyen kaydedici tabanlı 8, 16 ve 32-bit rasgele sayılar üreten devreleri VHDL ile geliştirmişler ve FPGA aygıtında test işlemini gerçekleştirmişlerdir. Üç farklı uzunlukta rasgele sayı üreten devreler performans ve rassallık açısından karşılaştırılmıştır. Nihayetinde FPGA'da geliştirilen devreler mantıksal elemanlarla oluşturulduğundan bit sayısı arttıkça eleman sayısı da arttığından dolayı en düşük bit sayısına sahip devrenin rasgele sayı üretme süresi daha az ancak en uzun bit sayısına sahip devrenin ise periyodu daha uzun olduğundan dolayı rassallığı çok daha gelişmiştir [16]. Rezk ve arkadaşları FPGA tabanlı kaotik özellikli sözde rasgele sayı üretici tasarımı gerçekleştirmişlerdir. Kaotik özelliğini rasgele sayılara dâhil eden sistemde Lorez ve Lü kaotik sistemleri beraber kullanılmıştır. Bunun için dört farklı etki elemanının biri Lorenz sistemine göre oluşturulurken diğerleri Lü sistemine göre oluşturulmuştur [17]. Stanchieri ve arkadaşları gürültü kaynağı jitter ve metastability olan gerçek rasgele sayı üretecini gerçekleştirerek karakterizasyonun raporunu

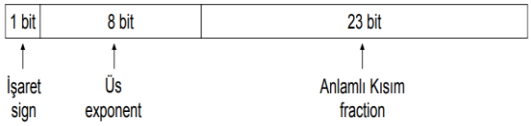
ortaya koyan bir çalışma gerçekleştirmişlerdir. Geliştirilen rasgele sayı üretecini gerçekleştirdikleri FPGA tasarımını mantık elemanları, LUT (Look Up Table) tablosu, çalışma frekansı ve flip-flop sayısı bakımından analiz etmişlerdir [18]. Koyuncu ve Özcerit yaptıkları çalışmada kaotik osilatör gürültüleri ile elde ettikleri gerçek rasgele sayı üretecini sayısal, Pspice ve FPGA ile ayrı ayrı modelleyerek her üç platformda da gerçekleştirmişler ve aradaki farkları ortaya koymuşlardır. Bunlar arasında FPGA ile modelledikleri sistemin diğer platformlara göre daha fazla avantaj ortaya koyduğunu göstermişlerdir [19].

Bu çalışmada ise yüksek hızlı işlemler için önemli bir platform olan FPGA tabanlı donanımsal rasgele sayı üreteci "0" ve "1" arasında 32-bit kayan noktalı sayılar üretmek için kullanılmıştır. Özellikle pek çok optimizasyon algoritmasında kullanılan bu sayılar aynı zamanda kullanıldığı algoritmanın en temel parametreleri arasındadır. Çalışmanın bundan sonraki bölümü üç alt bölüme ayrılmıştır. Birinci alt bölümde 0 ile 1 arasındaki 32-bitten oluşan sayıların analizi yapılarak geliştirilen devrenin optimize edilmesi için yollar aranmıştır. İkinci alt bölümde geliştirilen sayısal devrenin blok şeması verilerek detaylı bir şekilde sunumu gerçekleştirilmiştir. En son üçüncü alt bölümde ise elde edilen simülasyon ve gerçekleme sonuçları verilmiştir.

II. MATERYAL VE YÖNTEM

2.1. "0" ile "1" Arasındaki Kayan Noktalı Sayıların Analizi

Bu çalışmada 1985 yılında kabul edilen IEEE 754 kayan noktalı sayı standardını temel alan bir sistem tasarımı gerçekleştirilmiştir. Bu standarda göre ondalıklı sayılar tek hassasiyetli (single precision) 32-bit ve çift hassasiyetli (double precision) 64-bit uzunlukta olmak üzere iki farklı gösterime sahiptir [20]. IEEE 754 standardına göre kayan noktalı bir sayı işaret (1-bit), üs (8-bit) ve anlamlı kısım (23-bit) olmak üzere üç bölümden oluşmaktadır [21].



Şekil 2. 32-bit kayan noktalı sayı formatı bölümleri

Şekil 2'de göründüğü üzere sayı tek hassasiyetli kayan noktalı sayılar 32-bitten oluşmaktadır ve en değerli biti (MSB) işaret biti olarak isimlendirilir. Eğer sayı negatifse bu bit değeri "1" aksi halde bu bit değeri "0" dır. Sayısal bir sistemin bilgiyi temsil ederken kolaylık sağlaması adına ikilik tabana çevrildikten sonra yapılan kaydırma sayısı "Üs (exponent)" olarak isimlendirilir. 32-bit gösterimde sayının en son oluşan ondalık kısmına ise "anlamlı kısım (fraction)" adı verilir [22].

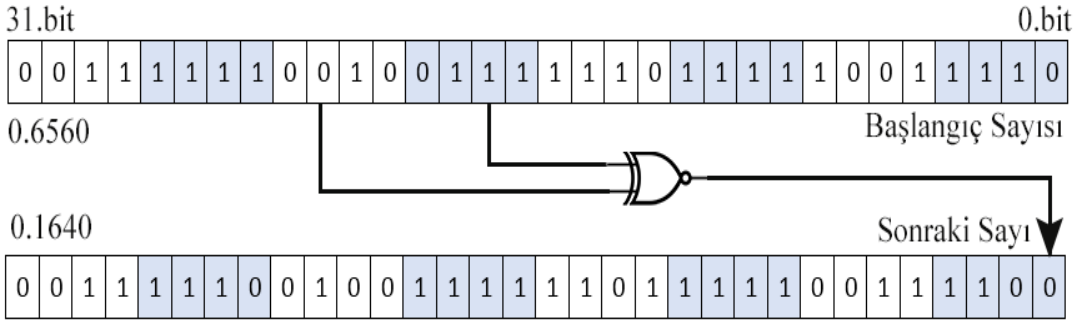
Tablo 1. 0 ile 1 arasındaki sayıların 10'luk ve 16'lık tabandaki gösterimi

10'luk Taban	16'lık Taban	10'luk Taban	16'lık Taban	10'luk Taban	16'lık Taban
0.1	3DCC CCCD	0.4	3ECC CCCD	0.7	3F33 3333
0.2	3E4C CCCD	0.5	3F00 0000	0.8	3F4C CCCD
0.3	3E99 999A	0.6	3F19 999A	0.9	3F66 6666

Tablo 1'de verilen "0" ile "1" arasındaki kayan noktalı sayılar incelendiğinde tamamında soldan ilk basamağının "3" olduğu açıkça görünmektedir. Dolayısıyla 2'lik taban olarak bakıldığında 28-31 arası bitler sabit yani "0011" olmalıdır. 27-24 arası bitlere bakıldığında ise bir adet "D - 1101", üç adet "E - 1110" ve beş adet "F - 1111" olmak üzere on altı değerden sadece üç değer aldıkları görünmektedir. Ayrıca değerlerin "0.1" den daha küçük ve "1.0" dan daha büyük olmaması gerekmektedir. O nedenle, literatürdeki diğer çalışmalardan farklı olarak gerçekleştirilecek tasarımda bu koşullar dikkate alındığında MSB 8-bit dışında kalan ilk 24-bit LFSR tekniğinin uygulanabileceği bitlerdir. 27-24 arası bitler normal dağılım yapılarak 0-3 arası bitlerin almış olduğu değere göre üç değer arasından seçilerek LFSR işleminden geçirilen 24-bitin MSB tarafına ilave edilmektedir. Geriye kalan 4-bit ise "3 - 0011" olarak en son aşamada 28-bitin MSB tarafına ilave edilerek 32-bitlik kayan noktalı sayı elde edilmektedir.

2.2. Doğrusal Geri Beslemeli Öteleyen Kaydedici (LFSR)

Doğrusal geri beslemeli öteleyen kaydedici, iyi istatistiki özellikleri, geniş tekrarlama periyodu ve yapısının basit ama kullanışlı olması nedeniyle güvenlik algoritmalarındaki anahtar oluşumu ve oturum yönetiminden en basit rasgele sayı oluşturma uygulamalarına kadar çok geniş kullanım alanına sahiptir [23, 24]. Bu çalışmanın konusu olan "0" ile "1" arasından rasgele sayı seçme işlemi ise yapay zeka optimizasyon tekniklerinde çözüme ulaşırken hesaplanan değere esneklik katmak amacıyla sıklıkla kullanılmaktadır ve bu algoritmaların en önemli parametreleri arasındadır [25].



Şekil 3. Bu çalışmada kullanılan LFSR tekniği

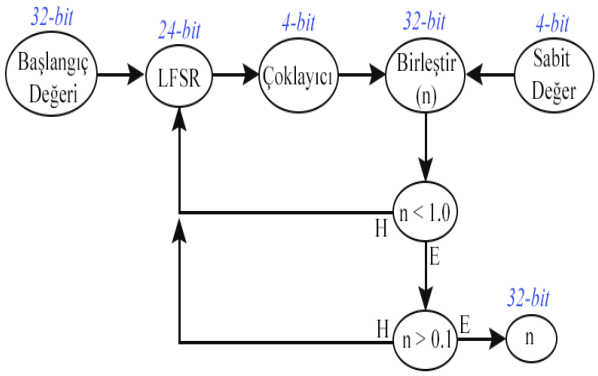
Şekil 3'te bu çalışmada kullanılan doğrusal geri beslemeli öteleyen kaydedicinin çalışma şekli görünmektedir. 31-28 arası bitler sabit kalmakta, 27-24 arası bitlerde 0-3 arası bitlerin değerine göre "D - 1101", "E - 1110" veya "F - 1111" değerlerinden biri seçilmektedir. 3-0 arası bitlerin hangi değerlerine karşılık 27-24 arası bitlerin alabileceği değerler Tablo 2'de gösterilmiştir. 22'nci ve 17'nci bitlerin EXNOR işlemine tabi tutularak elde edilen değerin 0'ncı bite ve bu bitten itibaren 24 bitin sola kaymasıyla yeni rasgele sayı elde edilmektedir. Şekil 3'te başlangıçtaki sayı 0,6560 ve bir sonraki aşamada ortaya çıkan sayı ise 0,1640 olarak görünmektedir. Bu şekilde de göründüğü gibi gerçekleştirilen tasarım sözde rasgele sayı üreteci olduğundan dolayı elde edilecek rasgele sayı dizisi başlangıç değerine bağlıdır.

Tablo 2. 27-24 arası bitlerin alabileceği değerler

3:0 bit değeri	27:24 bit değeri	3:0 bit değeri	27:24 bit değeri	3:0 bit değeri	27:24 bit değeri	3:0 bit değeri	27:24 bit değeri
0000	1101 – D	0100	1110 – E	1000	1111 – F	1100	1111 – F
0001	1101 – D	0101	1110 – E	1001	1111 – F	1101	1111 – F
0010	1110 – E	0110	1110 – E	1010	1111 – F	1110	1111 – F
0011	1110 – E	0111	1111 – F	1101	1111 – F	1111	1111 – F

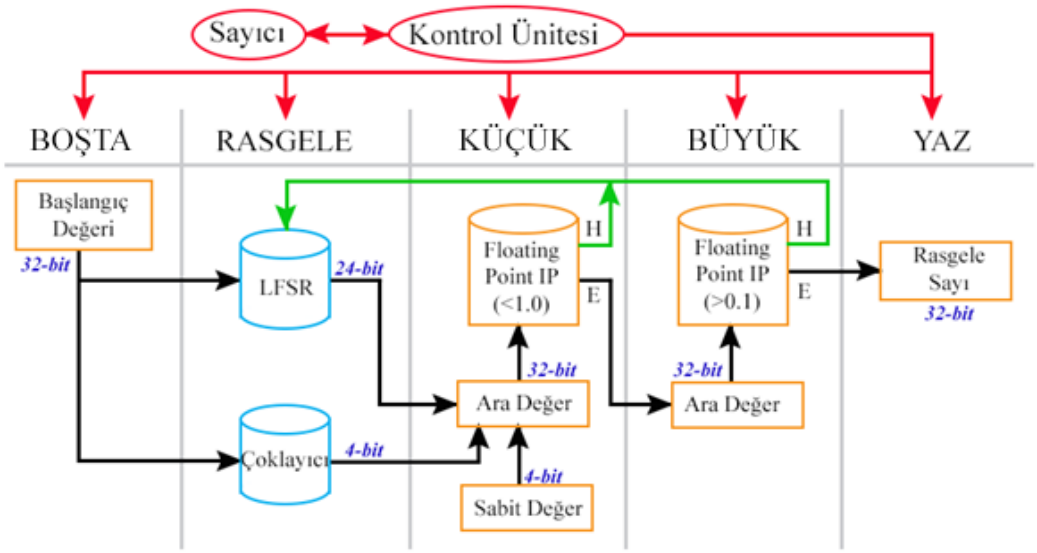
2.3. Önerilen Rasgele Kayan Noktalı Sayı Üreteci

Bu çalışmada önerilen sözde rasgele sayı üreteci, hem doğrusal geri beslemeli öteleyen kaydedici (LFSR) işlemi hem de çoklayıcı (Multiplexer) işlemini birlikte harmanlayarak gerçekleştirmektedir. Ancak sözde rasgele sayı üreteçlerinin özelliklerinden biri olan başlangıç değeri ile sayı üretmektedir.



Şekil 4. Önerilen sözde rasgele kayan noktalı sayı üreteci

Şekil 4'te MF-PRNG ismi verilen ve bu çalışmada önerilen rasgele sayı üretecinin akış şeması görünmektedir. Başlangıç değeri ile algoritma başlamaktadır ve bu değer 32-bit uzunluğundadır. Başlangıç değerinin LSB 24-biti doğrusal lineer öteleyen kaydedici işleminden geçerken MSB ilk 4-biti sabit bırakılıp sonraki 4-biti ise LSB ilk 4-bitin değerine göre bir çoklayıcı sayesinde "D", "E" ve "F" değerlerinden herhangi birini seçmektedir. Böylece son noktada bu bitler birleştirilerek yine 32-bitlik bir değer elde edilmektedir. Ancak bu değerin "0" ile "1" arasında olup olmadığı belli olmadığından geliştirilen algoritmada elde edilen 32-bitlik değerin öncelikle 1'den küçük olup olmadığı sorgulanmaktadır. Eğer bu değer 1'den küçük değilse tekrar başa yani LFSR aşamasına geçilmektedir. Yok, eğer 1'den küçükse bu durumda sayının 0,1'den büyük olup olmadığının sorgulandığı aşamaya geçilmektedir. Burada da benzer şekilde sayı 0,1'den büyük değilse algoritma LFSR aşamasına tekrar dönmektedir. Aksi halde elde edilen değer "0" ile "1" arasında olduğundan dolayı çıkış işlemi gerçekleşmektedir.

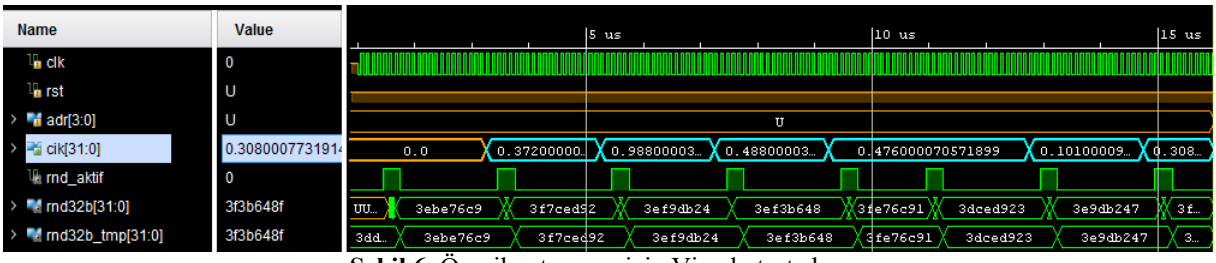


Şekil 5. Önerilen rasgele sayı üretecine ait FPGA blok şeması

Şekil 5'te, önerilen sözde rasgele sayı üretecinin FPGA'da gerçekleştirilen blok şeması görünmektedir. FPGA'da geliştirilen sayısal devre ardışıl bir çalışma özelliğine sahip olduğundan dolayı sonlu durum makineleri kullanılarak VHDL ile geliştirilmiştir. Şekil 5'te sonlu durumlar makinesi için kullanılan durumlar görünmektedir. BOSTA durumu tasarıma başlangıç bölümüdür ve bu tasarımda rasgele üretilecek sayı dizisine başlangıç için 32-bit uzunluğunda sabit bir değerin atandığı durumdur. RASGELE durumunda başlangıç değerinden gelen sayının ilk 24-biti LFSR işleminden geçirilmektedir. 0-3 arası bitler ise çoklayıcı işleminden geçerek 24-27 arası bitlerin değerini elde etmektedir. Sonunda bu değerler, sabit değer ile birlikte bir ara değerde 32-bit olarak birleştirilerek "Floating IP Core" ile "1,0" değerinden küçük olup olmadığı sorgulanmaktadır. Sonrasında ise yine başka bir "Floating IP Core" ile "0,1" sayısından büyük olup olmadığı sorgulanmaktadır. En son aşamada ise 32-bit uzunluğunda üretilen sözde rasgele kayan noktalı sayı çıkışa aktarılmaktadır. Burada bahsi geçen "Floating IP Core" üretici firma tarafından geliştirilen ve kayan noktalı sayılarla işlem yapabilmeyi kolaylaştıran bir alt devredir.

2.4. Simülasyon ve Gerçekleme Sonuçları

VHDL tasarım dili ile FPGA aygıtı için geliştirilen rasgele sayı üreteci sayısal devresi Xilinx firması tarafından geliştirilen Vivado IDE arabirimi ile tasarlanmış ve yine bu simülasyon aracı ile de test edilmiştir. Test işleminin başarıyla tamamlanmasının ardından yine Xilinx firması tarafından geliştirilen Nexys 4 DDR FPGA aygıtında gerçeklenmiştir.



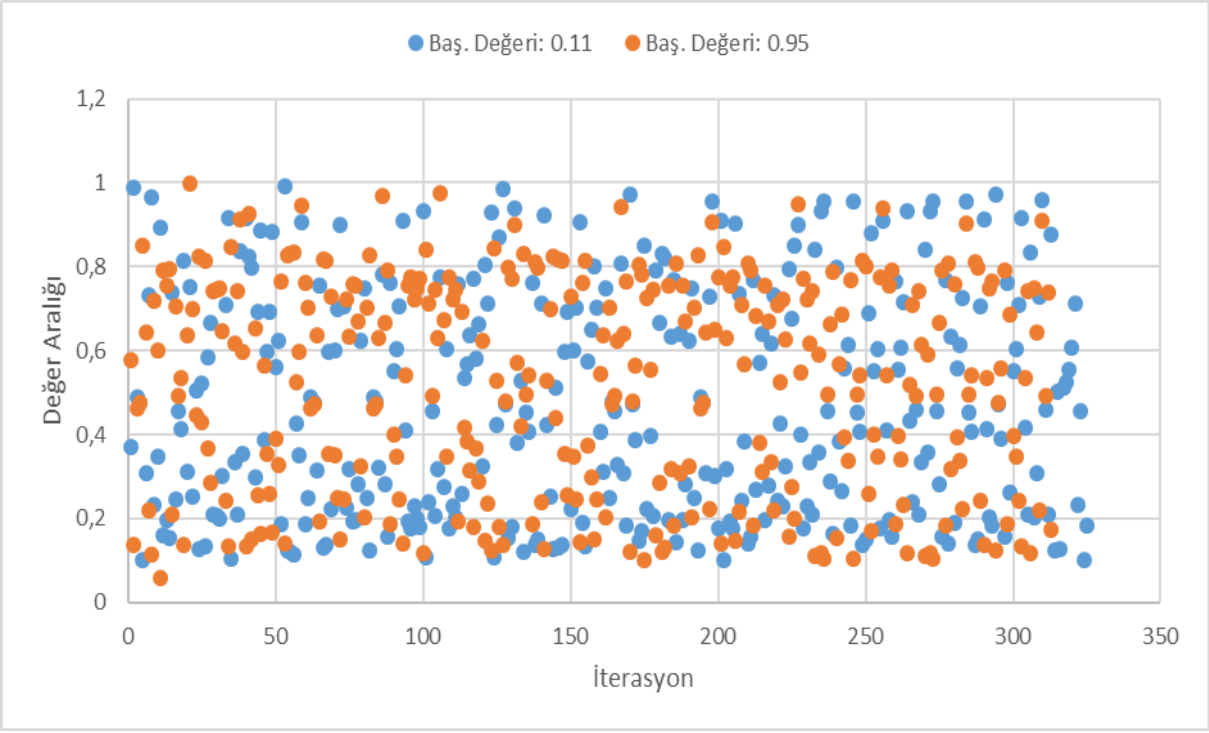
Şekil 6. Önerilen tasarım için Vivado test ekranı

Şekil 6'da ki simülasyon ekranında önerilen rasgele sayı üreteci için geliştirilen devrenin üretmiş olduğu sayılar buz mavisi renkte görünmektedir. Bu sayılardan bazıları kısa saat darbesinde bazıları ise daha uzun saat darbelerinde üretilmiştir. Bunun sebebi "0" ile "1" arasında olmayan sayıların çıkışa gönderilmeyip tekrar sayı üretilmesi için başa dönmesi nedeniyledir. Bu durumla ilgili detaylı veriler Çizelge 3'de verilmektedir.

Tablo 3. Geliştirilen FPGA tabanlı devrenin "0" ile "1" arasında ürettiği örnek sayılar ve üretme süreleri

Test-1. Başlangıç Değeri: 0.11				Test-2. Başlangıç Değeri: 0.95			
Test No	**Saat Darbesi**	**16'lık Taban**	**Kayan Nokta**	**Test No**	**Saat Darbesi**	**16'lık Taban**	**Kayan Nokta**
1	38	3ebe76c9	0.3720	1	53	3e9d14e7	0.3068
2	20	3f7ced92	0.9880	2	20	3f3a29cf	0.7272
3	20	3ef9db24	0.4880	3	20	3f74539e	0.9544
4	35	3ef3b648	0.4760	4	115	3ea73c07	0.3266
5	20	3dced923	0.1010	5	20	3f4e780e	0.8065
6	20	3e9db247	0.3080	6	35	3f39e03b	0.7260
7	20	3f3b648f	0.7320	7	20	3f73c077	0.9521
8	65	3f76c91e	0.9640	8	65	3f3c077e	0.7334
9	35	3e6c91ea	0.2310	9	20	3f780efd	0.9689
10	20	3eb247a8	0.3482	10	50	3ec077e8	0.3759

Tablo 3'te FPGA aygıtı için geliştirilen sayısal devrenin başlangıç değerleri farklı iki testte arka arkaya ürettiği 10 rasgele sayı ve her bir sayı için devrenin kullandığı saat darbesi verilmektedir. Normal bir akışta rasgele sayı üretmek için tasarlanan devre 20 saat darbesi kullanmaktadır. En fazla saat darbesi birinci testte 65, ikinci testte ise 115 saat darbesidir. Birinci testin başlangıç değerinden sonra ürettiği sayı için 38 saat darbesi kullanılırken aynı durum ikinci testte 53 saat darbesi olarak görünmektedir. Birinci testte 10 adet rasgele sayı üretmek için kullanılan toplam süre 293 saat darbesi iken benzer durum için ikinci test 418 saat darbesi süresi kullanmıştır. Üretilen rasgele sayılardaki saat darbelerinin bu derece farklı olmasının nedeni üretilen sıradaki sayının [0.1, 1.0] değer aralığında olmamasıdır. Örneğin 0.95 başlangıç sayısı ile başlatılan ikinci testte dördüncü sayı 115 saat darbesi sonucunda elde edilebilmiştir. Çünkü bu arada 1.0 sayısından daha büyük dört, 0.1 değerinden ise daha küçük altı farklı sayı üretilmiştir. Bu durumda üretilen bu sayılar kabul edilmediğinden işlem algoritmanın en başından yeniden başlatılmaktadır.



Şekil 7. Gerçekleştirilen tasarım vasıtasıyla 500 test ile üretilen rasgele sayıların dağılımı

Şekil 7'de bu çalışma kapsamında FPGA aygıtı için geliştirilen rasgele sayı üreteci tasarımı simülasyon ekranında üretilen 500 adet "0" ve "1" arasında rasgele kayan noktalı sayının iki farklı başlangıç değerine sahip test için dağılım grafiği görünmektedir. 0.11 başlangıç sayısıyla üretilen 500 adet sayıdan 325 tanesi [0, 1] aralığında iken 175 tanesi bu aralığın dışında kalmıştır. 0.95 başlangıç sayısıyla üretilen sayılardan ise 313 tanesi [0, 1] aralığında iken 187 tanesi bu aralığın dışında kalmıştır. O nedenle Şekil 7'de ki dağılım grafiğinde aralık dışında elde edilen sayılar görünmemektedir. Grafikte de göründüğü üzere sayıların normal bir şekilde 0 ile 1 arasında dağıldığı, dolayısıyla da herhangi bir devre, sistemde veya algoritmada rahatlıkla kullanılabilecek düzeyde olduğu açıktır. Test işleminde elde edilen 500 adet rasgele sayı ile ilgili istatistiki bilgiler Tablo 4'te verilmiştir. Burada değer aralığının başından (0.11), ortasından (0.55) ve sonundan (0.95) olmak üzere seçilen değerler ile toplamda 500 adet rasgele sayı üretilmiştir. "Toplam Saat Darbesi" sütunu bu 500 adet sayının üretilmesi için geçen süreyi, "Ortalama Saat Darbesi" ise 500 sayıdan [0, 1] değer aralığında olanların elde edilebilmesi için kullanılan saat darbesinin ortalama değerini göstermektedir. "Aralıktaki Sayı Adeti" sütunu 500 sayıdan [0, 1] değer aralığında olanların sayısını, "Aralık Dışındaki Sayı Adeti" sütunu ise 500 sayıdan [0, 1] aralığı dışında olanların sayısını göstermektedir. [0, 1] değer aralığında üretilip çıkışa aktarılan en fazla işlem süresini "Mak. Saat Darbesi" sütunu en az kullanılan işlem süresini ise "Min. Saat Darbesi" sütunu ifade etmektedir. Tablo 4'te göründüğü üzere ortaya çıkan değerler birbirine yakın olarak kabul edilebilir ölçüdedir. Hiçbir testte olağan dışı bir fark görünmemektedir.

Tablo 4. Test ile elde edilen 500'er adet rasgele sayı ile ilgili istatistikler

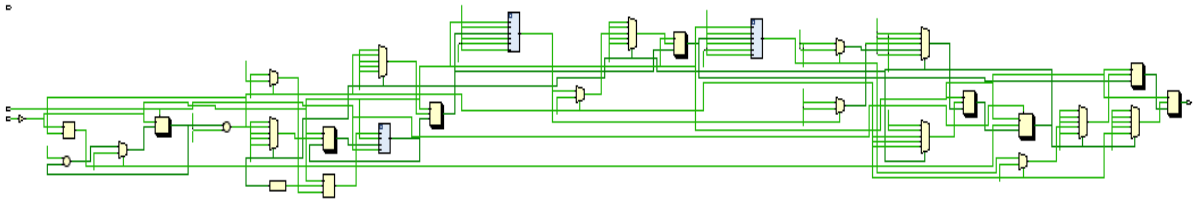
Test	Baş. Değeri	İterasyon	Aralıktaki Sayı Adeti	Aralık Harici Sayı Adeti	Toplam Saat Darbesi	Mak. Saat Darbesi	Min. Saat Darbesi	Ortalama Saat Darbesi
1	0.11	500	325	175	11387	95	20	35
2	0.55	500	296	204	11233	134	20	38
3	0.95	500	313	187	11352	115	20	36

Şekil 8'de FPGA tabanlı gerçekleştirilen sayısal devrenin RTL şeması yani devrenin mantıksal elemanlardan oluşan hali görünmektedir. Bu şekildeki sarı renkli olanlar mantıksal elemanlar olup Vivado arabiriminde "Leaf Cell" olarak adlandırılırken mavi renkli olanlar ise alt devrelerdir. Bu devrenin sentezlenmesi sonucunda elde edilen devrenin büyüklüğünü de ifade eden devre elemanları sayısı Tablo 5'te görünmektedir.

Tablo 5. Devrenin büyüklüğünü ifade eden elemanların sayısı

Devre	LUT Sayısı	FF Sayısı
Ana Devre (main)	22	86
RASGELE SAYI	29	113
SAYIDAN_BUYUK_MU	47	75
SAYIDAN_KUCUK_MU	44	70
TOPLAM	142	344

Şekil 8'de mavi renkte görünen elemanlar "RASGELE_SAYI", "SAYIDAN_BÜYÜK_MÜ" ve "SAYIDAN_KÜÇÜK_MÜ" etiketleriyle oluşturulan alt devrelerdir. "RASGELE_SAYI" alt devresi isminden de anlaşılacağı üzere LFSR tabanlı sözde rasgele sayıları üreten alt devredir ve 40 adet mantıksal hücre ile 68 ara bağlantıdan oluşmaktadır. Ara bağlantı olarak bahsi geçen elemanlar ise Şekil 8'de yeşil renkte görünen ve mantıksal blok içerisinde oluşturulan öğeleri birbirine bağlayan hatlardır. "SAYIDAN_BÜYÜK_MÜ" alt devresi ise Xilinx firması tarafından tasarlanıp tasarımcıların projelerinde kayan noktalı sayılarla olan işlemlerinde kullanmaları için kütüphaneye eklenmiş bir "IP-CORE" dur ve üretilen rasgele sayının "0,1" den büyük olup olmadığını kontrol etmektedir. "SAYIDAN_KÜÇÜK_MÜ" alt devresi de benzer şekilde bir "IP_CORE" dur ve üretilen rasgele sayının "1.0" dan küçük olup olmadığını kontrol eder. Aslında hem "SAYIDAN_BÜYÜK_MÜ" hem de "SAYIDAN_KÜÇÜK_MÜ" alt devreleri aynı "Floating IP-Core" yapısında olup sadece çıktıları farklı seçilmektedir.



Şekil 8. Geliştirilen sayısal devrenin RTL şeması

Tablo 6, bu çalışmaya benzer şekilde farklı yöntemlerle, farklı uzunlukta ve farklı tipte üretilen sözde rasgele sayılar temelinde gerçekleştirilen literatür karşılaştırmasını vermektedir. Açıkça görünmektedir ki, FPGA'da ondalık sayıları üretmek için daha fazla kaynak tüketilmektedir. Elbette bu tüketimde kullanılan teknikte büyük önem kazanmaktadır. Zira [1 ve 2] nolu çalışmalar sözde rasgele sayı üretimi için kaotik temelli bir teknik kullanırken bu çalışma kapsamında LFSR tekniğinden istifade edilmiştir. Dolayısıyla yapısının basit olması nedeniyle LFSR daha az sayıda kaynak tüketimini beraberinde getirmiştir.

Tablo 6. Literatürdeki farklı çalışmalara ait devre büyüklükleri karşılaştırması

No	Çalışma	Üretilen Sayı Türü	Üretilen Sayı	LUT Sayısı	FF Sayısı
1	Khanzadi ve arkadaşları [26]	Ondalıklı (Real)	32-bit	428	688
2	De la Fraga ve arkadaşları [27]	Ondalıklı (Fixed)	32-bit	575	-
3	Cerda ve arkadaşları [28]	Tamsayı (Integer)	8-bit	32	-
4	Justin ve arkadaşları [29]	Tamsayı (Integer)	16-bit	47	52
5	Tian ve Benkrid [30]	Tamsayı (Integer)	32-bit	213	193
6	Rezk ve arkadaşları [31]	Tamsayı (Integer)	32-bit	276	-

III. TARTIŞMA VE SONUÇ

Bu çalışmada, doğrusal geri beslemeli öteleyen kaydedici temelli sözde rasgele sayı üreteci tasarımı gerçekleştirilmiştir. Bu sözde rasgele sayı üretecinin en belirgin farkı ürettiği sayıların "0" ve "1" arasında 32-bitlik hassasiyete sahip kayan noktalı sayılar olmasıdır. O nedenle üretilen sayıların 0.1'den büyük ve 1.0'dan küçük olması sağlanmıştır. Geliştirilen rasgele sayı üreteci VHDL tasarım dili ile kodlanarak sayısal devresi oluşturulmuş ve testleri Xilinx firmasına ait Vivado IDE simülasyon yazılımında ve Artix7 işlemcili Nexys 4 DDR aygıtında gerçekleştirilmiştir. Çalışma kapsamında başlangıç değerleri değiştirilerek üç farklı test gerçekleştirilmiştir. Bu başlangıç değerleri özellikle [0, 1] değer aralığının başından, ortasından ve sonundan seçilerek bu durumun üretilen doğru rasgele sayı adetine, toplam ve ortalama işlem sürelerine etkisi incelenmiştir. Her üç başlangıç değerleriyle üretilen toplam rasgele sayı adetinin üretilme süreleri (Toplam Saat Darbesi) ve geçerli ([0, 1] aralığında) sayıların ortalama üretim süreleri birbirine oldukça yakındır. Üretilen sayıların işlem sürelerinin birbirinden farklı olmalarının sebebi her üretilen sayının değil 0 ile 1 aralığında olan sayıların kabul edilerek devre çıkışına aktarılmasıdır. Eğer sayı belirlenen aralıkta değilse tekrar başa dallanma sağlanır. Bu da işlem süresinin artmasına sebebiyet vermektedir. Bu çalışma ile [0, 1] aralığında üretilen sayıların bu aralıkta normal bir şekilde dağıldığı herhangi bir değerde kümelenmediği açıkça görünmektedir. Üretilen sayıların dağılımının normal olması, algoritma yapısının basit ve anlaşılır olması ve günümüzde hız aygıtı olarak öne çıkan FPGA'da uygulanmış ve kullanılabilir düzeyde olması nedeniyle bu sözde rasgele sayı üreteci araştırmacılar tafından [0, 1] aralığında rasgele sayıya ihtiyaç duyulan herhangi bir sistemde rahatlıkla kullanılabilir.

KAYNAKLAR

[1] Hendrik, M., Meijer, S., Velden, J.V.D., & Iosup, A., (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications*.

[2] Özkaynak. F., (2014). Cryptographically secure random number generator with chaotic additional input. *Nonlinear Dynamics*, 78, 2015-2020.

[3] Çavuşlu, M., Karakuzu, C., & Şahin, S., (2010). Parçacık Sürü Optimizasyonu Algoritması ile Yapay Sinir Ağı Eğitiminin FPGA Üzerinde Donanımsal Gerçeklenmesi. *Politeknik Dergisi*, 13, 83-92.

[4] Knuth, D., (1997). The Art of Programming, 3th Edition, Addison Wesley Lognman, Boston, USA.

[5] Ripley, B., (1983). Computer Generation of

Random Variables: A Tutorial. *International Statistical Review*, 51, 301-3019.

[6] James, F., (1990). A review of pseudorandom number generators. *Computer Physics Communications*, 60, 329-344.

[7] Lagarias, J., (1990). Pseudorandom Number Generators in Cryptography and Number Theory. *Advanced Mathematics*, 42, 115-143.

[8] Akhshani, A., Akhavan, A., Mobaraki, A., Lim, S., & Hassan, Z., (2014). Pseudo random number generator based on quantum chaotic map. Communications in Nonlinear Science and Numerical Simulation, 19, 101-111.

[9] Koyuncu, İ., Ozcerit, A., Pehlivan, İ., & Avaroglu, E., (2014). Design and implementation of chaos based true random number generator on FPGA. Signal Processing and Communications Applications Conference.

[10] Genç, Y., & Tuncer, S., (2019). İnsan Hareketleri Tabanlı Gerçek Rasgele Sayı Üretimi. BEÜ Fen Bilimleri Dergisi, 8, 261-269.

[11] Khalique, A., Lone, A., & Ashraf, S., (2015). A Novel Unpredictable Temporal based Pseudo Random Number Generator. International Journal of Computer Applications, 117, 975-987.

[12] Özkaynak, F., (2015). Kriptolojik Rasgele Sayı Üreteçleri. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi, 8, 37-44.

[13] Aydın, Ö., & Dalkılıç, G., (2016). Nesnelerin İnterneti için Sözderastsal Sayı Üreteci: Birleştirilmiş Doğrusal Geri Beslemeli Öteleyici Saklayıcı. Akıllı Teknoloji & Akıllı Yönetim, İzmir, Gülermat Matbaacılık, 121-129.

[14] Falih, S., (2016). A Pseudorandom Binary Generator Based on Chaotic Linear Feedback Shift Register. Iraq J. Electrical and Electronic Engineering, 12, 155-160.

[15] Masoodi, F., & Alam, S., (2012). An Analysis of Linear Feedback Shift Registers in Stream Ciphers. International Journal of Computer Applications, 46, 46-49.

[16] Panda, A., Rajput, P., & Shukla, B., (2012). FPGA implementation of 8, 16 and 32 bit LFSR with maximum length feedback polynomial using VHDL. International Conference on Communication Systems and Network Technologies, Rajkot.

[17] Rezk, A., Madian, A., Radwan, A., & Soliman, A., (2019). Reconfigurable chaotic pseudo random number generator based on FPGA. AEU - International Journal of Electronics and Communications, 98, 174-180.

[18] Stanchieri, G., Marcellis, A., Palange, E., & Faccio, M., (2019). A true random number generator architecture based on a reduced number of FPGA primitives. AEU - International Journal of Electronics and Communications, 105, 15-23.

[19] Koyuncu, İ., & Özcerit, A., (2017). The design and realization of a new high speed FPGA-based chaotic true random number generator. Computers & Electrical Engineering, 58, 203-214.

[20] George, A., Sharma, R., & Rao, S., (2019). IEEE 754 floating-point addition for neuromorphic architecture. Neurocomputing,366, 74-85.

[21] Melquiond, G., (2012). Floating-point arithmetic in the Coq system. Information and Computation, 216, 14-23.

[22] Parte, R., & Jain, J., (2015). Analysis of Effects of using Exponent Adders in IEEE-754 Multiplier by VHDL. International Conference on Circuits, Power and Computing Technologies, Nagercoil.

[23] Lin, Y., Wang, F., & Liu, B., (2018). Random number generators for large-scale parallel Monte Carlo simulations on FPGA. Journal of Computational Physics, 360, 93-103.

[24] George, S. N., & Pattathil, D. P. (2014). A secure LFSR based random measurement matrix for compressive sensing. Sensing and Imaging, 15(1), 85.

[25] Dereli, S., & Köker, R., (2018). IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator. Sigma J Eng Nat Sci, 36, 77-85.

[26] Khanzadi, H., Eshghi, M., & Borujeni, S. E., (2015). Design and FPGA Implementation of a Pseudo Random Bit Generator Using Chaotic Maps. ETE Technical Review, 32, 304-310.

[27] de la Fraga, L. G., Torres-Pérez, E., Tlelo-Cuautle, E., & Mancillas-López, C. (2017). Hardware implementation of pseudo-random number generators based on chaotic maps. Nonlinear Dynamics, 90(3), 1661-1670.

[28] Cerda, J. C., Martinez, C. D., Comer, J. M., Hoe, D. H. K., (2012). An efficient FPGA random number generator using LFSRs and cellular automata. IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS).

[29] Justin, R., Mathew, B. K., Abe, S., (2015). FPGA Implementation of High Quality Random Number Generator Using LUT Based Shift Registers," Procedia Technology, 24, 1155-1162.

[30] Tian, X., & Benkrid, K., (2009). Mersenne Twister Random Number Generation on FPGA, CPU and GPU. NASA/ESA Conference on Adaptive Hardware and Systems.

[31] Rezk, A. A., Madian, A. H., Radwan, A. G., Soliman, A. M., (2019). Reconfigurable chaotic pseudo random number generator based on

FPGA. International Journal of Electronics and Communications, 98, 174-180.