www.dergipark.gov.tr
ISSN:2148-3736
El-Cezerî Fen ve Mühendislik Dergisi
Cilt: 9, No: 4, 2022 (1315-1327)

El-Cezerî Journal of Science and Engineering
Vol: 9, No: 4, 2022 (1315-1327)

DOI: 10.31202/ecjse.1134799

ECJSE

Makale / Research Paper

Random Forest Algoritmasının FPGA Üzerinde Gerçekleştirilerek Performans Analizinin Yapılması

Cem Deniz KUMRAL[1a*], Ali TOPAL[1b], Mevlüt ERSOY[2a], Recep ÇOLAK[1c], Tuncay YİĞİT[2b]

[1]Isparta Uygulamalı Bilimler Üniversitesi, Uzaktan Eğitim MYO, Bilgisayar Teknolojileri Bölümü, Isparta, Türkiye
[2]Süleyman Demirel Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Isparta, Türkiye
cemkumral@isparta.edu.tr

Received/Geliş: 24.06.2022 **Accepted/Kabul:** 09.12.2022

Öz: Random Forest (RF), rastgele oluşturulmuş birden çok karar ağacının çıktısını birleştiren, regresyon ve sınıflandırma problemlerini çözmek için kullanılan bir makine öğrenme algoritmasıdır. Ormandaki ağaç sayısının artması algoritma sonucunun kesinliğini arttırır. RF algoritması birden çok karar ağacı üzerinde çalıştığı için paralel mimariye sahip platformlarda çalıştırılması ile olumlu sonuçlar elde edilebilir. Field Programmable Gate Array (FPGA) entegre devreler, paralel işlem yapabilme yeteneğine sahip olduğundan, RF algoritmasının donanım üzerinde gerçekleştirilen uygulamalarında kullanılması performansı arttırmaktadır. Gerçekleştirilen çalışmada RF algoritması sayısal bir veri seti ile hem MATLAB üzerinde hem de FPGA üzerinde çalıştırılarak sınıflandırma işlemleri gerçekleştirilmiştir. Algoritmadaki işlem modüllerinin ve tüm mantıksal tasarımların geliştirilmesi aşamalarında Very High Speed Integrated Circuit Hardware Description Language (VHDL) kullanılmıştır. MATLAB ve FPGA mimarisi üzerinde çalıştırılan RF algoritmasının performans, doğruluk ve bellek kullanım oranları açısından karşılaştırmaları yapılmıştır. Gerçekleştirilen çalışma sonucunda, RF gibi yoğun işlemler ve hesaplamalar yürüten uygulamalarda FPGA kullanımının performans ve bellek kullanımı yönünden bilgisayar işlemcilerine kıyasla yüksek oranda başarı sağladığı görülmüştür.

Anahtar kelimeler: Random Forest, FPGA, VHDL, Donanım gerçekleme, Performans analizi

Performing Performance Analysis by Implementing Random Forest Algorithm on FPGA

Abstract: Random Forest (RF) is a machine learning algorithm used to solve regression and classification problems, combining the output of multiple randomly generated decision trees. Increasing the number of trees in the forest increases the accuracy of the algorithm result. Since the RF algorithm works on multiple decision trees, positive results can be obtained by running it on platforms with parallel architecture. Because of Field Programmable Gate Array (FPGA) integrated circuits have the ability to perform parallel operations, the use of the RF algorithm in hardware-based applications increases performance. In this study, classification processes were carried out by running the RF algorithm on both MATLAB and FPGA with a numerical data set. Very High Speed Integrated Circuit Hardware Description Language (VHDL) was used in the development of the processing modules and all logical designs in the algorithm. Comparisons of the RF algorithm run on MATLAB and FPGA architectures were made in terms of performance, accuracy and memory usage rates. As a result of the study, it has been seen that the use of FPGAs in applications that carry out intensive operations and calculations such as RF provides a higher success rate compared to computer processors in terms of performance and memory usage.

Keywords: Random Forest, FPGA, VHDL, Hardware realization, Performance analysis

Bu makaleye atıf yapmak için
Kumral, C. D., Topal, A., Ersoy, M., Çolak, R., Yiğit, T., *"Random Forest Algoritmasının FPGA Üzerinde Gerçekleştirilerek Performans Analizinin Yapılması"* El-Cezerî Fen ve Mühendislik Dergisi 2022, 9(4); 1315-1327.

How to cite this article
Kumral, C. D., Topal, A., Ersoy, M., Çolak, R., Yiğit, T., "*Performing Performance Analysis by Implementing Random Forest Algorithm on FPGA* " El-Cezerî Journal of Science and Engineering, 2022, 9.(4); 1315-1327.

ORCID: [a] 0000-0002-1326-4537; [b] 0000-0001-5975-5932; [c] 0000-0003-2963-7729,[d] 0000-0002-7119-6202, [e]0000-0001-7397-7224

1. Giriş

Rastgele Orman (RF), yaygın kullanılan Destek Vektör Makinesi (SVM) ile kıyaslanabilir sınıflandırma performansına sahip güçlü bir denetimli makine öğrenmesi algoritmasıdır [1, 2]. RF, içerdiği her ağacın birbirinden bağımsız bir şekilde örneklenen rastgele vektörlerin değerlerine dayalı ve ormanda bulunan ağaçlar için aynı dağılımdaki ağaç tahmin edicilerinin kombine edilmiş halidir. Diğer denetimli makine öğrenmesi yöntemlerine benzer şekilde, RF tasarımı eğitim ve değerlendirme olmak üzere iki aşamada incelenmektedir. Temel olarak bu algoritma, etiketlenmemiş yeni girdi verilerinin etiketini tahmin etmek için etiketi bulunan eğitim verilerinden tahminler çıkararak işlem gerçekleştirmektedir. İşlemler sırasında ormanlar için genelleme hatası elde edilmektedir. Bir ağaç sınıflandırıcı ormanının genelleme hatası, ormanda bulunan her bir ağacın gücüne ve ağaçlar arasındaki korelasyona bağlıdır. Ormandaki ağaç sayısı arttıkça genelleme hatasına bağlı olarak genel sonuca yaklaşım sağlanmaktadır [3-5]. RF, ağaçları genişletirken ağaç modeline rastgele bir yaklaşım katmaktadır. Bir ağaç düğümünü parçalara ayırırken ağaçtaki en önemli özelliği önemli aramak yerine, rastgele bir özellik alt kümesi kapsamında en iyi özelliği aramaktadır. Bu sayede daha geniş çeşitliliğe sahip bir model elde edilmektedir.

Veri miktarının çok fazla olduğu durumlarda başarılı bir RF sınıflandırıcının eğitilmesi ve değerlendirilmesi, bellek kapasitesi ve sınırlı paralel işlem yapabilme yeteneği nedeniyle modern işlemcilerde çok uzun sürebilmektedir. RF algoritması eğitilirken, kendi yapısına uygun çalışma prensibine sahip teknolojiler üzerinde uygulanması genel sonuca ulaşmada verimlilik sağlayabilmektedir. Field Programmable Gate Array (FPGA) entegre devreler, paralel işlem yapabilme yeteneğine sahip olduğundan, makine öğrenmesine dayalı donanım uygulamaları üzerine kullanımı performansı arttırmaktadır [6-8]. Programlanabilir lojik cihaz teknolojilerindeki gelişmeler ile FPGA'lar, dijital sistem tasarımlarında fazlasıyla kullanılmaya başlamıştır. Geliştiriciler tarafından istenildiği zaman yeniden yapılandırılabilmesi ve tasarım girişi, sentez, programlama için güçlü araçları barındırması FPGA'ların kullanımını arttırmaktadır. Tamamen paralel yapıya sahip mimariler tasarlamak için Application Specific İntegrated Circuit (ASIC) ve Very Large Scale Integration (VLSI) teknolojileri kullanılabilir ancak bu çiplerin geliştirilmesi hem maliyetlidir hem de oldukça zaman almaktadır [9-12]. Bunlara ek olarak, bu çiplerde tasarım sadece bir hedef uygulama için uygundur yani RF için geliştirilmiş bir çip sadece bu görevi gerçekleştirir. FPGA'ler sadece paralellik değil aynı zamanda esnek tasarımlar sunmakta, maliyet ve tasarım döngüsünde tasarruf sağlamaktadır.

Bu çalışmada FPGA üzerinde uygulanmış bir Random Forest algoritmasının donanım tasarımı sunulmuştur. RF yazılımı, MATLAB ortamında makine öğrenmesi kütüphaneleri aracılığıyla geliştirilmiştir. RF algoritması eğitilirken, sokak ortamından elde edilmiş farklı türde seslerin sayısal biçimde bulunduğu bir veri seti kullanılmıştır [13]. Eğitilen RF algoritması çıktılarına göre çeşitli ortam, canlı ve nesne sesleri üzerinde bir sınıflandırma işlemi gerçekleştirilmiştir. RF algoritmasının eğitilmesinde en yaygın donanım tanımlama dili olan Very High Speed Integrated Circuit Hardware Description Language (VHDL) kullanılarak donanım tanımlama işlemi yapılmıştır. RF için gerekli işlem adımları VHDL üzerinde ayrı ayrı görevler halinde tanımlanmıştır. Xilinx ISE geliştirme ortamı üzerinde gerçekleştirilen tasarımın çalışma hızı, doğruluk ve kaynak kullanım oranları simülasyon ortamında incelenmiştir. Gerçekleştirilen çalışma sonucunda, RF gibi yoğun işlemler ve hesaplamalar yürüten uygulamalarda FPGA kullanımının performans ve kaynak kullanımı yönünden bilgisayar işlemcilerine kıyasla yüksek oranda başarı sağladığı görülmüştür.

Bu makale şu şekilde düzenlenmiştir: Bölüm 2'de çalışma alanı ile ilgili literatürde bulunan çalışmaların kısa özetleri verilmiştir. Bölüm 3'te, gerçekleştirilen uygulamada kullanılan algoritma ve yöntemler detaylı bir biçimde açıklanmaktadır. 4. bölümde gerçekleştirilen çalışma sonucunda

elde edilen sonuçlar tablolar ile desteklenerek sunulmaktadır. Son bölüm olan 5. bölümde ise çalışma sonucunda varılan genel çıkarım anlatılmaktadır.

2. Bilimsel Yazın Taraması

Teng vd. çalışmalarında, temel olarak CART algoritmasına dayalı karar ağacı oluşturma yöntemini ele alarak rastgele orman algoritmasının donanım tasarımını ve uygulamasını incelemişlerdir. Algoritmanın eğitim kısmı ve çıkarım kısmı da dahil olmak üzere rastgele orman algoritmasını modellemek için SystemC dilini kullanışlardır. Geliştirdikleri modelde eğitim modülü, veri depolama modülü, üst düzey kontrol modülü ve algoritma yürütme modüllerini tanımlamışlardır. Donanım uygulamasının işlevsel doğruluğunu analiz etmek amacıyla modeli FPGA üzerinde çalıştırmışlardır. Gerçekleştirdikleri çalışma sonucunda önerdikleri karar ağacının veri depolama verimliliği açısından yüksek pratik değere sahip olduğunu öne sürmüşlerdir [14].

Essen vd. çalışmalarında, kompakt rastgele orman makine öğrenimi sınıflandırıcıları tarafından oluşturulan modelleri kullanarak sınıflandırmayı hızlandırmak için FPGA'ların, GP-GPU'ların ve çok çekirdekli CPU'ların performansını karşılaştırmışlardır. Daha az sayıda, derin ağaç yerine çok sayıda küçük ağaçtan meydana gelen kompakt rastgele ormanlar oluşturabilen eğitim algoritmalarından yararlanarak, sınıflandırma işlemleri sırasında işlem sürelerinin azaltılabileceğini savunmuşlardır. Yapılan çalışma sonucunda FPGA'ların en yüksek performanslı çözümü sunduğunu, GP-GPU'ların da buna yakın bir performansa sahip esnek bir çözüm sunduğunu ortaya koymuşlardır. Son olarak OpenMP ile çoklu iş parçacığı üzerinde de işlemler gerçekleştirerek bu modelin FPGA ve GP-GPU'lara göre önemli ölçüde daha yavaş olduğunu öne sürmüşlerdir [15].

Lin vd. çalışmalarında, rastgele orman algoritması için orman tahmini doğruluğu, mevcut yeniden yapılandırılabilir yapının boyutu ve izin verilen maksimum geçiş süresi konularında optimum FPGA orman mimarisini seçebilmek amacıyla bellek merkezli, karşılaştırıcı merkezli ve sentez merkezli mimariler tasarlayarak bu mimariler arasında karşılaştırmalar yapmışlardır. Orman mimarilerinin uygulanabilirliğini kanıtlayabilmek için her biri farklı bir geçiş süresi kısıtlaması ile karakterize edilen çeşitli tasarım senaryolarını gözden geçirmişlerdir. Kullanılan mimarileri, 218.600 LUT ve 545 36 Kb BRAM kapasitesine sahip bir Xilinx ZYNQ-7 ZC706 FPGA platformunda uygulamışlardır. Gerçekleştirdikleri çalışma sonucunda her mimari için harcanan FPGA kaynak kullanımlarını ortaya koymuşlardır [16].

Jinguji vd. çalışmalarında, rastgele orman mimarileri için donanım gereksinimini daha da azaltmak amacıyla karar ağacındaki dal düğümlerinin karşılaştırıcılarını paylaşmak için k-ortalama kümelemesini kullanmışlardır. Önerdikleri yaklaşımın üst düzey sentez tasarımına dayanması sebebiyle geleneksel HDL tasarımlara kıyasla daha az tasarım süresi ile daha yüksek performansta çalışan rastgele orman algoritması elde edilebileceğini öne sürmüşlerdir. Rastgele orman tasarımını Xilinx ZC702 FPGA platformu üzerinde çalıştırmışlardır. FPGA mimarisini CPU (Intel Xeon (R) E5607 İşlemci) ve GPU (NVidia Geforce Titan) üzerinde oluşturulan mimariler ile karşılaştırmışlardır. Yaptıkları çalışma sonucunda performans açısından FPGA üzerinde oluşturulan mimarinin CPU mimarisinden 8,4 kat, GPU mimarisinden ise 62,8 kat daha hızlı olduğunu, güç tüketimi verimliliği konusunda ise FPGA mimarisinin CPU mimarisinden 7,8 kat, GPU mimarisinden ise 385,9 kat kat daha verimli olduğunu ortaya koymuşlardır [17].

Mametjanov vd. çalışmalarında, FPGA tasarım parametrelerini ayarlamak için makine öğrenimi tabanlı bir yaklaşım önermişlerdir. Parametre uzayının örnekleme tabanlı azaltılmasını gerçekleştirmeyi ve aramaları daha iyi parametre konfigürasyonlarına yöneltmeyi amaçlamışlardır. Gerçekleştirdikleri çalışma sonucunda inceledikleri seçici örnekleme, zamanlama kısıtlamalarını karşılayan ve optimum güç tüketiminin %0,2'si kadarını kullanan parametre konfigürasyonlarını bulmuşlardır. Yaklaşımlarının ek parametre konfigürasyon örneklemesi gerektirmeden,

optimizasyon tabanlı modeller ile kullanılan parametre aralığında yüksek doğrulukta tahmin ürettiği ve iyi bir performansta çalıştığını öne sürmüşlerdir. Elde ettikleri konfigürasyonlar ile kapsamlı aramalara göre daha kısa sürede sonuca ulaşmışlardır. Bu tür hızlandırmalar sayesinde FPGA tasarımı oluşturma sürecindeki kısıtların önemli ölçüde azaltılabileceğini önermişlerdir [18].

Biau ve Scornet çalışmalarında, rastgele orman algoritmasının geçmişten günümüze kadarki evrimleşmesi üzerine araştırmalar ve derlemeler yapmışlardır. İlk rastgele orman çalışması 2001 yılında L Breiman tarafından yapılmıştır. Rastgele orman algoritmasının, genel amaçlı sınıflandırma ve regresyon yöntemi olarak son derece başarılı olduğu, ayrıca algoritma değişken sayısının gözlem sayısının çok fazla olması halinde yüksek performans gösterdiği görülmüştür. Büyük verilerin olduğu problemler içinde uygulanabilecek kadar güçlü olup çeşitli geçici öğrenme işlerinde kolayca uygulanabilir ve değişken sayısına göre ölçümlerde verilir. Rastgele orman algoritmasında parametrelerin seçimine, yeniden örnekleme mekanizmasına ve değişkenlik gösteren önem ölçülerine dikkat edilerek algoritmayı yönlendiren matematiksel kuvvetlere vurgu yapılmaktadır. Gerçekleştirdikleri çalışma ile rastgele orman konusunda uzman olmayan kişilerin rahatça ana fikirlere erişim sağlayabilmesini amaçlamışlardır [19].

3. Yöntem ve Materyal

3.1. Random Forest Algoritmasına Genel Bakış

Rastgele Orman, temel olarak sınıflandırma ve regresyon ağaçlarını kullanan, çoğunlukla büyük bir sonuç üreten, esnek, kullanımı basit ve verimli olan bir makine öğrenmesi algoritmasıdır. RF algoritması torbalama yöntemine dayanan topluluk öğrenme algoritmaları grubuna girmektedir. Rastgele ormanın en önemli avantajlarından biri diğer makine öğrenmesi algoritmalarının temelini oluşturan hem sınıflandırma hem de regresyon problemlerinde çözüm sağlayabilmesidir. Rastgele ormanı meydana getiren er karar ağacı paralel bir biçimde oluşturulur ve bu ağaçlar hem sınıflandırma ağaçları hem de regresyon ağaçlarından oluşabilmektedir. Her karar ağacında bulunan düğümler, tüm özellikler içerisinde en uygun çözümü elde edebilecek en iyi öznitelikler kullanılarak bölünmektedir.

RF algoritması, büyük veri setleri içerisinde yararlı olan ancak gizli durumdaki bilgileri ayrıştırabilmek için kullanılan yaygın bir yöntem haline gelmiştir. Rastgele bir ormanda, ilk olarak önyükleme yöntemi aracılığıyla eğitim setleri oluşturulmaktadır. Sonrasında her eğitim seti için bir karar ağacı oluşturulur. Gerçekleştirilen işlemler sonucunda algoritma sonucu elde edilmektedir. Random Forest algoritmasının kaba kodu Şekil 1'de verilmiştir.

```
1. Ön koşul: Eğitim seti T: (x1,y1)...(xn,yn), özellikler F, ormandaki ağaç sayısı : K
2. fonksiyon Rastgele Orman (T,F)
3. P ← Ø
4. for i ∈1,....,K do
5. T(i) ← A, T'den önyükleme yapılır.
6. P i ← RastgeleAgacOgrenimi(T(i),F)
7. P ← P ∪ {P i}
8. end for
9. return H
10. end fonksiyon
11. fonksiyon RastgeleAgacOgrenimi(T, F)
12. Her düğüm için:
13. f ← F'nin en küçük alt kümesi
14. f en iyi özelliğe göre bölünür.
15. return öğrenme ağacı
16. end fonksiyon
```

Şekil 1. Random Forest Algoritmasının kaba kodu [19, 20]

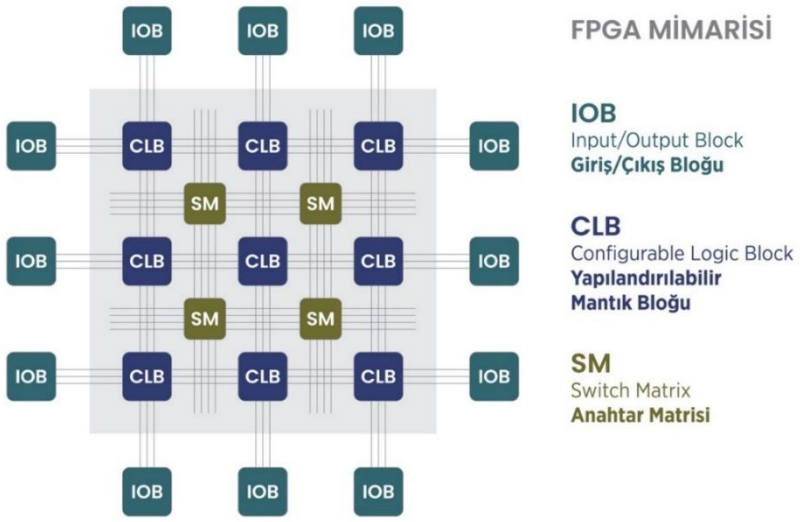
RF algoritması çalıştırıldığında öncelikli olarak eğitim seti (T), algoritma özellikleri (F) ve ormandaki ağaç sayısı (K) algoritma tarafından okunur. Başlangıçta olasılık kümesi (P) boş küme şeklindedir. Daha sonra eğitim seti için önyükleme işlemi gerçekleştirilir. Algoritma döngüsü başladığında rasgele ağaç öğrenimi için oluşturulan fonksiyon okunan eğitim seti ve algoritma özelliklerine göre ağacı alt kümelere bölerek işlemleri gerçekleştirir ve olasılıkları üretmeye başlar. Üretilen olasılıklar olasılık kümesine aktarılarak döngü tamamlanır. Daha sonra elde edilen alt kümeler arasında en küçük olan (f) bulunmuş olur. En küçük alt küme en iyi özelliğe göre bölünür ve algoritmanın sonucu döndürülür.

3.2. FPGA İşlemcisine Genel Bakış

FPGA'lar "Sahada Programlanabilir Kapı Dizileri" diye adlandırılan "Field Programmable Gate Array" ifadelerinin kısaltılmasıdır. FPGA üzerinde mevcut olan transistör sayısına göre tasarımcılar devre elemanlarının yapabildiği tüm işlevleri tek bir platform kullanarak tasarlama ve donanım gerçekleme işlemleri yapabilmektedir. FPGA'lar mantıksal devre tasarımları kapsamında tasarımcının kolay bir şekilde mantık devreleri tasarlayabilmesine ve bu devrelerin bir kart üzerinde gerçekleştirilerek test işlemlerinin yapılabilmesine imkân veren tümleşik sistemlerdir.

FPGA'lar üzerinde geliştiricilerin istediği işlevlere bağlı olarak donanım tamımlama dili ile devre yapıları değiştirilebilmektedir. Bu durum FPGA'ları elektronik alanında kullanımı oldukça tercih edilen mikroişlemcilerden farklı kılan önemli bir özelliktir. FPGA'lar paralel olarak işlem gerçekleştirebilme yeteneğine sahip olan platformlardır. VHDL dizaynında tanımlanmış bir entity yapısına bağlı olarak pek çok görev oluşturulabilmektedir. Bu görevler paralel bir biçimde çalışabilmektedir. Bir mikroişlemci üzerinde geliştirilen bütün yazılımlar sıralı bir biçimde işlem görmektedir, diğer işlemleri gerçekleştirmek için kesmeler, timer'lar vasıtasıyla görevler arası geçiş yapılabilir ama bu durum program üzerinde bir miktar yavaşlığı beraberinde getirmektedir. FPGA mimarisi paralel görev çalıştırabildiği için böyle problemlerle karşılaşılmamaktadır.

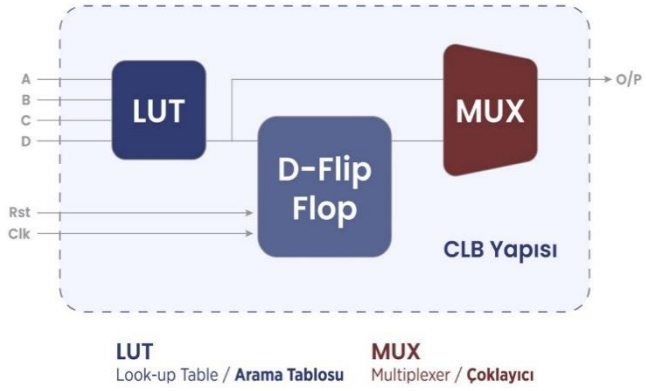
FPGA'lar; yapılandırılabilir mantık blokları, bu mantık bloklarının çevre birimler ile haberleşmesini sağlayan giriş-çıkış noktaları ve tüm birimlerin veri iletişimini sağlayan ara bağlantılar şeklinde üç temel bileşenden meydana gelmektedir. Bu birimlerin arasında anahtarlama görevlerini yerine getiren anahtar matrisleri bulunmaktadır. FPGA'nın temel yapısını gösteren çizim Şekil 2'de verilmiştir.



Şekil 2. Temel FPGA mimarisi [21]

Yapılandırılabilir mantık blokları gömülü bir biçimde ara bağlantıların arasında bulunmaktadır. Mantık hücrelerinin yapılandırılması ve diğer mantık hücreleri ile haberleşmesi ara bağlantılar vasıtasıyla yapılmaktadır. Mantık hücrelerinde oluşan ve ara bağlantılar ile iletilen verilerin çevre birimler ile haberleşmesi ise giriş-çıkış birimleri üzerinden sağlanmaktadır.

Yapılandırılabilir mantık blokları hafıza görevi gören bir adet Look Up Table (LUT) birimi, bir adet flip-flop devresi ve bloğun çıkısını belirleyen bir adet Multiplexer (MUX) devresinden oluşmaktadır. Yapılandırılabilir mantık bloğunun iç yapısını gösteren görsel Şekil 3'te verilmiştir.



Şekil 3. Yapılandırılabilir Mantık Bloğunun iç yapısı

3.3. VHDL Diline Genel Bakış

FPGA üzerinde dizayn tasarlayabilmek için geliştiriciler ya donanım tanımlama dili kullanılmalı ya da şematik tasarım yapılmalıdır. Günümüzde FPGA tasarımı geliştirmek amacıyla tercih edilen en yaygın metot VHDL (Very High Speed Integrated Circuit Hardware Description Language)'dir. Bundan farklı olar ak Verilog dili de kullanılmaktadır. ISE geliştirme ortamında VHDL dili ile tanımlanmış bir tasarım örneği Şekil 4'te verilmiştir.

```
architecture Behavioral of and_or_gates is

begin

PROCESS(h1,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16)
BEGIN
IF h1 ='1' THEN
o1 <= i1 and i2;
o2 <= i3 and i4;
o3 <= i5 and i6;
o4 <= i7 and i8;
ELSE
END IF;
END PROCESS;

PROCESS(h2,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16)
BEGIN
IF h2 = '1' THEN
o5 <= i9 or i10;
o6 <= i11 or i12;
o7 <= i13 or i14;
o8 <= i15 or i16;
ELSE
```

Şekil 4. Örnek VHDL kod parçası

VHDL, sayısal elektronik devreleri tanımlamak amacıyla tercih edilen bir donanım tanımlama dilidir. Temel olarak, VHDL'deki bir tasarımın yapısı arayüz tanımlanması ve mimari tanımlanması ile gerçekleştirilmektedir. Arabirim tanımlamaları ENTITY anahtar kelimesiyle başlar ve tasarımın giriş çıkış tanımlamalarının yapıldığı bölümdür. İlk olarak ENTITY anahtar sözcüğü sonrasında ise bileşenin adı yazılır. Daha sonra tasarımın mimari bölümü gelmektedir. Uygulamanın asıl mantık işlemleri mimari kısımda tanımlanmaktadır. Mimari bölümünü tanımlamak için ARCHITECTURE anahtar sözcüğü kullanılmaktadır. Aynı fonksiyonları yerine getirmek için geliştirilen uygulamalar birden fazla mimari gövdeye sahip olabilmektedir.

Sayısal devre tasarımları VHDL ile tanımlanır ve daha sonra geliştirilen programın simülasyon işlemleri gerçekleştirilir. Simülasyon işlemlerinin gerçekleştirilme nedeni VHDL tasarımının yapılması hedeflenen mantıksal işlemleri doğru bir biçimde yerine getirip getirmediğini doğrulamaktır. Geliştirilen VHDL tasarımı FPGA kartları üzerine aktarılarak donanım gerçekleme işlemi tamamlanmaktadır.

3.4. Uygulamanın Gerçekleştirilmesi

3.4.1. RF algoritmasının MATLAB üzerinde uygulanması

Gerçekleştirilen çalışmada RF algoritması ilk olarak MATLAB açık kaynak araç kutusu üzerinde kullanılmıştır. Kullanılan veri seti 8732 farklı kentsel ses dosyası içermektedir. Çeşitli ortam, canlı ve nesne sesleri üzerinde bir sınıflandırma işlemi gerçekleştirilmiştir. Gerçekleştirilen çalışmada veri setinin sınıflandırılması sürecinde tercih edilen RF algoritması için seslerin örnekleme hızı, bit derinliği ve kanal sayısı parametreleri kullanılmıştır. Veri setinde yer alan seslerin kategorilerine sayısal değerler tanımlanmıştır. Bu kategoriler classID (c) olarak ifade edilmiştir. Kullanılan kentsel seslerin orijinal kayıtlarında sesin farklı oluşumlarını ayırt etmek için occurrenceID (o) adında sayısal bir tanımlayıcı belirlenmiştir. Tanımlanan seslerin kategoriler düzeyinde benzerlikleri için farklı oluşumlara göre 1 ile 0 aralığında değerler atanmaktadır. MATLAB üzerinde gerçekleştirilen sınıflandırma işleminin örnek kod parçası Şekil 5'te verilmiştir.

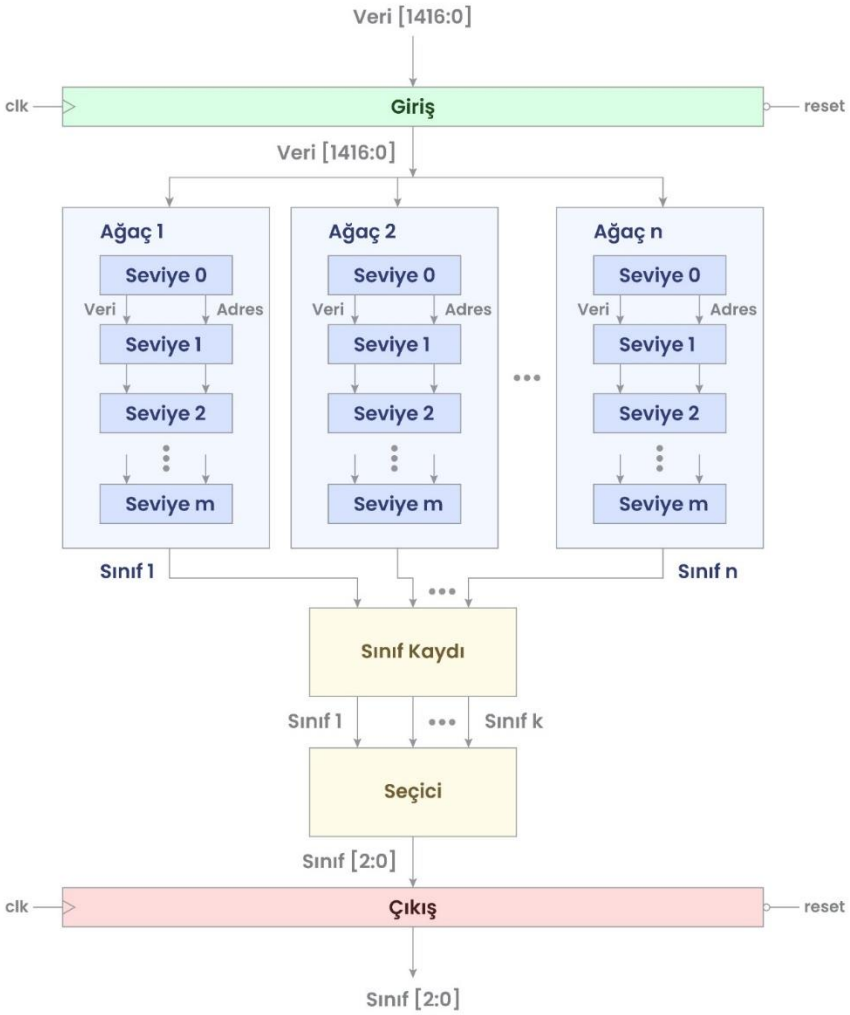
```
clc
clear all
close all
warning off
data=readtable('UrbanSound8k.csv');
c=["airconditioner","carhorn","childrenplaying","dogbark","drilling",
    "engineidling","gunshot","jackhammer","siren","street_music"];
o=[1,0];
g=data.soundClass;
number=zeros(length(g),1);
for i=1:length(c)
    rs=ismember(g,c(i));
    number(rs)=o(i);
end
data.category_encoded=number;
data.soundClass=[];
cv = cvpartition(size(data,1),'HoldOut',0);
idx = cv.test;
dataTrain=data(~idx,:);
dataTest=data(idx,:);
testing=dataTest(1:end,1:end-1);
model=fitensemble(dataTrain,'category_encoded','Bag',100,'Tree', ...
    'Type','classification');
```

Şekil 5. MATLAB uygulamasının örnek kod parçası

3.4.2. RF algoritmasının FPGA üzerinde uygulanması

Gerçekleştirilen çalışmada RF algoritması Xilinx Spartan XCS500E FPGA kartı üzerinde çalıştırılmıştır. Geliştirilen uygulamadaki VHDL tasarım özellikleri her bir özelliğin 56 bitlik sabit noktalı bir sayı olarak kodlandığı ikili sayılar kullanılarak kodlanmıştır. Genellikle kayan nokta mimarisinden daha hızlı olma eğiliminde olan çok daha basit bir donanım tasarımı oluşturulması sebebiyle, kayan nokta yerine sabit nokta yapısı seçilmiştir. Özelliklerin her biri, 30 bit tam sayı kısmına benzeyecek ve 26 bit sayının kesirli kısmını tanımlamak için kullanılacak şekilde oluşturulmuştur. Bu sebeple bir ağ paketinin özelliklerini tanımlamak için 1417 bit gerektiren bir kodlama yapısı oluşmaktadır.

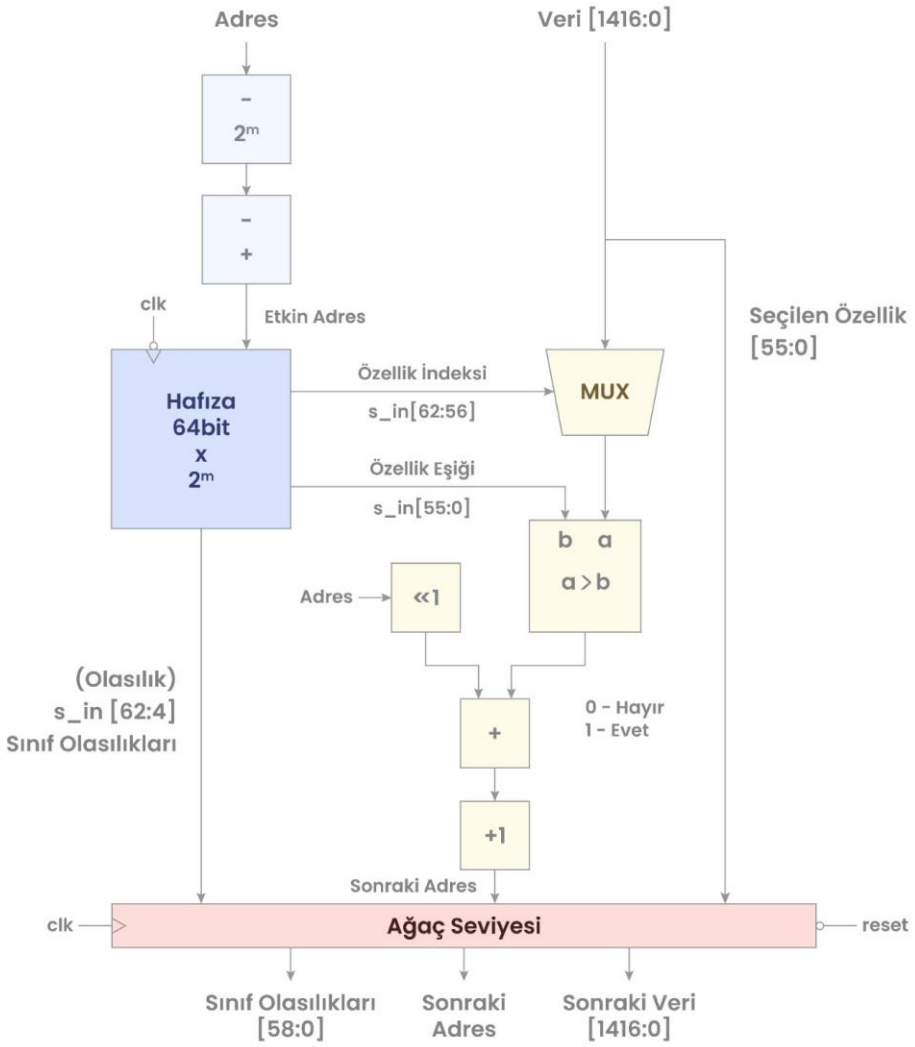
Donanım üzerinde rastgele orman sınıflandırıcısı tasarlarken temel amaç, birbirinin çalışmasını etkilemeden aynı anda çalışabilen bağımsız bileşenleri analiz etmektedir. En belirgin bağımsız bileşenler, ormandaki bireysel ağaçlardır çünkü bir test örneği, diğer ağaçların çıktısına bağımlı olmadan her ağaçta işleme sokulur. FPGA üzerinde oluşturulan RF tasarımının genel mimarisi Şekil 6'da gösterilmektedir. Bir test paketi, ağaçlarda işleme girmeden önce bir giriş kaydına kaydedilmektedir. Test örneği ağaçlara ulaştığında, ağaçların farklı seviyelerine girmektedir.



Şekil 6. FPGA RF tasarımının genel mimarisi

Her ağaç seviyesi, bir seferde yalnızca bir paketi incelemektedir, bu sebeple tüm ağacı tek bir büyük bileşen olarak incelemek yerine, farklı ağaç seviyeleri arasına ardışık düzen aşamaları eklenmektedir. Bu durum, geliştirilen tasarımın FPGA'ların paralel yürütme yeteneklerinden

yararlandığı bir başka yöndür çünkü bu sayede her ağaç seviyesi, ağaç içindeki diğer tüm seviyelere göre aynı anda ve bağımsız olarak çalışabilmektedir. Buraya kadar gerçekleştirilen işlemlerin çıktısı, ya çoğunluğa dayalı rasgele orman durumunda sınıf etiketi ya da olasılığa dayalı rasgele orman durumunda sınıf olasılıklarını oluşturmaktadır. Bu aşamadan sonra, tüm ağaçların çıktıları "Sınıf Kaydı" olarak adlandırılan bir modüle aktarılmaktadır. Sınıf Kaydı modülü, tüm ağaçların sonuçlarını basitçe toplayarak sonuçları "Seçici" modülüne iletmektedir. Seçici modülü, veri örneğinin sınıf etiketi olarak en çok ortaya çıkan sınıfı (çoğunluk temelli) veya en yüksek olasılığa sahip sınıfı (olasılığa dayalı) seçmektedir. Son olarak çıktı sınıfı, kullanıcıya hemen görüntülenebilecek şekilde bir çıkış kayıt modülüne aktarılmaktadır. Şekil 7'de bir ağaç seviyesindeki donanım yapısı gösterilmektedir.



Şekil 7. Bir ağaç seviyesindeki donanım yapısı

İlk olarak, etkin adres bir çıkarıcı ve bir toplayıcı kullanılarak hesaplanmaktadır. Daha sonra etkili adres ilgili ağaç seviyesindeki tüm düğümler hakkındaki bilgileri tutan ağaç hafızasını indekslemek için kullanılır. Ağaç hafızası, sadece bu düğümde kontrol edilen özelliğin indeksi olan özellik indeksini getirmektedir. Öznitelik indeksi daha sonra bir çoklayıcı tarafından seçim çizgileri olarak kullanılmaktadır. Bu da yalnızca incelenen özniteliğin değerinin sonraki bileşenlere aktarılmasını sağlar. Ayrıca ağaç belleği, özellik değerinin karşılaştırılacağı değer olan özellik eşiğini de sağlamaktadır. Karşılaştırıcı, seçilen özelliğin özellik eşiğinden büyük olup olmadığını kontrol eder. Karşılaştırmanın sonucu daha sonra bir sonraki adresi hesaplamak için kullanılmaktadır.

4. Tartışma ve Sonuçlar

4.1. Genel Sonuçlar

Uygulamanın gerçekleştirilme aşamasında FPGA tasarımı oluşturabilmek için mevcut olan VHDL, Verilog ve Sematik dizayn yöntemleri arasından tümleşik devreler donanım tanımlama dili olan VHDL tercih edilmiştir. VHDL dili diğer dizayn yöntemlerine kıyasla daha farklı bir yapıya sahip olmasına rağmen orta seviye donanım bilgisine sahip geliştiriciler tarafından hızlıca öğrenilip devre tasarımları yapılabilmektedir. Bu sayede pek çok geliştiricinin donanım tasarımları oluşturmasında tercih edilmektedir. Gerçekleştirilen çalışma sonucunda FPGA kullanılarak tasarlanan RF yapısının çok kısa sürelerde doğru sonuçlar ürettiği gözlemlenmiştir. Donanım tabanlı RF tasarımları oluşturmada FPGA kullanılmasının oldukça mantıklı ve doğru bir yöntem olduğu kanıtlanmıştır. Ayrıca FPGA tasarımları geliştirirken VHDL dili kullanılabilmesi sayesinde kodlama üzerinden dizayn oluşturmak mümkün olduğundan dizaynın harici bir programda çizilmesine gerek kalmamaktadır. FPGA üzerinde geliştirilen donanım tabanlı RF tasarımı algoritmanın çalışma süresini oldukça kısaltmıştır. MATLAB ile çok uzun süre çalışmaya devam edecek olan algoritma işlemleri FPGA-RF kombinasyonu ile daha kısa sürelerde yapılabilmektedir.

4.2. Performans, Doğruluk ve Kaynak Kullanım Sonuçları

Bilgisayar işlemcisi üzerinde MATLAB üzerinde ve FPGA mimarisi üzerinde çalıştırılan RF algoritmasının performans, doğruluk ve bellek kullanım oranları açısından karşılaştırmaları yapılmıştır. FPGA üzerinde tasarlanan RF dizaynı FPGA platformunun sahip olduğu mantıksal devreleri kullanırken, MATLAB programı bilgisayarın kendi işlemcisini kullanmaktadır. VHDL dilinin getirdiği esneklik sayesinde istenilen durumlarda donanım tasarımı üzerinde hızlı bir şekilde değişiklik yapılarak daha çok bit kullanımı ile daha fazla duyarlılık sağlayabilmek mümkündür.

4.2.1. Performans analizinin yapılması

Bilgisayar işlemcisi üzerinde çalıştırılan MATLAB yazılımının ve FPGA işlemcisi üzerinde çalıştırılan VHDL tabanlı tasarımın aynı RF algoritması kapsamında işlemleri gerçekleştirme süreleri karşılaştırılmıştır. Karşılaştırma sonuçları Tablo 1'de gösterilmektedir.

Tablo 1. İşlem sürelerinin karşılaştırılması

MATLAB	FPGA
3.7 sn (3700 ms)	930 ms

Tabloda görüldüğü üzere FPGA üzerinde gerçekleştirilen uygulamanın MATLAB ile çalıştırılan uygulamaya göre işlemleri çok daha hızlı bir şekilde tamamladığı görülmüştür. Daha yüksek kaynak kapasitesine sahip FPGA kartları kullanılarak daha hızlı sonuçlar elde etmek mümkündür.

4.2.2. Doğruluk oranlarının karşılaştırılması

FPGA üzerinde gerçekleştirilen RF tasarımının MATLAB üzerinde geliştirilen uygulama ile doğruluk açısından kıyaslaması yapılmıştır. Karşılaştırma sonuçları Tablo 2'de gösterilmektedir.

Tablo 2. Doğruluk oranlarının karşılaştırılması

% MATLAB	% FPGA
97.4	95.3

Karşılaştırma sonucunda MATLAB üzerinde gerçekleştiren uygulamanın doğruluk oranının FPGA işlemcisine göre daha yüksek olduğu görülmüştür. Fakat bilgisayar işlemcisi üzerinde 64 bit düzeyinde uygulama gerçekleştirilirken FPGA üzerinde gerçekleştirilen RF uygulamasında daha düşük bit seviyelerinde çalışıldığı için doğruluk oranlarının bu şekilde olması normal karşılanabilir.

4.2.3. FPGA kaynak kullanım sonuçları

FPGA üzerinde tasarlanan RF uygulamasının çalıştırılması sonucunda kullanılan kaynaklar ve kullanım miktarları Tablo 3'te verilmiştir.

Tablo 3. FPGA kaynak kullanım sonuçları

Kaynak	Toplam	Kullanılan	% Kullanım Oranı
Slice	4656	4573	98
Flip-Flop	9312	7342	79
LUT	9312	9146	98
Kaydediciler	N/A	478	N/A
Bellek Bitleri	368.640	243.174	66

Gerçekleştirilen tasarım FPGA kartı üzerinde %98 oranında Slice, %79 oranında Flip-Flop, %98 oranında LUT ve %66 oranında bellek biti kullanmaktadır. Tasarımın geliştirildiği FPGA kartındaki kaynaklar kısıtlı olduğundan kaynak kullanım oranları yüksek çıkmıştır. Daha geniş kaynak kapasitesine sahip FPGA kartları tercih edilerek performans ve kararlılık yönünden daha başarılı sonuçlara ulaşılabilir.

4.3. Gerçekleştirilen çalışmanın literatürdeki benzer çalışmalar ile karşılaştırılması

Gerçekleştirilen çalışmanın literatürdeki benzer çalışmalar ile karşılaştırılması Tablo 4'te verilmiştir.

Tablo 4. Literatürdeki benzer çalışmalar ile kıyaslama

Çalışma	FPGA Kaynak Kullanımı	Performans Analizi	Güç Verimlilik Analizi	FPGA'de Kullanılan Dil	Karşılaştırılan İşlemci
[14]	Var	Var	Yok	Verilog	Karşılaştırma yapılmamıştır.
[15]	Var	Var	Var	Belirtilmemiştir.	FPGA-GPU-CPU
[16]	Var	Var	Yok	Belirtilmemiştir.	Farklı FPGA Mimarileri
[17]	Var	Var	Var	Belirtilmemiştir.	FPGA-GPU-CPU
[18]	Yok	Var	Var	Belirtilmemiştir.	Karşılaştırma yapılmamıştır.
Bu çalışma	Var	Var	Yok	VHDL	FPGA-CPU

Yukarıdaki tabloda gerçekleştirilen çalışmanın literatürdeki benzer çalışmalar ile karşılaştırılması beş farklı kritere göre yapılmıştır. "FPGA Kaynak Kullanımı" yapılan çalışmalarda FPGA'lerde bulunan bellek birimleri ile ilgili kullanım oranlarının çalışmada bulunup bulunmadığını temsil etmektedir. "Performans Analizi" kullanılan platformlar arasında bir kıyaslama yapılıp yapılmadığını göstermektedir. "Güç Verimlilik Analizi" platformların çalışma esnasında tükettikleri

güç miktarlarını temsil etmektedir. “FPGA’de Kullanılan Dil” donanım tanımlama işlemi için tercih edilen dili ifade etmektedir. “Karşılaştırılan İşlemci” ise çalışmalarda karşılaştırılan işlemci türlerini göstermektedir.

5. Çıkarım

Gerçekleştirilen çalışma sonucunda RF modelinin VHDL dili ile FPGA üzerinde kullanımının hesaplama hızı ve bellek kullanımı açısından büyük ölçüde olumlu etkileri olduğu ortala koyulmuştur. Ayrıca kullanılan FPGA devresinin istenildiği kadar düzenlenebilir bir yapıda olması ve dizaynın VHDL ile gerçekleştirilmesi sayesinde esnek bir tasarım elde edilmiştir. Bu sayede bir dizaynın kısa bir sürede değiştirilerek farklı uygulamalarda da kullanılabileceği görülmüştür. FPGA’ların günümüzde kullanılan diğer işlemcilere kıyasla daha yüksek hız, güvenlik ve paralel işlem gerçekleştirebilme yeteneğine sahip olması sebebiyle rastgele orman algoritması kullanılarak çok uyumlu çalışmalar yapılabileceği ve benzer tarzda çalışma prensibi olan mimarilerde FPGA işlemcisi kullanmanın olumlu sonuçlar getirebileceği ortaya koyulmuştur. Bunun yanı sıra FPGA üzerinde kullanılan mantık kapıları ve flip-flop sayılarının artması ile daha hatasız sonuçlar elde edebileceği görülmüştür. RF tabanlı sistemler kapsamında FPGA’nın maliyet, performans, esnek tasarım ve paralel mimari açılarından oldukça uygun bir çözüm olduğu gösterilmiştir.

Kaynaklar

[1]. Cheng, C., Bouganis, C. S., Accelerating random forest training process using FPGA, 23rd International Conference on Field programmable Logic and Applications, 2013, 1-7, IEEE.

[2]. Caruana, R., Niculescu-Mizil, A., An empirical comparison of supervised learning algorithms, In Proceedings of the 23rd international conference on Machine learning, 2006, 161-168.

[3]. Freund, Y. Schapire, R., Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, 1996, 148–156.

[4]. Amit, Y., Geman, D., Shape quantization and recognition with randomized trees, Neural computation, 1997, 9(7), 1545-1588.

[5]. Breiman, L., Random forests. Machine learning, 2001, 45(1), 5-32.

[6]. Lin, X., College student employment data platform based on FPGA and machine learning, Microprocessors and Microsystems, 2020, 103471.

[7]. Leilei, W., Huina, C., Physical education image analysis based on virtual crowd simulation and FPGA, Microprocessors and Microsystems, 2020, 79, 103319.

[8]. Zhang, Z., Semi-supervised hyperspectral image classification algorithm based on graph embedding and discriminative spatial information, Microprocessors and Microsystems, 2020, 75, 103070.

[9]. Devika, K. N., Bhakthavatchalu, R., Design of reconfigurable LFSR for VLSI IC testing in ASIC and FPGA, International conference on communication and signal processing (ICCSP), 2017, 0928-0932, IEEE.

[10]. Vannal, N. S., Siddamal, S. V., Bidaralli, S. V., Bhille, M. S., Design and testing of combinational Logic circuits using built in self test scheme for FPGAs, Fifth international conference on communication systems and network technologies, 2015, 903-907, IEEE.

[11]. Bhakthavatchalu, R., Krishnan, S., Vineeth, V., Devi, M. N., Deterministic seed selection and pattern reduction in Logic BIST, In 18th International Symposium on VLSI Design and Test, 2014, 1-2, IEEE.

[12]. Panda, A. K., Rajput, P., Shukla, B., FPGA implementation of 8, 16 and 32 bit LFSR with maximum length feedback polynomial using VHDL, International Conference on Communication Systems and Network Technologies, 2012, 769-773, IEEE.

[13]. J. Salamon, C. Jacoby and J. P. Bello, A Dataset and Taxonomy for Urban Sound Research, 22nd ACM International Conference on Multimedia, Orlando USA, 2014.

[14]. Teng, Z., Chu, L., Chen, K., He, G., Fu, Y., Li, L., Hardware Implementation of Random Forest Algorithm Based on Classification and Regression Tree, IEEE International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), 2020, 1422-1427, IEEE.

[15]. Van Essen, B., Macaraeg, C., Gokhale, M., Prenger, R., Accelerating a random forest classifier: Multi-core, GP-GPU, or FPGA?, IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, 2012, 232-239, IEEE.

[16]. Lin, X., Blanton, R. S., Thomas, D. E., Random forest architectures on FPGA for multiple applications, In Proceedings of the on Great Lakes Symposium on VLSI, 2017, 415-418.

[17]. Jinguji, A., Sato, S., Nakahara, H., An FPGA realization of a random forest with k-means clustering using a high-level synthesis design, IEICE TRANSACTIONS on Information and Systems, 2018, 101(2), 354-362.

[18]. Mametjanov, A., Balaprakash, P., Choudary, C., Hovland, P. D., Wild, S. M., Sabin, G., Autotuning FPGA design parameters for performance and power, IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines, 2015, 84-91, IEEE.

[19]. Biau, G., Scornet, E., A random forest guided tour, 2016, Test, 25(2), 197-227.

[20]. Ali, A. B. W., Prediction of Employee Turn Over Using Random Forest Classifier with Intensive Optimized Pca Algorithm, Wireless Personal Communications, 2021, 1-18.

[21]. Ersoy, M., Kumral, C. D., Realization of Artificial Neural Networks on FPGA, In The International Conference on Artificial Intelligence and Applied Mathematics in Engineering, 2020, 418-428, Springer.