YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA İLE MOBİL ROBOT İÇİN ÖĞRENME ALGORİTMASI MODELLENMESİ

Elektronik ve Haberleşme Müh. Beril SIRMAÇEK

FBE Elektronik ve Haberleşme Anabilim Dalı Elektronik Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Yrd. Doç.Dr. Lale ÖZYILMAZ (YTÜ)

İSTANBUL,2007

İÇİNDEKİLER:

Sayfa

SİMGE LİSTESİ……iv

KISALTMA LİSTESİ……....v

ŞEKİL LİSTESİ……...…….……vi

ÇİZELGE LİSTESİ……...…….....vii

ÖNSÖZ……...……...ix

ÖZET……..x

ABSTRACT…….....xi

1. GİRİŞ……..…….……...1

2. YAPAY SİNİR AĞLARI VE ZEKA KAVRAMI……...……..3

2.1 Biyolojik Nöron……...……..3
2.2 Yapay Sinir Hücresi…….……...4
2.3 Yapay Sinir Ağları, Özellikleri ve Yapısı……...6
2.4 Yapay Sinir Ağlarının Sınıflandırılması…….7
2.5 Yapay Sinir Ağlarını Çekici Kılan Özellikler……...……..9
2.6 YSA Modelleri…….……11
2.7 Zeka…….……..12
2.8 Yapay Zeka…….....12
2.9 Yapay Zeka Çalışmalarının Tarih İçinde Gelişimi…….....14
2.10 Yapay Sinir Ağlarının Eğitilmesi……..15
2.11 Yapay Sinir Ağlarından Yararlanmanın Avantajları ve Dezavantajları……...16
2.12 Bulanık Mantık ve Yapay Zeka…….....17
2.13 Uzman Sistemler……..……18
2.14 Adaptasyon ve Öğrenme……..……19
2.15 Makine Öğrenmesi…….....19
2.16 Öğrenme Stratejileri……..…….....22

3. PROGRAMLANABİLİR LOJİK ELEMANLARIN MİMARİSİ VE PROGRAMLAMA TEKNİKLERİ…….……...…24

3.1 Programlanabilir Lojik Elemanların Gelişimi…….....25
3.2 VLSI…….……27
3.3 FPGA (Field Programmable Gate Array)…….....28
3.4 FPGA Teknolojisinin Ortaya Çıkış Sebebi…….....30
3.5 FPGA Mimarisi…….……...30
3.6 FPGA Programlama Teknolojileri……...…….....34
3.7 FPGA Kullanılarak Gerçekleştirilen Devrelerin Tasarım Süreci…….……35
3.8 Model Robot İçin Seçilen FPGA ve Özellikleri…...…….……36
3.9 FPGA Tabanlı Ağ Mimarisi……...…….……..37
3.10 YSA'nın FPGA Kullanılarak Gerçeklenmesi Üzerine Yapılan Çalışmalar.....38

3.11 VHDL……38

4. ROBOTİK……44

4.1 Robot……44
4.2 Robot Sensörleri……44
4.3 Robot Efektörleri……46
4.4 Mobil Robot……47
4.5 Robotikte Yapay Sinir Ağlarının Kullanımı……49
4.6 Haritalama……50
4.7 Robotun Öğrenme Algoritması……51
5. ÖĞRENME ALGORİTMASININ TASARIMI……55
5.1 PNN Ağları……55
5.2 Dezavantajların Kaldırılması için Yapılan Çalışma……59

6 LOJİK OLARAK ÖĞRENME ALGORİTMASININ GERÇEKLENMESİ……62

6.1 Yapay Sinir Ağlarının Donanımsal Olarak .Gerçeklenmesi……63
6.2 Kullanılan Öğrenme Algoritması……63
6.3 Tasarlanan FPGA Mimarisinin Blok Diyagramı……66
6.4 Yapay Sinir Ağının Xilinx Project Navigator Programında Yazılması……67
6.5 Modelsim ile FPGA Simülasyonu……72
6.7 Öğrenme Algoritmasının Matlab Simülasyonu……72
6.8 Mikrodenetleyici ile FPGA'den Gelen Sonuçların Yorumlanması……78
6.9 Arayüz ile Projenin Simülasyonu……79

7 MODEL ROBOTUN KULLANILABİLECEĞİ ALANLAR……80

8 SONUÇLAR……81

KAYNAKLAR……82

EKLER……83

SİMGE LİSTESİ

K	Giriş setlerinin ayrıldığı sınıf sayısı
Q	Giriş ve çıkış vektör seti sayısı
S1	Sağ sensörden alınan veri
S2	Sol sensörden alınan veri
Wij	i. nörona giren girişin j. ağırlığı
Δwij	i. nörona giren girişin j. ağırlığının öğrenme ile değiştirilme
X(t)	t anında robotun bulunması istenen konumun x eksenindeki değeri
X'(t+1)	t+1 anında robotun ilerleyeceği noktanın x eksenindeki değeri
(Xe,Ye)	Engelin bulunduğu noktanın koordinatları
(Xt,Yt)	Ulaşılmak istenen hedef noktanın koordinatları
Y(t)	t anında robotun bulunması istenen konumun y eksenindeki değeri
Y'(t+1)	t+1 anında robotun ilerleyeceği noktanın y eksenindeki değeri

KISALTMA LİSTESİ

AI	Artificial Intelligence
ANN	Artificial Neural Networks
ART	Adaptive Resonance Theory (Adaptif Rezonans Teorisi)
ASIC	Application Specific Integrated Circuit
CLB	Configurable Logic Block
CPLD	Complex Programmable Logic Device
EMI	Electromagnetic Interferance (Elektromanyetik etkileşim)
EPROM	Erasable Programmable Read Only Memory
FPGA	Field Programmable Gate Array
IEEE	International Electric and Electronic Engineers
MLP	Multilayer Perceptron
PAL	Programmable Array Logic
PCB	Printed Circuit Board
PLD	Programmable Logic Device
PNN	Probabilistic Neural Network
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RBF	Radial Based Function
ROM	Read Only Memory
SoC	System on Chip
SOM	Self Organizing Map
SOFM	Self Organizing Feature Map
SPLD	Simple Programmable Logic Device
SRAM	Static Random Access Memory
VHDL	VHISC High Level Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
YSA	Yapay Sinir Ağı
YSH	Yapay Sinir Hücresi
XST	Xilinx Synthesis Technology

ŞEKİL LİSTESİ

Sayfa

Şekil.2.1 Biyolojik sinir hücresi……..….....3
Şekil.2.2 Biyolojik sinir sisteminin blok gösterimi……...……4
Şekil.2.3 Bir yapay sinir hücresinin matematiksel modeli…….....4
Şekil.2.4 Çok katmanlı bir yapay sinir ağı mimarisinin gösterimi…….....8
Şekil.2.5 Öğrenme yaklaşımlarına göre yapay sinir ağları…….....9
Şekil.2.6 Tek bir nöronun çıkıştaki hataya etkisi……..…11
Şekil.2.7 Turing testinin gerçekleştirilmesi……...……..21
Şekil.2.8 Öğrenme stratejilerinin sınıflandırılması……...…...22
Şekil.3.1 PLA lojik devre elemanının iç mimarisi……..26
Şekil.3.2 PAL lojik devre elemanının iç mimarisi……..26
Şekil.3.3 CPLD lojik devre elemanının iç mimarisi……...27
Şekil.3.4 MPGA lojik devre elemanının iç mimarisi……..27
Şekil.3.5 Günümüz teknolojisi ile üretilmiş bir VLSI çip……...28
Şekil.3.6 FPGA blok diyagramı……...30
Şekil.3.7 FPGA'i oluşturan bloklar……31
Şekil.3.8 Kaba taneli lojik blok örneği…….....32
Şekil.3.9 FPGA ara bağlantıları……..32
Şekil.3.10 İki girişli bir LUT devresi……33
Şekil.3.11 İki girişli LUT……..33
Şekil.3.12 Programlanmış bir FPGA'in bir bölümü……..34
Şekil.3.13 FPGA kullanılarak tasarım sürecinin genel akış diyagramı……..…36
Şekil.4.1 Ultrasonik sensör ile cismin ve mesafesinin algılanması…….……45
Şekil.4.2 Ultrasonik uzaklık ölçümünde kullanılabilecek 40KHz'lik alıcı ve verici sensör çiftleri…….……46
Şekil.4.3 Model robotun blok diyagramı…….....47
Şekil.4.4 Robotun (Xr,Yr) noktasından, (Xt,Yt) hedef noktasına konumlanması…….….52
Şekil.4.5 Robotun (Xe,Ye) noktasında bulunan engelle karşılaşması…….....52
Şekil.5.1 PNN ağ mimarisi……...54
Şekil.5.2 σ sabitinin ayarlanarak kümelerin dağılımına göre olasılık yoğunluk fonksiyonlarının belirlenmesi……....57
Şekil.5.3 Yanlış sınıflandırmaya sebep olabilecek iki kümeye eşit uzaklıkta bir test örneği……....59
Şekil.6.1 Dijital bir yapay sinir ağı mimarisi……...60
Şekil.6.2 Mobil robotun blok diyagramı……..62
Şekil.6.3 Mobil robotun düz ilerleme istikametine göre dönüş açılarının ikisi……...63
Şekil.6.4 FPGA mimarisi blok diyagramı…….....65
Şekil.6.5 Xilinx ISE'de yeni bir projenin oluşturulması…….……66
Şekil.6.6 Xilinx ISE programında kullanılacak FPGA'in seçilmesi……...67
Şekil.6.7 VHDL modül dosyasının oluşturulması……...68
Şekil.6.8 Lojik yapının giriş ve çıkış pinlerinin oluşturulması……69
Şekil.6.9 VHDL kodlarının yazıldığı ekran……....70
Şekil.6.10 Yazılım ile YSA donanımının tanımlanması……....70
Şekil.6.11 Gerçeklenen yapay sinir ağının lojik yapısı……..……71
Şekil.6.12 FPGA üzerinde giriş çıkış pinlerinin seçilmesi……71
Şekil.6.13 Matlab kodları ile dizayn edilen ağ mimarisi…….…..…73
Şekil.6.14 Ağın beklenen çıkışları (mavi) ile ağın gerçek çıkışlarının (kırmızı) karşılaştırılması…….……75

Şekil.6.15 55 adet eğitim örneği için ağın beklenen çıkışları (mavi) ve ağın gerçek çıkışları (kırmızı)……..….76
Şekil.6.16 Mikrodenetleyicili motor kontrol sisteminin çalışma algoritması……78
Şekil.6.17 Model robotun engellerden kaçmayı öğrenerek hedefe varmasını gösteren simülasyon arayüzü…….79

ÇİZELGE LİSTESİ

Sayfa

Çizelge.2.1 Sinir sistemi ile YSA benzerlikleri…….....7
Çizelge.3.1 Xc3s200 genel özellikleri…….37
Çizelge.4.1 Öğrenebilen makineler ile klasik bilgisayar sistemlerinin Karşılaştırılması……49
Çizelge.6.1 Robotun dönüş açılarına karşılık gelen dijital kodlar……...64
Çizelge.6.2 Matlab simülasyonunda nöron girişlerine karşılık gelen değerler……...73
Çizelge.6.3 Matlab simülasyonunda nöron çıkışlarına karşılık gelen değerler….. ……....74
Çizelge.6.4 Eğitim örneği sayısı ile hata yüzdelerinin karşılaştırılması……..76
Çizelge.6.5 FPGA çıkış işaretleri ve mikrodenetleyicinin bunlara karşılık yapacağı işlemler……..77

ÖNSÖZ

Bu çalışmada FPGA üzerinde gerçekleşen mobil bir robotun çevreyi keşfedebilmesi ve engellerden sakınabilmesi amacı ile gerçeklenen yapay sinir ağı tasarımı, simülasyonu ve akış şemaları verilmiş, mobil robotun yapısı açıklanmıştır. Tasarımlar Xilinx ISE ve simülasyonlar ModelSim ve Matlab programlarının yardımları ile yapılmıştır.

Robotun engellerden kaçması için tasarlanan yapay sinir ağında olasılıksal yapay sinir ağı (PNN-Probabilistic Neural Network) yapısından faydalanılmış ve ağ yapısı geliştirilerek mobil robotumuza uyarlanmıştır. İki adet ultrasonik sensörden gelen bilgiler yapay sinir ağının girişlerini oluşturmaktadır. Gelen yeni sensör bilgilerini, eğitim setine göre yorumlayan robot hangi açıyla, hangi yöne yöneleceğine FPGA ile tasarlanmış yapay sinir ağı ile karar vermekte ve üç bit kodlanmış bir çıkışı mikrodenetleyicili motor sürücü devresine göndermektedir. Mikrodenetleyicili motor sürücü devresi iki adet DC motoru kontrol ederek, robotun dönme açısını ayarlamaktadır.

Tez çalışmalarıma ayırdığım zamanı anlayışla karşılayıp beni her zaman her şekilde destekleyen hayatımın anlamı İlker Oran'a, bana hem öğretmen hem de dost olan, bu zorlu dönemi atlatmamda ve yeni bilgilere ulaşmamda desteklerini gece gündüz esirgemeyen çok değerli hocam Yrd.Doç.Dr. Lale Özyılmaz ve araştırma görevlisi arkadaşım Oğuzhan Yavuz'a sonsuz teşekkürler ederim. Ayrıca çalışmadığım süreç boyunca maddi desteğinden dolayı Tübitak Bilim Adamı Yetiştirme Grubu'na teşekkürü bir borç bilirim.

ÖZET

Bu çalışmada mobil bir robot uygulaması için çevreyi öğrenecek ve engellerden kaçmayı öğrenecek bir YSA algoritması geliştirilmiş ve FPGA için donanımsal modellemesi yapılmıştır. Öğrenme algoritması PNN (Probabilistic Neural Network) yapısı kullanılarak modellenmiş ve PNN'e bu çalışmadaki dezavantajlarını düzeltecek bazı üstünlükler kazandırılmıştır. Tasarımın mimarisi, VHDL (Very High Speed Integrated Circuits Hardware Description Language) kullanarak gerçeklenmiştir. FPGA tasarımı, modelsim simülasyon programı ile test edildi. Aynı YSA mimarisinin kodların çalışma mantığı ve sınıflamalardaki başarımı Matlab simülasyonu ile kontrol edildi. Son olarak ise; mobil bir robotun engellerin olduğu bir odada hedef noktaya en kısa yoldan varmasını ve engel koordinatlarını öğrenmesini gösteren bir bilgisayar grafik arayüzü Delphi ortamında hazırlanmıştır. Hazırlanan arayüz programı boş bir odanın içine mobil robotun, engellerin, ulaşılması istenen hedef noktanın istenilen şekilde yerleştirilmesine müsaade etmektedir. Mobil robot, başla tuşuna basıldığında yavaş yavaş seçilen hedef noktasına doğru en kısa yolu seçerek ilerlemekte ve bu sırada karşısına gelen engellerden kaçmaktadır. Engelleri algılama, doğrultusundan ne kadar açı ile sapacağına karar verme ve tekrar hedefe konumlanma için bu çalışmada verilen algoritmadan yararlanılmıştır. Arayüz simülasyonu engellerin bulunduğu noktaları öğrenerek koordinatlarını kullanıcıya vermektedir, böylece robot odayı da öğrenmiş olmaktadır.

Model robotun Matlab ortamında hesaplanan sonuçlara göre öğrenme başarısı %88.88 olmuştur. Öğrenme algoritmasının Matlab simülasyonlarına ait başarım ve hata grafikleri bu çalışma içinde verilmiştir.

Anahtar Kelimeler: Yapay Zeka, yapay sinir ağları, olasılıksal yapay sinir ağları (PNN Probabilistic Neural Networks), FPGA, mobil robot, öğrenme algoritması

ABSTRACT

In this workout, a neural network algorithm has been developed for a mobile robot that learns to avoid obstacles in a room and learns the environment. The algorithm has been also simulated for implementing on a FPGA. The learning algorithm has been modelled with using PNN (Probabilistic Neural Network) architecture and some qualities have implemented to this architecture to avoid the disadvantages of the algorithm. The hardware architecture has been constructed ith VHDL (Very High Speed Integrated Circuits Hardware Description Language) language and simulated with Modelsim. The performance of the algorithm has been also tested with Matlab simulations. And finally, we have developed a simulation interface in Delphi environment which shows a demo room and a robot that learns the room. The interface program enables the user to put the mobil robot, obstacles and the target point to the desired places in an empty room. When the start button is pressed mobil robot begins to move through the target place with calculating the shortest distance and also avoiding from the obstacles in the room. The avoidance from the obstacles has been realised with using the learning algorithm which is briefly introduced in this study.

The success of the learning algorithm of the mobil robot has been calculated as 88.88% in Matlab environment. The performance results and graphics are also presented in this work.

Key words: Artificial Intelligence, Neural Network, PNN (Probabilistic Neural Networks), FPGA, mobil robot, learning algorithm

1. GİRİŞ

Yapay sinir ağları, yapay zeka biliminin altında araştırmacıların çok yoğun olarak ilgi gösterdikleri bir araştırma alanıdır. Yapay sinir ağlarının örnekler ile öğrenebilme ve genelleme yapabilme özellikleri onlara çok esnek ve güçlü araçlar olma özelliği kazandırmaktadır. Yapay sinir ağlarının az veriyle öğrenebilmeleri ve genelleme yapabilmeleri, doğrusal olmamaları diğer üstün özellikleridir. Bu sayede bir çok problemin çözümüne uygulanabilmektedir.

Yapay sinir ağları, model (desen) tanıma, görüntü işleme ve robotik uygulamaları gibi alanlarda oldukça zor problemleri çözebilecek yeteneğe sahiptir. Biyolojik yapıdan esinlenmiş YSA doğasında dağınık paralel işleme gerektirir. Bu sebeple, gerçek zamanlı ve çok yüksek hızlı uygulamalar yapay sinir ağları ile gerçeklenmeye çalışılır. Bu çalışmalar gerçek performanslarını ancak paralel çalışan mimariler üzerinde gösterebilirler.

Uygulamada, yapay sinir ağlarının FPGA ile gerçeklenmesi programlanabilir sistemlerde esneklik sağlar. Eğer gerçek zamanlı uygulamalar için VLSI teknolojisi kullanılarak bir yapay sinir ağı tabanlı işlemci yapılmak istenirse, bu gerçekleştirim hem zaman hem de maliyet açısından oldukça masraflı olacaktır. Tekrar düzenlenebilir FPGA programlanması ile özel amaçlı hızlı donanımlar çok geniş uygulamalar için kullanılabilecektir. FPGA'lerin geleneksel işlemcilerin sahip olmadığı hız, güvenlik ve paralel işlem yapabilme yeteneğine ve ayrıca VLSI teknolojisinin sahip olmadığı tekrar düzenlenebilirlik kabiliyetlerine sahip olması vasıtasıyla yapay sinir ağlarıyla çok uyumlu çalışmalar yapabilmekte ve yeni yapay sinir ağı algoritmalarına ışık tutmaktadır.

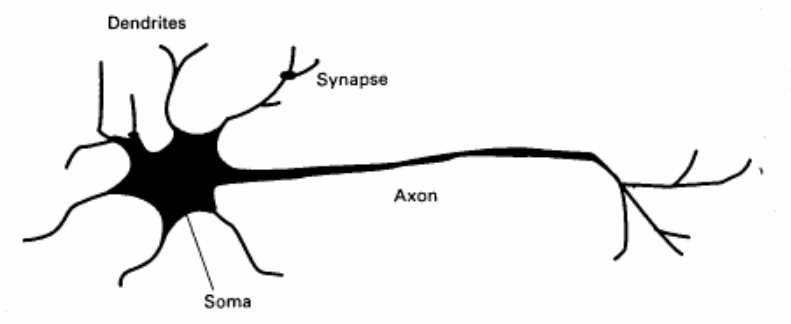
Bu çalışmanın amacı, FPGA kullanarak mobil bir robot uygulaması için çevreyi öğrenecek ve engellerden kaçmayı öğrenecek bir YSA eğitiminin simule edilmesidir. Dijital sistem mimarisi, PNN (Probabilistic Neural Network) algoritması kullanılarak çok katmanlı YSA'nın eğitimini gerçeklemek için tasarlanmıştır. Tasarım mimarisi, VHDL (Very High Speed Integrated Circuits Hardware Description Language) kullanarak gerçeklendi. FPGA modelsim simülasyon programı ile test edildi. Ve kodların çalışma mantığı, sınıflamalardaki hata miktarı aynı zamanda Matlab programı ile aynı YSA mimarisinin simülasyonu ile kontrol edildi.

2. YAPAY SİNİR AĞLARI VE ZEKA KAVRAMI

2.1 Biyolojik Nöron

Biyolojik sinir ağları insan beyninin çalışmasını sağlayan en temel yapı taşlarından birisidir. İnsanın bütün duyu organlarından gelen verileri alıp çevresini algılamasını sağlar.

İnsan beyninin işlem hızı oldukça yavaştır. Klasik bir işlemci saniyede 200 milyon tane işlem yapma kapasitesine sahipken biyolojik bir sinir ağının cevap verme süresi 1-2 mili saniye arasında değişmektedir.

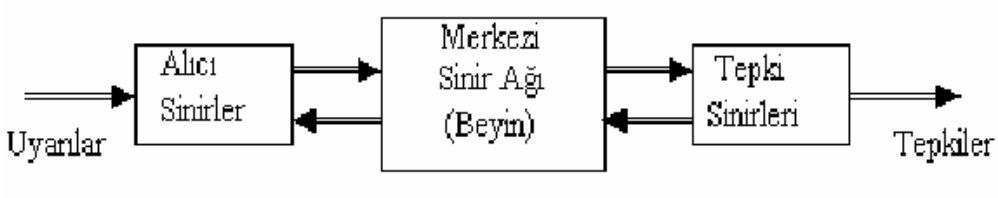


Şekil.2.1. Biyolojik sinir hücresi

Bir sinir hücresi hücre gövdesi, akson ve dentrit adı verilen bölümlerden oluşur. Hücreler birbirlerine dentrit adı verilen uzantılarla bağlanırlar. Bağlantı noktalarına sinaps denir. Bu bağlantılar aynı zamanda hücreler arasında iletişim kanalı olarak vazife görürler ve sinyalleri taşıma işlevini yerine getirirler. Sinyal taşıma ise elektrik yüklü iyonlar yardımıyla olur. Hücre, sinapslarda oluşan (+) ve (-) akımların sonucu olarak, aksonda bir sonuç gerilim oluşturur. Dengedeki bir nöronun hücre zarında görülen gerilim -85mV olmakla birlikte, eğer -40mV'luk eşik gerilimi aşılır ve böylece hücre zarını depolerize edecek yönde stoplazmaya akım aşılanırsa, hücre aktif moda geçer ve axon tarafından bir elektriksel darbe üretilir.

Biyolojik sinir ağları beynimizde bulunan bir çok sayıda sinir hücresinin bir koleksiyonudur. Bir sinir ağı milyonlarca sinir hücresinin bir araya gelmesiyle oluşmaktadır. Beynimizde 10^{10} adet sinir hücresi ve bunlarında 6.10^{13}'ten fazla sayıda bağlantısının olduğu söylenmektedir. Biyolojik sinir ağlarının performansları küçümsenemeyecek kadar yüksek ve karmaşık olayları kolaylıkla işleyebilecek yetenektedir. Yapay sinir ağları ile bu yeteneğin bilgisayarlara ve makinelere kazandırılması amaçlanmaktadır.

Biyolojik Sinir Sistemi: Biyolojik sinir sistemi, merkezinde sürekli olarak bilgiyi alan, yorumlayan ve uygun bir karar üreten beynin (merkezi sinir ağı) bulunduğu üç katmanlı bir sistem olarak açıklanır. Alıcı sinirler (receptor) organizma içerisinden yada dış ortamlardan algıladıkları uyarıları, beyne bilgi ileten elektriksel sinyallere dönüştürür. Tepki sinirleri (effector) ise; beynin ürettiği elektriksel darbeleri organizma çıktısı olarak uygun tepkilere dönüştürür. Şekil.2.2.'de bir sinir sisteminin blok gösterimi verilmiştir. Merkezi sinir ağında bilgiler, alıcı ve tepki sinyalleri arasında ileri ve geri besleme yönünde değerlendirirlerek uygun tepkiler üretilir. Bu yönüyle biyolojik sinir sistemi, kapalı çevrim denetim sistemi karakteristiklerini taşır.

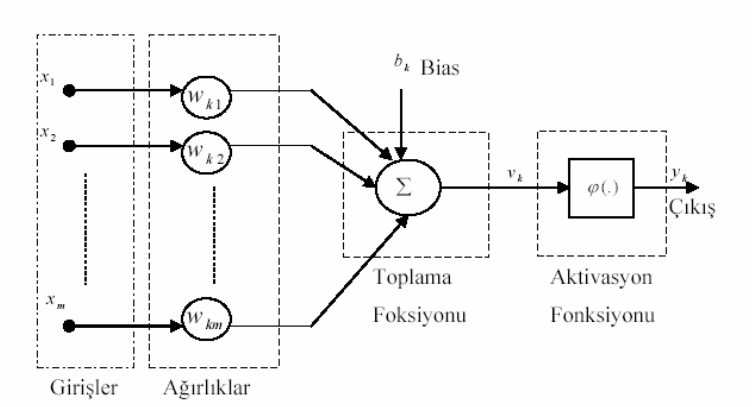


Şekil.2.2. Biyolojik sinir sisteminin blok gösterimi

Yapay sinir ağları, insan beyninin çalışma prensibi örnek alınarak geliştirilmeye çalışılmıştır ve aralarında yapısal benzerlikler bulunmaktadır.

2.2 Yapay Sinir Hücresi

Biyolojik sinir ağlarının hücreleri olduğu gibi yapay sinir ağlarının da hücreleri vardır. Sinir hücresinin biyolojik yapısı hakkında tüm bu veriler göz önünde bulundurularak, yapay sinir hücresi, eşik değerine sahip bir işlemci ünitesi olarak modellenebilir. Yapay sinir ağını oluşturan yapay sinir hücreleri tek başlarına ele alındıklarında çok basit işlevleri olan işlemcilerdir. Şekil.2.3'de bir yapay sinir hücresi görülmektedir.



Şekil.2.3. Bir yapay sinir hücresinin matematiksel modeli

Şekilden yola çıkarak k. yapay sinir hücresi için deklemler yazılırsa;

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

$$v_k = u_k + b_k \tag{2.2}$$

$$y_k = \varphi(v_k) \tag{2.3}$$

Her bir girdideki değişim, YSH çıkışında bnir değişime neden olmakta ve bu değişimin genliği, girişin etki derecesini belirleyen ağırlığa, toplayıcının eşik değerine ve aktivasyon fonksiyonuna bağlıdır. Yukarıdaki modelden de gözleneceği üzere eşik değeri girdilerden bağımsız olmasından dolayı, bütün girişler sıfır olsa bile çıkışta bir φ(0) değeri gözlenir.

Yapay sinir hücresi beş elemandan oluşur. Bunlar:

Girdiler: Yapay sinir ağına dış dünyadan gelen bilgilerdir. Ancak bazı durumlarda bu bilgiler dış dünyadan gelmez, ağ geri besleme yapısıyla çıktılarını girişlerine tekrar verebilir.

Ağırlıklar: Ağırlıklar gelen girdilerin hücre üzerindeki etkisini belirler. Burada ağırlığın küçük olması gelen girdinin önemsiz olduğu anlamına gelmez, hatta ağırlığın sıfır olması dahi gelen girdiyi en önemli veri haline getirebilir. Ağırlıklar değişken veya sabit değerler olabilir.

Toplama Fonksiyonu: Bu fonksiyon hücreye gelen net girdiyi hesaplar. Genellikle ağırlıklı toplam fonksiyonu kullanılır. Bu da ağırlıklar ile o ağırlığa ilişkin girdinin çarpılarak toplamının alınması şeklinde formülize edilir.

$$\mathbf{NET:} \sum_{i}^{n} G_i \bullet A_i \tag{2.4}$$

Burada n girdi sayısı, A girdilerin ağırlıkları, G ise girdileri göstermektedir. Ancak ağırlıklı toplam yerine daha farklı fonksiyonlar da kullanılabilir. Bu tasarımcının problemi çözmedeki ön görüsüne kalmıştır. Toplama fonksiyonu tasarıma göre işlemci elemanların hepsinde farklı da seçilebilir veya bazı işlemci grupları arasında bir toplama fonksiyonu kullanılırken, diğerlerinde farklı bir toplam fonksiyonu kullanılıyor olabilir.

Aktivasyon fonksiyonu: Bu fonksiyon hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Toplama fonksiyonunda olduğu gibi aktivasyon

fonksiyonunda da çıktıyı hesaplamak için değişik fonksiyonlar uygulanabilir. Bunlardan en yaygın kullanılanı sigmoid fonksiyonudur. Toplama fonksiyonunda olduğu gibi her işlemci elemanının aynı fonksiyonu kullanabileceği gibi farklı aktivasyon fonksiyonları kullanmaları da mümkündür. Hangi aktivasyon fonksiyonunun problemi çözmede daha iyi bir çözüm sağlayacağı sorusu halen kesin bir cevaba sahip değildir, deneme yanılma yöntemi izlenmektedir.

Doğrusal olmayan aktivasyon fonksiyonunun kullanılmasının sonucu nöron modeline doğrusal olmama özelliği kazandırmaktadır. Bundan dolayı ağda, çıkışlar ve girişler arasında daha kuvvetli bir nonlineer ilişki sağlama özelliği elde edilir. doğrusal olmayan bir fonksiyon kullanılmaması durumunda, yapay nöron doğrusal bir sistemi temsil eder. Bu tip yapay nöronlar doğrusal olmayan sistem eşleşmesinde kullanılmaz ve doğrusal olmayan hesapları yerine getiremez. Doğrusal aktivasyon fonksiyonları gürültüyü bastıramamaktadır. Bu sebepten ağ sağlam bir doğruluk yapısında değildir. Şu da belirtilmelidir ki doğrusal aktivasyon fonksiyonu içeren nöron modeline sahip ağlar da kullanılmaktadır. Fakat bunlar sadece doğrusal sistem modellenmesinde kullanılır. Doğrusal aktivasyon fonksiyonlu sinir modeli literatürde Widrow-Hoff modeli olarak bilinmektedir.

Hücrenin Çıktısı: Aktivasyon fonksiyonu tarafından belirlenen çıktı değerleridir. İşlemci elemanının birden fazla çıktısı olmasına rağmen tek bir çıktı değeri vardır ve bu diğer bir işlemci elemanına girdi olarak gelmektedir.

2.3 Yapay Sinir Ağları, Özellikleri ve Yapısı

Yapay sinir ağlarının çalışma prensibi bir girdi setini alarak onları çıktı setine çevirmek olarak tanımlanabilir. Bunun için ağın kendisine gösterilen girdiler için doğru çıktılar üretmesi gereklidir. Ağa gösterilecek girdiler bir vektör haline getirilir, bu vektör ağa gösterilir ve ağ çıktı vektörünü oluşturur. Ağın parametreleri doğru çıktıyı oluşturacak biçimde düzenlenir. Girdi vektörü, bir parmak izini gösteren nümerik değerler, borsada bir kağıdın haftalık satış miktarına ait nümerik değerler, bir resmin mavi tonları gibi nümerik değerlerden oluşabilir. Benzer şekilde çıktı vektörü de girdi vektörü de girdi vektörünün sınıfından olabilir. Girdi ve çıktı vektörlerinin tasarımı tasarımcı tarafından belirlenir ve girdiler belirlenen formatta toplanarak eğitim sırasında ağa gösterilir.

Yapay sinir ağlarının genel özelliklerinden bahsedecek olursak; öncelikle yapay sinir ağları makine öğrenmesi gerçekleştirirler, olayları öğrenerek benzer olaylar karşısında benzer kararlar vermeye çalışırlar. Programlama yöntemi, geleneksel programlama yöntemlerinden oldukça farklıdır. Bilgiler ağın üzerinde saklıdır ve ortaya çıkarılması zordur. Yapay sinir ağları örnekleri kullanarak öğrenirler, bu yüzden yapay sinir ağının performansı sunulan örneklerin problemi anlatma yeteneğine bağlıdır. Ağa olay bütün yönleri ile gösterilmez ve ilgili örnekler sunulmaz ise başarılı sonuçlar elde edilemez. Aynı yapıda iki ağı eğitirken birinde iyi seçilmiş örnekler, diğerinde ise problemi iyi tanımlamayan örnekler kullanılırsa, iki ağın performansı birbirinden çok farklı olur. Yapay sinir ağları, şekil (örüntü) ilişkilendirme ve sınıflandırma yapabilirler, eksik bilgiler ile verilen bir paterni tamamlayabilirler. Yapay sinir ağlarının kendilerine gösterilen yeni durumlara adapte olabilmesi mümkündür. Yapay sinir ağlarının hataya karşı toleranslı olmaları, dereceli bozulma göstermelerine sebep olur. Bir ağ zaman içinde dereceli olarak bozulur.

Yapay sinir ağları, insan beyninin çalışma prensibi örnek alınarak geliştirilmeye çalışılmıştır. Bu yüzden aralarında yapısal benzerlikler bulunmaktadır. Bu benzerlikler Çizelge.2.1'de verilmiştir.

Çizelge.2.1 Sinir sistemi ile YSA benzerlikleri

SİNİR SİSTEMİ	**YSA SİSTEMİ**
Nöron	İşlem elemanı
Dentrit	Toplama fonksiyonu
Hücre gövdesi	Transfer fonksiyonu
Aksonlar	Eleman çıkışı
Sinapslar	Ağırlıklar

Yapay sinir ağları, insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri herhangi bir yardım almaksızın otomatik olarak gerçekleştirmek amacı ile geliştirilen bilgi işleme modelleridir. İnsan beyninin fonksiyonel özelliklerine benzer şekilde, öğrenme, ilişkilendirme, sınıflandırma, genelleme, özellik belirleme ve optimizasyon gibi konularda başarılı bir şekilde uygulanmaktadırlar. Bu yetenekleri geleneksel programlama yöntemleri ile gerçekleştirmek oldukça zor veya mümkün değildir. Literatürde 100'den fazla sayıda yapay sinir ağı modeli vardır (Haton, 2003).

2.4 Yapay Sinir Ağlarının Sınıflandırılması

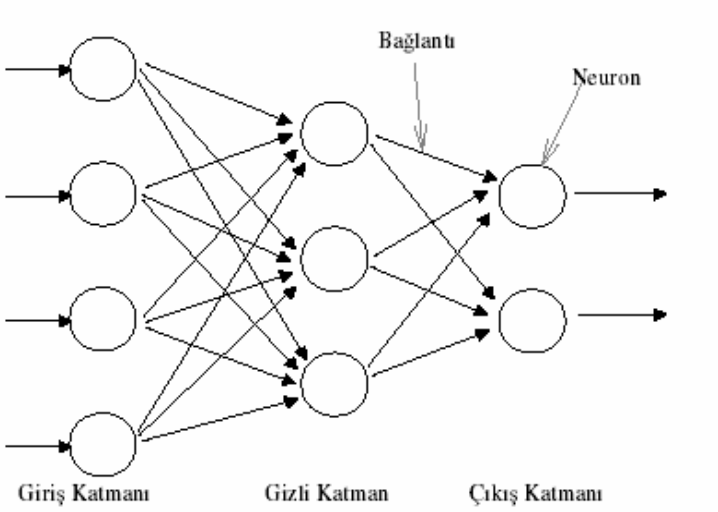
Biyolojik sinir hücreleri nasıl bir araya gelip sinir sistemini oluşturuyorsa, yapay sinir hücreleri de bir araya gelip yapay sinir ağını oluştururlar. Yapay sinir ağları üç katmandan oluşur:

Girdi Katmanı: Bu katmandaki elemanlar dış dünyadan alınan bilgileri ara katmanlara iletmekle sorumludurlar.

Ara Katmanlar: Girdi katmanından gelen bilgileri işleyip çıktı katmanına aktarmaktan sorumludurlar.

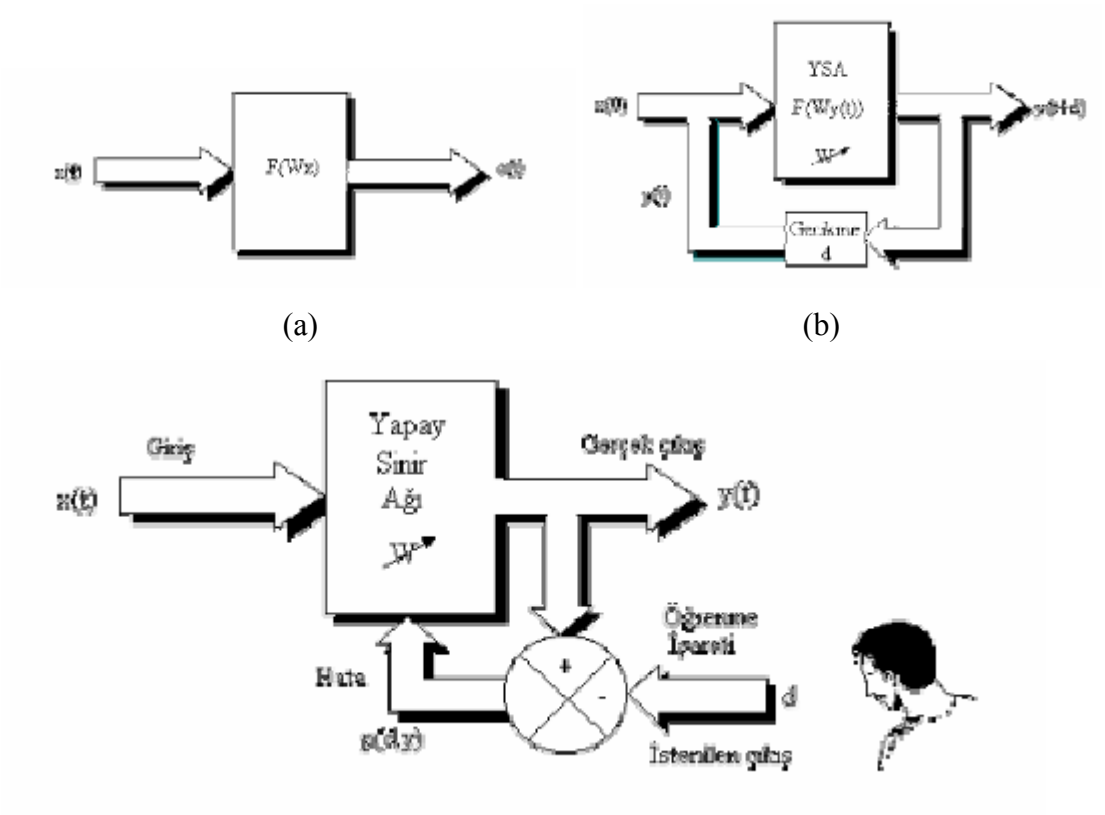
Çıktı Katmanı: Bu katmandaki işlemci elemanları ara katmandan gelen bilgileri işleyerek çıktı olarak dış dünyaya gönderirler.

Şekil.2.4'de bir yapay sinir ağı modeli görülmektedir.



Şekil.2.4 Çok katmanlı bir yapay sinir ağı mimarisinin gösterimi

Yapay sinir ağları mimari yapılarına göre iki sınıfa ayrılır. Bunlar geri beslemeli (Elman, Jordan) ve ileri beslemeli (MLP, LVQ) olarak adlandırılırlar.



(a) (b)

(c)

Şekil.2.5 Öğrenme yaklaşımlarına göre yapay sinir ağları (a) İleri beslemeli ağ (b) Geri beslemeli ağ (c) Destekleyici öğrenme gerçekleştiren ağ

Öğrenme yaklaşımlarına göre ise yapay sinir ağları; eğiticili öğrenme, eğiticisiz öğrenme ve destekleyici öğrenme olarak sınıflandırılır.

2.5 Yapay Sinir Ağları Kavramını Çekici Kılan Özellikler

Yapay sinir ağlarının hesaplama ve işleme gücünü, paralel dağıtık yapısından, öğrenebilme ve genelleme yeteneğinden aldığı söylenebilir. Genelleme, eğitim ya da öğrenme sürecinde karşılaşılmayan girişler için de yapay sinir ağlarının uygun tepkiler üretebilmesi olarak tanımlanır. Bu üstün özellikleri, yapay sinir ağlarının karmaşık problemleri çözebilme yeteneğini gösterir.

Doğrusal Olmama: YSA'ların yapıları gereği doğrusal ağlar olduğu gibi daha çok doğrusal olmayan yönleriyle öne çıkmışlardır. Ağın ya da temel işlem elemanı olan hücrenin, doğrusallığı aktivasyon fonksiyonu ile belirlenir. Bu özelliği ile YSA, özellikle doğrusal olmayan karmaşık problemlerin çözümünde önemli bir araç durumuna gelmiştir.

Yeni Bilgileri Öğrenme: YSA'nın arzu edilen davranışı gösterebilmesi için amaca uygun olarak tasarlanması gerekir. Bu durum, hücreler arasında doğru bağlantılar yapılması ve bağlantıların uygun ağırlıklara sahip olması gerektiğini ifade eder. YSA'nın karmaşık ve lineer olmayan yapısı nedeniyle bağlantılar ve ağırlıklar önceden ayarlı olarak verilemez yada tasarlanamaz. Genellikle ağırlıklar, rasgele yada sabit bir değerde seçilir. YSA, istenen davranışı gösterecek şekilde ilgilendiği problemden aldığı eğitim örneklerini kullanarak problemi öğrenmelidir. Belli bir hata kriterine ve öğrenme algoritmasına göre, ağırlıkların yenilenerek artık değişmediği yada fazla değişmediği durumda öğrenmenin gerçekleştiği söylenebilir.

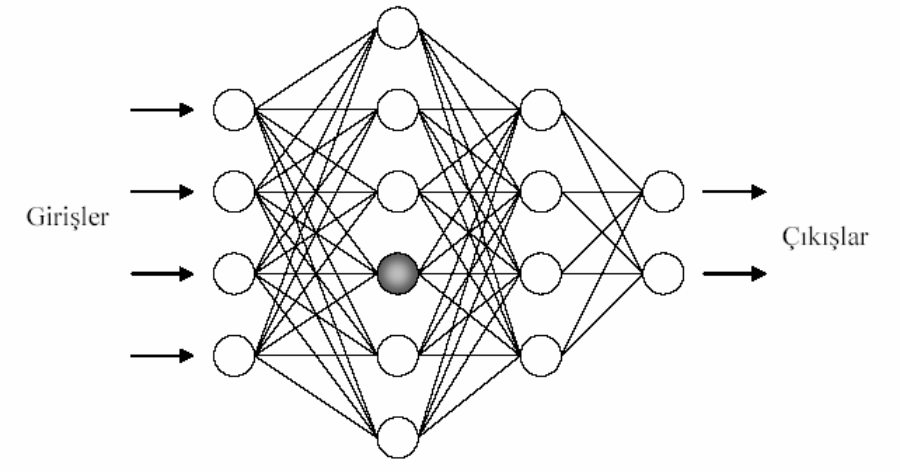
Uyarlanabilirlik: YSA, ilgilendiği problemdeki değişikliklere göre ağırlıklarını ayarlar. Yani, belirli bir problemi çözmek amacıyla eğitilen YSA, problemdeki değişimlere göre tekrar eğitilebilir, değişimler devamlı ise gerçek zamanda da eğitme devam edebilir. Bu özelliği ile YSA, sinyal işleme, uyarlamalı sistem tanıma ve kontrol gibi alanlarda etkin olarak kullanılırlar. Uyarlanabilir sistemler çok kısa zaman aralıklarında ağırlıklarını birçok değişikliğe uğratırlar, bu sebeple bozucu etkilere cevap vermeye yatkındırlar. Bu ise sistem performansını büyük ölçüde etkiler. Uyumluluk özelliklerini tam anlamıyla kullanmak için, sistemin alacağı ilk ağırlık değerleri çevreden gelen istenilen değişimlere cevap verecek kadar küçük ve sistemin bozucu etkilerinden etkilenmemesini sağlayacak kadar büyük olmalıdır.

Genelleme: Ağ yapısının, eğitim esnasında kullanılan nümerik bilgilerden eşleştirmeyi betimleyen kaba özellikleri çıkarsaması ve böylelikle eğitim sırasında kullanılmayan girdiler için de anlamlı yanıtlar üretebilmesidir. YSA, ilgilendiği problemi öğrendikten sonra eğitim sırasında karşılaşmadığı test örnekleri için de arzu edilen tepkiyi üretebilir. Örneğin, karakter tanıma amacıyla eğitilmiş bir YSA, bozuk karakter girişlerinde de doğru karakterleri verebilir yada bir sistemin eğitilmiş YSA modeli, eğitim sürecinde verilmeyen giriş sinyalleri için de sistemle aynı davranışı gösterebilir.

Paralellik: Sistemin paralelliği ve toplamsal işlevin yapısal olarak dağıtılmışlığıdır. Diğer bir deyişle bir çok nöron eş zamanlı olarak çalışır ve karmaşık bir işlev bir çok küçük nöron aktivitesinin bir araya gelmesinde oluşur.

Hata Toleransı: YSA, çok sayıda hücrenin çeşitli şekillerde bağlanmasından oluştuğundan paralel dağıtık bir yapıya sahiptir ve ağın sahip olduğu bilgi, ağdaki bütün bağlantılar üzerine

dağıtılmış durumdadır. Bu nedenle, eğitilmiş bir YSA'nın bazı bağlantılarının hatta bazı hücrelerinin etkisiz hale gelmesi, ağın doğru bilgi üretmesini önemli ölçüde etkilemez. Bu nedenle, geleneksel yöntemlere göre hatayı indirgeme yetenekleri oldukça yüksektir. Koyu nöronun işlev dışı kalması, görev paylaşımından dolayı başarımı dikkate değer ölçüde etkilemeyecektir.



Şekil.2.6 Tek bir nöronun çıkıştaki hataya etkisi

Donanım ve Hız: YSA, paralel yapısı nedeniyle büyük ölçekli entegre devre (VLSI) teknolojisi ile gerçeklenebilir. Bu özellik, YSA'nın hızlı bilgi işleme yeteneğini arttırır ve gerçek zamanlı uygulamalarda kullanılabilmesini mümkün kılar.

Analiz ve Tasarım Kolaylığı:YSA'nın temel işlem elemanı olan hücrenin yapısı ve modeli, bütün YSA yapılarında yaklaşık olarak aynıdır. Dolayısıyla, YSA'nın farklı uygulama alanlarındaki yapıları da standart yapıdaki bu hücrelerden oluşacaktır. Bu nedenle farklı uygulama alanlarında kullanılan farklı mimarilerdeki YSA'lar benzer öğrenme algoritmalarını ve teorilerini paylaşabilirler. Bu özellik, problemlerin YSA ile çözümünde kolaylık getirecektir.

2.6 Yapay Sinir Ağı Modelleri

Bir yapay sinir ağının öğrenmesi istenilen olayların girdi ve çıktıları arasındaki ilişkiler doğrusal olmayan ilişkiler olursa (örneğin XOR problemi) tek katmanlı bir yapay sinir ağı bu probleme çözüm üretemez. XOR problemini çözmek amacıyla yapılan çalışmalar sonucunda

çok katmanlı yapay sinir ağları geliştirilmiştir. Rumelhart ve arkadaşları tarafından geliştirilen bu modele hata yayma modeli veya geriye yayılım modeli de denilmektedir.

İleri beslemeli YSA'da, hücreler katmanlar şeklinde düzenlenir ve bir katmandaki hücrelerin çıkışları bir sonraki katmana ağırlıklar üzerinden giriş olarak verilir. Giriş katmanı, dış ortamdan aldığı bilgileri hiçbir değişikliğe uğratmadan orta (gizli-hidden) katmandaki hücrelere iletirler. Ağa sunulan giriş, orta ve çıkış katmanında işlenerek ağ çıkışı belirlenir. Bu yapısı ile ileri beslemeli ağlar doğrusal olmayan statik bir işlevi gerçekleştirir. İleri beslemeli 3 katmanlı YSA'nın, orta katmanında yeterli sayıda hücre olmak kaydıyla, herhangi bir sürekli fonksiyona istenilen doğrulukta yakınsanabileceği gösterilmiştir. Nöronlar, genellikle katmanlara ayrılmış olmaktadır. İşaretler, giriş katmanından çıkış katmanına doğru tek yönlü bağlantılarla iletilir. Nöronlar bir katmandan diğer bir katmana bağlantı kurarlarken, aynı katman içerisinde bağlantıları bulunamaz. İleri beslemeli ağlara örnek olarak çok katmanlı perceptron (MLP) ve LVQ (Learning Vector Quantization) ağları verilebilir.

Bir geri beslemeli sinir ağı, çıkış ve ara katmanlardaki çıkışların, giriş birimlerine veya önceki ara katmanlara geri belendiği bir ağ yapısıdır. Böylece girişler hem ileri yönde hem de geri yönde aktarılmış olur. bu çeşit sinir ağlarının dinamik hafızaları vardır. Bu ağlarda bir andaki çıkış, hem o andaki hem de önceki girişleri yansıtır. Bundan dolayı özellikle önceden tahmin uygulamaları için uygundurlar. Bu ağlar çeşitli tipteki zaman serilerinin tahmininde oldukça başarı sağlamışlardır. Geri beslemeli yapay sinir ağlarına örnek olarak Hopfield, SOM (Self Organizing Map), Elman ve Jordan ağları verilebilir.

2.7 Zeka

Zekanın tanımı kesin olarak yapılamamakla birlikte yıllardır herkes tarafından farklı yorumlanmıştır. Bir kaç kaynaktan alınan zeka kavramına ilişkin tanımlar aşağıdaki gibidir.

- İyi akıl yürütme , iyi hüküm verme ve kendini iyileştirme kapasitesidir. (BINET)
- Birey soyut düşünebildiği müddetçe zekidir. (TERMAN)
- Bireyin gayeli davranma, mantıklı düşünme ve çevresiyle ilişkilerinde etkili olma kapasitesinin tümüdür. (Wechsler)

2.8 Yapay Zeka

Yapay zeka terimi (İng: Artificial Intelligence, Kısa: AI) ise ilk olarak Standford Üniversitesi'nde profesör olan John McCarthy tarafından ortaya çıkarılmıştır. Yapay zeka kavramının Prof. John McCarthy tarafından yapılan tanımı şöyledir; "Zeki makineler özelliklede, zeki bilgisayar programları yapma bilimi ve mühendisliğidir. Benzer bir iş olan bilgisayarlar aracılığı ile insan zekasını anlamaya çalışmayla ilgili olmasına rağmen kendisini sadece biyolojik olarak gözlemlenebilen metotlar ile sınırlandırmaz." (Haton, 2003).

Yapay zeka'nın tanımını yapan kişiler, bu terimi kendi uygulama alanlarına hitap edişine göre tanımlamayı daha uygun görmüşlerdir. Yapay zekaya ilişkin yapılan bazı önemli tanımlamalar aşağıdaki gibidir.

- Yapay zeka yapay bir varlığın (genellikle bir bilgisayar) sergilediği zekadır.
- Yapay zeka bir makine ya da insan eliyle üretilmiş otonom bir sistem kullanarak insan zekasının benzetimini yapmaya çalışan bir araştırma alanıdır.
- Yapay zeka, insanın düşünme yöntemlerini analiz ederek bunların benzeri yapay yönergeleri geliştirmeye çalışan araştırma alanıdır.
- Yapay zeka, canlılarda (özellikle insanlarda) bulunan algılama, öğrenme, çoğul kavramları bağlama, düşünme, fikir yürütme, sorun çözme, iletişim kurma, çıkarım yapma ve karar verme gibi yüksek bilişsel fonksiyonları ve otonom davranışları sergilemesi beklenen yapay bir sistemdir (donanım ve yazılım).
- Bilgisayar bilimlerinin, bilgisayarlara insan zekasına niteliklerinin kazandırılması üzerinde çalışan bir dalıdır.
- Bilgisayar yazılımlarının, makinelere insanların zekalarını kullanarak yaptıkları işleri yapma yeteneği kazandırmaya yönelik olarak kullanılması. Önceki dönemlerde yapay zeka çalışmalarında insan psikolojisi göz ardı edilmiştir. Ama günümüzde bu yönelim beynin çalışmasına dair teorileri temel alan bağlantıcılığın (İng: Connectionism) gelişmesiyle değişmiştir. Bağlantıcılığın temelinde basit temel elemanlar (yada düğümler) arasında belirli bir biçimi olan bilgi parçalarının aktarılmasını da içeren ve öğrenmeyi modellemeye çalışan karmaşık fonksiyonların kullanılması vardır (Haton, 2003).

Yapay zeka çalışmalarında hedeflenen amaçlar şunun gibidir:

- İnsan beyninin fonksiyonlarını modellerle anlamaya çalışmak.

- İnsanın zihinsel yeteneklerini, bilgi kazanma, öğrenme ve buluş yapmada uyguladıkları strateji ve metotları araştırmak.
- Bu metotları formül haline getirmek, bilgisayarlarda uygulamak.
- Bilgisayar kullanımını kolaylaştıracak arayüzler geliştirmek.
- Uzman sistemler ve genel bilgi sistemleri geliştirmek.
- Yapay zeka iş yardımcıları ve zeki robot timleri geliştirmek.
- Bilimsel araştırma ve buluşlar için araştırma yardımcıları geliştirmek.

Yapay zeka kullanan akıllı bir makinenin kendisini ve çevresini doğru algılaması ve göstermesi, bu bilgileri kodlaması ve kodlanmış bilgiyi çözmesi, mantıksal çıkarım uygulaması, ve bilgiye kolay erilim için sıralaması gerekir. Tüm bu etkinlikler, yapay zekanın elemanlarıdır. Öğrenme, yapay zeka için çok önemli bir başka etkinliktir. Özel bilgilerden genel bilgileri çıkarım zekanın temel kavramlarından birisidir. Bu nedenle, mekatronik sistemlerin daha zeki makineler olarak tanımlanabilmeleri için öğrenme birimlerinin olması da istenmektedir. İleriye yönelik tahmin yapabilme, zeki makinelerden beklenen diğer bir özelliktir. Yine zekanın bir parçası olarak, hatalı yapılan işlerin farkına varılması ve düzeltilmesi de anlaşılmalıdır. Merak etme ve yaratıcılık, zeki davranışlar için çok önemli iki temel kavram olmakla birlikte henüz makine düzeyinde bu kavramların uygulaması görülmemektedir.

2.9 Yapay Zeka Çalışmalarının Tarih İçinde Gelişimi

- 1942 yılı YSA' nın gelişiminin başlangıç yılı olarak kabul edilmektedir. Bu tarihte sinirbilimcisi (neuroscientist) olan McCulloch ve matematikçi Pitts, ilk hücre modelini geliştirmişlerdir. Bu çalışmada biyolojik sinir hücresinden etkilenerek temel lojik işlemleri yapabilen basit bir işlem birimi modeli gerçekleştirilmiştir. Bunun yanında birkaç hücrenin ara bağlaşımını incelemişlerdir. Bundan sonra 1949 yılında Hebb, hücre bağlantılarını ayarlamak için ilk öğrenme kuralını önerdi (Öztemel, 2006).
- 1958'de Rosenblatt, algılayıcı (perceptron) modelini ve öğrenme kuralını geliştirerek, bugün kullanılan kuralların temelini koydu.
- 1960-1962 yılında Widrow ve Hoff tarafından ADALINE'lar ve LMS kuralı geliştirildi.
- 1969 yılında , Minsky ve Papert, algılayıcının kesin analizi yaptı ve algılayıcının karmaşık lojik fonksiyonlar için kullanılamayacağını ispatladılar. Bunun üzerine yapay sinir

ağları üzerine yapılan çalışmalar hemen hemen durma noktasına gelmiştir. 1960 yılının ortalarından, 1980 yılının başına kadar büyük bir durgunluk dönemi geçmiştir. Çalışmaları büyük ölçüde azaltan bu durgunluğun en önemli faktörlerinden birisi , YSA' nın bilgi işlemede alternatifi olan günümüz sayısal bilgisayarlarının yarı iletken teknolojisi ile yoğun, büyük çapta, ucuz ve güvenilir olarak gerçeklenme imkanı bulmasıdır. Seri olarak çalışan hızlı birimlerden oluşmuş sayısal bilgisayarlar, aritmetik işlemlerde yüksek hız, kapasite ve güvenirlik sağlamışlardır. Ancak tüm bunların yanında bazı bilim adamları (Gross, Amari, Fukushima, Kohonen, Taylor,...) çalışmalarına devam etmişlerdir.

- 1982 yılında Hopfield YSA' nın birçok problemi çözebilecek kabiliyeti olduğunu göstermiştir. Optimizasyon gibi teknik problemleri çözmek için doğrusal olmayan Hopfield ağını geliştirdi.
- 1982-1984 yılında Kohonen öz yinelemeli haritalı (self-organizing map) tanımladı. Kendi adında anılan eğiticisiz öğrenen bir ağ geliştirdi.
- 1986 yılında Rumelhart geriye yayılımı tekrar meydana çıkarttı.
- 1988 yılında Chua ve Yang hücresel sinir ağlarını geliştirdiler.

Yapay sinir ağlarındaki yeniden dirilişe donanım teknolojisindeki gelişmelerin katkısı büyük olmuştur. Bilgisayarlar boyut olarak küçülmüş, fakat bellek ve yetenek olarak büyümüş ve gelişmişlerdir. Optik ve dijital teknolojideki gelişmeler ve VLSI teknolojisi gibi yeni teknolojilerin geliştirilmesi bilgisayarların hızlarını arttırmış ve yapay sinir ağlarının kullanımını kolaylaştırmıştır.

Günümüzde yapay sinir ağları artık teorik ve laboratuar çalışmaları olmaktan çıkmış ve günlük hayatta kullanılan sistemler oluşturmaya ve pratik olarak insanlara faydalı olmaya başlamıştır.

2.10 Yapay Sinir Ağlarının Eğitilmesi

İşlemci elemanlarının ağırlık değerlerinin saptanması işlemine ağın eğitilmesi denir. Başlangıçta ağırlık değerleri rasgele atanır. Daha sonra ağa ilk girdiler verilir. Sonuçta oluşan çıktıya göre ağa yeniden örnekler verilir. Böylece defalarca verilen örnekler sayesinde ağda doğru bağlantı ağırlıklarının oluşması sağlanır. Test için ise daha önceden ağa gösterilmeyen örnekler kullanılır. Bu örnekler sayesinde ağ test edilir ve ağın performansı ölçülür. Eğitimde kullanılan örneklere *eğitim seti*, test de kullanılan örneklere de *test seti* denir.

Yapay sinir ağlarının öğrenme sürecinde, tıpkı dış ortamdan gözle veya vücudun diğer organlarıyla uyarıların alınması gibi dış ortamdan girişler alınır, bu girişlerin beyin merkezine iletilerek burada değerlendirilip tepki verilmesi gibi yapay sinir ağında da aktivasyon fonksiyonundan geçirilerek bir tepki çıkışı üretilir. Bu çıkış yine tecrübeyle verilen çıkışla karşılaştırılarak hata bulunur. Çeşitli öğrenme algoritmalarıyla hata azaltılıp gerçek çıkışa yaklaşılmaya çalışılır. Bu çalışma süresince yenilenen yapay sinir ağının ağırlıklarıdır. Ağırlıklar her bir çevrimde yenilenerek amaca ulaşılmaya çalışılır. Amaca ulaşmanın veya yaklaşmanın ölçüsü de yine dışarıdan verilen bir değerdir. Eğer yapay sinir ağı verilen giriş-çıkış çiftleriyle amaca ulaşmış ise ağırlık değerleri saklanır. Ağırlıkların sürekli yenilenip istenilen sonuca ulaşılana kadar geçen zamana öğrenme adı verilir. Yapay sinir ağı öğrendikten sonra daha önce verilmeyen girişler verilip, sinir ağı çıkışıyla gerçek çıkışı yaklaşımı incelenir. Eğer yeni verilen örneklere de doğru yaklaşıyorsa sinir ağı işi öğrenmiş demektir. Sinir ağına verilen örnek sayısı optimum değerden fazla ise sinir ağı işi öğrenmemiş ezberlemiştir. Genelde eldeki örneklerin yüzde sekseni ağa verilip ağ eğitilir, daha sonra geri kalan yüzde yirmilik kısım verilip ağın davranışı incelenir diğer bir deyişle ağ böylece test edilir.

2.11 Yapay Sinir Ağlarından Yararlanmanın Avantajları ve Dezavantajları

Yapay sinir ağlarını kullanmanın getirdiği bazı avantajlar aşağıdaki gibi sıralanabilir;

- Yapay sinir ağları ile matematiksel olarak formülize edilmesi mümkün olmayan karmaşık problemler hızlı ve rahat modellenerek çözülebilir.
- Modeli hazırlamak için çok detaylı önbilgiye ihtiyaç yoktur sadece olayla ilgili örneklere ihtiyaç vardır. Örnekleri toplamak ise çoğu zaman ön bilgileri toplamaktan daha kolaydır.
- Daha önceki örneklere bakarak, hiç görmediği durumlar karşısında mantıklı sonuçlar üretebilir.
- Pratik ve düşük maliyetli çözümler üretmektedir.
- Paralel çalışabilmeleri, gerçek zamanlı çalışmalarına olanak verir.

Yapay sinir ağlarını kullanmak her durumda avantajlı değildir. Bu uygulamaların getirdiği bazı dezavantajlar aşağıdaki gibidir.

- Yapay sinir ağlarının oluşturulmasında topolojisinin seçilmesinde herhangi bir kurallar seti yoktur. Kullanıcının deneyimlerine bağlıdır.

- Aynı problem için uygun örnekler seçilmezse ağın gösterdiği performans değişir. Yapay sinir ağına hangi örnekleri göstermenin uygun olacağı ise yine kullanıcının tecrübelerine bağlıdır.
- Sonuçların optimum sonuçlar olduğuna dair bir garanti verilemez.
- Probleme ilişkin problemi doğru temsil eden örneklerin olmaması durumunda ağın doğru sonuçlar vermesi beklenemez.

2.12 Bulanık Mantık ve Yapay Zeka

İnsan beyninin nasıl çalıştığı ve fonksiyonları uzun yıllar araştırılmıştır.1980 yılında beyinin fonksiyonları hakkında bilgi veren ilk eser yayınlanmıştır. Bu yıllardan sonra çalışmalar mühendislik alanlarına kaydırılmaya başlanmış ve günümüzdeki yapay sinir ağı temelleri oluşturulmuştur. Beynin üstün özellikleri, bilim adamlarını üzerinde çalışmaya zorlamış ve beynin nöro-fiziksel yapısından esinlenerek matematiksel modeli çıkarılmaya çalışılmıştır. Beynin bütün davranışlarını tam olarak modelleyebilmek için fiziksel bileşenlerinin doğru olarak modellenmesi gerektiği düşüncesi ile çeşitli yapay hücre ve ağ modelleri geliştirilmiştir. YSA kavramı beynin çalışma ilkelerinin sayısal bilgisayarlar üzerinde taklit edilmesi fikri ile ortaya çıkmış. İlk çalışmalar beyni oluşturan biyolojik hücrelerin yada literatürdeki ismiyle nöronların matematiksel olarak modellenmesi üzerine yoğunlaşmıştır. Bu çalışmaların ortaya çıkardığı bulgular, her bir yapay sinir hücresinin komşu yapay sinir hücresinden bazı bilgiler aldığı ve bu bilgilerin biyolojik sinir hücresi dinamiğinin öngördüğü biçimde bir çıktıya dönüştürüldüğü seklindeydi. Bu gün YSA olarak adlandırılan alan, birçok yapay sinir hücresinin belirli biçimlerde bir araya getirilip bir işlevi gerçeklenmesini yapısal olduğu kadar matematiksel ve felsefi sorulara yanıt arayan bir bilim dalı olmuştur.

Bulanık mantık, denetim ve bilgi süreçlerinin birçoğu için güçlü bir problem çözme yöntemidir ve kesin olmayan bulanık bilgiden dikkate alınacak kadar basit bir şekilde kesin sonuçlar elde edilmesine olanak sağlar. (Zadeh, 1965) Yapay sinir ağlarının en önemli özelliği öğrenebilme yeteneğidir. Bulanık mantık ve yapay sinir ağlarının birbirlerini tamamlayıcı özelliklerinden faydalanarak birçok uygulama gerçekleştirilmiştir. Model referans adaptif denetim yapısı, lineer olmayan özellikleri nedeniyle matematik modeli tam olarak kurulamayan sistemlerin, matematik modeli bilinen bir sistemin davranışını izlemesi esasına dayanır. (Haton, 2003)

Robot dinamiğindeki belirsizliklerin kaynağı sürtünme özellikleri, eylemsizlik momenti gibi zamanla değişen sistem parametreleri, dışarıdan robota etkiyen kuvvetlerin, robotun taşıdığı yükün değiştiği durumlar, modelleme hataları ve diğer doğrusal olmayan bozucu etkiler olabilir. Sadece robot dinamiğine bağlı bir denetim algoritması kullanarak arzu edilen performansı yakalamak zordur. Öğrenebilen bir denetleyici kullanıldığında, hesaplanmış moment gibi yöntemleri kullanan geleneksel denetleyiciler için gerekli olan dinamik modelin ve model parametrelerinin hassas bir şekilde elde edilmesi zorunluluğu önemini yitirir. Öğrenebilen denetleyiciler robot hareket denetiminde her geçen gün daha yoğun kullanım alanı bulmaktadırlar. Robot denetiminde iyi sonuçlar veren model referans adaptif denetim; sisteminin adaptasyon yeteneğini, dolayısıyla denetim hassasiyetini, hızını ve esnekliğini arttırmak, en önemlisi sisteme öğrenebilme yeteneği kazandırmak için, hataya bağlı bir fonksiyonun çıkışlarını eğitim girişi kabul eden çok eksenli bulanık denetleyici önerilere bir örnektir.

2.13 Uzman Sistemler

Yapay zeka uygulamaları üzerine çalışan bilim adamları ilk zamanlar, her sorunu çözecek genel amaçlı bir program yerine belirli bir uzmanlık alanı altındaki bilgiyle donatılmış programlar kullanma fikrini öne sürdüler. Kısa sürede uzman sistemler adı verilen bir metodoloji gelişti.

Uzman Sistemlere ait ilk çalışmalar Standford Üniversitesi'nde yapılmaya başlanmıştır. Stanford Üniversitesi profesörlerinden Edward Feigenbaum Uzman Sistemlerin tanımını "bilgi ve çıkarım prosedürlerini kullanarak uzman bilgisi gerektiren zor problemleri çözen akıllı bilgisayar programları" şeklinde yapmıştır.

Burada "bilgi" önemli bir kelimedir. Çünkü Uzman Sistemler bünyelerinde var olan bilgi ile sonuca varırlar. Uzman Sistemler, yordamsal programlardan çok farklıdır. Yordamsal programlar, algoritmaya dayanan yani kesin sonucu olan programlardır. Fakat Uzman Sistemlerde ise kesin sonuç olmayabilir. Elimizde bilgi vardır ve sistemimiz elimizdeki bu bilgiyi kullanarak yorum yapar. Uzman Sistemler, problem çözmede uzman bilgisini yani insan bilgisini kullanırlar. Bu sistemlerde uzman bilgisi, bilgi yada kurallar kümesi olarak adlandırılır. Bu bilgi ve kurallar kümesi problem çözümünde ihtiyaç duyulursa kullanılır. Yordamsal programlar, bir problemi çözmede bilgiden ziyade basit algoritmalar kullanırlar.

Ayrıca az olsa da kullanılan bu bilgi program kodunun içine yerleştirilmesi gerekmektedir. Fakat burada bir problem önümüze çıkmaktadır. Şayet bilgimizde bir değişiklik olduğu zaman, değişen bilginin tekrar kodumuzun içine değişen haliyle yazılması ve programın tekrar derlenmesi gerekmektedir. Uzman Sistemler ise, ufak bilgi parçalarını bilgi tabanına toplarlar ve bunu uygun bir problemin değerlendirilip çözülmesinde kullanırlar. Farklı bir problemle karşılaşıldığında ise, yordamsal programlamanın aksine, bilgi tabanının sınırları dahilinde, tekrar programlamadan problem çözülebilir. Ayrıca yapılan işlemlerin gerekçelerini açıklaması, güven seviyeleri ile uğraşması ve kuşku gibi özellikler Uzman Sistemlerin yordamsal programlara göre artı özellikleridir.

2.14 Adaptasyon ve Öğrenme

Öğrenme süreci, verinin belleğe depolanmasından önceki karşılaştırma ve sınıflandırma sürecidir. Aynı zamanda algılardan gelen ham bilgi bu süreçte belleğin algılayabileceği formatlara dönüştürülür.

İnsanın uyum sürecinin başlayabilmesi için bilgiye ihtiyaç vardır. Bilgi toplama süreci algılarla başlar. İnsan, karşısına çıkan bir nesneyi öğrenirken, nesne hakkında tüm algıları ile bilgi toplamaya çalışır. Bu bilgiler belleğe depolanırken nesneyi işaret etmek üzere aralarında ilişkiler kurulur. Artık ona ait koku algılandığında ona ait olan görüntü ve ses de kendiliğinden bilince çağrılacaktır. Nesneye ait objektif veriler, yani ölçülebilir (nicel) bilgiler, kavram kütüphanesinin özel bölümüne diğer bilgilerle yine ilişkiler kurularak depolanır.Daha sonra ise Bilinç tarafından değerlendirme sonucu oluşturulan subjektif veriler yine kavram kütüphanesinin farklı bir bölümüne depolanacaktır. Bir nesnenin öğrenilmesi demek sadece nicel olarak sınıflandırılması değil,aynı zamanda bu nesnenin insanın türdeşleri tarafından nasıl kullanıldığı veya nasıl tanımlandığı gibi soyut bir takım verilerinde öğrenilmesini gerektirir. Bellek, veri kapasitesini unutma süreci ile korur. Yani, unutmak bir kayıp değil bir koruma mekanizmasıdır. Çünkü bellek sınırlı, öğrenilecek bilgi ise sınırsızdır. Bellek sık sık karşısına çıkan canlı veya cansız nesneler ile ilgili bilgileri korurken, kullanılmayan bilgileri unutmaya başlar. Örneğin ilk önce görüntülerde çözünürlük kayıpları başlar. Sesler ve kokular görsel verilerden daha çabuk unutulur. Sıkıştırılarak veri kaybına uğratılan bilgiler belleğin kullanılmayan bölümlerine doğru kaydırılır. Böylece yeni bilgiler için bellek veri tabanında yer açılır.

2.15 Makine Öğrenmesi

Simon'a göre ; öğrenme, "zaman içinde yeni bilgilerin keşfedilmesi yoluyla davranışların iyileştirilmesi süreci" olarak tanımlanır. Makine öğrenmesi ise bu öğrenme sürecinin bilgisayarlar tarafından gerçekleştirilmesidir. Burada zaman içinde iyileşme kavramına dikkat çekmek gerekmektedir. Bilgisayarın da insan gibi zaman içerisinde tecrübe kazanması istenmektedir. Diğer bir değişle makine öğrenmesi "bilgisayarın bir olayla ilgili bilgileri ve tecrübeleri öğrenerek gelecekte oluşacak benzeri olaylar hakkında kararlar verebilmesi ve problemlere çözümler üretebilmesidir". Bilgisayarın öğrenebilmesi ve tecrübe sahibi olabilmesi bilgisayarın ilgili olay hakkında bilgiler ile donatılmasına bağlıdır. Yapay sinir ağları yolu ile öğrenen bilgisayarların bilgiler ile donatılması, örnekler yolu ile sağlanmaktadır. Öğrenecek olan bilgisayar sistemleri önce bir örnek almakta ve bu örnekten bazı bilgileri öğrenmektedir. Daha sonra ikinci örneğe bakarak biraz daha bilgi edinmektedir. Bu işlemi öğrenilecek olayla ilgili bütün örnekleri defalarca gözden geçirerek tekrarlamak sonucunda olay ile ilgili genellemeler yapılabilmektedir. Bu olaya tecrübelerden öğrenmeye bir örnek olarak bakmak mümkündür.

Yapay zeka üzerinde çalışan bilim adamları önceleri gerekli bilgileri programlamak yolunu seçiyorlardı. Ancak bunun anlamsız bir uğraş olduğu çok geçmeden anlaşıldı. Çünkü robotun doğa içinde varolabilmesi için sınırsız bilgiye ihtiyaç vardı. Bu yüzden şu anda tüm yapay zeka projelerinde öğrenme öne çıkmış durumdadır. Sensörlerden gelen bilgilerin sınıflandırılması ve ilişkilendirilmesi en doğru seçenek olacaktır.

En genel olarak, öğrenme işleminde sistemin ağırlık vektörü aşağıdaki gibi değiştirilir. Ve her iterasyonda bu işlem tekrar edilerek ağırlıklar kararlı bir dengeye getirilir. Ağırlık değerlerinin artık çok fazla değişmediğinde ağın öğrendiğini söyleyebiliriz.

$$W_{kj}(n+1) = W_{kj}(n) + \Delta W_{kj}(n) \qquad (2.5)$$

Öğrenme süresinde, seçilen öğrenme yaklaşımına göre ağırlıklar değiştirilir. Ağırlık değişimi, öğrenmeyi ifade eder. YSA'da ağırlık değişimi yoksa öğrenme işlemi de durmuştur. Başlangıçta ağırlık değerleri rasgele atanabilir. YSA'lar kendilerine örnekler gösterildikçe, bu ağırlık değerlerini yukarıda belirtildiği gibi değiştirirler. Ağın doğru ağırlık değerlerine

ulaşması örneklerin temsil ettiği olay hakkında genellemeler yapabilme yeteneğine kavuşması demektir. Bu genelleştirme özelliğine kavuşması işlemine, “ağın öğrenmesi” denir.

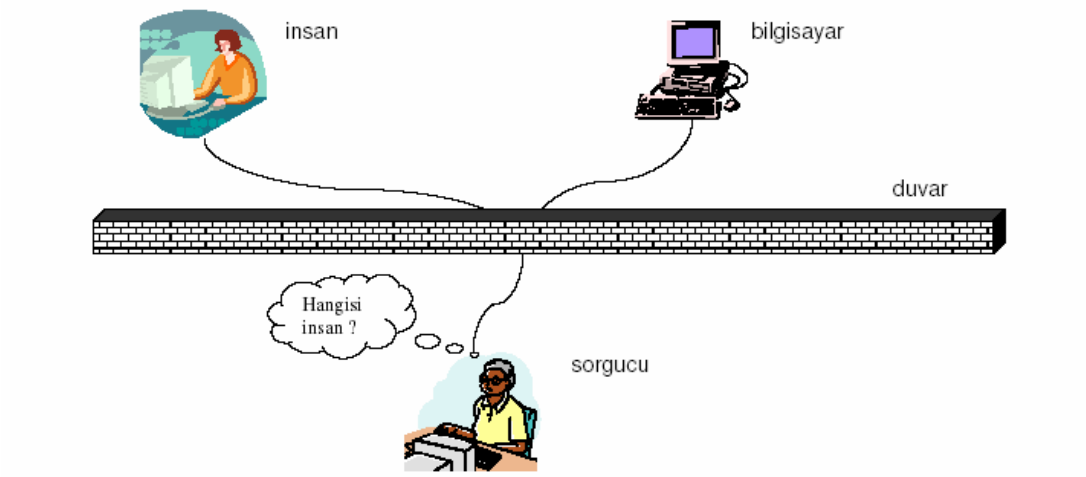
Bir yapay sinir ağı girdi setindeki değişiklikleri değerlendirerek öğrenir ve buna karşılık bir çıktı üretir. Öğrenme işlemi, benzer girdi setleri için aynı çıktıyı üretecek bir öğrenme algoritması ile gerçekleşir. Öğrenme girdilerin istatiksel özelliklerinin çıkarılarak benzer girdilerin gruplandırılmasını sağlayan bir işlemdir. Sinir yapılarına benzetilerek bulunan ağların eğitimi de normal bir canlının eğitimine benzemektedir. Sınıfların birbirinden ayrılması işlemi (dolayısıyla kendini geliştirmesi) öğrenme algoritması tarafından örnek kümeden alınan bilginin adım adım işlenmesi ile gerçeklenir. YSA kullanılarak makinelere öğrenme, genelleme yapma, sınıflandırma, tahmin yapma ve algılama gibi yetenekler kazandırılır.

Eğitim süreci sonunda yapay sinir ağında hesaplanan hatanın kabul edilebilir bir hata oranına inmesi beklenir. Ancak bir ağ gereğinden çok fazla girdi-çıktı ilişkisini öğrendiğinde, ağ verilerini ezberlemektedir (memorization). Bu durum genellikle gereğinden fazla gizli katman kullanıldığında ortaya çıkmaktadır. Ezberleme, genellemenin gerçekleşmediğini ve girdi-çıktı ilişkisinin düzgün olmadığını gösterir. Bu nedenle ezberleme, yapay sinir ağları için istenmeyen bir durum olarak karşımıza çıkmaktadır.

Turing Makinesi ve Turing Testi: Yapay zeka felsefesini ilk ortaya çıkaran kişi ünlü İngiliz mantık ve matematikçisi Alan Turing’dir. Dartmouth konferansından altı yıl önce , yani 1950 yılında Turing , Mind adlı felsefe dergisinin Ağustos sayısında “Computing Machinery and Intelligence” adlı bir makale yayınlamıştır. Bu makalede Turing “makineler düşünebilir mi?” sorusunu dikkatli bir felsefi tartışmaya açmış ve makineler düşünebilir iddiasına karşı olan itirazları reddetmiştir.

Alan Turing ayrıca Turing testi diye adlandırılan ve bir bilgisayarın veya başka bir sistemin insanlarla aynı zihinsel yetiye sahip olup olmadığını ölçen bir test geliştirmiştir. Genel anlamdan bu test bir uzmanın, makinenin performansı ile bir insanınkini ayırt edip edemeyeceğini ölçer. Eğer ayırt edemezse, makine insanlar kadar zihinsel yetiye sahip demektir. Bu testte bir insan ve bir bilgisayar, deneyi yapan kişiden gizlenir. Deneyi yapan hangisiyle haberleştiğini bilmeden bunların ikisiyle de haberleşir. Deneyi yapan kişinin sorduğu sorular ve deneklerin verdiği cevaplar bir ekranda yazılı olarak verilir. Amaç deneyi

yapanın uygun sorgulama ile deneklerden hangisinin insan, hangisinin bilgisayar olduğunu bulmasıdır. Eğer deneyi yapan kişi güvenilir şekilde bunu söyleyemez ise, o zaman bilgisayar Turing testini geçer ve insanlar kadar kavrama yeteneğinin olduğu varsayılır.



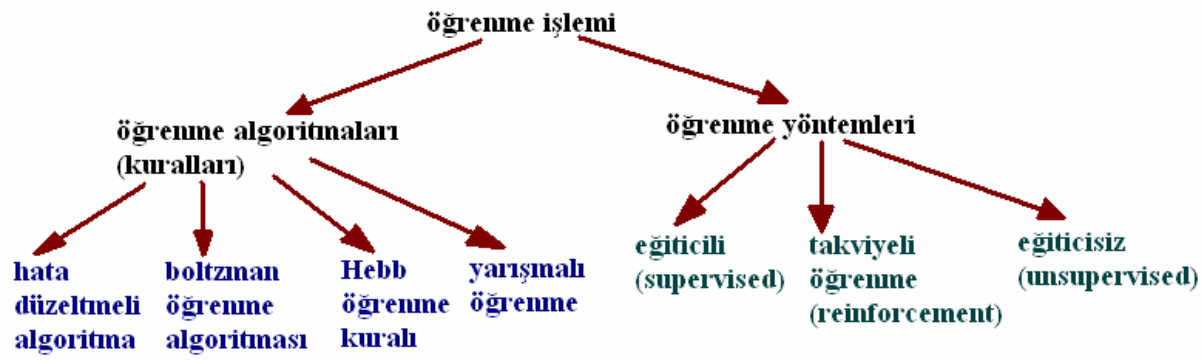
Şekil.2.7 Turing testinin gerçekleştirilmesi

Çin Odası Deneyi: California Üniversitesi'nden John Searle bilgisayarların düşünemediğini göstermek için bir düşünce deneyi tasarlamıştır. Bir odada kilitli olduğunuzu düşünün ve odada üzerinde Çince tabelalar olan sepetler olsun. Fakat siz Çince bilmiyorsunuz. Ama elinizde Çince tabelaları İngilizce olarak açıklayan bir kural kitabı olsun. Kurallar Çince'yi tamamen biçimsel olarak, yani söz dizimine uygun olarak açıklamaktadır. Daha sonra odaya başka Çince simgelerin getirildiğini ve size Çince simgeleri odanın dışına götürmek için başka kurallarda verildiğini varsayın. Odaya getirilen ve sizin tarafınızdan bilinmeyen simgelerin oda dışındakilerce soru diye, sizin oda dışına götürmeniz istenen simgelerin ise soruların yanıtları diye düşünün. Siz kilitli odanın içinde kendi simgelerinizi karıştırıyorsunuz ve gelen Çince simgelere yanıt olarak en uygun Çince simgeleri dışarı veriyorsunuz. Dışarıda bulunan bir gözlemcinin bakış açısından sanki Çince anlayan bir insan gibisiniz. Çince anlamanız için en uygun bir program bile Çince anlamanızı sağlamıyorsa, o zaman herhangi bir sayısal bilgisayarın da Çince anlaması olanaklı değildir. Bilgisayarda da sizde olduğu gibi, açıklanmamış Çince simgeleri işleten bir biçimsel program vardır ve bir dili anlamak demek, bir takım biçimsel simgeleri bilmek demek değil, akıl durumlarına sahip olmak demektir. (Rabunal, Dorado, 2006)

2.16 Öğrenme Stratejileri

Yapay sinir ağları gibi örneklerden öğrenen sistemlerde değişik öğrenme stratejileri kullanılmaktadır. Öğrenmeyi gerçekleştirecek olan sistem ve kullanılan öğrenme algoritması

bu stratejilere bağlı olarak değişmektedir. Bir yapay sinir ağı istenen çıkış kümesini üreten giriş kümesi uygulanacak şekilde konfigüre edilir. Şekil.2.8'de öğrenme işleminin yapay sinir ağları ile nasıl gerçekleştirilebileceğine dair bir çizelge görülmektedir. (Yıldırım, 2006)



Şekil.2.8 Öğrenme stratejilerinin sınıflandırılması

Eğiticili (Supervised) Öğrenme: Bu tür stratejide öğrenen sistemin olayı öğrenebilmesine bir öğretmen yardımcı olmaktadır. Öğretmen sisteme öğrenilmesi istenen olay ile ilgili örnekleri girdi/çıktı seti olarak verir. Yani her örnek için hem girdiler hem de o girdiler karşılığında oluşturulması gereken çıktılar sisteme gösterilir. Sistemin görevi girdileri, öğretmenin belirlediği çıktılara haritalamaktır. "Çok katmanlı algılayıcı ağı" bu stratejiyi kullanan ağlara örnek olarak verilebilir.

Destekleyici (Reinforcement) Öğrenme: Bu tür stratejide de öğrenen sisteme bir öğretmen yardımcı olur. Fakat öğretmen her girdi seti için üretilmesi gereken çıktı setini sisteme göstermek yerine sistemin kendisine gösterilen girdilere karşılık çıktısını üretmesini bekler ve üretilen çıktının doğru veya yanlış olduğunu gösteren bir sinyal üretir. Sistem, öğretmenden gelen bu sinyali dikkate alarak öğrenme sürecini devam ettirir. LVQ ağı olarak bilinen ağ, bu stratejiyi kullanan sistemlere örnek olarak verilebilir.

Eğiticisiz (Unsupervised) Öğrenme: Bu tür stratejide sistemin öğrenmesine yardımcı olan herhangi bir öğretmen yoktur. Sisteme sadece girdi değerleri gösterilir. Örneklerdeki parametreler arasındaki ilişkileri sistemin kendi kendisine öğrenmesi beklenir. Bu, daha çok sınıflandırma problemleri için kullanılan bir stratejidir. Yalnız sistemin öğrenmesi bittikten sonra çıktıların ne anlama geldiğini gösteren etiketlendirmenin kullanıcı tarafından yapılması gerekmektedir. ART ağları, bu yapıyı kullanan sistemlere örnek olarak gösterilebilir. (Öztemel, 2006)

Aktif Öğrenme: Geleneksel öğrenme yöntemlerinde öğrenici mekanizma pasif bir gözlemci gibi davranır. Eğitme kümesi, olay uzayının rasgele seçilen bir alt kümesidir. Eğitme kümesi, bilinmeyen bir kaynaktan gelen giriş çıkış çiftleridir denilebilir. Öğrenici mekanizmanın eğitme kümesi üzerinde herhangi bir işlem yapma yetisi yoktur. Öğrenicinin görevi, bu veriyi genelleştirerek daha önce görülmemiş veriler hakkında karar verebilmektir. Eğitme kümesi örnek uzayından rastgele seçilir. Eğitme kümesi büyüdükçe öğrenicinin olay uzayı hakkında bilgisi artarken, bu bilginin bir kısmı da gereksiz hale gelir.

Bu çalışmada mobil robotun engellerden kaçmayı öğrenmesi için kullanılan PNN algoritması aktif öğrenme gerçekleştiren bir yapay sinir ağıdır.

Aktif öğrenmenin bazı göze çarpan özelliklerini aşağıdaki gibi sıralayabiliriz;

- Aktif öğrenici, gözlemci gibi davranmaz. Çevresi ile her an etkileşim içerisindedir.
- Olay uzayını tarayarak daha önce hiç görmediği örnekleri seçme becerisine sahiptir.
- Bu seçimi yaparken, gereksiz bilgi taşıyan örnekleri reddetme yeteneğine sahiptir.

3. PROGRAMLANABİLİR LOJİK ELEMANLARIN MİMARİSİ ve PROGRAMLAMA TEKNİKLERİ

Sayısal işlem sistemleri, sayısal verileri işlemek için tasarlanmış ve hızlı donanımlara ve bu donanımlara işlevsellik kazandıracak esnek yazılımlara ihtiyaç duyan yapılardır. Sayısal işlem sistemi oluşturmada, donanım ve yazılım tabanlı olmak üzere iki geleneksel yöntem uygulanır.

Donanım Tabanlı Yöntem; Sayısal verileri işlemek için özel tüm devreler (ASIC) kullanılır. Bu tip devreler özel bir fonksiyonu gerçekleştirmek amacıyla üretildiğinden, bu fonksiyonları etkin ve hızlı bir şekilde gerçeklerler. Ancak işlevleri sınırlıdır ve sadece ilgili oldukları uygulamaya yönelik üretilmişlerdir. ASIC devreler, doğru ve hızlı sonuç vermesine rağmen çözüm ürettiği problemin çeşitli türevleri içim kullanılamayacaklardır. Yeni problemler için yeni donanımlara ve yeni ASIC yapılarına ihtiyaç duyulur. Bu da maliyet artışına ve zaman kaybına neden olur.

Yazılım Tabanlı Yöntem; Bu yöntemde tasarım değişikliklerine daha esnek olan mikroişlemciler kullanılır. Mikroişlemcinin çalıştırıldığı yazılımlar değiştirilerek hiçbir donanım değişişliğine gidilmeden tasarım ortamına yeni fonksiyonlar eklenebilir. Mikroişlemciler üzerinde çalışan yazılım uygulamaları aynı anda bir çok işlemi yerine getirebilmek için tek bir işlemcinin genel kaynakları kullanılırken, yavaş fakat esnek yazılımlar ile çalışırlar. Fakat sıradan bir uygulama için komutların bellekten okunması, onların yorumlanıp yerine getirilmesi, sistemin performansı ve hızını oldukça düşürmektedir.

Günümüzde sayısal işlem sistemlerinin kullanıldığı alanlardaki hızlı gelişmeler, üretim tamamlandıktan sonra da esnek, genel amaçlı olacak şekilde tasarlanan işlem sistemlerini ortaya çıkarmıştır. Özellikle iyi tasarlanmış ve işlemci yükünü azaltan, paylaşan donanımlar, performans artışı için iyi bir çözüm olabilir. Bununla beraber, işlevleri uygulama sırasında değiştirilebilen programlanabilir devre elemanlarının kullanımı da avantaj getirecektir. Bu devre elemanları ile gerçekleştirilen donanımlar, ASIC gibi devre elemanları ile yapılan klasik donanımlara göre daha işlevseldirler.

Tekrar düzenlenebilir sayısal işlem sistemlerin, ihtiyaç duyduğu esnek donanımlar, FPGA (Field Programmable Gate Array) kullanılarak karşılanır. Özellikle SRAM (Statik Rastgele

Erişimli Bellek) tabanlı FPGA'ler tekrar düzenlenebilirlik kabiliyetleri ve yüksek performanslı uygulamalardaki yeterlilikleri sayesinde, genel amaçlı sayısal donanımların tasarımlarında önemli rol oynar. Günümüz teknolojisinde FPGA ve VHDL gittikçe önem kazanmaya başlamıştır. VHDL ile SoC teknolojisinin kullanılmasıyla tanımlanan ve sentezlenen sistemler FPGA ile gerçeklenerek gerçek gücü ortaya koymaktadır.

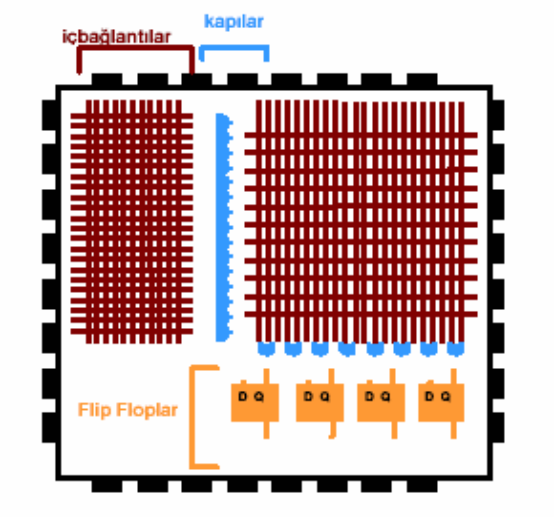
3.1 Programlanabilir Lojik Elemanların Gelişimi

Sayısal tasarımlarda kullanılan ilk programlanabilir lojik devre elemanı salt okunabilir bellek (PROM) elemanıdır. Bu elemanda, belleğin adres girişleri, lojik devrenin girişlerine, adreslenmiş gözdeki bilgiler de, lojik fonksiyonun çıkışlarına karşılık düşer. Genellikle karmaşık lojik fonksiyonlar için verimsizdirler.

Programlanabilir yapıların sonraki türleri PLD (Programlanabilir Lojik Devre)'lerdir. PLD'ler; SPLD (Basit Programlanabilir Lojik Eleman), CPLD (Karmaşık Programlanabilir Lojik Eleman), MPGA (Maske programlanabilir kapı dizileri), FPGA (Alan Programlanabilir Kapı Dizileri) olmak üzere beş bölümde incelenebilir.

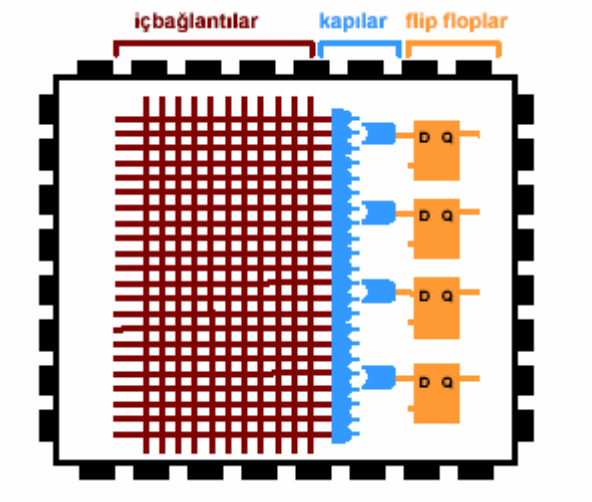
SPLD'ler, PLA (Programlanabilir Kapı Dizileri) ve PAL (Programlanabilir Dizi Lojiği) olmak üzere iki kısma ayrılırlar.

PLA (Programmable Logic Array): PLA mimarisi, Şekil.3.1'den de görüleceği gibi iki tane programlanabilir düzleme sahiptir. Bu iki programlanabilir düzlem AND ve OR kapılarının kombinasyonlarıyla oluşmakta ve AND işlemini birçok OR kapısı üzerinde paylaştırma esasına dayanmaktadır. Bu mimari oldukça esnektir fakat iki tane programlanabilir düzleme sahip olması üretim ve yollanma (mapping) gecikmelerini artırmaktadır.



Şekil.3.1 PLA lojik devre elemanının iç mimarisi

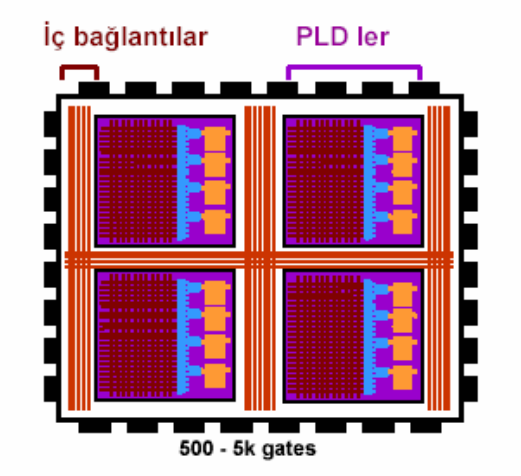
PAL (Programmable Array Logic): PAL mimarisinde ise sadece bir tane programlanabilir düzlem bulunmaktadır. Bu mimaride programlanabilir AND ve OR matrisleri bulunmaktadır. PAL'ler bu özellikleri ile ucuz ve yüksek hızlı performans sağlamaktadırlar.



Şekil.3.2 PAL lojik devre elemanının iç mimarisi

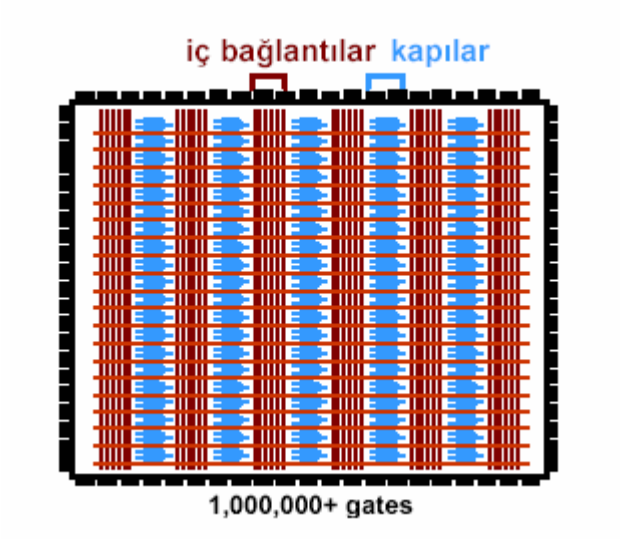
PLA ve PAL'ler basit programlanabilir lojik devreler (SPLD) olarak da adlandırılırlar.
SPLD'lerin sınırlı kapasiteleri daha yüksek kapasiteli programlanabilir devreler olan CPLD'lerin doğmasına neden olmuştur. CPLD'ler, SPLD benzeri birçok bloğun bir araya getirilmesiyle oluşmuş yapılardır.

CPLD sadece bireysel PLD'lerin aynı çip üzerinde toplanmış ve bu bireysel PLD'lerin birbirine bağlanmak için ara bağlantı yapılarının çip üzerinde düzenlenmiş halidir.



Şekil.3.3 CPLD lojik devre elemanının iç mimarisi

MPGA: MPGA, kullanıcıların lojik devresindeki özelliklere ve bağlantılara göre özel olarak üretilmiş transistör dizilerinden oluşur. Kullanıcı isteğine üretiminden dolayı, fabrikasyon süreci hem uzun hem de masraflıdır. MPGA'ler tam programlanabilir lojik tüm devreler olmasalar da programlanabilir türevleri olan FPGA'lerin gelişmesinde ilk aşama olmuştur.



Şekil.3.4 MPGA lojik devre elemanının iç mimarisi

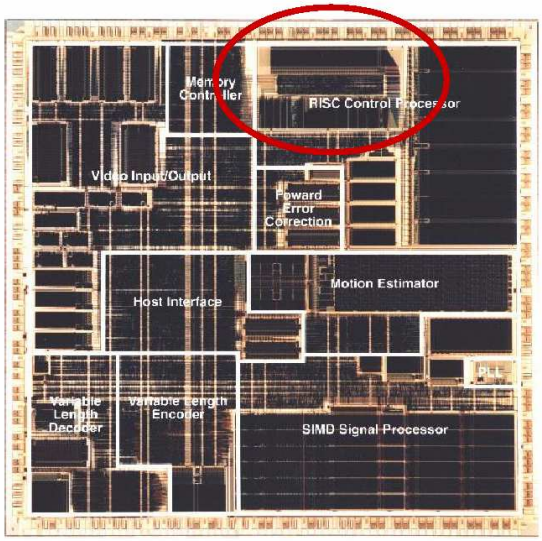
3.2 VLSI

Genel amaçlı YSA işlemcileri ile çok çeşitlilikteki YSA'ları bir benzetimi yapılabilir. Özel amaçlı VLSI sistemleri özel bir YSA için tasarlanırlar. Özel amaçlı VLSI tasarımları yüksek hız ve yoğun hesaplama isteyen tipik uygulamalar için idealdir. Bu alanda oldukça çeşitli uygulama çalışmaları ve teknikleri mevcuttur.

Özel amaçlı VLSI tümleşik devreler gerçek zamanlı uygulamalarda işlem hızını ve yoğunluğunu arttırma problemlerinin üstesinden gelmiştir. Fakat farklı farklı YSA'ların aynı tümleşik devre üzerinde kullanılamayışı (sınırlı kullanım) ve üretim aşamalarının hem maliyetli hem de çok zaman almasından dolayı bu tarz tümleşik devreler yaygın olarak kullanım alanı bulamamışlardır.

VLSI teknolojisi, öğrenme algoritmalarının donanımsal olarak gerçekleştirilebilmesine olanak tanımıştır. Yapay sinir ağlarıyla gerçekleştirilen öğrenme algoritmalarının sadece lokal bilgilere ve paralel işletime ihtiyaç duyuluyor olması donanımsal olarak gerçekleştirmede VLSI çiplerinin işleyişi ile birebir örtüşmüştür. VLSI çipleri içinde çarpıcıların, toplayıcıların, analog/dijital dönüştürücü ve benzeri blokların paralel olarak işleyebiliyor olması onları öğrenebilen algoritma tasarımında PC'lere göre çok daha güçlü kılmıştır.

VLSI çiplerinin tek olumsuz özelliği fabrikasyondan sonra iç bloklarının bir daha değiştirilme imkanının olmamasıdır. Bu yüzden daha esnek ve hızlı programlama için bu çalışmada bir sonraki bölümde anlatacağımız FPGA (Field Programmable Gate Array) çiplerini kullanacağız.



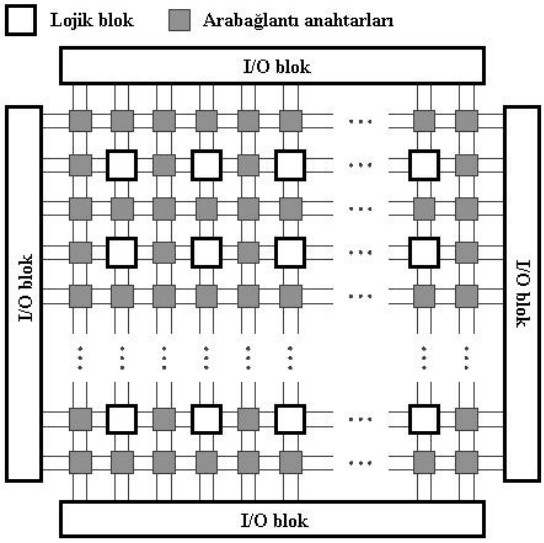
Şekil.3.5 Günümüz teknolojisi ile üretilmiş bir VLSI çip

3.3 FPGA (Field Programmable Gate Array)

FPGA (Field Programmable Gate Array), istenilen fonksiyonu gerçekleştirmek üzere programlanabilen lojik kapılar bütünüdür. Türkçe'si "Alan Programlanabilir Kapı Dizisi" olarak bilinir. CPLD ve MPGA'ların üstünlüklerini kullanmak ve sakıncalarından kurtulmak amacıyla Xilinx firması, 1985 yılında, FPGA'leri piyasaya sürmüştür. FPGA'ler programlanabilir lojik bloklar ve ara bağlantılardan oluşur. Kullanıcının tasarladığı devreye göre, FPGA üreticisi tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı örnek üretme özelliği FPGA'leri sayısal tasarım ortamlarının vazgeçilmezi haline getirmiştir.

Maliyet ve performans açısından bir devreyi mümkün olduğunca az sayıda çip kullanarak tasarlamak mantıklıdır. Çip üzerindeki devre miktarı ve fonksiyon kapasitesi önemlidir. Devrenin büyüklüğünü belirlemek için devrenin sadece genel lojik kapılarla gerçeklendiği kabul edilir ve kaç tane kapı gerektiği hesaplanır. Genellikle ölçme işleminde, devrenin gerçekleştirilmesinde kullanılan iki girişli NAND kapılarının sayısı bir ölçüt olarak alınabilir. Modern standartlara göre 20.000 lojik kapı içeren bir lojik devre büyük değildir. Büyük devreleri gerçekleştirmek için, daha fazla lojik kapasiteye sahip farklı tipte çipler kullanmak uygundur. FPGA'ler ile nispeten büyük lojik devreleri gerçekleyebilmek mümkündür. FPGA'ler lojik fonksiyonları gerçekleştirmek için lojik bloklar içerirler. Lojik elemanların her

biri istenilen lojik fonksiyonu yapmak üzere ve bağlantı noktalarının her biri bağlantı oluşturacak yada oluşturmayacak şekilde programlanabilir. Şekil.3.6'da bir FPGA çipin genel görünümü verilmiştir. Lojik bloklar iki boyutlu dizi şeklinde düzenlenmiştir ve ara bağlantı hatları, lojik blokların satır ve sütunları arasında yatay ve düşey olarak taşıyıcı kanallar olarak organize edilmiştir. Bu taşıyıcı kanallar, hatlar ve lojik blokların birbirlerine farklı yollar ile bağlanmalarını sağlayan programlanabilir anahtarlardan oluşmaktadır. Piyasada çeşitli sayılarda programlanabilir anahtarlar ve hatlar içeren FPGA çipleri bulunmaktadır.



Şekil.3.6 FPGA blok diyagramı

3.4 FPGA Teknolojisinin Ortaya Çıkış Sebebi

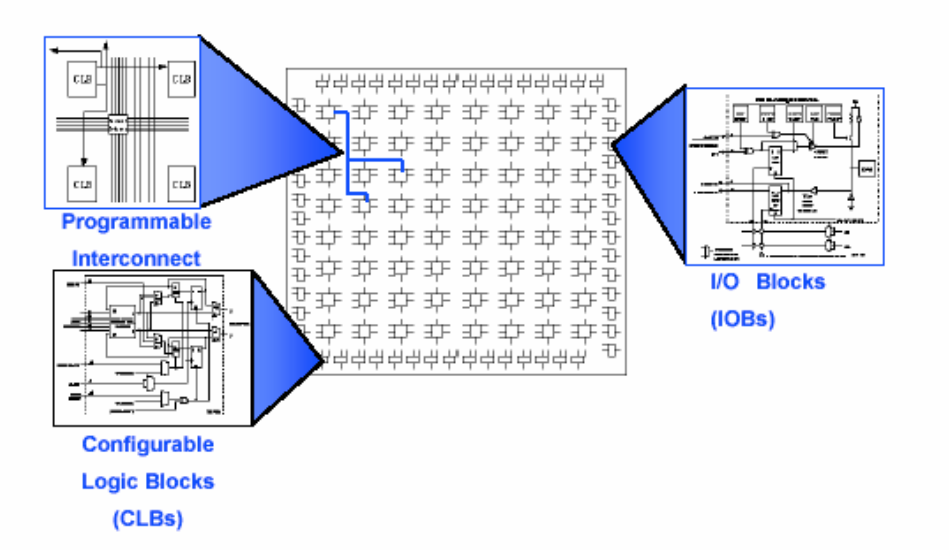
1980'lerde sayısal tüm devre teknolojileri arasında bir boşluk olduğu gün yüzüne çıkmıştı. Bir tarafta SPLD ve CPLD gibi programlanabilen ve hızlı tasarım sürelerine sahip mantıksal elemanlar bulunuyordu, fakat bu elemanlar büyük ve karmaşık fonksiyonları gerçekleyemiyordu. Diğer tarafta da uygulamaya yönelik tüm devreler (ASIC – Application Specific Integrated Circuit) vardı. ASIC çok büyük ve karmaşık fonksiyonları gerçekler fakat çok pahalıya üretilirler ve tasarım süreleri de daha uzundur. Ek olarak bir kez tasarlandıklarında bir daha programlanarak tasarım değiştirilemez. Sayısal tüm devre eleman çeşitleri arasındaki bu boşluğu 1984 yılında Xilinx firması FPGA'i üreterek kapatmıştır.

FPGA genel olarak dört ana piyasa kesimine rakip olmuştur. Bunlar ASIC, sayısal işaret işleme, gömülü mikrokontrolörler ve haberleşmenin fiziksel katmanıdır. Bunların dışında

FPGA yeniden yapılandırılabilir hesaplama (Reconfigurable Computing) adıyla yeni bir alan da oluşturmuştur. Bu alanda şirketler algoritmalarını yazılımdan FPGA'e geçirerek algoritmaların işleyişlerini hızlandırmaktadırlar. Bu sayede donanım benzetimlerinden, kriptoloji analizlerine varıncaya kadar birçok uygulama gerçekleştirilmektedir.

3.5 FPGA Mimarisi

FPGA'ler son kullanıcı tarafından hiçbir üretim basamağına gerek duyulmadan, istenilen sayısal işlevleri yerine getirecek şekilde programlanabilen yapılardır. Genel yapısı Şekil.3.7'de gösterilmiştir. FPGA üç temel bloktan oluşur; lojik bloklar, giriş çıkış blokları ve bağlantı blokları. Bu blokların dışında aritmetik işlem blokları ve bellek blokları gibi özel bloklar içeren FPGA yapıları da mevcuttur.

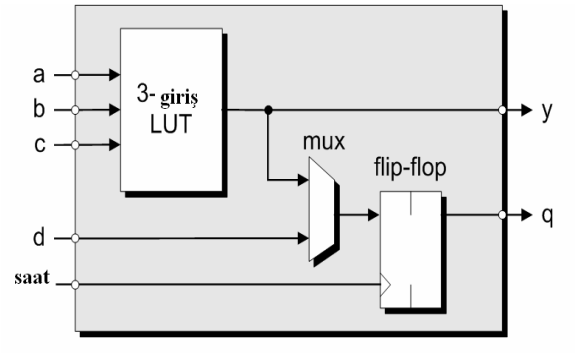


Şekil.3.7 FPGA'i oluşturan bloklar

Bir bağlantı noktasının bağlı olması yada olmaması o noktadaki transistörün açık yada kapalı olması ile ayarlanır. (Karen, Mehta, 2002)

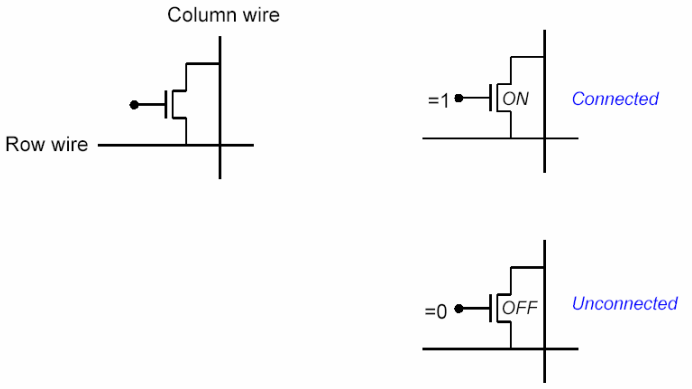
Küçük taneli lojik bloklar, genellikle iki girişli bir lojik kapıya veya birkaç girişli çoklayıcıya eşlik eden saklama elemanından oluşur. Bu bloklar karmaşık lojik fonksiyonların gerçeklenmesinde yüksek verimli yapı taşları olarak kullanılır. Ancak, bu gerçekleme sırasında kullanılan bağlantı kanallarının ve programlanabilir anahtarların sayısı oldukça

fazladır. Kaba taneli lojik blokların yapıları çok çeşitlilik göstermelerine karşın yaygın olarak doğruluk tablosu veya çoklayıcı gibi daha büyük yapılardan ve saklama elemanlarından oluşur. Kaba taneli lojik blok örneği Şekil 3.8'de gösterilmiştir. Bu tür lojik bloklar, karmaşık lojik fonksiyonların gerçeklenmesinde küçük boyutlu lojik bloklar kadar verimli olmasalar da gerçekleme sırasında kullanılan bağlantı kanallarının ve programlanabilir anahtarların sayısı daha azdır.



Şekil.3.8 Kaba taneli lojik blok örneği

Bağlantı blokları; lojik bloklar ile giriş çıkış blokları arasındaki bağlantıyı sağlayan yapılardır. Bu yapılar, yollandırma kanalları ve programlanabilir anahtarlardan oluşur. Yollandırma kanalları, farklı uzaklıktaki lojik blokların en etkin şekilde bağlanması amacıyla farklı uzunlukta tasarlanır. FPGA içinde gerçeklenebilecek lojik fonksiyon miktarı, yollandırma kanallarının sayısı ile doğru orantılıdır. Yollandırma kanallarının sayısı az olan FPGA yapılarında, lojik blok sayısı fazla olsa bile, gerçeklenebilecek lojik fonksiyon sayısı, lojik blok sayısına oranla kısıtlıdır. Programlanabilir anahtarlar; statik RAM hücresi ile kontrol edilen geçiş transistörü, antifuse, EPROM ve EEPROM teknolojileri kullanılarak oluşturulur.



Şekil.3.9 FPGA ara bağlantıları

FPGA'lerin bir adet *configuration bit'*leri bulunur. Configuration bitleri buraya tek tek shift edilerek yerleştirilir (Karen, Mehta, 2002).

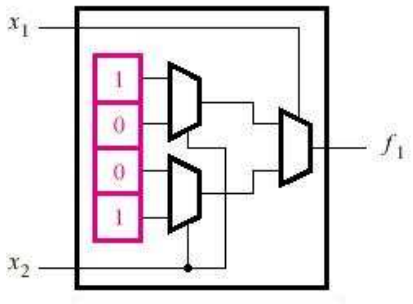
FPGA' deki her bir lojik blok tipik olarak az sayıda girişe ve bir çıkışa sahiptir. Piyasada farklı özellikli lojik bloklara sahip FPGA'ler bulunmaktadır. Çok yaygın olarak kullanılan lojik blok LUT (look up table'dır). LUT küçük lojik fonksiyonların gerçekleştirilmesini sağlayan depolama hücreleri içerir. Her bir hücre tek bir lojik ifadeyi tutabilir "0" yada "1". Depolanmış değer depolama hücresinin çıkışının elde edilmesinde kullanılır. İki değişkenli doğruluk tablosu dört satıra sahiptir. Bu satırlar LUT' un depolama hücrelerine karşılık gelmektedir. Herbir hücre doğruluk satırındaki çıkış değerlerinden birini içermektedir. "x1" ve "x2" giriş değerleri üç çoklayıcının seçici girişi olarak kullanılmaktadır. LUT' un çıkışı olarak, dört depolama hücresinden birinin içeriğini seçmek x1 ve x2'nin giriş değerlerine bağlıdır. Şekil.3.10'deki doğruluk tablosu göz önüne alınırsa, iki girişli bir LUT' un nasıl gerçekleştirilebileceği görülür.

x_1	x_2	f_1
0	0	1
0	1	0
1	0	0
1	1	1

$$f_1 = \bar{x}_1\bar{x}_2 + x_1x_2$$

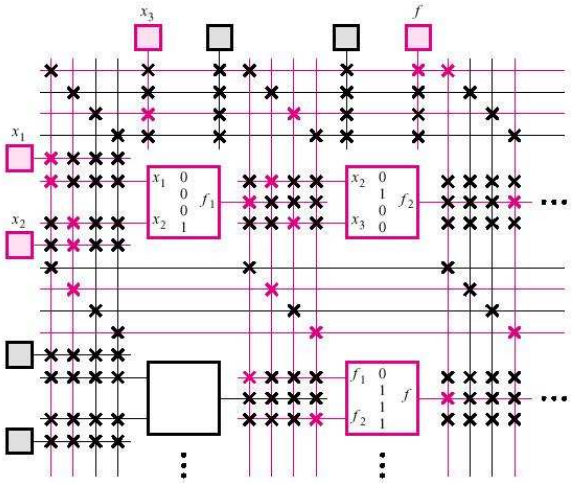
Şekil.3.10 İki girişli bir LUT için doğruluk tablosu

Bu tablodan f1 fonksiyonunun LUT'a depolanabildiği aşağıdaki şekilde gösterilmiştir. LUT'daki çoğullayıcıların ayarlanması ile f1 fonksiyonu sağlanır. x1=x2=0 iken, LUT çıkışı depolama hücresi tarafından sürülür.



Şekil.3.11 İki girişli LUT

Lojik devreleri FPGA ile gerçekleştirebilmek için, devredeki her bir fonksiyonun tek bir lojik blok içerisine sığabilmesi için küçük olması gerekmektedir. CAD yazılımları kullanılarak kullanıcı devreleri gerekli formun içerisine otomatik olarak çevrilmektedir. LUT'lardaki depolama hücreleri uçucudurlar. Yani çipin güç kaynağı kesildiğinde depolanmış değerleri kaybederler. Bu yüzden FPGA'lerin programlanabilmesi için güç kaynağının açık olması gerekmektedir. FPGA elemanını barındıran devre kartları üzerinde verileri devamlı olarak tutabilen küçük hafıza çipleri vardır. Bu hafıza çipleri PROM (Programmable Read Only Memory) olarak adlandırılır. Güç kaynağı aktif olduğunda, depolama hücrelerinin içerikleri PROM'dan otomatik olarak yüklenecektir. Aşağıda bir devreyi gerçekleştirmek için programlanmış küçük bir FPGA görülmektedir.



Şekil.3.12 Programlanmış bir FPGA'in bir bölümü

ASIC devrelerden aldığı paralel çalışabilme ve geleneksel işlemlerin sahip olduğu tekrar düzenlenebilirlik ile FPGA, yapay sinir ağları ile gerçekleştirilecek bir yapı için en uygun seçimdir. Oysa paralel çalışan yapay sinir ağlarının, geleneksel işlemcilerle sadece benzetimi yapılabilmektedir. Ayrıca VHDL kodlarıyla donanımı gerçekleştirilen FPGA'ler güvenlik kavramını en üst düzeye çıkartmaktadır (Karen, Mehta, 2002).

3.6 FPGA Programlama Teknolojileri

FPGA'lerin programlanabilme özellikleri, içerdikleri programlanabilir anahtarlardan ve lojik bloklardan gelmektedir. Programlanabilir anahtarın gerçeklenme teknolojilerine göre FPGA'lerin programlanma teknolojileri belirlenir.

Programlanabilir anahtar matrisleri üretildikleri teknolojiden bağımsız olarak açık veya kapalı olmak üzere iki konumda bulunabilirler. Programlanabilir anahtarların istenilen özellikleri aşağıdaki gibi sıralanabilir:

- Yarı iletken üzerinde en küçük alanı kaplamaları
- İletim durumundayken küçük bir direnç ve kesim durumundayken de yüksek bir direnç göstermeleri
- İşlev gördükleri noktalarda yollandırma kanallarına minimum kapasite getirmeleri
- Çok sayıda programlanabilir anahtarlar entegre içinde güvenilir bir şekilde gerçeklenebilmeli.

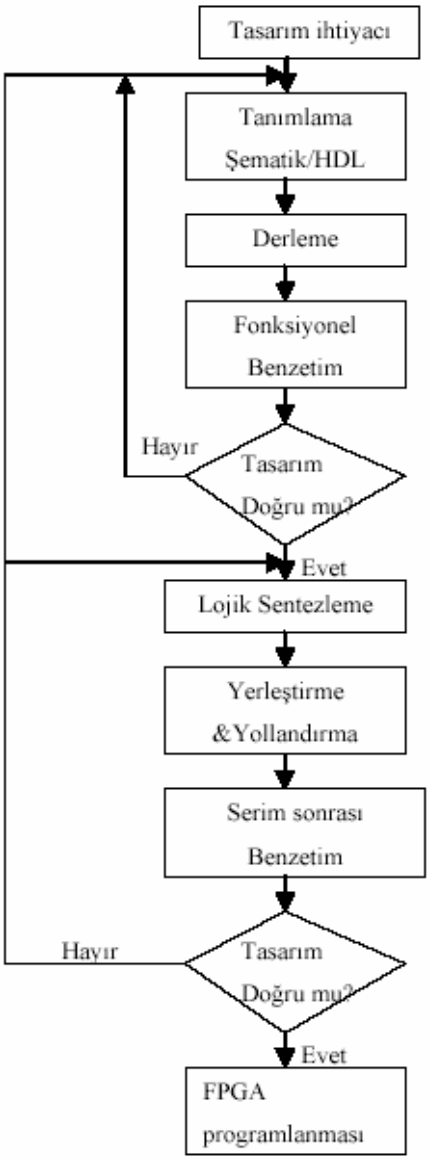
SRAM Programlama Teknolojisi: Bir SRAM programlı FPGA'de, normal işlem boyunca program statik hafıza hücresinde tutulur. Bunun için çip sık sık "SRAM programlanabilir" olarak ifade edilir. Tümleşik devre üzerinde SRAM hafıza hücreleri için ayrılmış yer yoktur. SRAM hücreleri kontrol ettikleri lojik elemanlar arasında dağıtılmışlardır. SRAM hücreleri ile üç ayrı inşa bloğu kontrol edilir. Look-up tablosu, adres hatlarını kontrol eden fonksiyon girişleriyle, hafıza hücrelerinden yapılmış önemli inşa bloklarından biridir. Diğer bir inşa bloğu, programlanabilir ara bağlantı noktası (PIP) olarak isimlendirilir.

SRAM programlama teknolojisinin en büyük dezavantajı geniş alan kaplamasıdır. Diğer dezavantajı ise uçuculuktur. Ancak, SRAM pogramlama teknolojisi iki önemli avantaja sahiptir; sadece standart tümleşik devre teknolojisinin sağladığı hızlı tekrar programlanabilirlik ve hatta karmaşık devreler için bile düşük güç tüketimidir. SRAM tabanlı FPGA'ler, tekrar düzenlenebilirlik kabiliyetleri ve yüksek performanslı uygulamadaki yeterlilikleri nedeniyle sayısal işaret işlemede temel devre elemanı olarak yaygın olarak kullanılmaktadırlar. Ayrıca tasarım tamamlanır tamamlanmaz test etmek mümkün olduğundan uygulamalarda oldukça avantaj sağlamaktadırlar.

3.7 FPGA Kullanılarak Gerçekleştirilen Devrelerin Tasarım Süreci

Tasarım süreci , gerçeklenecek devre fonksiyonlarının, sözle veya şematik olarak tanımlanması ile başlar. Sözde tasarımda genellikle yüksek seviyeli donanım tasarım dilleri (Hardware Description Language, HDL) kullanılır. Şematik tanımlamada ise bir çok firma

tarafından geliştirilmiş şematik editör programlarından faydalanılır. Tanımlama ne şekilde olursa olsun, derleme işlemi sonrasında, tüm tanımlamalar, standart bağlantı listesi (netlist) biçimine çevrilir. Yapılan tanımlamaların, istenilen fonksiyonları yerine getirip getirmediği fonksiyonel benzetim (functional simulation) yapılarak test edilir. Benzetim sonucuna göre, tanımlamada gerekli değişiklikler yapılır. Şekil.3.13'de tasarım sürecinin akış şeması verilmiştir.



Şekil.3.13 FPGA kullanılarak tasarım sürecinin genel akış diyagramı

Bu çalışmadaki tasarımda, sistem VHDL kullanılarak sözle tanımlanmıştır. Benzetimler için Modelsim, sentezleme içinse Xilinx ISE programı kullanılmıştır.

3.8 Model Robot İçin Seçilen FPGA ve Özellikleri

Model robot tasarımı için seçilen xc3s200 çipi, Xilinx firmasının üretmiş olduğu, Spartan-3 ailesinden bir programlanabilir kapı dizisidir. Kapı başına yada bacak sayısı başına hesaplandığında en düşük maliyete sahip, bir çok uygulama için çok iyi performans gösteren bir FPGA'dir. Programlama ve geliştirme kitlerinin ücretleri makul değerlerdedir. Xc3s200 kodunun en sonunda bulunan 200 sayısı, FPGA'de kaç bin tane kapı bulunduğunu ifade eder. Seçilen FPGA'in temel özellikleri Çizelge.3.1'de görülmektedir (Xilinx).

Çizelge.3.1 Xc3s200 genel özellikleri

Sistem Kapı Sayısı:	200.000 adet
Lojik Hücre Sayısı:	4320
Blok RAM bitleri:	216K adet
Dağılmış RAM bitleri:	30K adet
DCM:	4 adet
Çarpıcı:	12 adet
I/O standart:	24 adet

3.9 FPGA Tabanlı Ağ Mimarisi

Analog ve dijital devrelerle oluşturulmuş sistem mimarileri yapay sinir ağları için önerilebilir. Analog uygulamaların hassasiyet oranı yüksektir fakat gerçeklenmesi zordur ve ağırlık saklamada sorun yaşarlar. Dijital sistemlerde ise hassasiyet oranı düşüktür fakat ağırlık saklama koşulu aşılmıştır. YSA'ların mimarisi biyolojik sinirlerden esinlendiği için doğasında paralellik barındırmaktadır. Mikroişlemci ve DSP paralel tasarımlar için uygun değildir. Tamamen paralel bir yapı ASIC ve VLSI teknolojileriyle yapılabilir fakat bu yöntem maliyet ve zaman açısından oldukça masraflıdır. Buna ek olarak özel bir YSA için üretilmiş ASIC mimari sadece hedef uygulama için kullanılabilir. FPGA paralel mimarinin yanında esnek tasarım, maliyet ve zamandan tasarruf sağlar. FPGA mimarisi üzerinde en etkili şekilde gerçekleştirilen ağlar sınıflandırma yapan ağlardır.

FPGA'de haritalanma yapılırken paralellikten mümkün olan en büyük ölçüde yararlanılması beklenir. Yer tasarrufu ve hız artırımı için YSA yapısında kullanılacak doğrusal olmayan fonksiyonun çeşitli girişlere vereceği çıkışlar bir tablodan elde edilebilir.

YSA sistemlerinde genellikle analog veriler işlenir. Oysaki FPGA dijital bir devre elemanıdır. FPGA giriş değerleri olarak sadece dijital (0 veya 1) verileri kabul eder. Uygulama safhasında yapılması gereken ilk çalışma; analog bir sistemin dijital sistemlerle ifade edilmesidir. Analog hesaplama sistemini dijital sayı değerleriyle hesap etmek için bir çok sayı formu geliştirilmiş ve standartlaştırılmıştır (Vonk, Veelenturf, Jain, 1996).

3.10 YSA'nın FPGA Kullanılarak Gerçekleştirilmesi Alanında Yapılan Çalışmalar

Guccine ve Gonzalez (1993) geleneksel bir programlama modeli önerdiler. Bu model hesaplamalarda vektör tabanlı veri-paralel yaklaşımıdır. Bu model, yüksek seviyeli dillerde (örneğin C) yazılan algoritmaları, yüksek performanslı dijital devrelere dönüştürür. Bu çalışmada, bahsedilen teknik öğrenme algoritması modellemede kullanılacaktır. Zhu et al (1999) tablo (LUT-look-up-table) tabanlı çarpma devreleri ile yapay sinir ağlarının kalıtımsal paralelliğinin başarılı bir şekilde gerçeklenebileceğini göstermişlerdir. Kalp hastalarının sınıflandırılması için taşınabilir dijital bir sistem prototipi geliştirmişlerdir. Bu sistem eğitilmiş bir ağın Xilinx XC 4000 serisi FPGA üzerinde gerçeklenmesiyle oluşturulmuştur. Nicholas ve Al (2002) ileri beslemeli bir yapay sinir ağının Xilinx Virtex 2E FPGA üzerinde gerçeklediler. Bu gerçekleştirim sırasında kayan noktalı sayı duyarlılığı ile sabit noktalı sayı duyarlılığının karşılaştırmasını yaptılar ve geri yayılım algoritması için en uygun sayı sisteminin 16 bit duyarlıklı sabit noktalı sayılar olduğunu kanıtladılar.

3.11 VHDL

VHDL kelimesi "VHSIC (Very High Speed Integrated Circuits) Hardware Description Language"in kısaltılmasından gelmektedir. Gelişen teknolojiyle beraber çiplerin büyümesiyle dijital devreleri gerçekleştirmeden önce lojik diyagramlarını çizmek içinden çıkılmaz bir hal almıştır. Bu sorunu çözmek için VHDL programlama dili geliştirilmiştir. VHDL dili ilk olarak 1980'lerin başında geliştirilmeye başlamıştır.

VHDL dili sayesinde çok karmaşık devre dizaynlarını bile çok hızlı şekilde gerçekleştirmek mümkündür çünkü VHDL dili tüm lojik kapıları tek tek elinizle oluşturmanızı istemez, sadece kendisine beklenen davranışın ne olduğunu söylemenizi ister. Ayrıca VHDL dili ile bir program yazdığınızda ve çipinizi değiştirmek istediğinizde tüm lojik tasarımı baştan yapmak zorunda kalmazsınız.

VHDL dilinin temel kapsamları;

- Davranış (behaviour)
- Arabirimler (interfaces)
- Yapı (structure)
- Test (test benches)
- Zamanlama (timing)
- Analiz, detaylandırma, benzetim (analysis, elaboration, simulation)
- Sentezleme (synthesis) şeklinde tanımlanabilir.

VHDL, C yada Java gibi bir programlama dili değildir. VHDL, dijital sistemlerdeki fiziksel donanımların modellenmesinde kullanılır. Bu nedenle, donanımda sistem dizaynını tanımlamak istediğimizde VHDL kullanırız.

VHDL, başlangıçta Department of Defense (DoD) tarafından geliştirilmeye başlanmıştır. DoD, makine ve insan için aynı zamanda okunaklı ve yapısal olarak güçlü, anlaşılabilir, kod yazmaya elverişli, böylece kaynak kodunun kendisi bir çeşit kullanım kılavuzu görünümünde olan bir donanım tanımlama dili talep etmekteydi. DoD'un geliştirici grubu, dili geliştirmek için bir kontrat yaptı ve dilin ilk versiyonu 1985'de yayınlandı. Daha sonra dil standartlaştırılmak için IEEE'ye transfer edildi. Sonra dil IEEE 1076-1987 standartı haline geldi ve 1987'de onaylandı. Beş yıl sonra dilin 1076-1993 biçimindeki yeni özellikler eklenerek oluşturulan versiyonu yeniden onaylandı. Yeni versiyon, saf standardın genişletilmişidir. Bu genişletme, örneğin geniş çapta ihtiyaç duyulan veri tipleri ve alt programlar içeren paketleri (std_logic_1164, numeric_bit, numeric_std,...) yada IEEE 1076.6'nın sentez alt kümesine benzeyen özel VHDL altkümelerinin tanımını kapsar. Son genişletme, VHDL'e analog ve karışık sinyal elemanlarının eklenmesi ile yapıldı. Sonuçta oluşan VHDL'in superset'idir ve VHDL-AMS (analog-mixed-signal) olarak bilinir.

VHDL, ASIC (Application Specific Integrated Circuits)'ın geliştirilmesi için kullanılır. VHDL kodunda bir kapı – seviye net listesinin (gate-level netlist) otomatik dönüşümü için araçlar, erken bir zamanda geliştirildi. Bu dönüşüm, sentez ve genel dizayn akışının gerekli bir bölümüdür.

VHDL dili kullanılarak tasarım yapılmasının standart donanım tasarım yöntemine göre bazı üstünlükleri vardır. Bunlar aşağıdaki gibi sıralanabilir.

Tasarım süresi: Teknolojinin hızla gelişimi beraberinde gerçeklenen bir devrenin kullanım ömrünü azaltmaktadır. Bu, devrenin tasarım zamanının kısıtlanması gerektiği anlamına gelir. Böyle durumlarda, devrenin optimum tasarımlanmış olmasının ötesinde tasarım süresinin kısa olması ön plana çıkar. VHDL dili kullanılarak tasarım yapılması özellikle bu noktada tasarımcıya önemli yararlar sağlar. VHDL dili doğrıdan donanım tasarımına göre, tasarım süresi açısından çok daha kısa sürede sonuçlanır.

Tasarım esnekliği: Teknolojideki değişimle birlikte kullanılan elemanların yapıları da değişmektedir. Yapı değişikliklerinin daha önce yapılmış tasarımlarda çalışabilmesi için kullanılan tasarım ortamının buna uygun olabilmesi gerekir. VHDL dili fonksiyon bağımlı çalışır. Dönüştürücü programlar yardımıyla yazılımın donanım yapısı oluşturulur. Teknoloji değişimleri durumunda sadece bu dönüştürücü programların yeni teknolojiye uygun hale getirilmiş olması (uygun kütüphane dosyalarının eklenmesi ve yeni çipe ait donanımsal bilgilerinin eklenmesi gibi) yeterli olacaktır.

Tasarım kolaylığı: Genel olarak VHDL dili kullanılarak yapılan tasarımlarda, klasik donanım tasarımına göre daha az donanım bilgisine ihtiyaç duyulur.

Yenilenebilme kolaylığı: Teknolojideki hızlı değişim, gerçeklenen devrelerin değişim sürelerini azaltmıştır. Bu nedenle yapılacak değişimlerin hızlı bir şekilde gerçeklenebilmesi gerekmektedir. VHDL dili ile yapılan tasarımlarda değişim esnekliği tasarımcının yazılım gücü ile sınırlıdır.

Veri Gösterimi: YSA sistemlerinde genellikle analog veriler işlenir. Oysa ki FPGA dijital bir devre elemanıdır. FPGA giriş değerleri olarak sadece dijital verileri kabul eder. Uygulama safhasında yapılması gereken ilk çalışma; analog sistemin dijital değerlerle ifade edilmesidir. Analog hesaplama sistemini dijital sayı değerleriyle hesap etmek için bir çok sayı formu geliştirilmiş ve standartlaştırılmıştır. Fakat bunlardan duyarlılık bakımından en göze çarpanı IEEE'nin, kayan noktalı sayılar ve sabit noktalı sayılar formatlarıdır. Bu projede, hassasiyet derecesinin daha yüksek olmasından dolayı IEEE 754 formatındaki (IEEE 1981) tek duyarlıklı kayan noktalı sayılar ile çalışıldı. Bu format sayesinde gerçel sayılar, tek

duyarlılıkta (32-bit) veya çift duyarlılıkta (64-bit) gösterilme imkanına sahiptir. Gerçel sayılar bilgisayarda 32 bit formata çevrildikten sonra FPGA'e girilmektedir. Tabi ki FPGA içindeki bütün sayısal işlemler kayan noktalı sayılar formatına uygun olarak yapılmak zorundadır. Bu noktada kayan noktalı sayılara ait dört işlem algoritmalarının VHDL ile gerçeklenmesi gerekmektedir.

VHDL Veri Nesneleri: Bir veri nesnesi, belirli bir tipin değerini tutar. Veri nesneleri donanımda direk sentez yapabilir;

Sinyal (Signal): Fiziksel bir çevrim gösterir. Güncel değeri ve belirlenmiş bir sonraki olası değerleri içeren değerlerin listesini tutar.
Sinyal tanımlama;
SIGNAL sresetn: STD_LOGIC;
SIGNAL address: STD_LOGIC_VECTOR (7 downto 0);

Değişken (Variable): Hesaplamaların sonuçlarını tutmak için kullanılır. Mutlaka fiziksel bir çevrim göstermesi gerekmez.
Değişken tanımlama;
VARIABLE index: INTEGER range 0 to 99:=20;
VARIABLE memory: BIT_MATRIX(0 to 7, 0 to 1023);

Sabit (Constant): Değişmeyecek bir değeri içerir. Simülasyonun başlamasından önce saptanır.
Sabit tanımlama;
CONSTANT cycle_time: TIME:=100ns;
CONSTANT cst: UNSIGNED (3 downto 0);

Öntanımlamalı Veri Tipleri:
Standart Lojik Tip: STD_LOGIC, STD_LOGIC_VECTOR (0, 1, Z ve – değerlerini tutabilir.),
Bit Tip: BIT, BIT_VECTOR
Integer (Tam Sayı) Tip: INTEGER
Floating-point (Kayan noktalı) Tip: REAL
Physical (Fiziksel) Tip: TIME
Enumeration (Liste) Tip: BOOLEAN, CHARACTER

STD_LOGIC ve STD_LOGIC_VECTOR tiplerinin kullanımı için, VHDL dizayn dosyası the std_logic_1164 paketini içermelidir.

VHDL Yapısal Elemanları: Temel olarak varlık (entities), mimari (architecture), biçim (configurations) ve paket (packages) (paket gövdesiyle birlikte) VHDL'in yapısal elemanlarıdır.

Varlık (Entity) Tanımlaması: VHDL'de her bir blok kendi başına bir devre olarak düşünülür ve "entity" olarak adlandırılır. Verilen bir lojik fonksiyon için bütün giriş ve çıkışları tanımlar. Yani lojik fonksiyonun dış dünya ile bağlantısını tanımlar. Her VHDL tasarım mutlaka en az bir entity içerir. Port tanımlamaları farklı biçimlerde olabilir.

IN: Sadece okunabilir. Giriş için kullanılabilir.
OUT: Sadece yazma yapılabilir. Çıkış için kullanılır.
INOUT: Hem okunabilir hem de yazılabilir. Giriş ve çıkış olarak kullanılabilir.
BUFFER: Hem okunabilir hem de yazılabilir.

Burada port diye bahsedilen, giriş ve çıkış işaretleridir. Port ifadesi, her bir işaret için, port tanımlayıcı, port yönü ve port veri tipini belirlemelidir. Portda 3 yön kullanılır. Giriş için in, çıkış için out, çift yönlü portlar için ise inout kullanılır. Pek çok veri tipi kullanılabilir (Ashenden)

Mimari (Architecture) Tanımlaması: Lojik fonksiyonunun işlevi mimari kısım tarafından belirlenir. Her entity için bir "architecture" tanımlanır.

Davranışsal Tanımlama: Donanım komponentlerini modellemek için kullanılan davranışsal yaklaşımda, diğer yaklaşımlardan farklı olarak dizaynın nasıl yapılacağı önemli değildir. Genel olarak kara kutu modellemesi olarak da isimlendirilir. Davranışsal modelleme, devrenin iç yapısından daha ziyade giriş ve çıkışların ne olacağının modellemesi üzerine kuruludur. Davranışsal tanımlama VHDL dilinde iki şekilde kullanılır. Birinci olarak, diğer yöntemler ile yapılması zor ve uğraştırıcı olan kompleks komponentleri modellemek için başvurulur. Örnek vermek gerekirse bir ticari parçaya takılan bir mikroişlemciyi simule etmede kullanılabilir. Mikroişlemcide, iç yapısından daha ziyade giriş ve çıkışları önem

kazanır. Bu nedenle davranışsal modelleme kullanmak daha uygun olur. İkinci olarak, bazı dizaynlarda davranışsal modelleme kullanmak daha etkili ve daha rahat olabilir. Bu durumda, davranışsal modellemei dizaynın bir kısmında yer alacaktır.

Davranışsal modellemeler process ifadesi ile tanımlanır. Bu ifade architecture gövdesinin başında signal bloğunun bulunduğu gibi bulunur. Process ifadesi, programlama dillerinde kullanıldığı gibi birden fazla sıralı ifadeden oluşabilir. Bu ifade satırları, giriş değerleri üzerinde işlemleri yaparak çıkış değerlerini hesaplar. Sıralı ifadeler bazen çok etkili olmakla beraber, bazen donanımsal dengi olmayabilir. Process ifadesi, process'in çıkışlarını tespit etmek için sinyal atamaları içerebilir. Basit bir process ifadesi aşağıdaki gibidir:

```
compute_xor: process(b, c)
begin
a<=b xor c;
end process;
```

Parantez içinde bulunan sinyal listesi sensivity list sayesinde olarak isimlendirilir. Process ifadesinin iç yapısı bilinmediğinden, tekrar hesaplama için ne zaman kullanılacağı tespit edilemez. Sensivity list sayesinde, hangi sinyallerin değişmesi durumunda process'in tekrar hesaplanacağı bilinebilir. Sensivity listesinde bulunan sinyallerden birinde olay olması durumunda, process tekrar hesaplanır. Bir process, içerdiği ifadelerin hesaplanması ile tekrar hesaplanır.

Process ifadesini kullanabilmenin en önemli avantajı C gibi programlama dillerinde olduğu gibi döngü, karşılaştırma gibi işlemleri kolaylıkla yapmamıza müsaade vermesidir. Aksi halde tek tek lojik elemanlarla bu işlemleri yapmaya çalışmak durumu işin içinden çıkılmaz hale getirebilecekti. Projemizde bu yapıdan faydalanarak donanımsal yapıyı gerçekleştireceğiz.

Yapısal Benzetim Yoluyla Tanımlama: Dizaynları daha anlaşılır kılmak ve daha kolay yönetebilmek için dizayn genellikle alt bloklara bölünür. Bu alt bloklar birleştirildiği zaman istenilen dizayn ortaya çıkar. Dizayna şematik bir yaklaşım kullanılacak olursa, bu bir blok diyagram editörü ile gerçekleştirilir. VHDL dizaynının her bir parçası bir blok olarak görülür. Bir VHDL dizayn sadece bir blok içinde yapılabileceği gibi alt bloklara ayrılarak da yapılabilir. Her bir blok entitiy kısmına sahiptir. Bu kısım ilgili bloğun arabirimini tanımlar.

Başka bir parça ise bloğun ne iş yaptığını tanımlar. Arabirim tanımlaması datasheet'e pin giriş ve çıkışlarını belirtmeye benzer. Tanımlama kısmı ise bloğun şemasının çizildiği kısma benzer.

4. ROBOTİK

4.1 Robot

Robot sözcüğünü ilk olarak Karel Capek adlı Çekoslovak bir yazar 1921'de yazdığı RUR (Rossum's Universal Robots) adlı tiyatro oyununda kullanmıştır. Çekoslovakça'da "robota" sözcüğü "zorla çalıştırılan işçi" anlamına gelmektedir (Haton, 2003). Robotlar fiziksel dünyayla ilgili görevleri gerçekleştirmek ve insanın yapacağı işleri yaptırarak insan yükünü azaltmayı amaçlayan cihazlardır. Robotlar, dış dünyadan bilgileri alabilmek için sensörler ve dış dünyaya tepki işaretlerini verebilmek için efektörler ile donatılmıştır. Robotlar üç ana kategoriye ayrılabilir: idareci (manipulators) robotlar, mobil robotlar ve insansı (humonoid) robotlar.

Robotlar sensörlerden aldıkları verileri kendileri için anlamlı hale gelecek şekilde işlerler, ne işlem yapmaları gerektiğine karar verirler ve dış dünyaya tepkilerini verirler. Gelişen yapay sinir ağları teknikleri; robotlara bu işlemleri yaparken aynı zamanda kendi kendine ya da bir eğitici yardımıyla öğrenebilme, davranışlarını iyileştirebilme gibi yetenekler kazandırmıştır.

4.2 Robot Sensörleri

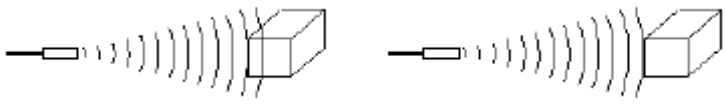
Sensörler robotların çevreleri ile arasındaki alıcılardır. Sensörler, sistem dışından gelen uyarılara tepki veren, bunları algılayan, ve önceden belirlenmiş bazı değişiklikleri ölçebilen algılayıcı cihazlardır. Robotik uygulamalarda kullanılabilecek iki çeşit sensörden söz edebiliriz. Bunlardan bir tanesi; çevredeki diğer kaynakların yansıttığı sinyalleri algılamaya yarayan pasif sensörlerdir. Pasif sensörlere örnek olarak kameralar gösterilebilir. Diğer sensör tipi ise; çevreyi algılayabilmek için enerji yayması gereken aktif sensörlerdir. Aktif sensörlere örnek olarak sonar, lazer, radar sensörleri gösterebiliriz. Bu sensörlerin yaydığı enerji, çevredeki cisimler tarafından sensöre geri yansıtılır ve bu sayede çevre hakkında istenilen bilgi edilebilir. Genellikle aktif sensörler pasif sensörlere göre çevre hakkında daha çok bilgi taşırlar. Ancak daha çok güç tüketirler. Bu durum enerjisini baterilerinden alan mobil robotlar için sorun teşkil edebilir.

Eğer robot gerçek dünyada insanlarla çalışıyorsa, insan gibi gerçek zamanlı olarak olaylara tepki vermesi beklenir. Bu, etrafındaki ani değişiklikleri sensörleri yardımıyla gerçek zamanlı olarak algılama ve yorumlayıp tepki üretebilmek demektir. Yapılan uygulamaya göre kimi

zaman bir sensörden gelen bilginin yorumlanmasındaki 10 dakikalık bir gecikme tehlikeli sonuçlara neden olabilir.

Ultrasonik Sensörler: Ulrasonik sinyaller çok yüksek frekansta ses sinyalleridir. Ultrasonik, ses üstü anlamına gelmektedir. Duyma sınırımız olan 300Hz-14000Hz aralığından daha yüksek frekanslı sesler ultrasonik ses olarak nitelendirilir. Bu kadar yüksek frekanslı ses dalgalarının kullanılmasının nedeni ise bu dalgaların oldukça düzgün ve doğrusal bir şekilde ilerlemeleri, taşıdığı enerjinin yüksek oluşu ve sert yüzeyli nesnelerden kolaylıkla yansımalarıdır.

Ultrasonik sensörler, genellikle, istenen frekansta titreşimler yapan piezo kristalden oluşurlar.Yayıcı ve algılayıcı iki kısımdan oluşur. Yayıcı kısım, titreşen kristalin üzerindeki elektrik enerjisini akustik enerjiye dönüştürür. Algılayıcı kısmı ise geri yansıyan ses dalgalarını benzer işlemlerin tersini yaparak elektrik enerjisine dönüştürür. Yayıcı kısmın yaydığı ses dalgaları Şekil.4.1'de görüleceği gibi koni biçiminde bir alana yayılır. Cismin algılanabilmesi için ultrasonik dalgaların geri dönebileceği kadar uzakta, yayılım ile algılama arasındaki zaman farkını algılayabilecek kadar da yakında olması gerekir. Ultrasonik sensörün çalışmasını etkileyen faktörler; hedef cismin yüzey açısı, cismin geri yayılıma izin veren bir yüzeyinin olması, sıcaklık ve nem sayılabilir.



(a) cisim algılandı (b) cisim algılanamadı

Şekil.4.1 Ultrasonik sensör ile cismin ve mesafesinin algılanması

Ultrasonik sensör ile hareketli nesneler dahil tüm cisimlerin mesafesi ve ne kadar hareket ettiği ölçülebilir. Hedef cismin yüzeyinden, yapıldığı maddeden diğer sensörlere göre çok daha az etkilenir. Cismin renginden ise hiç etkilenmez. Uzak mesafedeki çok küçük cisimleri bile algılayabilir. Ayrıca infrared radyosyonu, EMI gibi etkilere karşı direnci çok yüksektir.

Ultrasonik uzaklık ölçümü ise şu şekilde gerçekleşir; öncelikle yüksek frekanslı bir ses dalgası gönderilir ve ses dalgasının karşısındaki nesneden yansıyıp geri gelmesine kadar olan süre ölçülür. Bu sürenin sesin o ortamdaki birim hızıyla (hava içerisinde 344m/sn)

çarpılmasıyla da sesin kat ettiği yolun uzunluğu tespit edilir. bu uzunluğun yarısı da bize o nesnenin uzaklığını verir.

Bu projede, ilerleme yönünde ve çevresindeki engelleri ve bu engellerin yerlerini algılayan mobil robotumuzun en önemli dış giriş işaretini, ultrasonik sensörlerden gelen bilgiler oluşturmaktadır (Çayurpınar, 2005). Ultrasonik sensörler maliyet açısından uygun bir çözüm getirir. Ancak robotun çevresindeki daha geniş bir alan kapsanmak isteniyorsa ikiden daha fazla sayıda ultrasonik sensör kullanmamız gerekir. Bu durum ise sistemin çok daha yavaş çalışarak algılama ve tepki verme sürelerinin çok artmasına sebep olacaktır. Gerçek zamanlı çalışması beklenen bir robot için bu istenmeyen bir durumdur. Eğer daha rezolüsyonu yüksek ve daha uzak mesafeleri kapsayan ölçüm yapmak istiyorsak biraz maliyeti arttırmayı göze alarak lazer sensör kullanılabilir. Ultrasonik sensörlerde birkaç santimden, birkaç metreye kadar çok hassas olmayan bir uzaklık bilgisi alınırken, lazer sensörler ile birkaç milimetreden başlayarak onlarca metre ötedeki cisimlerin bile tam yeri tespit edilebilir. Yapılacak uygulamaya göre bu seçim önem taşır.

Şekil.4.2 Ultrasonik uzaklık ölçümünde kullanılabilecek 40KHz'lik alıcı ve verici sensör çiftleri

4.3 Robot Efektörleri

Efektör, uyarıya karşılık yanıt veren organ anlamına gelmektedir. Canlılarda efektör işlemini kaslar yapar. Kaslar sayesinde örneğin hareket eder, konuşur yada yüz mimikleri üretiriz. Bunların her biri dış ortama verilen tepkilerdir.

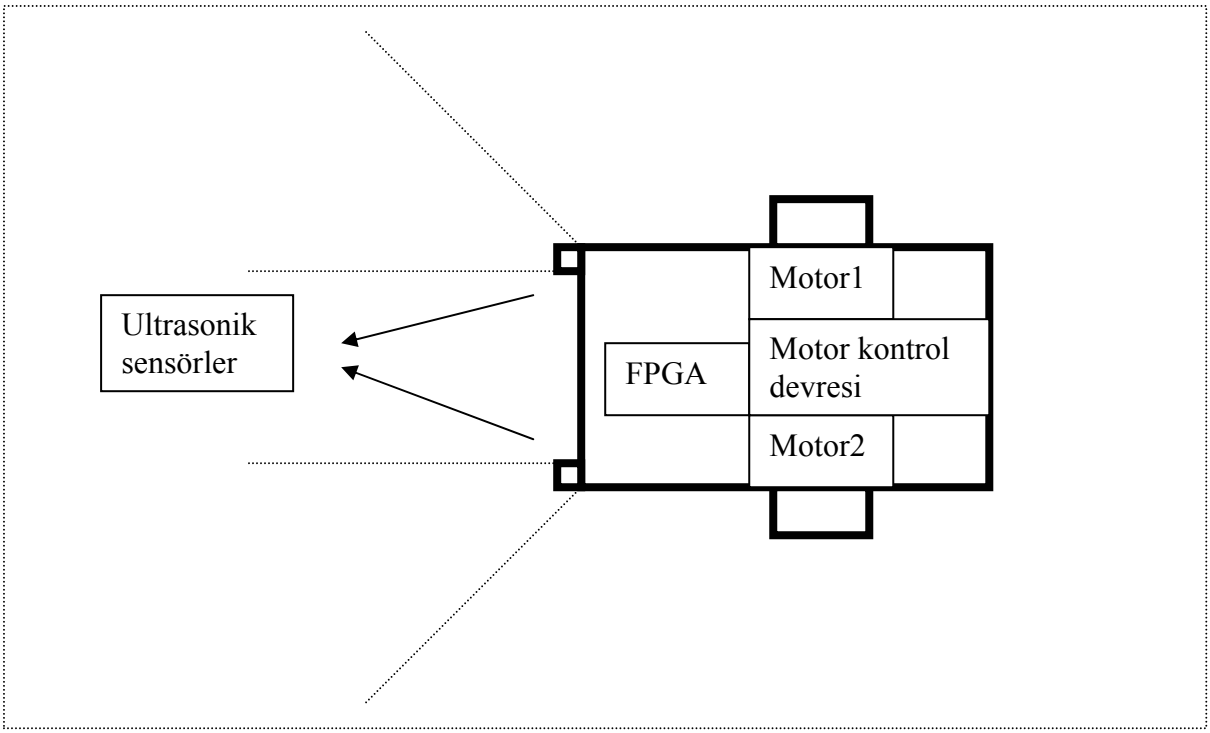
Robot davranışının sonraki aşaması çevreyi değiştiren bir eylem içerir. Efektörler, algılama ve biliş sistemlerinin görevlerinin tamamlanmasından sonra genellikle bir hareket başlatan, enerji aktarımı ve değişimi içeren, önceden belirlenmiş bir amaca yönelik olarak çevreyi değiştirebilen cihazlardır. Yapı olarak efektörler de sensörler gibi transduser yapısında olup,

kendilerine gelen bir enerjiyi başka bir enerji türüne dönüştürürler. Sensör seçiminde olduğu gibi efektör seçiminde de göz önüne alınması gereken bir çok etmen vardır. Bu etmenler; çalışma koşullarına göre hızın denetimli olması, kısa tepki süresinin olması, işlem gücünün yüksek olması gibi etmenlerdir. Robotlar, sensörlerden dış ortamdan gelen girişleri alır, bunları yorumladıktan sonra ne çıkış vereceğine karar verir ve efektör cihazlar yardımıyla dış ortama bir tepki verir. Genellikle efektör olarak, DC yada step motorlar tercih edilir. Bu çalışmada modellediğimiz mobil robotta da efektör olarak iki adet DC motor kullanacağız.

4.4 Mobil Robot

Genel olarak robotların kullanım alanları insanoğlunun gücünün yetmediği yada tehlikeli gördüğü iş sahalarıdır. Örnek olarak; radyoaktif malzemelerle çalışılan laboratuarlar, bir otomobil üretim bandı yada bir Mars yolculuğu verilebilir. Bu çalışmada tasarlanması planlanan model robotu gerçekleştirmek için iki adet motor kullanılmıştır. Bu motorların hızları ayarlanarak, istenilen miktarda sağa ve sola dönme işlemleri gerçekleştirilebilir. Mobil robot ilerlerken ulrasonik sensör çifti ile yolu üzerindeki engellere olan mesafesini algılayacak ve onlara çarpmayacaktır. Aynı zamanda bu engellerin koordinatlarını belleğine yazarak bulunduğu ortamı haritalayacaktır.

Model robotun bir şeması Şekil.4.3'de görülmektedir.



Şekil.4.3 Model robotun blok diyagramı

Robotun verebileceği tepkiler önceden belirlenmiş ve eeprom'una yüklenmiştir. Hangi tepkiyi vereceğine, sensörlerden aldığı bilgiyi işledikten sonra bir look-up-table yardımıyla elde eder.

Klasik mikroişlemcilerle bu algoritmaları gerçekleştirmek kolay olmayacaktır. İşlemci paralel bilgi işleyemediğinden robot gerçek bir insan beyninin düşündüğü gibi düşünemeyecek, harici bir bellek elemanına ihtiyaç duyulacak ve hem seri çalışıyor olması hem de bellek elemanının dışarıda bulunması yavaş çalışmasına neden olacaktır. Robotlar genellikle mobil olduklarından dolayı üzerlerinde güç kaynağı olarak sadece batarya bulundururlar. Klasik işlemcilerin harici elemanlarla birlikte daha çok güç harcaması mobil robotlar için ayrı bir problem olacaktır. Bu duruma FPGA teknolojisi çok etkili bir çözüm getirir. FPGA, paralel işlem yeteneği sayesinde verileri insan beyni gibi işler, hızlı çalışır. Bellek elemanını içerisinde barındırdığı için ekstra zaman, enerji kaybına ve maliyete yol açmaz. Düşük güç tüketimli oldukları için mobil robotlar için çok efektif çözüm sunarlar.

Öğrenme Ünitesi: Öğrenme ünitesi, ağırlıkların nasıl değişeceğini bildirdiği için bu ismi almıştır. Robot çalışırken ultrasonik sensörler çevreye sinyal yayarlar. Sinyal geri alındığında geri dönme süresi hesaplanır ve en yakın engelin ne kadar mesafede olduğu bu süreden hesaplanır. Engelin robota olan uzaklığını FPGA'e dijital olarak vermek zorundayız, bu

yüzden engelin robota olan uzaklığını 1'den 7'ye kadar sayılarla ifade edelim. Yani 3 bit olarak kodlayalım. Hiçbir yansıma olmuyorsa 7 ile kodlayalım yani 7'nin anlamı görünürde hiç bir engelin olmadığı olsun. 4'den düşük değerler de engelin robota çok yakın olduğu anlamına gelsin. Burada dikkat edilmesi gereken nokta iki sensörün farklı zaman dilimlerinde çalıştırılmasıdır. Aksi halde bir sensörün yaydığı ultrasonik dalgaları diğer bir sensör algılayabilir ve bu engeli algılamada karışıklığa sebep olabilir. Bu yüzden ultrasonik sensörler tek tek aktif edilmelidir. Burada mesafe ölçüm hassasiyetini arttırmak için lineer yerine logaritmik yaklaşımı tercih edebiliriz. Bu yaklaşım sayesinde yakın olan cisimleri algılama hassasiyetini arttırabiliriz.

Motorların durumlarını da yine FPGA'e dijital olarak tanımlamak zorundayız. Her iki motorun da durum bilgisini ayarlamak için FPGA 3 bit kodlanmış çıkış bilgisi verir. Bu kodlama, robotun hangi yöne kaç derece döneceğini söyler. Robotun iki yanına yerleştirilen bu motorlar sayesinde sensörlerden alınan bilgilere göre robot; farklı hızlarla ileri, geri gidebilir yada sağa, sola dönebilir.

4.5 Robotikte Yapay Sinir Ağlarının Kullanımı

Robotlar insanları taklit insansı özelliklere sahip makineler olduklarına göre insansı tepkiler vermesi beklenir. Bu tepkileri vermeden önce insan gibi çevresindeki değişiklikleri algılaması, bunlarla ilgili yorum yapması ve bir tepki üretmesi gerekir. Yapay sinir ağları beynin bazı fonksiyonlarını ve özellikle öğrenme yöntemlerini benzetim yolu ile gerçekleştirmek için tasarlandığı için, geleneksel yöntem ve bilgisayarların yetersiz kaldığı duyu-veri işleme, karar verme, tahmin etme gibi insansı özellikleri gerçekleştirmede çok başarılı sonuçlar verir. Daha önceki bölümlerde yapay sinir ağlarının giriş katmanı, gizli katmanlar ve çıkış katmanlarından oluştuğundan bahsetmiştik. Yapay sinir ağlarından faydalanarak geliştirilen bir robot uygulamasında hangi mimari yapı seçilirse seçilsin bu katmanlar yer alacaktır. İnsanların sinir ağlarında dışarıdan girişler duyu organları vasıtasıyla olur. Duyu organlarındaki alıcılar ise koku, görüntü, ses, ısı, basınç,… gibi girişleri alıp diğer sinir hücrelerine yayarlar. Robotlarda ise dış ortamdaki değişiklikleri algılama işlevi sensörler ile gerçekleştirilir yani robotlarda yapay sinir ağlarının giriş katmanını sensörler oluşturur. İnsan sinir sistemi alınan veriyi beyinde işledikten sonra hangi tepkiyi yapacağına karar verir ve bu karar sonucunda gerekli tepkiyi üretmek üzere kaslara sinyal gönderir. Kaslar da çıkış tepkisi üretilir. Burada beyin ara katmanı, kaslar ise çıkış katmanını oluşturmaktadır.

Robotlarda ise alınan sinyaller bir işlemcide işlenir ve yapay sinir ağlarının bir tekniği ile çalışan bir yazılım ya da donanımla karar verme işlemi gerçekleştirilir. Karar verildikten sonra çıkış sinyali ilgili motora aktarılır. Burada işlemci ara katmanı, motorlar ise çıkış katmanını oluşturmaktadır.

Çizelge.4.1'de klasik bir bilgisayarın çalışma şekli ile yapay sinir ağları ile donatılmış öğrenebilen bir robot beynin karşılaştırması yapılmıştır.

Çizelge.4.1 Öğrenebilen makineler ile klasik bilgisayar sistemlerinin karşılaştırılması

ÖĞRENEBİLEN ROBOT	KLASİK BİLGİSAYAR
Kararlar alır.	Sonuçlar hesaplar.
Sezgisel yöntemlere dayanır.	Algoritmalara dayanır.
Daha esnek.	Daha az esnektir.
Belirsizliği ele alabilir.	Belirsizliği ele alamaz.
Kısmi bilgi, tutarsızlıklar ve kısmi kanaatlerle çalışabilir.	Komple bilgiye ihtiyaç duyarlar.
Sorunların açıklamalarını sağlayabilirler.	Sonuçları açıklamasız verirler.
Sembolik muhakeme yapabilirler.	Sayısal hesaplamalar yaparlar.
Kontrol ve bilgi ayrılmıştır.	Kontrol ve bilgi içiçedir.

4.6 Haritalama

Hareketli robotumuzun kullanıldığı uygulamalarda, robotun kendisine söylenen bir kaynak bölgeden yola çıkıp kendisine bir yörünge belirleyerek ve engellere çarpmayarak hedef bölgeye ulaşması istenebilir. Örneğin bu robotun kullanılacağı bir uygulamada gidilecek bölgeler için adres bilgisi olarak bir numara atanır. Robota bulunduğu nokta ve gideceği adres söylendiğinde robot ilk olarak bu iki nokta arası için çizilmiş bir yol haritası bilgisinin belleğinde olup olmadığını sorgular, eğer belleğinde yoksa çevrim içi olarak bu yolu öğrenecek ve belirtilen hedef adrese ulaşacaktır. Böyle bir uygulama akıllı engelli sandalyesi uygulamalarında büyük bir gelişmeye yol açacaktır.

Mobil robot kullanılarak yapılacak kontrol uygulamalarında karşılaşılan ilk problemler robotun nerde olduğu, nereye gideceği ve hangi yolla gideceğiydi. Burada ilk problem kendi konumunu tespit etmek (localization) ile ilgilidir. Diğer problemler ise konumlanacağı noktanın tespiti ve yön belirleme ile ilgilidir. Haritalama üç farklı şekilde gerçekleşebilir: Global haritalama, lokal haritalama, hibrit haritalama. Global haritalamada ortam hakkında bütün bilgiler robota verilmiştir. Lokal haritalamada, robot dış çevre hakkında bilgiye sahip değildir ve sensörleri yardımıyla öğrenir. Hibrit haritalama metotlarında ise optimal planlama yapılmıştır, ancak robot sensörlerinden aldığı verileri de değerlendirerek izleyeceği yolu belirler (Zarate, Becker, Garrido, Rocha, 2002). Global haritalama yöntemi, ortamın tamamen statik olduğu durumlarda hızlı ve iyi çözüm getirebilir. Ancak dinamik bir çevrede hiçbir nesnenin yerinin değişmeyeceğini söylemek imkansızdır. Bu yüzden lokal ve hibrit yöntemler tercih edilir.

Bu çalışmada robota başlangıç konumu ve gideceği nokta hakkında bilgiler verilmiştir. Ancak robot dinamik bir ortamdadır. Ve izleyeceği optimal yol üzerinde önüne engeller çıkabilecektir. Robotun üzerindeki ultrasonik sensörler, robotun karşısına çıkabilecek engelleri görmesi ve yeni bir planlama işlemi gerçekleştirmesi için yeterli olacaktır.

4.7 Robotun Öğrenme Algoritması

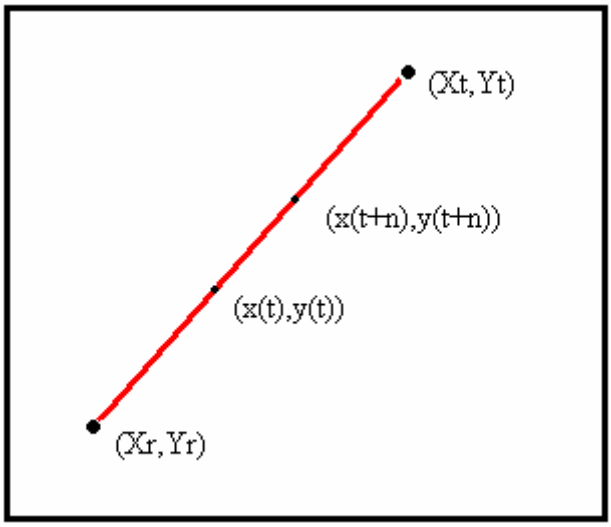
Haritalama ile ilgili bölümde bahsettiğimiz gibi; haritalama işlemini dinamik ortamlarda gerçekleştirebilmek için seçilecek yöntemlerden biri lokal haritalama diğeri ise hibrit haritalama yöntemidir. Bu çalışmada hibrit haritalama yöntemi kullanılarak bir odada yön bulma işleminin gerçekleştirilmesi hedeflenmiştir. Yani model olarak alınan odanın ebatları, robotun hangi noktadan harekete başladığı ve hangi koordinatlardaki noktaya ulaşması gerektiği bilinmektedir. Robot öncelikle iki nokta arasındaki en kısa mesafe olan doğruyu belirlemekte ve bu doğru üzerindeki (x,y) değerlerini istenen çıkışlar olarak almaktadır. Ancak sensör girişlerinden alınan verilere göre eğer robotun yolu üzerinde bir engel varsa yapay sinir ağından alınan sonuca göre tekrar konumlanacak ve istenen çıkışa yaklaşacak şekilde yeni yolunu belirleyecektir. Model olarak alınan odanın 4mx4m boyutlarında olduğunu kabul ettik. Bu odayı da haritalayabilmek için x ve y koordinatlarında parçalara bölerek odayı 160x160 büyüklüğünde bir konum matrisi olarak ele aldık. Bu durumda her bir adım hassasiyeti x ve y koordinatlarında 2,5cm'dir. Oda daha sık aralıklı parçalara ayrılarak hareket hassasiyeti istenilen miktarda arttırılabilir, ancak matrisin çok fazla büyümesi

durumunda hafızada saklanması gereken konum sayısı da aynı ölçüde artacaktır. Bu ise; sistemin daha yavaş çalışmasına neden olacak ve maliyeti arttıracaktır.

Bu çalışmada kullanılan yapay sinir ağını aşağıdaki gibi modelledik. Bu yapıda robot on-line olarak dinamik ortamı öğrenmekte ve hedefe en kısa yoldan ulaşmayı amaçlamaktadır. Robotun tüm oda boyunca haritaladığı konumlar aşağıdaki gibi ifade edilmektedir.

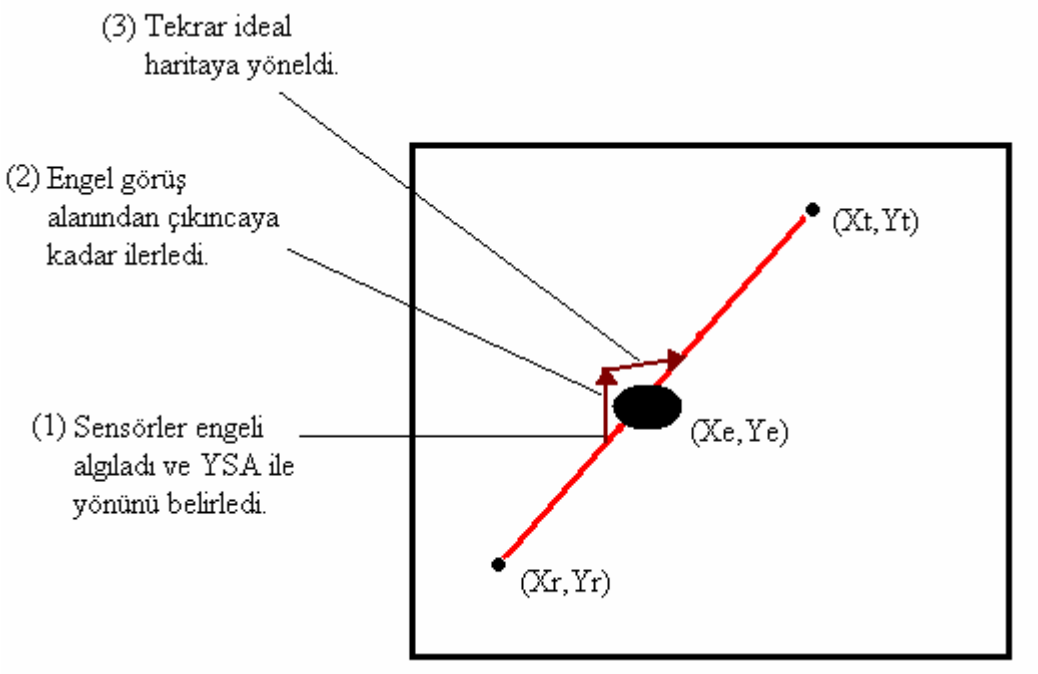
$$c(X,Y) = \{(x,y)|\ x\varepsilon R \geq 0,\ y\varepsilon R \geq 0\} \qquad (4.1)$$

Bu eşitlikte bahsedilen (x,y)'ler 160x160 büyüklüğündeki matrisi oluşturmaktadır ve koordinatlarının odanın hangi noktalarına karşılık geldiği bilinmektedir. Robotun harekete başlama noktası (Xr,Yr) bilinmektedir ve robotun ulaşmasını istediğimiz (Xt,Yt) noktası da robota verilmektedir. Bu iki nokta arasındaki en kısa yol aşağıdaki gibi bir doğrudur.



Şekil.4.4 Robotun (Xr,Yr) noktasından, (Xt,Yt) hedef noktasına konumlanması

Ancak bu yol üzerinde bir engel bulunduğunu düşünelim.



Şekil.4.5 Robotun (Xe,Ye) noktasında bulunan engelle karşılaşması

Bunun için modellenen sinir ağı yapısı aşağıdaki gibidir. Buradan da görüldüğü gibi iki adet yapay nöron hücresi ile mimari tasarlanmıştır. Her bir nöron hücresi giriş olarak 5 değişken almakta ve buna karşılık bir çıkış (x yada y ekseninde yeni koordinatları) vermektedir.

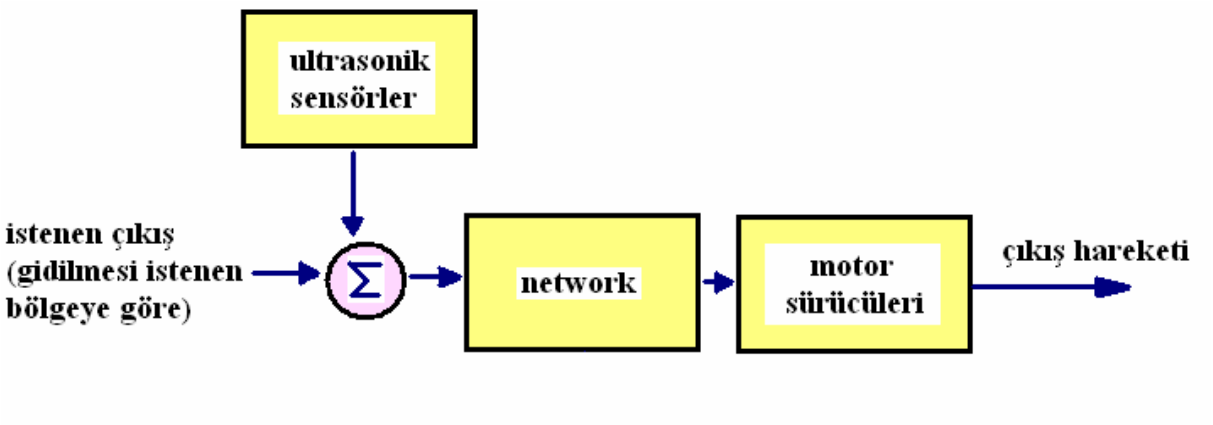
$$x'(t+1) = f(x(t-1), x(t), x(t+1), S1, S2) \quad (4.2)$$

$$y'(t+1) = f(y(t-1), y(t), y(t+1), S1, S2) \quad (4.3)$$

Burada S1 ve S2; sırasıyla sağ ve sol sensörlerden alınan bilgilerdir. x(t-1) ve y(t-1) robotun bir önceki konumu, x(t) ve y(t) robotun şuan ki konumu, x(t+1) ve y(t+1) ise olması istenen konumdur. x'(t+1) ve y'(t+1) bir sonraki adımda bulunacağı koordinatlardır. Sensörlerden alınan S1 ve S2 bilgileri engellerin yerini işaret ettiği için (Xe,Ye) olarak da gösterilebilir. Bir sonraki adımdaki bulunacağı koordinatları sadece bir önceki ve bir sonraki istenen (x,y) bilgilerine bakarak bulduğu için sistem birinci dereceden haritalama yapmaktadır diyebiliriz. Sistemin k sayıda önceki hareket noktalarına ve bundan sonra bulunması istenen k adet koordinat noktasına bakarak çıkış üretecek şekilde k. dereceden bir haritalama ağı da modellemek mümkündür. Böyle bir yapıyla daha iyi bir kestirim yapmak mümkün olabilir, ancak donanımsal olarak karmaşıklığı çok fazla arttıracağı bir gerçektir.

Robotun öğrenme algoritmasını çevrimiçi (online) yada çevrim dışı (offline) olarak modelleyebilmek mümkündür. Hangi modelin seçileceği yapılan uygulamanın amacına bağlı olarak değişir. Genellikle yapılan uygulama ve ortam şartlarında çok fazla değişiklik olmuyorsa çevrimdışı çalışma şekli tercih edilir. Çevrim dışı çalışma yöntemi hızlı çözüm verir, ancak ortam şartlarında değişikliklerin olması durumunda verimi çok hızlı şekilde düşeceği göz önünde bulundurulmalıdır. Eğer ortam şartları sıklıkla değişiyor ve robotun bu yeni şartlara adapte olması gerekiyorsa çevrimiçi çalışma tercih edilmelidir. Çevrimiçi çalışırken robot çalışma anında gerçek zamanlı olarak değişen şartlara göre ağırlık değerlerini değiştirebilir ve yeni şartlara adapte olmayı öğrenebilir.

Robotun kendi kendine öğrenme algoritması aşağıdaki diyagramda gösterilmiştir. Buradan da görüleceği gibi; istenen çıkış yanıtı, ultrasonik sensörlerden gelen bilgilerle birlikte sinir ağında değerlendirilir. Ağın çıkışı , robota istenen hareketi yaptırmak üzere motor sürücülerine verilir.



Şekil.4.6 Robot blok diyagramı

5. ÖĞRENME ALGORİTMASININ TASARIMI

YSA uygulamasının başarısı, uygulanacak olan yaklaşımlar ve deneyimlerle yakından ilgilidir. Uygulamanın başarısında uygun metodolojiyi belirlemek büyük önem taşır. Yapay sinir ağının geliştirilmesi sürecinde ağın yapısına ve işleyişine ilişkin şu kararların verilmesi gerekir:

1- Ağ mimarisinin seçilmesi ve yapı özelliklerinin belirlenmesi (katman sayısı, katmandaki nöron sayısı gibi),
2- Nörondaki fonksiyonların karakteristik özelliklerinin belirlenmesi,
3- Öğrenme algoritmasının seçilmesi ve parametrelerin belirlenmesi,
4- Eğitim ve test verisinin oluşturulması

Bu kararların doğru verilmemesi durumunda, YSA'ların sistem karışıklığı artacaktır. Sistem karmaşıklığı yapısal ve toplam hesaplama karmaşıklığının bir fonksiyonudur. Toplam hesaplama karmaşıklığı ise; genellikle yapısal karmaşıklığın bir fonksiyonu olarak ortaya çıkar ve bu hesaplamanın en aza indirilmesi amaçlanır. Bir YSA'nın uygun parametrelerle tasarlanması durumunda YSA sürekli olarak kararlı ve istikrarlı sonuçlar üretecektir. Ayrıca sistemin tepki süresinin yeterince kısa olabilmesi için de ağ büyüklüğünün yeterince küçük olması gerekir. İhtiyaç duyulan toplam hesaplama da bu sayede sağlanmış olacaktır.

5.1 PNN Ağları

PNN ağlar, aşağıdaki özelliklerinden dolayı bu ve benzer çalışmalarda kullanılabilecek en iyi ağ yapılarından biridir.

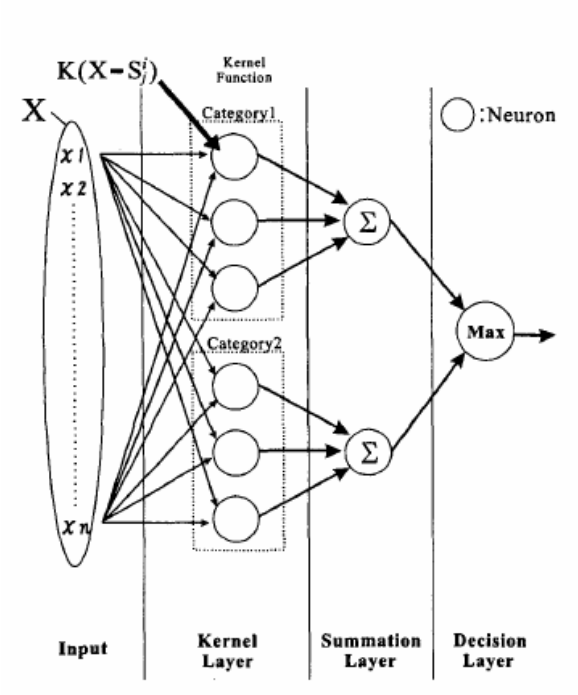
* 1966 yılında Donald Specht tarafından ortaya atılan PNN yada "Probabilistic Neural Network", gizli katmandaki nöron sayısı eğitimde sunulan örnek sayısına bağlıdır. Bu gizli katmanın ağırlıkları ya sıfır yada birdir. Bu yüzden register'larda tutulacak ağırlık değerleri işlem hızını düşürecek virgüllü sayılar değildir.
* Bir kere eğitildikten sonra eğiticisiz çalışabilir yada eğitime devam edilebilir.
* Seçilen olasılık yoğunluk fonksiyonuna göre karar düzlemi oluşturulabilir.
* Hızlı eğitilebilir. Hızlı yanıt üretir.
* Sınıflama uygulamalarında başarımı çok yüksektir.

PNN (Probabilistic Neural Network - PNN), olasılıksal sinir ağları olarak bilinir. Literatürde, Bayes-Parzen kestiriciler olarak da anılırlar. K1 ve K2 sınıflarından birine ait, m-boyutlu bir x vektörü olsun. K1 ve K2 sınıflarına ait olasılık yoğunluk fonksiyonları F1(x) ve F2(x) olsun. Bayes teoremine göre x vektörü;

$$\frac{F_1(x)}{F_2(x)} > \frac{L_1 P_2}{L_2 P_1} \tag{5.1}$$

eşitsizliği doğru ise K1, eşitsizliğin tersi doğru ise K2 sınıfına aittir. Burada P1 ve P2, K1 ve K2 sınıflarının görülme olasılığıdır.L1, x vektörünün K1 sınıfına ait iken K2 olarak yanlış sınıflama oranı; L2 ise x vektörünün K2 sınıfına ait iken K1 olarak yanlış sınıflandırma oranıdır ve maliyet fonksiyonu olarak adlandırılır. Buradan görüleceği gibi, F1(x), F2(x), L1 ve L2'nin bilinmesi durumunda x vektörünün en yüksek olasılıkla hangi sınıfa ait olduğu tespit edilebilir.

PNN ağı Şekil.5.1'de görüleceği gibi 2 gizli katmandan oluşan, geri beslemesiz bir ağ mimarisine sahiptir. Giriş katmanında test vektörünün özelliklerini belirten girişler alınır. Bu girişler her bir eğitim örneği ile tek tek karşılaştırılır ve (1) ile belirtilen benzerlikler hesaplanır. Toplam katmanında giriş vektörünün her bir sınıfa benzerliğinin ortalaması alınır. Bu işlem (2) ifadesindeki gibi gerçekleştirilir. Bu ağ mimarisinin en büyük dez avantajı; giriş vektörünün eğitim örneklerinin her biriyle tek tek karşılaştırılması gerektirdiği için eğitim örneği kadar gizli nörona ihtiyaç duymasıdır. Eğitim örneğinin çok olduğu durumlarda bu durum ağ karmaşıklığına ve hesaplama süresinin uzamasına sebep olacaktır.



Şekil.5.1 PNN ağ mimarisi

Vektörün yaklaşıklığını bulmanın en iyi yolu Gauss fonksiyonunu kullanmaktır.

$$K\left(\mathbf{X}-\mathbf{S}_j^i\right)=\exp\left(\frac{-(\mathbf{X}-\mathbf{S}_j^i)^T(\mathbf{X}-\mathbf{S}_j^i)}{2\sigma^2}\right) \tag{5.2}$$

$$\hat{p}\left(\mathbf{X}|C_i\right)=\frac{1}{N_p}\sum_{j=0}^{N_p}K\left(\mathbf{X}-\mathbf{S}_j^i\right), \tag{5.3}$$

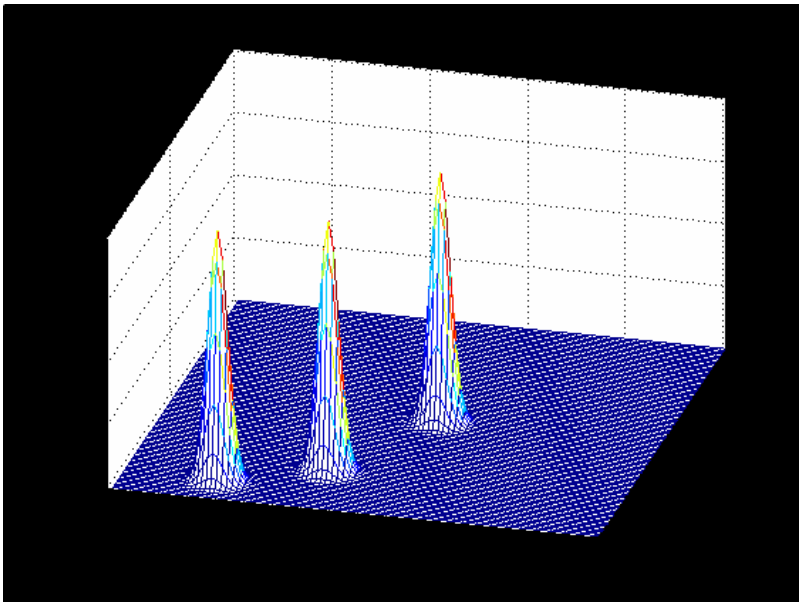
Ancak bu fonksiyon içindeki exponansiyel ifadesi donanımsal olarak gerçekleştirmede problem yaratır. Exponansiyel ifadenin olası değerlerini saklamak için büyük bir LUT (Look-up-table) tutmak gerekir. Bu problemden sakınmak için donanımsal olarak gerçekleştirmelerde yaklaşıklık fonksiyonu olarak aşağıda görülen Parzen pencere fonksiyonu kullanılabilir. PNN'lerde sınıflara ait yoğunluk fonksiyonları Parzen pencereleri kullanılarak aşağıdaki şekilde bulunur.

$$K\left(\mathbf{X}-\mathbf{S}_j^i\right)=\begin{cases}1: & \left|x_k-(s_j^i)_k\right|\le\frac{\sigma}{2}\\ 0: & \text{otherwise}\end{cases}$$
$$(\text{for all } k,k=1,2,\ldots,n) \tag{5.4}$$

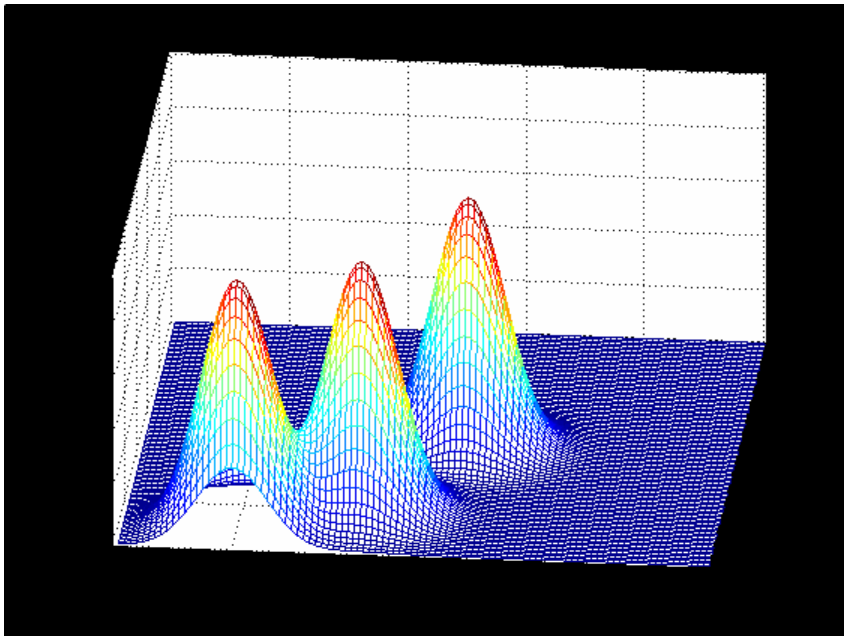
Burada σ sabittir, d(x,yi) saklanmış yi örneği ve x test örneği arasındaki uzaklıktır. P, çok değişkenli örnek vektöründeki bileşenlerin sayısıdır ve n ise; verilen kategorideki eğitilen kategorilerin sayısıdır. Bu formülasyonun avantajı şudur ki; birkaç örnek olasılık yoğunluk fonksiyonu temelini teşkil eden, esnek bir tahmin yapmak için gereklidir.

Öncelikle her giriş vektörü ile eğitim vektörleri arasındaki öklit uzaklığı hesaplanır. Gauss doğrusalsızlığı bu uzaklık ölçümü ile uygulanır. Toplama katmanındaki her bir birim, verilen bir sınıfın örüntü katman çıkışlarını içerir ve ölçekleme bu kategoride eğitme örneklerinin sayısının tersi ile toplanır. Toplama katmanının çıkışları tahmin edilen şart yoğunluklarıdır. Son katman maksimum olasılığın kategorisini belirleyen karar katmanıdır.

Şekil.5.2'de yukarıda verilen F(x) olasılık fonksiyonunun bir test kümesi için aldığı değerler birinci şekilde σ = 2 için, ikinci resimde ise σ = 6 için görülmektedir. Bu örnekte eğitim setinin farklı üç çıkışı vardır. Yatay düzlem test girişlerini, düşey düzlem ise bu girişlerin yakınında bulunduğu eğitim örneğinin sınıfına ait olma olasılığını göstermektedir.



(a)



(b)

Şekil.5.2 σ sabitinin ayarlanarak kümelerin dağılımına göre olasılık yoğunluk fonksiyonlarının belirlenmesi (a) σ = 1 (b) σ = 5

Bu incelemeden de anlaşılacağı gibi σ ifadesi pencerenin genişliğini belirlemektedir. Eğitim kümelerinin birbirine yakınlığına göre σ değerinin ne olacağına önceden karar vermeliyiz.

5.2 Dezavantajların Kaldırılması için Yapılan Çalışma

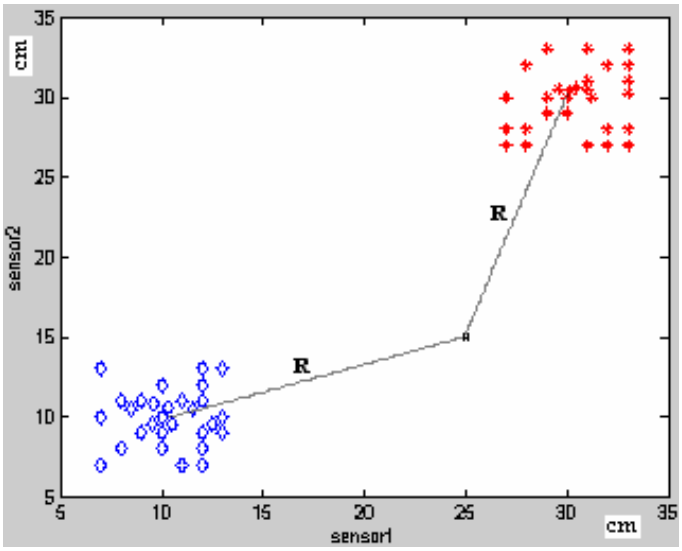
Mobil robot için sensör giriş verilerinden yola çıkarak karar verme işlemi yukarıda detaylı olarak açıklanan klasik PNN ile gerçekleştirildiğinde yanlış sınıflandırmalara yol açabilmektedir. Klasik PNN algoritması bir test vektörünün ait olduğu sınıfı belirlerken test vektörünün eğitimde kullanılan örneklere olan uzaklığına ve eğitim örneklerinin σ yumuşatma sabitine göre karar vermektedir. σ her eğitim örneği için sabit seçildiğinden yalnızca eğitim örneklerine olan R mesafeleri belirleyici olmaktadır. Ancak robot sensörlerinden gelecek olan (1,7) yani sol sensörden gelen bilgi 1, sağ sensörden gelen bilgi 7 olduğunda ve (7,1) yani sol sensörden gelen bilgi 7, sağ sensörden gelen bilgi 1 olduğunda hesaplanacak R mesafeleri aşağıda gösterildiği gibi birbirine eşittir.

$$\|(1,7)\| = \sqrt{1^2 + 7^2} = \sqrt{50} \tag{5.5}$$

$$\|(7,1)\| = \sqrt{7^2 + 1^2} = \sqrt{50} \tag{5.6}$$

Bu durumda klasik PNN algoritması test vektörünü herhangi bir kümeye atayabilir. Ancak robotun içinde bulunduğu durumu yakından inceleyecek olursak bu durum çok tehlikeli sonuçlar doğurabilir. Örneğin sensörlerden gelen bilginin (1,7) olması sağ sensöre çok yakın bir engel olduğunu belirtir ve sola ani bir dönüş yapmasını gerektirirken, (7,1) sensör girişi sol sensöre çok yakın bir yerde engel olduğu ve ani bir şekilde sağa dönüş yapılması gerektiği anlamına gelmektedir. Bu test girişinin yanlış sınıflandırılması robotun engelin üzerine yönelmesine yol açabilir. Bu dezavantajı ortadan kaldırmak için bu öğrenme algoritmasında karar katmanında en büyük benzerliği bulan algoritmanın yanı sıra giriş vektörünün ağırlıklarını karşılaştıracak bir algoritma eklenmiştir. Örneğin (s1,s2) sensör girişi ağa geldiğinde ağın eğitim örneklerine yakınlığı hesaplanır. Bundan sonra R yakınlığının benzerliği ve aşağıdaki gibi tanımlanan ağırlık benzerlikleri karşılaştırılır. R ve α değerleri en çok benzeyen kümenin çıkışı, test vektörünün çıkışı olarak kabul edilir.

$$\alpha = \arctan(s2 / s1) \tag{5.7}$$



Şekil.5.3 Yanlış sınıflandırmaya sebep olabilecek iki kümeye eşit uzaklıkta bir test örneği

Kaldırılan diğer bir dezavantaj ise gizli nöron sayısının azaltılmasına yöneliktir. Örneğin her ikisi de 50 adet eğitim örneği içeren iki adet küme içeren bir ağa test örneği geldiğinde hangi kümeye ait olduğunu hesaplamak için toplam 100 (50+50) adet örnek ile karşılaştırılması gerekir. Bu çalışmada bu dezavantajı ortadan kaldırmak için her kümenin merkezi hesaplanmış ve her küme için sadece merkezi temsil eden bir tek gizli nöron kullanılmıştır.

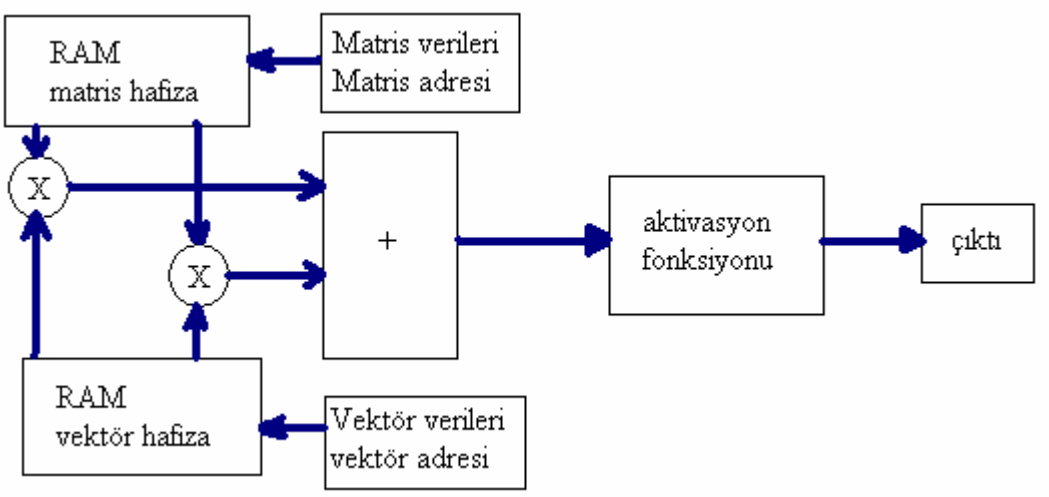
Böylece 2 küme içeren yukarıdaki gibi bir örnek ağ için 100 adet karşılaştırma yerine 2 adet karşılaştırma yaparak sınıflandırmayı gerçekleştirmek mümkün olmaktadır.

6. LOJİK OLARAK ÖĞRENME ALGORİTMASININ GERÇEKLENMESİ

6.1 Yapay Sinir Ağlarının Donanımsal Olarak Gerçeklenmesi

Günümüzde yapay sinir ağı uygulamalarının çoğu yazılım teknolojisi olarak görülmektedir. Belirli bir modelin yazılımı gerçeklenmekte ve seri bilgisayarlarda çalıştırılarak sorunların çözülmesi istenmektedir. Yapay sinir ağlarının paralellik gibi önemli özelliklerinin gösterilebilmesi için özel donanım teknolojisine ihtiyaç vardır. FPGA gibi paralel çalışan çip teknolojilerinin geliştirilmesi, bu öğrenebilen ağların donanım ile gerçekleştirilmesine olanak tanımıştır.

Dijital teknolojinin verilerin gürültüden ayrılmış olması, ağırlıkları saklamak için RAM kullanılması, çarpma ve toplama işlemlerinde duyarlılığın (doğruluk derecesinin artması) ve özelliklede mevcut sistemlere entegrasyonun sağlanması gibi bir çok avantaj sağlar. Şekil.6.1.'de dijital bir yapay sinir ağının donanım yapısı genel olarak görülmektedir.



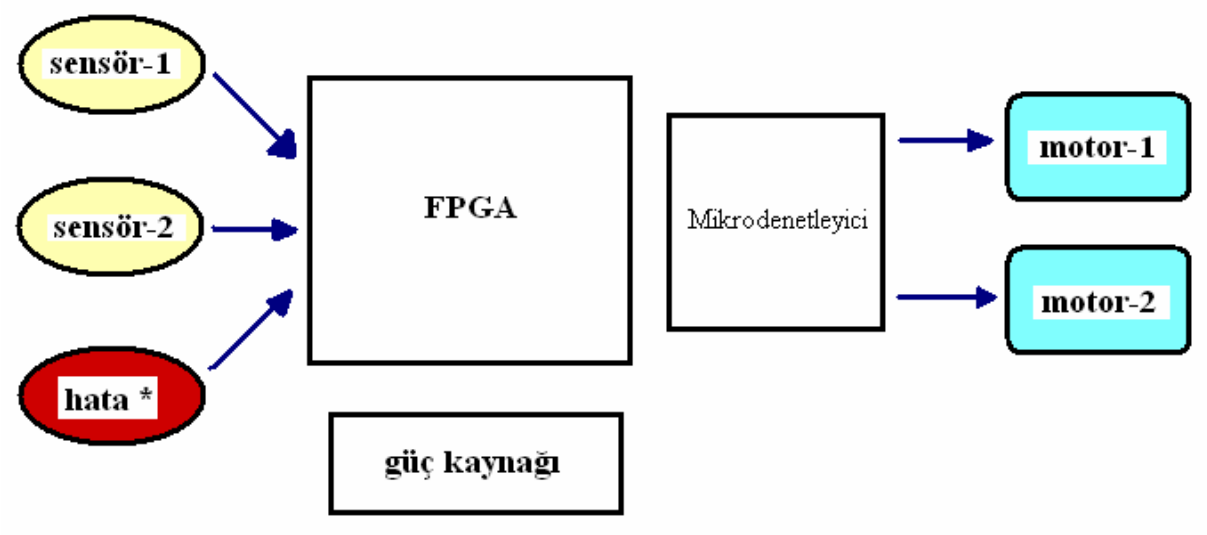
Şekil.6.1 Dijital bir yapay sinir ağı mimarisi

Özel amaçlı YSA donanımları günümüz teknolojisinde etkili gerçeklenmeli ve dikkatlice eğitilmelidir. Herhangi bir YSA gerçeklenmesini iki safhada inceleyebiliriz. Öğrenme aşaması; topoloji ve ağırlıkların geriye yayılım algoritmaları kullanarak kararlı hale getirilmesi. Sonradan düzeltme veya test etme safhası veya test etme safhası ki bu safhada ağ eğitimi sırasındaki ağırlıklarıyla ve yapısında sakladığı temel bilgilerle örnekleri tanır. Bu şekilde düşünülürse YSA gerçeklenmesi dağılmış öğrenme ve geri dönüşüm safhasıdır.

Öğrenme aşamasındaki alt yapı ihtiyaçları ve hesaplamalar test etme aşamasındaki alt yapı ihtiyaçları ve hesaplamalara göre çok daha karmaşıktır. Donanımdaki ağ parametreleri öğrenme safhasında sabitlenir. Öğrenme sonrası, bu parametreler test görevi için donanıma yüklenir.

6.2 Kullanılan Öğrenme Algoritması

Bu uygulamayı yaparken robotumuzun iki adet ultrasonik sensörü olduğunu düşündük. Daha ileri algılama niteliklerine sahip bir robot dış dünyayı algılamak için dijital kameralar, lazerler gibi daha farklı sensör cihazları kullanabilir. Ancak bu uygulama için sadece çevredeki cisimlerin robota olan mesafelerini ölçen iki ultrasonik sensörün bilgi toplamak için yeterli olacağını düşündük. Robotun dış dünyaya tepkilerini vermesi içinde her iki yanında da ileri geri istenilen hızda hareket ettirilebilen birer motor bulunmaktadır. Motorlardan birinin durup diğerinin hareket etmesiyle yada birinin ileri giderken diğerinin geri gitmesiyle robotun sağa yada sola istenilen şekilde yönelebilmesi de mümkündür. Şekil.6.2'de model robot üzerinde bulunan temel elemanlar gösterilmiştir.



Şekil.6.2 Mobil robotun blok diyagramı

Sensör-1 ve sensör-2; yukarıda bahsettiğimiz robotun sağında ve solunda bulunan ultrasonik sensörlerdir. Gelen verileri FPGA ile değerlendireceğiz, hatadan faydalanarak robotun öğrenmesini sağlayacağız ve çıkışları motor-1 ve motor-2 ile gösterilen DC motorlara

aktaracağız. Robotun mobil olduğunu düşündüğümüz için güç ünitesi de en önemli elemanlarından birisidir, bu yüzden yukarıdaki modelde gösterilmiştir.

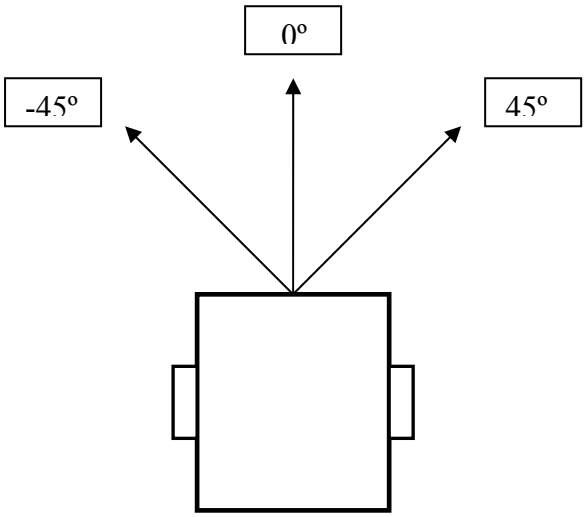
Önceki bölümde ayrıntılı olarak açıklandığı gibi FPGA üzerinde gerçekleştirilecek YSA modeli PNN ağ modelinde olacaktır. Bu model; eğitimde kullanılan girişleri vektör olarak alarak kümeler. Her bir küme için bir küme merkezi belirler. Test sırasında gelen girişleri yine bir vektör olarak ele alır ve küme merkezlerine olan mesafelerini kontrol eder. Hangi küme merkezine daha yakın ise vektör o kümeye dahil edilir.

Tasarladığımız yapay sinir ağının girişlerinin sağ ve sol sensörden gelen uzaklık verileri olduğunu söylemiştik. Yapay sinir ağı, örnek setleri ile gelen giriş değerleri ile eğitilerek, odada dolaşırken aldığı giriş verilerine göre bir çıkış üretmesi beklenir. Tasarladığımız yapay sinir ağının sadece bir çıkış nöronu vardır. Yani verilen girdiler için sadece bir tane sonuç üretecektir. Üretilen bu sonuç, tasarladığımız sisteme göre üç bitlik binary bir sayı olacak ve robotun düz ilerleme istikametine göre ne kadar açıyla sağa yada sola döneceğini verecektir.

Eğitim sırasında robota hangi girişler için hangi çıkışları vereceğine dair örnekler sunulur. Robotun vereceği çıkışlar 3 bit ile temsil edilen, düz ilerleme istikametine göre olan dönüş açılarıdır. Bu dönüş açılarının -90° ve 90° arasında sonsuz tane değer alabileceği aşikardır. Ancak bu kadar detaylı bir hesaplama algoritmayı aşırı derecede karmaşıklaştırarak lojik sentezlemek için içinden çıkılmaz hale getirebilir. Bu yüzden robotun öğrenme algoritması çıkışlarını Çizelge.6.1'de görüleceği gibi yedi seviyeye ayırdık. Burada artı işaret saat yönünde dönüşü, eksi işaret ise saat yönünün aksi yönde dönüşü işaret etmektedir.

Çizelge.6.1 Robotun dönüş açılarına karşılık gelen dijital kodlar

Dönüş açısı:	**Dijital kodlama:**
-60° (sola dön)	001
-45° (sola dön)	010
-30° (sola dön)	011
0° (düz ilerle)	100
30° (sağa dön)	101
45° (sağa dön)	110
60° (sağa dön)	111



Şekil.6.3 Mobil robotun düz ilerleme istikametine göre dönüş açılarının ikisi

Yukarıda çizelgede de görüldüğü gibi, ultrasonik sensörler tarafından engelin hangi yönde olduğu ve robottan ne kadar uzaklıkta olduğu bilgisi alınır. Ultrasonik sensörlerden alınan bu bilgi iki amaç için kullanılır. Birinci olarak sensör girdileri tasarladığımız yapay sinir ağına verilir ve yapay sinir ağı çıkışı olarak ilerleme doğrultusuna göre robotun dönme açısının ne olacağı kararı verilir. Çıkış olarak üretilen dönme açısı binary olarak kodlanmış 3 bitlik bir değerdir. Bu çıkış, mikrodenetleyici tabanlı motor sürücü devresine iletilir. Mikrodenetleyici tabanlı motor sürücü devresinde bir look-up-table içerisinde FPGA'den gelen kodlanmış 3 bitlik bu değerlerin hangi açı değerlerine karşılık geldikleri bilinmektedir. Bu açı değeri göz önünde bulundurularak kısa bir zaman aralığında sağ yada sol motor hızlandırılarak sola yada sağa dönme işlemi gerçekleştirilir. Bu sayede mobil robota yol belirleme (path planning) yeteneği kazandırmış oluruz.

Sensör girdisinden ikinci olarak ise, engelin yerinin tespit edilmesi ve daha sonra hatırlanması için yararlanılır. Ultrasonik sensör ile engelin robottan olan uzaklığı ve doğrultusu bilgileri alınır. Robotun o anda bulunduğu (x,y) koordinatları da bilindiğine göre, engelin bulunduğu (xe,ye) koordinat noktası hesaplanabilir. Robot odanın içinde ilerledikçe karşılaştığı her bir engel için hesapladığı (xe,ye) koordinat bilgisini belleğe kaydeder. Bu sayede, robot oda içerisinde bir sonraki hareketinde hangi koordinatlarda engel olduğunu bileceğinden buna göre kendisini hedefe konumlandıracak optimum yolu seçer. Böylece haritalama yeteneği de robota kazandırılmış olur. Robot yeni hedefine doğru ilerlerken ultrasonik sensörlerden gelen

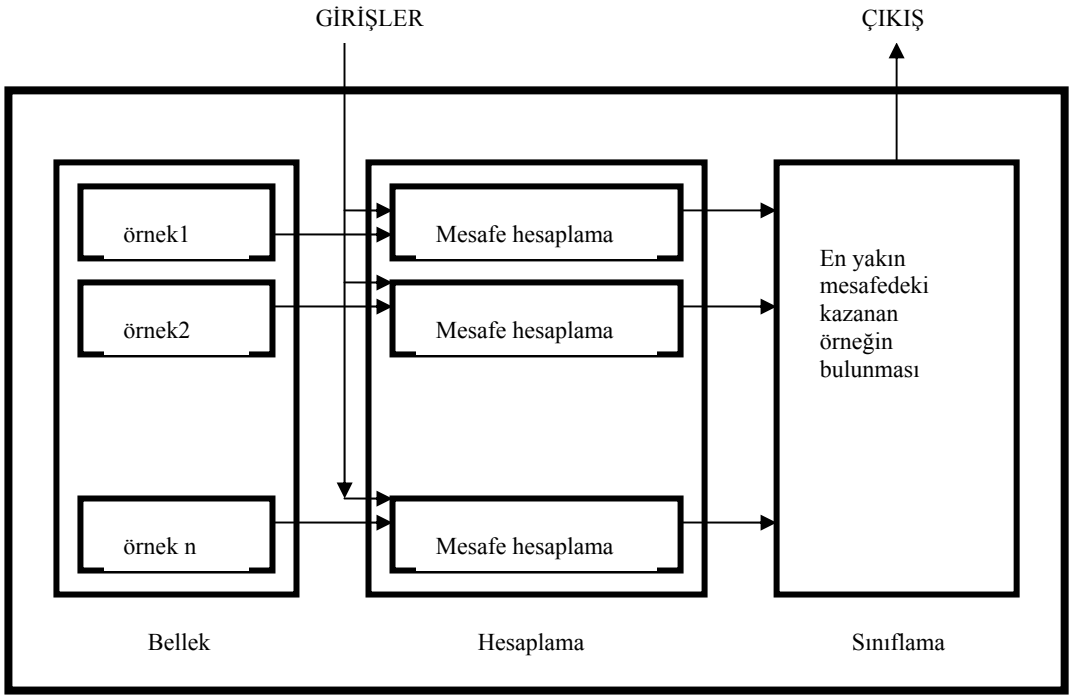
girişleri, dönüş işlemlerini gerçekleştirmek üzere hala değerlendirdiği için karşısına çıkan yeni engellerden de kaçabilir ve bu yeni engellerin (xe,ye) koordinatlarını da hafızasına ekleyebilir. Bu işlem ise; robotlarda online yani çevrim içi yeteneğine karşılık gelmekte ve en çok tercih edilen bu öğrenme işlemi mobil robot sistemimizde bu kadar basit bir yöntem ile gerçekleştirilebilmektedir.

6.3 Tasarlanan FPGA Mimarisinin Blok Diyagramı

Tamamen paralel mimariye sahip bir ağın FPGA ile gerçeklenmesi olasıdır. Tamamen paralel mimariye sahip ağ hızlıdır fakat esnek değildir. Bu tip ağ mimarilerinde yapay sinir hücresine düşen çarpıcı sayısı YSH bağlantılarıyla eşit miktardadır. Bu sebepten her katman için değişik yapay sinir hücresi mimarileri tasarlanmalıdır. Çünkü çarpıcı, YSH yapısında en çok kullanılan elemandır. Tam paralel ağ için ikinci dezavantaj kapı kullanımında artıştır.

Amaç; gerçek hayata en uygun ve çalışma anında katman ve YSH sayısını değiştirebilen yapay sinir ağlarını FPGA yapısında gerçeklemektir.

Tasarlanacak olan FPGA'in blok diyagramı Şekil.6.4'de gösterilmiştir.

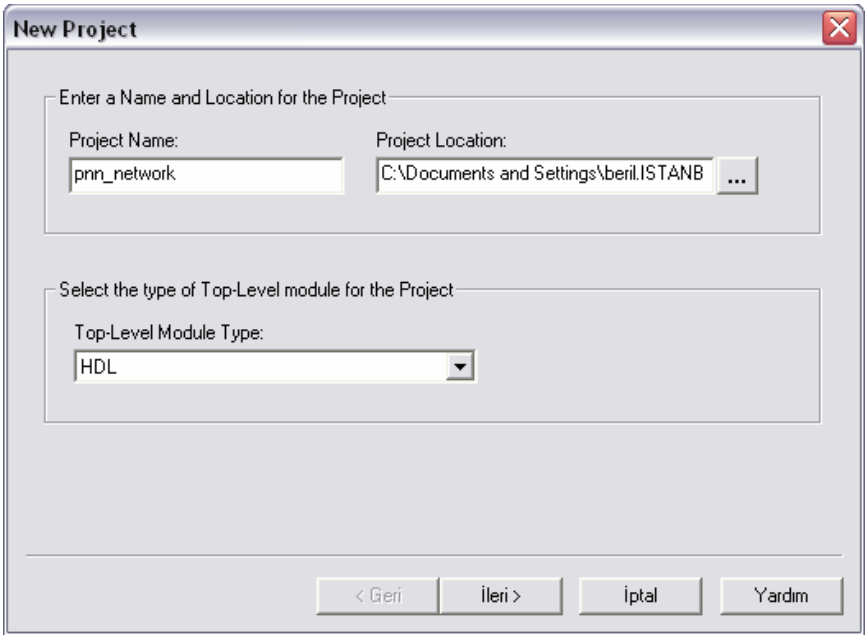


Şekil.6.4 FPGA mimarisi blok diyagramı

6.4 Yapay Sinir Ağının Xilinx Project Navigator Programında Yazılması

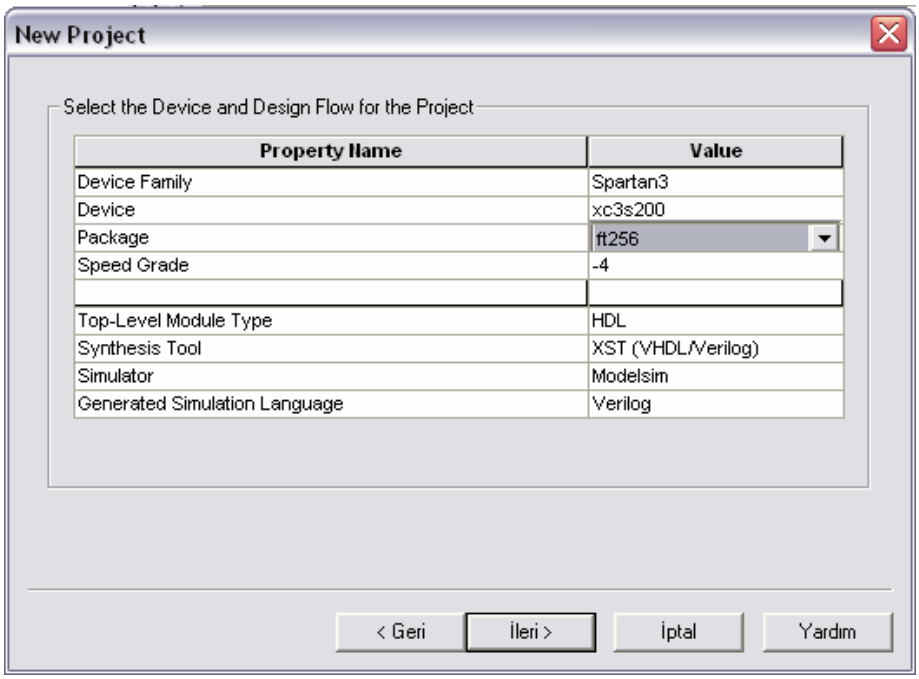
Xilinx ISE programında bir programın tasarım aşamaları aşağıdaki gibidir. Bu aşamaları tek tek izleyerek programın oluşturulması ile ilgili bilgi vereceğiz.

Öncelikle yeni bir proje oluşturalım. Bunun için File → New Project yolu izlenir ve Şekil.6.5'deki ekran karşımıza çıkar. Bu ekranda Project Name yazan boşluğa projemize vermek istediğimiz ismi giriyoruz. Project Location boşluğuna ise projemizi kaydetmek istediğimiz yolu tanımlıyoruz. Top-Level Module Type kısmı ise HDL olacaktır. Bu işlemleri gerçekleştirdikten sonra ileriyi tıklayarak bir sonraki adıma geçeriz.



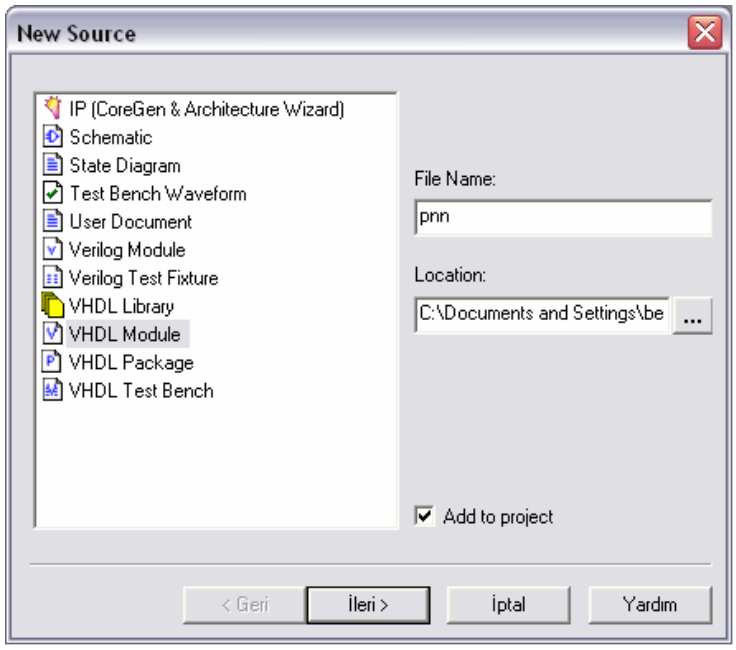
Şekil.6.5 Xilinx ISE'de yeni bir projenin oluşturulması

Daha sonra kullandığımız FPGA ile ilgili bilgileri karşımıza çıkan pencerede tanımlamalıyız. Bu projede Spartan3 xc3s200 ft256 -4 FPGA için modelleme yapılmıştır. Bu çipi tanımlamak için Family kısmına; Spartan3, Device kısmına; xc3s200, Package kısmına ft256 ve Speed Grade kısmına ise -4 yazmalıyız. Bu işlemler gerçekleştirildikten sonra ileri seçeneği ile diğer aşamaya geçilir. Diğer aşamalarda işlem gerçekleştirmeyeceğimiz için ileri tuşuna basılarak geçilir. Ve son tuşuna basılarak proje dosyası oluşturulur.



Şekil.6.6 Xilinx ISE programında kullanılacak FPGA'in seçilmesi

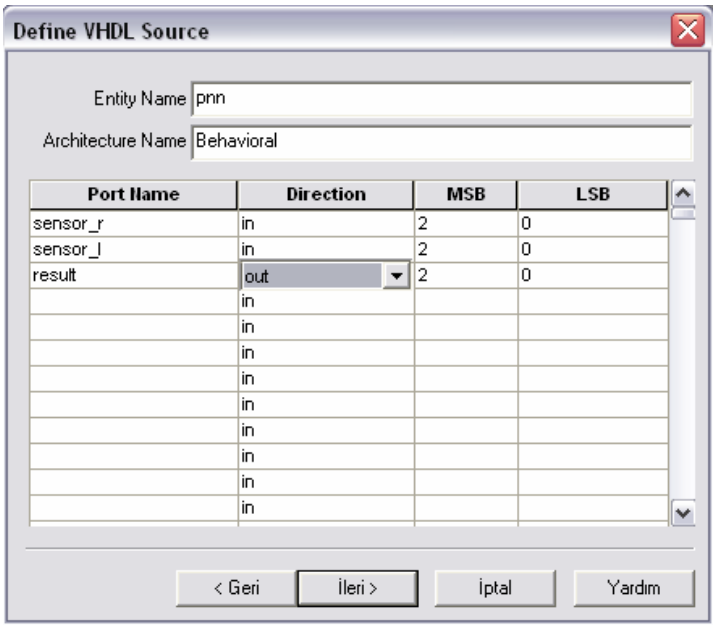
Proje çalışmasına yeni bir VHDL modülü eklemek için Project → New Source seçilir ve karşımıza Şekil.6.7'deki pencere açılır. Burada VHDL modülü yazılacağı için VHDL Module işaretlenir. Location kısmına is uzantılı dosyanın yolu tanımlanır. Genellikle bu yol değiştirilmemelidir. Çünkü program oluşturulacak dosyaların yollarını bu yola yönlendirecektir. İleri tuşuna basılarak bir sonraki aşamaya geçilir. Burada işlem sonlandırılarak dosyamız oluşturulur.



Şekil.6.7 VHDL modül dosyasının oluşturulması

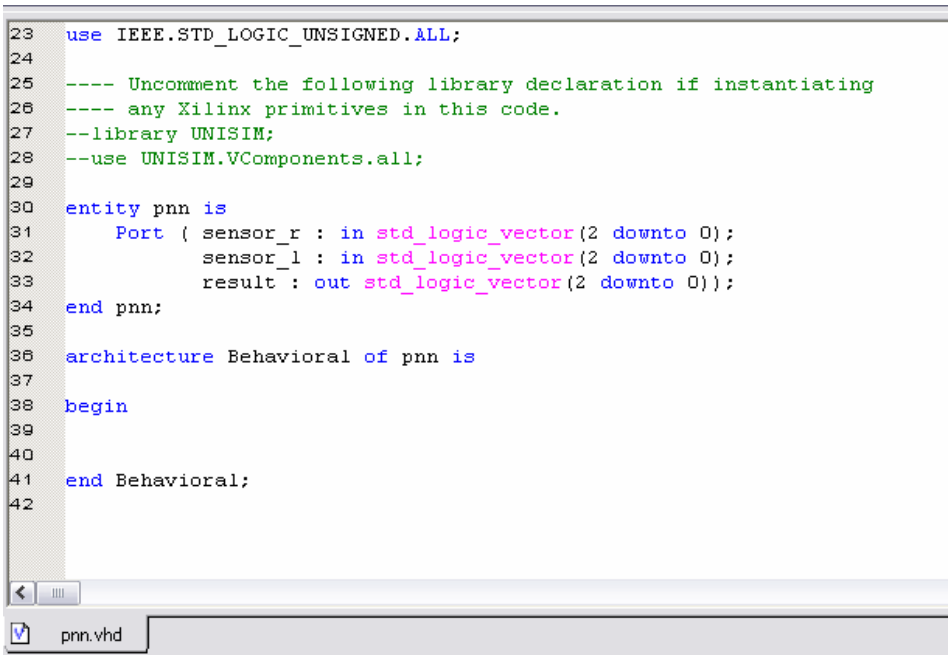
Oluşturduğumuz VHDL dosyasında kütüphaneler ve bazı özel terimler otomatik olarak atanmaktadır.

Tasarımı yapmaya devam edebilmek için sistemin giriş/çıkışlarının kaçar adet olduğunu belirtmemiz gerekir.



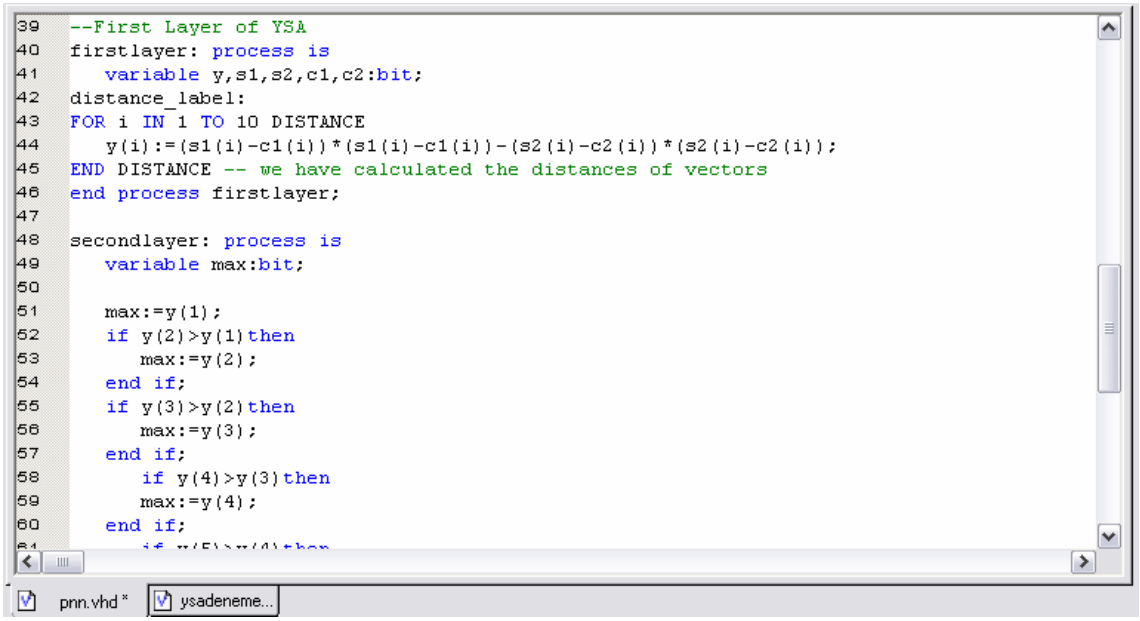
Şekil.6.8 Lojik yapının giriş ve çıkış pinlerinin oluşturulması

Şekilden görüldüğü gibi sistemin girişleri; üçer bitlik sağ sensör (sensor_r) ve sol sensör (sensor_l) girdileri, sistemin çıkışı ise; yine üç bitlik result çıkışı olarak alınmıştır. Burada üçer bitlik sensör girişleri; ultrasonik sensördeki bilgiyi alacak ve onu üç bitlik seviyelere kuantalayacak mikrodenetleyicimizden gelecektir. Result olarak isimlendirilen üç bitlik çıkış ise; mikrodenetleyiciye geri gidecek olan çıkış pinleridir. Oluşan proje dosyamızın entity bölümüne bu port tanımlamaları otomatik olarak yerleştirilecektir.



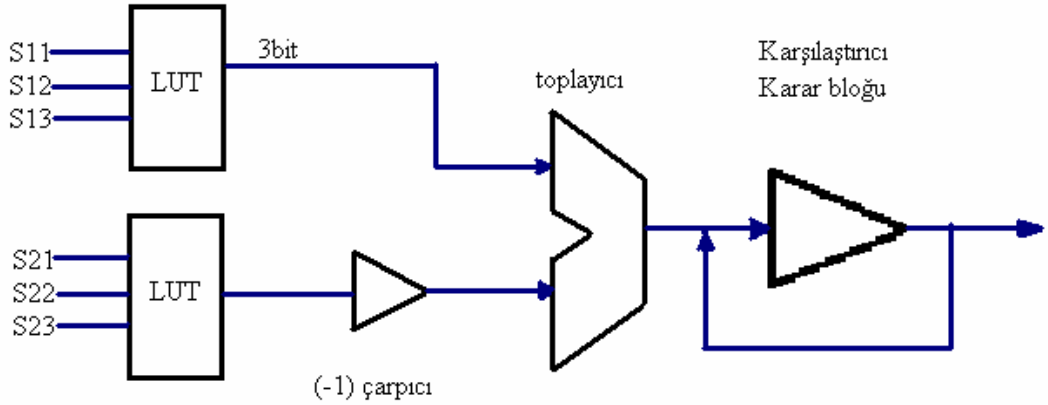
Şekil.6.9 VHDL kodlarının yazıldığı ekran

Tasarladığımız devrenin portlarını tanımladıktan sonra devrenin davranışını da tanımlamalıyız. Bu işlemler begin ile end Behavioral arasında olmalıdır. Bu alana her iki katman için de YSA'nın davranışı ve nasıl sonuç üretileceği yazılacaktır.



Şekil.6.10 Yazılım ile YSA donanımının tanımlanması

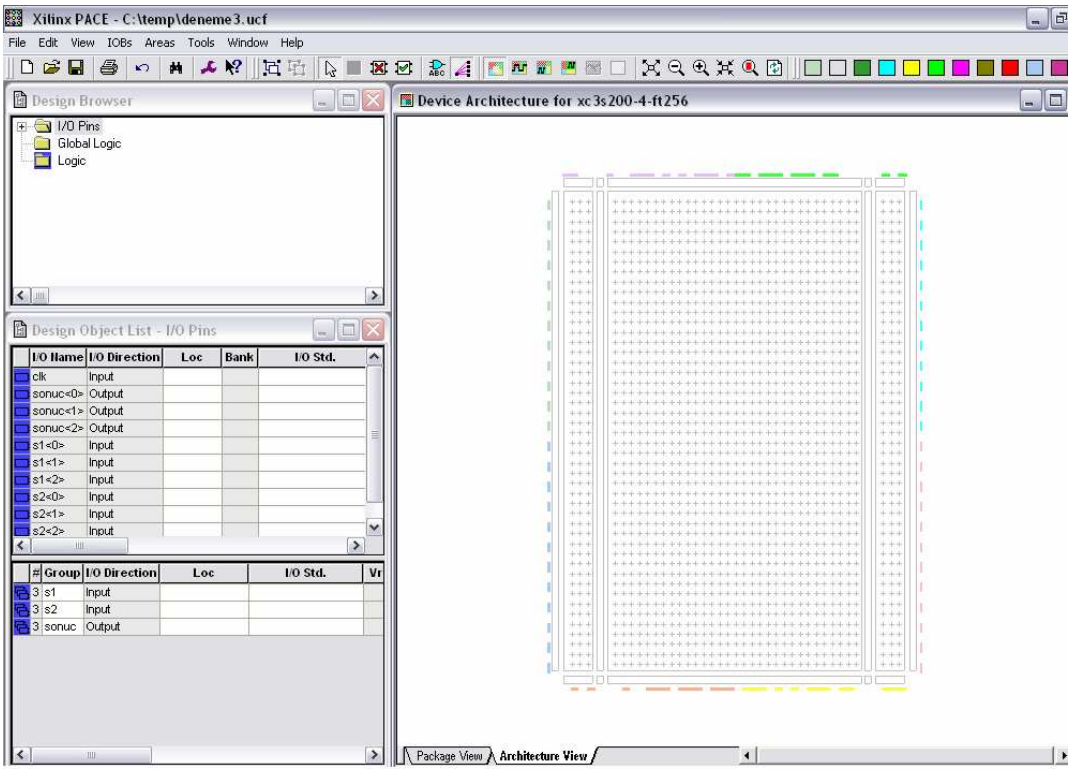
Tasarladığımız lojik devreyi görmek için ise Xilinx Project Navigator programında Synthesize-XST → View RTL Schematic seçilir. Oluşturduğumuz YSA'nın lojik kapılarla gerçekleştirilmiş devre şekli aşağıdaki şekildeki gibidir.



Şekil.6.11 Gerçeklenen yapay sinir ağının lojik yapısı

Bu çalışmada FPGA'i programlamak için kullanılan VHDL kodları EK2'de bulunabilir.

Modelleme işlemi bittikten sonra, program FPGA üzerine yüklendiğinde hangi pinlerin giriş ve hangi pinlerin çıkış olacağı ve hangi yerlerdeki lojik bloklarının kullanılacağını da grafik ekran üzerinden seçebilmek mümkündür. Bunun için ***Assign Packet Pins*** seçeneği seçilir. Karşımıza Şekil.6.12'daki ekran çıkacaktır.



Şekil.6.12 FPGA üzerinde giriş çıkış pinlerinin seçilmesi

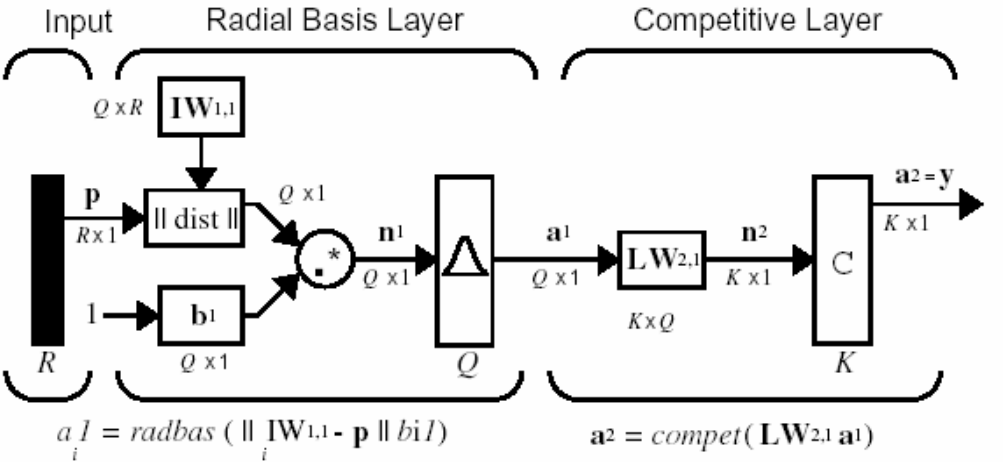
6.5 Modelsim ile FPGA Simülasyonu

Tüm tasarım aşamalarında, tasarım doğrulama yapılmaktadır. Bu aşamada yapılmış olan tasarım test sinyalleri ile devre lojiğini ve zamanlamasını test eder. Programlanabilir lojiğe uygulanan bazı temel testler vardır. Yerleştirme ve yönlendirme yapılmadan önce lojik doğruluğu kontrol edilir. Tam zaman testi, yerleştirme ve yönlendirmeden sonra yapılabilir. Yerleştirme ve yönlendirmeden sonra zamanlama simülasyonunu uygulayarak yollardaki ve lojik bloklardaki bekleme sürelerini hesaplar. Simülasyon hep tavsiye edilmekle birlikte, kapı dizilerinde olduğu gibi yoğun simülasyon aşamalarına ihtiyaç yoktur. Oysa bir kapı dizisinde yoğun simülasyon çok önemlidir.

Bir önceki bölümde anlatıldığı gibi devremizin davranışı da tanımlandıktan sonra artık simülasyon işlemlerine geçebiliriz. Öncelikle Xilinx ISE'deki projemizde; Design Entry Utilities → Create Schematic Symbol yapılır. Daha sonra Launch Modelsim Similator kısmı tıklanarak Modelsim programına yazdığımız kodlar aktarılır ve karşımıza Modelsim ekranı çıkar. Burada Modelsim ile birlikte açılan wave-default penceresinde devremizin portları tanımlı olarak bulunmaktadır.

6.6 Öğrenme Algoritmasının Matlab Simülasyonu

ART ağları, yapay sinir ağları bölümünde bahsedildiği gibi sınıflama problemlerinin çözümünde kullanılabilir. Robotun sensör girişlerine karşı hangi çıkışı vereceği yani hangi dönüş yönünü seçeceği problemini bir sınıflama problemi olarak ele alabiliriz. Bir giriş seti ağa verildiğinde, birinci katmanda bu set bir vektör olarak ele alınır ve eğitimde kullanılan vektörlere ne kadar yakın olduğuna bakılır. İkinci katmanda ise yarışmalı bir ağ tarafından, eğitim vektörlerinden hangisinin giriş vektörüne daha yakın olduğu seçilir ve bu eğitim vektörünün çıkışı ağın o andaki çıkışı olarak verilir. Matlab kodları ile tasarladığımız ve simule ettiğimiz ağ mimarisi aşağıdaki gibidir.



Şekil.6.13 Matlab kodları ile dizayn edilen ağ mimarisi

R = Giriş vektöründeki özellik sayısı

Q = Giriş/Çıkış vektörleri sayısı (1. katmandaki nöron sayısı)

K = Sınıf sayısı (2. katmandaki nöron sayısı)

$IW^{1,1}$ olarak isimlendirilmiş olan ağırlık vektörleri eğitim setleri ağa sunuldukça değiştirilerek uygun bir değere yaklaşması sağlanır. Test setinden ağa bir örnek sunulduğunda, ||dist|| isimli blokta giriş vektörleri arasındaki uzaklığın normu hesaplanır ve sonuçlar ağın ikinci katmanına gönderilir. İkinci katmana giren girdiler yarışmalı fonksiyondan geçerek normu en düşük olan yani eğitim setindeki en çok benzeyen vektör bulunur. İkinci katmanda $IW^{1,2}$ olarak isimlendirilmiş ağırlıklar, her satırında sadece bir tane "1" değeri bulunan bir matrisdir. Satırlardaki birler, girişin hangi sınıfa ait olduğunu belirler. Böylece test girişi için üretilen çıkış, eğitim seti sonuçları için ayrılmış K adet sınıftan birisine ait olur Rabunal, Drado,

2006). Matlab'de öğrenme algoritmamızı test ederken sensör değerlerine karşılık gelen analog değerleri nöron girişleri olarak kullandık. Sensör girişlerinden alınan analog değerlerin nasıl dijital seviyelere kodlandığını devrenin lojik olarak sentezlenmesi bölümde açıklamıştık. Matlab testlerinde Çizelge.6.2'de görüldüğü gibi bu dijital değerlere karşılık gelen değerler nöron girişi olarak alınmıştır.

Çizelge.6.2 Matlab simülasyonunda nöron girişlerine karşılık gelen değerler

Dijital Kodlama:	**Matlab-nöron giriş değeri:**	**Engele olan mesafe:**
000	0	Engel yok
001	1	1.75m<r<2m
010	2	1.5m<r<1.75m
011	3	1.25m<r<1.5m
100	4	1m<r<1.25m
101	5	0.75m<r<1m
110	6	0.5m<r<0.75m
111	7	0.25m<r<0.5m

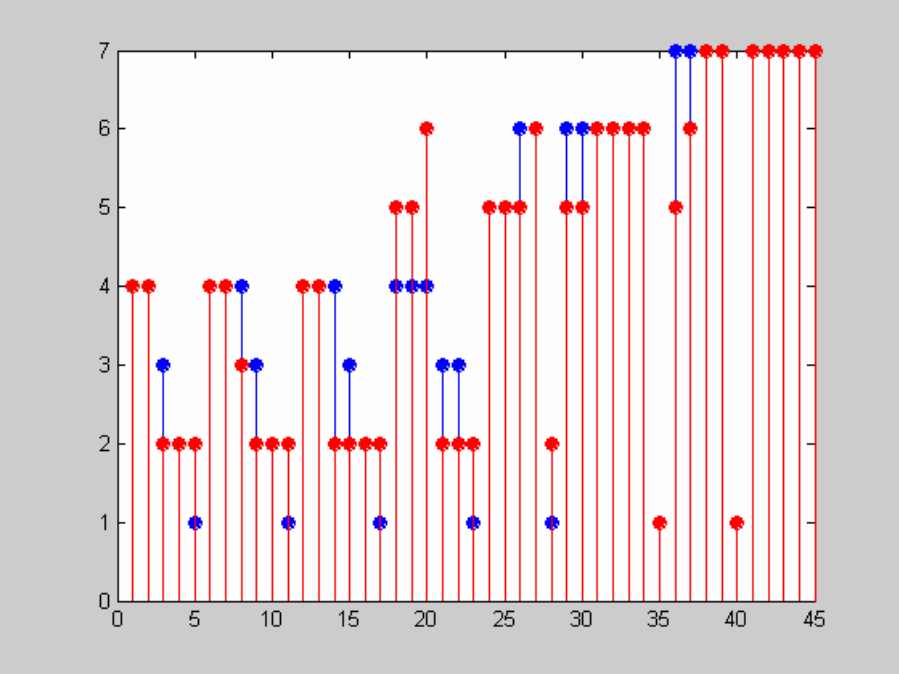
Nöron çıkışları da benzer şekilde lojik sentez bölümünde kullanılan dijital değerlerin ondalık tabanda karşılıkları olarak alınacaktır. Çizelge.6.3.'de nöron çıkışları için matlab simülasyonunda alınan değerler görülmektedir.

Çizelge.6.3 Matlab simülasyonunda nöron çıkışlarına karşılık gelen değerler

Dijital Kodlama:	**Matlab-nöron çıkış değeri:**	**Dönüş açısı:**
001	1	-60°
010	2	-45°
011	3	-30°
100	4	0°
101	5	30°
110	6	45°
111	7	60°

Yapay sinir ağını tasarlamak, eğitmek ve test etmek için kullanılan matlab kodları açıklamalarıyla birlikte EK1'de görülmektedir.

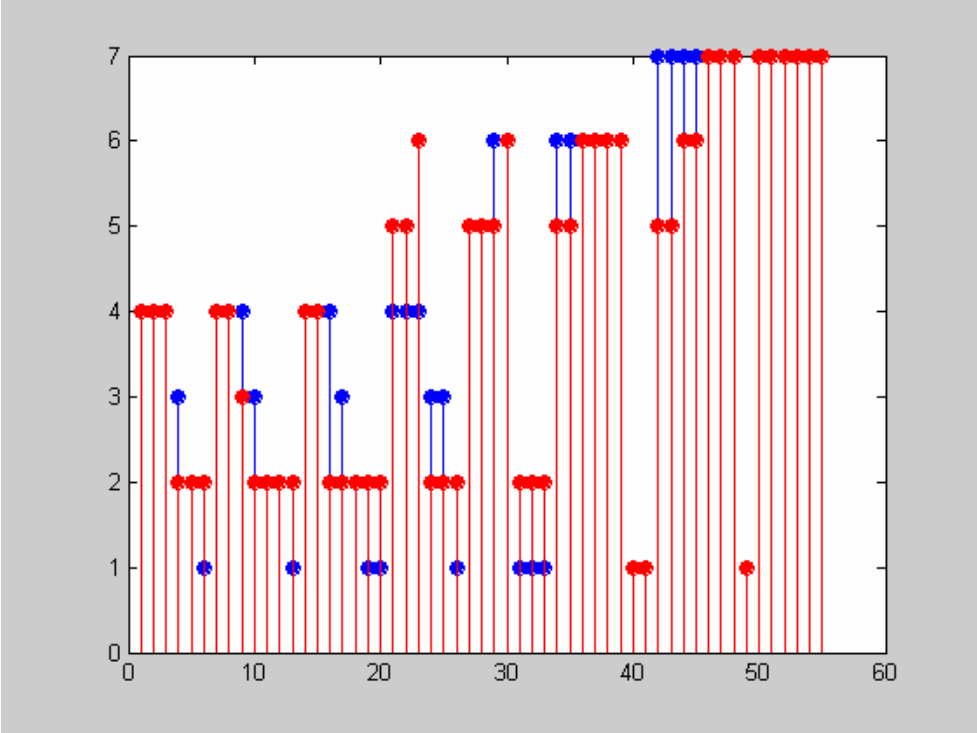
Matlab'de simülasyonunu yapmak için oluşturduğumuz ağı, örnek bir uygulama olarak önce 45 adet örnek içeren eğitim seti ile eğittik. Sonra bu eğitim örnekleri ile aynı çıkışları vereceğini bildiğimiz bir test setini ağa gösterdik ve çıkışlarını beklenen çıkışlar ile karşılaştırdık. Bulunan sonuçlara ait grafikler aşağıdaki gibidir. İlk şekilde mavi ile gösterilen noktalar, her bir örnek için beklenen sonuçları, kırmızı ile gösterilen noktalar ise her bir örnek için ağın ürettiği çıkışı göstermektedir. İkinci olarak ise, beklenen çıkış değerleri ile ağın çıkış değerlerinin farkları alınarak ağın oluşturduğu hata miktarları incelenmiştir.



Şekil.6.14 Ağın beklenen çıkışları (mavi) ile ağın gerçek çıkışlarının (kırmızı) karşılaştırılması

Hata değerlerinden yola çıkarak şunu söyleyebiliriz tasarlamış olduğumuz öğrenme algoritması mimarisinin başarım oranı % 88.88'dir.

İkinci bir uygulama olarak, eğitim setini değiştirelim ve 55 adet örnek ile eğitelim. Bu durumda elde edilen sonuçlar aşağıdaki gibidir. İlk şekilde mavi ile gösterilenler beklenen çıkışlar, kırmızı ile gösterilen noktalar ise ağın çıkışlarıdır.



Şekil.6.15 55 adet eğitim örneği için ağın beklenen çıkışları (mavi) ve ağın gerçek çıkışları (kırmızı)

Tasarladığımız ağın Matlab simülasyonunda da göreceğimiz gibi eğitim setindeki örnek sayısı 10 adet arttırıldığında bile hata oranı %2.02 kadar düşmüştür. Çizelge.6.4'de alınan eğitim örneği sayısı ve çıkıştaki hata yüzdesi karşılaştırmaları verilmiştir.

Çizelge.6.4 Eğitim örneği sayısı ile hata yüzdelerinin karşılaştırılması

Eğitimde kullanılan örnek sayısı:	**Ağ çıkışındaki hata yüzdesi:**
10	%40
30	% 10
45	% 11.11
55	% 9.09
60	%6.6

Bu çizelgedeki sonuçlara göre şunları söyleyebiliriz. Eğitim örneği sayısı arttıkça, genelde hata yüzdesi azalmaktadır. Ancak yine de hata yüzdesinin ne olacağı, eğitim için seçilen örneklerin olayı ne kadar iyi temsil ettiğine bağlıdır. Bu yüzden, 30 örnekle eğitilen ağın hatası %10 iken, 45 adet örnekle eğitilen ağın hatası daha fazla olmuştur. Ancak genel olarak

yüzdeleri, eğitim setindeki örnek sayısı arttıkça hızlı bir düşüş göstermiştir. Buradan da anlaşılacağı gibi, 10 adet eğitim örneği olayı yeterince iyi temsil edememekte ve sağlıklı sonuçlara ulaşmamıza izin vermemektedir.

Eğer aynı sonuçları yapay sinir ağlarını kullanmadan elde etmek isteseydik 2^6 adet her bir sensör giriş değerine karşılık, çıkış değerlerinin girilmesi gerekmektedir. Oysa bu örnekte de görüldüğü gibi yapay sinir ağlarıyla gerçekleştirdiğimiz tasarımımızdan faydalanarak sadece 17 adet tanımlanmış değerden yola çıkarak, çıkış değeri büyük yaklaşıklıkla elde edilmektedir.

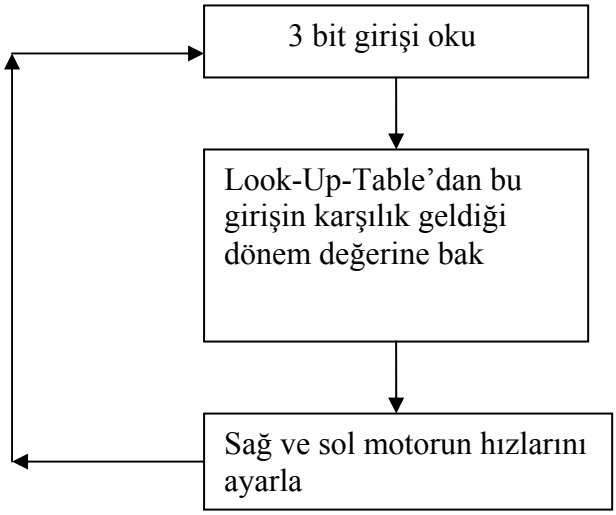
6.7 Mikrodenetleyici ile FPGA'den Gelen Sonuçların Yorumlanması ve Motor Kontrolü

Yukarıdaki bölümde FPGA ile robotun engel ile karşılaştığında engele çarpmadan nereye yöneleceğine karar verme işlemi yapay sinir ağları ile oluşturulmuş algoritmalar ile yapıldı. Robotun karşısına bir engel çıktığında FPGA devreye girecek, robot hangi yöne yöneleceğine karar verirken aynı zamanda karşılaşılan engelin yerinin öğrenilmesi için engelin koordinatları FPGA hafızasına yüklenecektir. Sağa yada sola hangi açı ile dönüş gerçekleşeceğine karar verdikten sonra FPGA, 'y' çıkış sinyalini dışarıya verir. Bu sinyal yukarıdaki bölümde incelendiği gibi üç adet çıkış nöronundan dışarıya verilen sonuç olacağından 3-bitlik bir işarettir. FPGA'in verebileceği y çıkışları ve mikrodenetleyicinin bunlara yükleyeceği anlamlar Çizelge.6.5'deki gibi seçilmiştir.

Çizelge.6.5 FPGA çıkış işaretleri ve mikrodenetleyicinin bunlara karşılık yapacağı işlemler

'y' çıkışı:	**Anlamı:**	**Mikrodenetleyicinin yapacağı işlem:**
001	60º sola dön	Sağ motora sinyal göndererek hızlandır
010	45º sola dön	Sağ motora sinyal göndererek hızlandır
011	30º sola dön	Sağ motora sinyal göndererek hızlandır
100	Düz ilerle	Motorları aynı hızlarda sürmeye devam et
101	30º sağa dön	Sol motora sinyal göndererek hızlandır
110	45º sağa dön	Sol motora sinyal göndererek hızlandır
111	60º sağa dön	Sol motora sinyal göndererek hızlandır

Mikrodenetleyicinin çalışma algoritmasının akış diyagramı basit olarak aşağıdaki gibidir.



Şekil.6.16 Mikrodenetleyicili motor kontrol sisteminin çalışma algoritması

6.8 Arayüz ile Projenin Simülasyonu

Model robotun, rasgele yerlere koyulmuş engellerle dolu bir odada her hangi bir noktadan harekete başlayarak hedef noktaya ulaşmasını ve bu sırada engellerden kaçmayı ve odayı öğrenmesini gösteren bir simülasyon programı delphi ortamında hazırlanmıştır. Ekranda görülen mobil robot engellerden kaçarken bu çalışmada açıklanan PNN algoritmasını kullanmaktadır: Bu arayüz sayesinde tez çalışmasında ele alınan problem ve çözüm yaklaşımı görsel olarak daha iyi açıklanmaya çalışılmıştır.

Şekil.6.17 Model robotun engellerden kaçmayı öğrenerek hedefe varmasını gösteren simülasyon arayüzü

7. MODEL ROBOTUN KULLANILABİLECEĞİ ALANLAR

Bu çalışmada modellenen robot pek çok yaşam veya üretim alanında kullanılabilir. Örneğin bir fabrika veya ofis ortamında çalışma boyunca sürekli parça taşınması gerekebilir. Mevcut insan gücünün bu işlerle uğraşması zaman kaybıdır. Böylece insan gücü daha verimli işler için kullanılabilir.

Model robot geliştirildiği takdirde biyomedikal amaçlı da kullanılabilir. Örneğin engelli insanlar çoğunlukla yaşam alanlarının kısıtlı bir bölümünü kullanabilmektedir. Eğer bu robot modeli bir yürüyen sandalyeye uygulanabilirse sandalye evin içerisinde gidilecek yerleri, odaları ve koridorları öğrenebilir ve bir yere çarpmadan en uygun yoldan gidilecek yere gidebilir. Ayrıca robotun öğrenebilme yeteneği sayesinde yeni bir mekana gidildiğinde ev yada ortam değiştiğinde robot tekrar öğrenecektir.

Donanımsal mimaride haritalama işlemi şu şekilde gerçekleştirilebilir; Robot resetlendikten sonra işlem modunda çalışmaya başlar. Öğrenme modunda robot bulunduğu odayı tanır. Öğrenilen bilgiler, kısa zamanlı hafıza olarak bilinen dahili RAM bellekte saklanır. X ve Y eksenindeki koordinatları tutabilmek için iki tane counter bulunur. Countx ve county saymaçları robot hareket ederken X ve Y eksenindeki hareketleri sayarlar. Robot bir engelle karşılaştığında sayıcı değerlerine karşılık gelen hafıza biti “1” yapılır. Böylece bir odayı tanıma, haritalama işlemi yapılmış olur. Robot işlem moduna alındığında, saymaçlar yine çalışmaya başlar. Her bir X, Y koordinatı için saymacın değerinin gösterdiği bellek bölgesine bakılır ve “1” olur olmamasına göre harekete devam edilir.

8. SONUÇLAR

Gerçek zamanlı bir çok uygulama; öğrenme aşamasında hızlı bir eğitim ve test aşamasında da hızlı hatırlayabilmek ister. Bu tip problemlerin olabilecek çözümlerinden biri de yapay sinir ağlarını tekrar düzenlenebilir tümleşik devreler üzerinde gerçeklemektir. FPGA, programlanabilen bir lojik elemandır ve yazılım gibi esnek tasarım önerirler. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı ilk üretme özelliği ile FPGA'ler sayısal tasarım ortamlarının vazgeçilmez yapıları haline gelmiştir.

Bu projede, mobil bir robotun donanımsal yapısı modellenmiş, engellerden kaçmasını, bir odayı haritalamasını ve aktif öğrenmesini sağlayacak bir öğrenme algoritması geliştirilmiş ve PNN algoritmasından yola çıkarak modellenmiştir. Modellenen bu öğrenme algoritmasının FPGA ile gerçeklenmesi gösterilmiş, aynı zamanda öğrenme algoritmasının Matlab ortamında simülasyonları yapılmış başarım yüzdeleri hesaplanmıştır. Son olarak ise; mobil bir robotun engellerin olduğu bir odada hedef noktaya en kısa yoldan varmasını ve engel koordinatlarını öğrenmesini gösteren bir bilgisayar grafik arayüzü Delphi ortamında hazırlanmıştır. Delphi ortamında geliştirilen arayüz, boş bir oda içerisine hedef noktanın, mobil robotun ve engellerin istenilen yerlere kullanıcı tarafından yerleştirilmesine müsaade etmektedir. Mobil robot, bu rasgele koordinat noktasından başlayarak yavaş yavaş seçilen hedef noktasına ilerlemekte ve bu sırada karşısına gelen engellerden kaçmaktadır. Engellerin bulunduğu noktaları öğrenerek koordinatlarını kullanıcıya vermektedir. Model robotun Matlab ortamında hesaplanan sonuçlara göre öğrenme başarısı %88.88 olmuştur. Hata grafikleri bu çalışma içinde verilmiştir.

İlerleyen çalışmalarımda modellenen bu mobil robotun ve öğrenme algoritması ve mimarisinin gerçek yaşam uygulamaları üzerine çalışacağım. Aynı zamanda mobil robotun koordinatlarını bilmediği bir odaya girdiğinde odayı ve engelleri keşfi üzerinde çalışmalarıma devam edeceğim. Mobil robotu geliştirmek için ultrasonik sensörlerden alınan bilgilerin yanı sıra CCD kameralardan alınan bilgilerin FPGA ile işlenmesi ve yorumlanması üzerine çalışacağım.

KAYNAKLAR

Ashenden, P.J., VHDL Quick Start, The University of Adelaide, Ders Notları

Çayurpınar, Ö. (2005), "Sensörler", Tübitak Bilim ve Teknik Dergisi, Aralık 2005

Haton, J. P. (2003), "Yapay zeka", Bilim ve Teknik Dergisi, Mayıs 2003, Sayfa 32

Jain, L.C. ve Medsker, L. (2001), Recurrent Neural Networks Design and Applications, CRC Press

Karen, P. ve Mehta, N. (2002), Programmable Logic Design Quick Start Hand Book, Xilinx Second Edition

Matlab Neurosolutions tool 5.0 User Guide

Öztemel, E. (2006), Yapay Sinir Ağları, Papatya Yayıncılık,

Pearson, M.J., Melhuish, C., Pipe, A.G., Nibouche, M., Gilhesphy, L., Gurney, K. ve Mitchinson, B. (2005), "Design and FPGA implementation of an embedded real-time biologically plausible spiking neural network processor", Field Programmable Logic and Applications 2005 IEEE International Conferance, Sayfa: 582-585

Rabunal, J. R.ve Dorado, J. (2006), Artificial Neural Networks in real life applications, Idea Group Inc., London Press

Vonk E., Veelenturf L.P.J. ve Jain L.C. (1996), "Neural Networks: implementations and applications", Aerospace and Electronic Systems Magazine, IEEE July1996, Sayfa: 11-16

Xilinx Spartan Series FPGA's, Spartan-3 family, Product Guide

Yıldırım T. (2006), Yapay Sinir Ağları Ders Notları, Yıldız Teknik Üniversitesi

Zarate, L. E., Becker, M., Garrido, B.D.M. ve Rocha, H.S.C. (2002), "An Artificial Neural Network Structure Able to Obstacle Avoidance Behaviour Used in Mobile Robots", IECON'02 Industrial Electronics Society, IEEE 28th Annual Conference, Vol.3, Sayfa: 2457-2461

EKLER

EK1

Bu bölümde, tasarlanan öğrenme algoritmasının test edilmesi için Matlab ortamında simulasyonuna imkan veren kodlar sunulmuştur.

```
c=fopen('train55.txt','r');
s=fscanf(c,'%f',[2,55]);
fclose(c);
P=s';
cd=fopen('output55.txt','r');
s=fscanf(cd,'%f',[1,55]);
fclose(cd);
Tc=s';
m=55;
n1=1;n2=1;n3=1;n4=1;n5=1;n6=1;n7=1;
%her sınıftan kacar tane ornek oldugunu bul:
for(i=1:m)
   if Tc(i)==1
      clss(1)=clss(1)+1; %1.sinifin sayisini 1 arttir
      c1(n1,:)=P(i,:); %egitim girisini 1. sinif matrisine al
      n1=n1+1;
   end
   if Tc(i)==2
      clss(2)=clss(2)+1;
      c2(n2,:)=P(i,:); %egitim girisini 2. sinif matrisine al
      n2=n2+1;
   end
   if Tc(i)==3
      clss(3)=clss(3)+1;
      c3(n3,:)=P(i,:); %egitim girisini 3. sinif matrisine al
      n3=n3+1;
   end
```

```
    if Tc(i)==4
        clss(4)=clss(4)+1;
        c4(n4,:)=P(i,:); %egitim girisini 4. sinif matrisine al
        n4=n4+1;
    end
    if Tc(i)==5
        clss(5)=clss(5)+1;
        c5(n5,:)=P(i,:); %egitim girisini 5. sinif matrisine al
        n5=n5+1;
    end
    if Tc(i)==6
        clss(6)=clss(6)+1;
        c6(n6,:)=P(i,:); %egitim girisini 6. sinif matrisine al
        n6=n6+1;
    end
    if Tc(i)==7
        clss(7)=clss(7)+1;
        c7(n7,:)=P(i,:); %egitim girisini 7. sinif matrisine al
        n7=n7+1;
    end
end
%**********************************************************************
c=fopen('train15.txt','r');
s=fscanf(c,'%f',[2,15]);
fclose(c);
P2=s';
k=15;
n=1;
for (index=1:k)
v1(1,:)=P2(index,:);
% %1.katman. tüm eğitim örneklerine olan benzerliği bul.
for(x=1:clss(1)) %test ornekleri sayisina kadar
   % for(y=1:m) %egitim ornekleri sayisi
    %    v1=P(y,:);
```

```
        v2=c1(x,:);
        fark1(n)=dist(v1,v2');
        n=n+1;
        %end
end
n=1;
for(x=1:(clss(2))) %test ornekleri sayisina kadar
   % for(y=1:m) %egitim ornekleri sayisi
    %    v1=P(y,:);
        v2=c2(x,:);
        fark2(n)=dist(v1,v2');
        n=n+1;
        %end
end
n=1;
for(x=1:(clss(3))) %test ornekleri sayisina kadar
   % for(y=1:m) %egitim ornekleri sayisi
    %    v1=P(y,:);
        v2=c3(x,:);
        fark3(n)=dist(v1,v2');
        n=n+1;
        % end
end
n=1;
for(x=1:clss(4)) %test ornekleri sayisina kadar
   % for(y=1:m) %egitim ornekleri sayisi
    %    v1=P(y,:);
        v2=c4(x,:);
        fark4(n)=dist(v1,v2');
        n=n+1;
        % end
end
n=1;
for(x=1:clss(5)) %test ornekleri sayisina kadar
```

```
   %for(y=1:m) %egitim ornekleri sayisi
    %   v1=P(y,:);
       v2=c5(x,:);
       fark5(n)=dist(v1,v2');
       n=n+1;
       % end
end
n=1;
for(x=1:clss(6)) %test ornekleri sayisina kadar
   %for(y=1:m) %egitim ornekleri sayisi
    %   v1=P(y,:);
       v2=c6(x,:);
       fark6(n)=dist(v1,v2');
       n=n+1;
       %end
end
n=1;
for(x=1:clss(7)) %test ornekleri sayisina kadar
   %for(y=1:m) %egitim ornekleri sayisi
    %   v1=P(y,:);
       v2=c7(x,:);
       fark7(n)=dist(v1,v2');
       n=n+1;
       % end
end
%************************************************************
% % %2.katman. her bir sınıfa ait benzerlikleri topla
top1=0;top2=0;top3=0;top4=0;top5=0;top6=0;top7=0;
for(n=1:clss(1))
   top1=top1+fark1(n);
end
for(n=1:clss(2))
   top2=top2+fark2(n);
end
```

```
for(n=1:clss(3))
   top3=top3+fark3(n);
end
for(n=1:clss(4))
   top4=top4+fark4(n);
end
for(n=1:clss(5))
   top5=top5+fark5(n);
end
for(n=1:clss(6))
   top6=top6+fark6(n);
end
for(n=1:clss(7))
   top7=top7+fark7(n);
end
sonuc(1)=(1/clss(1))*top1; %ortalamalarini al
sonuc(2)=(1/clss(2))*top2;
sonuc(3)=(1/clss(3))*top3;
sonuc(4)=(1/clss(4))*top4;
sonuc(5)=(1/clss(5))*top5;
sonuc(6)=(1/clss(6))*top6;
sonuc(7)=(1/clss(7))*top7;
%3.katman. benzerlik toplamları arasında en büyük olanı seç
out(index)=1;
if(sonuc(1)>sonuc(2))
   out(index)=2;
end
if(sonuc(2)>sonuc(3))
   out(index)=3;
end
if(sonuc(3)>sonuc(4))
   out(index)=4;
end
if(sonuc(4)>sonuc(5))
```

```
    out(index)=5;
end
if(sonuc(5)>sonuc(6))
    out(index)=6;
end
if(sonuc(6)>sonuc(7))
    out(index)=7;
end
%1-7 yanlis algilamasi icin duzeltme:
a=P2(index,1)+1;
b=P2(index,2)+1;
if(((out(index))<4)&&((a/b)>1))
    out(index)=8-out(index);
end
if(((out(index))>4)&&((a/b)<1))
    out(index)=8-out(index);
end
if(a==b)
    if((a>5)&&(out(index)<4))
        out(index)=8-out(index);
    end
end
end
out
```

EK2

Bu bölümde, hazırlanan öğrenme algoritmasının FPGA için VHDL dilinde Xilinx ISE ortamında yazılımı sunulmaktadır.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------
entity deneme3 is
   Port ( s1 : in std_logic_vector(2 downto 0);
          s2 : in std_logic_vector(2 downto 0);
                        clk: in bit;
                        i: integer;
          sonuc : out std_logic_vector(2 downto 0));
end deneme3;
----------------------------------------------------------------
architecture Behavioral of deneme3 is
signal fark: std_logic_vector(2 downto 0);
begin
        layer1: process(clk)            -- 1. neuron katmani girisleri okur,
                                        -- egitim setleri ile olan benzerlige bakar.
        begin
        for i in 1 to 50 loop
if(clk='1')then                 --clock=1 oldugunda islem yap
        fark(i)<=(s1-g1(i))*(s1-g1(i))+(s2-g2(i))*(s2-g2(i));   --aradaki vektorel uzakligi
bulmak icin
        end loop
end if;
        end process     layer1;
        layer2: process(clk,fark)    -- 2. neuron katmani farklarin toplamini hesaplar
        begin
        for i in 1 to 7 loop  --7 adet cikis grubuna olan uzakliklara bak
```

```
toplam(i)=toplam(i)+sonuc(i);
end loop;
        end process layer2;
        layer3: process(clk)              -- 3. neuron katmani en yaklasik sonucu disariya verir
        variable c1: std_logic_vector(2 downto 0);   --egitim sensor cikisi
                                                      -- en yakin sonuc seciliyor
        begin
        c1:=”001”;
        if (toplam(1)>toplam(2)) then
        c1:="010";
        end if;
        if (toplam(2)>toplam(3)) then
        c1:="011";
        end if;
if (toplam(3)>toplam(4)) then
        c1:="100";
        end if;
        if (toplam(4)>toplam(5)) then
        c1:="101";
        end if;
        if (toplam(5)>toplam(6)) then
        c1:="110";
        end if;
        if (toplam(6)>toplam(7)) then
        c1:="111";
        end if;
        sonuc<=c1;                        -- egitim sensor cikisini disariya ver
        end process layer3;
end Behavioral;
```

ÖZGEÇMİŞ

Doğum tarihi	16.02.1984	
Doğum yeri	İstanbul	
Lise	1998-2002	Haydarpaşa Anadolu Lisesi
Lisans	2002-2003	Ankara Üniversitesi Mühendislik Fak. Elektronik Müh.
	2003-2005	Yıldız Teknik Üni. Elektrik-Elektronik Fak. Elektronik ve Haberleşme Müh.
Yüksek Lisans	2005-2006	Yıldız Teknik Üni. Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Müh. Anabilim Dalı Elektronik Müh. Programı
Çalıştığı kurumlar		
	2004-2005	Indefia Electronics
	2005-2006	Bilko Otomasyon A.Ş.
	2006-Devam	Yeditepe Üniversitesi Elektrik-Elektronik Müh. Bölümü Araştırma Görevlisi