T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA ÜZERİNDE DİFERANSİYEL GELİŞİM ALGORİTMASI İLE YAPAY SİNİR AĞI EĞİTİMİ

ALİ RIZA YILMAZ

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ ANABİLİM DALI
ELEKTRONİK PROGRAMI

DANIŞMAN
YRD. DOÇ. DR. BURCU ERKMEN

İSTANBUL, 2014

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

GENEL AMAÇLI YAPAY SİNİR AĞLARININ DİFERANSİYEL GELİŞİM ALGORİTMASI İLE EĞİTİMİNİN FPGA ÜZERİNDE GERÇEKLENMESİ

Ali Rıza YILMAZ tarafından hazırlanan tez çalışması 03.02.2014 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Burcu ERKMEN

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Burcu ERKMEN

Yıldız Teknik Üniversitesi _____

Prof. Dr. Tülay YILDIRIM

İstanbul Üniversitesi _____

Yrd. Doç. Dr. Songül ALBAYRAK

Yıldız Teknik Üniversitesi _____

Bu çalışma, Yıldız Teknik Üniversitesi Bilimsel Araştırma Projeleri Koordinatörlüğü' nün 2012-04-03-KAP02 numaralı projesi ile desteklenmiştir.

ÖNSÖZ

Öncelikle benim bu günlere gelmemi sağlayan ve hep bana destek olan aileme, yüksek lisansım boyunca elinden gelen tüm yardımı hiçbir zaman esirgemeyen çok değerli danışman hocam Yrd. Doç. Dr. Burcu ERKMEN'e ve tez sürecinde hep yardımına başvurduğum dostum Dr. Oğuzhan YAVUZ'a yürekten teşekkür ederim.

Ocak, 2014

Ali Rıza YILMAZ

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

w	Ağırlık değeri
c	Merkez değeri
ω	Açı değeri
x	Ağın giriş vektörü
$f(.)$	Aktivasyon fonksiyonu
$f'(.)$	Aktivasyon fonksiyonun türevi
δ	Yerel eğim hesabı
γ	Öğrenme oranı
α	Momentum sabiti
$x_{j,i,G}$	G nesli için, i kromozomunun j parametresi (gen)
$n_{j,i,G+1}$	Mutasyon ve çaprazlamaya tabi tutulmuş ara kromozom
$u_{j,i,G+1}$	xj,i,G 'den bir sonraki jenerasyon için üretilen kromozom (child-trial)
F	Ölçekleme faktörü

KISALTMA LİSTESİ

ASIC	Application Specific Integrated Circuit
CLB	Configurable Logic Blok
CMT	Clock Management Tile
CPLD	Complex Programmable Logic Device
CSFNN	Conic Section Function Neural Network
ÇKA	Çok Katmanlı Algılayıcı
DGA	Diferansiyel Gelişim Algoritması
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Array
FIFO	First-In First-Out
GYA	Geri Yayılım algoritması
HSSIO	High-Speed Serial Input/Output
KKFSA	Konik Kesit Fonksiyonlu Sinir Ağı
LUT	Look Up Table
PLD	Programmable Logic Device
PLL	Phase Lock Loop
PCA	Principal Component Analysis
RBF	Radial Basis Function
RTF	Radyal Temelli Fonksiyon
TBA	Temel Bileşenler Analizi
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLSI	Very Large Scale Integrated Circuit
YSA	Yapay Sinir Ağları

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

FPGA ÜZERİNDE DİFERANSİYEL GELİŞİM ALGORİTMASI İLE YAPAY SİNİR AĞI EĞİTİMİ

Ali Rıza YILMAZ

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doc. Dr. Burcu ERKMEN

İleri yönlü yayılım ve eğitim algoritmalarındaki yoğun işlem yüküne sahip Yapay Sinir Ağlarının (YSA) donanım olarak tümdevre veya gömülü sistem üzerinde gerçeklenmesi, hızlı ve ağın yapısına uygun paralel tasarımlar sağlamasından dolayı yazılım uygulamalarına göre tercih edilmektedir. Tezde YSA'nın ileri yönlü yayılım işlemlerinin ve eğitim sürecinin gömülü sistem üzerinde tasarımı gerçekleştirilmiştir.

Esnek tasarım özelliklerine, programlanabilir ara bloklara ve paralel işlem yeteneğine sahip Alan Programlanabilir Kapı Dizileri (FPGA: Field-Programmable Gate Array), zaman ve maliyetten önemli kazanımlar sağlaması nedeniyle yoğun paralel yapıya ve işlem yüküne sahip YSA için kullanılacak en uygun platformlardan biridir. YSA eğitim ve test süreçlerinin tasarımı, hedeflenen gömülü sistem üzerinde tamamen sayısal olarak gerçekleştirilmesi sağlanmıştır.

YSA'ların eğitim işlemi yapılırken, yerel minimumlara takılma riski, gradyen tabanlı algoritmalara göre daha az olan, Diferansiyel Gelişim Algoritması (DGA) kullanılmıştır.

Tez dört temel aşamadan oluşmaktadır. Birinci aşamada optimizasyon için kullanılacak olan DGA MATLAB ortamında, ileride donanımda kullanılacağı düşünülerek gerçeklenmiş ve iki bilinmeyenli bir denklem ile doğruluğu gözlenmiştir. İkinci aşamada bu algoritmanın YSA eğitimi için uygun olup olmadığını gözlemlemek amacıyla MATLAB

ortamında Konik Kesit Fonksiyonlu Sinir Ağı (KKFSA) ve Çok Katmanlı Algılayıcı (ÇKA) ağlarının eğitimi gerçeklenmiş ve çok boyutlu, doğrusal olmayan bir problem olan imza tanıma veri kümesi ile analiz edilmiştir. Daha sonraki aşamada DGA'nın donanım için uygun olup olmadığını gözlemlemek için FPGA'de gerçeklenmiş ve iki bilinmeyenli bir denklem optimizasyonu yapılmıştır. Son aşamada ise, FPGA üzerinde eğitim için tasarlanan bir ÇKA yapısının eğitim işlemi, DGA ile FPGA üzerinde gerçeklenmiştir. İleri yönlü YSA ve optimizasyon algoritmasının FPGA üzerinde tasarımında, hesaplamalarda sağladığı hassaslıktan dolayı kayan noktalı sayı formatı kullanılmıştır.

Anahtar Kelimeler: Alan programlanabilir kapı dizileri, yapay sinir ağları, diferansiyel gelişim algoritması

ABSTRACT

TRAINING OF ARTIFICIAL NEURAL NETWORK WITH DIFFERENTIAL EVOLUTION ALGORITHM ON FPGA

Ali Rıza YILMAZ

Department of Electronics and Communications Engineering

MSc. Thesis

Adviser: Assist. Prof. Dr. Burcu ERKMEN

Design of Artificial Neural Networks with a very heavy computational complexity of feed forward propagation and training algorithms on a chip or an embedded system is preferred to their software counterparts due to rapid and parallel hardware design suitable for the structure of network. In this thesis, feed forward computation and training process of ANN were realized on embedded system.

FPGA that has a flexible design features, programmable intermediate blocks and parallel processing capability is the most appropriate platform for ANN with massively parallel structure and the processing load since it provides significant gains in time and costs. ANN training and testing processes were carried out in completely digital on the targeted embedded system design.

In training process of ANN, Differential Evolution Algorithm (DEA) which has less risk to get stuck at the local minima than gradient based algorithms were used.

DEA which will be used for optimization has been designed in MATLAB environment considering that it would be used in hardware implementation in the future and the accuracy was demonstrated using an equation with two unknowns. In the secont part, the training of Conic Section Function Neural Network (CSFNN) and MLP has been realized on the MATLAB with this algorithm in order to see whether or not DEA is convenient for ANN training or not and it has been analyzed by using high-dimensional

and non-linear signature recognition data base. Then, DEA was implemented on FPGA to show the suitable for hardware and the optimization of an equation with two unknowns was realized. In the last part, the training of MLP designed for training was implemented with DEA on FPGA. Since the floating-point arithmetic provides sensitivity in calculations, this arithmetic is used for the design of feed-forward CSFNN and the optimization algorithms on FPGA.

Keywords: FPGA, artificial neural networks, differential evolution algorithm

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Günümüzde Alan Programlanabilir Kapı Dizileri (FPGA: Field-Programmable Gate Array) teknolojisi, yüksek hızda işlem yapabilme özelliği tekrar tekrar programlanabilir bir yapıda olmasından dolayı, iletişim ağı bağlantıları [1], sinyal ve görüntü işleme[2],[3] ve şifreleme uygulamaları[4] gibi birçok alanda kullanılmaktadır. Yoğun paralel işlem yapabilen FPGA ve çok geniş ölçekli tümleşik devre (VLSI) teknolojisinin gelişimi, yapılan çalışmalar ile yapay sinir ağı modellerinin tasarımı için elverişli olduğunu göstermektedir[5-8]. Yapay sinir ağlarıın FPGA üzerinde gerçeklenmesi ile yapılan çalışmalar genellikle eğitim kısmı olmadan çevrim dışı yapılan uygulamalardır [9]. Son zamanlarda YSA yapılarıın FPGA üzerinde gerçeklenirken eğitim kısımlarınında dâhil edilerek çevrimiçi sistem yapıların oluşturulması hedeflenmiştir. Böylelikle hem eğitim işleminin hem de eğitilen ağın probleme uygulanması çok daha hızlı yapılabilmektedir. Levernberg-Marquardt algoritması ile eğitilen Çok Katmanlı Algılayıcı (ÇKA) yapısı [10],[11] ve delta algoritması ile eğitilen Radyal Tabanlı Fonksiyon (RTF) yapılarının [12] çeşitli uygulamalar ile FPGA gerçeklemeleri yapılmıştır. Yapay sinir ağı modellerinden KKFSA'da, imza tanıma uygulaması için FPGA'de gerçeklenmiştir [13].

KKFSA yapısı ÇKA ve RTF ağlarının özelliklerini içinde barındırarak, karar sınırlarındaki esnek yapısı ile daha iyi sınıflandırma yapma olanağı sağlamaktadır [14].

Son zamanlarda YSA'ların eğitim işlemi için yapay arı kolonisi [15], parçacık sürü optimizasyonu [16] diferansiyel gelişim algoritması (DGA) [17-19] gibi sezgisel optimizasyon yöntemleri kullanılmaya başlanmıştır. Çok boyutlu ve doğrusal olmayan

problemler için DGA kullanılarak, YSA'nın eğitimde kullanılan geriye yayılım algoritmasında karşılaşılan yerel minimuma takılma problemine çözüm aranmıştır[20].

KKFSA, GYA ve diferansiyel gelişim algoritması (DGA) ile eğitimi yapılarak imza tanıma işlemi gibi uygulamalarda kullanılan yapay sinir ağı modellerindendir [21-24].

Sezgisel algoritmalardan karınca kolonisi optimizasyonu, geniş spektrumlu tümleşik bir problemin optimizasyonu için FPGA'de gerçeklenmiş ve sonuçlar yorumlanarak bu tür algoritmaların donanım gerçeklemeleri için uygun olduğu gözlenmiştir [25].

[26-28] çalışmaları bize FPGA teknolojisinin genetik algoritmaların gerçeklenmesi için uygun olduğunu göstermektedir.

1.2 **Tezin Amacı**

Konik Kesit Fonksiyonlu Sinir Ağının ileri yönlü yayılım işlemlerini ve eğitimde kullanılacak optimizasyon algoritmalarını esnek ve programlanabilir yapıya sahip FPGA üzerinde gerçekleştirmek ve farklı problemlerin uygulanmasına imkân verecek biçimde ayarlanabilir mimariye sahip genel amaçlı bir yapay sinir ağı donanımı tasarlamaktır. Bu amaç doğrultusunda öncelikle yapay sinir ağı modellerinden KKFSA ve MLP ağlarının MATLAB ortamında DGA ile eğitiminin sağlanması amaçlanmıştır. Bu çalışma ile sezgisel algoritmalardan biri olan, genetik algoritma tabanlı diferansiyel gelişim algoritmasının FPGA teknolojisi ile gerçeklenerek çok daha hızlı bir şekilde optimizasyon yapmasını sağlamak ve bunun çalışmalarda kullanılabileceğini göstermek amaçlanmıştır.

1.3 **Orjinal Katkı**

Bu çalışmada yapay sinir ağı modellerinden KKFSA'nın eğitimi sezgisel algoritmalardan biri olan DGA ile gerçeklenmiş ve çok boyutlu ve doğrusal olmayan imza tanıma veri kümesi ile analiz edilmiştir.

DGA algoritması FPGA üzerinde gerçeklenerek matematiksel bir denklemin çözümü sağlanmıştır.

Bu çalışma ile DGA ile bir yapay sinir ağının eğitimi, çok boyutlu ve doğrusal olmayan bir problem için yine FPGA üzerinde gerçeklenmiştir.

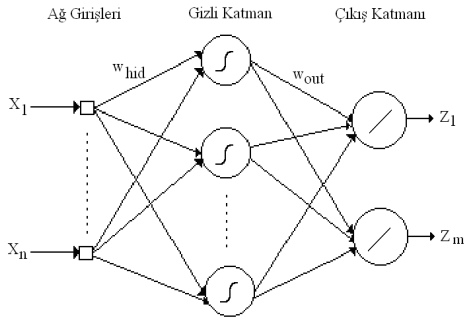
BÖLÜM 2

GENEL BİLGİLER

Bu bölümde yapılan çalışmanın anlaşılabilmesi için gerekli tanımlar yapılmıştır. Kullanılan ağ modelleri, eğitim amaçlı kullanılan DGA ve uygulamada kullanılan FPGA ile ilgili genel tanımlar yapılmıştır.

2.1 Çok Katmanlı Algılayıcılar

ÇKA, günümüzde birçok mühendislik probleminin çözümünde kullanılabilmekte ve sınıflandırma amacıyla etkin şekilde sonuçlar üretebilmektedir. ÇKA ağları, 3 katmandan oluşmaktadır (Şekil 2.1). Bunlar; girdi katmanı, gizli katmanlar ve çıktı katmanıdır. Bilgiler girdi katmanından ağa tanıtılır, gizli katmanlardan çıkış katmanına ulaşır ve çıkış katmanından dış dünyaya aktarılır. Ağ, örneklere bakarak problem uzayında bir çözüm üretir ve bu genellemeye bağlı olarak gelecek yeni örnekler için de çözüm üretebilmektedir [19].



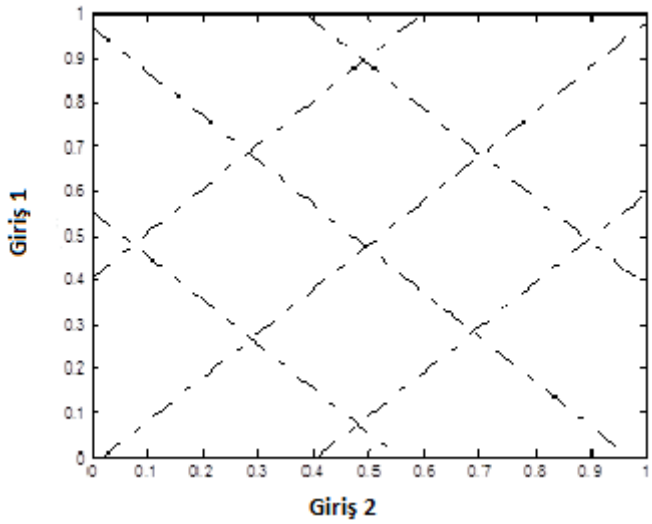
Şekil 2. 1 Çok Katmanlı Algılayıcı Yapısı

Bu ağ yapısında, gizli katmandaki nöronlarda ve çıkış katmanında bulunan nöronlarda ağırlıklı toplam işlemi gerçekleştirilir. Gizli katmanda genellikle sigmoid fonksiyonu veya

hiperbolik tanjant fonksiyonu kullanılır. Bu fonksiyonların türevleri kendileri cinsinden ifade edilebilir böylece dönüşüm işlemlerinin kontrolü kolaylaşır [29].

$$\varphi(v) = \tanh(\frac{v}{2}) = \frac{e^{v}-1}{e^{v}+1} \qquad (2.1)$$

Eşitlik (2.1)'de hiperbolik tanjant fonksiyonunun eşitliği verilmiştir. Burada v, nöronlardaki ağırlıklı toplam işleminin sonucunu ifade eder. Şekil 2.2'de gösterildiği gibi, global aktivasyon fonksiyonuna sahip ÇKA, n boyutlu giriş uzayının sonsuz bir kısmını hiperdüzlemler ile ayırır. Bu ağ yapısının eğitimi katmanlar arasındaki ağırlıkların uygulanan probleme göre optimizasyonu ile gerçekleştirilir [8].



Şekil 2. 2 ÇKA nöronu ile elde edilen karar sınırlarının değişimi [8]

2.2 **KKFSA Yapısı**

KKFSA, ÇKA ve RTF ağlarının yayılım kuralları birleştirilerek oluşturulan bir yapay sinir ağı modelidir. Dorffner tarafından 1994 yılında önerilmiştir [30]. KKFSA'da, ÇKA ve RTF ağlarının özellikler birleştirilerek daha üstün karar sınırları olan bir ağ yapısı oluşturma hedeflenmiştir. KKFSA'nın daha rahat anlaşılması için ÇKA ve RTF ağlarının yapıları ve özellikleri hakkında bilgi verilecektir.

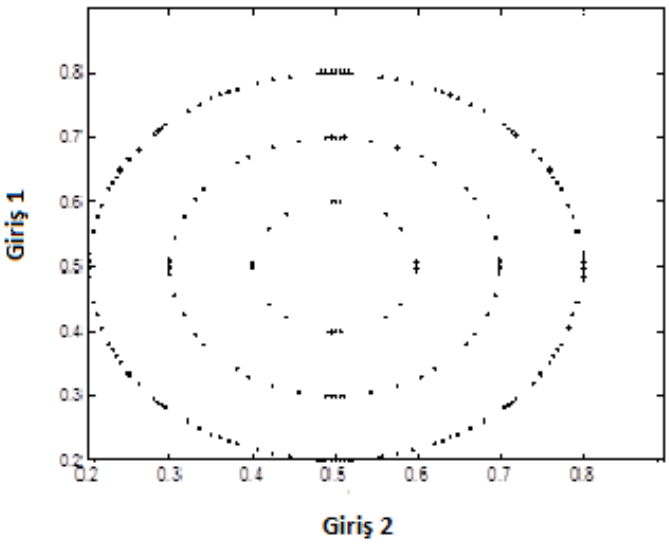
Radyal tabanlı fonksiyon ağları, gizli katmanda bulunan RTF'lerin uygun genişlik ve merkez parametreleri ile eğri uydurma yaklaşımdır. RTF ağı tek gizli katmandan oluşur, gizli katmanda bulunan nöronlar radyal tabanlı fonksiyonlardan oluşmaktadır. RTF ağında, giriş katmanından orta katmana dönüşüm, radyal tabanlı aktivasyon

fonksiyonları kullanılarak doğrusal olmayan sabit bir dönüşümdür. Orta katmandan çıkış katmanına ise doğrusal bir dönüşüm gerçekleştirilir. Radyal tabanlı ağlar geri yayılım ağları ile benzerlik göstermesine rağmen, bu ağların bazı avantajları vardır. RTF ağlarının eğitimi çok daha hızlı gerçekleştirilir ve girişleri durgun olmayan problemlerde gizli katmanda kullanılan radyal tabanlı fonksiyon uygulaması ile çok daha güvenilir sonuçlar vermektedir [31]. RTF ağı ile geri yayılımlı ağların arasındaki en genel fark gizli katmandaki nöronların özellikleri ile oluşmaktadır. Geri yayılım ağlarında genellikle sigmoid veya hiperbolik tanjant fonksiyonu kullanırken, RTF ağlarında genellikle, denklem 2.2'de ifade edildiği gibi standart öklid uzaklıklarını üstel fonksiyondan geçiren Gauss fonksiyonudur veya temel Kernel fonksiyonları kullanılmaktadır.

$$\varphi(\|x-c\|) = \exp(-\frac{\|x-c\|^2}{2\sigma^2}) \qquad (2.2)$$

Bu eşitlikte yer alan x, c, σ sırasıyla ağın giriş vektörlerini, merkez vektörlerini ve gauss fonksiyonun genişliğini ifade eder.

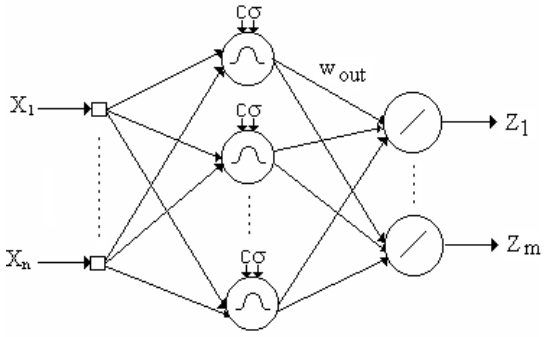
Şekil 2.3'de görüldüğü gibi, RTF ağı lokal aktivasyon fonksiyonuna sahiptir böylece n boyutlu giriş uzayını hiperkürelere ayırır.



Şekil 2. 3 RTF nöronu ile elde edilen karar sınırlarının değişimi [8]

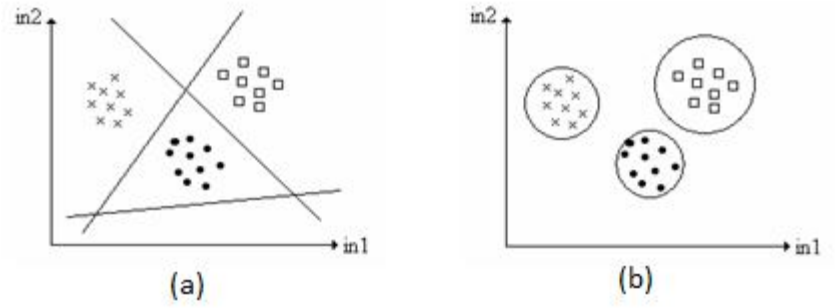
Ağın eğitimi sırasında optimize edilen parametreler merkez vektörleri, gauss fonksiyonunun genişliği ve çıkış katmanındaki ağırlıklardır. Yukarda bahsettiğimiz

aktivasyon fonksiyonları çıkış ağırlıklarının güncellenmesinde kullanılır. Standart bir RTF ağının yapısı şekil 2.4'te gösterilmektedir.



Şekil 2. 4 Standart RTF Ağının Yapısı [8]

Şekil 2.5'de ÇKA ve RTF ağlarının karar sınırları sırasıyla a ve b'de verilmiştir.

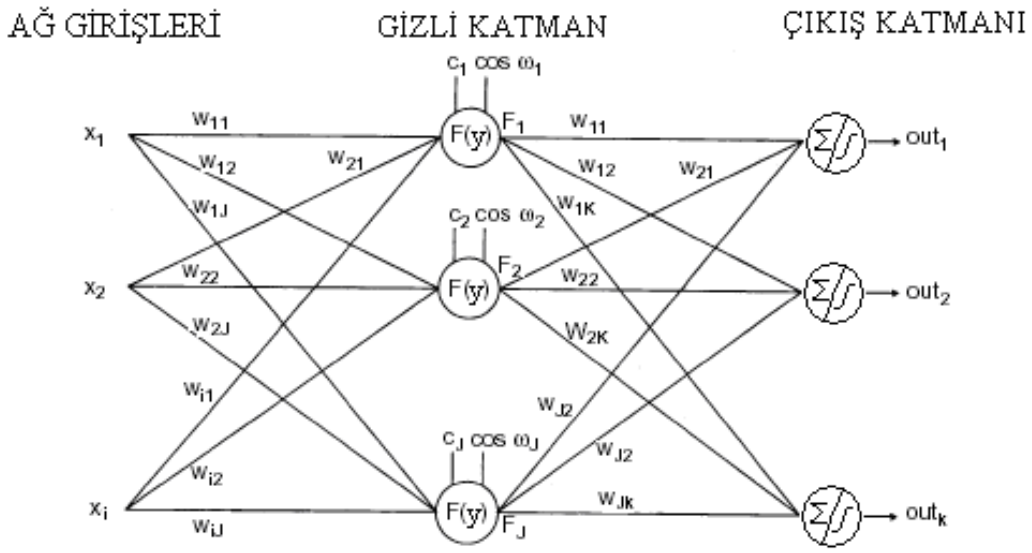


Şekil 2. 5 ÇKA ve RTF ağı tarafından elde edilen karar sınırları [14]

Şekil 2.5'de görüldüğü gibi ÇKA ağı global RTF ağı ise lokal sınıflandırma yapmaktadır. Karar sınırlarının değişimi problemin yapısına göre ağın performansını doğrudan etkilemektedir. KKFSA, uygulanan probleme bağlı olarak karar sınırını otomatik olarak oluşturur. ÇKA ağının global karar sınırı ile RTF ağının lokal karar sınırları KKFSA'nın özel sınırlarını oluşturmaktadır. Düzlemsel veya dairesel karar sınırlarına bağlı kalmadan probleme göre karar sınırı oluşturmak ağın performansının artmasına doğrudan katkı sağlar.

2.2.1 Konik Kesit Fonksiyonlu Sinir Ağı

KKFSA'nın gizli nöronundaki yayılım kuralı, koni ile düzlemin kesişmesi ile oluşan analitik eşitlikler ile belirlenir. KKFSA'nın mimarisi aşağıda gösterilmiştir.



Şekil 2. 6 KKFSA ağ mimarisi [14]

KKFSA'da gizli katman ile çıkış katmanında yer alan nöronlardaki hesaplamalar aynı değildir. Çıkış katmanındaki nöronlarda, geri beslemeli ağ yapılarında olduğu gibi, ağırlıklı toplam işlemi yapılmaktadır. Gizli katmandaki nöronlarda ise farklı olarak KKFSA'nın yayılım kuralını belirleyen hesaplamalar yapılmaktadır. Aşağıdaki eşitlik n boyutlu bir giriş için tanımlıdır.

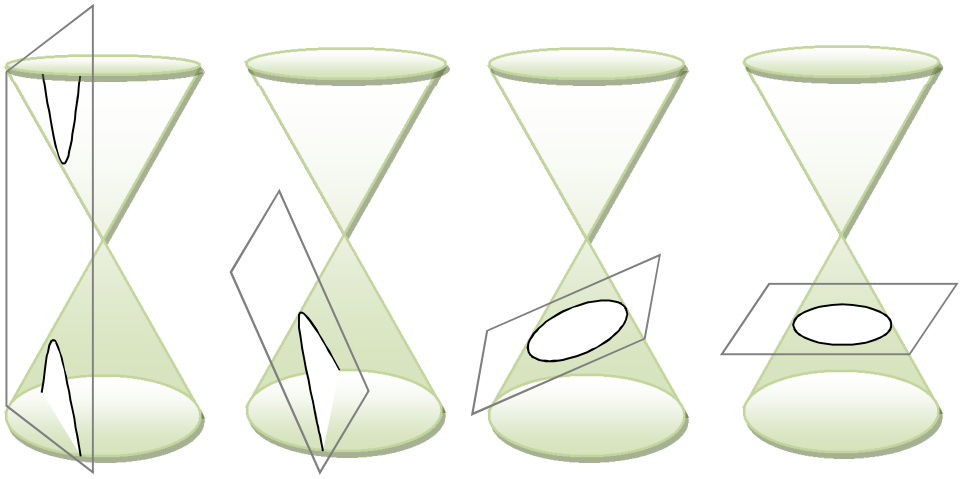
$$u_j^p(x) = \sum_{i=1}^{n} (x_i^p - c_{ij}) w_{ij} - \cos \omega_j \sqrt{\sum_{i=1}^{n} (x_i^p - c_{ij})^2} \qquad (2.3)$$

KKFSA'nın gizli nöronlarında aktivasyon fonksiyonu olarak genelde sigmoid tipi hiperbolik tanjant fonksiyonu kullanılmaktadır.

$$f_j^p(x) = \frac{2}{1 + e^{-2 \cdot u_j^p}} - 1 \qquad (2.4)$$

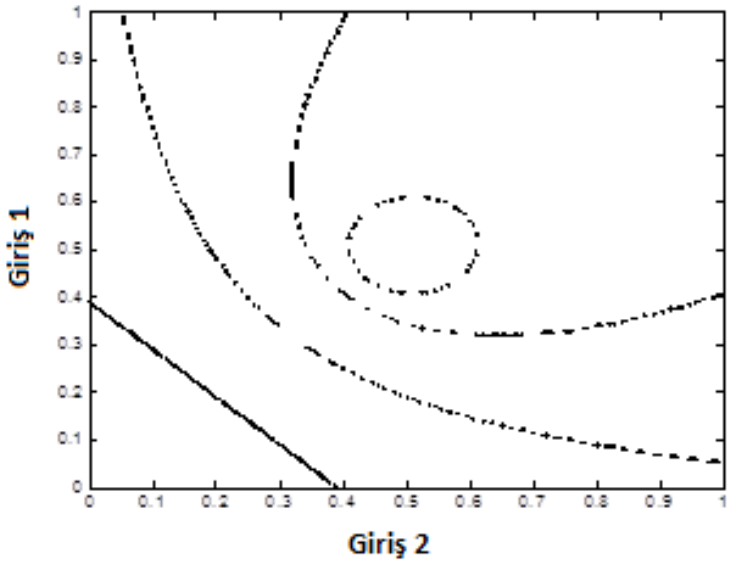
Eşitlik 2.8'de yer alan x_i^p , w_{ij}, c_{ij} değerleri sırasıyla, ağa uygulanan p. örnek için giriş vektörünü, giriş katmanı ile gizli katman arasındaki ağırlık matrisini, j. gizli nöron için merkez vektörünü ifade etmektedir. ω_j, j. gizli nöron için açılım açısını ifade etmektedir. Ayrıca i indisi, giriş katmanındaki nöronları, j indisi ise gizli katmandaki nöronları ifade etmektedir. Eşitlik, RTF ve ÇKA ağlarının yayılım kurallarını kendi bünyesinde büyük ölçüde barındırmaktadır. Eşitlik 2.3'ün birinci kısmında, ağırlıklı toplama işlemi yapılmakta, ikinci kısmında ise öklid uzaklığı hesaplanmaktadır. Eşitlikte yer alan açılım açısının ω_j değişimi ile ağın karar sınırları hiperdüzlem ile hiperküre arasında değişmekte,

açının değerine göre eliptik, parabolik ve hiperbolik düzlemler de oluşmaktadır. Daha önce de ifade edildiği gibi, hiperdüzlem ve hiperküre sırasıyla ÇKA ve RTF'in karar sınırlarını oluşturmaktadır. Koni ile düzlemin kesişmesi ile ortaya çıkan şekiller KKFSA'nın yayılım kuralını belirlemektedir (Şekil 2.7).



Şekil 2. 7 Koni ile düzlemin kesişimiyle oluşan konik kesit düzlemler

Şekil 2.8'de ise KKFSA kullanılarak elde edilen karar sınırlarının değişimi verilmiştir. İki boyutlu uzayda çizgisel (ÇKA), dairesel (RTF), parabolik, hiperbolik ve eliptik karar sınırları tek KKFSA ile elde edilmiştir.



Şekil 2. 8 KKFSA nöronu ile elde edilen karar sınırlarının değişimi [8]

KKFSA'nın eğitimi, probleme bağlı olarak, hata fonksiyonunu minimize edecek şekilde ağ üzerindeki serbest parametrelerin güncellenmesiyle yapılır. KKFSA'da güncelleme yapılarak optimum değerleri bulunan serbest parametre değerleri ağırlık, merkez ve açıdır. Ağın eğitiminde kullanılan optimizasyon algoritmalarında, KKFSA'nın ileri yönlü hesaplamaları baz alınmıştır.

KKFSA'nın ileri yönlü hesaplamaları;

$$\mathbf{u}_j = \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{c}_{ij})\mathbf{w}_{ij} - \cos\omega_j \sqrt{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{c}_{ij})^2} \tag{2.5}$$

$$a_j = f\left(u_j\right) \tag{2.6}$$

$$u_k = \sum_{j=1}^{m} a_j w_{jk} \tag{2.7}$$

$$a_k = f\left(u_k\right) \tag{2.8}$$

Burada a_j ve a_k sırasıyla, gizli katmanının ve çıkış katmanının çıkışını; $f(\cdot)$, sigmoid fonksiyonunu; n ve m, giriş ve gizli katmandaki nöron sayısını ifade etmektedir.

$$E^p = \frac{1}{2}\sum_{k=1}^{r}\left(d_k^p - a_k^p\right)^2 \tag{2.9}$$

Ağın toplam karesel hatası 2.9 eşitliği yardımıyla hesaplanmıştır. Ağın toplam karesel hatası, ağa uygulanan *p.* örnek için elde edilmiştir. Burada *d, p*. örnek için hedef değeri; *r*, çıkış katmanındaki nöron sayısını ifade eder.

Bu çalışmada KKFSA'nın eğitim işlemi için sezgisel algoritmalardan biri olan diferansiyel gelişim algoritması ve hatayı geriye yayılım algoritması kullanılmıştır.

2.3 **Sezgisel Algoritmalar**

Sezgisel algoritmalar son zamanlarda bilgisayar bilimlerinde yaygın olarak kullanılan bir problem çözme tekniğidir. Sezgisel algoritmalarda en iyi çözümün bulunması garanti değildir, fakat bu algoritmalar makul bir sürede bir çözüm elde edeceklerini garanti eder [15-19]. Genellikle bu çözüm en iyiye yakındır, çok hızlı ve kolay bir şekilde bu çözüme ulaşılır. Sezgisel algoritmalara örnek olarak, parçacık sürü optimizasyonu (PSO), yapay arı kolonisi optimizasyonu, genetik algoritma (GA) ve diferansiyel gelişim algoritmaları verilebilir.

2.3.1 Diferansiyel Gelişim Algoritması

Diferansiyel gelişim algoritması (DGA), genetik algoritma tabanlı, popülasyon temelli sezgisel bir global optimizasyon tekniğidir. DGA, Rainer Storn tarafından ortaya atılan Chebyshev polinomsal uyum problemini çözme amacı ile Kenneth Price tarafından geliştirilmiştir [32]. Genetik algoritmalarda çaprazlama, karşılaştırma ve mutasyon işlemleri ayrı ayrı gerçekleştirilir. Bu nedenle optimizasyon için uzun zamana ihtiyaç duyulmaktadır. DGA olarak adlandırılan gelişime dayalı bir strateji önerilerek bu dezavantajın giderilmesi sağlanmıştır.

Algoritmada öncelikle başlangıç popülasyonundaki en iyi kromozom seçilir ve ardından tüm popülasyon, istenilen durdurma kriterleri sağlanıncaya kadar mutasyon, çaprazlama ve karşılaştırma işlemlerine tabi tutulur. Durdurma kriteri, iterasyon sayısı olabileceği gibi belirli bir hata değeri de olabilir.

2.3.2 Diferansiyel Gelişim Algoritmasının Avantajları

DGA, durdurma kriterlerini göz önünde bulundururken, problem objektiflerini modelleyerek objektif fonksiyonunu minimuma götüren bir sezgisel optimizasyon metodudur. Algoritma temel olarak üç avantaja sahiptir. Bunlar;

- Başlangıç parametrelerine bağlı olmaksızın gerçek global minimumun bulunması,
- Bölgesel minimuma hızlı yakınsaması,
- Az sayıda kontrol parametresi kullanmasıdır.

2.3.3 Diferansiyel Gelişim Algoritmasının Temel Adımları

Diferansiyel gelişim algoritması 4 temel adımdan oluşur. Bunlar, başlangıç popülasyonunu oluşturma, mutasyon, çaprazlama ve karşılaştırma işlemleridir [33]. Bu adımlar Şekil 2.9'da görünmektedir.

2.3.3.1 **Başlangıç Popülasyonu**

Diferansiyel gelişim algoritmasında ilk olarak bir sonraki adımlarla gelişimi sağlanacak olan uygulanan probleme bağlı olarak bir başlangıç popülasyonu oluşturulur.

Başlangıçta probleme ait değişken sayısı belirlenir, değişken sayısı her bir kromozoma ait gen (boyut) sayısını belirlemektedir (D). NP ise probleme bağlı olarak kullanıcı tarafından belirlenen kromozom sayısıdır. Her zaman üçten büyük olmalıdır. Çünkü DGA da yeni kromozomların üretilmesi için mevcut kromozom dışında üç adet kromozom gerekmektedir ($r_{1,2,3}$). Başlangıçta NP adet D boyutlu kromozomdan meydana gelen başlangıç populasyonu (P0) aşağıdaki gibi üretilir [33].

$$\forall I \leq Np \wedge \forall j \leq D: x_{j,I,G=0} = x_j^{(l)} + rand_j[0,1].\left(x_j^{(u)} - x_j^{(l)}\right) \quad (2.10)$$

Başlangıç populasyonu üretildikten sonra, aşağıda açıklanan operatörler kullanıcı tarafından belirlenen durdurma kriteri sağlanıncaya kadar tekrarlanarak algoritma tamamlanır. Durdurma kriteri G_{max} veya hatanın istenilen bir seviyeye düşmesi şeklinde belirlenebilir. Son jenerasyondaki en iyi birey çözüm vektörüdür.

2.3.3.2 **Mutasyon**

Mutasyon, mevcut kromozomun bir kısım genleri üzerinde, rasgele belirlenmiş miktarlarda değişiklikler yapmaktır. Bu değişiklikler sayesinde kromozomunun temsil ettiği çözüm noktası, çözüm uzayında hareket etmektedir. Mutasyonun hedefine ulaşabilmesi için, doğru yönde doğru miktarda hareketi sağlayacak değişikliklerin belirlenmesi gerekmektedir.

Diferansiyel gelişim algoritmasında, mutasyon işlemine tabi tutulacak olan kromozom dışında ve birbirlerinden de farklı olan üç kromozom seçilir ($r_{1,2,3}$). Bu uygulamada r_1 kromozomu her iterasyonda x_{best} olarak belirlenmiştir. Seçilen kromozomlardan ilk ikisinin farkı alınır. Daha sonra bu fark kromozomu F parametresiyle çarpılır. F parametresi genellikle 0-2 arasında değerler almaktadır. Elde edilen ağırlıklandırılmış fark kromozomu ile seçilen üçüncü kromozomu (r_3) ile toplanır. Böylece mutasyon sonucu çaprazlamada kullanılacak olan kromozom elde edilmiş olur ($U_{j,I,G}$) [33].

$$\forall j \leq D : U_{j,\mathrm{I},G} = x_{best} + F.\left(x_{j,r_1,G} - x_{j,r_2,G}\right) \quad (2.11)$$

2.3.3.3 **Çaprazlama**

Çaprazlama işlemi tamamlayıcı bir işlemdir ve amacı, var olan amaç vektör parametrelerinden faydalanarak yeni vektörleri oluşturmak suretiyle araştırmanın başarılı olması için yardımcı olmaktır. Sezgisel algoritmalar temelde, oluşturdukları çözüm kümeleri içerisinden en iyi olanı seçerek bir sonuç üretirler. Çaprazlama işlemiyle seçenek sayısının fazla olması ve seçeneklerin çözüm uzayında farklı yerlerden seçilebilmesi sağlanmıştır.

Çaprazlama yapılırken, mutasyon sonucu elde edilen fark kromozomu ve $x_{i,G}$ kromozomu kullanılarak yeni jenerasyona aday, deneme kromozomu ($u_{i,G+1}$) üretilir. Deneme kromozomuna ait her bir gen CR olasılıkla fark kromozomundan 1-CR olasılıkla mevcut kromozomdan seçilir. Diferansiyel Gelişim Algoritmasında eşit olasılık yerine CR olasılığı söz konusudur. 0 ile 1 arasında üretilen rastgele sayı CR' den küçükse gen, ($n_{j,i,G+1}$)'den aksi takdirde mevcut kromozomdan seçilir.

Amaç belirlenen oranda genin yeni fark kromozomundan alınmasıdır. En az bir tane genin üretilen yeni kromozomdan alınmasını garanti etmek amacıyla, ilk olarak rasgele seçilen jrand noktasındaki gen CR' ye bakılmaksızın ($n_{j,i,G+1}$)'den seçilir [33].

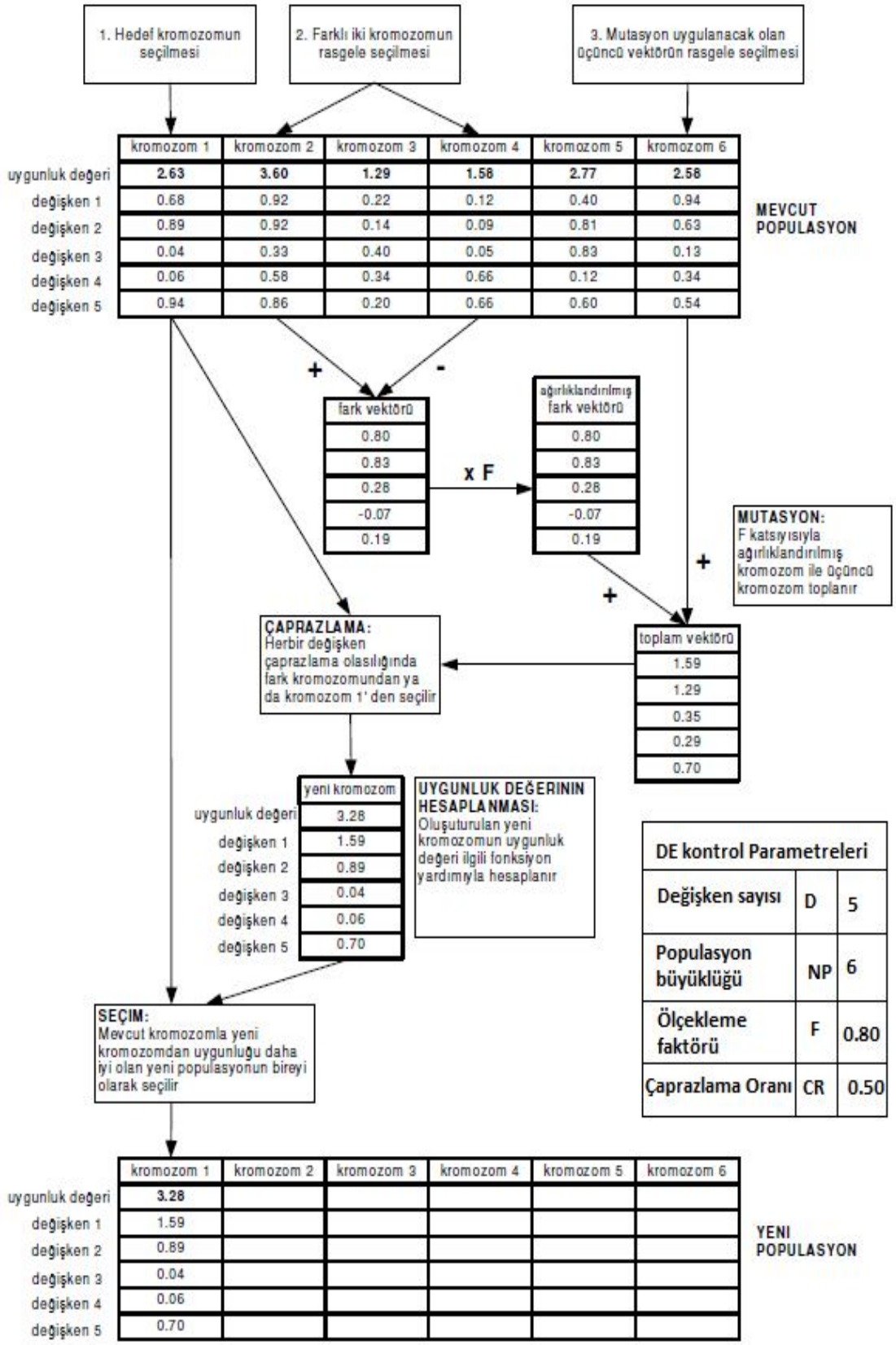
Eğer $rand\,[0,1] \leq CR \ \vee j = j_{rand}$ ise $\forall v_{j,i} = u_{j,\mathrm{I},G}$ (2.12)

2.3.3.4 **Karşılaştırma**

Karşılaştırma işlemi yeni üretilen vektörlerin hangi şartlar altında popülasyona gireceğini tanımlayan bir kriterdir.

Örneğin, bir turnuva karşılaştırma işlemi rastgele seçilmiş vektör çiftleri arasında bir dizi araştırma gerçekleştirmek suretiyle yeni jenerasyonun üyesini belirlemektir. Tipik olarak yarışan vektörler ebeveyn-çocuk popülasyonundan seçilir ve en fazla kazanan vektörlerin gelişmesine müsaade edilir.

DGA'nın karşılaştırma işleminde, yeni üretilen vektör ebeveynine göre daha gelişmiş veya en azından aynı gelişme seviyesinde değilse ebeveyn vektör en az bir jenerasyon daha popülasyonda kalmaya devam etmekte ve başka vektörle yer değiştirmemektedir [33].



Şekil 2. 9 DGA' nın Adımları. Kullanılan Fonksiyon: F(X)=X1+X2+X3+X4+X5 [34]

Çizelge 2. 1 DGA Parametreleri

Sembol	Açıklama
NP	Popülasyon büyüklüğü (kromozom sayısı) NP ≥4 (1, 2, 3, …, i)
D	Değişken sayısı (gen sayısı) (1, 2, 3, …, j)
CR	Çaprazlama oranı [0.1,1.0]
G	Nesil (1, 2, 3, …, Gmax)
F	Ölçekleme faktörü [0.1,2]
$x_{j,i,G}$	G nesli için, i kromozomunun j parametresi (gen)
$n_{j,i,G+1}$	Mutasyon ve çaprazlamaya tabi tutulmuş ara kromozom
$u_{j,i,G+1}$	$x_{j,i,G}$ 'den bir sonraki jenerasyon için üretilen kromozom (child-trial)
$r_{1,2}$	Yeni kromozomun üretilmesinde kullanılacak rasgele seçilmiş kromozomlar $r_{1,2,3}$ ε {1, 2, …, NP} r1 ≠ r_2 ≠ r_3 ≠ i
$x_j^{(u)}, x_j^{(l)}$	Değişkenlerin üst ve alt limitleri

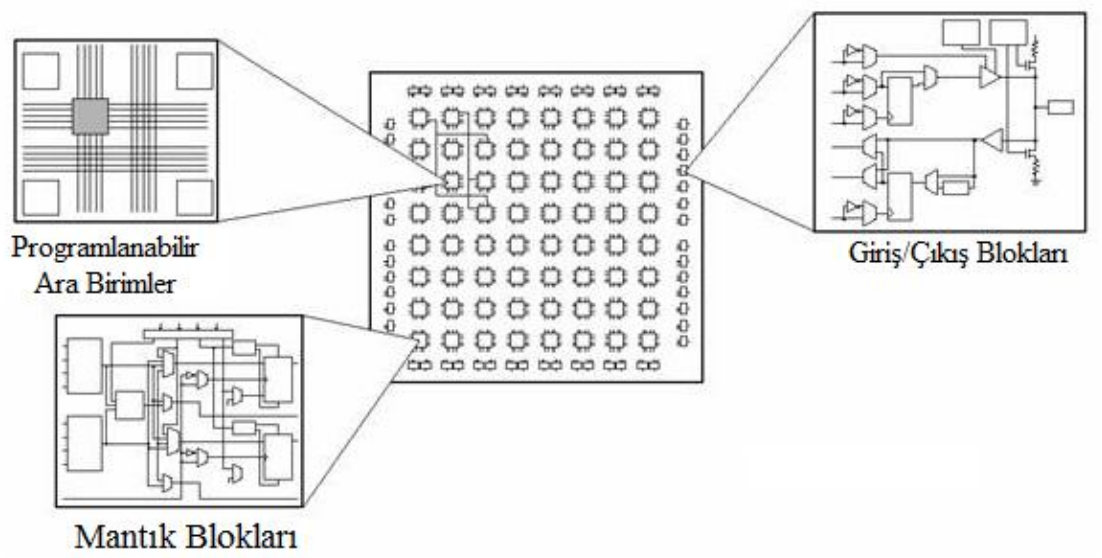
2.4 **Sahada Programlanabilir Kapı Dizileri**

FPGA, "Sahada Programlanabilir Kapı Dizileri" anlamındaki "Field Programmable Gate Array" ifadesinin kısaltılmış şeklidir. FPGA'in yapısı Gate Array ASIC'lere benzer bu nedenle kapı dizileri ismi verilmiştir.

FPGA, üretimi yapıldıktan sonra istenilen fonksiyonlara göre kullanıcı tarafından donanım yapısı tekrar tekrar değiştirilebilen tümleşik devre olarak tanımlanabilir. Kullanıcının uyguladığı probleme göre, içerisindeki trasistörler birbirlerine bağlanarak istenilen yapı gerçeklenebilir. Ancak tanımdan da anlaşılacağı gibi kullanıcı yalnızca sahip olduğu transistor kapasitesi dahilinde çalışma yapabilir.

2.5 **FPGA Teknolojisi**

PLD'nin günümüzdeki en önemli uygulaması FPGA'de mantık hücreleri dizi şeklinde yerleştirilmiştir. CPLD'lerden farklı olarak dizi şeklinde yerleştirilen mantık hücreleri arasındaki bağlar programlanabilir ara bağlantılar sayesinde sağlanır. Bu bağlantılar yatay ve dikey olarak tasarlanmıştır. Bu yapının getirmiş olduğu rahatlık, hücreler arasında ihtiyaç duyulan bağlantı sayısı, mantık hücresi sayısındaki artış ile doğru orantılıdır. Bu özelliği ile FPGA'ler, büyük tasarımlarda CPLD'de karşılaşılan bağlantı sorunları en aza indirilmiş olur [35].



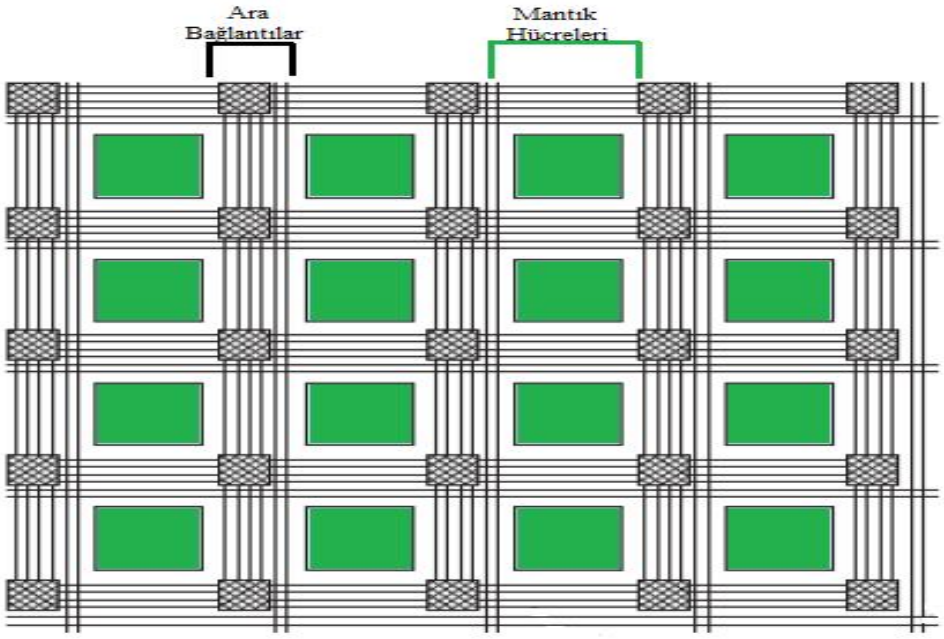
Şekil 2. 10 FPGA Yapısı

Programlanabilir yapısı sayesinde FPGA'ler, gömülü işlemciler ve sayısal sinyal işleme (DSP) blokları, PLL'ler, yüksek hızlı paralel ve seri Giriş/Çıkış desteği, harici bellek ara yüzleri, PCI Express ve Gigabit Ethernet gibi alıcı verici ara yüzleri, gelişmiş saat ve güç dağıtım yöntemleri ile günümüz PLD teknolojisinin en önemli uygulamasıdır.

2.6 **FPGA Mimarisi**

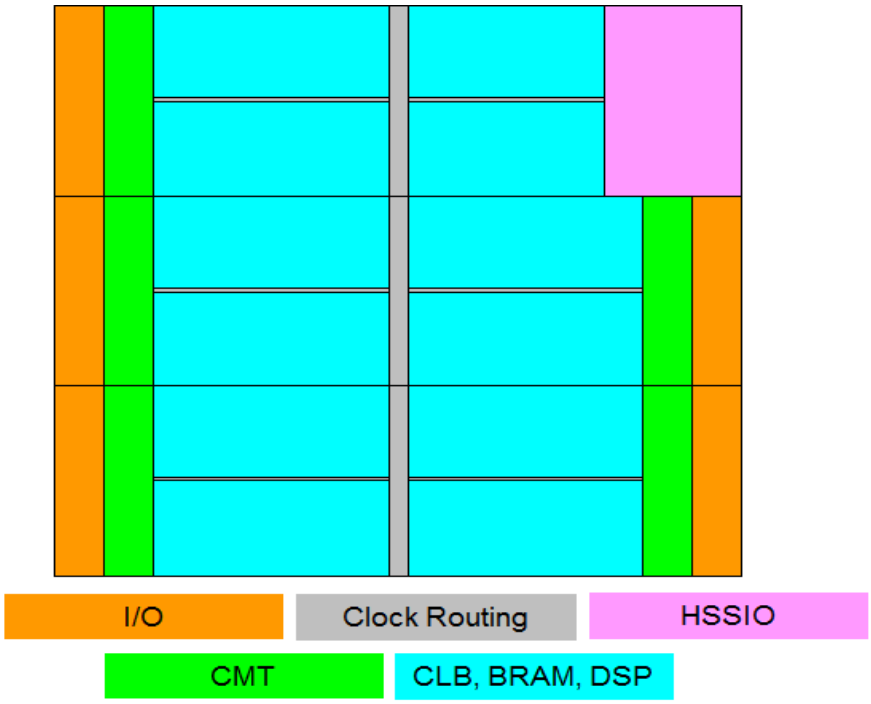
FPGA'ler, mantık hücreleri, Giriş/Çıkış blokları ve programlanabilir ara bloklardan oluşan ve yarı iletken teknolojisiyle üretilen tümleşik devrelerdir. FPGA yapısında, mantık hücreleri, programlanabilir ara bağlantılar ile birbirlerine bağlanmıştır ve bu hücreler FPGA içerisinde matris şeklinde yerleştirilmiştir. FPGA yapısı hem üreticiye hem de ürüne

bağlı olarak değişim gösterir, bu bölümde sadece genel olarak fikir vermek amacıyla en temel yapılardan bahsedilecektir. Bu çalışmada Virtex-7 VC707 geliştirme kiti kullanılmıştır. Bu nedenle FPGA mimarisi ile ilgili bilgi verilirken Xilinx 7 serisi (Artix-7, Kintex-7 ve Virtex-7) üzerinde durulacaktır.



Şekil 2. 11 FPGA Mimarisi

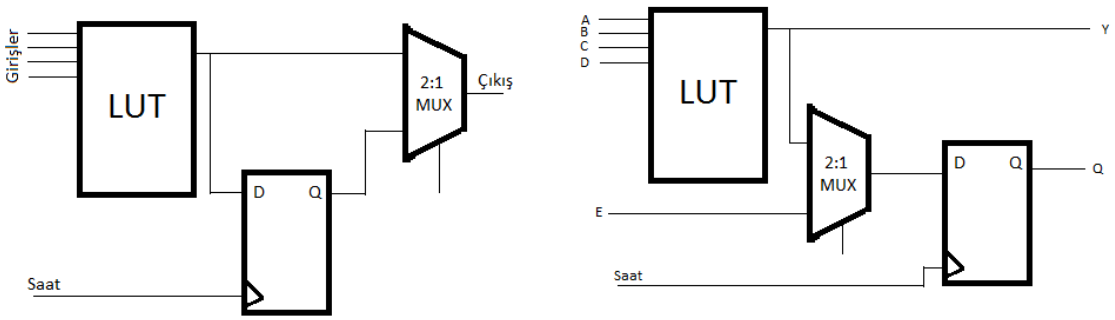
Şekil 2.12'de Xilinx 7 serisine ait FPGA modeli gösterilmiştir. Tüm yapılar iki adet paralel Giriş/Çıkış bloğuna sahiptir, Saat Ayarlama Birimi olan CMT (Clock Management Tile) Giriş/Çıkış biriminin çok hızlı çalışmasına olanak sağlar. Bu yapılarda ayrıca Saat yönlendirme (Clock Routing) birimi ve yüksek hızlı Giriş/Çıkış Birimi olan HSSIO (High-Speed Serial I/O) bulunmaktadır [35].



Şekil 2. 12 Xilinx FPGA modeli [36]

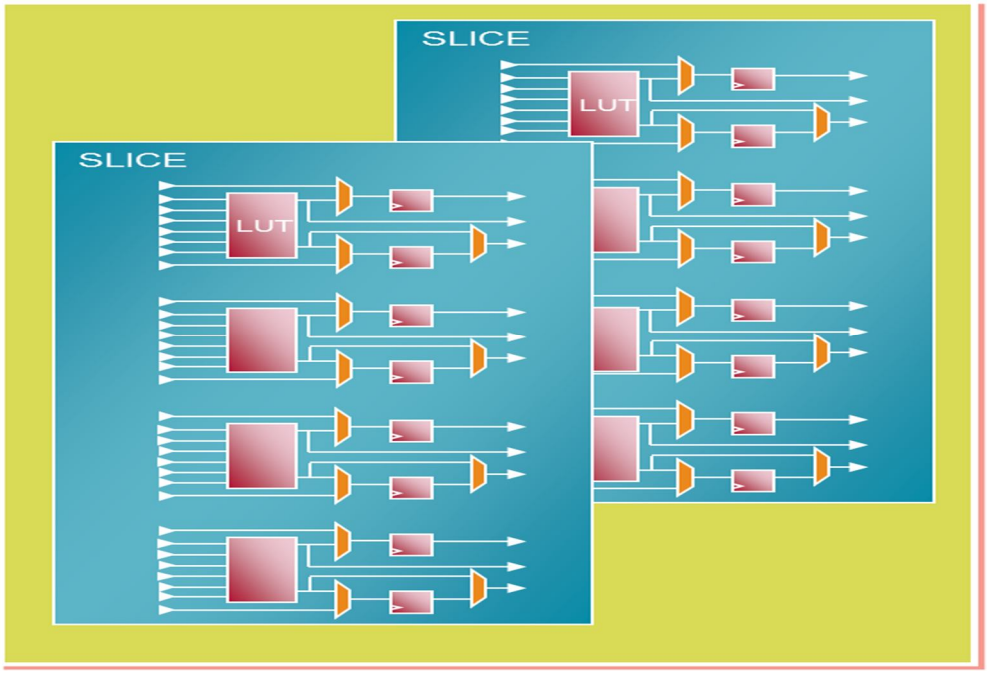
2.6.1 Mantık Bloğu

FPGA'in temel yapısını oluşturan mantık hücrelerinin temelinde, LUT (Look Up Table), çoklayıcı (MUX) ve flip flop bulunur. FPGA'in kapasitesi genellikle bu mantık hücrelerinin sayısı ile belirlenir. Şekil 2.13'te en temel yapıları gösterilmiştir.



Şekil 2. 13 Mantık Hücresi Yapıları

Mantık bloklarında en küçük mantıksal birim, CLB (Configurable Logic Blok) olarak adlandırılır. Şekil 2.14'de Xilinx 7 serisine ait FPGA'lerin CLB yapıları görünmektedir. Her bir CLB iki "Slice"'dan oluşur. Her bir "slice" ın yapısında, altı girişli, dört adet LUT bulunur. Her bir LUT iki adet flip floba bağlıdır. Böylece bir ardışık tasarım oluşturulur.



Şekil 2. 14 CLB Yapısı [36]

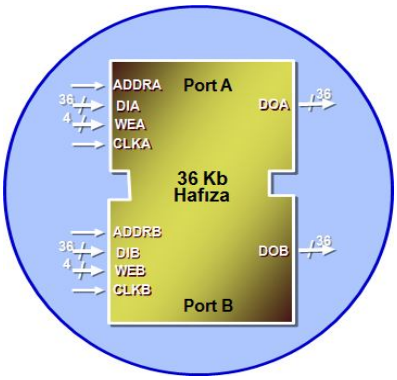
2.6.2 Giriş/Çıkış ve Alıcı/Verici Blokları

Giriş/Çıkış blokları FPGA'in dış dünya ile olan bağlantısıdır ve bu yapıları uygulanan tasarıma göre yalnız giriş, yalnız çıkış veya hem giriş hem çıkış olarak programlanabilir.

Giriş/Çıkış pinleri yanında FPGA yapısında seri-paralel ya da paralel-seri dönüştürücüler, DSP, PLL, ve FIFO blokları ile saat üreteçleri bulunur. Bu alıcı/verici birimler tek yönlü veya çok yönlü olarak kullanılabilir.

2.6.3 Blok RAM

Tasarımlarda küçük veriler için mantık hücreleri arasındaki küçük bellekler kullanılırken, FPGA'in büyük bellek ihtiyaçları Blok RAM'ler ile karşılanır. Şekil 2.15'de Xilinx 7 serisine ait Blok RAM yapısı gösterilmektedir.



Şekil 2. 15 Blok RAM Yapısı [36]

Aşağıdaki tabloda Xilinx 7 serisine ait özellikler verilmiştir.

Maximum Capability	ARTIX-7	KINTEX-7	VIRTEX-7
Mantık Hücreleri Sayısı	20K – 355K	70K – 480K	285K – 2,000K
Blok RAM Sayısı	12 Mb	34 Mb	65 Mb
DSP Sayısı	40 – 700	240 – 1,920	700 – 3,960
Alıcı/Verici Blok Sayısı	4	32	88
Alıcı/Verici Performansı	3.75Gbps	6.6Gbps ve 12.5Gbps	12.5Gbps, 13.1Gbps ve 28Gbps
Bellek Performansı	1066Mbps	1866Mbps	1866Mbps
G/Ç Pin Sayısı	450	500	1,200
G/Ç Voltajı	3.3V ve altı	3.3V ve altı 1.8V ve altı	3.3V ve altı 1.8V ve altı

Şekil 2. 16 Xilinx 7 serisine ait özellikler [36]

BÖLÜM 3

DGA İLE KKFSA VE ÇKA AĞLARININ EĞİTİMİ

Bu çalışmada başlangıç aşaması olarak, DGA'nın yapay sinir ağlarını eğitme işlemi için uygun olup olmadığının belirlenmesi amacıyla, MATLAB ortamında bu eğitme işlemi gerçeklenmiş ve sonuçlar yorumlanmıştır. Bu işlemin gerçekleşebilmesi için öncelikle DGA'nın MATLAB programı ile tasarımı yapılmış daha sonra KKFSA ve ÇKA ağları eğitim için uygun olarak MATLAB ortamında oluşturulmuş ve son olarak bu yapılar birleştirilerek eğitim işlemi gerçekleştirilmiştir.

Bu uygulamada yapay sinir ağları eğitimi ve sonuçları analiz edilirken, çok boyutlu ve doğrusal olmayan bir problem olan imza tanıma veri kümesi kullanılmıştır.

İmza, bazı bürokratik, yetki ve mali işlemler, sözleşmeler ve kredi kartı ödeme bilgisini doğrulamak için kullanılmaktadır. Özellikle mali ve ticari kuruluşlar imzanın görünümüne bakarak güvenlik ve kimlik doğrulama yapmaktadırlar. Ancak imzalar, kişilerin psikolojik durumlarına göre değişim göstermektedir. Veri kümesi oluşturulurken aynı kişiden farklı zaman ve koşullarda elde edilen imza örnekleri farklı olabilmektedir. Görünümdeki bu farklardan dolayı imza tanıma ve doğrulama işlemi kolay olmamaktadır.

Literatürde, imza tanımaya yönelik birçok başarılı çalışma bulunmaktadır. Bu çalışmaların ortak amacı dolandırıcılığı engellemek veya tespit etmektir.

İmza tanıma sürecinde, imzadan elde edilen özellikler daha önce kaydedilmiş özelliklerle kıyaslanır. Bu işlem için saklı markov model, destek vektör makineleri, doğrusal ayırma analizi yöntemlerine yoğunlukla başvurulmuştur [37],[38]. Bunların yanında, yapay sinir

ağları (YSA), iyi bir sınıflayıcı olmasından dolayı birçok imza tanıma uygulamasında kullanılmıştır [19],[23],[24].

3.1 DGA'nın YSA Eğitimindeki Performansı

Bu çalışmada, bir KKFSA ve ÇKA ağı, DGA ile eğitilmiş ve ağın performansı çok boyutlu, doğrusal olmayan bir problem olan imza tanıma veri kümesi ile analiz edilmiştir. Başlangıç ağırlık değerlerine bağlı olmayan ve lokal minumuma takılmayan DGA, global optimizayon gerçekleştirmektedir. Popülasyon temelli sezgisel bir optimizasyon tekniği olan DGA'nın ağın imza tanıma problemi için eğitimindeki performansı, gradyant temelli geriye yayılım algoritmasının (GYA) eğitim performansı ile karşılaştırılmıştır. İmza tanıma veri kümesi için DGA ile eğitilmiş KKFSA ve ÇKA ağı'nın sınıflama başarısının, GYA ile eğitilmiş olandan daha yüksek olduğu gözlenmiştir.

Bu uygulamada iki ağ yapısınında eğitiminde aynı imza data seti kullanılmıştır. İmza tanıma probleminin sınıflandırma performansı, eğitim kümesi içinde ağa tanıtılan imzalar haricindeki imzaların kimliklendirme oranları değerlendirilerek belirlenmiştir. İmza tanıma işleminin performansı, imzalar arasındaki farkı belirleyici karakteristik özelliklerin doğru bir şekilde çıkarılmasına bağlıdır. Ağa uygulanan, gerçek zamanlı olmayan imza tanıma veri kümesi, [22] tarafından oluşturulmuştur. Veri kümesi elde etmek için 8 kişiden 32'şer adet imza alınmış, her kişinin attığı imzadan 25 tanesi ağın eğitiminde 7 tanesi test işleminde kullanılmıştır. Veri kümesi 8 sınıfta tanımlanmıştır. Bu çalışmada ağa uygulanan imzaların yatay merkezi, düşey merkezi, taban çizgisi kayması ve grid bilgisi özellik vektörünü oluşturmaktadır. Bu durumda, özellik vektörünün boyutu 99 olmaktadır. Özellik çıkartma sürecinin ardından 99x200 boyutunda eğitim kümesi, 99x56 test kümesi elde edilmektedir. Özellik vektörünün oldukça büyük olması, yazılım ortamında işlem hızının yavaşlamasına neden olur. Ayrıca ağ boyutu büyüdükçe ağın donanım gerçeklemelerine uygunluğu azalmaktadır. Özellik vektörünün boyutunu azaltmak amacıyla veri kümesine temel bileşenler analiz tekniği uygulanmıştır. 99 olan özellik vektörünün boyutu, TBA uygulamasının ardından 10'a indirgenmiştir [8].

3.1.1 DGA'ın MATLAB Ortamında Gerçeklenmesi

Algoritmada öncelikle başlangıç popülasyonundaki en iyi kromozom seçilir ve ardından tüm popülasyon, istenilen durdurma kriterleri sağlanıncaya kadar mutasyon, çaprazlama ve karşılaştırma işlemlerine tabi tutulur. Durdurma kriteri, iterasyon sayısı olabileceği gibi belirli bir hata değeri de olabilir.

Algoritmanın temel adımları ise aşağıda adım adım belirtilmiştir.

1. Adım: Algoritma kontrol parametrelerinin belirlenmesi. (D, Gmax, NP, F, CR)

2.Adım: Başlangıç popülasyonun oluşturulması**.**

- NP büyüklüğündeki popülasyonun i. bireyi için D parametre sayısı büyüklüğünde vektörel gösterim oluştur.

3.Adım: Popülasyondaki en iyi bireyi belirle Xbest

4.Adım: Durdurma kriteri sağlanana kadar mutasyon, çapraz-lama ve karşılaştırma.

5.Adım: Mutasyon.

- r_1, r_2, r_3 değerlerini rastgele (1..NP) tamsayı sınırında birbirlerine ve i değerine eşit olmayacak biçimde seç.

- rastgele (1..D) aralığında tamsayı seç.

- her j < D için aday bireyleri belirle.

- $V_{j,i,G+1} = X_{best} + F * (X_{j,r1,G} - X_{j,r2,G})$

6.Adım: Çaprazlama.

- her bir değişkeni çaprazlama oranında $X_{j,i}$ veya $V_{j,i}$ bireyinden seçerek $U_{j,G+1}$ (yeni nesil) oluştur.

7.Adım: Karşılaştırma.

- Eğer $f(U_{i,G+1}) <= f(X_{i,G})$ ise $X_{i,G+1} = U_{i,G+1}$

- Diğer durumlarda $X_{i,G+1} = X_{i,G}$

8.Adım: Bitir.

3.1.2 Uygulama ve Analiz

DGA algoritması ile KKFSA ve ÇKA'nın eğitimini gerçekleştirmek için eşitlik 3.2'de verilen ortalama karesel hata fonksiyonunun [29] minimize edilmesi amaçlanmıştır. Burada tanımlanan hata fonksiyonunda, hedeflenen çıkıştan 'd', ağın çıkışının 'y' çıkarılarak elde edilmiş olan anlık hataların 'e' kareleri toplanarak veri seti üzerinden ortalaması elde edilmektedir.

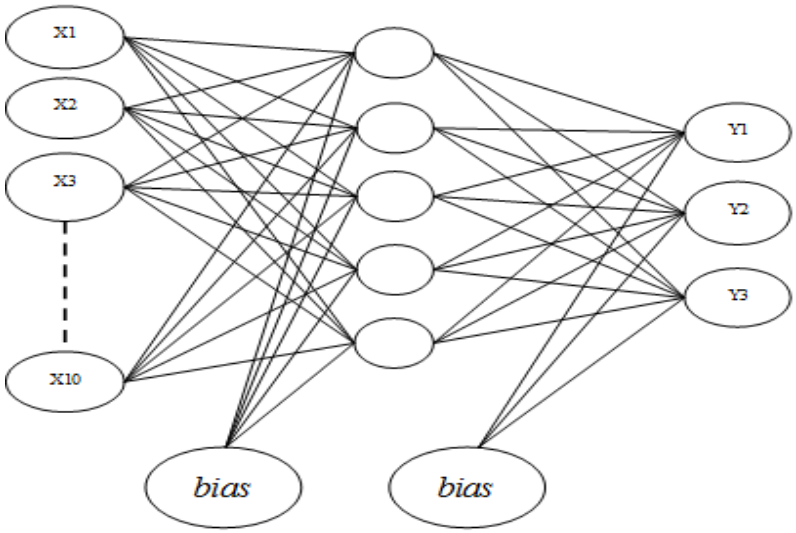
$$\varepsilon_{av}(n) = \frac{1}{2N}\sum_{n=1}^{N}\sum_{i}\left[e_i^2(n)\right] \tag{3.1}$$

$$e_i(n) = d_i(n) - y_i(n) \tag{3.2}$$

Yukarıdaki eşitliklerde 'n' eğitim kümesindeki örnekleri, 'i' çıkış nöronlarını, N ise eğitim kümesindeki toplam örnek sayısını ifade etmektedir. Gradyan temelli olan geri yayılım algoritmasında ağırlıkların güncellenmesinde eşitlik 3.3'deki ifade kullanılmaktadır [29]. Burada, α momentum sabiti, μ öğrenme oranı ve δ lokal gradyan olarak tanımlanmaktadır. DGA'dan farklı olarak ağın başlangıç ağırlıkları (w_0) belirli değerler arasında sınırlandırılmıştır.

$$\Delta w_{i,j}^{k} = \alpha\Delta w_{i,j}^{k-1} + \mu\delta_i^k y_j^k \tag{3.3}$$

İmza verisi tanıma için tasarlanan ÇKA, giriş nöron sayısı 10, gizli nöron sayısı 5, çıkış nöron sayısı 3 ve bias değeri (bias=1) olacak şekilde tasarlanmıştır. Tasarımdan dolayı bu ağda optimizasyon yapılması gereken 73 adet ağırlık bulunmaktadır. Bu tasarlanan ÇKA'nın eğitimi, sezgisel bir global optimizasyon yöntemi olan DGA ile gerçeklenmiştir. Başka bir deyişle algoritmada her bir kromozomda 73 adet gene karşı düşen ÇKA'nın ağırlıkları probleme uygun olarak optimize edilmiştir. Şekil 3.1'de kullanılan ağ yapısı görülmektedir.



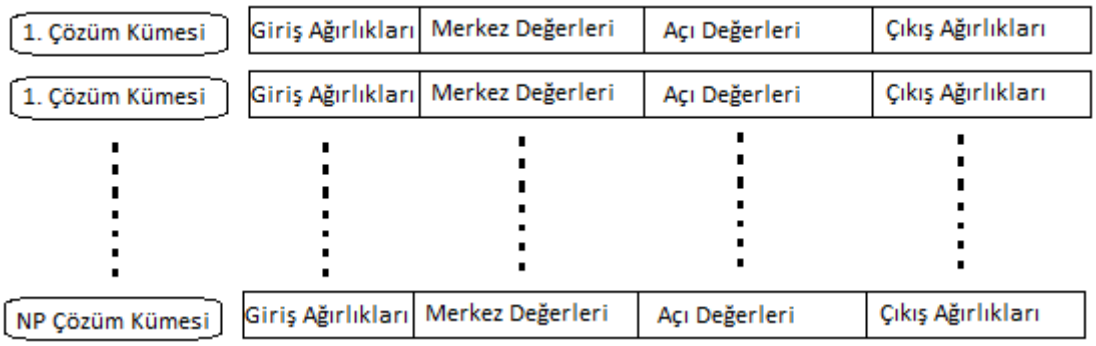
Şekil 3. 1 Uygulamada kullanılan ÇKA modeli

Bu yapıyı KKFSA için düşünürsek, KKFSA'da ağırlıkların yanı sıra açı ve merkez değerlerininde optimizasyonunun yapılması gerekmektedir. Giriş sayısı n, gizli katmandai nöron sayısı m ve çıkış katmanındaki nöron sayısı k olan bir KKFSA yapısı için, $m.(k+n)$ adet ağırlık değeri, $(m \times n)$ adet merkez değeri ve m tane açı değerinin optimize edilmesi gerekmektedir. Yani toplamda optimize edilmesi gereken değer sayısı $m.(2n+k+1)$ tanedir.

Bu çalışmada oluşturulan KKFSA mimarisi 10 giriş, 12 gizli nöron ve 3 çıkış nöronu bulunmaktadır. Ağın mimarisine bakıldığında 156 ağırlık değeri, 120 merkez değeri ve 12 açı değeri olmak üzere toplamda 288 değer optimize edilmiştir.

KKFSA'nın eğitimi için kullanılan DGA kullanılırken, eğitime başlanması için başlangıç çözüm kümeleri oluşturulmalı ve daha sonra eğitimdeki adımlar gerçeklenmelidir. Bu çalışma için oluşturulan KKFSA mimarisi göz önüne alınarak, çözüm kümeleri vektör şeklinde oluşturulmuş ve bu şekilde sisteme uygulanmıştır.

Şekil 3.2'de KKFSA'nın eğitiminde kullanılacak olan DGA için oluşturulmuş başlangıç çözüm kümesinin yapısı görülmektedir. Her bir çözüm kümesi 1 X 288 vektör ile ifade edilmiştir.



Şekil 3. 2 KKFSA mimarisi için oluşturulan başlangıç çözüm kümesi

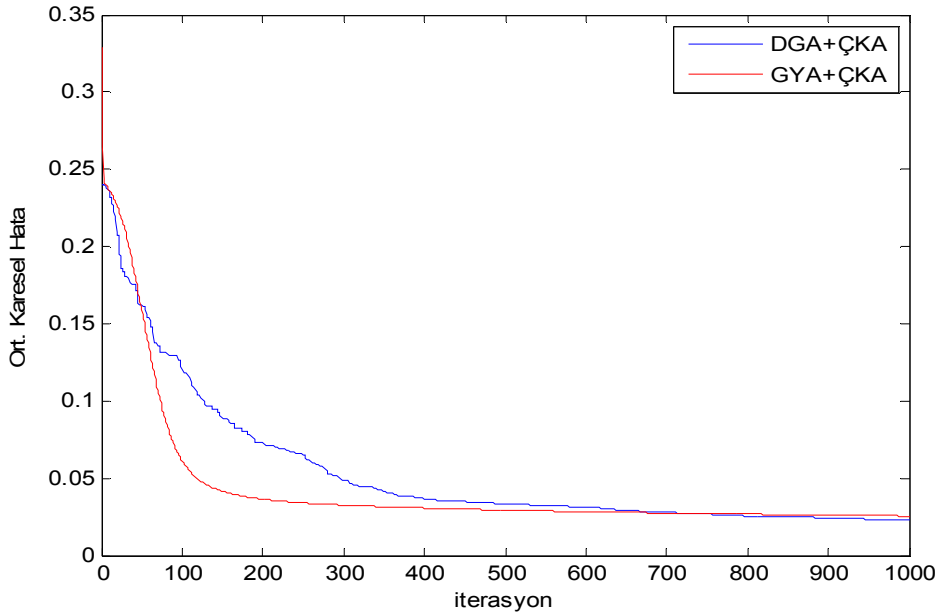
3.1.3 Simulasyon Sonuçları

Bu çalışmada, çok boyutlu doğrusal olmayan bir veri kümesine sahip imza tanıma probleminin DGA ile eğitilen KKFSA ve ÇKA'ya uygulanması gerçeklenmiştir. Sonraki bölümlerde önerilen sistemin gömülü bir sistem üzerinde tasarımı yapılması amaçlanmıştır. Bu nedenle bu algoritmaların yazılım MATLAB programında herhangi bir hazır kod veya kütüphane kullanılmadan donanıma uygun bir şekilde yalın olarak yazılmıştır.

3.1.3.1 DGA İle ÇKA Eğitimi

ÇKA'nın eğitim işleminde eğim düşümü yöntemine dayanan GYA kullanılarak ağın performansı, DGA'nın performansı ile karşılaştırılmıştır. ÇKA'nın DGA ile eğitimine geçmeden önce belirlenen sınırlar içinde rastgele seçilen çözüm kümeleri oluşturulmuştur. Her bir çözüm kümesi DGA algoritmasında kromozomu, kromozomlar da popülasyonu oluşturmaktadır. Ayrıca kromozom içerisindeki her bir değişkende gen olarak adlandırılmıştır.

DGA ile eğitilen ÇKA'nın sınıflama başarısı %94.5 iken GYA ile eğitilen ÇKA'nın sınıflama başarısı bu değerin altında kalmıştır. Aynı şekilde DGA ile eğitilen ÇKA'nın ortalama karesel hatası, GYA ile eğitilen ÇKA'nın ortalama karesel hatasından daha azdır. Ayrıca GYA, başlangıç ağırlık değerlerine bağlı olduğundan ve lokal minimum noktalarına takılabildiğinden performansı değişim göstermekte iken DGA'da bu gibi sorunlar ile karşılaşılmamaktadır. Dolayısıyla DGA ile eğitilen ÇKA'nın performansı daha kararlıdır.



Şekil 3. 3 Hata Fonksiyonunun İterasyon Sayısına Göre Değişimi [19]

Şekil 3.2'de, geri yayılım algoritması ile optimizasyon yaparken minimum hataya çok daha hızlı bir şekilde ulaşılabilindiği görülmektedir. Ancak GYA'nın bu performansı gösterebilmesi başlangıç değerleri ve diğer kontrol parametrelerine çok bağlıdır. Uygulanacak parametrlerin belirlenmesi uzun zaman alabilir. Sezgisel algoritmalarda başlangıç parametlerinin bu derece etki ettiğini söyleyemeyiz. İterasyon sayısı arttırılarak gerçek çözüm kümesine yakın değerler elde edilebilir.

ÇKA'nın GYA ile eğitilmesinde kullanılan kontrol parametreleri Çizelge 3.2'te verilmiştir.

Çizelge 3. 1 ÇKA'nın eğitiminde GYA'nın kontrol parametreleri [19]

μ	α	w_0	İterasyon Sayısı
0.5	0.2	[-0.5 +0.5]	1000

ÇKA'nın eğitiminde kullanılan DGA'nın kontrol parametreleri Çizelge 3.1'de verilmiştir.

Çizelge 3. 2 ÇKA'nın Eğitiminde DGA'nın Kontrol Parametreleri [19]

NP	F	CR	İterasyon Sayısı
20	0.6	0.6	1000

İmza tanıma problemi için çok katmanlı algılayıcıların DGA ile GYA için eğitim ve test kümesindeki sınıflandırma performansları ve ortalama karesel hata değeri Çizelge 3.3'te verilmiştir.

Çizelge 3. 3 Sınıflama Performansı ve Ortalama Karesel Hata [19]

Yöntem	Ortalama karesel hata	Eğitme başarısı(%)	Test Başarısı(%)
DGA+ÇKA	0.0193	94.5	80.3571
GY+ÇKA	0.0255	92.5	78.5714

Çizelge 3.3'te görülen sonuçlar, yukarıdaki parametre değerleri girilerek ÇKA ağının GYA ve DGA ile 10 defa eğitilmesi ve çıkan sonuçların ortalaması alınarak oluşturulmuştur.

3.1.3.2 **DGA İle KKFSA Eğitimi**

KKFSA'nın eğitim işlemi yapılırken optimizasyonu yapılan parametreler çizelge 3.4'de gösterilmiştir.

Çizelge 3. 4 KKFSA eğitim parametreleri

Sembol	Açıklama
w_{ij}	i. giriş ve j. saklı nöron arasındaki ağırlık değeri
w_{jk}	j. saklı nöron ve k. çıkış nöronu arasındaki ağırlık değeri
ω_j	j. gizli nöron için açı değeri
c_{ij}	i. giriş ve j. saklı nöron arasındaki merkez değeri
x	Girişin vektör formu
a_j	j. saklı nöron
a_k	k. çıkış nöronu

Aşağıda KKFSA'nın GYA ile eğitim işleminin aşamaları pseudo kod şeklinde gösterilmiştir.

1. /* w_{ij}, w_{jk}, ω_j, c_{ij} all generated
2. /* for each train input
3. /*$\Delta w_{ij} \leftarrow (w_{ij})_{new} - (w_{ij})_{old}$
4. /*$\Delta w_{jk} \leftarrow (w_{jk})_{new} - (w_{jk})_{old}$
5. /*$\Delta c_{ij} \leftarrow (c_{ij})_{new} - (c_{ij})_{old}$
6. /*$\Delta \omega_j \leftarrow (\omega_j)_{new} - (\omega_j)_{old}$
7. for ep_num =1 to Epoch_Number do
8. for tr =1 to Train_Set_Size do
9. error $\leftarrow$ d_train $- a_k$
10. $\delta_k \leftarrow$ error . derivative_act_func (u_k) for all Output_Neurons
11. $\delta_j \leftarrow$ derivative_act_func(u_j)* w_{jk}*δ_k for all Hidden_Neurons
12. /* Calculation Δw_{jk}, Δw_{ij}, Δc_{ij}, $\Delta \omega_j$ for p. input
13. $\Delta w_{jk}^{p} \leftarrow \gamma * a_j * \delta_k$
14. $\Delta w_{ij}^{p} \leftarrow \gamma * (x_i - c_{ij}) * \delta_j$
15. $\Delta c_{ij}^{p} \leftarrow \gamma * [-w_{ij} + \cos(\omega_j).*(x_i - c_{ij})./ \text{distance}(x_i, c_{ij})]$
16. $\Delta \omega_j^{p} \leftarrow \gamma * \delta_j .* \sin(\omega_j) .* \text{distance}(x_i, c_{ij})$
17. /* Updating Δw_{jk}, Δw_{ij}, Δc_{ij}, $\Delta \omega_j$ for (p+1). input
18. /* with momentum coefficient
19. $\Delta w_{jk}^{p+1} \leftarrow \Delta w_{jk}^{p} + \alpha * \Delta w_{jk}^{p-1}$
20. $\Delta w_{ij}^{p+1} \leftarrow \Delta w_{ij}^{p} + \alpha * \Delta w_{ij}^{p-1}$
21. $\Delta c_{ij}^{p+1} \leftarrow \Delta c_{ij}^{p} + \alpha * \Delta c_{ij}^{p-1}$
22. $\Delta \omega_j^{p+1} \leftarrow \Delta \omega_j^{p} + \alpha * \Delta \omega_j^{p-1}$

23. end for

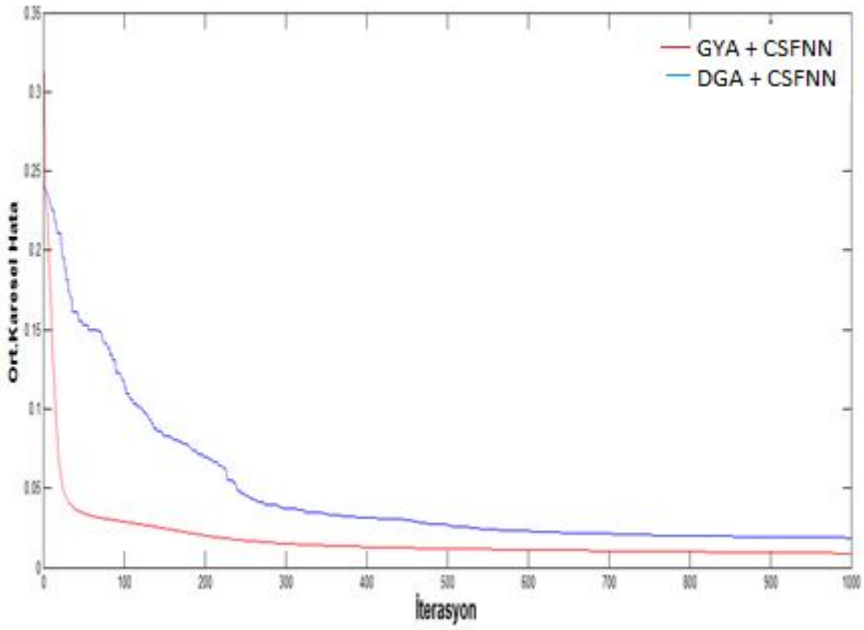
24. end for

KKFSA'nın DGA ile eğitimi sonucunda sınıflama başarısı %83.75 iken eğitim GYA ile yapıldığında sınıflama başarısı bu değerin altında kalmıştır. Aynı şekilde DGA ile eğitilen KKFSA'nın ortalama karesel hatası, GYA ile eğitilen KKFSA'nın ortalama karesel hatasından daha azdır. Ayrıca GYA, başlangıç ağırlık değerlerine bağlı olduğundan ve lokal minimum noktalarına takılabildiğinden performansı değişim göstermekte iken DGA'da bu gibi sorunlar ile karşılaşılmamaktadır (Çizelge 3.5). Dolayısıyla DGA ile eğitilen KKFSA'nın performansının daha kararlı olduğu görülmüştür.

Çizelge 3. 5 KKFSA'nın eğitim sonuçları

KKFSA+DE				KKFSA+BPA		
Eğitimler	Min. OKH	Eğitim başarısı (%)	Test başarısı (%)	Min. OKH	Eğitim başarısı (%)	Test başarısı (%)
1	0.0301	96.5	83.9286	0.0210	99	80.3571
2	0.0298	96	83.9286	0.0198	98	82.1429
3	0.0280	96.5	85.7143	0.0205	98.5	83.9286
4	0.0280	95	82.1429	0.0487	86	78.5714
5	0.0227	96	85.7143	0.0200	99	80.3571
6	0.0320	96	82.1429	0.0160	98	83.9286
7	0.0298	96.5	82.1429	0.0165	98.5	83.9286
8	0.0296	97	85.7143	0.0180	97	83.9286
9	0.0340	95	82.1429	0.0305	90	78.5714
10	0.0304	96	83.9286	0.0180	99	82.1429
Ortalama	0.0295	96.05	83.75	0.0229	96.3	81.7857

Sonuçlara bakıldığında eğitimde kullanılan GYA'nın lokal minimumlara takılma ihtimalinin daha fazla olduğu görülmektedir (Çizelge 3.5'de 4. ve 9. satırlar).



Şekil 3. 4 KKFSA eğitiminde hata fonksiyonlarının değişimi [24]

Şekil 3.4'te görülen hata fonksiyonları, eğitim işleminde kullanılan algoritmalar DGA ve GYA'larının kontrol parametreleri Çizelge 3.6 ve Çizelge 3.7'de verilmiştir.

Çizelge 3. 6 Eğitiminde kullanılan GYA'nın kontrol parametreleri [24]

μ	α	w_0	İterasyon Sayısı
0.5	0.3	[-0.5 +0.5]	1000

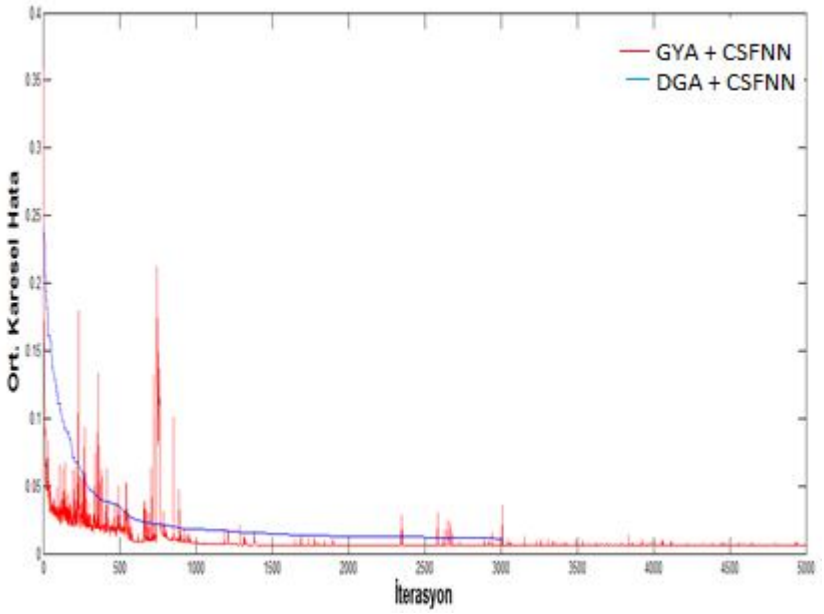
Çizelge 3. 7 Eğitiminde kullanılan DGA'nın kontrol parametreleri [24]

NP	F	CR	İterasyon Sayısı
20	0.8	0.5	1000

Çizelge 3. 8 Sınıflama Performansı ve Ortalama Karesel Hata [24]

Yöntem	Ortalama karesel hata	Eğitme başarısı(%)	Test Başarısı(%)
DGA+KKFSA	0.0193	94.5	80.3571
GY+KKFSA	0.0105	96.5	81.5714

Şekil 3.4'te görüldüğü gibi geri yayılım algoritması, optimizasyon yaparken minimum hataya çok daha hızlı bir şekilde ulaşmaktadır. Ancak bu performans başlangıç değerlerine bağlıdır ve her uygulama için bu değerlerin baştan belirlenmesi gerekmektedir.



Şekil 3. 5 KKFSA eğitiminde hata fonksiyonlarının değişimi

Şekil 3.5'te görülen hata fonksiyonları, eğitim işleminde kullanılan algoritmalar DGA ve GYA'larının kontrol parametreleri Çizelge 3.9 ve Çizelge 3.10'da verilmiştir.

Çizelge 3. 9 Eğitiminde kullanılan GYA'nın kontrol parametreleri

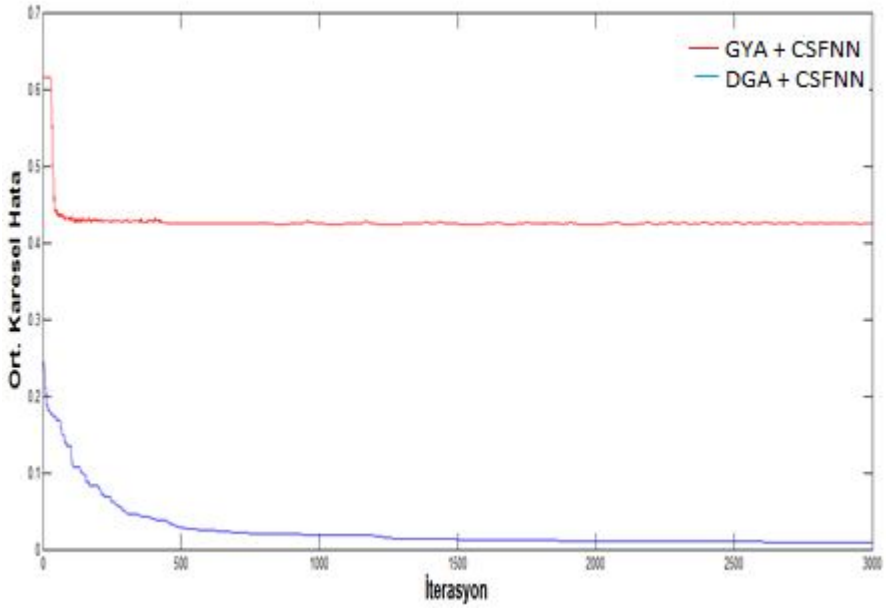
μ	α	w_0	İterasyon Sayısı
1.5	1.5	[-0.5 +0.5]	5000

Çizelge 3. 10 Eğitiminde kullanılan DGA'nın kontrol parametreleri

NP	F	CR	İterasyon Sayısı
20	1.2	0.6	5000

Çizelge 3. 11 Sınıflama Performansı ve Ortalama Karesel Hata

Yöntem	Ortalama karesel hata	Eğitme başarısı(%)	Test Başarısı(%)
DGA+KKFSA	0.0193	97.5	80.3571
GY+KKFSA	0.0190	97.5	79.5714



Şekil 3. 6 KKFSA eğitiminde hata fonksiyonlarının değişimi

Şekil 3.5'te görülen hata fonksiyonları, eğitim işleminde kullanılan algoritmalar DGA ve GYA'larının kontrol parametreleri Çizelge 3.12 ve Çizelge 3.13'de verilmiştir.

Çizelge 3. 12 Eğitiminde kullanılan GYA'nın kontrol parametreleri

μ	α	w_0	İterasyon Sayısı
1.5	1.5	[-5 +5]	3000

Çizelge 3. 13 Eğitiminde kullanılan DGA'nın kontrol parametreleri

NP	F	CR	İterasyon Sayısı
30	1.2	0.6	3000

Çizelge 3. 14 Sınıflama Performansı ve Ortalama Karesel Hata

Yöntem	Ortalama karesel hata	Eğitme başarısı(%)	Test Başarısı(%)
DGA+KKFSA	0.0193	97.5	80.3571
GY+KKFSA	0.0255	23.5	23.2143

BÖLÜM 4

DİFERANSİYEL GELİŞİM ALGORİTMASININ FPGA ÜZERİNDE GERÇEKLENMESİ

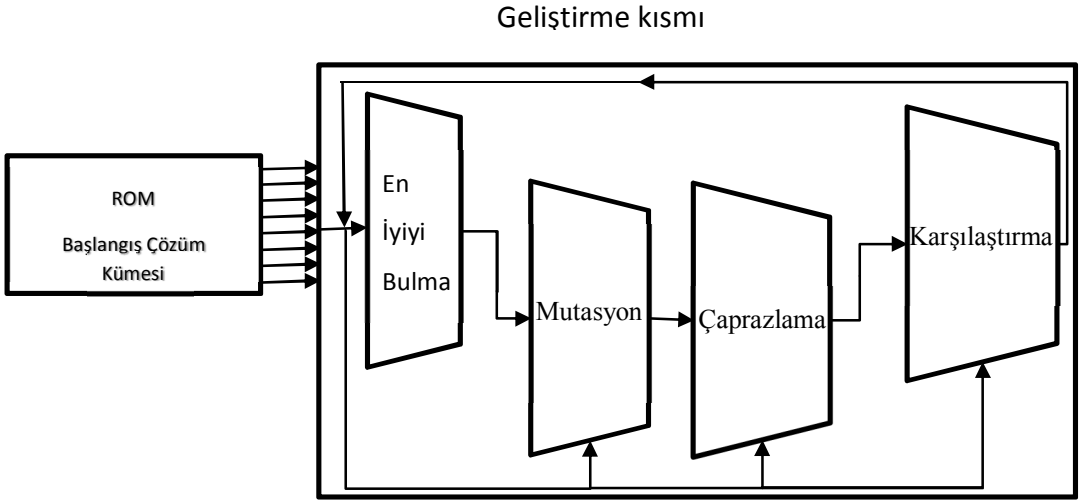
FPGA teknolojisi ile çok karmaşık hesaplar doğru ve çok hızlı bir şekilde yapılabildiğinden kullanım alanı oldukça geniştir [1-4]. FPGA'lerin yapay sinir ağları ve genetik algoritmalar gibi birçok programlama tekniği için uygun olduğu görülmüştür[26-28]. Sezgisel algoritmaların, problem çözümlerinde kullanıldığında çözümün veya çözüme yakın sonuçların bulunabileceği görülmüştür. Bu algoritmalar FPGA ile gerçeklenerek çözümün çok daha hızlı bir şekilde elde edilmesi amaçlanmıştır.

Çalışmanın bu bölümünde DGA, iki bilinmeyenli bir matematiksel problem çözümü için FPGA kitine uygulanmış ve sonuçlar gözlenmiştir. Bu uygulamada en yaygın kullanılan programlama dillerinden biri olan VHDL programlama dili kullanılarak Xilinx firmasının 7 serisine ait olan Virtex-7 VC707 geliştirme kitinin programlanması amaçlanmıştır.

4.1 Problem Çözümü İçin DGA Yapısının FPGA Kitinde Gerçeklenmesi

DGA'nın VHDL diliyle kodlanması sürecinde yapılan işlemlerde kullanılan gerçel sayılar, bilgisayar ortamında gösterim şekillerinden biri olan floating point (kayan nokta) kullanılmıştır. En sık kullanılan IEEE 754 standardına göre şekillendirilen floating point kullanımı ile gerçek dünyada sonsuza giden sayıların bilgisayar ortamına aktarılırken en az etkiyle, gerçeğe en yakın şekilde temsili sağlanmıştır. İşlemlerde tüm sayılar 16 bitlik kayan nokta sistemine dönüştürülmüştür.

DGA, VHDL diliyle gerçeklenmesi 5 ana bileşenden oluşmaktadır. Bunlar, başlangıç çözüm kümelerini oluşturma, en iyinin bulunma işlemi, mutasyon, çaprazlama ve karşılaştırma işlemleri.



Şekil 4. 1 DGA'nın akış diyagramı

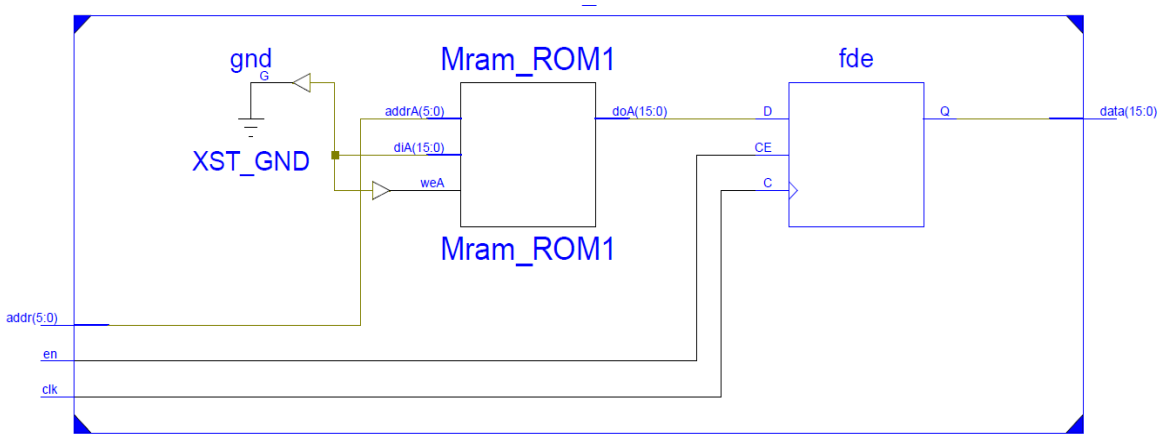
Bu bileşenlerin her biri ayrı ayrı oluşturulmuş ve bir ana modülde birleştirilerek program oluşturulmuştur. Bu uygulamada iki bilinmeyen bir denklemin optimizasyonun yapılması amaçlanmıştır (Eşitlik 4.1).

$$f = \left({X_1}^3 - 50.X_1\right) + \left({X_2}^2 - 10.X_2\right) \quad (4.1)$$

Bu uygulamada, eşitlik 4.1'de verilen fonksiyon için, sonucu sıfıra en yakın çıkan X_1 ve X_2 değerlerinin bulunması hedeflenmiştir.

4.1.1 Başlangıç Çözüm Kümesinin Oluşturulması

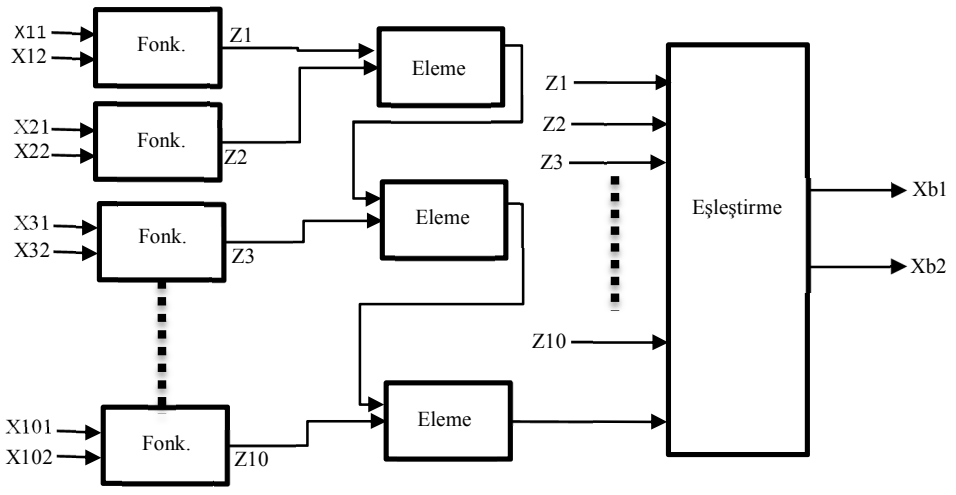
Bu uygulamada başlangıç çözüm kümesi sadece bir defa kullanılmak üzere ROM'a yazılmıştır. Çözüm kümeleri ana modülde romdan alınmış ve istenilen durdurma kriteri sağlanıncaya kadar yeni çözüm kümeleri geliştirme işlemlerine tabi tutulmuştur. Bu uygulamada başlangıçta 10 adet çözüm kümesi oluşturulmuştur. Yani ROM1 kısmında 20 adet 16 bit floating point ile gösterilen sayı bulunmaktadır. Sayıların seçimi girişe uygulanan clc (saat) sinyaline bağlı olarak 6 bitlik addr girişi ile belirlenmiştir. Akış işareti en (enable) ve clk girişlerine bağlı olduğundan, RAM çıkışı D tipi bir flip-flop'a (fde) bağlanmıştır(Şekil 4.2).



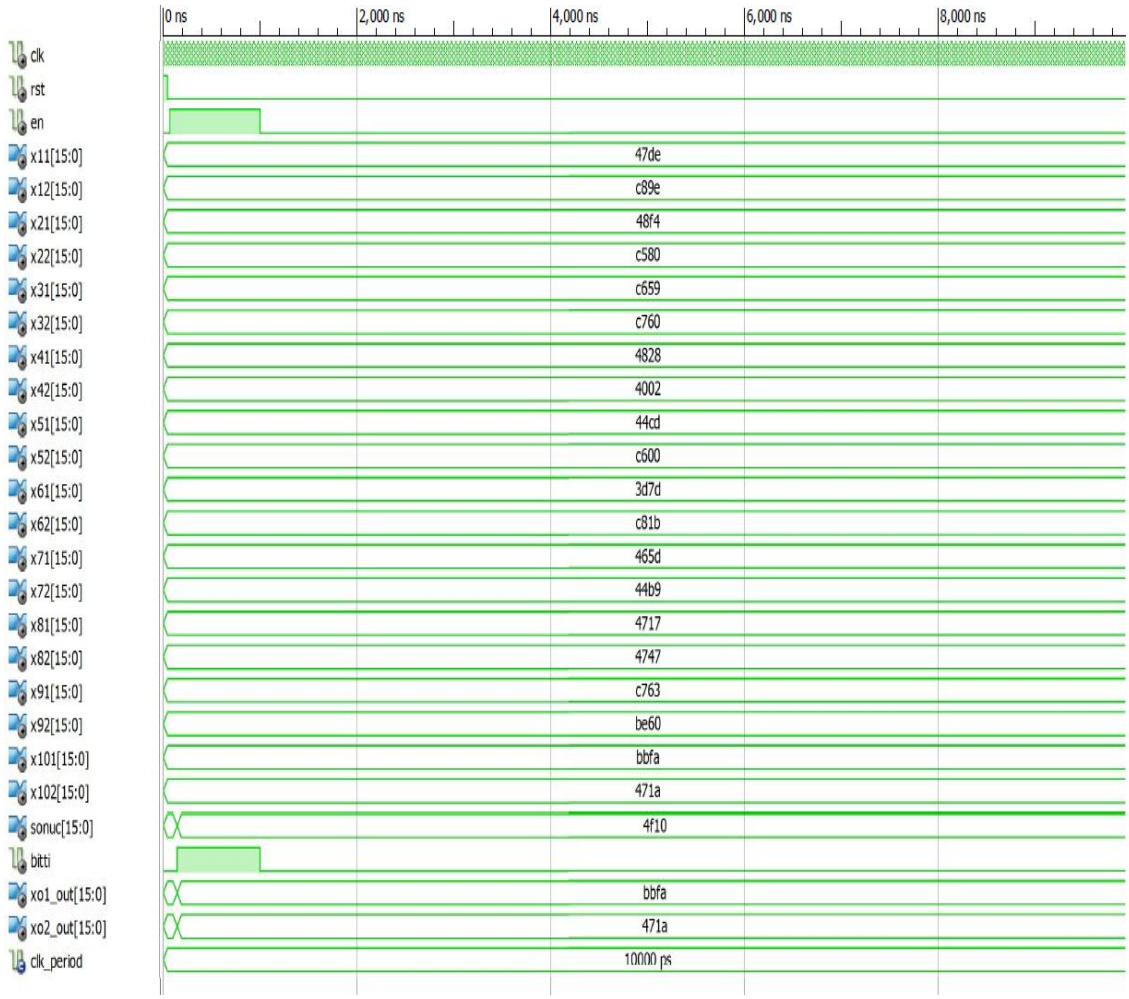
Şekil 4. 2 Başlangıç Çözüm Kümesi Oluşturma Şematik Gösterimi

4.1.2 En İyiyi Bulma

ROM'dan gelen başlangıç çözüm kümeleri, en iyi çözüm kümesini bulmak için bu kısma iletilir. Bu uygulamada 10 adet çözüm kümesi kullanılmıştı, her çözüm kümesinde 2 adet değer bulunuyor. Toplamda 20 değer en iyiyi bulma bileşenine giriş olarak atanmış ve bunlar ikişer ikişer fonksiyona (eşitlik 4.1) uygulanarak en iyi çözüm kümesinin bulunması amaçlanmıştır. Çözüm kümleri fonksiyona uygulandıktan sonra alt bileşen olarak kullanılan karşılaştırma işlemine uygulanır. Daha sonra karşılaştırma işleminin sonucu fonksiyon çıkışları ile eşlenerek hangi çözüm kümesinin en iyi olduğu bulunur.



Şekil 4. 3 En iyiyi bulma işlemi



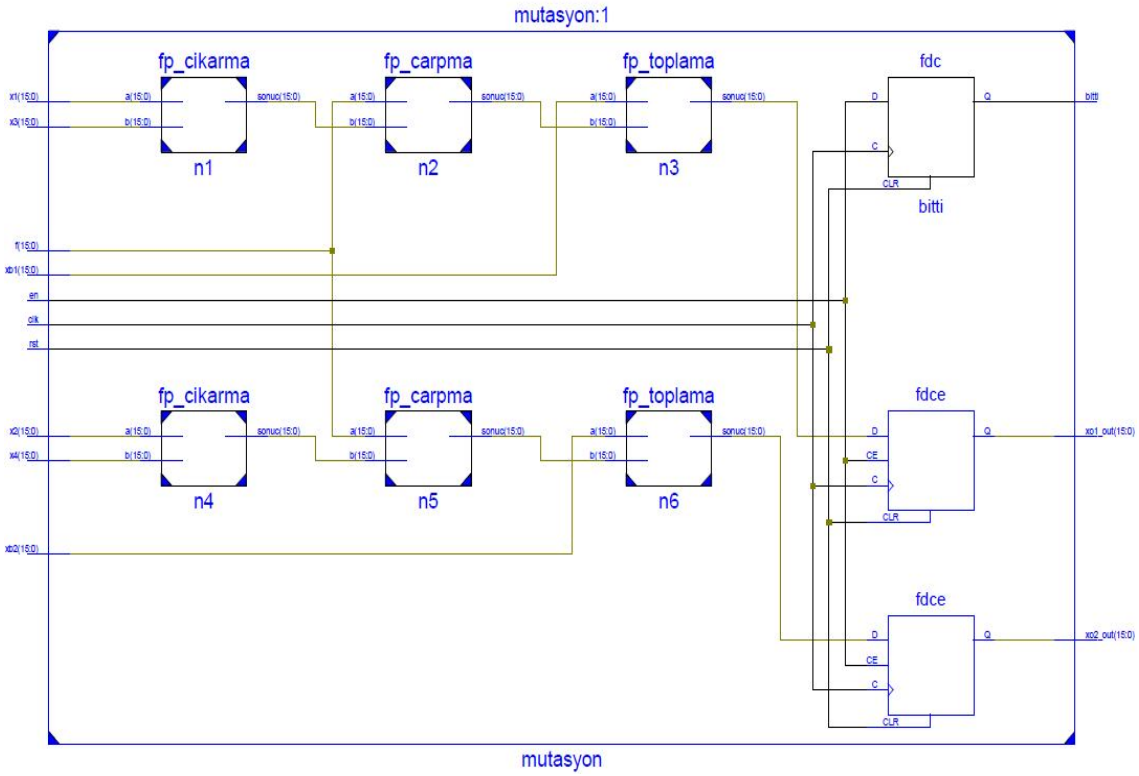
Şekil 4. 4 En iyiyi bulma işleminin test sonuçları

4.1.3 Mutasyon

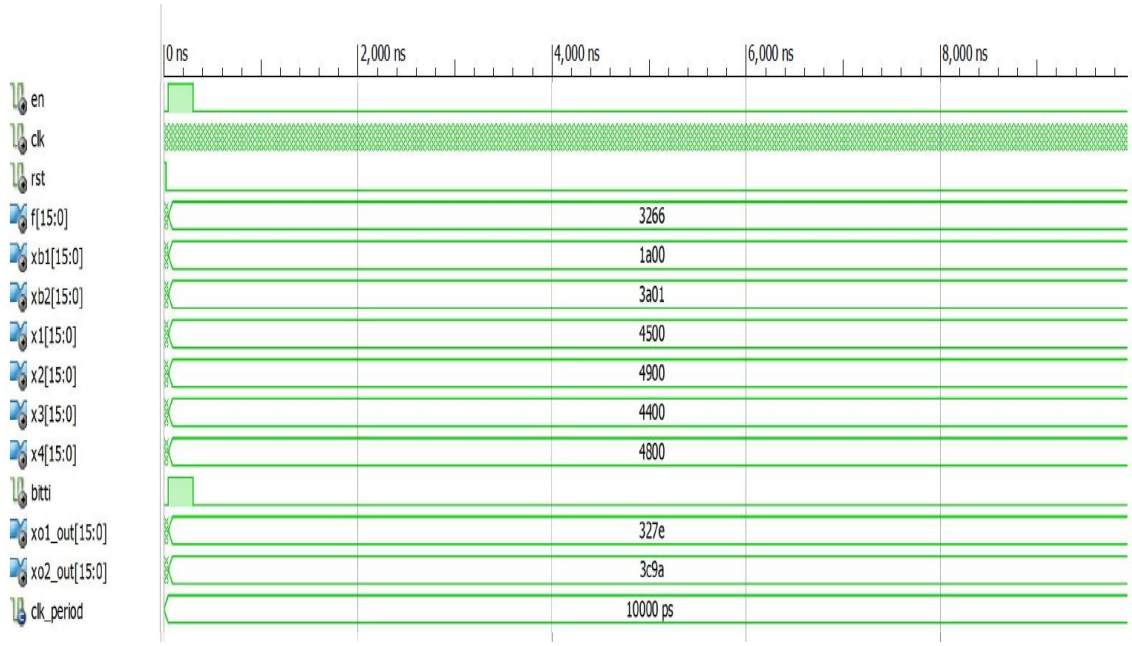
Mutasyon işleminin yapılabilmesi için 3 farklı çözüm kümesine ihtiyaç vardır. Normal DGA'da bu işlem 3 farklı çözüm kümesinin birden rasgele seçilmesi ile yapılır. Bu uygulamada çözüm kümelerinden birine en iyi çözüm kümesi atanmış ve sabit olarak tüm işlemlerde bu çözüm kümesi kullanılmıştır. Diğer 2 çözüm kümesi rasgele olarak seçilmiştir.

$$U_{j,\mathrm{I},G} = x_{best} + F.\left(x_{j,r_1,G} - x_{j,r_2,G}\right) \tag{4.2}$$

Mutasyon işlemi eşitlik 4.2'de gösterilmiştir. Mutasyonda 2 çözüm kümesinin ($x_{j,r_1,G}$, $x_{j,r_2,G}$) rasgele bir şekilde seçilmesi için bir rasgele numara üreteci kullanılmıştır. Çözüm kümeleri öncelikle bu bileşene sokulmuş ve çıkışta rasgele olarak çıkan 2 çözüm kümesi sırasıyla mutasyon işlemine sokulmuştur.



Şekil 4. 5 Mutasyon işleminin şematik gösterimi

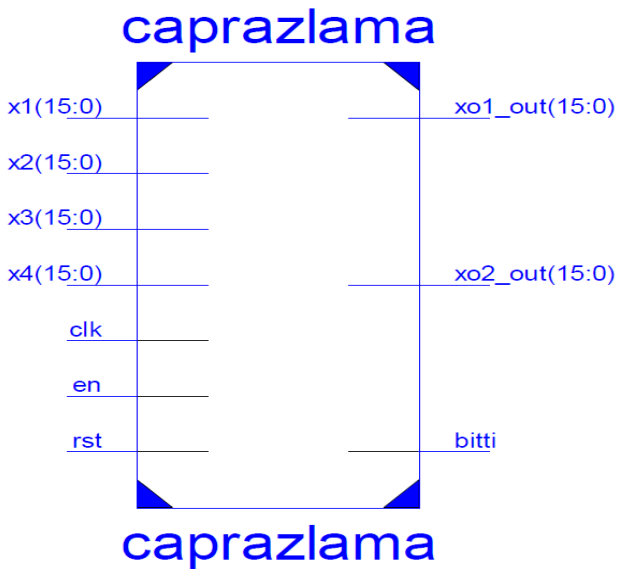


Şekil 4. 6 Mutasyon işleminin test sonuçları

Mutasyon işleminde 2 çözüm kümesinin her iterasyonda değişen rasgele çözüm kümesi olarak seçilmesi gerekmektedir. Aksi taktirde ilk iterasyon dışında herhangi bir güncelleme işleminin yapılması zor bir ihtimaldir.

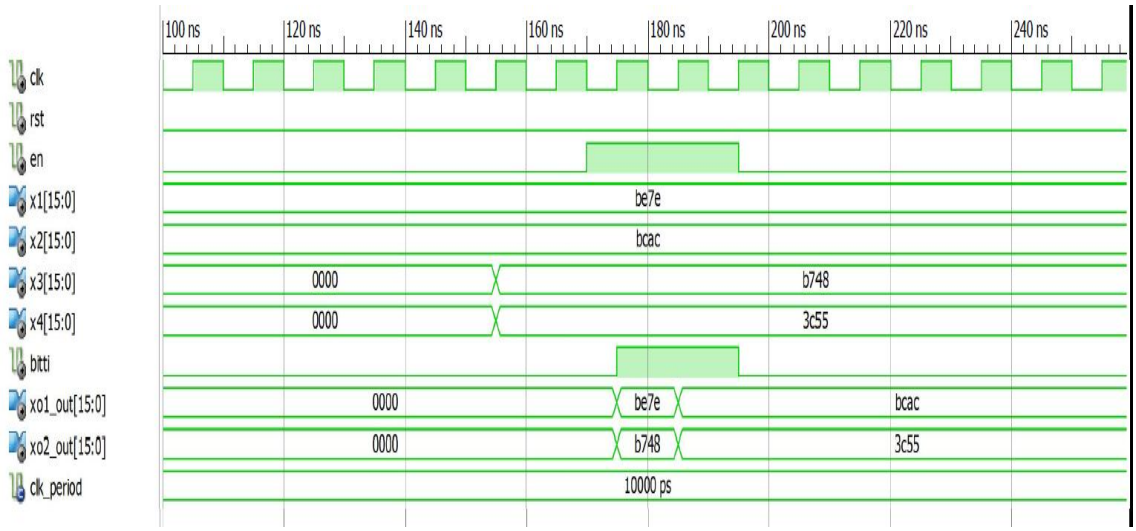
4.1.4 Çaprazlama

Mutasyon sonucunda DGA'da yeni nesil olarak tabir edilen, yeni çözüm kümeleri oluşturulur. Oluşturulan bu çözüm kümeleri bir önceki çözüm kümeleri ile çaprazlama işleminde tabi tutularak daha iyi bir nesil (çözüm kümeleri) oluşturma amaçlanmıştır. Bu işlem VHDL yazılım dili ile gerçeklenirken, 2 adet çözüm kümesi yani 4 adet sayı çaprazlama işleminde giriş olarak atanmıştır. Bu çözüm kümeleri sırasıyla ilk nesilden ve yeni oluşturulan nesilden alınmıştır.



Şekil 4. 7 Çaprazlama işleminin şematik gösterimi

Çaprazlama işlemi normal DGA'da dışarıdan girilen bir oranla yapılmaktadır. Yalnız VHDL dilinde bu işlem hem çok karmaşık hem de çözüme çok fazla etki etmediğinden dolayı, çaprazlama oranı her zaman 0.5 olarak alınmıştır. Bunun anlamı her bir çözüm kümesinin %50'si alınıp birbirine eklenerek yeni bir çözüm kümesi oluşturulmuştur. Bu %50'lik kısmı her iterasyonda değişmektedir. Yani ilk iterasyon birinci yarısı alındıysa ikinci iterasyonda ikinci yarısı alınmaktadır. Böylece çözüm kümelerindeki çeşitlilik arttırılmaya çalışılmıştır.

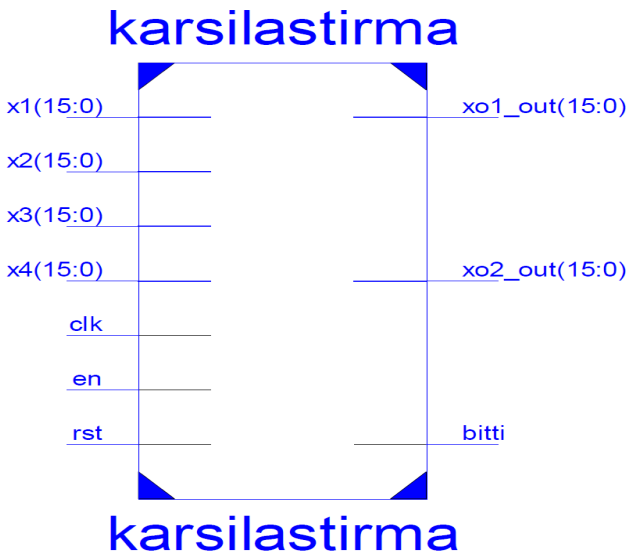


Şekil 4. 8 Çaprazlama işleminin test sonuçları

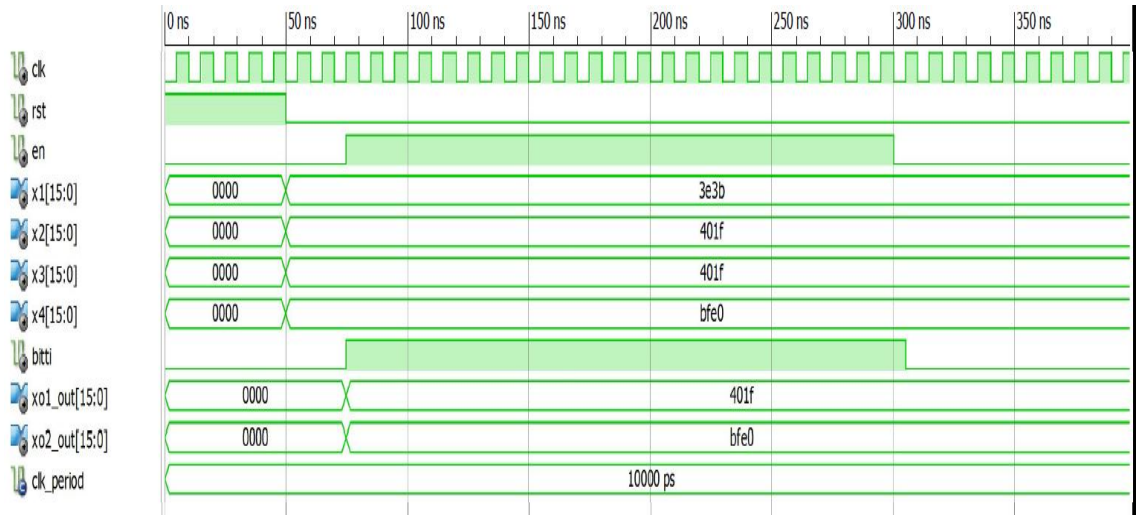
4.1.5 Karşılaştırma

Çaprazlama sonucunda DGA'da yeni nesil oluşturma işlemi sonlanmış ve yeni çözüm kümeleri oluşturulmuştur. Bu kısımda algoritmanın son aşaması olan oluşturulan yeni nesil bireyleri ile başlangıçtaki neslin bireyleri karşılaştırılmış ve iyi olan birey yani daha iyi sonuç veren çözüm kümesi bir sonraki nesile aktarılmıştır.

Bu işlem VHDL'de gerçeklenirken başlangıçtaki çözüm kümeleri ile işlemler sonucunda oluşturulan çözüm kümeleri sırayla karşılaştırma bileşenine giriş olarak atanmış ve bu çözüm kümeleri içeride fonksiyona uygulanarak sonuçlar elde edilmiştir. Bu sonuçlara bakılarak iyi olan çözüm kümesi çıkış olarak verilmiştir.



Şekil 4. 9 Karşılaştırma işleminin şematik gösterimi

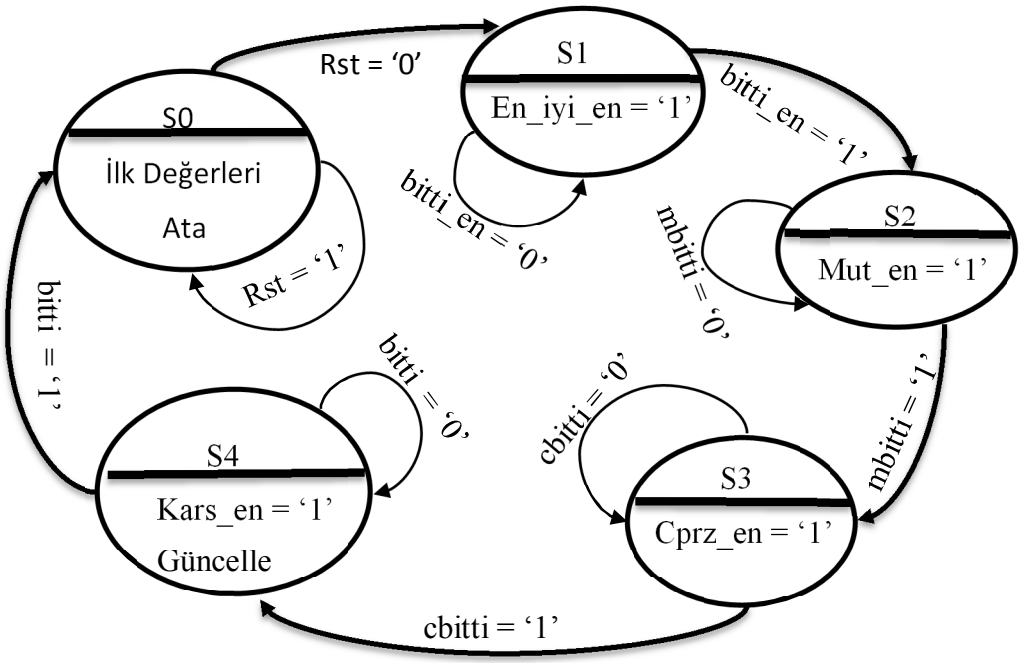


Şekil 4. 10 Karşılaştırma işleminin test sonuçları

4.1.6 Ana Modül

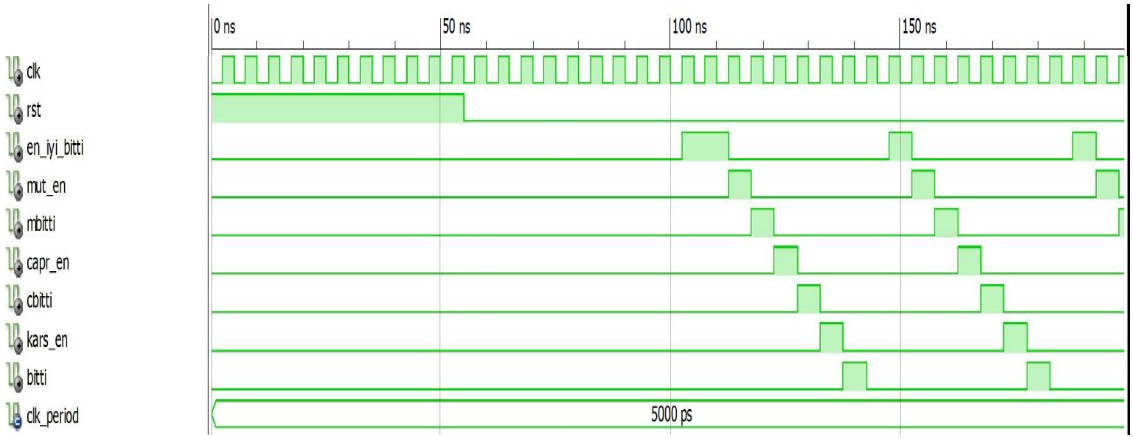
Yukarıda anlatılan işlemler bileşen olarak olarak bir ana modülde toplanmış ve algoritma gerçeklenmiştir. Bilindiği gibi VHDL kodlamada yazılan tüm işlemler eş zamanlı olarak çalışmaktadır. Yalnız bu algoritmanın çalışma prensibine göre tüm bileşenler başlamak için bir önceki bileşenin bitmesini beklemelidir. Çünkü bir önceki bileşenden alınan sonuç ile hesaplamalar yapılmaktadır. Bu sistemin uygulanabilmesi için yukarıda da görüldüğü gibi her bileşene bir başlama (en) ve bitti sinyalleri atanmıştır. Bu sinyaller arasında bir senkronizasyon oluşturularak iterasyon işleminin yapılması amaçlanmıştır. Sinyaller arasında senkronizasyon durum makineleri kullanılarak oluşturulmuştur. Bu durumlarda başlama sinyalleri (en), bitti sinyalleri temel alınarak şekillendirilmiştir.

Şekil 4.11'de görüldüğü gibi bitti sinyallerinin ve başlama sinyallerinin senkronizasyonu durum makineleri ile beş durum oluşturularak yapılmıştır. İlk olarak reset (rst) sinyali programa ilk değerler yüklenmiş ve bu değerler en iyiyi bulma bileşenine gönderilerek bu bileşenin başlama sinyali (en_iyi_en) aktif hale getirilmiştir. Bu bileşenden alınan bitti sinyalinin (bitti_en) durumuna göre mutasyon işleminin başlama sinyali (mut_en) aktif hale getirilmiştir. Mutasyon bileşeninin bitti sinyali ile (mbitti) çaprazlama işlemi başlatılmış (cprz_en) ve çaprazlama işleminin bitmesiyle son bileşen olan karşılaştırma işlemi aktif hale getirilmiştir (kars_en). Döngü tamamlandığı zaman son durum olan S4 durumunda tüm çözüm kümelerinin güncelleme işlemi yapılmaktadır ve son olarak bitti sinyalinin durumuna göre iterasyon tekrardan başlatılmaktadır.



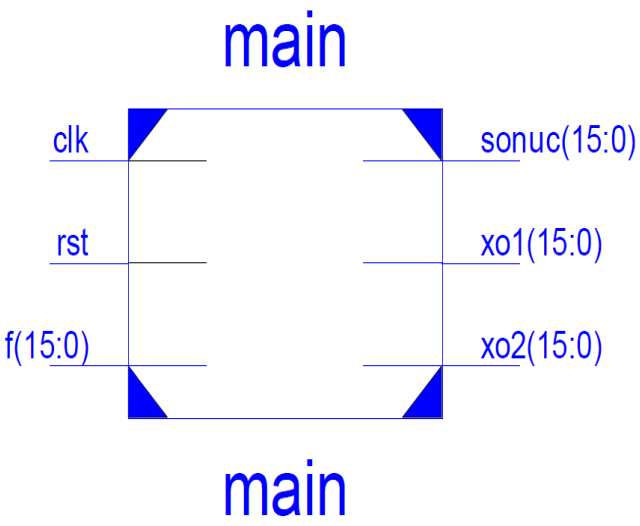
Şekil 4. 11 Başlama ve bitti sinyallerinin senkronizasyonu

Isim programı ile elde edilen simülasyon sonuçları şekil 4.12'de gösterilmiştir.



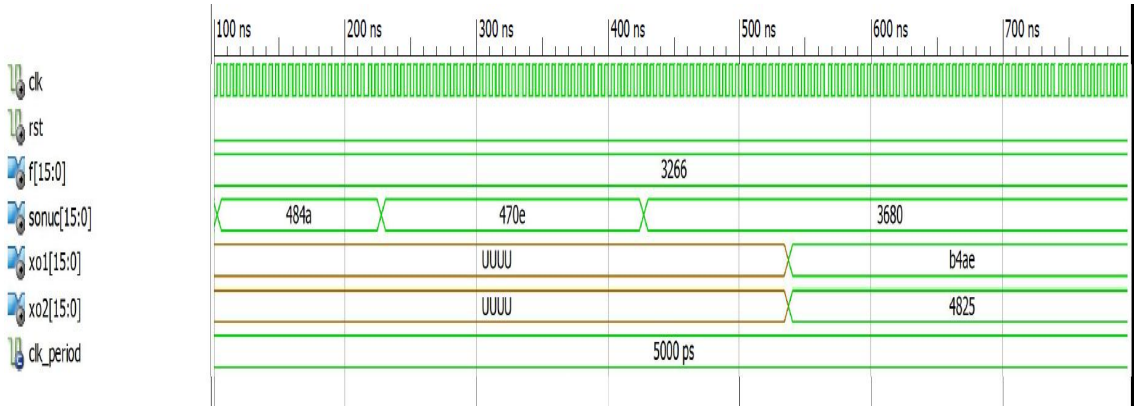
Şekil 4. 12 Başlama ve bitti sinyallerinin simülasyonu

Yukarıda bahsedilen başlama sinyalleri ayarlandıktan sonra, tüm bileşenler koda eklenerek DGA algoritmasının VHDL programlama dili ile kodlanması gerçeklenmiş olur. Bu çalışmada durdurma kriteri iterasyon sayısı olarak ayarlanmıştır. Girilen bir iterasyon sayısı sonunda bulunan çözüm kümesi ve sonuçlar çıktı olarak elde edilir.



Şekil 4. 13 Ana modül şematik gösterimi

Ana modülde frekans değerinin ayarlandığı saat (clk) girişi, başlangıç değerlerinin atandığı ve programın durmasını sağlayan (rst) bir giriş ve DGA'da ölçekleme faktörü olan f girişi bulunmaktadır.



Şekil 4. 14 Ana modülün simülasyonu

BÖLÜM 5

ÇKA AĞININ DGA İLE EĞİTİMİNİN FPGA ÜZERİNDE GERÇEKLENMESİ

Yapılan çalışmalarda, YSA'nın eğitim işlemi gradyan temelli algoritmalar ile FPGA üzerinde gerçeklenmiştir [10-12]. Bu çalışmalar ile YSA modellerinin eğitimlerinin donanım üzerinde gerçeklenebildiği ve problem çözümlerinde kullanılabileceğini göstermiştir. Gradyan temelli algoritmaların en büyük dezavantajı olan bölgesel minimumlara takılma riski olduğundan YSA'ların eğitiminde son zamanlarda sezgisel algoritmalar kullanılmış ve sonuçların verimli olduğu gözlenmiştir [15-19],[24].

DGA'nın FPGA üzerinde gerçeklenmesi bu çalışmanın üçüncü bölümünü oluşturulmaktadır. Sonuçlara bakıldığında bu algoritmanın donanım üzerindeki çalışmasının verimli olduğu görülmüştür.

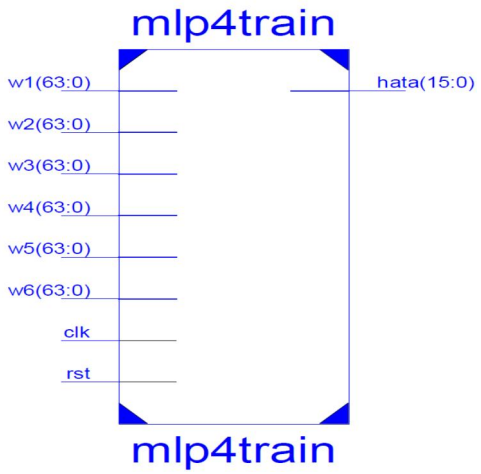
Literatürde sezgisel algoritmaların FPGA üzerinde gerçeklenmesi çalışmaları bulunmaktadır [25-28]. FPGA üzerinde bu algoritmaların YSA eğitiminde kullanılmasına yönelik bir çalışma yapılmıştır [39].

Çalışmanın bu bölümünde DGA algoritması ile ÇKA ağının eğitimi FPGA üzerinde gerçeklenmiş ve sonuçlar gözlenmiştir. Bu yapının kodlaması, Xilinx ISE Design Suite 14.3 programında VHDL programlama dili ile Virtex-7 VC707 geliştirme kitine uygun olarak yapılmıştır. Kodlama işleminde ikinci bölümde olduğu gibi, her bir bileşen ayrı ayrı oluşturulmuş, simülasyon sonuçları çalışması kontrol edilmiş ve bir ana modül ile birleştirilerek amaçlanan yapı oluşturulmuştur.

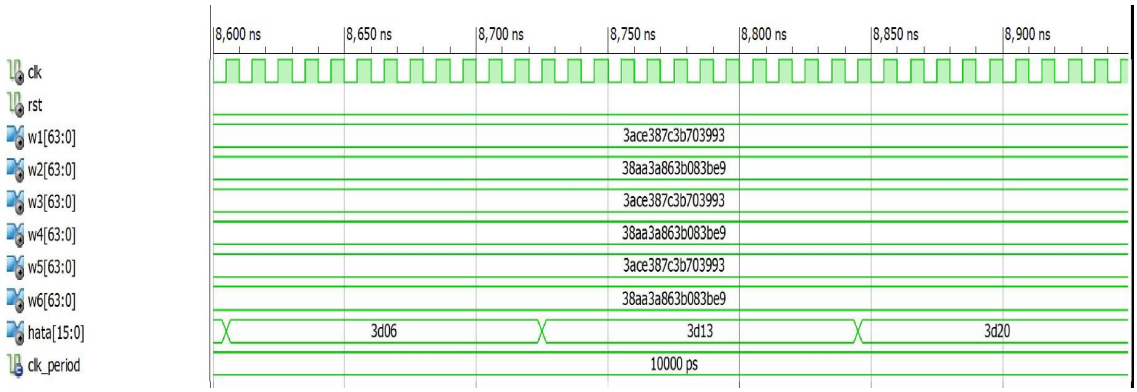
5.1 Eğitim İçin Tasarlanan ÇKA Yapısı

Bu çalışmada eğitim işlemi gerçekleştirilirken iris çiçeğine ait veri kümesi kullanılmıştır. Iris verisi için tasarlanan ÇKA, giriş sayısı 4, gizli nöron sayısı 4, çıkış nöron sayısı 2 olacak şekilde tasarlanmıştır. Tasarımdan dolayı bu ağda optimizasyon yapılması gereken 24 adet ağırlık değeri bulunmaktadır. Tasarımın VHDL diliyle kodlaması yapılırken bu ağırlık değerleri giriş olarak atanmıştır. Yapılan işlemlerde bu değerler 16 bitlik floating-point yapısı ile temsil edilmiştir. Girişler atanırken her bir nörona gelen toplam ağırlıklar birleştirilerek bir vektör olarak ifade edilmiş ve program içerisinde yapı bölünerek ağırlık değeri olarak atanmıştır. Yani programın 64 bitlik 6 adet girişi ve bir adet hata çıkışı bulunmaktadır.

Veri kümesi program içine ROM olarak gömülmüş ve her iterasyonda veriler sırayla ROM'dan çekilerek iterasyon tamamlanmıştır.



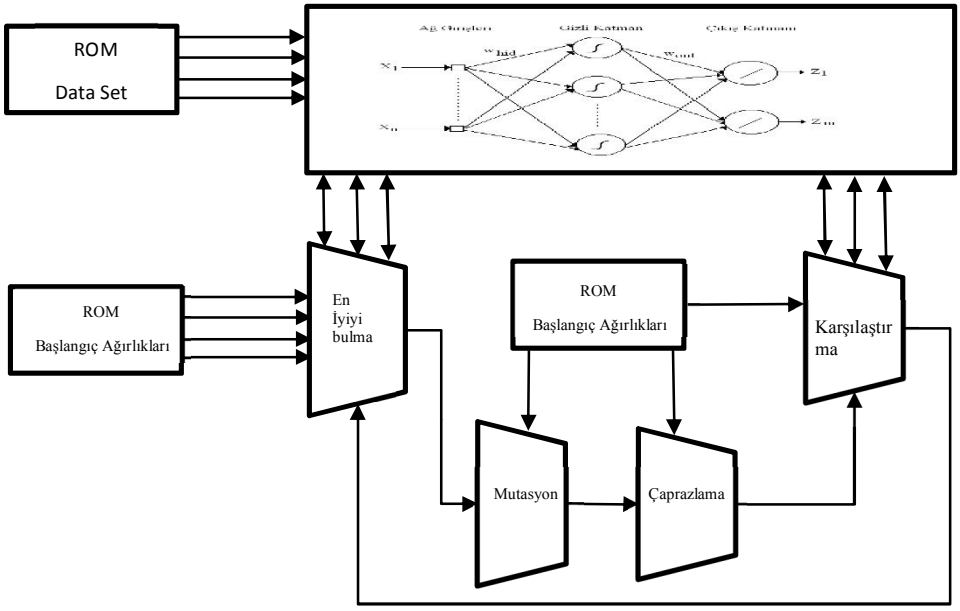
Şekil 5. 1 Eğitim için tasranan ÇKA yapısının şematik gösterimi



Şekil 5. 2 Eğitim için tasranan ÇKA yapısının simülasyon sonucu

5.2 **ÇKA Eğitimi İçin Tasarlanan DGA Yapısı**

DGA yapısı, ÇKA ağının eğitimi için VHDL diliyle gerçeklenirken, optimize edilmesi gereken bileşen sayısı fazla olduğundan, başlangıç çözüm kümesi oluşturulurken ve diğer bileşenler tasarlanırken, bileşenler arasındaki geçişin rahat bir şekilde kodlanabilmesi amacıyla giriş ve çıkışlar uzun boyutlu vektörler cinsinden ifade edilmişlerdir. Bu vektörler programlar içinde ayrıştırılarak işlemlerde kullanılmıştır.

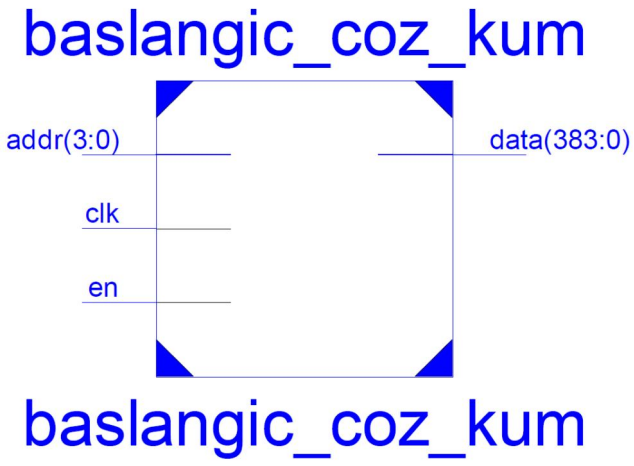


Şekil 5. 3 Akış Diyagramı

Şekil 5.3'te yazılan programın akış diyagramı görülmektedir. Program bu akış diyagramına uygun olarak kodlanmış ve sentezlenmiştir.

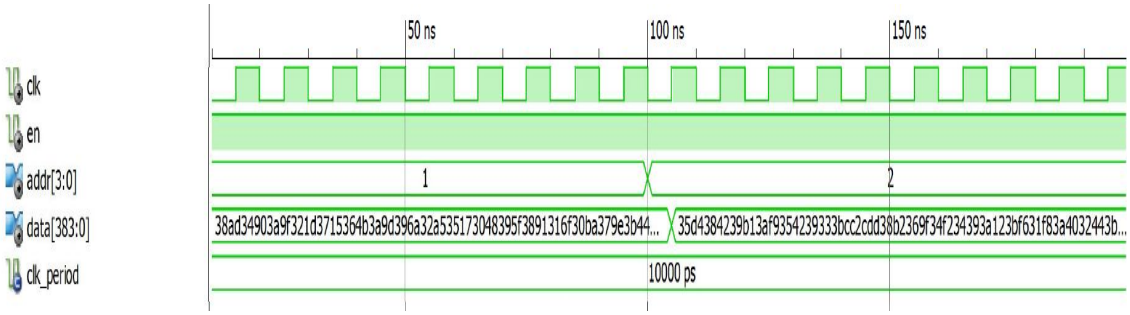
5.2.1 Başlangıç Çözüm Kümesinin Oluşturulması

Tasarımda 24 adet optimizasyon yapılacak değer bulunmaktadır. Bunun anlamı her çözüm kümesinde 16 bitlik 24 sayı bulunuyor. Bu yapı için başlangıç çözüm kümesi sayısı yani popülasyon büyüklüğü (NP) 10 olarak belirlenmiştir. Çözüm kümeleri vektör cinsinden ifade edilmiştir yani her çözüm kümesi 1X384 boyutunda vektörlerden oluşmaktadır.



Şekil 5. 4 Başlangıç çözüm kümesi bileşeninin şematik gösterimi

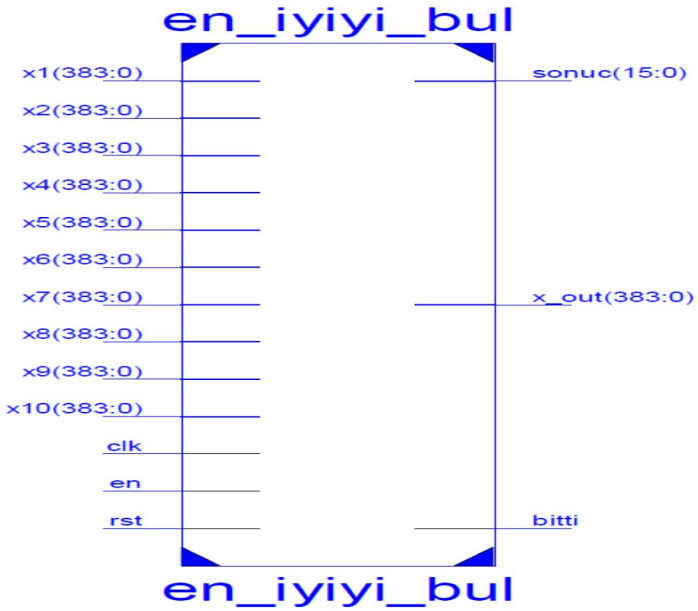
Bu bileşende, seçim sinyali (addr), frekansın ayarlanacağı saat sinyali (clk) ve başlatma sinyali (en) giriş olarak atanmış ve bir adet çıkış sinyali (data) atanmıştır.



Şekil 5. 5 Başlangıç çözüm kümesi bileşeninin simülasyon sonucu

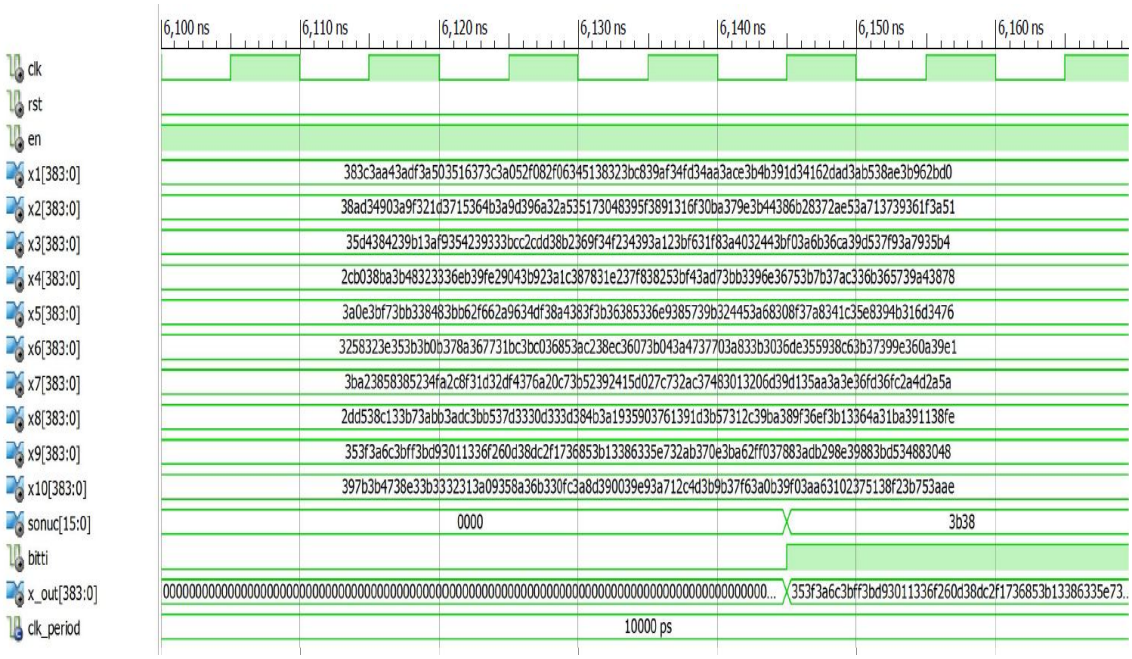
5.2.2 En İyiyi Bulma

Başlangıçta üretilen çözüm kümleri eğitim için tasarlanan ÇKA ağına uygulanarak çıkan sonuçlar bir eleme işlemine tabi tutulur ve çıkışa en iyi sonucu veren çözüm kümesi atanır. Bu bileşende de, iki bilinmeyenli denklemden farklı olarak optimize edilen problemin çok fazla parametresi olduğundan girişler uzun boyutlu vektörler olarak atanmıştır. Program içerisinde bu girişler ayrıştırılarak YSA'ya uygulanmaktadır.



Şekil 5. 6 En iyiyi bulma bileşeninin şematik gösterimi

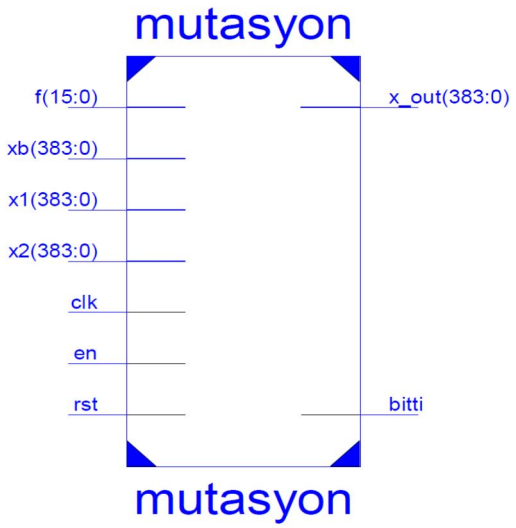
En iyiyi bulma bileşeni, program yazıldıktan sonra simülasyon programı ile doğruluğu gözlenmiştir (Şekil 5.7). En iyiyi bulma işleminde çözüm kümeleri her seferinde YSA'ya uygulanır ve çıkan sonuçlar eleme işlemine tabi tutularak en iyi çözüm kümesi belirlenir.



Şekil 5. 7 Başlangıç çözüm kümesi bileşeninin simülasyon sonucu

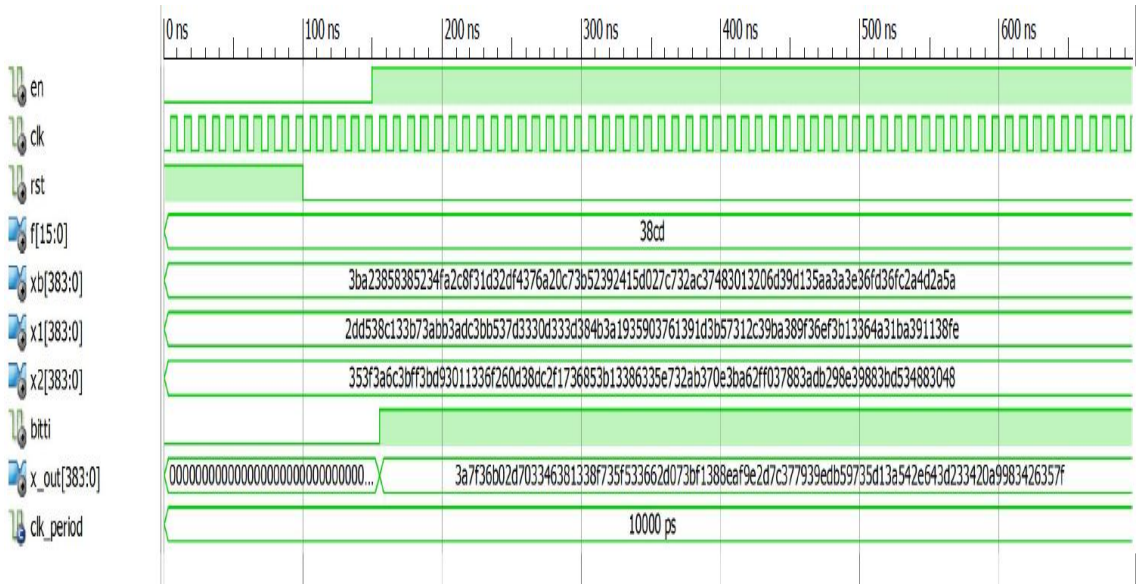
5.2.3 Mutasyon

Mutasyon işleminde girişler 1X384 uzunluğunda vektörler olarak tanımlanmış ve program içerisinde her bir çözüm kümesi parametrelerine ayrılmıştır. Yani her bir çözüm kümesi 16 bitlik 24 adet sayı olarak işleme sokulmuştur. Tüm çözüm kümeleri diğerlerinden bağımsız olarak kendi aralarında mutasyon işlemine tabi tutulmuştur.



Şekil 5. 8 Mutasyon bileşeninin şematik gösterimi

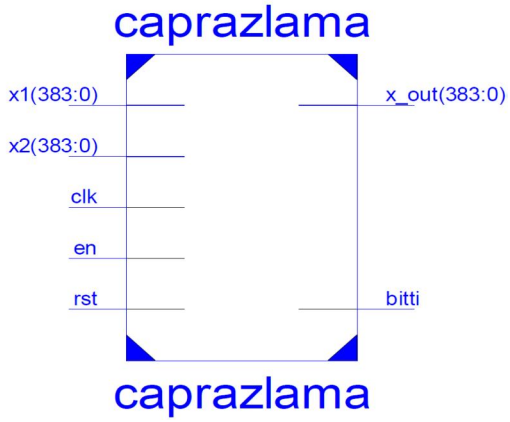
Mutasyon işleminin simülasyon sonuçları alınmış ve doğruluğu gözlenmiştir (Şekil 5.9).



Şekil 5. 9 Mutasyon bileşeninin simülasyon sonucu

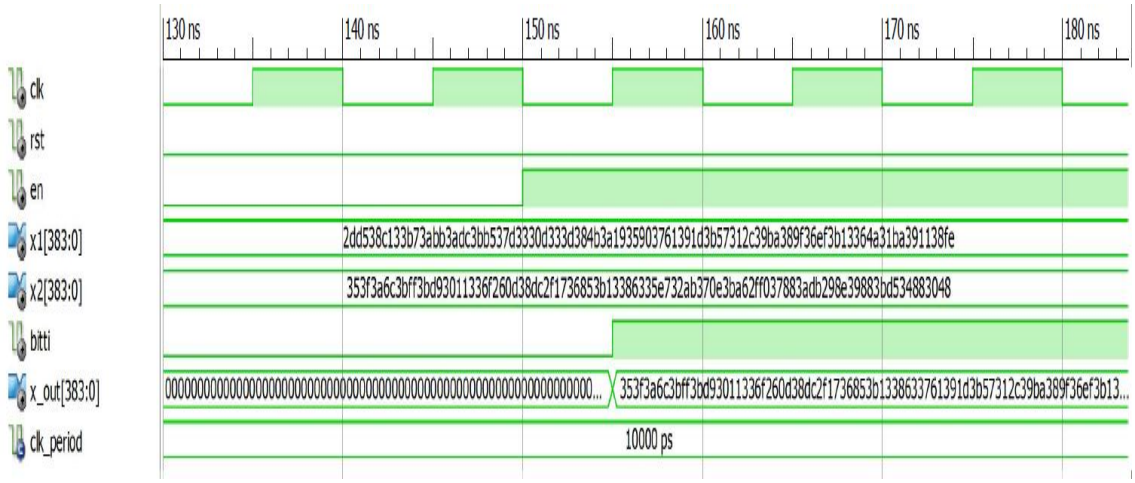
5.2.4 Çaprazlama

DGA'nın temel adımlarından biri olan çaprazlama, girilen çözüm kümelerinin verilen oran (CR) ile çaprazlanması ile yeni bir çözüm kümesi oluşturmayı amaçlar. Girilen çözüm kümelerinden hangi oranda eleman alınacağı belirlendikten sonra bu elemanlar rastgele bir şekilde alınmaktadır. Bu işlem donanımda gerçeklenirken CR sabit değer olarak 0.5 kabul edilmiştir. Girilen çözüm kümelerinin yarısı alınıp bir birlerine eklenerek çözüm kümesi oluşturulmuştur. MATLAB ortamında işlem gerçeklenirken parametrelerin rastgele alımı ve sabit bir şekilde ilk yarısının veya son yarısının alımı yapılmış ve arada % 2.27'lik bir fark oluştuğu görülmüştür. Bu oran bulunurken aynı işlem 5 defa tekrarlanmış ve sonuçların ortalaması alınmıştır. Bu oran sabit olmamakla beraber her uygulama için değişim gösterecektir. Bu çalışmada oranın sonucu çok fazla etkilemediği görülmüş ve donanım gerçeklemesi yapılırken kodlamada rahatlık olması ve alanın daha etkili kullanılması amacı ile çaprazlamada girilen çözüm kümelerinin yarısı alınarak yeni çözüm kümesi oluşturulmuştur.



Şekil 5. 10 Çaprazlama bileşeninin şematik gösterimi

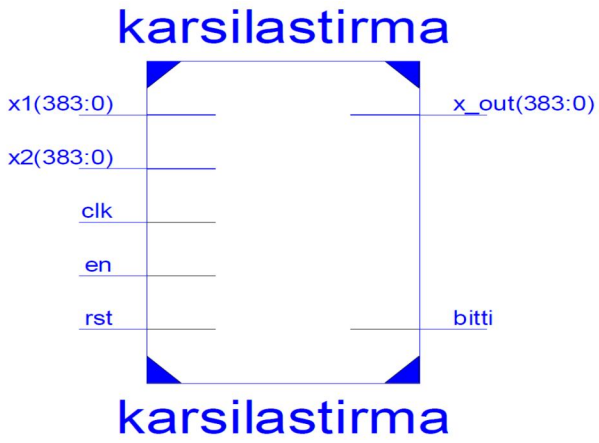
Bu bileşeninde, diğer bileşenler gibi başlama (en) ve bitiş sinyaller eklenerek simülasyonu yapılmış ve doğruluğu gözlenmiştir (Şekil 5.11).



Şekil 5. 11 Çaprazlama bileşeninin simülasyon sonucu

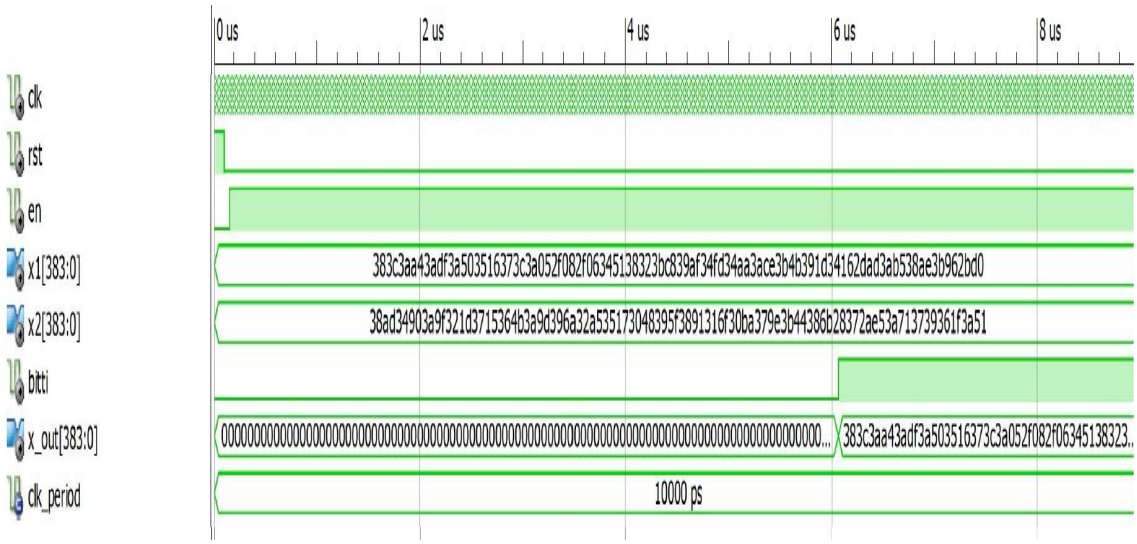
5.2.5 Karşılaştırma

Karşılaştırma bileşeninde girilen çözüm kümeleri eğitim için tasarlanan YSA yapısına giriş olarak atanır. YSA çıkışında alınan sonuçlar kıyaslanarak iyi olan çözüm kümesi çıkışa verilir ve yeni nesil olarak adlandırılan yeni çözüm kümesine eklenir.



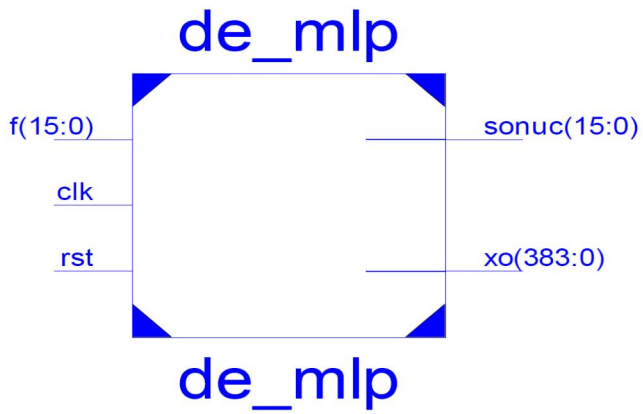
Şekil 5. 12 Karşılaştırma bileşeninin şematik gösterimi

İterasyonun son aşaması olan karşılaştırma işlemi sonucunda çıkan veriler yeni nesile aktarılır ve yeni bir çözüm kümesi oluşturulur. Karşılaştırma bileşenin simülasyonu yapılmış ve doğruluğu gözlenmiştir (Şekil 5.13).



Şekil 5. 13 Karşılaştırma bileşeninin simülasyon sonucu

Karşılaştırma işlemi programın son bileşenidir. Tüm bileşenler oluşturulduktan sonra, bunların belirlenen bir senkronizasyon içinde çalışmalarını sağlmak için her bileşene bir başlangıç sinyali (en) ve birde bitti sinyali atanmıştır. Başlangıç ve bitti sinyallerinin senkronizasyonu Şekil 4.11 ve 4.12'de gösterildiği gibi yapılmıştır.



Şekil 5. 14 Ana modülün şematik gösterimi

5.2.6 Analiz Sonuçları

VHDL programa dili ile gerçeklenen, DGA ile eğitilmiş ÇKA ağının analiz sonuçlarında eğitim başarısının yaklaşık % 70 olduğu görülmüştür. Bu başarı MATLAB ortamında tasarlanan ağ yapısından düşük olmasına rağmen, donanım üzerinde eğitim ve test işlemlerinin çok hızlı yapılması bir avantajdır.

Çizelge 5. 1 ÇKA'nın Eğitiminde DGA'nın Kontrol Parametreleri

NP	F	CR	İterasyon Sayısı	Hata	Test Başarısı (%)
10	0.6	0.5	150	0.637	70.3571
10	0.8	0.5	150	0.725	67
10	1.2	0.5	150	0.830	65.5714

Çizelge 6.1'de elde edilen değerler eğitim ve test işlemi 5 defa tekrarlandıktan sonra ortalaması alınarak elde edilmiştir.

Eğitim işlemi MATLAB ortamında yapıldığında, hesaplamalar daha hassas yapıldığından iterasyon sayısı artırılarak hata düşürülebilmektedir. VHDL programlama dilinde yaklaşık ilk 100 iterasyondan sonra sonuç değişmemektedir.

Şekil 5. 15 Ana modülün simülasyon sonucu

İstenilen yapı tasarlandıktan sonra, programın girişinde F parametresinin değeri atanmış ve Şekil 5.15'teki simülasyon sonucu elde edilmiştir.

BÖLÜM 6

SONUÇ VE ÖNERİLER

Bu çalışmada, sezgisel algoritmalardan DGA'nın FPGA üzerinde çalışma performansı ve verimliliği gözlenmiştir.

Çalışmanın ilk kısmında sezgisel algoritmalardan DGA'nın, GYA gibi gradyan tabanlı algoritmalara göre daha verimli olduğunun görülebilmesi için bu iki algoritma MATLAB üzerinde gerçeklenerek performansları karşılaştırılmıştır. Bu aşamada GYA önce basit bir denklem optimizasyonu için kodlanmış ve performansı gözlenmiştir. İstenilen performansı sağladığı görüldükten sonra, program biraz daha genişletilerek yoğun işlem yüküne sahip YSA'ların eğitimi için kullanılacak şekilde tasarlanmıştır. YSA modellerinden olan KKFSA ve ÇKA ağlarının eğitimi DGA ile gerçeklenmiş ve sonuçlar analiz edilerek GYA ile eğitilen ve aynı mimarideki YSA'lar ile karşılaştırılmıştır. Bu analiz sonucunda gradyan tabanlı algoritmaların daha hızlı olduğu fakat yerel minimumlara takılma risklerinin daha fazla olduğu görülmüştür. Uygulamalar sonucunda GYA'nın performansının giriş parametrelerine çok bağımlı olduğu, DGA'nın ise GYA'ya göre giriş parametrelerine olan bağımlılığın daha az olduğu görülmüştür.

Çalışmanın sonraki bölümlerinde, MATLAB ortamında uygulaması yapılarak performansı gözlenen ve verimliliği ispatlanan DGA, VHDL programlama dili ile gerçeklenerek donanıma uygun hale getirilmesi amaçlanmıştır. Bu aşamada DGA, iki bilinmeyenli bir denklemin optimizasyonu için VHDL dili ile kodlanmış ve simülasyon sonuçları teorideki sonuçları ile karşılaştırılarak programın doğruluğu ve verimliliği ispatlanmıştır. Analiz sonucunda istenilen performans alındıktan sonra, program genişletilerek bir YSA'nın eğitimine uygun hale getirilmiştir. Yazılan programın test edilmesi için VHDL yazılım dili

ile eğitimi amacıyla bir ÇKA yapısı tasarlanmıştır. Tasarlanan bu ağın eğitimi GYA ile yapılarak performansı test edilmiştir.

Yapılan analiz sonucunda GYA'nın donanım ortamında bir YSA yapısının eğitimi için uygun olduğu görülmüştür. FPGA üzerinde performansı gözlenen ve verimliliği ispatlanan algoritma ileriki çalışmalar için YSA yapılarının eğitimine uygun hale getirilmiştir.

Çalışmanın sonraki aşamalarında DGA ve seçilen başka sezgisel algoritmalar ile KKFSA eğitiminin yapılması ve medikal veri setleriyle performansının gözlenmesi amaçlanmıştır.

Bu çalışma incelendiğinde iki farklı algoritma aynı anda kullanılarak, GYA'nın hızı ve DGA'nın güvenilirliği sağlanabileceği düşünülebilir. DGA ile başlatılan bir optimizasyon ile GYA algoritmasının başlangıç değerleri belirlenir, böylece belirlenen bir alanda çok daha etkin bir arama yapılabilir.

KAYNAKLAR

[1] Cheung O. ve Leong P.,(2002) “Implementation of an FPGA based accelerator for virtual private networks”, IEEE International Conference on Field Programmable Technology (FPT), 2002 , Hong Kong.

[2] Gause J., Cheung P. ve Luk W., (2000). “Static and dynamic reconfigurable designs for a 2D shape-adaptive DCT”, Field-Programmable Logic and Applications, 27-30 August, 2000 Belin .

[3] Villasenor J., Jones C. ve Schoner B., (1995). “Video communications using rapidly reconfigurable hardware”, IEEE Transactions on Circuits and Systems for Video Technology, 5:565–567.

[4] Hämäläinen A., Tommiska M. ve Skyttä J., (2002). “6.78 Gigabits per Second Implementation of the IDEA Cryptographic Algorithm”, Proceedings of the 12th Conference on Field-Programmable Logic and Applications, 2-4 September 2002 France.

[5] Lysaght P., Stockwood J., Law J. ve Girma D.,(1994), “Artificial neural network implementation on a fine-grained fpga”, Field-Programmable Logic, 849:421–431.

[6] Bade S. ve Hutchings B., (1994). “Fpga based stochastic neural network implementation”, Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, 189–198.

[7] Gadea R., Cerd J., Ballester F. ve Mochol A., (200). “Artificial neural network implementation on a single fpga of a pipelined on-line backpropagation”, Proceedings of the 13th Conference on International Symposium on System Synthesis, 20-22 Sept 2000, Madrid.

[8] Erkmen, B., (2007). Genel Amaçlı Bir Yapay Sinir Ağının Karma Bir Donanımla gerçeklenmesi, Doktora Tezi, YTÜ Fen Bilimleri Enstitüsü, İstanbul.

[9] Merchant S.G. ve Peterson G.D.,(2008). “ An evolvable artificial neural network platform for dynamic environments”, 51st Midwest Symposium on Circuits and Systems, Aug. 2008, Tennessee.

[10] Shawash J. ve Selviah D., (2013). “Real-time non-linear parameter estimation using the Levenberg-Marquardt algorithm on Field Programmable Gate Arrays”, IEEE Trans. on Industrial Electronics, 60: 170-176.

[11] Gomperts A., Ukil A. ve Zurfluh F., (2011). “Development and Implementation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications” IEEE Trans. on Industrial Informatics, 7: 78-89.

[12] Brassai S.T. , Bako L., Pana G. ve Dan S.,(2008). “Neural control based on RBF network implemented on FPGA,” 11th International Conference on Optimization of Electrical and Electronic Equipment, 22-23 May 2008, Optim.

[13] Elitas M., Yavuz O. ve Erkmen B., (2012) “Field Programmable Gate Array implementation of Conic Section Function Neural Network: An alternative to analog CSFNN circuitry”, INES 2012 IEEE 16th International Conference on Intelligent Engineering Systems, 13-15 June 2012, Lizbon.

[14] Yıldırım T., (1997), Development of Conic Section Function Neural Networks in Software and Analogue Hardware, Doktora Tezi, University of Liverpool.

[15] Shi H., ve Wanqing L.,(2009). “Artificial neural networks with ant colony optimization for assessing performance of residential buildings”, Int. Conf. on Future Biomedical Inf., 13-14 Dec. 2009, Handan, 379-382.

[16] Carvalho, M. ve Ludermir, T.B. (2007). “ Particle Swarm Optimization of Neural Network Architectures and Weights”, 7th Int. Conf. on Hybrid Intelligent Systems, 17-19 Sept. 2007, Kaiserlautern, 336 – 339.

[17] Ilonen J., Kamarainen J. K. ve Lampinen J., (2003). “Differential evolution training algorithm for feed-forward neural networks”, Neural Processing Letters, 17: 93-105.

[18] Masters, T. ve Land, W., (1997). "A new training algorithm for the general regression neural network", Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference, 12-15 Oct 1997, Orlando.

[19] Yilmaz, A.R. Yavuz, O. ve Erkmen, B., (2013). "Training multilayer perceptron using differential evolution algorithm for signature recognition application," Signal Processing and Communications Applications Conference (SIU), 24-26 April 2013, Haspolat.

[20] Slowik, A., ve Bialko, M.,(2008) “Training of artificial neural networks using differential evolution algorithm” conf. on Human System Interactions, 25-27 May 2008, Krakow

[21] Erkmen B., Kahraman N., Vural R. A. ve Yildirim T., (2008). “CSFNN Optimization of Signature Recognition Problem for a Special VLSI NN Chip” The 3rd International Symposium on Communications, Control and Signal Processing (ISCCSP 2008), 12-14 March 2008,St. Julians.

[22] Şenol C. ve Yıldırım T., (2005). “Signature Verification Using Conic Section Function Neural Network”, Lecture Notes in Computer Science, 3733: 524 – 532.

[23] Erkmen B., Kahraman N, Acar-Vural R. ve Yildirim T.,(2010). “Conic Section Function Neural Network Circuitry for Offline Signature Recognition,” IEEE Trans. Neural Networks, 21(4): 667- 672

[24] Yilmaz, A.R., Erkmen, B. ve Yavuz, O., (2013). "The performance of differential evolution algorithm for training CSFNN using a pattern recognition application," Intelligent Control and Information Processing (ICICIP), 2013 Fourth International Conference, 9-11 June 2013, Beijing.

[25] Scheuermann B. , Guntsch K. So, M. , Middendorf M. , Diessel O. , ElGindy H. ve Schmecka H., (2004). “FPGA implementation of population-based ant colony optimization”, Applied Soft Computing, 4:303 – 322.

[26] Graham P. ve Nelson B.,(1996). “Genetic algorithms in software and in hardware—a performance analysis of workstation and custom computing machine implementations”, IEEE Symposium on FPGAs for Custom Computing Machines, 17-19 Apr 1996, Napa Valley, 216–225.

[27] Bland I.M. ve Megson G.M.,(1998). “The systolic array genetic algorithm, an example of systolic arrays as a reconfigurable design methodology”, IEEE Symposium on FPGAs for Custom Computing Machines, 15-17 Apr 1998, Napa Valley.

[28] Aporntewan C. ve Chongstitvatana P.,(2001). “Hardware implementation of the compact genetic algorithm”, Proceedings of the 2001 IEEE Congress on Evolutionary Computation, 27-30 May 2001, Seoul.

[29] Haykin S., (1994), Neural Networks: A Comprehensive Foundation, MacMillan College Publishing Company, New York.

[30] Dorffner, G., (1994), “Unified frameworks for MLP and RBFNs: Introducing Conic Section Function Networks”, Cybernetics and Systems, 25: 511-554.

[31] Downs J.W. (2003), Practical Conic Sections: The Geometric Properties of Ellipses, Parabolas and Hyperbolas, Dover Publications Inc., Mineola, New York.

[32] Storn, R., (1997).“Differential Evolution-A Simple and Efficient Heuristic Strategy for Global Optimization Over Continuous Spaces,” *Journal of Global Optimization*, 11: 341-359.

[33] Keskintürk T., (2006) “Diferansiyel Gelişim Algoritmasi”, İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi,9: 85-99.

[34] Schmidt, H. ve Thierauf, G., (2005), “A Combined Heuristic Optimization Technique”, Advances in Engineering Software, 36:11-19.

[35] Sarıtaş E. ve Karataş S., (2012), Her Yönüyle FPGA ve VHDL, Palme Yayınevi, Ankara.

[36] Xilinx All Programmable, 7 Series FPGA Overview, http://www.xilinx.com/support/, 16 Kasım 2013.

[37] Fuentes M., Garcia-Salicetti S. ve Dorizzi B., (2002). “On-line signature verification: Fusion of a hidden Markov model and a neural network via a

support vector machine", 8th Int. Workshop Front. Handwriting Recognit. (IWFHR-8), Aug.2002,Canada.

[38] Huh S., and Lee D., (2010). "Linear Discriminant Analysis for Signatures" *IEEE Transactions on Neural Networks*, 21(12): 1990 – 1996.

[39] Vasumathi, B. and S. Moorthi, (2012). "Implementation of hybrid ANN–PSO algorithm on FPGA for harmonic estimation" ,Eng. Appli. Artif. Intell., 25: 476-483.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı :Ali Rıza YILMAZ

Doğum Tarihi ve Yeri :27.03.1986

Yabancı Dili :İngilizce

E-posta :aliriza@yildiz.edu.tr

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Elektrik-Elektronik Mühendisliği	Fırat Üniversitesi	2010
Lise		Balakgazi Lisesi	2004

YAYINLARI

Bildiri

1.Yilmaz, A.R. Yavuz, O. ve Erkmen, B., (2013). "Training multilayer perceptron using differential evolution algorithm for signature recognition application," Signal Processing and Communications Applications Conference (SIU), 24-26 April 2013, Haspolat.

2. Yilmaz, A.R., Erkmen, B. ve Yavuz, O., (2013). "The performance of differential evolution algorithm for training CSFNN using a pattern recognition application," Intelligent Control and Information Processing (ICICIP), 2013 Fourth International Conference, 9-11 June 2013, Beijing.