KTO KARATAY
ÜNİVERSİTESİ

T.C.

KTO Karatay Üniversitesi

Fen Bilimleri Enstitüsü

ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

TEZLİ YÜKSEK LİSANS PROGRAMI

FPGA TABANLI YAPAY SİNİR AĞI KULLANILARAK BUĞDAY TÜRLERİNİN SINIFLANDIRILMASI

Murat LORTOĞLU

KONYA

HAZİRAN 2019

T.C.

KTO Karatay Üniversitesi

Fen Bilimleri Enstitüsü

ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

TEZLİ YÜKSEK LİSANS PROGRAMI

FPGA TABANLI YAPAY SİNİR AĞI KULLANILARAK BUĞDAY TÜRLERİNİN SINIFLANDIRILMASI

Murat LORTOĞLU

KONYA

HAZİRAN 2019

Fen Bilimleri Enstitü Onayı

Prof. Dr. Hüseyin Bekir YILDIZ
Fen Bilimleri Enstitüsü Müdürü

Bu tezli yüksek lisans tezinin yapılması gereken bütün gerekliliklerinin yerine getirdiğini onaylıyorum.

Dr. Öğr. Üyesi Hüseyin Oktay ALTUN
Anabilim Dalı Başkanı

Murat LORTOĞLU tarafından hazırlanan "FPGA TABANLI YAPAY SİNİR AĞI KULLANILARAK BUĞDAY TÜRLERİNİN SINIFLANDIRILMASI" başlıklı bu çalışma, 26/06/2019 tarihinde yapılan savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından yüksek lisans tezi olarak kabul edilmiştir.

Dr. Öğr. Üyesi Ali ÖZTÜRK
Tez Danışmanı

Jüri Üyeleri

Başkan: Dr. Öğr. Üyesi Ali ÖZTÜRK .....

Üye: Dr. Öğr. Üyesi Şekip Engin MENDİ .....

Üye: Dr. Öğr. Üyesi İlker Ali ÖZKAN .....

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu tezde sunduğum çalışmanın özgün olduğunu bildirir aksi bir durumda aleyhime doğabilecek tüm hak ve kayıplarını kabullendiğimi beyan ederim.

26/06/2019

Murat LORTOĞLU

TEŞEKKÜR

Tez çalışmasının bütün aşamalarında değerli bilgi ve deneyimleriyle bana yardımcı olan, tez süresi boyunca bana büyük emeği geçen ve bana rehberlik eden tez danışmanım sayın hocam Dr. Öğr. Üyesi Ali ÖZTÜRK'e katkılarından dolayı teşekkür ve saygılarımı sunarım. Benim bu günlere gelmemi sağlayan ve bana her zaman destek olan sevgili aileme sonsuz teşekkürlerimi sunarım.

Murat LORTOĞLU

ÖZET

FPGA TABANLI YAPAY SİNİR AĞI KULLANILARAK BUĞDAY TÜRLERİNİN SINIFLANDIRILMASI

LORTOĞLU, Murat

Yüksek Lisans- Elektrik ve Bilgisayar Mühendisliği Ana Bilim Dalı

Tez Danışmanı: Dr. Öğr. Üyesi Ali ÖZTÜRK

Haziran 2019

Günümüzde teknoloji akıl almaz bir hızda gelişmekte insanların hayatını kolaylaştırmaktadır. Artık ordularda piyade robotlar görev almakta, insansız hava araçları silahlı/silahsız olarak kullanılarak ülkelere büyük üstünlük sağlamaktadır. İnsansız marketlerde hiçbir görevli ile karşılaşmadan alışveriş yapılabilmekte ve kasalarda beklemeden alışveriş tamamlanabilmektedir. Tüm bu araçlar/uygulamalar özellikle yapay zeka teknolojilerindeki son gelişmeleri gözler önüne sermektedir. Ancak bu teknolojilerin uygulama alanlarında yoğun veri işleme ve hızlı karar alabilme gereklilikleri günümüzde paralel işlem yapan platformları öne çıkarmaktadır. Bu tez çalışmasında ülkemizde alanında uzman kişiler tarafından yapılan buğday sınıflandırması işlemi nasıl hızlı ve objektif bir şekilde yapılabilir sorusuna çözüm aranmaktadır. Bu amaç doğrultusunda seri ve paralel işlem yapan platformlar üzerinde yapay sinir ağları kullanılmış, paralel işlem yapan platformların üstünlükleri ortaya konmuştur. Paralel işlem yapan platform olarak FPGA seçilmiş, FPGA için VHDL dilinde yazılan yapay sinir ağı kodunun doğrulaması ve simülasyonu ModelSim programında gerçekleştirilmiştir. Yapay sinir ağlarının çok uzun süren eğitim safhası paralel işlem yapan platformların aynı anda birden fazla işlem yapabilme yeteneği kullanılarak kısaltılmaya çalışılmıştır.

Anahtar kelimeler: Yapay Sinir Ağları, FPGA, Sınıflandırma, ModelSim

ABSTRACT

CLASSIFICATION WITH ARTIFICIAL NEURAL NETWORK ON FPGA

LORTOĞLU, Murat

M.Sc. - Electrical and Computer Engineering

Advisor: Assist Prof. Dr. Ali ÖZTÜRK

June 2019

Nowadays technology makes life easier for people developing at an unbelievable pace. Infantry robots are now employed in armies, and unmanned aerial vehicles are used as armed / unarmed. In the unmanned markets, shopping can be done without encountering any staff and the shopping can be completed without waiting in the crates. All these tools / applications reveal the latest developments in artificial intelligence technologies. However, the intensive data processing and rapid decision-making requirements in the application areas of these technologies make parallel processing platforms stand out today. In this thesis, it is sought to solve the question of how wheat classification process can be done quickly and objectively by experts in our country. For this purpose, artificial neural networks have been used on serial and parallel platforms and the advantages of parallel platforms have been demonstrated. FPGA was selected as the parallel processing platform, and validation and simulation of artificial neural network code written in VHDL language for FPGA was performed in ModelSim program. The training phase of the artificial neural networks has been tried to be shortened by using parallel processing platforms' ability to perform more than one process at the same time.

Keywords: Artificial Neural Networks, FPGA, Classification, ModelSim

İÇİNDEKİLER

ÇİZELGELERİN LİSTESİ

ŞEKİLLERİN LİSTESİ

KISALTMALAR

Kısaltmalar	Açıklama
YSA	Yapay Sinir Ağları
FPGA	Alanda Programlanabilir Kapı Dizileri
CART	Sınıflandırma ve Regresyon Ağacı
UART	Evrensel Asenkron Alıcı Verici
DGA	Diferansiyel Gelişim Algoritması
HD	High Definition
VHDL	Yüksek Hızlı Entegre Devreler Donanım Tanımlama Dili
XOR	Özel Veya
TH	Toplam Hata
PLC	Programlanabilir Lojik Cihaz
PAL	Programlanabilir Dizi Lojik
PLA	Programlanabilir Lojik Dizi
PALCE	Silinebilir PAL
GAL	Jenerik PAL
SPLD	Basit PLD
JTAG	Ortak Test Eylem Grubu
CPLD	Karmaşık PLD
EEPROM	Elektronik Olarak Silinebilir Programlanabilir Salt Okunur Bellek
SRAM	Rastgele Erişimli Bellek
ROM	Salt Okunur Bellek
AND	Ve
OR	Veya
SOP	Çarpımların Toplamı
PROM	Programlanabilir Salt Okunur Bellek
EPROM	Silinebilir Programlanabilir Salt Okunur Bellek
I/O	Giriş/Çıkış
PMB	Programlanabilir Mantık Bloğu
LUT	Arama Tabloları
PCI	Çevre Birim Bileşen Bağlantısı
HDL	Hızlı Entegre Devreler Donanım Tanımlama Dili
EDA	Elektronik Tasarım Otomasyonu
CPU	Merkezi İşlemci Birimi

1. GİRİŞ

Teknolojik gelişmeler insanlık tarihi boyunca hayatımızı doğrudan değiştiren ve etkileyen en önemli faktörlerden biridir. Özellikle son yıllarda teknolojik gelişmeler akıl almaz bir şekilde hızlanmış ve günlük hayatımızdaki değişim sıklığı artmıştır.

Teknolojideki bu akıl almaz hızda yaşanan gelişmeler araştırmacı ve bilim insanlarını yeni araştırma konularına yönlendirmektedir. Artık ordular kendi kendini eğiten yeni robot askerlerden oluşmaya başlamıştır. Savaş uçakları insansız hava araçlarına dönüşmekte, ya kendi kendine otonom uçmakta ya da yerden uydular aracılığıyla yönetilmektedir.

İnsanların elinde devasa bilgi yığınları olup, bu bilgiyi kullanabilen, sınıflayabilen, faydalı bilgiye çevirebilen toplumlar bir adım öne çıkmakta, geleceği şekillendirmektedir.

Bu noktada günümüzde iki yeni teknoloji öne çıkmaktadır. Bunlardan ilki Yapay Sinir Ağları (YSA) ve diğeri paralel işlem yapabilen Alanda Programlanabilir Kapı Dizileridir (FPGA).

YSA günümüzde birçok alanda problemlerin çözümünde kullanılan temelde insan beynini taklit eden bir ağ sistemidir. YSA'lar sınıflandırma, görüntü işleme, süreç modelleme, geleceği tahmin algoritmalarında, kontrol mekanizmalarında vb. birçok alanda yaygın olarak kullanılmaktadır.

FPGA'lar isminden de anlaşılacağı üzere üretimi tamamlanmış ama yapılandırılması tamamlanmamış programlanabilir mantıksal bloklardan oluşur. Bu mantıksal blokları giriş-çıkış blokları çevreler. Mantıksal bloklar kullanıcının amacına uygun olarak karmaşık fonksiyonları yerine getirebilmek için programlanabildikleri gibi basit mantıksal kapıların işlevlerini yerine getirmek için de programlanabilmektedirler. İmalat sürecinden sonra mantık bloklarının, giriş-çıkış bloklarının ve ara bağlantıların programlanabilmesinden dolayı alanda programlanabilir ismi verilmiştir. Ayrıca çoğu FPGA hafıza ve aritmetik işlem bloklarına sahiptir.

Yoğun işlem yüküne sahip ileri beslemeli ağ ile geriye yayılım algoritmalarını kullanan YSA'ların gömülü sistemler üzerinde gerçekleştirilmesi zaman ve maliyetten önemli kazanımlar sağlaması nedeniyle, FPGA'lar YSA'lar için kullanılabilecek en uygun platformlardan biridir.

Bu tez çalışmasında ülkemizde alanında uzman kişiler tarafından yapılan buğday sınıflandırması işlemi nasıl hızlı ve objektif bir şekilde yapılabilir sorusuna çözüm aranmaktadır. Bu amaç doğrultusunda seri ve paralel işlem yapan platformlar üzerinde ileri beslemeli ağ ile geriye yayılım algoritmalarını kullanan yapay sinir ağları oluşturulmuş, paralel işlem yapan platformların üstünlükleri ortaya konmuştur. Paralel işlem yapan platform olarak FPGA seçilmiş, FPGA için VHDL dilinde yazılan yapay sinir ağı kodunun doğrulaması ve simülasyonu ModelSim programında gerçekleştirilmiştir. Yapay sinir ağlarının çok uzun süren eğitim safhası paralel işlem yapan platformların aynı anda birden fazla işlem yapabilme yeteneği kullanılarak kısaltılmaya çalışılmıştır.

1.1. Literatür Araştırması

Literatürde, FPGA tabanlı YSA kullanılarak sınıflandırma problemlerinin çözümünde çeşitli çalışmalar mevcuttur. Bunlardan bazıları aşağıda verilmiştir.

Tuncay Soylu, gerçek zamanlı olarak yüksek doğrulukta ve hızlarda trafik sınıflandırma yapabilmek için donanım üzerinde gerçeklenen makine öğrenmesi tabanlı sınıflandırma yöntemlerini incelemiş, pipeline (paralel boru hatlı) mimariler üzerinde uygulanan makine öğrenmesi tabanlı yüksek doğruluk ve hızlarda sınıflandırma yapabilen Genişletilmiş Basit CART (Sınıflandırma ve Regresyon Ağacı) mimarisini önermiştir. Benzer katkıları sağlamak maksadıyla her bir uygulama sınıfından bir ağaç elde edilerek ağaçları Bitmaplerle zenginleştirilmiş iki aşamalı hibrit yapı olan Basit CART Ormanları mimarisi üzerinde çalışmıştır. Sonuç olarak düşük sınıflandırma gecikmesi ve yüksek doğruluk sağlayan tek adımlı Bitmap Kodlu Basit CART veri yapısını FPGA üzerinde tasarlamıştır [1].

Mustafa Manisalı, yapıları gereği lineer olmayan davranışa sahip dört motorlu insansız hava aracının bulanık mantık algoritmaları kullanılarak kontrol algoritmasını oluşturmuş, bu algoritmayı labview kullanarak programlamış ve FPGA üzerinde test etmiştir [2].

Erdem Köse, yaptığı çalışmada şablon öğrenmesi yapabilen sayısal bir hücresel YSA emülasyon işlemcisi tasarlayarak bu tasarımı DigilentAtlys geliştirme kartı üzerinde gerçekleştirmiştir. İşlemci hesaplama ve denetleme birimi olmak üzere iki parçadan oluşmaktadır. İşlemci 41.5 santigrad derece sıcaklıkta 100 MHz frekansta 1W güç

tüketmektedir. Denetleme biriminde ağ ile bağlantı kısımları, rastgele sayı üreteci ve Microblaze mikroişlemci bulunmaktadır. Microblaze mikroişlemcisi Ethernet, UART (Evrensel Asenkron Alıcı Verici) ve rastgele erişimli belleği denetlemektedir. Hesaplama biriminde ön bellek ve sonlu durum makinesi ile aritmetik birim bulunmaktadır [3].

Volkan Meriç, yaptığı çalışmada ayrık zamanlı Hücresel Yapay Sinir Ağı işlemcisi mimarisi sunmuştur. Hücresel Yapay Sinir Ağı bir işlemcinin aritmetik fonksiyonu olarak tasarlanmıştır. İşlemci yapay sinir ağını bir fonksiyon olarak çağırabilmektedir. Komut çağrıldığında 3x3'lük yapay sinir ağı fonksiyonunu n kere girişe uygular. Komut çıkışında üretilen değer bir sonraki komutta kullanılmak üzere farklı hafıza bölümlerinde tutulmaktadır [4].

Ali Rıza Yılmaz, diferansiyel gelişim algoritması (DGA) kullanan bir YSA'yı kayan noktalı sayı formatında FPGA üzerinde gerçeklemiştir. Bu işlemi yaparken önce optimizasyon için kullanılacak olan DGA MATLAB ortamında gerçeklemiş ve iki bilinmeyenli denklem ile doğrulanmıştır. Sonra DGA algoritmasının YSA eğitimine uygunluğunu test etmek için Konik Kesit Fonksiyonlu Sinir Ağı ve Çok Katmanlı Algılayıcı ağlarının eğitimi gerçeklemiştir [5].

Mehmet Ali Çavuşlu, yaptığı çalışmada geriye yayılım ve Levenberg&Marquardt algoritmaları ile parçacık sürü optimizasyon ve yapay arı koloni algoritmalarını kullanarak YSA eğitimini FPGA üzerinde donanımsal olarak gerçeklemiştir. FPGA programlanırken kayan noktalı sayı formatı kullanılmıştır. Donanım gerçeklemesi araç plaka bölgesi belirleme ve dinamik sistem tanıma problemleri kullanılarak test edilmiştir [6].

Murathan Alpay, 1080p@60 full-HD video görüntülerini işleyebilen ayrık zamanlı iki ve çok katmanlı hücresel sinir ağı mimarisi tasarımlarını FPGA üzerinde gerçeklemiştir. Gerçekleme safhasında iki katmanlı hücresel sinir ağı mimarisi genel matematiksel modeldeki şablonların tümünü gerçekleyebiliyorken, çok katmanlı hücresel sinir ağı mimarisi için yeni bir katman komşuluğu kavramı tanımlanmıştır. Buna göre katmanlar arası bağlantılar belirli bir komşuluk mesafesiyle sınırlanmıştır [7].

İlker Ali Özkan, Elektromanyetik filtrelerin yüksek performanslı çalışmasını sağlayacak ve performansını etkileyen parametreleri değerlendirip bu parametrelerdeki değişimleri erken algılayan FPGA tabanlı zeki hibrit bir kontrolör tasarlamış ve gerçekleştirmiştir. Gerçekleştirme safhasında yapay zeka tekniklerinden bulanık kontrol, sinirsel bulanık kontrol ve adaptif bulanık kontrol yapıları kullanılarak beş farklı kontrolör tasarlanmış ve avantaj ve dezavantajları göz önüne alınarak elektromanyetik filtrelerin kontrolü için zeki hibrit bir kontrolör yapısı oluşturulmuştur. Bu kontrolör bir erken algılama sistemi ile güçlendirilerek teknolojik parametrelerin değişiminin elektromanyetik filtre performansını etkilemesinin önüne geçilmiştir. FPGA tabanlı olarak tasarlanan kontrolörler endüstriyel ortamlarda bilgisayarlardan bağımsız olarak çalışabilmektedir [8].

Evren Cesur, yaptığı çalışmada 1080x1920@60 Hz yüksek çözünürlüklü basit taramalı video işaretini işleyebilen Gabor benzeri hücresel sinir ağı filtresinin tasarımı ve gerçeklemesi yapılmıştır. Yöntem olarak hesaplama iş yükünün zamanda bölünmesi temel alınmıştır [9].

Nerhun Yıldız, yaptığı çalışmada 1080p@60 full-HD video görüntülerini işleyebilen gerçek zamanlı gelişmiş bir sayısal YSA önermiş, bu YSA'yı VHDL (Yüksek Hızlı Entegre Devreler Donanım Tanımlama Dili) dilinde kodlayarak iki farklı FPGA üzerinde gerçeklemiştir [10].

Günay Temür, YSA'ların otomatik olarak FPGA platformlarında gerçeklemesi için bir denetleyici tasarım aracı ile YSA veri yolunu birleştirerek bir YSA sistemi geliştirmiştir. Bu sistem sayesinde YSA'ların FPGA'lere uygulanmasını otomatikleştirerek tasarım ve uygulama süresi kısaltılmış debug aşaması ve uzman gereksinimi ortadan kaldırılmıştır [11].

Namık Kemal Sarıtekin, yaptığı çalışmada FPGA çiplerine uygulanabilecek formatta tasarlanmak istenen YSA'lar için otomatik veri yolu tasarımı yapan bir araç geliştirmiştir [12].

Ahmet Turan Özdemir, yaptığı çalışmada taşınabilir bir YSA tabanlı otomatik aritmi sınıflandırıcıyı tek bir FPGA çipi üzerinde donanımsal olarak gerçeklemiştir [13].

Necla Yılmaz, çip üzerinde XOR ve bir sensör doğrusallaştırma problemi sabit noktalı sayı sistemi tabanlı eğitilebilir bir YSA yapısını FPGA platformunda gerçeklemiştir [14].

Alper Uçar, fonem tabanlı Türkçe ifade tanıma sistemlerinde kullanılacak radyal tabanlı bir fonksiyon ağı tasarlayarak FPGA üzerinde donanımsal olarak gerçeklemiştir [15].

Javier Gomez Casado, bir biyolojik nöronun davranışını inceleyip simüle ederek sinir dürtülerinin, dendritlerden aksonlara, nöronun gövdesi üzerinden nasıl iletildiğini araştırıp, bu davranışı VHDL dili kullanarak FPGA üzerinde aynı şekilde çalışan bir sistem elde etme hedeflemiştir [16].

Jonathan Tettero yaptığı çalışmada, Konvolüsyonel Sinir Ağları ρ-VEX işlemcisine nasıl uygulanabilir ve işlemci performansını optimize etmek için ne yapılabilir sorularına cevap aramıştır [17].

Gabriele-Maria LOZITO ve arkadaşları yaptıkları çalışmada FPGA mimarisi üzerindeki yapay sinir ağları uygulamalarında kayan nokta hızlandırıcılar (floating point accelerators) kullanarak farklı mimari çözümlerin analizini yapmışlardır. Bu çözümlerin FPGA kaynak kullanımı ve elde edilen performanslarının kıyaslanması yapılmıştır [18].

Teng Wang ve arkadaşları yaptıkları çalışmada sistematik olarak özel problemler için tasarlanmış FPGA tabanlı yapay sinir ağı hızlandırıcılarını algoritma ve şablon tabanlı olarak incelemişler, tasarımları karşılaştırarak işlemci ve ağ modelleri kıyaslanmıştır [19].

Zbigniew Hajduk yaptığı çalışmada ileri beslemeli yapay sinir ağının iki farklı uygulamasını FPGA üzerinde gerçekleştirmiş, uygulamaların kaynak gereksinimlerini ve hesaplama hızlarını ortaya koymuştur [20].

Giulia Altamura yaptığı çalışmada konvolüsyonel yapay sinir ağı için FPGA tabanlı bir donanım hızlandırıcı tasarlamış ve Xilinx Virtex 7 cihazı üzerinde gerçeklemiştir [21].

Ahmet Babalık yaptığı çalışmada, buğday yığınlarının içerisindeki son ürün kalitesini olumsuz yönde etkileyen süne tahribatlı tanelerin, yabancı ot tohumlarının, yabancı maddelerin ve özellikle sert camsı türlerde önemli olan dönmüş buğday tanelerinin

tespitinde kullanılacak yapay sinir ağı tabanlı gerçek zamanlı bir sistem geliştirmiştir [22].

Müslüm Öztürk ve arkadaşları yaptıkları çalışmada yapay sinir ağlarının eğitiminde kullanılmak üzere .NET ortamında C# programlama dili kullanılarak görsel arayüze sahip bir eğitim yazılımı geliştirmiştir. Gerçekleştirilen yapay sinir ağı sınıflandırma yazılımı buğday türü sınıflandırılması örneğine uygulanmış ve başarılı sonuçlar alınmıştır [23].

Alper Taner ve arkadaşları yaptıkları çalışmada makarnalık buğday çeşitlerinin sınıflandırmasını yapmak amacıyla bir Yapay Sinir Ağları (YSA) modeli geliştirmiştir. Bu amaçla makarnalık buğdaylara ait fiziksel özellikler belirlenmiş ve YSA teknikleri kullanılmıştır [24].

1.2. Tezin Amacı

Bu tezin amacı yoğun işlem yükü ve hesaplama gerektiren YSA'ları yazılımsal olarak gerçekleştirirken seri işlem yapan platformlar ile paralel işlem yapan platformları kıyaslayıp avantaj ve dezavantajlarını ortaya çıkarmaktır.

Ayrıca özellikle ülkemizde alanında uzman kişiler tarafından yapılan buğday sınıflandırması işlemi nasıl hızlı ve objektif bir şekilde yapılabilir sorusuna çözüm aranmaktadır. Bu amaç doğrultusunda seri ve paralel işlem yapan platformlar üzerinde ileri beslemeli ağ ile geriye yayılım algoritmalarını kullanan yapay sinir ağları oluşturulmuş, paralel işlem yapan platformların üstünlükleri ortaya konmuştur. Buğday sınıflandırılması işlemi buğday alımında buğday fiyatını ortaya çıkardığından bu işlemin insan faktörünü karıştırmadan objektif bir şekilde makinalar tarafından yapılması önemlidir. İşlemin hızı üreticinin emeğinin karşılığını bir an önce alması bakımından yine üzerinde çalışılması gereken bir konu olduğundan tezimin amaçlarından biridir. Genel olarak eğitilmiş bir ağın kullanılmasında paralel ve seri işlem yapan platformlar arasında çok büyük bir süre farkı oluşmamaktadır. Bu tez de yapay sinir ağının eğitim süresinin kısaltılması amaçlanmaktadır.

2. MATERYAL VE YÖNTEM

2.1. YAPAY SİNİR AĞLARI

Yapay sinir ağları (YSA), insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri herhangi bir yardım almadan otomatik olarak gerçekleştirmek amacı ile geliştirilen bilgisayar sistemleridir [25].

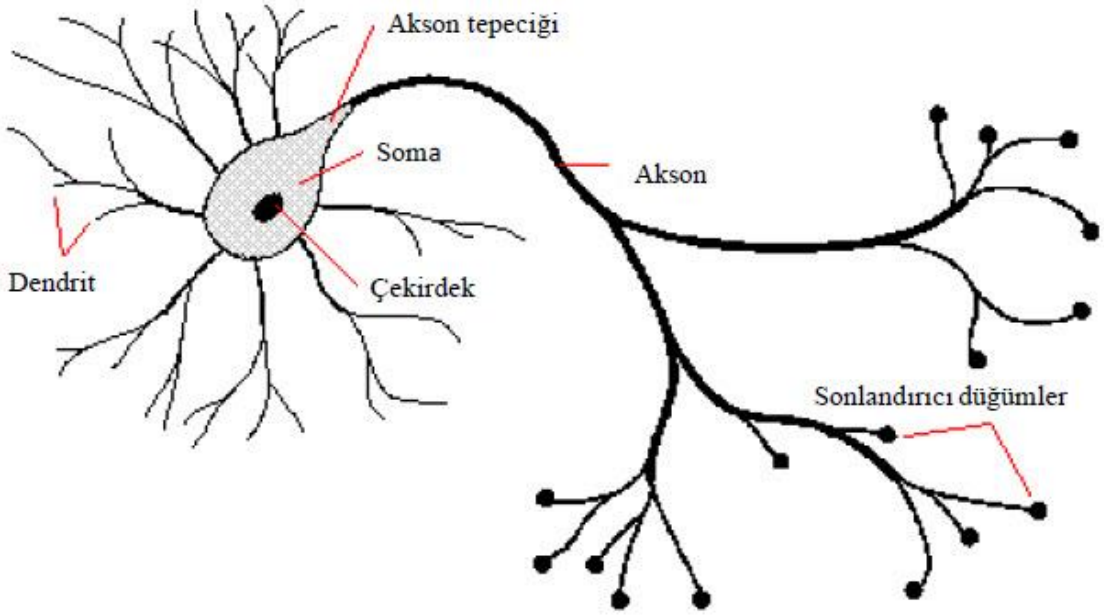
İsminden de anlaşılacağı üzere YSA biyolojik sinir ağlarını taklit etmek üzere kurgulanmıştır. Bu amaca yönelik olarak insan beyninin nasıl çalıştığını anlamak ve onu modellemek YSA tasarımcılarının ilk hedefleri arasında yer almıştır. İnsan beyni birbirlerine bağlantılı trilyonlarca nöron denilen sinir hücrelerinden meydana gelmektedir. YSA'ları da birbirlerine farklı ağırlık seviyelerinde bağlantılı katmanlardan oluşan yapay sinir hücreleri meydana getirir. Günümüzde anlaşılmaktadır ki insan beynini çok hızlı çalışan mükemmel bir bilgisayar gibi görebiliriz. Henüz yapay sinir ağları insan beynini tam anlamıyla taklit etmekten çok uzaktır. Eğer insan beyninin kapasitesinin çok küçük bir oranında kapasiteye sahip çalışabilen bir bilgisayar yapılabilse fevkalade kontrol edebilme ve bilgi işleme mekanizmaları geliştirmek ve mükemmel sonuçlar elde etmek mümkün olabilir. İnsan beynindeki sinir ağlarının başarımı küçümsenemeyecek kadar yüksektir ve çok karmaşık olayları işleyebilecek kabiliyettedir. Yapay sinir ağları ile bu başarımı ve kabiliyeti bilgisayarlara kazandırmak hedeflenmektedir.

İstatistik biliminde veya zamana dayalı olaylarda tüm girdi ve çıktılar sıralı işlenmektedir. Oysa yapay sinir ağlarında tüm girdi ve çıktılar paralel olarak işlenmektedir. Yapay sinir ağlarındaki mimari yapı bir ışık prizmasına benzetilebilir. Işık prizması toplu gelen ışığı renklerine ayırmaktadır. Yapay sinir ağlarındaki mimarı yapı da gelen verileri basit bileşenlerine ayırdıktan sonra bunları istenen çıktılara uygun olacak şekilde yeniden birleştirir. Yapay sinir ağlarındaki mimariyi bir nehir üzerindeki köprü yapısına benzetebiliriz. Yapay sinir ağları girişler ve çıkışlar arasında köprü olur. Nasıl köprü mimarisinde kiriş, kolon, döşeme vb. parçalar arasında fiziksel ve matematiksel kurallara uygun olarak yapılan bağlantılar olduğu gibi yapay sinir ağlarında da hücreler ve katmanlar arasında benzer bağlantılar oluşturulur.

2.1.1. Biyolojik Sinir Hücresi

Öğrenme, düşünme, algılama, hatırlama vb. temel bilişsel insan davranışlarının merkezinde sinir hücreleri bulunmaktadır. Sinir sisteminin temel işlem elemanları sinir hücreleridir. Temel bir biyolojik sinir hücresi Şekil 2.1'de görüldüğü gibi çekirdek, dendritler, aksonlar ve sinapslerden (sonlandırıcı düğümler) oluşmaktadır. Bu parçaların görevleri Çizelge 2.1'de verilmektedir.

Tipik bir sinir hücresi, dendritleri aracılığıyla dış kaynaklardan gelen elektrik darbelerinden üç şekilde etkilenir. Gelen elektrik darbelerinden bazıları hücreyi uyarırken bazıları bastırılır. Bazıları ise hücre davranışında değişikliğe yol açar. Eğer hücre yeterince uyarılırsa aksonundan aşağıya bir elektriksel işaret göndererek tepkisini gösterir. Genelde biyolojik sinir hücrelerinde tek bir akson üzerinde çok sayıda dallar bulunur. Sinir hücresinin aksonundan ilerlemekte olan elektriksel işaret dallara, alt dallara ve sonunda başka hücrelere ulaşarak onların davranışlarını etkiler. Sinir hücreleri diğer başka sinir hücrelerinden gelen verileri elektriksel darbe biçiminde alırlar. Sinir hücresinin yaptığı iş ise bu gelen verilerin karmaşık ve dinamik bir toplamını alarak sonucu elektrik darbesi şeklinde aksonundan aşağıya göndererek çok sayıda başka sinir hücresine iletmektir. Sonuç olarak sinir sistemindeki sinir hücrelerinin temel görevi işaret alıp, işareti işleyip göndermek yani bilgi alışverişidir.



Şekil 2.1. İnsan Sinir Hücresi Yapısı

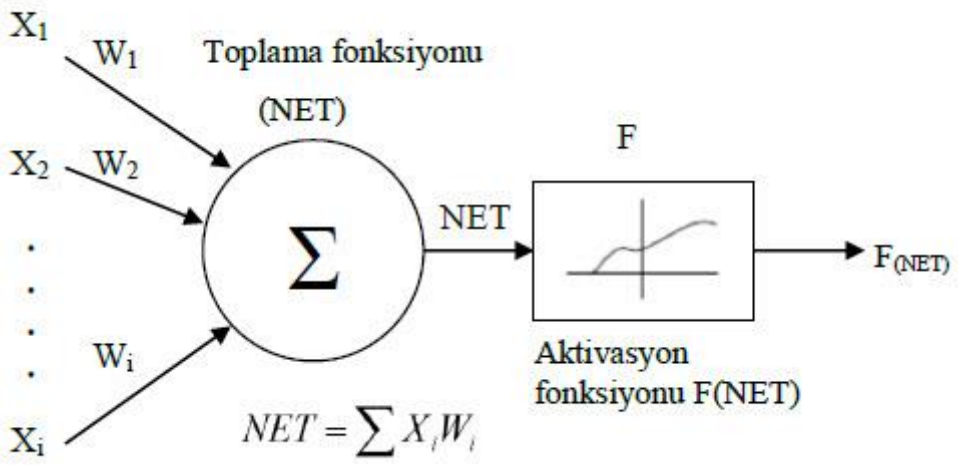
Çizelge 1.1. Sinir hücresini oluşturan parçalar ve görevleri [26]

Parça	**Görevi**
Çekirdek	Sinir hücresinin çekirdeğidir.
Akson	Hücrenin diğer hücrelere kimyasal iletimde bulunduğu ucudur. Bir hücre sadece bir adet aksona sahip olmasına rağmen her aksonun birden fazla ucu vardır. Boyları 1mm'den küçük olabileceği gibi 1m'den de büyük olabilir.
Dendrit	Hücrenin alıcısıdır. Kimyasal olarak algılama işlemi yapar. Bir sinir hücresi birden fazla dendrite sahiptir ve her dendritin de birden fazla ucu vardır.

Sinaps	Bir sinir hücresinin aksonu ile diğer sinir hücresinin dendritini birbirine bağlayan bağlantı elemanıdır.

2.1.2. Yapay Sinir Hücresi

Yapay sinir hücresi insan sinir hücresi temel alınarak oluşturulan matematiksel modellerdir. Şekil 2.2'de i adet girişi ve tek çıkışı olan yapay bir sinir hücresi verilmiştir. Şekil 2.2'deki yapay sinir hücresinde X_i sinir girişlerini, W_i sinir hücresi ile girişler arasındaki ağırlık katsayılarını ve F aktivasyon (transfer) fonksiyonunu ifade etmektedir. Transfer fonksiyonu sinir hücresinin toplamdan sonraki çıkışını bir işlemden geçirdikten sonra hücre çıkışı F(NET)'i oluşturur. Transfer fonksiyonu çoğunlukla lineer değildir.



Şekil 2.2. Yapay Sinir Hücresi

Giriş değerleri ile ağırlıkların çarpımının toplamını ifade eden NET fonksiyonu;

$$NET = \sum_{i=1}^{n} X_i W_i \tag{2.1}$$

Şeklinde hesaplanır. Sinir çıkışını oluşturan $F_{(NET)}$ fonksiyonu ise;

$$F_{(NET)} = F\left(\sum_{i=1}^{n} X_i W_i\right) \quad (2.2)$$

Formülü ile hesaplanmaktadır. Yapay sinir hücresinin temel elemanları aşağıda açıklanmıştır.

1. Girişler: Girişler $(X_1, X_2, \ldots\ldots X_N)$ diğer bir sinir hücresinden veya dış dünyadan gelen bilgilerdir. Ağın öğrenme aşamasında kullanılması istenen örnekler tarafından belirlenir.

2. Ağırlıklar: Ağırlıklar $(W_1, W_2, \ldots\ldots W_N)$ yapay sinir hücresine gelen bilginin etkisini belirleyen katsayılardır. Yapay sinir ağlarında herbir hücrenin herbir girişi için bir ağırlık katsayısı vardır.

3. Toplama fonksiyonu: Toplama fonksiyonu hücreye gelen net girdiyi hesaplamakta kullanılır. En yaygın olanı yukarıda NET fonksiyonu olarak ifade edilen fonksiyondur. Bir yapay sinir ağında bulunan bütün yapay sinir hücrelerinin tamamı aynı toplama fonksiyonuna sahip olmaları gerekmez. Her yapay sinir hücresi farklı bir toplama fonksiyonuna sahip olabilecekleri gibi hepsi aynı toplama fonksiyonunu kullanabilirler. Çizelge 2.2'de literatürde kullanılan bazı toplama fonksiyonlarına örnekler verilmiştir.

Çizelge 2.2. Literatürde kullanılan bazı toplama fonksiyonları [25]

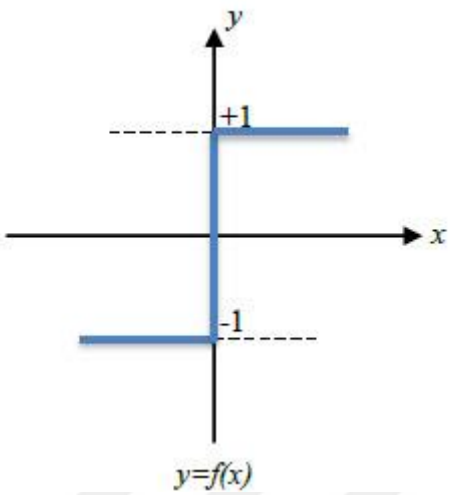
NET GİRİŞ	**AÇIKLAMA**
Çarpım Net=$\prod_i G_i A_i$	Ağırlık değerleri girdiler ile çarpıldıktan sonra bulunan değerler birbirleri ile çarpılarak net girdi hesaplanır.
Maksimum Net= Max($G_i A_i$) , i=1…..N	Bütün girdiler kendi ağırlıkları ile çarpıldıktan sonra en büyüğü yapay sinir hücresinin net girdisi kabul edilir.
Minimum Net= Min($G_i A_i$) , i=1…..N	Bütün girdiler kendi ağırlıkları ile çarpıldıktan sonra en küçüğü yapay sinir hücresinin net girdisi kabul edilir.

Çoğunluk Net= $\sum_i sgn(G_i A_i)$	Bütün girdiler kendi ağırlıkları ile çarpıldıktan sonra pozitif ve negatif olanların sayısı bulunup büyük olan sayı sinir hücresinin net girdisi kabul edilir.
Kümülatif toplam Net= Net(eski)+$\sum_i (G_i A_i)$	Ağırlık değerleri girdiler ile çarpıldıktan sonra bulunan değerler birbirleri ile toplanarak daha önce gelen bilgilere eklenip net girdi hesaplanır.

4. Transfer fonksiyonu: Toplama işlevi sonucu elde edilen toplam hücre giriş değerini işleyerek hücrenin bu girdiye karşılık nasıl çıktı üreteceğini belirleyen fonksiyonlardır. Hücrenin işlevine göre çeşitli türlerde transfer fonksiyonları kullanılabilir. Bir yapay sinir ağındaki sinir hücrelerinin transfer fonksiyonları aynı olabileceği gibi hepsi birbirinden farklı olabilir. Kompleks ve lineer olmayan problemlerin çözümü için kullanılan yapay sinir ağlarındaki transfer fonksiyonları nonlineer olmalıdır. Sıklıkla kullanılan transfer fonksiyonları şöyledir;

4.1. Keskin Sınırlayıcı (Signum(x)) Transfer Fonksiyonu; Mantıksal işlemler veya mantıksal sınıflandırma gerektiren problemlerin çözümünde kullanılmaktadır. Matematiksel ifadesi Denklem 2.3'de ve şekil olarak gösterimi Şekil 2.3'de verilmiştir [8].

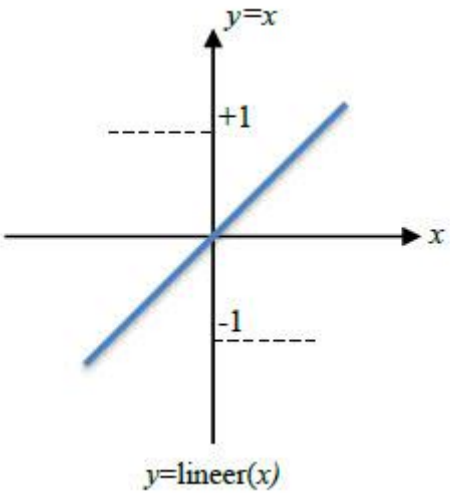
$$F(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x \leq 0 \end{cases} \tag{2.3}$$



Şekil 2.3. Signum transfer fonksiyonu grafiksel yapısı

4.2. Lineer (Lin(x)) Transfer Fonksiyonu: Toplayıcıdan gelen veriler bir α katsayısı ile çarpılarak hücre çıkışına yansıtılır. Eğer α=1 ise fonksiyon çıkışı girişine eşittir. Bu transfer fonksiyonu sıklıkla klasik işaret işleme ve regresyon analizlerinde kullanılır. Matematiksel ifadesi Denklem 2.4'de ve şekil olarak gösterimi Şekil 2.4'de verilmiştir [8].

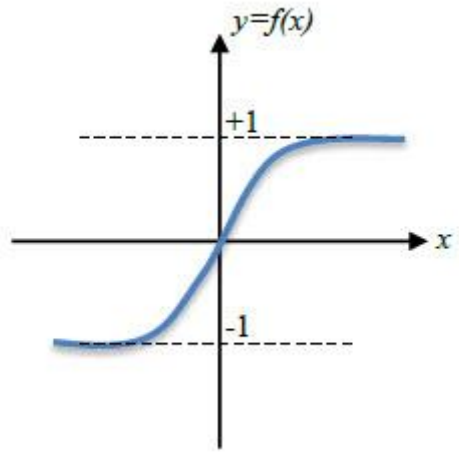
$$F(x) = \alpha * x \tag{2.4}$$



Şekil 2.4. Lineer transfer fonksiyonu grafiksel yapısı (α=1)

4.3. Tanjant Hiperbolik (Tansig(x)) Transfer Fonksiyonu: Giriş olarak -∞ ile +∞ arasında değer alabilir ve çıkış olarak -1 ile +1 arasında değer üretir. Türevi alınabilir sürekli ve doğrusal olmayan bir fonksiyon olması nedeniyle doğrusal olmayan problemlerin çözümünde kullanılır. Matematiksel ifadesi Denklem 2.5'de ve şekil olarak gösterimi Şekil 2.5'de verilmiştir [8].

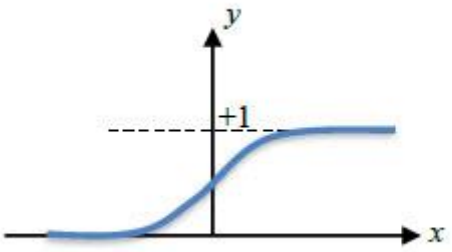
$$F(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.5}$$



Şekil 2.5. Tanjant Hiperbolik transfer fonksiyonu grafiksel yapısı

4.4. Sigmoid (Logsig(x)) Transfer Fonksiyonu: Yapay sinir ağlarında en çok kullanılan doğrusal ve doğrusal olmayan davranışlar arasında denge sağlayan sürekli artan bir fonksiyon olarak tanımlanır. Matematiksel ifadesi Denklem 2.6'de ve şekil olarak gösterimi Şekil 2.6'de verilmiştir [8].

$$F(x) = \frac{1}{1+e^{-x}} \tag{2.6}$$



Şekil 2.6. Sigmoid transfer fonksiyonu grafiksel yapısı

5. Hücrenin çıktısı: Hücrenin çıktısı F(x), transfer fonksiyonu tarafından belirlenen çıktı değeridir. Bir sinir hücresinden çıkan sadece tek bir çıktı değeri vardır. Bu değer sinir hücresinin kendisine girdi olarak gönderilebildiği gibi kendisinden sonra gelen bir veya daha fazla hücreye giriş de olabilir.

2.1.3. Yapay Sinir Ağlarının Yapısı

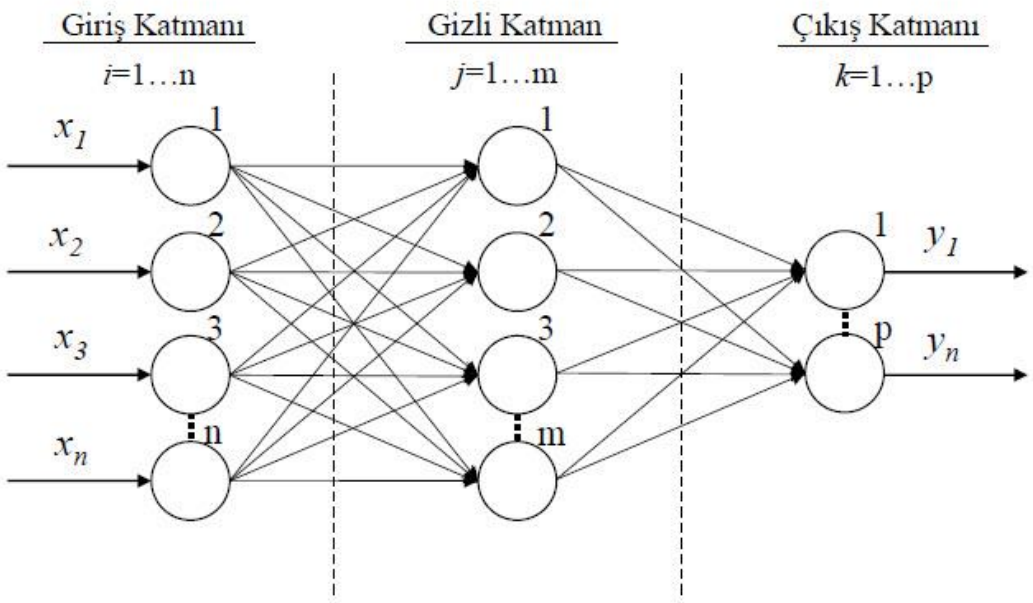
Yapay sinir ağları biyolojik insan beynini oluşturan sinir hücrelerini (nöron) matematiksel olarak taklit ederek akıllı bir sistem oluşturmaya çalışan ve bu sinir sisteminin çalışmasını elektronik ortama taşımayı hedefleyen bir bilgisayar programlama yaklaşımıdır. Bir yapay sinir ağı birçok nörondan ve birçok katmandan oluşmaktadır. Bir yapay sinir ağındaki ilk katman giriş katmanı ve son katman ise çıkış katmanıdır. Arada kalan katmanlar gizli katman veya ara katman olarak adlandırılırlar. Bir yapay sinir ağındaki ara katman sayısı birden fazla olabilir.

Yapay sinir ağları bilinen bilgi işleme yöntemlerinden farklı olarak paralellik, hata toleransı, öğrene bilirlik ve gerçekleme kolaylığı gibi özelliklere sahiptirler. Bu gibi özelliklere sahip olmasından dolayı yapay sinir ağları alışılagelmiş hesaplama yöntemlerine göre daha başarılı sonuçlar üretebilmektedirler. Yapay sinir ağlarında bilgilerin işlenmesi birbirinden bağımsız ve paralel olarak yapılmaktadır. Aynı katmandaki bağlantılar arasında zaman bağımlılığı olmadığından tamamı ile eşzamanlı çalışabilmekte ve sonuçta bilgi akış hızı artmaktadır. Paralel çalışma prensibinden dolayı bir nöronda meydana gelen hatanın tüm sisteme yayılması çok sınırlıdır. Sadece nöronun ağırlık katsayıları oranında bir etkileşim gerçekleşmektedir. Sonuç olarak yerel hatalardan en az şekilde etkilenmektedir.

Yapay sinir ağları öğrenme aşamasında bir programcılık yeteneği gerektirmeyen kendi kendine öğrenebilen düzeneklerdir. Öğrenme yapay sinir ağlarında bağlantı ağırlıklarının yenilenmesiyle olur. Elde edilen yeni bilgiler bağlantı ağırlıklarında uzun süre saklanabilmektedirler. Öğrenme yeteneği sayesinde yapay sinir ağları ile tam tanımlı olmayan problemlerin çözülmesi mümkündür. Paralel çalışan bir yapay sinir ağı modeli karmaşık bir yapıya sahip olmadığından ve basit işlemler içerdiğinden birçok sorunun çözümünde tercih edilir. Bir yapay sinir ağı için eldeki verilerin türlerine ve istenilen hedefe karar verildikten sonra beklenilen çıktıları girdilerden elde etmek için bu ağdaki bilinmeyen bağlantı değerleri ardışık yaklaşımlarla eğitilerek tespit edilir.

Yapay sinir ağlarında bulunan her nöron n. dereceden tercihen lineer olmayan bir birimdir. Nöronlar arasında bağlantılar bulunmakla birlikte her bağlantı tek yönlü iletim yoludur. Bir nöron birden fazla nörona veri aktarabilir.

Sadece iki katmandan oluşan (giriş ve çıkış katmanı) yapay sinir ağları karmaşık işlemleri hesaplama yeteneklerine sahip değildirler. Bu nedenden dolayı karmaşık problemlerin çözümünde en az bir ara katman olmalıdır. Şekil 2.7'de bir ara katmana (gizli katman) sahip üç katmanlı bir yapay sinir ağı verilmiştir.



Şekil 2.7. Üç katmanlı bir yapay sinir ağı modeli

Bu tez çalışmasında çok katmanlı algılayıcı yapay sinir ağı modeli kullanılmıştır. Bu yapay sinir ağı modeli Bölüm 2.1.4'de açıklanmıştır.

2.1.4. Çok Katmanlı Algılayıcı Yapay Sinir Ağı Modeli

Çok Katmanlı Algılayıcı Yapay Sinir Ağı modeli girdi ve çıktıları arasında lineer olmayan ilişkiler olan problemlerin çözümü için Rumelhart ve arkadaşları tarafından geliştirilmiştir [25]. Hata yayma modeli veya geriye yayım modeli (back propogation network) olarak da isimlendirilmektedir. Bu model özellikle tanıma, sınıflandırma ve genelleme yapmayı gerektiren mühendislik problemlerinin hemen hemen hepsinde kullanılabilecek çok önemli bir çözüm aracıdır. Temel olarak delta öğrenme kuralı denilen bir öğrenme yöntemini kullanarak ağın beklenen çıktısı ile ürettiği çıktı arasındaki hatayı en aza indirmeyi hedeflemektedir. Bu amacını hatayı ağa yayarak gerçekleştirdiği için bu modele hata yayma modeli veya geriye yayım modeli denilmektedir.

Çok katmanlı algılayıcı yapay sinir ağı modelinin yapısı en az 3 katmandan (girdi katmanı (input layer), ara katmanlar (hidden layers) , çıktı katmanı (output layer)) oluşur. Girdi katmanı dış dünyadan gelen girdileri alarak ara katmanlara gönderir. Bu katmanda bilgi işleme olmadığı için gelen her bilgi geldiği gibi bir sonraki katmana gider. Girdi katmanında bulunan her nöron bir sonraki ara katmanda bulunan bütün nöronların hepsine bağlıdır. Ara katmanlar girdi katmanından gelen bilgileri işleyerek bir sonraki katmana gönderir [25]. Çok Katmanlı Algılayıcı Yapay Sinir Ağı modelinde birden fazla ara katman olabileceği gibi her katmanda birden fazla nöron olabilir. Ara katmandaki her nöron bir sonraki katmandaki bütün nöronlara bağlıdır. Çıktı katmanı ara katmandan gelen bilgileri işleyerek ağa girdi katmanından verilen girdilere karşılık ağın ürettiği çıktıları belirleyerek dış dünyaya gönderir. Bir çıktı katmanında birden fazla nöron olabilir. Her nöron bir önceki katmanda bulunan bütün nöronlara bağlıdır. Her nöronun sadece bir çıktısı vardır.

Çok katmanlı algılayıcı yapay sinir ağı modelinde bilgiler girdi katmanından ağa sunularak ara katmanlardan geçip çıktı katmanına gider. Böylece ağa sunulan girdilere karşılık ağın cevabı dış dünyaya iletilmiş olur. Çok katmanlı algılayıcı yapay sinir ağı modeli öğretmenli öğrenme yöntemini kullanır. Bunun için ağa hem örnek girdiler hem de örnek girdilerden elde edilmesi gereken çıktılar gösterilir. Ağ kendisine

gösterilen örnek değerlerden genellemeler yaparak bir çözüm uzayı üretmektedir. Daha sonra gösterilen yeni girdiler için bu çözüm uzayı sonuç ve çözümler üretebilmektedir.

Çok katmanlı algılayıcı yapay sinir ağı modelinin çalışma adımları aşağıda açıklanmıştır.

- Örnek uzayının toplanması: Bu adımda ağın çözmesi istenen problem için daha önce gerçekleşmiş örnek uzayı bulunur. Ağın eğitilmesi için örnek uzayı toplandığı gibi ağın test edilmesi içinde örnek uzayının toplanması gerekmektedir. Ağın eğitimi esnasında test örnek uzayı ağa hiç gösterilmez. Eğitim safhasında eğitim örnek uzayındaki örnekler ağa tek tek gösterilerek ağın problemi öğrenmesi sağlanır. Eğitim safhasının sonunda test uzayındaki örnekler ağa gösterilerek ağın performansı ölçülür. Daha önce hiç gösterilmeyen örnekler karşısındaki ağın başarısı eğitim safhasında ağın ne kadar iyi öğrenip öğrenmediğini ortaya koymaktadır.
- Ağın topolojisinin belirlenmesi: Bu adımda ağda kaç tane girdi ünitesi, kaç tane ara katman, ara katmanlarda kaçar tane nöron ve çıktı katmanında kaç tane çıktı elemanı olması gerektiği belirlenmektedir.
- Öğrenme safhasındaki değişkenlerin belirlenmesi: Bu adımda ağın öğrenme katsayısı, nöronların toplama ve transfer fonksiyonları, momentum katsayısı vb. değişkenler belirlenmektedir.
- Ağırlık katsayılarının başlangıç değerlerinin tayin edilmesi: Bu adımda nöronları birbirine bağlayan ağırlık katsayılarının ve eşik değer ünitesinin ağırlık katsayılarının başlangıç değerlerinin tayini yapılmaktadır. Genellikle ilk değerler rastgele atanır.
- Örnek uzayından örneklerin seçilmesi ve ağa gösterilmesi: Bu adımda ağın eğitime başlaması ve aşağıda anlatılan delta öğrenme kuralına göre ağırlık katsayılarını değiştirmesi için ağa girdi-çıktı değerleri belirli bir düzende gösterilir.
- Öğrenme safhasındaki ileri hesaplamaların yapılması: Bu adımda bir önceki adımda anlatıldığı şekilde ağa sunulan girdi değerleri için ağın çıktıları hesaplanır.
- Gerçekleşen çıktı ile beklenen çıktının karşılaştırılması: Bu adımda ağın ürettiği hata değerleri hesaplanır.

- Ağırlık katsayılarının değiştirilmesi: Bu adımda delta öğrenme kuralındaki geri hesaplama yöntemi uygulanarak üretilen hatanın azalması için ağırlıkların değiştirilmesi yapılır.

Bu adımlar beklenen çıktı değerleri ile gerçekleşen çıktı değerleri arasındaki fark kabul edilebilir düzeye ininceye kadar devam ettirilir. Beklenen çıktı değerleri ile gerçekleşen çıktı değerleri arasındaki fark bir eşik değerinin altına düştüğünde çok katmanlı algılayıcı yapay sinir ağı modelinin öğrenmesi tamamlanmış demektir.

2.1.5. Delta Öğrenme Kuralı

Çok katmanlı algılayıcı yapay sinir ağı modelinin öğrenme kuralı Delta Öğrenme Kuralı'dır. İki aşamadan oluşmuştur.

1. İleri Doğru Hesaplama Aşaması: Bu aşama örnek uzayındaki bir örneğin girdi katmanından ağa gösterilmesi ile başlar. Girdi katmanında herhangi bir bilgi işleme olmadığından gelen girdiler hiçbir değişikliğe uğramadan ara katmana gönderilir. Girdi katmanındaki m. nöronun çıktısı $Ç_m^i$ şu şekilde olmaktadır.

$$Ç_m^i = G_m \tag{2.7}$$

Ara katmandaki her nöron girdi katmanındaki bütün nöronlardan gelen bilgileri bağlantı ağırlıklarının (A_1, A_2……) etkisi ile alır. Önce ara katmandaki nöronlara gelen net girdi (NET_j^a) Denklem 2.8 ile hesaplanır.

$$NET_j^a = \sum_{m=1}^{n} A_{mj} Ç_m^i \tag{2.8}$$

Bu formülde A_{mj}m. girdi katmanı elemanını j. Ara katman elemanına bağlayan bağlantının ağırlık değerini göstermektedir. j. Ara katman elemanının çıktısı ise bu net girdinin transfer fonksiyonundan geçirilmesi ile hesaplanır. Kullanılan transfer fonksiyonu türevi alınabilir bir fonksiyon olmalıdır. Eğer transfer fonksiyonu olarak sigmoid fonksiyonu kullanılırsa çıktı,

$$Ç_j^a = \frac{1}{1+e^{-(NET_j^a + \beta_j^a)}} \tag{2.9}$$

Olacaktır. Burada geçen β_j ara katmanda bulunan j. Elemana bağlanan eşik değer elemanının ağırlığını göstermektedir. Ara katmanın bütün nöronları ve çıktı katmanının nöronlarının çıktıları aynı şekilde kendilerine gelen net girdinin hesaplanması ve transfer fonksiyonundan geçirilmesi sonucu belirlenir. Çıktı

katmanındaki nöronların çıktıları bulununca ileri doğru hesaplama aşaması tamamlanmış olur.

2. Geriye Doğru Hesaplama Aşaması: Bu aşamada ağa sunulan girdiler için ağ tarafından üretilen çıktı değerleri beklenen çıktı değerleri (B_1, B_2 ….) ile karşılaştırılır. Aradaki fark hata olarak kabul edilir. Hedef bu hata değerinin azaltılmasıdır. Dolayısıyla bu hata değeri ağın ağırlık değerlerine dağıtılarak bir sonraki iterasyonda hatanın azaltılması temin edilir. Çıktı katmanındaki k. Nöron için oluşan hata (E_k);

$$E_k = B_k - Ç_k \tag{2.10}$$

Olarak bulunur. Bu bir nöron için oluşan hatadır. Çıktı katmanı için oluşan toplam hata (TH) şöyle hesaplanır;

$$TH = \frac{1}{2}\sum_k E_k^2 \tag{2.11}$$

Toplam hatayı en aza indirmek için bu hatanın kendisine neden olan nöronlara dağıtılması gerekmektedir. Bu ise nöronların ağırlıklarını değiştirmek demektir. Ağırlık değiştirme işlemi çıktı katmanı – ara katman arasında olmak üzere ve ara katmanlar veya ara katman – girdi katmanı arasında olmak üzere ayrı ayrı anlatılacaktır.

Çıktı katmanı – ara katman arasındaki ağırlık değiştirilmesi işleminde çıktı katmanındaki m. nörona ara katmandaki j. Nöronu bağlayan bağlantının ağırlığındaki değişim miktarına ΔA^a denildiğinde t. iterasyonda ağırlığın değişim miktarı aşağıdaki formül ile hesaplanır.

$$\Delta A_{jm}^a(t) = \lambda \delta_m Ç_j^a + \alpha \Delta A_{jm}^a(t-1) \tag{2.12}$$

Bu formülde α momentum katsayısını, λ öğrenme katsayısını gösterir. Momentum katsayısı çok katmanlı algılayıcı yapay sinir ağı modelinin öğrenmesi esnasında yerel bir optimum noktaya takılıp kalmaması için ağırlık değişim değerinin belirli bir oranda bir sonraki değişime eklenmesini sağlarken öğrenme katsayısı ağırlıkların değişim miktarını sağlar. Formüldeki δ_m m. çıktı ünitesinin hatasını göstermektedir. Aşağıdaki şekilde hesaplanır.

$$\delta_m = f'(\text{NET}).E_m \tag{2.13}$$

Bu formüldeki f'(NET) transfer fonksiyonunun türevidir. Transfer fonksiyonu olarak sigmoid fonksiyonu kullanılırsa

$$\delta_m = Ç_m(1-Ç_m).E_m \quad (2.14)$$

Olur. Ağırlıkların t. iterasyondaki yeni değerleri Denklem (2.15)'deki gibi olacaktır.

$$A^a_{jm}(\text{t}) = A^a_{jm}(\text{t-1}) + \Delta A^a_{jm}(t) \quad (2.15)$$

Aynı şekilde eşik değer ünitesinin de ağırlıklarının değiştirilmesi gerekmektedir. Bu amaçla önce değişim miktarı hesaplanır. Eğer çıktı katmanındaki nöronların eşik değer ağırlıkları $\beta^ç$ olursa değişim miktarı

$$\Delta\beta^ç_m(t) = \lambda\delta_m + \alpha\Delta\beta^ç_m(t-1) \quad (2.16)$$

Olur. t. iterasyondaki eşik değerin yeni ağırlık değeri

$$\beta^ç_m(\text{t}) = \beta^ç_m(\text{t-1}) + \Delta\beta^ç_m(t) \quad (2.17)$$

Olacaktır.

Ara katmanlar veya ara katman – girdi katmanı arasındaki ağırlık değiştirilmesi işleminde ağırlıkların değişim miktarı ΔA^i ile gösterildiğinde

$$\Delta A^i_{kj}(t) = \lambda\delta^a_j Ç^i_k + \alpha\Delta A^i_{kj}(t-1) \quad (2.18)$$

Olur. Hata değeri δ^a ise Denklem 2.19'daki gibi hesaplanır.

$$\delta^a_j = f'(\text{NET})\sum_m \delta_m A^a_{jm} \quad (2.19)$$

Eğer transfer fonksiyonu olarak sigmoid fonksiyonu kullanılırsa hata değeri Denklem 2.20'deki gibi olacaktır.

$$\delta^a_j = Ç^a_j(1- Ç^a_j)\sum_m \delta_m A^a_{jm} \quad (2.20)$$

Hata değeri bulunduktan sonra ağırlıkların yeni değeri

$$A^i_{kj}(\text{t}) = A^i_{kj}(\text{t-1}) + \Delta A^i_{kj}(t) \quad (2.21)$$

Olarak hesaplanır. Aynı şekilde eşik değer ünitesinin yeni ağırlıkları da yukarıda anlatıldığı şekilde hesaplanır. Ara katmanlar için eşik değer ağırlıkları β^a ile gösterildiğinde değişim miktarı Denklem 2.22'deki gibi hesaplanır.

$$\Delta\beta^a_j(t) = \lambda\delta^a_j + \alpha\Delta\beta^a_j(t-1) \quad (2.22)$$

Olur. t. iterasyonda ağırlıkların yeni değerleri Denklem 2.23'deki gibi olacaktır.

$$\beta^a_j(t) = \beta^a_j(t-1) + \Delta\beta^a_j(t) \quad (2.23)$$

Sonuç olarak ağırlıkların yeni değerleri bir iterasyon için hem ileri hem de geriye hesaplamaları yapılmış olarak tamamlanmış olacaktır. Bundan sonra ikinci bir örnek

ağa verilerek sonraki iterasyona başlanır ve ağırlıklar yeniden hesaplanır. Bu işlemler ağın öğrenmesi tamamlanana kadar tekrarlanır.

2.2. FPGA TEKNOLOJİSİ

Programlanabilir Lojik Cihazlar (PLC) 1970'lerin ortalarından itibaren hayatımıza girmiştir [27]. Programlanabilir kombinasyonel mantık devreleri oluşturmak fikrinden ortaya çıkmıştır. Bununla birlikte, bir programı çalıştırabilen ancak sabit bir donanıma sahip olan mikroişlemcilerin aksine, PLC'lerin programlana bilirliği donanım düzeyinde tasarlanmıştır. Başka bir deyişle, bir PLC, donanımı belirli özelliklere göre yeniden yapılandırılabilen genel amaçlı bir çiptir.

İlk PLC'ler, programlama şemasına bağlı olarak PAL (Programmable Array Logic) veya PLA (Programmable Logic Array) olarak adlandırıldı [27]. PAL'ler sadece mantık kapıları kullandılar. Flip-flop kullanmadılar. Dolayısıyla sadece kombinasyonel devrelerin uygulanmasına izin verdiler. Bu sorunu aşmak için kısa bir süre sonra, devrenin her çıkışında bir flip-flop içeren kayıtlı PLC'ler üretildi. Kayıtlı PLC'ler ile basit sıralı işlevler uygulanabilir hale geldi. 1980'lerin başında, her PLC çıkışına ek mantık devresi eklendi. Macrocell adı verilen yeni çıkış hücresi, (flip-flop dışında) mantık kapıları ve çoklayıcıları içeriyordu. Dahası, hücrenin kendisi programlanabilir oldu ve çeşitli çalışma modlarına izin verildi. Buna ek olarak, devrenin çıkışından PLC'ye daha fazla esneklik sağlayan programlanabilir diziye bir 'dönüş' (geri bildirim) sinyali sağlandı. Bu yeni PLC yapısı jenerik PAL (GAL) olarak adlandırıldı. Benzer bir mimari PALCE (PAL CMOS/silinebilir / programlanabilir) aygıt olarak biliniyordu. Tüm bu çipler (PAL, PLA, kayıtlı PLC ve GAL / PALCE) artık topluca SPLD'ler olarak adlandırılır. GAL / PALCE cihazı, bağımsız bir pakette üretilen tek cihazdır [27].

Daha sonra, daha sofistike bir yönlendirme şeması, daha gelişmiş silikon teknolojisi ve birkaç ek özellik (JTAG desteği ve çeşitli mantık standartlarına arayüz gibi) kullanarak aynı çip üzerinde birkaç GAL cihazı üretildi. Bu yaklaşım CPLD olarak tanındı. CPLD'ler şu anda yüksek yoğunluklu, yüksek performanslı ve düşük maliyetli olmaları nedeniyle çok popülerdir (bir doların altındaki CPLD'ler bulunabilir).

1980'lerin ortalarında FPGA'ler (Alan Programlanabilir Kapı Dizileri) üretildi. FPGA'lerin herşeyden önce bahsedilmesi gereken (belki de en önemli) özellikleri

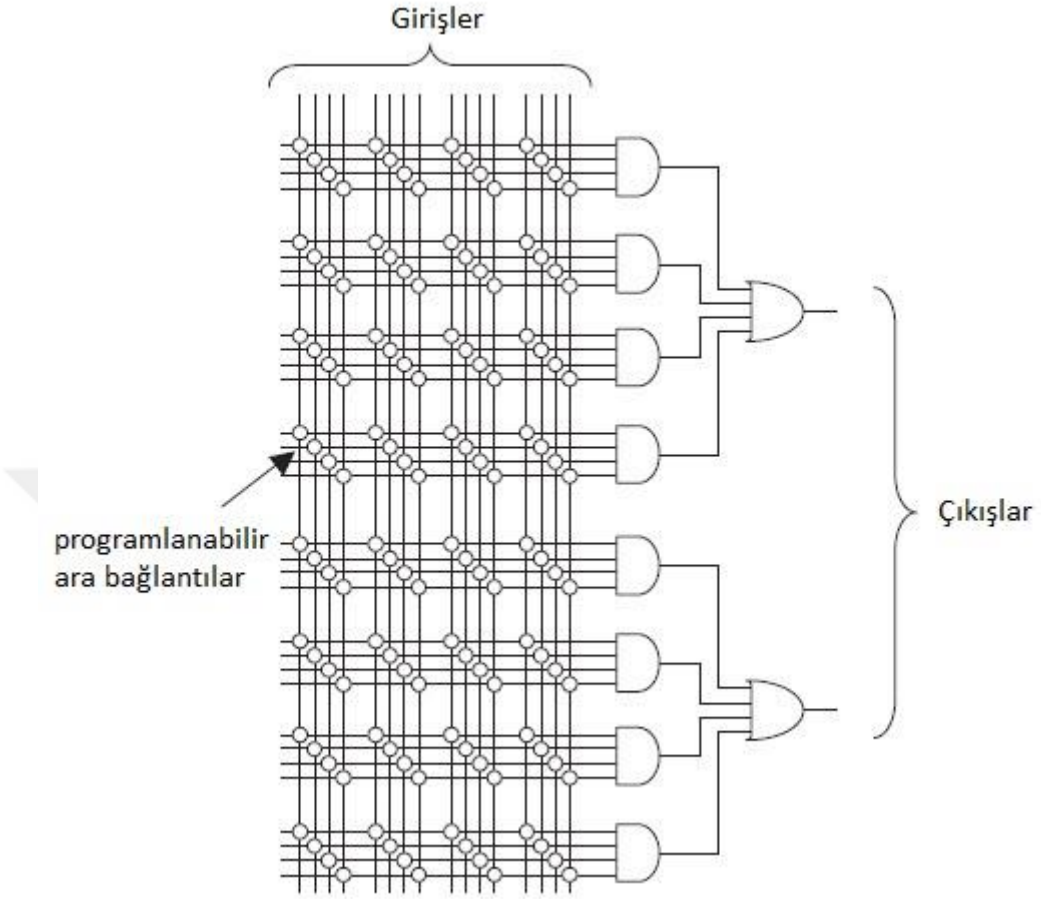
sayısal olarak yapılabilen tüm tasarımların tek bir entegre üzerinde gerçekleştirilebilmesine olanak sağlamalarıdır [28]. FPGA'ler, mimari, teknoloji, yerleşik özellikler ve maliyet açısından CPLD'lerden farklıdır. Bunlar esas olarak büyük boyutlu, yüksek performanslı devrelerin uygulanmasına yöneliktir. PLC'ler (basit veya karmaşık) uçucu değildir. Bir yonganın uçucu olması demek güç bağlantısı kesildiğinde programın kaybolması demektir. PLC'ler bir kerelik programlanabilirler, bu durumda sigortalar kullanılır veya EEPROM veya Flaş bellek ile yeniden programlanabilir (flaş, çoğu yeni cihazda tercih edilen teknolojidir). Öte yandan, FPGA'ler çoğunlukla uçucudur. Yani güç bağlantısının kesilmesi programın kaybolmasına neden olur. Dolayısıyla bağlantıları saklamak için SRAM'den yararlanırlar, bu durumda bağlantıları açılışta yüklemek için bir yapılandırma ROM gereklidir. Bununla birlikte, antifuse kullanımı gibi uçucu olmayan seçenekler vardır.

2.2.1. SPLD

Yukarıda belirtildiği gibi, PAL, PLA ve GAL cihazları topluca basit PLC'ler olarak adlandırılır.

PAL Cihazları

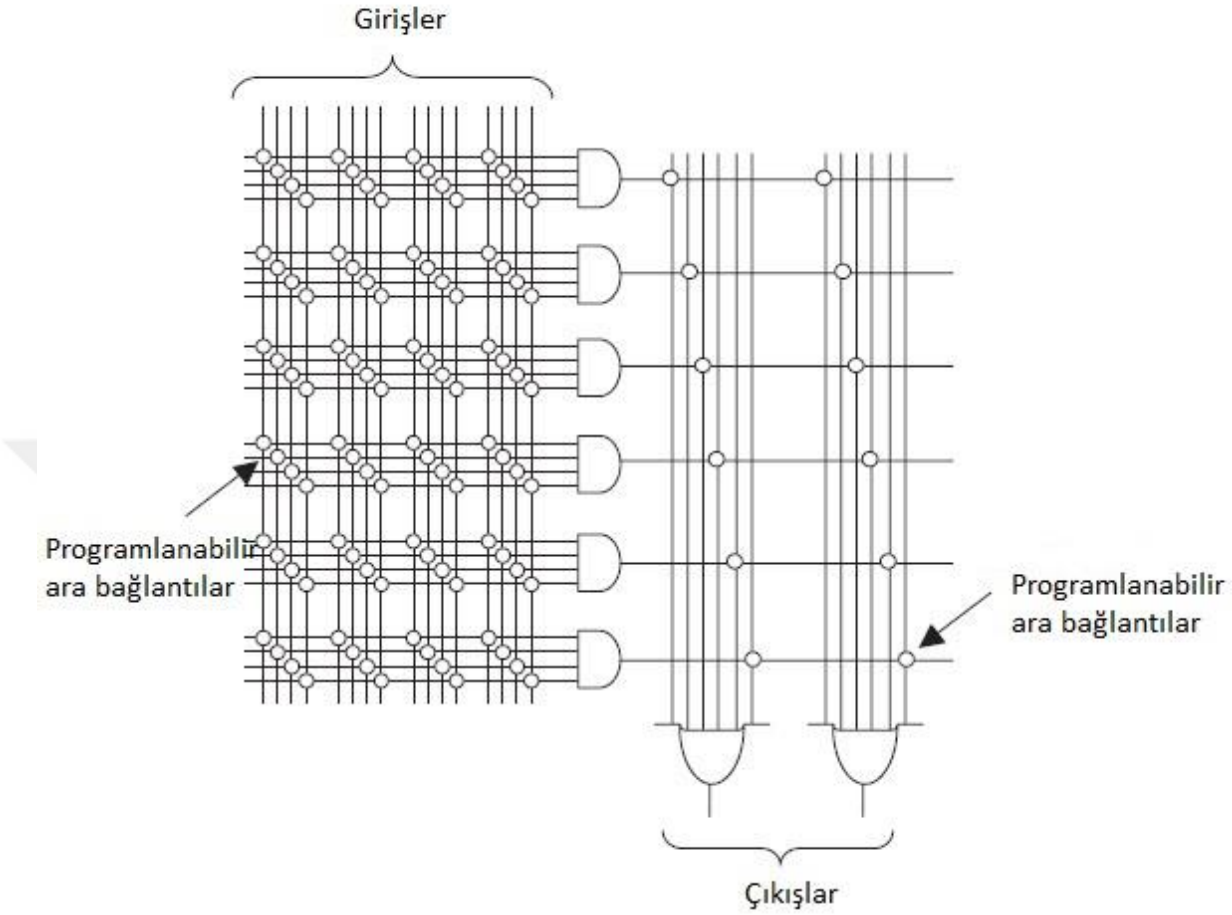
PAL (Programmable Array Logic) yongaları 1970'lerin ortalarında Monolithik Memories tarafından tanıtıldı. Temel mimarisi, küçük dairelerin programlanabilir bağlantıları temsil ettiği şekil aşağıda sembolik olarak gösterilmiştir. Görüldüğü gibi, devre, programlanabilir bir dizi AND kapılar ve ardından sabit bir dizi OR kapılardan oluşur. Aşağıdaki şeklin uygulanması, herhangi bir kombinasyonel fonksiyonun bir ürün toplamı (SOP) ile temsil edilebileceği gerçeğine dayanıyordu [27].



Şekil 2.8. PAL Mimarisi

PLA Cihazları

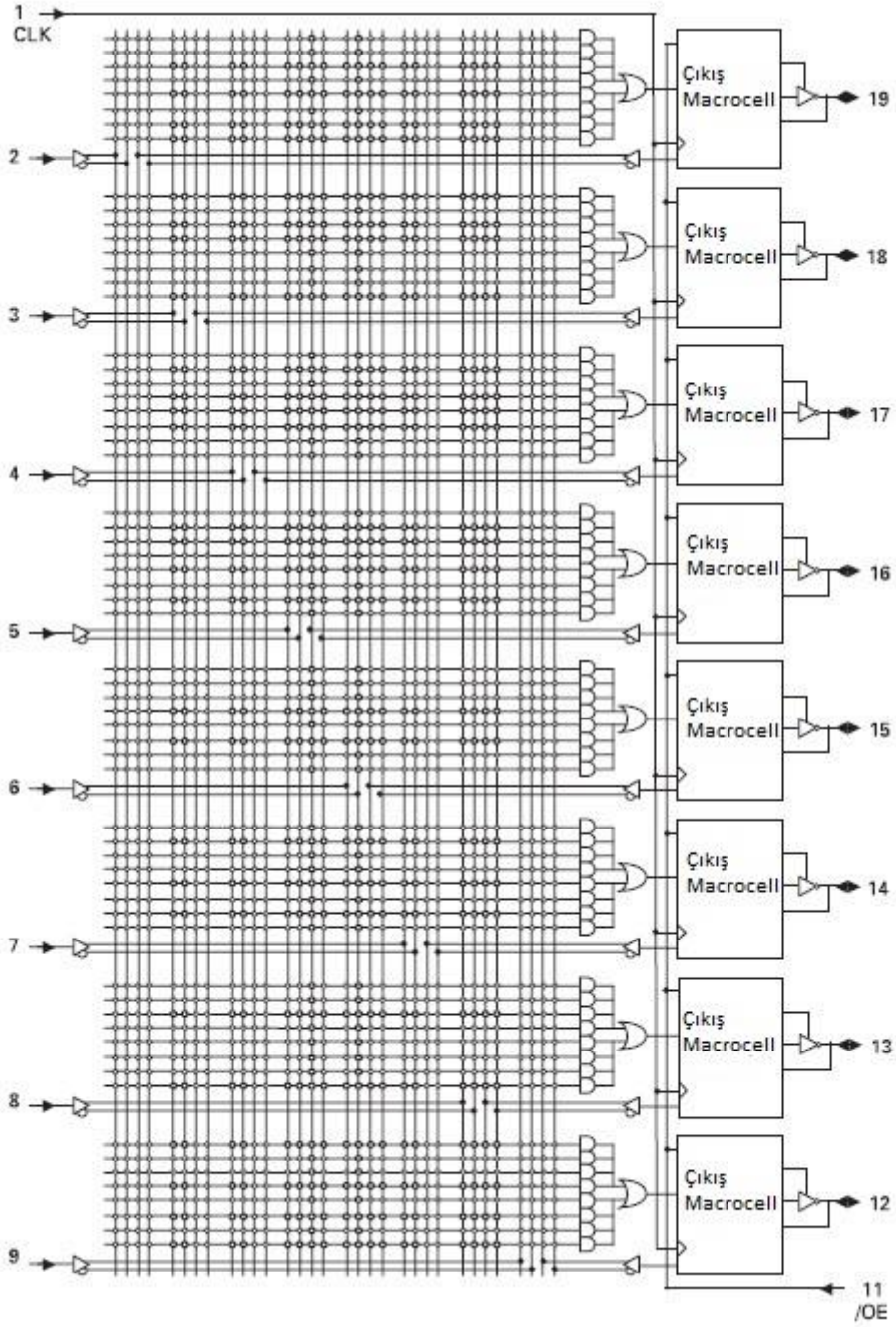
PLA (Programmable Logic Array) çipleri 1970'lerin ortalarında Signetics tarafından üretildi [27]. Bir PLA'nın basit mimarisi aşağıda sembolik olarak gösterilmiştir. Şekil 2.8 ile kıyaslandığında temel farklılık olarak PAL cihazında AND bağlantıları programlanırken PLA cihazında AND ve OR bağlantıları programlanabilmektedir. Bariz avantajı daha fazla esneklik olmasıdır. Bununla birlikte, iç düğümlerdeki daha yüksek zaman sabitleri devre hızını düşürmüştür. Şekil 2.9'da örnek bir PLA mimarisi verilmiştir.



Şekil 2.9. PLA Mimarisi

GAL Cihazları

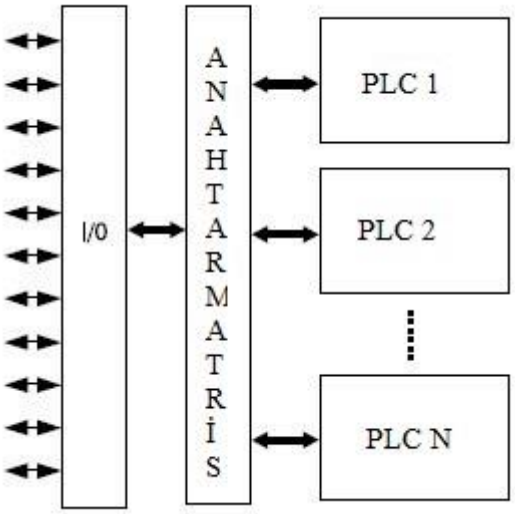
GAL (Jenerik PAL) mimarisi 1980'lerin başında Lattice tarafından tanıtıldı. İlk PAL cihazları üzerinde birkaç önemli gelişme içeriyordu: birincisi, flip-flop, birkaç kapı ve Çoklayıcı yanı sıra daha sofistike bir çıkış hücresi (Macrocell) inşa edildi; ikincisi, Macrocell'in kendisi programlanabilir, birkaç çalışma moduna izin verildi; üçüncüsü, Macrocell'in çıkışından programlanabilir diziye bir 'dönüş' sinyali de dahil edildi; dördüncüsü, PROM veya EPROM yerine EEPROM kullanıldı. Tanımlama için elektronik bir imza da dâhil edildi. Daha önce de belirtildiği gibi, GAL hala bağımsız bir pakette üretilen tek SPLD'dir. Ayrıca, aynı zamanda birçok CPLD'in yapımında temel yapı taşı olarak hizmet vermektedir [27]. Şekil 2.10'da temel bir GAL mimarisi verilmiştir.



Şekil 2.10. GAL Mimarisi

2.2.2. CPLD

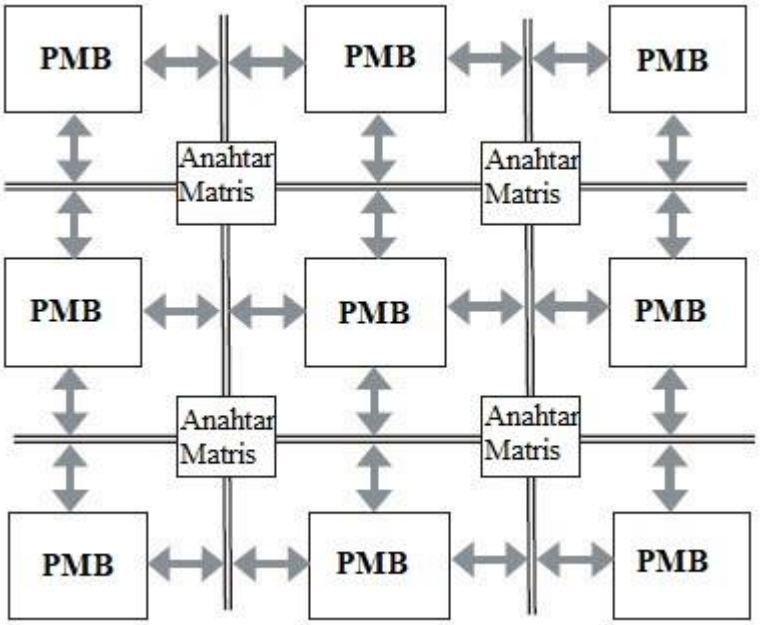
Bir CPLD yapımında temel yaklaşım aşağıdaki şekilde gösterilmiştir. Görüldüğü gibi, tek bir çip üzerinde üretilen birkaç PLC'den (genel olarak GAL tipi) oluşur ve bunları bir araya getirmek için kullanılan programlanabilir bir anahtar matrisi ve I/O pimlerine bağlanır. Ayrıca, CPLD'ler normalde diğer mantık standartlarına (1.8 V, 2.5 V, 5 V, vb.) arayüz ve JTAG desteği gibi birkaç ek özellik içerir.



Şekil 2.11. CPLD Mimarisi

2.2.3. FPGA

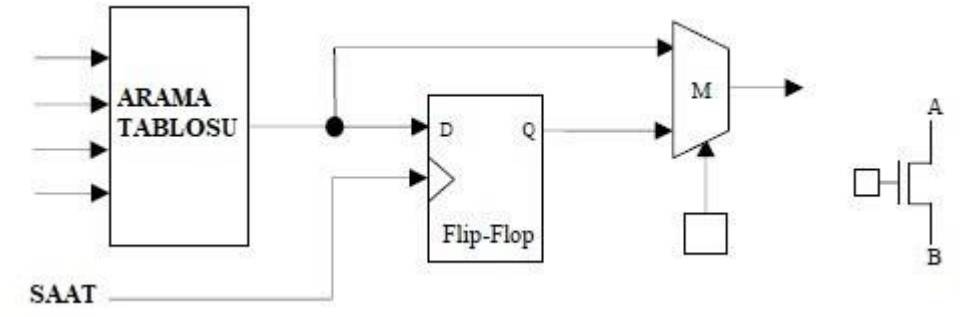
Alan Programlanabilir Kapı Dizisi (FPGA) cihazları 1980'lerin ortalarında Xilinx tarafından tanıtıldı. Mimaride CPLD'lerden, depolama teknolojisinden, yerleşik özelliklerin sayısından ve maliyetten yararlanılır ve yüksek performanslı, büyük boyutlu devrelerin uygulanmasını amaçlanır. Şekil 2.12'de mimari yapısı verilmiştir.



Şekil 2.12. FPGA Mimarisi

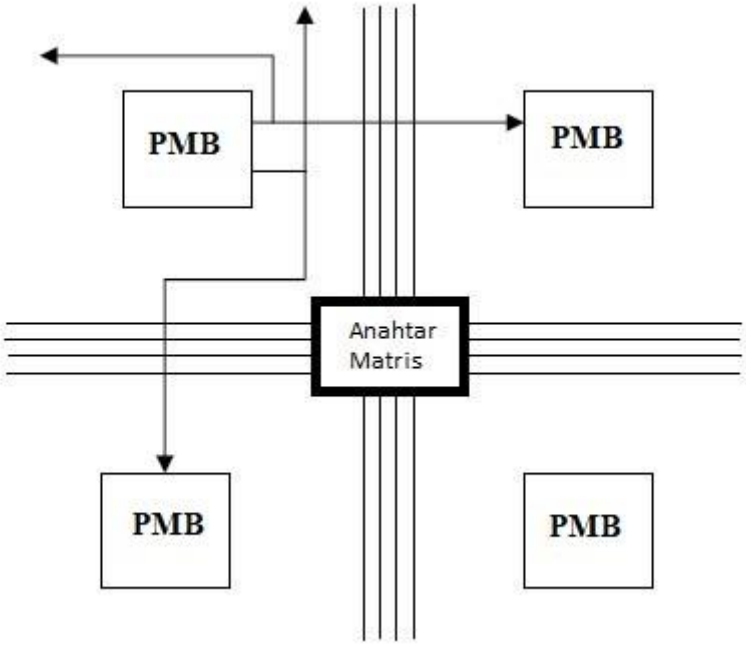
Şekil 2.12'deki mimaride FPGA, anahtar matris dizisi tarafından bağlanan programlanabilir mantıksal bloklar (PMB) dizisi tarafından oluşturulur. Ayrıca bu yapıyı çevreleyen giriş/çıkış blokları mevcuttur. PMB iç mimarisi PLC iç mimarisinden farklıdır. Öncelikle OR kapılarının takip ettiği AND kapılarıyla uygulanan SOP (Sum of Product) ifadeleri yerine işlemlerinde arama tablolarını (LUT (Lookup Table)) temel alır. Dahası, bir FPGA'da flip-flopların sayısı bir CPLD'den çok daha fazladır, böylece daha sofistike sıralı devrelerin inşasına izin verir. Çeşitli mantık seviyelerine JTAG desteği ve arayüzünün yanı sıra, diğer ek özellikleri de SRAM bellek, PCI arabirimi vb. gibi FPGA yongalarına dâhildir. Bazı yongalar ayrıca çarpanlar, DSP'ler ve mikroişlemciler gibi özel blokları içerir. Bir FPGA ve bir CPLD arasındaki bir diğer temel fark, ara bağlantıların saklanması hakkındadır. CPLD'ler uçucu olmasa da çoğu FPGA SRAM kullanır ve bu nedenle uçucudur. Bu yaklaşım yerden tasarruf sağlar ve çipin maliyetini düşürür, çünkü FPGA'lar çok sayıda programlanabilir bağlantı sunar, ancak harici bir ROM gerektirir. Bununla birlikte, yeniden programlama gerekli olmadığında avantajlı olabilecek uçucu olmayan FPGA'lar (antifuse ile) vardır.

Programlanabilir Mantıksal Bloklar (PMB); PMB'ler arama tabloları (LUT), Flip-Flop'lar ve Carry Logic'ten oluşmaktadırlar. Geleneksel bir FPGA onbinlerce flip-flop ve PMB içerebilmektedir PMB'nin hafıza kapasitesini içerisindeki LUT sayısı belirler. PMB'nin sahip olduğu esneklik ve simetri yapı uygulamaların kolaylıkla yerleştirilmesine olanak tanır. Şekil 2.13'de PMB'nin içyapısı verilmiştir.

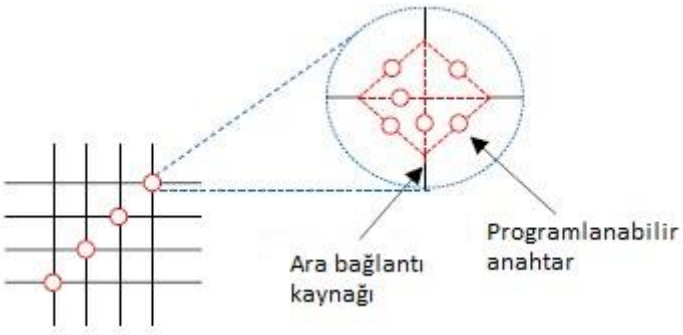


Şekil 2.13. PMB İç Yapısı

Ara Bağlantı Kanalları; Bu kanallar PMB'lerin kendi arasında ve PMB'ler ile giriş/çıkış blokları arasında bağlantıyı oluşturmada kullanılır. Programlanabilir olmaları esnek bir yapıya sahip olmalarını getirmiştir. Şekil 2.14'de ara bağlantı kanallarının gösterimi verilmiştir. Yatay ve dikey ara bağlantı kanallarının birleştiği yerlerde anahtar matrisler vardır. Şekil 2.15'de içyapısı verilmiştir. Bu anahtar matrislerin içyapısında altı transistörlü yönlendirme mekanizması bulunur. Programlanabilir bu anahtarlar sayesinde giriş yapılan taraftan kendisine komşu diğer üç tarafa yönlendirme sağlanabilir [12].

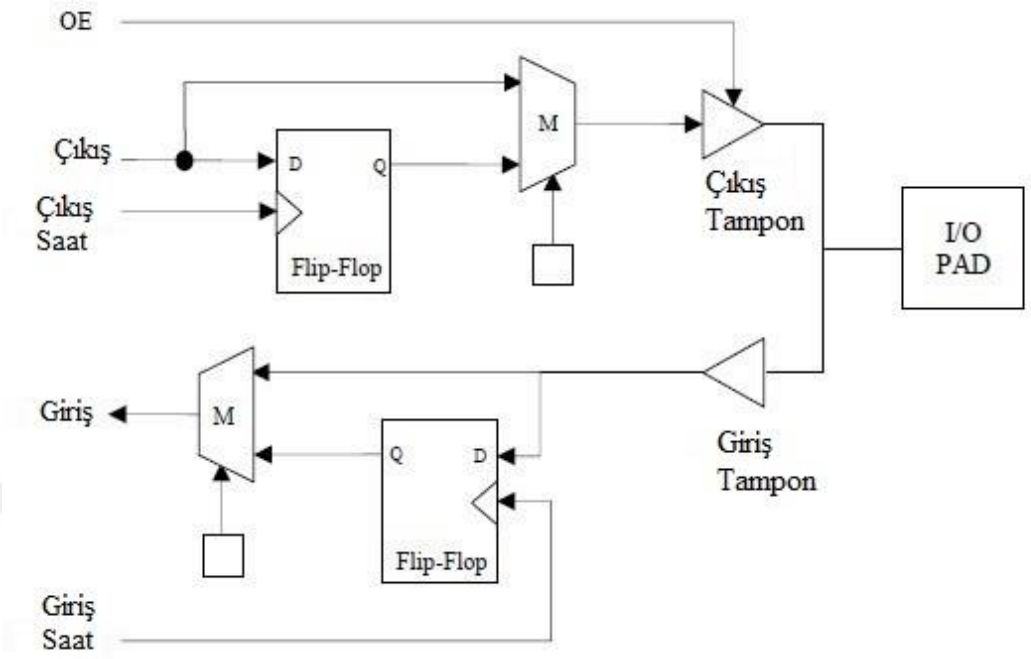


Şekil 2.14. Ara Bağlantı Kanalları



Şekil 2.15. Anahtar Matris

Giriş/Çıkış Blokları; Yonganın iç sinyal hatları ile pinleri arasında programlanabilir ara birim görevini giriş/çıkış blokları yaparlar. Giriş/çıkış blokları sayesinde FPGA pinleri çıkış, giriş ya da iki yönlü olarak programlanabilirler. Giriş/çıkış bloklarının içyapısı Şekil 2.16'da verilmiştir.



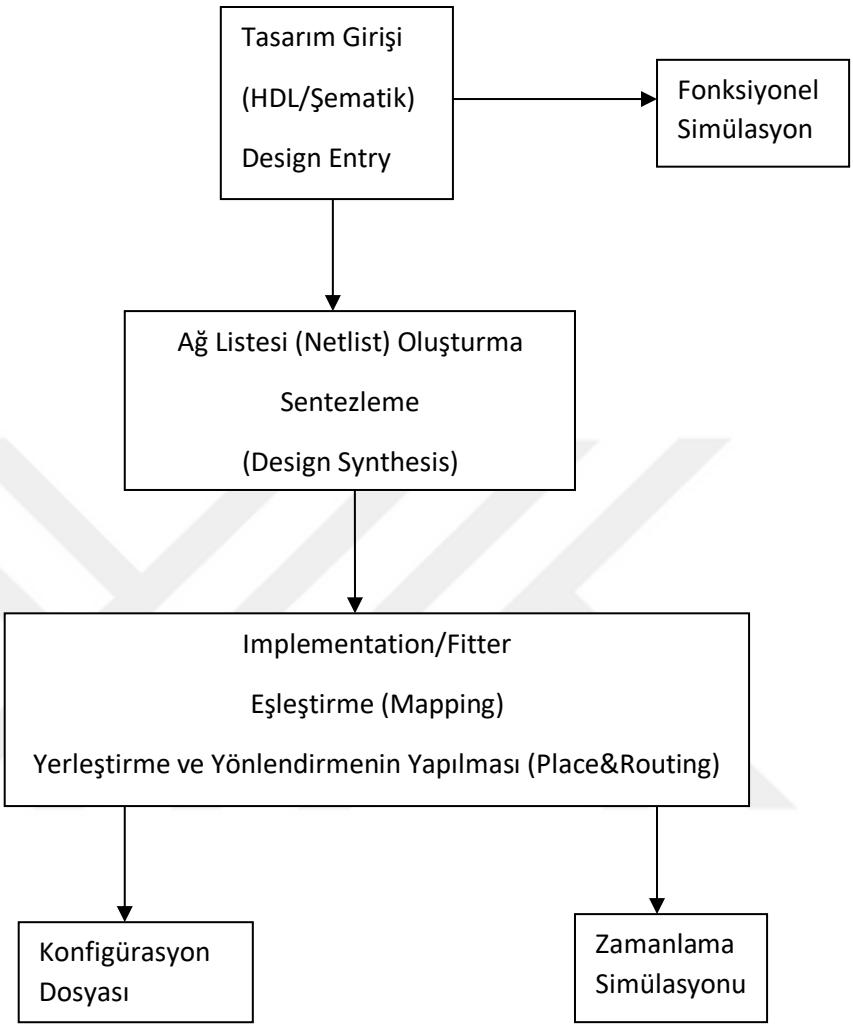
Şekil 2.16. Giriş/Çıkış Blokları İç Yapısı

2.2.4. FPGA Tasarım Akışı

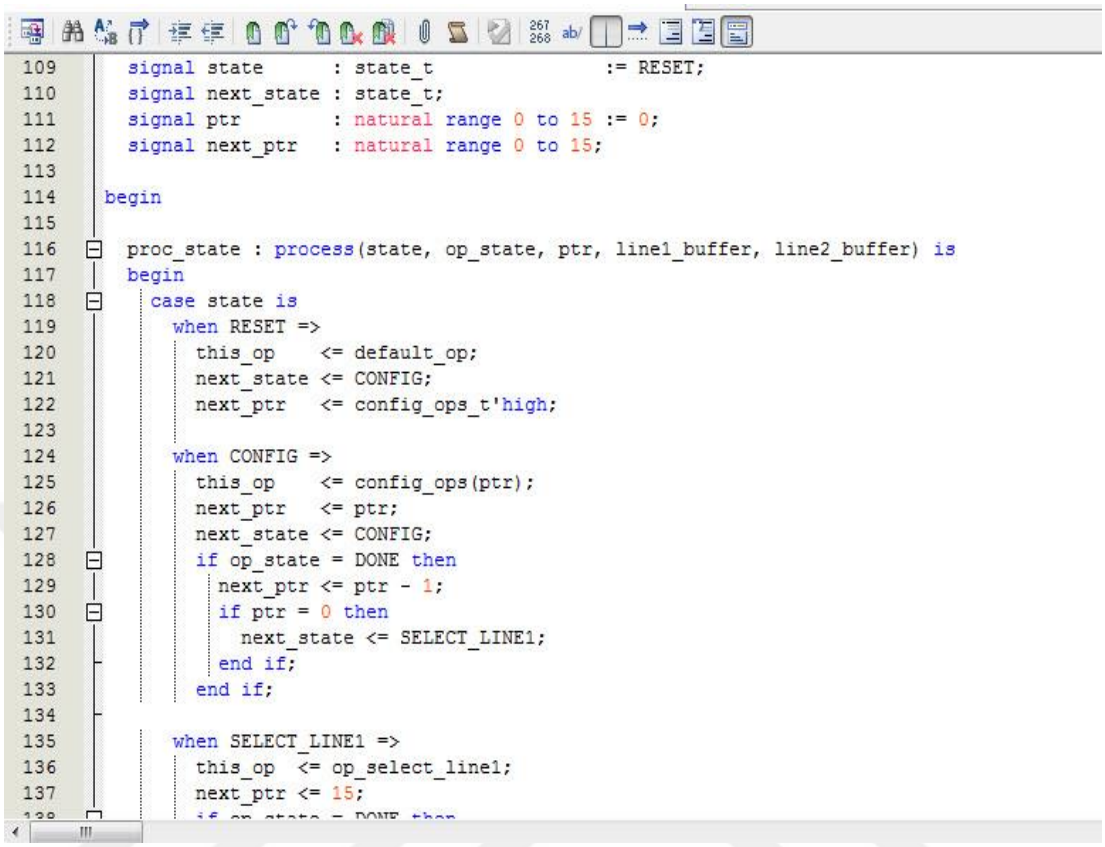
FPGA tasarım akışı sayısal tasarım aşamaları, geleneksel yazılım geliştirme adımlarına benzerdir. Tasarım, tasarım girişi ile başlayıp kodun üretilmesine kadar bir dizi adımı içerir. Şekil 2.17'de ana hatları verilen tasarım akışı sırasında birçok tasarım aracının kullanılması gerekliliği vardır.

2.2.4.1. Tasarım Girişi

Tasarım girişi aşamasında tasarımcı tasarım giriş yöntemini belirleyerek tasarladığı devreyi derleyici programa girer. Genellikle iki farklı tasarım giriş yöntemi kullanılır. Küçük tasarımlar için şematik tasarım daha çok tercih edilirken büyük tasarımlar için esneklik taşınabilirlik ve okunabilirlik yönünden daha avantajlı olan HDL (High Speed Integrated Circuits Hardware Description Language) donanım tanımlama dillerinden birisi kullanılır. Bu iki tasarım girişi yöntemi tasarımın durumuna göre bir arada da kullanılabilir. Eğer HDL tasarım girişi tercih edilmiş ise yaygın olarak kullanılan HDL tasarım dilleri VHDL (Very High Speed Integrated Circuits Hardware Description Language) ve Verilog'dur. Şekil 2.18'de VHDL dilinde yazılmış örnek bir kod parçası gösterilmektedir.



Şekil 2.17. FPGA Tasarım Akışı



```
109   signal state      : state_t              := RESET;
110   signal next_state : state_t;
111   signal ptr        : natural range 0 to 15 := 0;
112   signal next_ptr   : natural range 0 to 15;
113
114 begin
115
116   proc_state : process(state, op_state, ptr, line1_buffer, line2_buffer) is
117   begin
118     case state is
119       when RESET =>
120         this_op    <= default_op;
121         next_state <= CONFIG;
122         next_ptr   <= config_ops_t'high;
123
124       when CONFIG =>
125         this_op    <= config_ops(ptr);
126         next_ptr   <= ptr;
127         next_state <= CONFIG;
128         if op_state = DONE then
129           next_ptr <= ptr - 1;
130           if ptr = 0 then
131             next_state <= SELECT_LINE1;
132           end if;
133         end if;
134
135       when SELECT_LINE1 =>
136         this_op  <= op_select_line1;
137         next_ptr <= 15;
```

Şekil 2.18. Örnek VHDL Kod Parçası

2.2.4.2. Ağ Listesi (Netlist) Oluşturma

Tasarım akışının bu adımında tasarımcı tasarlanan devrenin metinsel şekilde tanımlanmış hali olan ağ listesini oluşturur. Bu adım tasarım girişi yönteminden bağımsızdır. Ağ listesi tasarımın depolama elemanları ve mantık kapıları kullanılarak temsil edildiği bir listedir. Eğer tasarım giriş yöntemi olarak HDL yöntemi kullanılmış ise ağ listesinin oluşturulabilmesi için tasarımın önce sentezlenmesi gerekmektedir.

Sentezleme: Sentezleme işlemi tasarım girişi adımının çıktısını devre bağlantılarına dönüştüren adımdır. Sentezleme işlemi sırasında bir sentez aracı ile uygulamaya ait mantık kapısı seviyesinde bağlantı listesi (Gate Level Netlist) oluşturulur. Bu aşamada eğer devreye ait kısıtlamalar (saat frekansı, yerleştirme, kritik yollar, zamanlama vb.) varsa dikkate alınmalıdır.

2.2.4.3. Implementation/Fitter (Tasarım Gerçekleme)

Bu adımda ağ listelerinin donanım düzenine çevrildiği Implementation/Fitter süreci işletilir. Bu adım için Xilinx ISE geliştirme programında ‘Implementation’ terimi,

Quartus II geliştirme programında ‘Fitter’ terimi kullanılır. Geliştirme programları aşağıda detaylı olarak incelenecektir. Tasarım bu aşamada bazı işlemlerden geçerek FPGA kartına yüklenecek dosya formatı halini alır.

Eşleştirme (Mapping): Eşleştirme sürecinde ağ listesine sahip tasarımın FPGA içerisinde bulunan kaynaklar ile nasıl gerçekleneceği belirlenir. Ağ listesinde bulunan bölümlerin FPGA yongasındaki hangi bölümler kullanılarak uygulanacağı bu aşamada ortaya çıkar. Bu işlem karmaşık bir süreçtir. Ancak günümüzde geliştirilen algoritmalar yardımıyla işlem kolaylaştırılmıştır.

Yerleştirme ve Yönlendirmenin Yapılması (Place&Routing): Bu aşamada derleyici program, tasarımda bulunan kısıtlamaları da göz önünde bulundurarak tasarımın hedef teknoloji içerisinde hangi konumdaki kaynaklara yerleştirileceği ve bunlar arasındaki bağlantıların nasıl yapılacağını belirler. Yerleştirme işleminden sonra derleyici program yönlendirme işlemini gerçekleştirir. Yönlendirme işlemi esnasında FPGA yongası içerisindeki hücrelerin kendi arasındaki ve Giriş/Çıkış blokları arasındaki bağlantılar belirlenir. Bu aşamada kaynaklar optimize edilir. Yani uygulamanın FPGA yongası içerisindeki CLB’lere en dar alanda yerleştirilmesi ve bu yerleşim sonucunda toplam bağlantı yolunun olabilecek en kısa ölçüde tutulması hedeflenir.

Tasarım gerçekleme aşamasının sonucunda ayrıca uygulama içindeki kritik yol ve bu yol ile bağlantılı olarak uygulamanın çalıştırılabileceği en yüksek saat frekansı ortaya çıkar.

2.2.4.4. Fonksiyonel/Zamanlama Simülasyonu

Tasarımın herhangi bir simülasyon ortamında (Kullanılacak FPGA yongasının üretici firmasının geliştirdiği yazılım araçları olabildiği gibi diğer firmalar tarafından geliştirilmiş programlar da olabilir.) kullanılacak FPGA yongasına yüklenmeden önce hatalarının ayıklanıp doğruluğunun test edilmesi sürecine simülasyon denilmektedir. Tasarım girişi adımından sonra yapılabildiği gibi yerleştirme ve yönlendirme adımından sonra da yapılabilir. Tasarımın fonksiyonunu test etmek için tasarım girişinden sonra yapılırsa fonksiyonel simülasyon, zamanlama bilgileri ile tasarımı yerleştirme ve yönlendirme adımından sonra test etmek için yapılıyorsa zamanlama simülasyonu diye isimlendirilir.

2.2.4.5. Konfigürasyon Dosyasının Elde Edilmesi

Tasarımın yerleştirme ve yönlendirme aşamasını geçip tüm fonksiyonları ile test edilip doğrulandıktan sonra FPGA yongasına yüklenecek formatta bir yapılandırma dosyası elde edilmesi işlemidir. Bahsedilen format ve dosya uzantıları FPGA yonga üreticilerine göre farklılık gösterir. FPGA yongasına yüklenecek yapılandırma dosyası oluşturulduktan sonra JTAG (Joint Test Action Group (Ortak Test Eylem Grubu)) arayüzü ile bu dosya FPGA kartına yüklenebilir. SRAM tabanlı FPGA yongaları uçucu özelliktedirler. Dolayısıyla güç bağlantısının kesilmesi FPGA yongasında programın kaybolmasına neden olur. Bu tip FPGA yongaları konfigürasyon dosyasını her açılışta uçucu olmayan bir hafızadan okuyarak konfigüre olurlar. Uçucu olmayan hafıza ünitesine sahip FPGA yongaları konfigürasyon dosyalarını hafızalarında tutabilirler.

2.3. DONANIM TANIMLAMA DİLİ VHDL

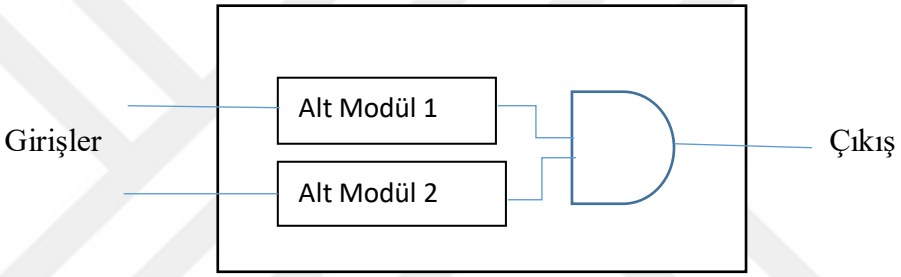
Donanım tanımlama dilleri bir elektronik sistemin tasarlanmasında test edilmesinde ve sentezlenmesinde kullanılır [29]. En yaygın kullanılan dillerden birisi VHDL'dir. VHDL, Çok Yüksek Hızlı Entegre Devre Donanımı Tanımlama Dili anlamına gelen " Very High Speed Integrated Circuit Hardware Description Language" ifadesinin kısaltması olup bir donanım parçasını modellemek için kullanılır [30]. VHDL dili kullanılarak yapılan tasarımlarda senkron ve asenkron ardışıl devreler kontrol yapılarıyla kolayca yapılabilir. Tasarımlar daha küçük bileşenlerine ayrılarak hiyerarşik bir düzen sağlanabilir. Tasarımların fonksiyonel ve zamansal simülasyonları donanıma uygulamadan yapılabilir. Bizim gibi birçok tasarımcının donanım tanımlama dili olarak VHDL kullanmasına sebep olan bazı özellikleri aşağıda verilmiştir.

- Tasarımı oluşturan sistemlerin farklı seviyelerde yapısal veya davranışsal olarak belirlenebilmesi
- VHDL'in birçok EDA (Elektronic Design Automation) araç üreticisi tarafından desteklenmesi
- Simülasyon araçlarının uygun fiyatlarda olması
- Sadece elektronik modelleme dili olmayıp kimyasal, elektromanyetik, kimyasal sistemlerin modellenmesinde de kullanılması.

VHDL dili kullanılarak yapılan tasarımlarda iki tür modelleme tekniği kullanılabilir. Bunlardan birincisi yapısal modelleme diğeri davranışsal modellemedir.

2.3.1. VHDL Yapısal Modelleme Tekniği

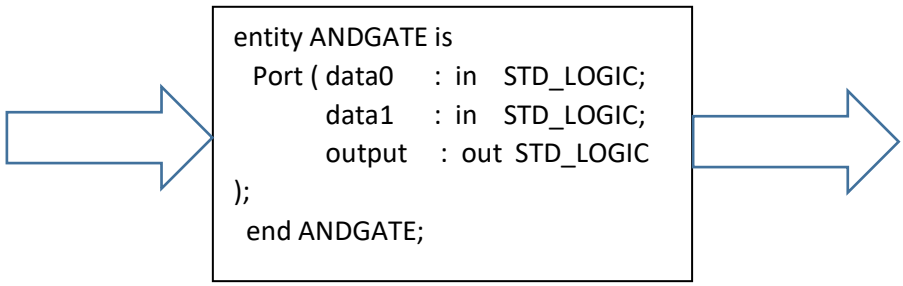
VHDL yapısal modelleme tekniğinde tasarım bir bütün olarak değil her biri özel bir işlevi yerine getiren küçük alt birilerden oluşan bir yapı olarak tasarlanır. Çok karmaşık ve büyük tasarımlarda tasarımcılara büyük kolaylıklar sağlayan bu modelleme tekniği modelin yapısının tasarımcı tarafından yapılandırılması ilkesine dayanır. VHDL tasarımlarında yaygın olarak kullanılan yapısal modelleme tekniği birbirinden bağımsız alt modüllerin davranışsal modelleme ile oluşturulup bunların üst seviyede yapısal modelleme ile birbirlerine bağlanması şeklinde yapılmaktadır.



Şekil 2.19. VHDL Yapısal Modelleme Tekniği Gösterimi

2.3.2. VHDL Davranışsal Modelleme Tekniği

VHDL davranışsal modelleme tekniğinde modelin çıkış ve giriş tepkileri davranışsal olarak tanımlanır. Devrenin yalnızca işlevi ile ilgilenilir, içyapısı ile ilgilenilmez.



Şekil 2.20. VHDL Davranışsal Modelleme Tekniği Gösterimi

2.3.3. VHDL Nesne Sınıfları

VHDL dilinde verilerin işlenebilmesi için belirli kalıplar halinde saklanması gerekmektedir [29]. Program içerisinde belirtilen veri türündeki değerleri tutan yapılara veri nesneleri denir [30]. Veri nesnelerinin programa dâhil edilebilmesi program içerisinde deklarasyonunun yapılarak isim, sınıf ve türünün belirtilmesi ile olur. VHDL bir donanım tanımlama dili olarak diğer programlama dillerine kıyasla desteklediği veri türleri bakımından daha zengindir. VHDL dilinde nesnelerin deklerasyonu tasarım dosyasının farklı bölgelerinde yapılabilir. Bir paket içinde yapılan tanımlamalar o paketi kullanan bütün tasarımlar tarafından erişilebilir [29]. Varlık (entity) içinde tanımlanan nesneler o varlığa ait mimariler (architecture) içinde kullanılabilir. Mimari içinde yapılan nesne tanımlamaları sadece o mimaride kullanılabilir.

VHDL nesneleri aşağıdaki gibi kullanılır.

Nesne_sınıfı nesne adı:nesne_türü [:=ilk_değer];

VHDL donanım tanımlama dilinde kullanılan dört farklı nesne sınıfı vardır.

- Değişkenler (Variables)
- Sabitler (Constants)
- Sinyaller (Signals)
- Dosyalar (Files)

2.3.3.1. Değişkenler (Variables)

Değişkenler alt programlar (prosedür, fonksiyon) ve işlev (process) içerisinde tanımlanan, tanımlandığı alt program ve işlevde kullanılabilen, değeri değiştirilebilir, tanımlandığı yerde geçici değerleri tutmak için kullanılan nesnelerdir.

VHDL değişken nesnesi aşağıdaki gibi kullanılır.

variable değişken adı:veri_türü [:=ilk_değer];

İşlev ve alt programlar dışına değer aktarabilmek için veri türleri aynı olmak kaydı ile değişkenden sinyale sinyalden değişkene atama gerçekleştirilebilir. Çünkü daha öncede belirttiğimiz üzere değişkenler sadece tanımlandığı alt program ve işlevde kullanılabilir.

2.3.3.2. Sabitler (Constants)

Sabitler bir kez atama yapıldıktan sonra değeri değiştirilemeyen nesnelerdir. Bu nesne sınıfı programın okunabilirliğini artırmak için kullanılabilir. Eğer tasarımı yapılan modelde değişmez değerlere ihtiyaç varsa sabitler kullanılır. Tasarımcı isterse sabitlerin değerini elle değiştirip programı tekrar derleme ve optimum değeri bulma olanağına kavuşur [8].

VHDL değişken nesnesi aşağıdaki gibi kullanılır.

constant sabit adı:veri_türü := sabitin_değeri;

2.3.3.3. Sinyaller (Signals)

Sinyaller bir donanım sisteminde veri akışını açıklayan nesne sınıfıdır. Mimari içinde eş zamanlı yapılar arasındaki ara bağlantıları temsil ederler. Sinyal nesne sınıfı blok içerisinde mimarinin tanımlama kısmında tanımlanır. Mimari içinde tanımlanan sinyaller mimarinin tamamında kullanılabilecek şekilde globaldirler. Eğer process içerisinde kullanılırlarsa bellek görevi yaparlar.

VHDL sinyal nesnesi aşağıdaki gibi kullanılır.

signal sinyal adı:veri_türü := sinyalin_değeri;

2.3.3.4. Dosyalar (Files)

VHDL dilinde bildirimi ile tasarımda dosyanın veri yazılabilir veya okunabilir hale getirilmesini sağlayan nesne sınıfıdır. Bilgisayarda fiziksel olarak kayıtlı dosya ile VHDL tasarımı arasında iletişim yolu oluşturur.

Tasarımcılar için veri girişi yapmak maksadıyla önceden hazırlanmış bir dosyanın kullanılması ve devrenin ürettiği çıktı bilgilerinin başka bir dosyaya yazılması faydalı ve sıkça kullanılan bir yöntemdir.

2.3.4. VHDL DİLİNİN YAPISAL ELEMANLARI

VHDL dili yapısal olarak aşağıdaki elemanlardan oluşur.

- Entity (Varlık)
- Architecture (Mimari)
- Configuration (Konfigürasyon)
- Package (Paket)
- Library (Kütüphane)

- Process (İşlem)
- Function (Fonksiyon)
- Procedure (Prosedür)

Temel bir VHDL tasarımı Entity, Architecture ve Library olmak üzere bu 3 yapısal elemanı barındırmak zorundadır.

2.3.4.1. Entity (Varlık) Elemanı

Tasarımın dış dünya ile olan ilişkisini tanımlamak için kullanılan elemandır. Burada devrenin giriş ve çıkış portlarının tanımlaması yapılır. Her bir tasarımda sadece bir entity elemanı olmalıdır. Fakat aynı entity için birden fazla configuration ve architecture kullanılabilir. Port tanımlaması dört farklı şekilde yapılabilir.

- IN: Devreye dışarıdan gelen sinyalleri tanımlayıp giriş portu olarak görev yapar. Bu veriler sadece okunurlar
- OUT: Devreden dışarıya gönderilen sinyalleri tanımlayıp çıkış portu olarak görev yapar.
- INOUT: Bu şekilde tanımlanan port hem giriş hem de çıkış olarak kullanılabilir.
- BUFFER: Çıkış portuna benzer fakat istenildiğinde değeri okunabilir.

Entity elemanı aşağıdaki şekilde kullanılır.

ENTITY entity_adı **IS**
PORT (port_adı:sinyal_modu sinyal_türü;
port_adı:sinyal_modu sinyal_türü;
….);
END entity_adı;

2.3.4.2. Architecture (Mimari) Elemanı

Bir tasarımın nasıl davranması gerektiğini tanımlayan ve içyapısını gösteren elemandır. Architecture elemanında entity elemanında tanımlanmış portlar arasındaki ilişki gösterilir. Bir tasarımda aynı entity için birden fazla architecture yazılıp bunlardan bir tanesi seçilebilir. Architecture elemanı içinde tanımlanan bütün ifadeler ve işlemler eş zamanlı olarak gerçekleştirilir.

Architecture elemanı aşağıdaki şekilde kullanılır.

ARCHITECTURE architecture_adı **OF** entity_adı **IS**

Sabit tanımlamaları;

Sinyal tanımlamaları;

Alt program tanımlamaları;

Bileşen tanımlamaları;

BEGIN

Sinyal atamaları;

İşlem ifadeleri;

END architecture_adı;

2.3.4.3. Configuration (Konfigürasyon) Elemanı

Configuration (Konfigürasyon) Elemanı bir varlık için birden fazla mimari oluşturulduğunda ve bu mimarilerin çağırılması gerektiğinde kullanılır.

Configuration elemanı aşağıdaki şekilde kullanılır.

CONFIGURATION configuration_adı **OF** entity_adı **IS**

FOR architecture_adı

FOR çağırma_etiketi:bileşen_adı

USE entity library_adı.entity_adı(architecture_adı);

END FOR;

END FOR;

END configuration_adı;

2.3.4.4. Package (Paket)Elemanı

Package elemanı tasarımcıların ortak kullanacağı sabitleri, sinyalleri, veri türlerini, bileşenleri, fonksiyonları ve prosedürleri tanımlamak için oluşturdukları yapılardır. Bu yapılar tasarımcıyı aynı ifadeleri tekrar tekrar yazmaktan kurtarır.

Paketler gövde ve paket bildirimi olmak üzere iki kısımdan oluşur.

Paket bildirimi aşağıdaki şekilde yapılır.

PACKAGE paket_adı **IS**

Veri türü tanımlamaları;

Fonksiyon, Prosedür bildirimleri;

END PACKAGE paket_adı;

Paket gövdesi aşağıdaki şekilde tanımlanır.

PACKAGE BODY paket_adı **IS**

Fonksiyon tanımlamaları;

Prosedür tanımlamaları;

END PACKAGE BODY paket_adı;

2.3.4.5. Library (Kütüphane) Elemanı

Library (Kütüphane) elemanı mimari, varlık, konfigürasyon ve paket yapılarından oluşan derlenmiş modellerdir. VHDL dilinde bu yapılar genelde üç kütüphanede tutulur. Bunlar ieee, std ve work kütüphaneleridir. İeee ve std kütüphaneleri VHDL diline ait olan kütüphanelerdir ve bir kısmı VHDL yüklenmesi ile beraber gelirken bir kısmı ise internetten indirilebilmektedir. İeee kütüphanesinde matematiksel işlemler, nümerik işlemler, text işlemleri paketleri yer alırken std kütüphanesinde dosya okuma fonksiyonları ve standart değişkenlerin bulunduğu paketler yer alır. work kütüphanesi ise kullanıcı tarafından yazılmış olan paketleri içerir. Diğer kütüphaneler ise kullanılan devre tipini veya özel işlemleri içeren paketlerdir [14].

Library elemanı aşağıdaki şekilde kullanılır.

LIBRARY kütüphane_adı;

USE paket_adı;

2.3.4.6. Process (İşlem) Elemanı

Process (İşlem) elemanı sıralı işlem yapmayı gerektiren durumlarda kullanılan ve içerisinde bulunan ifadelerin eş zamanlı olarak işlendiği yapılardır. İşlem elemanı bildirimi mimari içerisinde yapılır. Bir mimari birden fazla işlem içerebilir.

Process elemanı aşağıdaki şekilde kullanılır.

Process_adı**:PROCESS** (duyarlılık listesi)

Nesne_sınıfı_bildirimleri;

BEGIN

Ardışık ifadeler;

END PROCESS process_adı;

2.3.4.7. Function (Fonksiyon)) Elemanı

VHDL dilinde tek bir değer döndüren program parçacıklarına Function (Fonksiyon) elemanı denir. Fonksiyonun sadece giriş parametresi vardır. Çıkış parametresi yoktur.

Giriş parametresiyle alınan girdiler işlenerek elde edilen sonuç return ifadesi ile ana programa döndürülür.

Fonksiyonlar gövde ve paket bildirimi olmak üzere iki kısımdan oluşur.

Fonksiyon bildirimi aşağıdaki şekilde yapılır.

FUNCTION fonksiyon_adı (parametre listesi[girişler:tür]) **RETURN** tür;

Fonksiyon gövdesi aşağıdaki şekilde tanımlanır.

FUNCTION fonksiyon_adı (parametre listesi) **RETURN** tür **IS**

Sabit ve değişken nesne bildirimleri;

BEGIN

Fonksiyon gövde kısmı;

END fonksiyon_adı;

2.3.4.8. Procedure (Prosedür) Elemanı

Procedure (Prosedür) elemanı hem giriş hem de çıkış parametresi bulunduran birden fazla çıkış değeri döndürebilen program parçacıklarıdır.

Prosedürler gövde ve paket bildirimi olmak üzere iki kısımdan oluşur.

Fonksiyon bildirimi aşağıdaki şekilde yapılır.

PROCEDURE prosedür_adı (parametre listesi);

Fonksiyon gövdesi aşağıdaki şekilde tanımlanır.

PROCEDURE prosedür_adı (nesne sınıfı girişler:**IN** tür

nesne sınıfı çıkışlar:**OUT** tür) **IS**

Sabit ve değişken nesne bildirimleri;

BEGIN

Prosedür gövde kısmı;

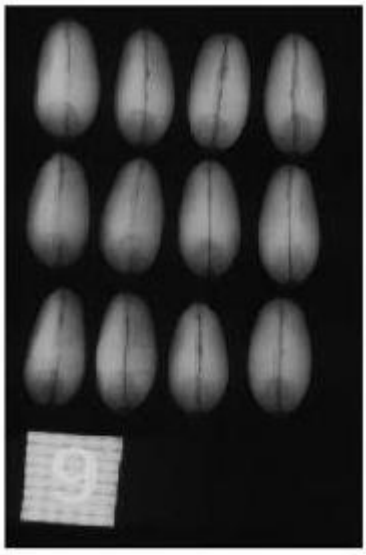
END prosedür_adı;

3. BUĞDAY SINIFLANDIRMA PROBLEMİNİN ÇÖZÜMÜ

Bu bölümde tezin de amacı olan buğday sınıflandırma problemi hem seri hem de paralel işlem yapan platformlar kullanılarak çözülecek, sonuçlar ilgili bölümde tartışılacaktır. Buğday sınıflandırma probleminin çözümünde " UCI Machine Learning Repository" veri tabanından alınan bir buğday veri seti kullanılacak ve bu veri setindeki parametreler yapay sinir ağının giriş ve çıkış değerleri olacaktır.

Buğday Sınıflandırılması Veri Seti:

EK 1 Çizelge A.1'de verilen bu veri seti; X-ışını tekniği metodu ile çekilen görüntüler üzerinden çıkarılan öznitelikler ile buğday türü sınıflandırılmasına aittir [31].



Şekil 3.1. X-ışını tekniği metodu ile çekilen buğday görüntüleri [31]

Veri setinde öznitelikleri içeren 7 giriş ve 3 buğday türü çıkışı bulunmaktadır. Bu girişler sırası ile alan (x1), çevre (x2), yoğunluk (x3), çekirdek uzunluğu (x4), çekirdek genişliği (x5), asimetri katsayısı (x6) ve çekirdek iz uzunluğu (x7) girişleri bulunmaktadır. Çıkış olarak Kama (y1), Rosa (y2) ve Kanada (y3) buğday türleri verilmiştir.

3.1. Buğday Sınıflandırma Probleminin Seri İşlem Yapan Platformda Çözülmesi

Yukarıda ön bilgileri verilmiş veri setini sınıflandırmak için, yapay sinir ağını hatayı geriye yayma (Back propagation) algoritması ile eğiten ve sınıflandıran bir program delphi 7.0 nesneye dayalı programlama dilinde yazılmış ve bu program seri işlem yapan 32 bit işletim sistemi Windows 7 service pack 1 kurulu olan intel core 2 duo cpu 2.2 GHz işlemcisi olan 2GB belleği olan bir bilgisayarda uygulanmıştır. Sınıflandırma probleminin çözümünde giriş nöron sayısı 8, gizli katman sayısı 1, gizli katman nöron sayısı 4, çıkış nöron sayısı 3 olan bir yapay sinir ağı oluşturulmuştur. Yapay sinir ağının hata oranı E ≤ 0,0001 olmuştur. Başlangıç ağırlık değerleri -1 ile +1 arasında alınmıştır.

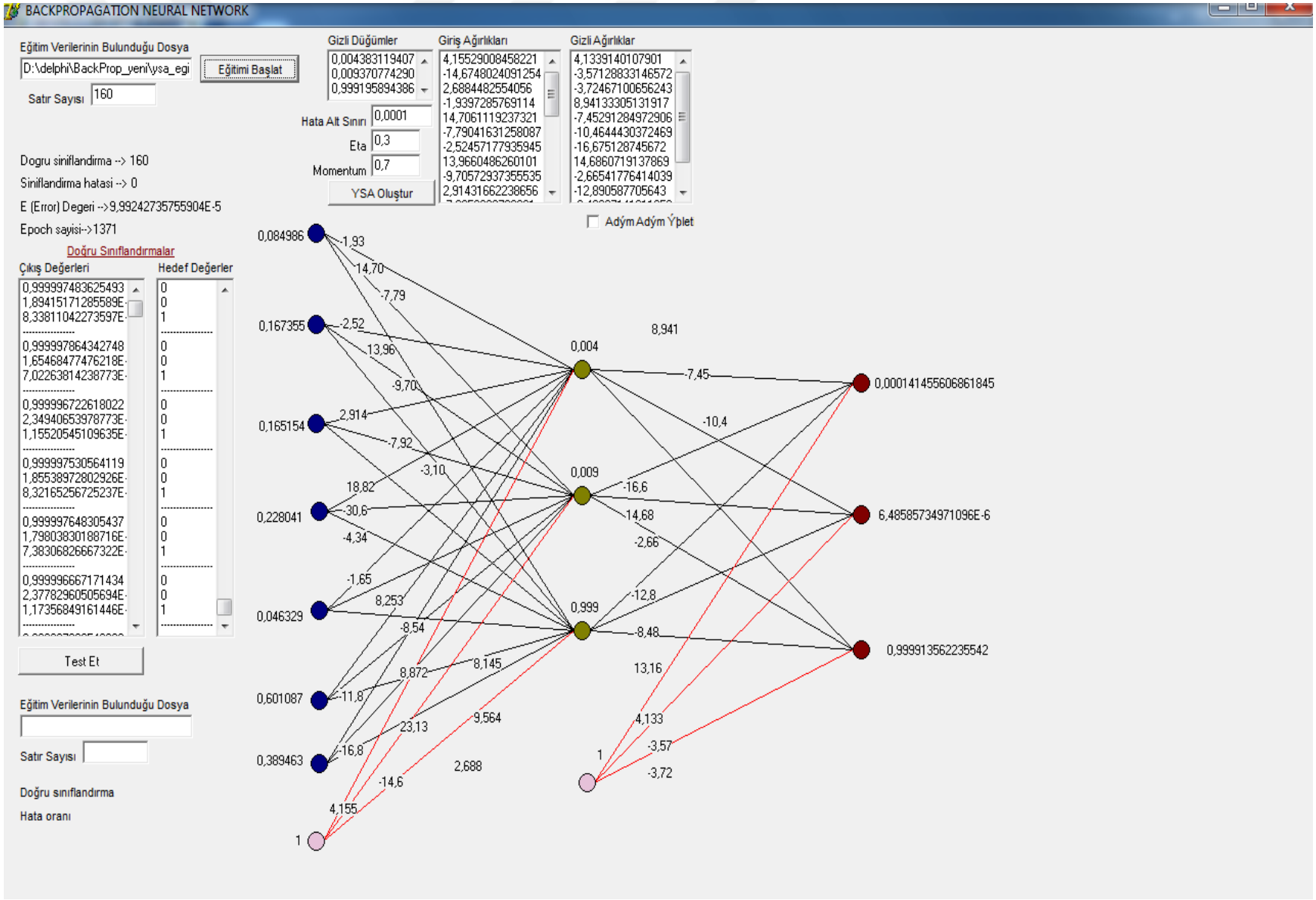
Programın çalışması esnasında öncelikle ağın eğitilmesi gerçekleştirilmiştir. Yapılan eğitim sonucu elde edilen ağırlık değerleri bir dosyada tutulmaktadır. Veri setinin tamamı eğitim için kullanılmamıştır. Bunlardan yaklaşık %25'i test aşaması için kullanılmıştır. Test için ağın giriş değerleri dışarıdan girilebilecek şekildedir. Eğitim ve test verilerinin sınıflandırmadaki başarısı ayrıca programda belirtilmiştir. Test safhasında yapay sinir ağı eğitiminin başarısını ölçmekte kullanılan test hata oranı Denklem 3.1'deki gibi hesaplanır.

$$\text{Test Hata Oranı} = \frac{\text{Yanlış Sınıflandırılan Örnek Sayısı}}{\text{Toplam Örnek Sayısı}} \tag{3.1}$$

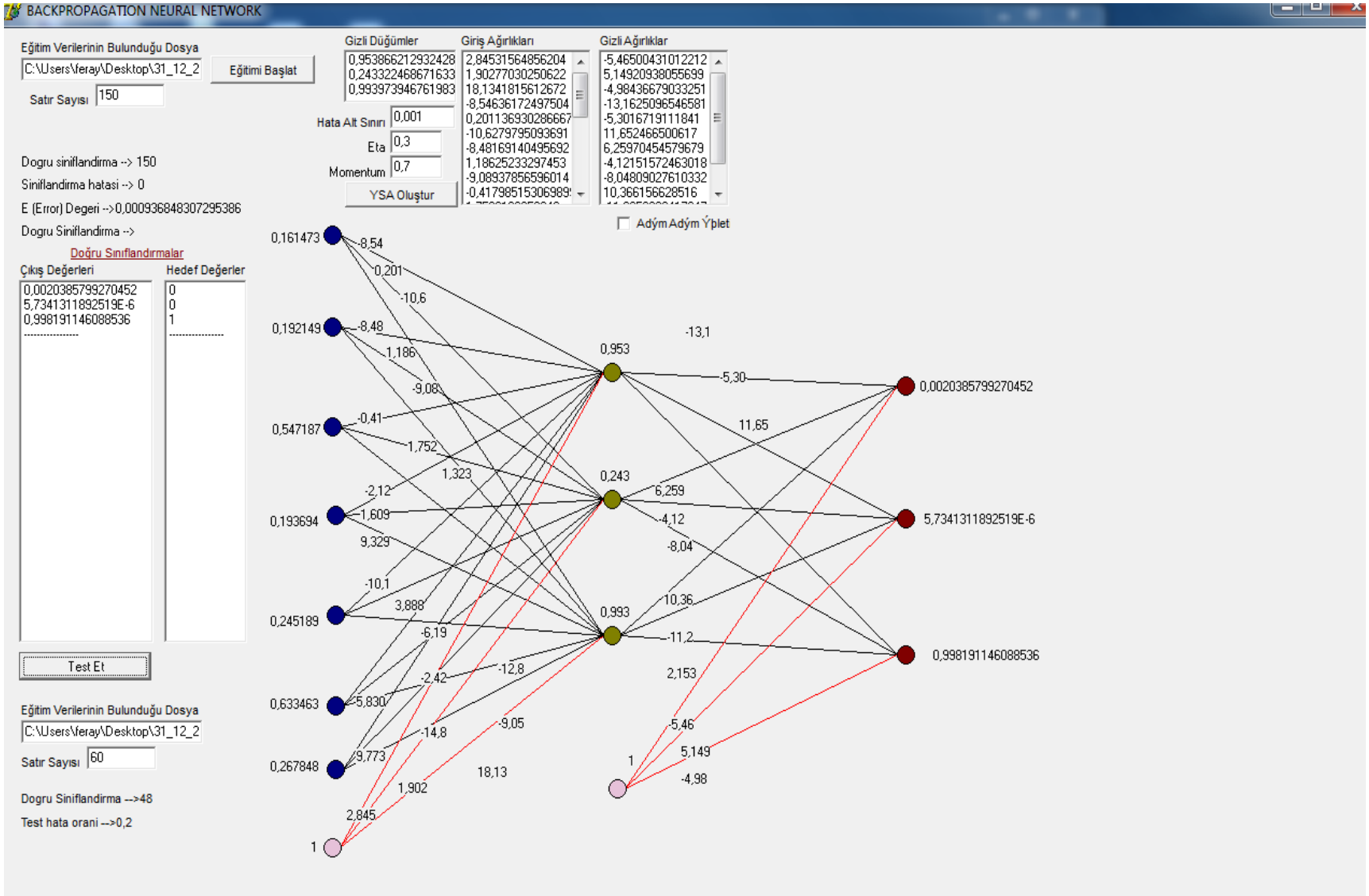
İlk denemede tasarlanan ağı eğitmek için kullanılan eğitim dosyası veri setinde verilen ilk 160 örnek durumdan oluşmaktadır. Yukarıda özellikleri verilen bilgisayar ağı eğitmek için 10.06 saat harcamıştır. İlk eğitimde hata alt sınır değeri 0.0001, eta 0.3 ve momentum 0.7 alınmış ve bu hata alt sınır değerine kadar 1371 epok hesaplama yapılmıştır. Doğru sınıflandırılan örnek sayısı 160 ve ağın toplam hatası 9.99242735755904E-5 olarak bulunmuştur. Bulunan ağırlık değerleri veri setinde verilen son 50 örnek duruma sahip test dosyasıyla test edilmiştir. Doğru sınıflandırılan örnek sayısı 43, test hata oranı 0.14 olarak bulunmuştur. Şekil 3.2'de bu denemede yapay sinir ağının eğitim safhasının sonucu ayrıntılı olarak görsel biçimde verilmiştir. Yapay sinir ağının ağırlık değerleri, ağın toplam hatası ve hata alt sınır değerine kadar hesaplanan epok sayısı şekil üzerinde verilmiştir.

İkinci denemede tasarlanan ağı eğitmek için kullanılan eğitim dosyası veri setinde verilen ilk 160 örnek durumdan oluşmaktadır. Fakat ilk ağırlık değerleri rastgele seçildiğinden ilk denemedeki ağırlık değerlerinden farklıdır. Yukarıda özellikleri verilen bilgisayar ağı eğitmek için 15.03 saat harcamıştır. İkinci eğitimde hata alt sınır değeri 0.0001, eta 0.3 ve momentum 0.7 alınmış ve bu hata alt sınır değerine kadar 1513 epok hesaplama yapılmıştır. Doğru sınıflandırılan örnek sayısı 160 ve ağın toplam hatası 9.9903711598087E-5 olarak bulunmuştur. Bulunan ağırlık değerleri veri setinde verilen son 50 örnek duruma sahip test dosyasıyla test edilmiştir. Doğru sınıflandırılan örnek sayısı 43, test hata oranı 0.14 olarak bulunmuştur.

Üçüncü denemede tasarlanan ağı eğitmek için kullanılan eğitim dosyası veri setinde verilen örneklerden çıkışları farklı olan ilk 50 örnek seçilerek oluşturulan 150 örneklik veri setinden oluşmaktadır. Yukarıda özellikleri verilen bilgisayar ağı eğitmek için 11.26 saat harcamıştır. Üçüncü eğitimde hata alt sınır değeri 0.001, eta 0.3 ve momentum 0.7 alınmış ve bu hata alt sınır değerine kadar 420 epok hesaplama yapılmıştır. Doğru sınıflandırılan örnek sayısı 150 ve ağın toplam hatası 9.368483072953E-4 olarak bulunmuştur. Bulunan ağırlık değerleri veri setinde verilen diğer kalan 60 örnek duruma sahip test dosyasıyla test edilmiştir. Doğru sınıflandırılan örnek sayısı 48, test hata oranı 0.2 olarak bulunmuştur. Şekil 3.3'de bu denemede yapay sinir ağının test safhasının sonucu ayrıntılı olarak görsel biçimde verilmiştir. Test safhasında kullanılan yapay sinir ağının ağırlık değerleri, ağın toplam hatası, doğru sınıflandırılan örnek sayısı, test safhasında kullanılan örnek sayısı ve test hata oranı şekil üzerinde verilmiştir.



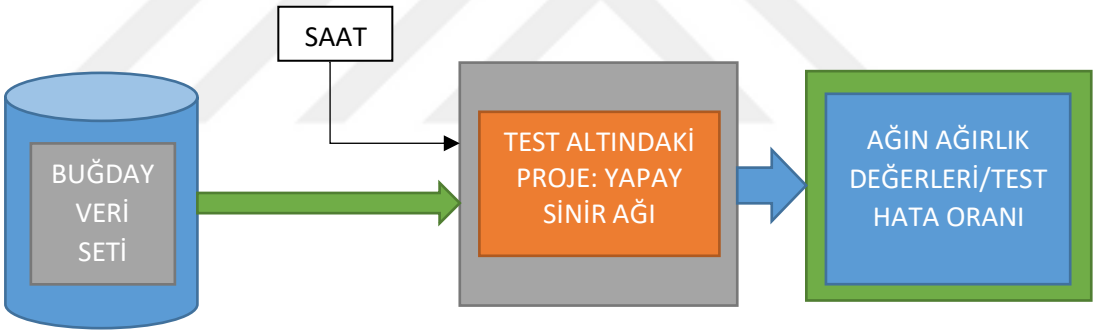
Şekil 3.2. Seri İşlem Yapan Platform İçin Ağın Eğitim Sonucu



Şekil 3.3. Seri İşlem Yapan Platform İçin Ağın Test Sonucu

3.2. Buğday Sınıflandırma Probleminin Paralel İşlem Yapan FPGA Platformunda Çözülmesi

Aynı veri setini sınıflandırmak için oluşturulan yapay sinir ağı VHDL dilinde Intel FPGA 19.1 Pro Edition programlama aracı kullanılarak yazılmış ve ModelSim-Intel FPGA Starter Edition 10.6d modelleme programında simülasyonu gerçekleştirilmiştir. Yapay sinir ağının eğitimi ve testi ayrı ayrı kodlanarak projelendirilmiş, projelerin simülasyonunu yapmak için test programları yazılmıştır. Test programları, test altındaki projelere buğday veri setindeki örnekleri giriş olarak beslemiş, yapay sinir ağının çıktıları program içindeki değişkenlerde tutulmuştur. Test programının blok diyagram gösterimi Şekil 3.4'de verilmiştir.



Şekil 3.4. Test Programının Blok Diyagram Gösterimi

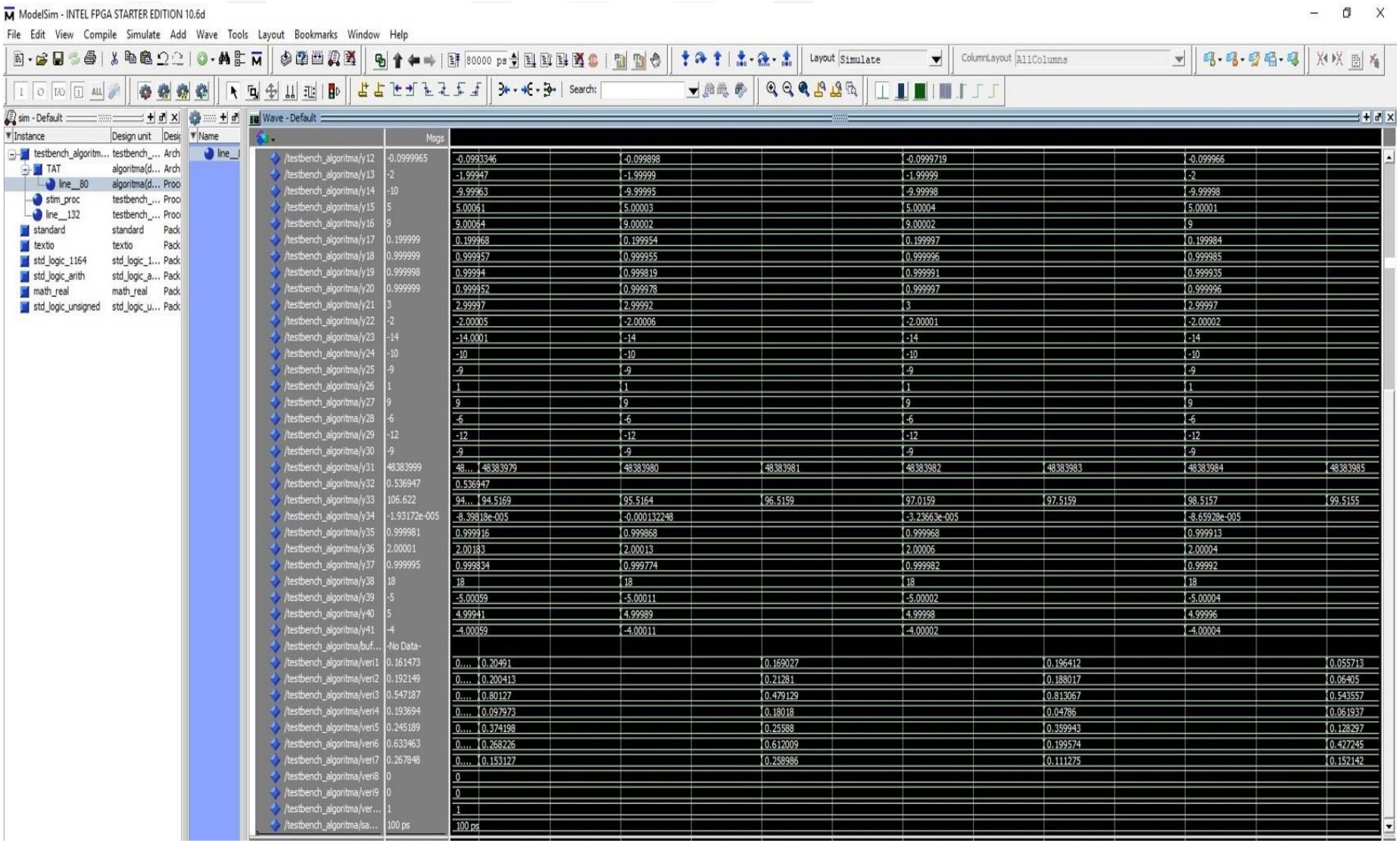
Sınıflandırma probleminin çözümünde giriş nöron sayısı 8, gizli katman sayısı 1, gizli katman nöron sayısı 4, çıkış nöron sayısı 3 olan bir yapay sinir ağı oluşturulmuştur. Programın çalışması esnasında öncelikle ağın eğitilmesi gerçekleştirilmiştir. Yapay sinir ağı eğitimin sonunda elde edilen ağırlık değerleri kullanılarak test edilmiştir. Veri setinin tamamı eğitim için kullanılmamıştır. Bunlardan yaklaşık %25'i test aşaması için kullanılmıştır. Yapay sinir ağının hata oranı $E \leq 0{,}0001$ olmuştur. α momentum katsayısı 0.7, λ öğrenme katsayısı 0.3 olarak seçilmiştir.

İlk denemede tasarlanan ağı eğitmek için kullanılan eğitim dosyası veri setinde verilen ilk 160 örnek durumdan oluşmaktadır. ModelSim simülasyon programında ağı eğitmek için yaklaşık 48 dakika harcanmış, 2419 epok hesaplama yapılmıştır. Doğru

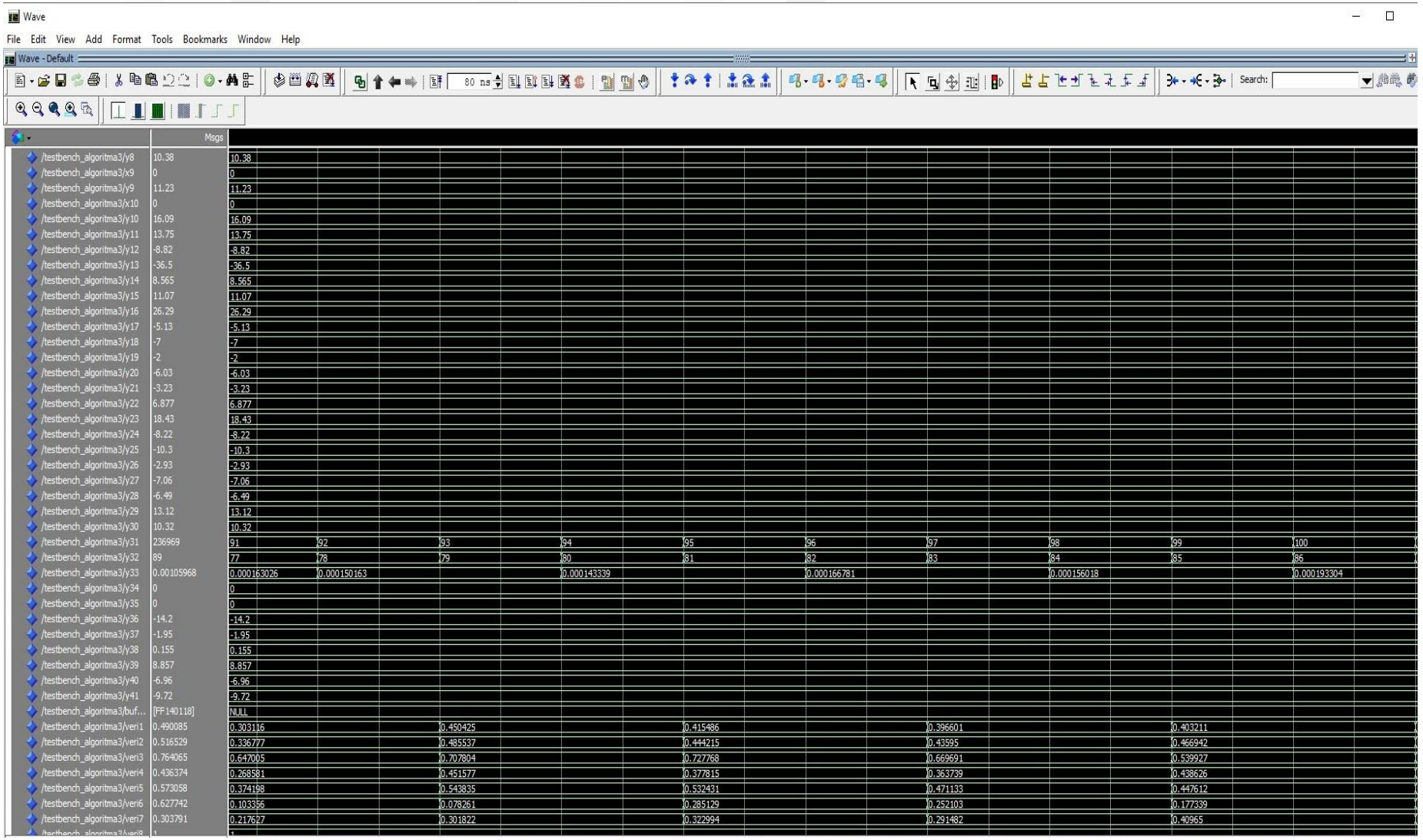
sınıflandırılan örnek sayısı 160 ve ağın toplam hatası 8.65928E-5 olarak bulunmuştur. Bulunan ağırlık değerleri veri setinde verilen son 50 örnek duruma sahip test dosyasıyla test edilmiştir. Ağın sınıflandırma başarısı %88 olarak bulunmuştur. Şekil 3.5'de bu deneme için ortaya çıkan ağın eğitim sonucu ModelSim ekran görüntüsü olarak verilmiştir. Bu görüntüde wave penceresindeki testbench_algoritma/y34 nesnesi ağın toplam hatası, testbench_algoritma/veri1-testbench_algoritma/veri7 nesneleri ağın giriş değerleri, testbench_algoritma/y31 iterasyon sayısı, testbench_algoritma/veri8-testbench_algoritma/veri10 nesneleri ağın beklenen çıktı değerleri ve diğer nesneler ağın ağırlık katsayılarıdır.

İkinci denemede ağı eğitmek için yaklaşık 43 dakika harcanmış, 1183 epok hesaplama yapılmıştır. Doğru sınıflandırılan örnek sayısı 140 ve ağın toplam hatası 6.92913E-5 olarak bulunmuştur. Bulunan ağırlık değerleri veri setinde verilen son 70 örnek duruma sahip test dosyasıyla test edilmiştir. Ağın sınıflandırma başarısı %76 olarak bulunmuştur.

Üçüncü denemede ağı eğitmek için yaklaşık 53 dakika harcanmış, 2092 epok hesaplama yapılmıştır. Doğru sınıflandırılan örnek sayısı 150 ve ağın toplam hatası 2.22429E-5 olarak bulunmuştur. Bulunan ağırlık değerleri veri setinde verilen son 60 örnek duruma sahip test dosyasıyla test edilmiştir. Ağın sınıflandırma başarısı %86 olarak bulunmuştur. Şekil 3.6'de bu deneme için ortaya çıkan ağın test sonucu ModelSim ekran görüntüsü olarak verilmiştir. Bu görüntüde wave penceresindeki testbench_algoritma3/y32 ağın sınıflandırma başarısıdır.



Şekil 3.5. Paralel İşlem Yapan Platform İçin Ağın Eğitim Sonucu



Şekil 3.6. Paralel İşlem Yapan Platform İçin Ağın Test Sonucu

4. SONUÇLAR VE TARTIŞMA

Bu tez çalışmasında ülkemizde alanında uzman kişiler tarafından yapılan buğday sınıflandırması işlemi nasıl hızlı ve objektif bir şekilde yapılabilir sorusuna çözüm aranmıştır. Bu amaç doğrultusunda seri ve paralel işlem yapan platformlar üzerinde yapay sinir ağları kullanılmış, paralel işlem yapan platform olarak FPGA seçilmiş, FPGA için VHDL dilinde yazılan yapay sinir ağı kodunun doğrulaması ve simülasyonu ModelSim programında gerçekleştirilmiştir. Sonuçlar Çizelge 4.1'de verilmiştir.

Sonuçlar kıyaslandığında YSA'ların eğitim safhasında seri işlem yapan platformlar oldukça fazla zaman harcamakta, bu durum ise buğday sınıflandırma probleminin çözümünde amacımıza hizmet etmemektedir. Ayrıca YSA'lar yüksek CPU gücüne ve bellek kapasitesine ihtiyaç duymaktadır.

FPGA'lar ise yapıları gereği paralel işlem yaptıklarından bir iterasyon hesaplamayı tek saat döngüsünde yapmakta, bu durum ise YSA'ların eğitim safhasında harcadıkları zamanı anlamlı şekilde kısaltmaktadır. FPGA'lar hata alt sınır değerine bağlı olarak daha fazla epok hesaplama yapmalarına rağmen YSA'ların eğitimi çok daha kısa sürelerde tamamlanmıştır. Ayrıca ağın toplam hata değerleri çok daha küçük değerler çıkmıştır. Bu sonuç ağın sınıflandırma başarısını artırmıştır. Çizelge 4.1 üzerinde FPGA platformu için elde edilen başarılı değerler koyu renkler ile yazılmıştır.

Seçilen FPGA platformunun türüne göre saat frekansları kullanıcının isteğine göre değiştirilebilmekte optimum fiyat/performans ayarlaması yapılabilmektedir. FPGA kullanımının bir diğer avantajı ise sonsuz kez yeniden yapılandırılabilir olmasıdır. Tasarımcı istediği kadar platform değişikliğine gitmeden tasarımı değiştirebilir. Ayrıca FPGA kullanımının diğer bir avantajı tasarım son halini aldıktan ve FPGA programlandıktan sonra programın FPGA içerisinde kalıcı olarak yüklenebilmesidir. Bu ise FPGA koduna dışarıdan müdahaleleri ortadan kaldırarak sistemin güvenilirliğini artırmaktadır. Dezavantaj olarak sayabileceğimiz bir konu ise YSA'ların FPGA üzerinde uygulanması işlemi oldukça zaman alan ve uzmanlık gerektiren bir işlem olmasıdır.

Sonuç olarak tezin amacına ulaşılmış, buğday sınıflandırma probleminin paralel işlem yapan platformlar üzerindeki çözümünün üstünlükleri ortaya konmuştur.

Çizelge 4.1 Sonuçlar

Değişkenler	Seri İşlem Yapan Platform			Paralel İşlem Yapan Platform		
	1. Deneme	2. Deneme	3. Deneme	1. Deneme	2. Deneme	3. Deneme
Eğitim İçin Harcanan Süre (Saat)	10.06	15.03	11.26	**0.8**	**0.71**	**0.88**
Eğitimde Kullanılan Örnek Sayısı	160	160	150	160	140	150
α Momentum Katsayısı	0.7	0.7	0.7	0.7	0.7	0.7
λ Öğrenme Katsayısı	0.3	0.3	0.3	0.3	0.3	0.3
Hata Alt Sınır Değeri	0.0001	0.0001	0.001	0.0001	0.0001	0.0001
Eğitim İçin Hesaplanan Epok Sayısı	1371	1513	420	**2419**	**1183**	**2092**
Ağın Toplam Hatası	9.99242E-5	9.99037E-5	9.36848E-4	**8.65928E-5**	**6.92913E-5**	**2.22429E-5**
Test İçin Kullanılan Örnek Sayısı	50	50	60	50	70	60
Doğru Sınıflandırılan Örnek Sayısı	43	43	48	**44**	**53**	**51**
Yanlış Sınıflandırılan Örnek Sayısı	7	7	12	6	17	9
Ağın Sınıflandırma Başarısı (%)	86	86	80	**88**	76	**86**

KAYNAKLAR

[1] Soylu T., 2018, Makine Öğrenmesi Yöntemleri Kullanılarak FPGA Tabanlı Gerçek Zamanlı Yeni Bir Trafik Sınıflandırma Mimarisi Tasarımı, Doktora Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne.

[2] Manisalı M., 2017, İnsansız Hava Araçlarında Denge Sistem Kontrolünün Bulanık Mantıklı FPGA ve Gömülü Sistemler İle Tasarımı ve Uygulaması, Yüksek Lisans Tezi, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü, Denizli.

[3] Köse E., 2017, FPGA Üzerinde HSYA ve HSYA'ya Özel Öğrenme Algoritmalarının Birlikte Gerçeklenmesi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[4] Meriç V., 2016, Hücresel Yapay Sinir Ağı İşlemcisi Tasarımı ve FPGA Gerçeklemesi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[5] Yılmaz A.R., 2014, FPGA Üzerinde Diferansiyel Gelişim Algoritması İle Yapay Sinir Ağı Eğitimi, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[6] Çavuşlu M.A., 2013, Yapay Sinir Ağları Eğitiminin Gradyen Tabanlı ve Global Arama Algoritmaları İle FPGA Üzerinde Donanımsal Gerçeklenmesi, Yüksek Lisans Tezi, Niğde Üniversitesi Fen Bilimleri Enstitüsü, Niğde.

[7] Alpay M., 2013, Çok Katmanli Bir Hücresel Sinir Ağı Emülatörünün Tasarlanması ve FPGA Üzerinde Gerçeklenmesi, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[8] Özkan İ. A., 2013, Elektromanyetik Filtreler İçin FPGA Tabanlı Adaptif Kontrolör Tasarımı ve Gerçekleştirilmesi, Doktora Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya.

[9] Cesur E., 2013, Gerçek Zamanlı Bir Hücresel Sinir Ağı Yapısının Tasarımı ve Bu Yapıyla Gabor Filtrelerinin FPGA Üzerinde Gerçeklenmesi, Doktora Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[10] Yıldız N., 2013, Bir Hücresel Sinir Ağı Emülatörünün Tasarlanması ve FPGA Üzerinde Gerçeklenmesi, Doktora Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[11] Temür G., 2013, Yapay Sinir Ağlarının Otomatik Olarak FPGA Çipine Uygulanması İçin Denetleyici Tasarım Aracı, Yüksek Lisans Tezi, Düzce Üniversitesi Fen Bilimleri Enstitüsü, Düzce.

[12] Sarıtekin N.K., 2011, Yapay Sinir Ağlarının Otomatik Olarak FPGA'ya Uygulanması İçin Veri Yolu Tasarım Aracı, Yüksek Lisans Tezi, Düzce Üniversitesi Fen Bilimleri Enstitüsü, Düzce.

[13] Özdemir A.T., 2010, Erken Ventriküler Kasılmalarda YSA Tabanlı Bir Sınıflandırıcının FPGA İle Gerçekleştirilmesi, Doktora Tezi, Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Kayseri.

[14] Yılmaz N., 2008, Alan Programlamalı Kapı Dizileri (FPGA) Üzerinde Bir Yapay Sinir Ağları (YSA)'nın Tasarlanması ve Donanım Olarak Gerçekleştirilmesi, Yüksek Lisans Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya.

[15] Uçar A., 2007, Türkçe Fonemlerin Sınıflandırılmasında Kullanılan Sinir Ağının FPGA Uygulaması, Yüksek Lisans Tezi, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

[16] Casado J. G., 2008, Implementation of An Artificial Neuron In VHDL, Master Degree Thesis, Malardalens Högskola Universty of Melardalen, Vasteras.

[17] "Exploring Convolutional Neural Networks On The ρ-vex Architecture", Erişim Adresi: https://www.semanticscholar.org/paper/exploring-convolutional-neural-networks-on-the-tetteroo/64de1fcfed56dc8c338260afec09ea2faf1b176f, Erişim Tarihi: 20 Mayıs 2019.

[18] Lozıto G. M., Laudani A., Fulginei F.R., Salvini A., FPGA Implementations of Feed Forward Neural Network by Using Floating Point Hardware Accelerators, Theoretical and Applied Electrical Engineering, 12(1), 30-39, 2014.

[19] "A Survey of FPGA Based Deep Learning Accelerators: Challenges and Opportunities", Erişim Adresi: https://arxiv.org/pdf/1901.04988.pdf, Erişim Tarihi: 23 Nisan 2019.

[20] Hajduk Z., Reconfigurable FPGA Implementation of Neural Networks, Neurocomputing, 308, 227-234, 2018.

[21] Altamura G., 2018, Development of Hardware Accelerators on FPGA for Convolutional Neural Networks, Master Degree Thesıs, Politecnico di Torino, Torino.

[22] Babalık A, 2007, Yapay Sinir Ağları İle Buğday Tanelerinin Kalite Tespiti, Doktora Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya.

[23] Öztürk M., Paksoy T., Buğday Tipi Sınıflandırma İçin Yapay Sinir Ağı Uygulaması: Yeni Bir Yapay Zeka Eğitimi Yazılımı, Kahramanmaras Sutcu Imam University Journal of Engineering Sciences, 21(3), 246-257, 2018.

[24] Taner A., Tekgüler A., Sauk H., Yapay Sinir Ağları İle Makarnalık Buğday Çeşitlerinin Sınıflandırılması, Anadolu Tarım Bilimleri Dergisi, 2015(30), 51-59, 2015.

[25] Öztemel Prof. Dr. E., Yapay Sinir Ağları, Papatya Yayıncılık, İstanbul, 2012.

[26] "Neural Networks:Basics", Erişim Adresi: http://www.site.uottawa.ca/~petriu/nn basics-tutorial-2004.pdf, Erişim Tarihi: 23 Mayıs 2019.

[27] Volnei A. P., Circuit Design With VHDL, Mit Press, London, 2004.

[28] Çavuşlu M. A., Kösten M. M., VHDL İle Sayısal Tasarım ve FPGA Uygulamaları, Kodlab Yayınları, İstanbul, 2015.

[29] Savran Dr. İ., Donanım Tanımlama Dili VHDL ve FPGA Uygulamaları, Papatya Yayıncılık, İstanbul, 2017.

[30] Sarıtaş E., Karataş S., Her Yönüyle FPGA ve VHDL, Palme Yayıncılık, Ankara, 2015.

[31] " Uci Machine Learning Repository", Erişim Adresi: http://archive.ics.uci.edu/ml/machine-learning-databases/00236/seeds_dataset.txt. Erişim Tarihi: 23 Ocak 2019.

EK 1

Çizelge A.1. Buğday Veri Seti [31]

x1	x2	x3	x4	x5	x6	x7	y1	y2	y3
0,440982	0,502066	0,57078	0,486486	0,486101	0,189302	0,34515	1	0	0
0,405099	0,446281	0,662432	0,368806	0,501069	0,032883	0,215165	1	0	0
0,349386	0,347107	0,87931	0,220721	0,50392	0,251453	0,150665	1	0	0
0,306893	0,316116	0,793103	0,239302	0,533856	0,194243	0,140817	1	0	0
0,524079	0,533058	0,864791	0,427365	0,664291	0,076701	0,322994	1	0	0
0,357885	0,371901	0,789474	0,274212	0,486101	0,220637	0,215165	1	0	0
0,387158	0,429752	0,651543	0,373874	0,448325	0,366784	0,344658	1	0	0
0,332389	0,349174	0,753176	0,293356	0,478974	0,251583	0,236829	1	0	0
0,570349	0,630165	0,604356	0,649775	0,595153	0,165767	0,668636	1	0	0
0,552408	0,586777	0,725045	0,554617	0,623664	0,156536	0,499261	1	0	0
0,440982	0,504132	0,558076	0,458896	0,436208	0,491217	0,391433	1	0	0
0,324835	0,36157	0,64882	0,303491	0,406985	0,12377	0,237322	1	0	0
0,311615	0,332645	0,725045	0,304054	0,40556	0,418794	0,107829	1	0	0
0,301228	0,340909	0,615245	0,326577	0,374911	0,308273	0,173806	1	0	0
0,29745	0,338843	0,601633	0,328266	0,344975	0,281749	0,150665	1	0	0
0,377715	0,386364	0,827586	0,254505	0,501069	0,444668	0,129	1	0	0
0,321058	0,293388	1	0,123874	0,536707	0,581063	0,129	1	0	0
0,481586	0,483471	0,88657	0,353604	0,630078	0,108427	0,259478	1	0	0
0,388102	0,371901	0,972777	0,172297	0,595866	0,130271	0,064008	1	0	0
0,201133	0,239669	0,549002	0,184122	0,298646	0,433876	0,194485	1	0	0
0,33711	0,411157	0,456443	0,427365	0,355666	0,299952	0,323486	1	0	0
0,332389	0,382231	0,58167	0,349662	0,383464	0,250023	0,344658	1	0	0

0,499528	0,514463	0,823049	0,404842	0,625089	0	0,281635	1	0	0
0,140699	0,169421	0,529038	0,112613	0,218104	0,084502	0,217627	1	0	0
0,417375	0,485537	0,522686	0,501126	0,438346	0,133391	0,237322	1	0	0
0,528801	0,568182	0,696915	0,525901	0,563792	0,01793	0,387986	1	0	0
0,229462	0,278926	0,508167	0,279279	0,282252	0,339089	0,150665	1	0	0
0,203022	0,260331	0,438294	0,279279	0,232359	0,226098	0,172329	1	0	0
0,332389	0,365702	0,670599	0,361486	0,42124	0,258604	0,255539	1	0	0
0,270066	0,332645	0,474592	0,34741	0,31005	0,359633	0,284589	1	0	0
0,242682	0,291322	0,527223	0,3125	0,245902	0,011702	0,264402	1	0	0
0,462701	0,522727	0,583485	0,483108	0,528154	0,34416	0,349089	1	0	0
0,3305	0,413223	0,406534	0,460586	0,396294	0,410212	0,384047	1	0	0
0,316336	0,363636	0,587114	0,386261	0,370634	0,176689	0,242738	1	0	0
0,421152	0,469008	0,633394	0,45777	0,497505	0,177339	0,414082	1	0	0
0,522191	0,535124	0,833938	0,456081	0,609408	0,195673	0,454948	1	0	0
0,529745	0,590909	0,592559	0,521959	0,59444	0,267576	0,496307	1	0	0
0,612842	0,613636	0,905626	0,525338	0,750535	0,284869	0,475135	1	0	0
0,397545	0,43595	0,673321	0,426239	0,468995	0,305153	0,388971	1	0	0
0,348442	0,363636	0,783122	0,280405	0,476123	0,769728	0,237322	1	0	0
0,278565	0,297521	0,716878	0,252815	0,374911	0,23689	0,324471	1	0	0
0,274788	0,297521	0,699637	0,254505	0,376336	0,192942	0,323486	1	0	0
0,242682	0,235537	0,842105	0,134572	0,406985	0,220507	0,129985	1	0	0
0,463645	0,506198	0,670599	0,550676	0,545973	0,513061	0,4968	1	0	0
0,426818	0,440083	0,821234	0,382883	0,593015	0,307233	0,325455	1	0	0
0,303116	0,336777	0,647005	0,268581	0,374198	0,103356	0,217627	1	0	0
0,450425	0,485537	0,707804	0,451577	0,543835	0,078261	0,301822	1	0	0
0,415486	0,444215	0,727768	0,377815	0,532431	0,285129	0,322994	1	0	0

0,396601	0,43595	0,669691	0,363739	0,471133	0,252103	0,291482	1	0	0
0,403211	0,466942	0,539927	0,438626	0,447612	0,177339	0,40965	1	0	0
0,362606	0,411157	0,607985	0,386261	0,457591	0,417363	0,30773	1	0	0
0,490085	0,516529	0,764065	0,436374	0,573058	0,627742	0,303791	1	0	0
0,368272	0,454545	0,414701	0,459459	0,344262	0,435697	0,431807	1	0	0
0,353163	0,386364	0,680581	0,340653	0,40556	0,333238	0,34712	1	0	0
0,371105	0,452479	0,431942	0,474099	0,344262	0,093084	0,476613	1	0	0
0,419263	0,487603	0,523593	0,45214	0,414825	0,151855	0,452979	1	0	0
0,365439	0,400826	0,668784	0,275338	0,532431	0,264845	0,258493	1	0	0
0,408876	0,417355	0,839383	0,273086	0,557377	0,049006	0,280158	1	0	0
0,452314	0,487603	0,704174	0,429617	0,562366	0,160436	0,346135	1	0	0
0,143532	0,219008	0,282214	0,146396	0,286529	0,095815	0	1	0	0
0,078376	0,092975	0,546279	0,061374	0,156807	0,251583	0,043328	1	0	0
0,060434	0,045455	0,688748	0,001689	0,177477	0,195543	0,090596	1	0	0
0,167139	0,161157	0,764065	0,099662	0,293656	0,319195	0,042344	1	0	0
0,248347	0,295455	0,543557	0,279279	0,313614	0,441028	0,280158	1	0	0
0,206799	0,239669	0,576225	0,204392	0,282252	0,053427	0,129493	1	0	0
0,216242	0,225207	0,724138	0,135135	0,348539	0,206335	0,043328	1	0	0
0,354108	0,404959	0,585299	0,411599	0,399145	0,07124	0,310684	1	0	0
0,322946	0,38843	0,493648	0,399775	0,376336	0,188782	0,301822	1	0	0
0,356941	0,409091	0,585299	0,377252	0,372773	0,090874	0,38454	1	0	0
0,202077	0,27686	0,342105	0,288851	0,179615	0,359893	0,269818	1	0	0
0,664778	0,737603	0,537205	0,727477	0,663578	0,430496	0,75874	0	1	0
0,590179	0,673554	0,491833	0,618806	0,608696	0,50838	0,668636	0	1	0
0,629839	0,68595	0,618875	0,607545	0,687099	0,490697	0,626292	0	1	0
0,804533	0,795455	0,907441	0,706644	0,926586	0,282269	0,768095	0	1	0

0,588291	0,640496	0,639746	0,629505	0,610121	0,421134	0,650911	0	1	0
0,583569	0,663223	0,505445	0,578829	0,575909	0,540236	0,628262	0	1	0
0,635505	0,72314	0,470054	0,655968	0,550962	0,39773	0,690793	0	1	0
0,955619	0,995868	0,618875	0,945946	0,843906	0,479255	0,951256	0	1	0
0,78848	0,842975	0,607078	0,870495	0,719173	0,558959	0,907435	0	1	0
0,616619	0,64876	0,735935	0,535473	0,667142	0,272127	0,604136	0	1	0
0,560907	0,605372	0,673321	0,54955	0,596579	0,61981	0,670113	0	1	0
0,767705	0,780992	0,813067	0,623311	0,874555	0,592765	0,669621	0	1	0
0,90746	0,92562	0,73775	0,780405	0,879544	0,573132	0,82127	0	1	0
0,84797	0,894628	0,633394	0,836149	0,81397	0,091914	0,863614	0	1	0
0,842304	0,88843	0,634301	0,826014	0,83464	0,285649	0,820286	0	1	0
0,725212	0,760331	0,715971	0,717342	0,727726	0,218167	0,826194	0	1	0
0,782814	0,795455	0,805808	0,66723	0,808268	0,114928	0,782866	0	1	0
0,792257	0,878099	0,461887	0,929054	0,741269	0,380437	0,974397	0	1	0
1	0,991736	0,823956	0,942568	1	0,652056	0,842935	0	1	0
0,971671	0,958678	0,862069	0,873311	0,999287	0,552718	0,887248	0	1	0
0,898017	0,946281	0,603448	0,947072	0,823236	0,154715	0,950271	0	1	0
0,771483	0,783058	0,819419	0,716779	0,831076	0,306193	0,755293	0	1	0
0,776204	0,801653	0,748639	0,773086	0,757662	0,321406	0,755293	0	1	0
0,75543	0,752066	0,893829	0,640766	0,876693	0,680792	0,668636	0	1	0
0,733711	0,849174	0,336661	0,994932	0,609408	0,541926	0,949778	0	1	0
0,593012	0,669421	0,514519	0,698198	0,593728	0,381087	0,712949	0	1	0
0,823418	0,863636	0,666062	0,811937	0,841055	0,352612	0,846381	0	1	0
0,792257	0,859504	0,549909	0,872748	0,657163	0,17929	0,95224	0	1	0
0,71577	0,795455	0,504537	0,772523	0,628653	0,271477	0,863614	0	1	0
0,767705	0,811983	0,661525	0,743243	0,751247	0,185011	0,776957	0	1	0

0,549575	0,586777	0,712341	0,461149	0,637919	0,448829	0,541113	0	1	0
0,698772	0,71281	0,826679	0,557995	0,758375	0,169408	0,648941	0	1	0
0,837583	0,845041	0,820327	0,683559	0,899501	0,460661	0,733629	0	1	0
0,811143	0,871901	0,577132	0,827703	0,749109	0,337009	0,84195	0	1	0
0,789424	0,828512	0,678766	0,759572	0,801853	0,338439	0,802068	0	1	0
0,778093	0,801653	0,758621	0,640766	0,823949	0,23247	0,669621	0	1	0
0,779981	0,77686	0,884755	0,705518	0,838204	0,270176	0,827671	0	1	0
0,664778	0,71281	0,65245	0,638514	0,672131	0,387718	0,694239	0	1	0
0,882908	0,931818	0,608893	1	0,807555	0,323356	1	0	1	0
0,751653	0,78719	0,711434	0,706081	0,74412	0,1265	0,677006	0	1	0
0,74221	0,766529	0,76225	0,68018	0,811832	0,191122	0,62777	0	1	0
0,830028	0,890496	0,576225	0,790541	0,827512	0,378746	0,711965	0	1	0
0,806421	0,805785	0,865699	0,722973	0,906629	0,174739	0,691777	0	1	0
0,807365	0,867769	0,58167	0,765766	0,789024	0,769338	0,755293	0	1	0
0,98017	1	0,705989	0,936937	0,970064	0,50864	0,884786	0	1	0
0,799811	0,834711	0,701452	0,854167	0,776194	0,192812	0,809453	0	1	0
0,790368	0,783058	0,903811	0,648649	0,903065	0,464042	0,606105	0	1	0
0,80831	0,834711	0,73412	0,757883	0,844619	0,301512	0,820286	0	1	0
0,783758	0,789256	0,841198	0,747748	0,811832	0,373675	0,712457	0	1	0
0,891407	0,927686	0,662432	0,897523	0,874555	0,298782	0,886755	0	1	0
0,911237	0,929752	0,740472	0,797297	0,949394	0,667789	0,821763	0	1	0
0,712937	0,766529	0,627042	0,653153	0,665004	0,371075	0,734613	0	1	0
0,526912	0,613636	0,460073	0,485923	0,539558	0,457801	0,582964	0	1	0
0,740321	0,735537	0,903811	0,608671	0,813257	0,28851	0,682422	0	1	0
0,509915	0,512397	0,892015	0,261261	0,678546	0,334278	0,30773	0	1	0
0,770538	0,778926	0,833031	0,682432	0,883108	0,445058	0,725258	0	1	0

0,761095	0,826446	0,559891	0,780405	0,687099	0,471453	0,779419	0	1	0
0,697828	0,710744	0,827586	0,608108	0,753386	0,193982	0,689316	0	1	0
0,903683	0,954545	0,593466	0,908784	0,814683	0,148864	0,820286	0	1	0
0,657224	0,671488	0,825771	0,502252	0,755524	0,598226	0,562285	0	1	0
0,728045	0,719008	0,931942	0,608108	0,801853	0,269396	0,710487	0	1	0
0,78848	0,807851	0,781307	0,701014	0,851746	0,278628	0,704087	0	1	0
0,452314	0,514463	0,567151	0,554617	0,45474	0,480685	0,628262	0	1	0
0,525968	0,603306	0,510889	0,532658	0,54526	0,4552	0,628262	0	1	0
0,469311	0,512397	0,673321	0,493806	0,554526	0,546997	0,653865	0	1	0
0,452314	0,464876	0,824864	0,32545	0,595153	0,368604	0,452979	0	1	0
0,639282	0,692149	0,638838	0,701577	0,672844	0,358983	0,714919	0	1	0
0,470255	0,566116	0,404719	0,574887	0,428368	0,243782	0,669621	0	1	0
0,473088	0,557851	0,452813	0,525338	0,467569	0,254834	0,60709	0	1	0
0,532578	0,572314	0,697822	0,54786	0,600143	0,390578	0,690793	0	1	0
0,234183	0,311983	0,362069	0,322635	0,259444	0,590165	0,431315	0	0	1
0,25779	0,316116	0,482759	0,361486	0,315752	0,815236	0,453471	0	0	1
0,259679	0,318182	0,489111	0,275901	0,316465	0,680011	0,387986	0	0	1
0,153919	0,188017	0,518149	0,182995	0,2402	0,611619	0,345643	0	0	1
0,116147	0,204545	0,175136	0,233671	0,104775	0,481855	0,324471	0	0	1
0,058546	0,14876	0,07804	0,213964	0,040627	0,702636	0,37223	0	0	1
0,07932	0,14876	0,23049	0,155968	0,063435	0,189302	0,301822	0	0	1
0,179415	0,216942	0,523593	0,207207	0,2402	0,475354	0,237814	0	0	1
0,199245	0,268595	0,372051	0,274212	0,200285	0,324396	0,392418	0	0	1
0,018886	0,107438	0,023593	0,23536	0,01283	0,610709	0,332349	0	0	1
0,117092	0,169421	0,376588	0,204955	0,149679	0,575992	0,387986	0	0	1
0,134089	0,229339	0,15245	0,28491	0,104063	0,809645	0,369769	0	0	1

0,157696	0,245868	0,228675	0,286599	0,14469	0,518912	0,414082	0	0	1
0,055713	0,130165	0,167877	0,180743	0,044904	0,333758	0,237322	0	0	1
0,07271	0,132231	0,27314	0,155405	0,089095	0,426855	0,366322	0	0	1
0,056657	0,132231	0,15608	0,197635	0,032074	0,656347	0,344658	0	0	1
0,070822	0,095041	0,467332	0,086712	0,156094	0,335708	0,238306	0	0	1
0,14542	0,272727	0	0,278716	0,081967	0,527884	0,34515	0	0	1
0,109537	0,229339	0,000907	0,306869	0,034212	0,469763	0,389463	0	0	1
0,084986	0,167355	0,165154	0,228041	0,046329	0,601087	0,389463	0	0	1
0,184136	0,260331	0,31216	0,310811	0,177477	0,301252	0,478582	0	0	1
0,135033	0,190083	0,38294	0,253941	0,128297	0,45585	0,388479	0	0	1
0,137866	0,206612	0,303993	0,207207	0,154669	0,549077	0,259478	0	0	1
0,18508	0,239669	0,432849	0,244369	0,240912	0,475094	0,323486	0	0	1
0,051936	0,078512	0,432849	0,063063	0,116892	0,731111	0,260955	0	0	1
0,142587	0,152893	0,646098	0,115991	0,221668	0,186701	0,264402	0	0	1
0,174693	0,243802	0,345735	0,236486	0,190306	0,540756	0,369769	0	0	1
0,147309	0,214876	0,328494	0,291667	0,147541	0,373545	0,40325	0	0	1
0,071766	0,146694	0,190563	0,155968	0,027085	0,464432	0,301822	0	0	1
0,061379	0,121901	0,252269	0,107545	0,060584	0,358333	0,280158	0	0	1
0,040604	0,121901	0,098004	0,239865	0,050606	0,776229	0,317085	0	0	1
0,090652	0,142562	0,339383	0,150901	0,153243	0,773629	0,215165	0	0	1
0,064212	0,115702	0,306715	0,106419	0,094797	0,460791	0,236829	0	0	1
0,076487	0,13843	0,266788	0,133446	0,094797	0,627092	0,28065	0	0	1
0,022663	0,113636	0,016334	0,213401	0,00784	0,574302	0,327917	0	0	1
0,01983	0,033058	0,461887	0,046171	0,136137	0,521122	0,267848	0	0	1
0,063267	0,123967	0,248639	0,161599	0,057021	0,594196	0,282127	0	0	1
0,014164	0,066116	0,225045	0,138514	0,008553	0,511891	0,218612	0	0	1

0,084042	0,132231	0,355717	0,158221	0,091233	0,664539	0,237814	0	0	1
0,152975	0,219008	0,337568	0,257883	0,187455	0,116488	0,324471	0	0	1
0,077432	0,11157	0,434664	0,107545	0,10335	0,545047	0,150665	0	0	1
0,176582	0,206612	0,567151	0,189752	0,275837	0,548947	0,309207	0	0	1
0,151086	0,196281	0,451906	0,192005	0,19886	0,532044	0,314623	0	0	1
0,100094	0,136364	0,448276	0,11768	0,156807	0,577813	0,303299	0	0	1
0,217186	0,280992	0,417423	0,335586	0,282252	0,704716	0,392418	0	0	1
0,091596	0,18595	0,106171	0,261261	0,037776	0,428675	0,32644	0	0	1
0,115203	0,214876	0,106171	0,289414	0,061297	0,537375	0,410143	0	0	1
0,030217	0,080579	0,264065	0,106419	0,032074	0,443888	0,215165	0	0	1
0,060434	0,084711	0,465517	0,106982	0,136137	0,878818	0,215657	0	0	1
0	0	0,514519	0	0,111903	0,547387	0,135401	0	0	1
0,032106	0,080579	0,280399	0,08277	0,06201	0,602387	0,258986	0	0	1
0,064212	0,092975	0,437387	0,108108	0,12402	0,418664	0,237322	0	0	1
0,120869	0,126033	0,647913	0,131194	0,230221	0,368214	0,301822	0	0	1
0,021719	0,086777	0,158802	0,158221	0	0,531524	0,28065	0	0	1
0,143532	0,177686	0,506352	0,189752	0,245902	0,437777	0,242738	0	0	1
0,208687	0,219008	0,706897	0,146959	0,353528	0,534125	0,194485	0	0	1
0,207743	0,231405	0,639746	0,182995	0,30221	0,613439	0,21615	0	0	1
0,262512	0,283058	0,696915	0,23705	0,354954	0,50773	0,281635	0	0	1
0,19169	0,260331	0,362976	0,287725	0,200285	0,330377	0,350566	0	0	1
0,20491	0,200413	0,80127	0,097973	0,374198	0,268226	0,153127	0	0	1
0,169027	0,21281	0,479129	0,18018	0,25588	0,612009	0,258986	0	0	1
0,196412	0,188017	0,813067	0,04786	0,359943	0,199574	0,111275	0	0	1
0,055713	0,06405	0,543557	0,061937	0,128297	0,427245	0,152142	0	0	1
0,199245	0,206612	0,719601	0,15991	0,328582	1	0,236829	0	0	1

0,168083	0,219008	0,441016	0,171734	0,23521	0,410082	0,237322	0	0	1
0,151086	0,163223	0,637024	0,134009	0,250178	0,372635	0,172821	0	0	1
0,060434	0,097107	0,3902	0,135698	0,117605	0,462872	0,238306	0	0	1
0,246459	0,258264	0,727768	0,189752	0,429081	0,981667	0,264402	0	0	1
0,118036	0,165289	0,399274	0,155405	0,146828	0,368344	0,258493	0	0	1
0,161473	0,192149	0,547187	0,193694	0,245189	0,633463	0,267848	0	0	1

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: LORTOĞLU, Murat
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 05.02.1978
Eskişehir Medeni hali	: Evli
Telefon	: 0 (312) 219 57 87
Faks	: 0 (312) 219 57 97
e-mail	: mlortoglu@havelsan.com.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Osmangazi Üniversitesi/Elektrik-Elektronik Müh.	1999
Yüksek Lisans	Selçuk Üniversitesi/Elektrik-Elektronik Müh.	2007

İş Deneyimi

Yıl	Yer	Görev
2002-2019	Havelsan A.Ş.	Kıdemli Sistem Mühendisi

Yabancı Dil

İngilizce, Fransızca