TÜRKÇE FONEMLERİN SINIFLANDIRILMASINDA KULLANILAN SİNİR AĞININ FPGA UYGULAMASI

FPGA IMPLEMENTATION OF A NEURAL NETWORK FOR TURKISH PHONEME CLASSIFICATION

ALPER UÇAR

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK ve ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

Olarak Hazırlanmıştır.

2007

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **ELEKTRİK ve ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan	:….. Prof. Dr. H. Selçuk GEÇİM
Üye (Danışman)	:….. Yrd. Doç. Dr. Ali Ziya ALKAR
Üye	:….. Yrd. Doç. Dr. Derya ALTUNAY
Üye	:….. Yrd. Doç. Dr. Harun ARTUNER
Üye	:….. Yrd. Doç. Dr. Kayhan İMRE

ONAY

Bu tez ...../...../..... tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

Prof. Dr. Erdem YAZGAN
Fen Bilimleri Enstitüsü Müdürü

Anneme ve Babama;

TÜRKÇE FONEMLERİN SINIFLANDIRILMASINDA KULLANILAN SİNİR AĞININ FPGA UYGULAMASI

Alper Uçar

ÖZ

Çekimli bir dil olması nedeniyle, Türkçe sesli ifade tanıma sistemlerinin fonem tabanlı gerçekleştirilmesi etkin bir yöntemdir. Bu tür tanıma sistemlerinde sesli ifadeyi oluşturan fonemlerin tek tek tanınması ve daha sonra birleştirilmesi yoluyla sesli ifade tanıma gerçekleşir.

Fonem tabanlı tanıma sistemlerinde en kritik işlem fonemlerin sınıflandırılmasıdır. Veri işleme gücünü paralel dağılmış yapısından, öğrenme/genelleme ve hatayı tolere etme yeteneğinden alan yapay sinir ağları gerçek-zamanlı örüntü tanıma problemleri için uygun ve etkin bir çözümdür.

Sinir ağlarının donanım olarak gerçeklenmesindeki temel motivasyon, sesli ifade tanıma gibi karmaşık problemler için gerçek-zamanlı örüntü tanıyacak hızlara ulaşma çabası ve bu tür yapıların gömülü sistemler olarak kullanılması gereksinimidir.

Bu tez kapsamında, fonem tabanlı Türkçe sesli ifade tanıma sistemlerinde kullanılacak bir radyal tabanlı fonksiyon ağı (RTFA) tasarlanmış ve donanım mimarisi oluşturulmuştur. Gerçeklenen donanım mimarisinin benzetim ve FPGA sentez sonuçları, yazılım sonuçlarına yakın başarı oranı ile gerçek-zamanlı fonem sınıflandırması yapabilen bir sistem oluşturulduğunu göstermektedir.

ANAHTAR SÖZCÜKLER: Türkçe sesli ifade tanıma, fonem sınıflandırma, sinir ağları, radyal tabalı fonksiyon ağı, donanım, FPGA

Danışman: Yrd. Doç. Dr. Ali Ziya ALKAR, Hacettepe Üniversitesi, Elektrik Elektronik Mühendisliği Anabilim Dalı.

FPGA IMPLEMENTATION OF A NEURAL NETWORK FOR TURKISH PHONEME CLASSIFICATION

Alper Uçar

ABSTRACT

It is an effective method to implement Turkish speech recognition systems by phoneme-based methods as words in Turkish are mostly produced by derivational and inflectional affixes to the roots. Phoneme based speech recognition systems are based on the recognition of the phonemes and then recognition of the speech by combining them.

Phoneme classification is considered as the most critical part of the process. An artificial neural network (ANN), which derives its computational power through inherent parallel structure and ability to learn and generalize, is an effective solution to real-time pattern recognition tasks.

Real-time recognition of speech and the need for embedded speech recognizers are the fundamental motivations behind implementation of ANNs on hardware.

In this study, hardware architecture of a Radial Basis Function Neural Network was designed and implemented for phoneme-based Turkish speech recognition systems. Synthesis results for FPGA indicate a real-time phoneme classifier is designed with a success rate close to the software simulations.

KEYWORDS: Turkish speech recognition, phoneme classification, artificial neural network, Radial Basis Function Neural Network, hardware, FPGA

Advisor: Assistant Prof. Dr. Ali Ziya ALKAR, Hacettepe University, Electrical and Electronics Engineering Department

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesi sırasında gerek yol göstericiliği gerekse sağladığı imkanlar nedeniyle danışmanım Yrd. Doç. Dr. Ali Ziya ALKAR'a,

çalışma süresince değerli katkılarını esirgemeyen Yrd. Doç. Dr. Harun ARTUNER, Dr. Koushik MAHARATNA, Prof. Dr. Fikret GÜRGEN, ve Araş. Gör. Kenan BOZDAŞ'a,

çalışmalarım boyunca daima yanımda olan sevgili Damla IŞIK'a,

ve yaşamım boyunca maddi ve manevi destekleri esirgemeyen annem Hüsniye UÇAR ve babam Dr. Tamer UÇAR'a teşekkürlerimi sunarım.

İÇİNDEKİLER DİZİNİ

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

KISALTMALAR DİZİNİ

BSRAM: Block SelectRAM

CLB: Configurable Logic Block

dRAM: Distributed RAM

FPGA: Field-programmable gate array

ISE: Integrated Software Environment™

LSFR: Linear Shift Feedback Register

RAM: Random Access Memory

RTFA: Radyal Tabanlı Fonksiyon Ağı

RTL: Register Transfer Level

XOR: Exclusive OR

XST: Xilinx® Synthesis Technology

VHDL: VHSIC Hardware Description Language

VHSIC: Very-High-Speed Integrated Circuit

YSA: Yapay Sinir Ağları

BÖLÜM 1

1 GİRİŞ

Ses insanlar arasında kullanılan doğal iletişim aracıdır. İnsanların ses yoluyla iletişim kurması doğal yollarla edinilen bir yetenek olduğundan, çoğu zaman bunun ne kadar karmaşık bir olgu olduğunun farkına varılmaz. İnsanın sesli ifade oluşturmak için kullandığı nefes borusu, ses telleri, ve ses yolu doğrusal olmayan özellikler içeren biyolojik yapılardır. Kişinin cinsiyeti, coğrafi konumu, eğitim seviyesi, ve psikolojik durumu gibi etmenler konuşmasını etkiler. Bunun yanı sıra konuşmanın yapıldığı ortama göre sesli ifade gürültü ve yankılanma nedeniyle bozulabilmektedir. Bütün bu değişkenler sesli ifade tanımayı karmaşık bir problem haline getirmektedir.

Sesli ifade tanıma; bilgisayar-insan etkileşimi, telefon santralleri, ev otomasyonu, ve otomatik tercüme hizmetleri gibi pek çok alanda gereksinim duyulan bir uygulama alanıdır. 1950'lerin başından beri bu alanda araştırmalar yürütülmektedir. 1970'lerde şablon eşleme (template matching), bilgi tabanlı sistemler (knowledge based systems), ve istatistiksel metotlar gibi yaklaşımlar sayesinde sesli ifade tanıma sistemleri geliştirmede önemli adımlar atılmıştır. Bütün karmaşıklığına rağmen insan beyninin sesli ifade tanıma ve diğer pek çok bilişsel alandaki üstün becerisi, beynin çalışma prensiplerini temel alan bir hesaplama modeli geliştirme konusunda çalışmalar yapılmasına önayak olmuştur. Günümüzde yapay sinir ağları (YSA) olarak adlandırılan bu alan, 1980'lerin ortalarına kadar istikrarsız bir gelişme göstermiş ancak çok katmanlı sinir ağları (MLP) için bir öğrenme algoritmasının bulunması ile beraber başta sesli ifade tanıma olmak üzere pek çok probleme uygulanır olmuştur.

Sesli ifade tanıma sistemleri, tanıma için seçilen birime göre ikiye ayrılabilir. Sözcük tabanlı sesli ifade tanıma sistemlerinde, tanıma için seçilen en küçük birim sözcük olurken, fonem tabanlı sistemlerde en küçük birim fonem olmaktadır. Türkçe, Fince ve Japonca gibi fonem tabanlı bir dildir ve Türkçe'de her harfin bir foneme karşılık geldiği söylenebilir. Bu tez kapsamında fonem tabanlı bir sesli ifade tanıma yaklaşımına temel oluşturacak fonem sınıflandırma çalışması, tek konuşmacı için benimsenmiştir.

Bunun için, ilk aşamada Türkçe fonemleri temsil edecek öznitelik vektörlerinin çıkartılması ile ilgili çalışmalar yürütülmüş ve sayısal filtre dizisi ve *cepstrum* tekniğine dayalı öznitelik vektörlerinin bulunduğu Türkçe fonem veritabanı oluşturulmuştur. Bu, başlangıcı ve sonu belirlenmiş fonemlerin, öznitelik vektörleriyle temsil edildiği bir veritabanıdır.

Türkçe fonemlerin sınıflandırılmasında YSA yaklaşımı benimsenmiş ve oluşturulan veritabanı sinir ağının eğitilmesinde kullanılmıştır. Bu bağlamda, öğrenme algoritması diğer pek çok modele göre hızlı yakınsayan radyal tabanlı fonksiyon ağı (RTFA) modeli kullanılmış ve uygun benzetim sonuçlarının alınmasının ardından, gerçek-zamanlı fonem sınıflandırabilen RTFA'nın donanım mimarisi oluşturulmuştur.

Sinir ağı tabanlı bir fonem sınıflandırma sisteminin donanım olarak gerçeklenmesindeki temel motivasyon, gerçek-zamanlı örüntü tanıyacak hızlara ulaşma çabasının yanı sıra, bu tür yapıların gömülü sistemler olarak kullanılması gereksinimidir. VHDL donanım tanımlama dili kullanılarak sayısal donanım mimarisi oluşturulan ve ardından FPGA sentezi yapılan sistem dört temel modülden oluşmaktadır. Kontrol birimi, sinir ağının genel denetimini sağlarken; K-ortalama kümeleme (K-means clustering) birimi öznitelik (feature) vektörlerini daha önceden belirlenmiş uygun sayıda kümeye bölerek, her kümenin merkezini (center) ve değişintisini (variance) bulmakta; Gauss birimi, K-ortalama kümeleme biriminin çıktılarına göre ve bir başvuru çizelgesi (look-up table – LUT) kullanarak gerekli radyal tabanlı fonksiyonların değerlerini hesaplamakta; ve son olarak LMS (Least mean squares) birimi en küçük ortalama kareler yöntemini kullanarak sinir ağının çıktısını oluşturan fonem sınıflarını belirlemektedir.

Bu şekilde fonem sınıflandırmada kullanılan RTFA'nın donanım mimarisi başarıyla gerçekleştirilmiştir. Çalışma, British Council *"Science Partnership Programme"* ve Hacettepe Üniversitesi Bilimsel Araştırmalar Birimi tarafından desteklenmiştir. British Council desteği kapsamında önce Bristol Üniversitesi ardından Southampton Üniversitesi ile ortak çalışmalar yürütülmüştür.

Bu tez kapsamında ikinci bölümde sesin genel özelikleri ve Türkçe'deki fonemlere ait bilgiler, üçüncü bölümde sesli ifade tanıma sistemleri hakkında bilgi verilmiştir. Dördüncü bölümde Türkçe fonem sınıflandırma süreci anlatılmakta ve benzetim sonuçları verilmektedir. Beşinci bölüm YSA'nın FPGA uygulamaları hakkındadır. Altıncı bölümde geliştirilen donanım mimarisi anlatılarak bir performans değerlendirilmesi yapılmış, yedinci bölümde ise sonuçlar sunulmuştur.

BÖLÜM 2

2 SESLİ İFADEYE GENEL BAKIŞ

2.1 Sesin Fiziksel Özellikleri

Akustik bir dalga olan ses (phon), üç temel özelliğe sahiptir [3]:

a) Sıklık (frequency)
b) Tını (waveform)
c) Şiddet (intensity)

Tiz sesler, yüksek sıklıktaki; pes sesler ise düşük sıklıktaki seslerin titreşimlerinden oluşur. Sağlıklı bir insan 20Hz ile 20KHz arasındaki seslere duyarlıdır ve 130Hz ile 15KHz sıklık aralığında ses üretebilir. İnsanda ses tellerinin titreşmesiyle oluşan vuru (impulse) dizisinin sıklığı sesin perdesi (pitch) olarak adlandırılır [4]. Ses perdesinin birimi *mel*'dir ve fizyolojide tanımlı perde,

$$m \cong 1127.01048\left(\log_e\left[1+\frac{f}{700}\right]\right) \qquad (2\text{-}1)$$

şeklinde hesaplanmaktadır [5]. İnsan kulağı, her biri farklı perdeye karşılık gelen sesin harmoniklerini ayrı ayrı duymaz. Ses, temel harmoniğe karşı gelen perdede algılanır.

Tını, üretilen sesin içerdiği harmoniklerdeki farklılıkların belirlediği bir özellik olduğundan, farklı insanların ürettiği seslerin ayırt edilmesini sağlar [6].

Ses şiddeti, sesi oluşturan akustik dalganın genliği ile orantılıdır. Sağlıklı bir insan 20dB ile 120dB şiddetindeki seslere duyarlıdır.

2.2 Fonem

Fonetik bilimine göre fonem, sesli ifadeyi oluşturan temel birimi olarak kabul edilmektedir. Fonem, sesin aksine soyut bir kavramdır ve bir sözcüğü diğerinden ayırt etmeyi sağlar. Örnek vermek gerekirse, *FAS* ve *FES* sözcüklerinde anlam

farklılaşmasını sağlayan /a/ ve /e/ fonemleridir. Türkçe'nin fonem tabanlı bir dil olduğu ve her harfin bir foneme karşılık geldiği kabul edilmektedir. Bu bağlamda Türk alfabesi fonetik olarak "kusursuz" bir alfabedir.

Öte yandan bir fonemin hece veya sözcük içindeki konumuna göre farklı şekillerde seslendirilmesi mümkündür. Bir foneme karşılık gelen tüm seslerin oluşturduğu kümenin elemanlarına eşsesli (allophone) denir.

Fonemler parçalı (segmental) ve parçalarüstü (suprasegmental) olarak iki grupta incelenebilir. Parçalı fonemler, harflerle temsil edilen fonemlerdir. Parçalar üstü özellik vurgu, uzatma, ezgi gibi sembollerle ifade edilemeyen sessel özelliklere verilen addır [4].

2.3 Türkçe'de Parçalı Fonemler

Türkçe'de parçalı fonemler;

a) Ünlüler
b) Ünsüzler
c) Kayan Ünlüler

şeklinde sınıflandırılır.

2.3.1 Ünlüler

Ünlüler, ses telleriyle oluşturulan titreşimin ses yolunda (vocal tract) rezonansa sokulmasıyla elde edilir. Türkçe'de ünlü olarak nitelendirilen 16 ses vardır. Her ünlünün bir açık bir de kapalı türü olduğundan, 16 sese karşılık gelen 8 fonem (/a/, /e/, /o/, /ö/, /ı/, /i/, /u/, /ü/) bulunur. Ünlüler, çıkış biçimlerine (çene açısı ve dudakların biçimi) ve çıkış yerlerine (dilin konumu) göre sınıflandırılabilirler [7]. Bu tür bir sınıflandırma Çizelge 2.1'de verilmiştir.

Çizelge 2.1 Türkçe'de ünlülerin çıkış yeri ve biçime göre sınıflandırılması

		a	**e**	**o**	**ö**	**ı**	**i**	**u**	**ü**
Çene Açıklığı	Geniş	x	x	x	x				
	Dar					x	x	x	x
Dudak Biçimi	Düz	x	x			x	x		
	Yuvarlak			x	x			x	x
Dilin Konumu	Arka	x		x				x	
	Orta					x			
	Yuvarlak Ön				x				x
	Düz Ön		x				x		

Türkçe ünlü fonemlerin sözcük içinde yer aldığı konumlar incelendiğinde /a/, /e/, /ı/, /i/, /ü/ fonemlerinin sözcük içinde ön, iç ve son hecelerde bulunduğu; /o/ ve /u/ fonemlerinin ön ve iç hecelerde bulunduğu; /ö/ foneminin ise ön ve orta hecelerde bulunduğu gözlenmiştir [8].

2.3.2 Ünsüzler

Türkçe'de 20 ünsüz fonem (/b/, /c/, /ç/, /d/, /f/, /g/, /h/, /j/, /k/, /l/, /m/, /n/, /p/, /r/, /s/, /ş/, /t/, /v/, /y/, /z/) vardır. Yazılı dilde kullanılan "ğ" harfinin ses niteliği taşıyıp taşımadığı tartışma konusudur. Türkçe'de ünsüzlerin çıkış yeri, biçimi ve titreşimine göre sınıflandırılması Çizelge 2.2'de verilmiştir.

Çizelge 2.2 Türkçe'de ünsüzlerin çıkış yeri, biçimi ve titreşimine göre sınıflandırılması

Özellikler		***b***	***c***	***ç***	***d***	***f***	***g***	***h***	***j***	***k***	***l***	***m***	***n***	***p***	***r***	***s***	***ş***	***t***	***v***	***y***	***z***
Çıkış Biçimi	Patlamalı	x					x			x				x				x			
	Geniz											x	x								
	Çarpmalı														x						
	Yan Daralmalı										x										
	Sızmalı		x	x		x		x	x							x	x		x	x	x
Çıkış Yeri	Çiftdudak	x										x		x							
	Dudak-diş					x													x		
	Dilucu-dişardı				x												x				
	Dilucu-dişeti												x		x	x					x
	Dil-öndamak		x	x					x								x			x	
	Dilucu-öndamak										x										
	Dilucu-artdamak						x			x											
	Gırtlak							x													
Ses teli titreşimi	Ötümlü	x	x		x		x		x		x	x	x		x				x	x	x
	Ötümsüz			x		x		x		x				x		x	x	x			

Türkçe'de ünsüzlerin sözcük içinde bulunduğu konuma göre özellikleri aşağıda verilmiştir [4]:

- /b/, sözcükte ön ve iç ses olabilir. Son seste ötümsüzleşip /p/ sesine dönüşür.
- /c/, iki sesten oluşan bileşik bir sestir. İç seste bunlardan birisi düşerken, son seste çok nadir bulunur
- /ç/, iki sesten oluşan bileşik bir sestir. İç seste bunlardan birisi (/t/) düşer.
- /d/, sözcükte ön ve iç ses olabilir. Son seste ötümsüzleşip /t/ sesine dönüşür.
- /g/, sözcükte ön ve iç ses olabilir. Son seste ötümsüzleşip /k/ sesine dönüşür.
- /h/, sözcükte ön, iç ve son ses olabilir.

- /j/, sözcükte ön, iç ve son ses olabilir.
- /k/, sözcükte ön, iç ve son ses olabilir.
- /l/, sözcükte ön, iç ve son ses olabilir.
- /m/, sözcükte ön, iç ve son ses olabilir.
- /n/, sözcükte ön, iç ve son ses olabilir.
- /p/, sözcükte ön, iç ve son ses olabilir.
- /r/, ön seste birden çok çarpmalı, iç seste tek çarpmalı, son seste sızdırmalıdır.
- /s/, sözcükte ön, iç ve son ses olabilir.
- /ş/, sözcükte ön, iç ve son ses olabilir.
- /t/, sözcükte ön, iç ve son ses olabilir.
- /v/, sözcükte ön, iç ve son ses olabilir.
- /y/, sözcükte ön, iç ve son ses olabilir. Son ses olduğunda ötümsüzdür.
- /z/, sözcükte ön, iç ve son ses olabilir. Son ses olduğunda ötümsüzdür.

2.3.3 Kayan Ünlüler

Aynı hece içinde yan yana duran ve tek sesmiş gibi çıkarılan ünlüler kayan ünlü olarak tanımlanır. Türkçe'de iki ünlü aynı hece içinde yan yana bulunmaz. Ancak yabancı dillerden alıntı sözcüklerde /y/ ve /v/ ünsüzlerinin araya girdiği ve bunlardan bazılarının yazıya da geçtiği gözlenir (fiat – fiyat örneğinde olduğu gibi) [4]. Ayrıca ses niteliği tartışmalı "ğ" ünlü kaymasına neden olmaktadır.

2.4 Hece

Hece, bir göğüs atışı periyodunda çıkarılan sesler bütünüdür. Göğüs atışı süresince soluk, belirli bir basınçta ses yolundan dışarı çıkar. Bu basınçlı hava ile ses tellerinin titreşimi ünlü sesleri meydana getirir. Heceler, çıkan ünlülerin ses yolunda engellenmesi ile oluşmaktadır ve bu engel ünsüz seslerle sağlanmaktadır. Hece türleri ünsüzlerin dağılımına göre belirlenir. Türkçe'de hece türleri Çizelge 2.3'te verilmiştir [2].

Çizelge 2.3 Türkçe'de hece türleri (Ü:**ü**nlü – Z: ünsü**z**)

Ü	*o*-kul
ÜZ	*al*-tın
ZÜ	*ku*-rak
ÜZZ	*üst*
ZÜZ	*bul*-gu
ZÜZZ	*sırt*-lan
ZZÜZ	*tren*
ZZÜ	*spi*-ker
ZÜZZZ	*tekst*
ZZÜZZ	*flört*
ZZZÜZ	*stres*

Çizelge 2.3'te listelenen hece türlerinden ilk altısı Türkçe sözcüklerde kullanılmakta, diğerleri ise yabancı dillerden Türkçe'ye geçmiş sözcüklerde bulunmaktadır.

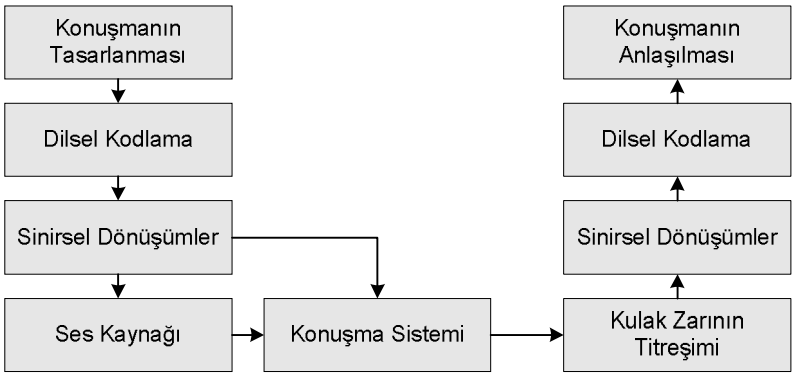
BÖLÜM 3

3 SESLİ İFADE TANIMA SİSTEMLERİ

İnsanlar arası sesli iletişim Şekil 3.1'deki gibi modellenebilir [2]. Bu tür bir modelde sesli iletişim süreci;

- Konuşmacıda oluşan düşüncenin sözcüklere dönüştürülmesi,
- Ses telleri ve konuşma sistemi aracılığı ile sözcüklerin seslendirilmesi,
- Seslerin dinleyicinin kulağına ulaşması,
- Sesin, dinleyicinin işitsel sinirleri aracılığı ile beyne iletilmesi
- Beynin titreşimleri dilsel karşılıklara çevirmesi,
- Dilsel kodların, yani sözcüklerin yan yana getirilerek anlam çıkarılması,

şeklinde özetlenir.



Şekil 3.1 İnsanlar arasında sesli iletişim

Sesli ifade sistemlerinin amacı dinleyicinin yerini almaktır. Tüm sesli ifade tanıma sistemlerinde temel ilke, akustik girdiyi daha önceden kaydedilmiş referans şablonlarıyla karşılaştırmaktır. Karşılaştırma sonucunda girdinin en çok benzediği şablon, o girdinin simgesel gösterimi kabul edilir.

3.1 Sesli ifade Tanıma Sistemlerinde Başarı Kıstasları

Sesli ifade tanıma sistemlerinde başarı oranı, değerlendirildiği kıstaslara göre değişmekle beraber, sınırlı koşullar söz konusu olduğunda dinleyiciye yakın başarı oranları elde edilebilmektedir. Genel koşullar altında ise aynı başarı oranının yakalanması zordur. Değerlendirme kıstasları aşağıdaki şekilde özetlenebilir.

3.1.1 Korpüs Büyüklüğü

Genel bir kural olarak, az sayıda sözcük içeren bir sistemde sözcükleri birbirinden ayırmak görece daha kolaydır. Sözcük sayısı arttıkça tanıma zorlaşır. Ancak sözcük sayısı az olsa dahi birbirine karıştırılabilen sesler içeren bir korpüste tanıma zorlaşacaktır.

3.1.2 Konuşmacıdan Bağımsızlık

Konuşmacıya bağlı sistemlerde tek bir kişi için referans şablonları hazırlanır. Sesli ifadeler konuşmacıya bağımlı unsurlar içerdiğinden, konuşmacıdan bağımsız bir sistem tasarlanması için, söz konusu farklılıkların tanımayı etkilemeyecek biçimde ortadan kaldırılması gerekmektedir.

3.1.3 Ayrışık, Süreksiz ve Sürekli Konuşma

Ayrışık konuşma, kısa aralıklarla seslendirilen sözcüklerden; süreksiz konuşma yapay olarak birbirinden ayrılmış cümlelerden; sürekli konuşma ise ara verilmeden seslendirilen sözcük ve cümlelerden oluşur. Ayrışık ve süreksiz konuşmaya dayalı sistemlerde yüksek başarı oranları yakalamak görece kolaydır. Bunun nedeni sözcüklerin başlangıç ve bitiş yerlerinin belirli olmasının yanısıra sözcüklerin anlaşılır şekilde telaffuz edilmesidir.

Sürekli sözcük tanımada iki farklı yöntem kullanılmaktadır. Bunlardan sürekli sözcük tanıma yönteminde, sesli ifade içinde sözcük sınırları bulunarak sözcük tabanlı tanıma yapılır. Bu yöntemde sözcük yakalamak için *time warping* türevi algoritmalar kullanılır. Sürekli sesli ifade tanıma yönteminde ise tanıma fonem gibi alt düzey ses birimlerine indirgenir. Fonem tabanlı sesli ifade tanımada, sesli ifadeler üzerinde fonetik çözümleme ve kesimleme yapılır. İkinci aşamada ise fonem kümelemesi ile sözcük çakıştırma gerçekleştirilir [4].

Sözcük tabanlı tanıma sistemlerinde başarım oranı daha yüksek olmaktadır. Bunun nedeni fonem tabanlı sistemlerde fonemlerim ardı ardına sıralanması sırasında ortaya çıkan geçişlerin yarattığı olumsuz etkilerdir. Ancak fonemler sesli ifadelerin yapıtaşları olarak kabul edilirse fonem tabanlı bir dilde tanınacak sözcük

sayısının sınırı yoktur. Sözcük tabanlı sistemlerde ise sözcük sayısı sınırlı tutulmak zorundadır.

3.1.4 Spontane Konuşma

Spontane konuşma ile hazır bir metni okuyarak konuşma ya da dikte etme arasında önemli farklılıklar vardır ve bu başarı oranını büyük ölçüde etkiler. Spontane konuşmada kimi sesler konuşmacı tarafından yutulmakta, akıcılığı bozan kimi sesler konuşmacı tarafından eklenmekte, sözcükler doğru telaffuz edilmemekte ve cümleler tamamlanmayabilmektedir.

3.2 Sesli ifade Tanımada Kullanılan Yaklaşımlar

3.2.1 Şablon Tabanlı Yaklaşımlar

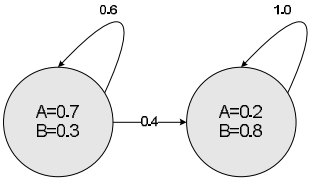
Şablon tabanlı sistemlerde, tanınması istenilen sesli ifade önceden kaydedilmiş şablonlarla karşılaştırılır ve en uygun eşleşme bulunmaya çalışılır. Sözcüklerin şablon olarak kullanıldığı sistemlerde işleyici zamanı ve bellek gereksinimi gibi kısıtlamalar nedeniyle sözcük sayısı sınırlı tutulmalıdır. Bu nedenle, konuşmanın doğasından kaynaklanan değişimleri modellemek güç olacağından şablon tabanlı sistemler elverişsiz olabilmektedir. Sözcük altı birimlerin (ses, fonem, hece) şablon olarak kullanıldığı sistemlerde ise, seslerin sayıca kısıtlı olması ve sözcüklerin bu birimlerin kombinasyonu ile ifade edilebilmesi gerekmektedir. *Time warping* ve YSA, sesli ifade tanımada şablon tabanlı yaklaşımlar olarak düşünülebilir.

3.2.2 Bilgi Tabanlı Yaklaşımlar

Bilgi tabanlı sistemlerde, konuşmanın doğasından kaynaklanan değişimler bir uzman tarafından sisteme girilir. Bu tür bir yaklaşımda gerek konuşmaya özgü değişimlerin eksiksiz kodlanması gerekse kodlanan bilgilerin başarıyla kullanılması elverişsiz olabilmektedir [1].

3.2.3 İstatistiksel Yaklaşımlar

Konuşmanın doğasından kaynaklanan değişimlerin istatistiksel modellerle ifade edildiği yaklaşımlardır. *Saklı Markov modeli* (HMM) [9] yaygın olarak kullanılan bir istatistiksel yaklaşımdır. Saklı Markov modelinde sesli ifade üretiminin sonlu durumlu bir makine (finite state machine) tarafından üretildiği varsayılır. Sesli ifade üretilirken, bir durumdan diğerine geçildiği ve her durumda belirli bir sesin üretildiği varsayılır. Hangi duruma geçileceği ve hangi sesin üretileceği olasılık dağılımları ile kontrol edilir. Şekil 3.2'de *A* ve *B* biçiminde iki çıkışı olan basit bir Markov modeli gösterilmiştir. Şekilde durumlar daire, geçişler ve geçiş olasılıkları oklarla gösterilmektedir.



Şekil 3.2 Basit bir HMM örneği

Bir HMM aşağıdaki bileşenlerden oluşur:

- $\{s\}$: Durum kümesini tanımlar.
- $\{a_{ij}\}$: i durumundan j durumuna geçiş olasılığıdır.
- $\{b_{ij}(k)\}$: k sembolünün i durumundan j durumuna geçişi sırasında üretilme olasılığıdır.

Saklı Markov modeli çerçevesinde üç soruna yanıt arayarak modelin pratikte kullanılması sağlanır [4]. Bu sorunlar değerlendirme sorunu, kod çözümü sorunu ve öğrenme sorunudur. Değerlendirme sorunu çerçevesinde bir dizi gözlemin verilen bir modelce oluşturulma olasılığı araştırılır. Bu amaçla *forward algorithm* [10] adı verilen bir yöntem kullanılır. Kod çözme sorunu eldeki bir dizi gözlemin hangi durum dizisi ile üretildiği sorununa yanıt arar. Bunun için *Viterbi* algoritması [11] kullanılır. Öğrenme aşamasında model parametreleri forward-backward algoritması [12] kullanılarak eniyilenir.

3.3 Yapay Sinir Ağları

YSA biyolojik sinir ağlarından esinlenerek geliştirilmiş bilgi işlem sistemleridir. Bir sinir ağı çok sayıda basit işlemciden (nöron) oluşur ve bu işlemciler farklı biçimlerde ifade edilebilen sayısal verileri taşıyan bağlantılar (synapse) ile birbirlerine bağlıdır. Bir sinir ağı insan beynini iki yönden taklit eder [13]:

- Bilgi ya da tecrübe bir öğrenme süreci sonunda kazanılır.
- Nöronlar arası bağlantılar edinilen bilgiyi depolamak için kullanılır.

YSA'nın genel özellikleri aşağıdaki gibi sıralanabilir:

- **Öğrenme yeteneği.**YSA, herhangi bir girdi/çıktı veri kümesi arasında ilişki kurabilecek şekilde eğitilebilir. "Öğrenme" nöronlar arası bağlantıların verilen girdi/çıktı kümesine göre düzenlenmesidir.

- **Genelleme.** Öğrenme sonucunda sinir ağı, eğitim sırasında kullanılan vektörlerin yanısıra aynı girdi/çıktı kümesine ait yeni vektörleri (test vektörleri) de sınıflandırabilir. Aynı sesli ifadeye ait akustik örüntüler birebir aynı olamayacağından, bu özellik sesli ifade tanımada önem arz etmektedir.

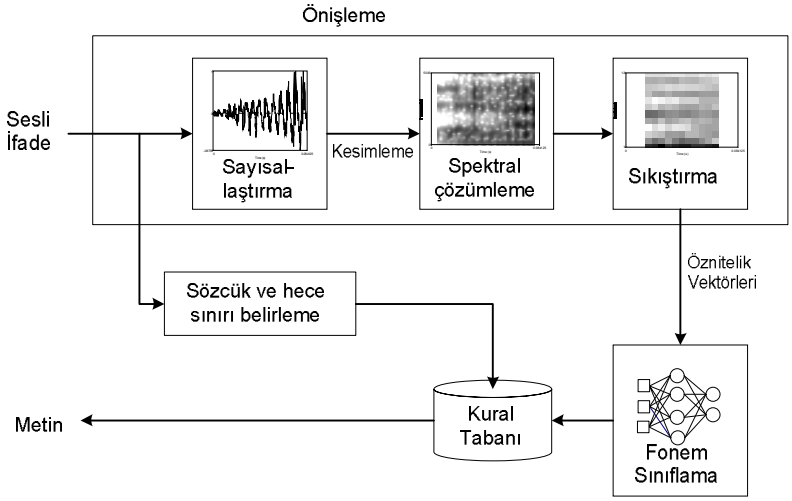
- **Hata toleransı.** YSA, gürültüye ve hasar görmüş girdilere karşı toleranslıdırlar. Sesli ifadeler gürültüye maruz kaldığından, sinir ağlarının bu özelliği de önemlidir. Öte yandan sinir ağının donanım uygulamalarında nöronlar arası bazı bağlantılarda kusurlu çalışmalar olabilir. YSA'nda bilgi, çok sayıda bağlantı arasında dağıtılmış olduğundan, bu tür bir kusur sonucu sistem performansında önemli bir düşüş gözlenmez.

- **Uyarlanabilirlik.** YSA, bağlantılarını çevredeki değişimlere uyarlama yeteneğine sahiptir. Bu sayede başlangıçta kısıtlı sayıda fonem sınıflandıran bir sinir ağına, eğitim için kullanılan girdi/çıktı kümesini genişleterek yeni fonemleri de sınıflandıracak şekilde genişletmek mümkündür.

- **VLSI gerçeklenebilirliği.**YSA'nın büyük çapta paralel yapısı, donanıma uygulanması için elverişlidir. Çoğu sinir ağı modeli çok sayıda çarpma ve toplama işlemi gerektirir ve bu tip işlemler sistolik yapılarla hızlı bir şekilde gerçekleştirilebilir. Bir sinir ağı modelini donanıma uyarlamak, ağın öğrenme sürecini gerçek-zamanlı hızlarda işlem yapabilecek biçimde hızlandıracaktır.

3.3.1 Fonem Tabanlı Sesli ifade Tanıma

Sesli ifade tanıma sistemlerinde sinir ağları yaygın olarak kullanılmaktadır [14–18]. YSA kullanılarak geliştirilen bir fonem tabanlı ayrışık sesli ifade tanıma sistemi ana hatlarıyla Şekil 3.3'te verilmiştir [2].



Şekil 3.3 Fonem tabanlı sesli ifade tanıma sistemi

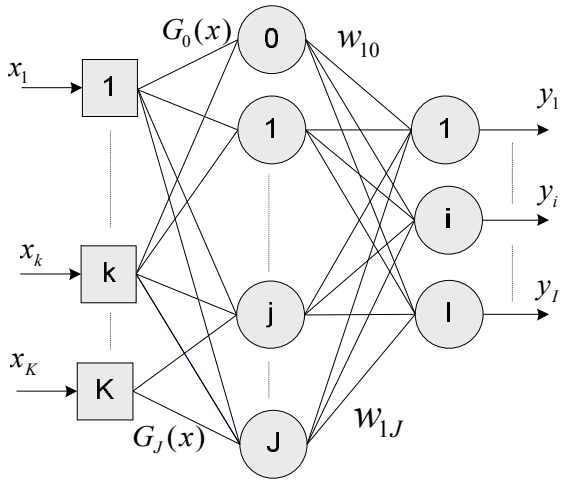
Sesli ifade tanıma sürecinin ilk aşaması sesli ifadenin sayısallaştırılmasıdır. Bu işlemden her bir fonemi temsil eden öznitelik vektörlerinin çıkartılmasına kadar geçen süreç önişleme süreci olarak tanımlanır. Önişleme süreci sırasıyla fonem kesimleme, *time warping,* filtreleme, pencereleme, spektral çözümleme, ve *cepstrum* katsayıları ile modelleme aşamalarından oluşur. Spektral çözümleme sonrası sesli ifadenin *cepstrum* katsayıları ile modellemesi sesin sıkıştırılması olarak tanımlanabilir. Bu aşamalar Bölüm 4'te ayrıntılı olarak verilmiştir. Önişleme sürecinden sonra bir sinir ağı aracılığı ile öznitelik vektörleri fonetik sınıflara

dönüştürülür. Sözcük sınırlanırının belirlenmesi sonucu kural tabanlı bir yöntemle fonemlerin metne dönüştürülmesi sağlanabilir [2].

3.3.2 Radyal Tabanlı Fonksiyon Ağı

YSA kullanılarak yapılan sınıflandırma problemlerinde sınıflar arasındaki ilişkiler doğrusal değilse tek katmanlı ağlar yetersiz kalmaktadır. 1980'lerin ortalarında çok katmanlı sinir ağları için bir öğrenme algoritmasının [19] geliştirilmesi, ağın gizli katmanında yeterli sayıda nöron kullanmak şartıyla, herhangi bir girdi/çıktı takımının başarıyla yaklaşıklanmasına (approximation) imkan vermiştir. Bu nedenle çok katmalı sinir ağları *universal approximator* olarak tanımlanmışlardır. RTFA'nın da *universal approximator* kriteri gösterdiği Park ve Sandberg tarafından kanıtlanmıştır [20-21]. Ancak bu kriter yaklaşıklamanın kalitesi hakkında bir bilgi vermemektedir. Girosi ve Poggio, *universal approximator* kriterine ek olarak, en iyi yaklaşıklama kriterini tanımlamış ve RTFA'nın bu kritere de uyduğunu göstermiştir [22]. Çok katmanlı sinir ağlarının, bir dizi sürekli fonksiyon kullanılarak yapılan çalışmalarda bu kritere uymadığı görülmüştür.

RTFA [23-24], üç katmanlı ve ileri-bildirimli (feed-forward) bir sinir ağı modelidir. K-boyutlu giriş vektör uzayı, *J* adet radyal tabanlı fonksiyonu ve *I*-boyutlu çıkış vektör uzayı olan bir RTFA'nın topolojisi Şekil 3.4'te verilmiştir.



Şekil 3.4 RTFA topolojisi

Giriş katmanı, giriş vektörlerinin sisteme beslendiği arayüzdür. Giriş katmanına beslenen giriş vektörleri kümesi $[\bar{x}(1),\ldots\bar{x}(p),\ldots\bar{x}(PT)]$ olarak gösterilebilir. Burada PT toplam örüntü sayısını temsil etmektedir. Fonem sınıflandırma problemi söz konusu olduğunda giriş vektörlerini fonetik öznitelik vektörleri oluşturur. Giriş katmanı ile orta katman arasında bağlantı yoktur. Orta ya da gizli katman, giriş vektörleri kümesine temel olacak fonksiyonları biçimlendirir. Bu fonksiyonlar radyal simetri özelliği gösterirler. Bu tür ağlara "radyal tabanlı fonksiyon ağları" denilmesinin nedeni de budur. Eş.3.1'de verilen Gauss fonksiyonu en yaygın kullanılan radyal tabanlı fonksiyondur.

$$G_j[\bar{x}(p)] = \exp-\left[\frac{\left\|\bar{x}(p)-\bar{\mu}_j\right\|^2}{2\sigma_j^{\,2}}\right], \quad j=1,2,\ldots J, \tag{3.1}$$

Eş.3.1'de $G_j[\bar{x}(p)]$, gizli katmanda bulunan j. fonksiyonu; $\bar{x}(p)=[x_1(p),\ldots x_K(p)]$ *p.* *K*-boyutlu giriş vektörünü; $\bar{\mu}_j=[\mu_1,\ldots\mu_K]$ $j.$ fonksiyonun *K*-boyutlu merkez noktasını; σ_j ise değişintisini temsil etmektedir. Ağın çıkış katmanı ise örüntü sınıfları ile ilişkilidir:

$$y_i(\bar{x}(p)) = \sum_{j=0}^{J} w_{ij} G_j[\bar{x}(p)]. \tag{3.2}$$

Eş.3.2'de y_i, ağın $i.$ çıkışını; $\sum w_{ij}$ ise bu çıkışa ait bağlantı vektörünü temsil etmektedir.

Radyal tabanlı fonksiyon ağlarında temel fikir, bir grup radyal taban fonksiyonu istenilen fonksiyonuna yaklaşacak şekilde ağırlıklandırarak toplamaktan ibarettir. Bu açıdan bakıldığında, öğrenme süreci çok boyutlu uzayda girdi/çıktı kümesine eğri uydurmak olarak tanımlanabilir. RTFA'nda, çok katmanlı sinir ağından farklı olarak, ağ parametreleri önceden belirlenir. Ağ parametreleri belirlendikten sonraki öğrenme süreci tek katmanlı sinir ağının öğrenme sürecine eştir. İşte bu yüzden RTFA, çok katmanlı sinir ağına göre çok daha hızlı sonuç verebilmektedir.

RTFA'nda öğrenme iki aşamada gerçekleşir:

- Ağ parametrelerinin (Gauss fonksiyonlarının merkezi ve değişintisi) ve fonksiyonların değerlerinin bulunması
- Gizli katman ile çıkış katmanı arasındaki bağlantı değerlerinin hesaplanması

Ağ parametrelerinin hesaplanmasında K-ortalama kümeleme [25] ve *Self-Organizing MAP* (SOM) [26] gibi güdümsüz (unsupervised) öğrenme algoritmalarının yanısıra *gradient descent* türü güdümlü (supervised) algoritmalar da kullanılabilir. Tez kapsamında ağ parametrelerinin belirlenmesinde oldukça hızlı yakınsayan ve donanım olarak gerçeklenmeye elverişli olan K-ortalama kümeleme algoritması kullanılmıştır.

K-Ortalama Kümeleme

K-boyutlu uzayda PT adet örüntü vektörü $\left[\overline{x}(1),\ldots\overline{x}(p),\ldots\overline{x}(PT)\right]$ olduğu varsayılsın. K-ortalama kümeleme algoritması verilen PT adet vektörü, sayısı önceden belirli, *J*[1] adet kümeye ayırmada ve her bir kümenin merkez noktasını bulmada kullanılır. Algoritmanın temelinde Eş.3.3'de verilen uzaklık kriterine bağlı bir hata fonksiyonunu özyineli (iterative) olarak enküçültmek vardır. Kullanılan uzaklık ölçütü *Euclidean*, *Manhattan,* ya da *Mahalanobis* [27] gibi farklı ölçütler olabilir.

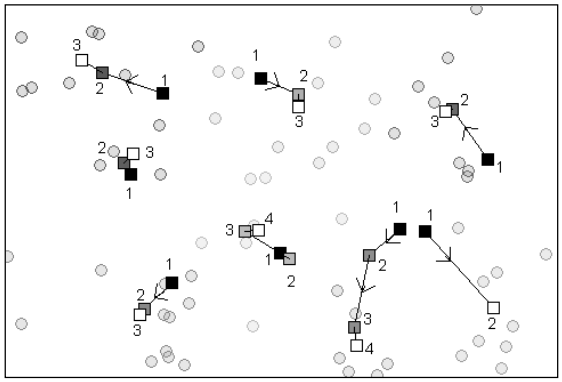
$$\varepsilon = \sum_{p=1}^{PT}\sum_{j=1}^{J} z_{jp}\left\|\overline{x}(p)-\overline{\mu}_j\right\|^2 \qquad (3.3)$$

Eş.3.3'te z_{jp}; $x(p)$ *j.* kümeye aitse 1, değilse 0 değeri alan ikil bir göstergedir.

[1] Algoritmanın aslında bu sayı *K* olup, simglemden kaynaklanan bir karışıklığa yol açmaması açısından *J* denilmiştir.

Algoritma 4 adımda hata fonksiyonunu enküçültmeye çalışır:

- Küme sayısının belirlenmesi,
- Her küme için merkez noktalarının ilklendirilmesi (initialization),
- Veri takımının en yakın merkez noktası ilkesine göre sınıflandırılması,
- Merkez noktalarının güncellenen veri takımına göre yeniden hesaplanması.



Şekil 3.5 K-ortalama kümeleme ile 64 veri noktasının 4 özyinelemede 8 kümeye ayrılması.

K-ortalama kümeleme ile merkez noktalarının nasıl güncellendiği Şekil 3.5'te, algoritmanın aşamaları ise aşağıda verilmiştir.

1. *J* adet kümenin merkez noktaları rasgele seçilir. Genellikle, PT adet örüntü vektörünün ilk *J* adeti merkez alınır.

2. $t.$ özyinelemede örüntü vektörleri $\left(\sum_{p=1}^{PT} \overline{x}(p)\right)$, *J* adet kümeye aşağıdaki eşiklik kullanılarak dağıtılır.

$$x(p) \in S_j(t) \text{ eğer } \left\|\overline{x}(p) - \overline{\mu}_j(t)\right\| < \left\|\overline{x}(p) - \overline{\mu}_{j^*}(t)\right\|, j^* = 1,\ldots J, j^* \neq j.. \qquad (3.4)$$

Eş 3.4'te $S_j(t)$, merkez noktası $\overline{\mu}_j(t)$ olan örüntü kümesini göstermektedir.

3. Kümelerin merkez noktaları yeniden hesaplanır:

$$\mu_j(t+1) = \frac{1}{N_j} \sum_{x \in S_j(t)} x, j = 1, \ldots J \qquad (3.5)$$

Eş.3.5'te N_j, $S_j(t)$ kümesinde bulunan örüntü sayısıdır.

4. Eş. 3.3'te tanımlı hata fonksiyonu belirli bir değerin altına ininceye kadar ya da $\overline{\mu}_j(t+1) \approx \overline{\mu}_j(t)$ koşulu gerçekleşinceye kadar özyinelemelere devam edilir.

K-ortalama kümeleme algoritması ile merkezi hesaplanan bir Gauss fonksiyonunun değişintisi, merkezi ile yakın komşuluğundaki birkaç Gauss fonksiyonunun merkezleri arasındaki uzaklığın ortalamasına eşit alınabilir [28].

En Küçük Ortalama Kareler Algoritması

Ağ parametreleri belirlenmiş RTFA'nda öğrenme süreci, tek katmanlı sinir ağı modeline indirgenmiş olur. Gizli katman ile çıkış katmanı arasındaki bağlantı değerleri en küçük ortalama kareler (LMS) algoritması ile hesaplanabilir. RTFA söz konusu olduğunda, LMS algoritmasının girdisi gizli katmanda bulunan Gauss fonksiyonlarının değerleridir:

$$\left(G\left[\overline{x}(1)\right], \ldots, G\left[\overline{x}(p)\right], \ldots G\left[\overline{x}(PT)\right]\right) \qquad (3.5)$$

Eş.3.4'te tanımlı $G\left[\overline{x}(p)\right]$, *J* boyutlu bir vektördür:

$$G\left[\overline{x}(p)\right] = \left[G_0\left[\overline{x}(p)\right], \ldots, G_j\left[\overline{x}(p)\right], \ldots, G_J\left[\overline{x}(p)\right]\right] \qquad (3.6)$$

LMS güdümlü bir algoritmadır ve her bir girdi vektörü için istenilen çıktılar $\left(\overline{d}(1), \ldots, \overline{d}(p), \ldots, \overline{d}(PT)\right)$ önceden biliniyor olmalıdır. Her girdi/çıktı çifti için bir hata fonksiyonu Eş.3.7'daki gibi tanımlanabilir:

$$E = \frac{1}{2}\left[d_i(p) - y_i(p)\right]^2 = \frac{1}{2}\left[d_i(p) - \sum_{j=1}^{J} w_{ji} G_j\left[x(p)\right]\right]^2 . \quad (3.7)$$

Eş 3.4'te d_i, çıkış katmanındaki $i.$nöron için istenilen çıktı değeridir. Hata fonksiyonu türevlenebilir olduğundan enküçültmek için *gradient descent* yöntemi uygulanabilir:

$$\overline{w}_{ij}(n+1) = \overline{w}_{ij}(n) + \eta\left(y_i(p) - d_i(p)\right) G_j\left[\overline{x}(p)\right]. \quad (3.8)$$

Eş.3.6 delta ya da *Widrow-Hoff* öğrenme kuralı olarak bilinir ve η, 0 ile 1 arasında seçilmesi gereken öğrenme oranı sabitidir.

K-ortalama kümeleme ve LMS algoritması kullanan bir RTFA için öğrenme süreci aşağıdaki gibi özetlenebilir:

1. K-ortalama kümeleme algoritması ile $\left[\overline{x}(1),\ldots\overline{x}(p),\ldots\overline{x}(PT)\right]$ örüntü kümesi için *J* adet Gauss fonksiyonun merkez ve değişinti değerleri hesaplanır.
2. Gizli katman ile çıkış katmanı arasındaki bağlantılara küçük rasgele değerler atanır.
3. *p*. örüntü sisteme verilir.
4. $\overline{x}(p)$ için; gizli katmanda bulunan her bir Gauss fonksiyonunun aldığı değer $\left(G_j\left[\overline{x}(p)\right]\right)$ Eş.3.1, çıkış katmanında bulunan her bir nöron aldığı değer (y_i) Eş.3.2 kullanılarak hesaplanır.
5. $1 \le i \le I$ için, eğer $y_i(p) = d_i(p)$[1] koşulu sağlanıyor ise 6. adıma geçilir. Bu koşul sağlanmıyor ise Eş.3.8 ile bağlantı değerleri güncellenir.
6. $p = PT$ koşulu sağlandıysa ve Eş.3.7'de tanımlı hata fonksiyonu istenilen değerin altında ise öğrenme süreci sona erer. Aksi durumlarda $p,\ p+1$'e eşitlenir ve 3.adıma dönülür.

[1] Pratikte bu koşul sağlanmadan, bir *hard limiter* kullanılarak sonuç alınmıştır

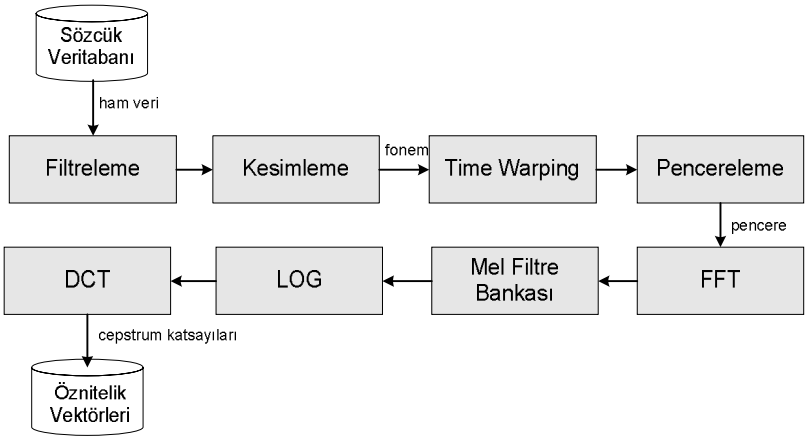
BÖLÜM 4

4 TÜRKÇE FONEM SINIFLANDIRMA SÜRECİ

Fonem sınıflandırma sürecinde ilk aşama sesli ifade verilerinin hazırlanmasıdır. Veriler hazırlanırken Türkçe'deki tüm fonemlerin temsil edilmesi sağlanmış, ayrıca aynı foneme ait eşsesler bulunduğu dikkate alınarak, her bir fonemin sözcük içinde başta, ortada ve sonda bulunduğu sözcükler seçilmiştir. Eğitim için Mengüşoğlu'nun korpüsü [2] kullanılmış, ünlü fonemler için 10, ünsüz fonemler için 8 sözcük seçilmiştir. Korpüste bulunan 240 sözcüğün her biri bir konuşmacı tarafından 2 kere seslendirilmiş, bu setlerden biri eğitim, diğeri test için kullanılmıştır. Sesli ifadeler oda ortamında 16KHz sıklığında ve 16-bit genlik düzeyinde bilgisayara kaydedilmiştir. Eğitim ve test sırasında kullanılan veritabanları için toplam 480 fonem kesimlenmiştir. Sözcükler Ek 1'de verilmiştir.

4.1 Önişleme

Veritabanında bulunan ham veriler, çok sayıda örnek (sample) içerdiğinden doğrudan RTFA'nın giriş katmanına uygulanmaya elverişli değildir. Bunun yerine ham veri üzerinde belirli kesimleri temsil edebilecek öznitelik vektörlerinin oluşturulabilir. Bu süreç, önişleme süreci olarak anılır. Oluşturulacak öznitelik vektörlerinin fonemleri en iyi şekilde temsil etmesi, işlenebilir büyüklükte ve ayırt edici özelliklere sahip olması gerekmektedir. Gerçekleştirilen önişleme süreci Şekil 4.1'de gösterilmiştir.



Şekil 4.1 Önişleme aşamaları

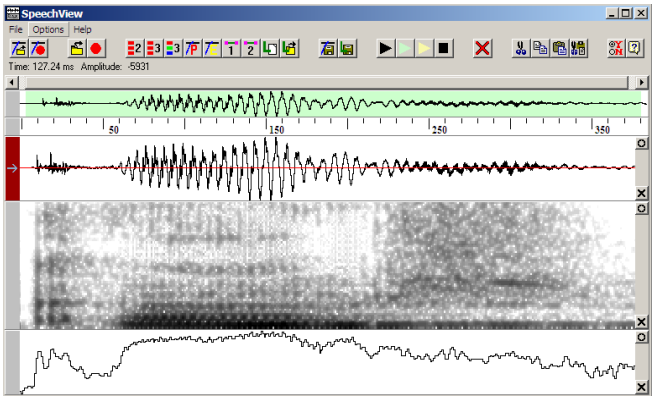
Sesli ifadelerin kaydedilmesinden sonraki ilk aşama yüksek frekans bileşenlerini transfer fonksiyonu

$$H(z) = 1 - cz^{-1} \tag{4.1}$$

şeklinde olan birinci derece bir filtreden geçirerek süzmektir. Eş.4.1'de c iyileştirme katsayısı 0.96 olarak alınmıştır.

4.1.1 Kesimleme

Sesli ifade tanımada yürütülen en zor işlemlerden biri sözcüklerin kesimlenerek fonem sınırlarının bulunmasıdır. Ünlüler ses telleri titreştirilerek oluşturulan ve görece uzun süreli seslerdir. Sesli ifadenin dalga biçimi ve spektrumu incelendiğinde ünlü fonemlerin belirgin sınırlarının olduğu görülür. Ünsüz fonemler için ise aynı şeyi söylenemeyeceğinden sesli ifade içinde yerlerini belirlemek zorlaşır. Fonem sınırlarını belirlemede esli ifadenin dalga biçimi ve spektrumundan yararlanılır ve sıfırı geçme oranı (zero crossing rate) ile enerji fonksiyonları kullanılır. Kesimleme için Matlab ve SpeechView [29] programları (Bkz. Şekil 4.2) kullanılmıştır.



Şekil 4.2 Kesimlemenin yapıldığı *SpeechView* programında "kar" sözcüğünün dalga biçimi, spektrogramı ve enerjisi.

Sıfırı Geçme Oranı

Bir sinyalin sıfırdan geçiş sayısı sıfırı geçme oranı olarak tanımlanır. Sesli ifade kayıtlarında, sesli ifadenin yer almadığı kısımlarda bu oran gürültü nedeniyle fazla olacaktır. Bu özellik sesli ifadelerin sınırlarını belirlemede kullanılır. Sıfırı geçme sayısı, *n.* örnek için daha önceki *N* örnek kullanılarak aşağıdaki gibi hesaplanabilir:

$$Z(n) = \sum_m \left| \text{sgn}\left[x(m)\right] - \text{sgn}\left[x(m-1)\right] \right| w(n-m) . \qquad (4.2)$$

Eş.4.2'de $x(m)$ ses sinyali, *sgn* işaret (signum) fonksiyonudur; $w(n)$ ise

$$w(n) = \begin{cases} \dfrac{1}{2N}, 0 \le n \le N-1 \\ 0, \text{diğer durumlar} \end{cases} \qquad (4.3)$$

şeklinde tanımlanabilir.

Enerji

Enerji sesin şiddetine bağımlı bir parametredir. Sesli ifadenin sadece gürültü içeren kesimlerinde sesin şiddeti çok düşüktür. Sesli ifadenin kısa bir süre için hesaplanan enerjisi, sinyaldeki değişimleri belirgin biçimde ortaya çıkarır [2]. Kısa süreli enerji *n.* örnek için aşağıdaki gibi hesaplanır:

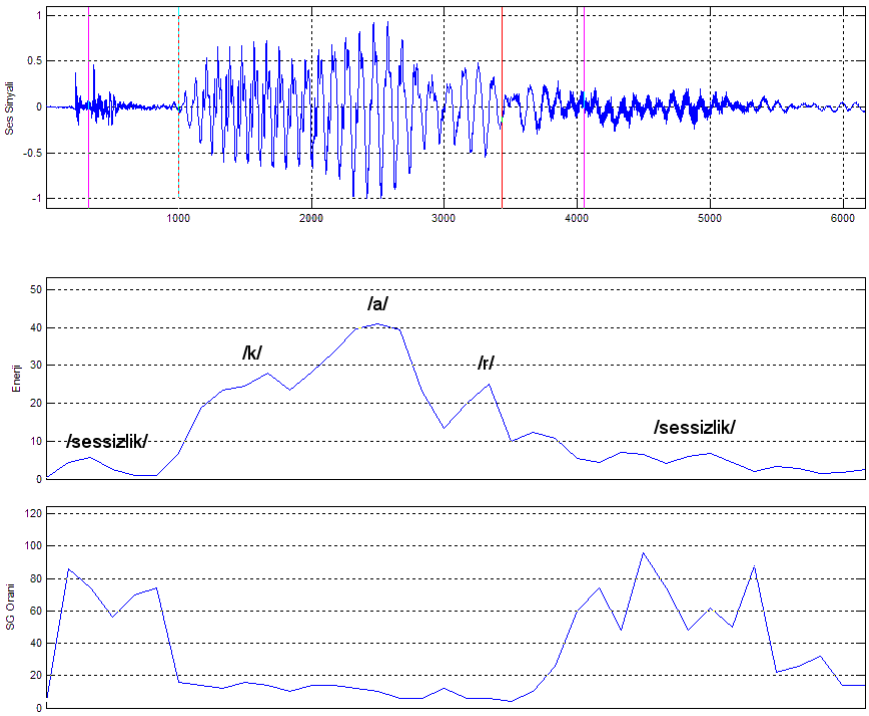
$$E(n) = \sum_m \left[x(m) w(n-m) \right]^2 . \qquad (4.4)$$

Bu işlem yapılırken pencereleme fonksiyonları kullanarak sonuçların daha belirgin olması da mümkündür. Pencereleme fonksiyonlarına Bölüm 4.1.3'te değinilecektir.

Enerji ve sıfırı geçme oranı kullanılarak sözcüklerin kesimlenmesi şu ilkelere dayalı olarak yapılmıştır [4]:

- Sözcük sınırlarında enerji değerleri çok düşüktür.
- Ünlülerin bulunduğu kesimlerde enerji değerleri yüksek, sıfırı geçme oranları düşüktür.
- Ünsüzlerin bulunduğu yerlerde enerji değerleri düşük, sıfırı geçme oranları yüksektir.

Bu ilkelere dayalı olarak “kar” sözcüğü için yapılan kesimleme Şekil 4.3’te gösterilmiştir.



Şekil 4.3 “kar” sözcüğünün dalga biçimi ve Matlab ile hesaplanan enerji ve sıfırı geçme (SG) oranı değerleri

4.1.2 Time Warping

Fonem kesimleme işlemi sonucu, farklı fonemlere ait örneklerin zaman ekseninde farklı uzunluklara sahip olması normaldir. Ancak seslendirmenin doğasından kaynaklanan nedenlerle, aynı foneme ait örnekler arasında da küçük farklılıklar gözlenir. Bu sorunu çözmek için *time warping* yönteminden faydalanılabilir. Bu yöntemde sesli ifadelerin süreleri sıkıştırılarak ya da genişletilerek bir referans şablonu ile karşılaştırılır [30].

Time warping işleminde amaç *A* ve *B*'ye ait örneklerin çakıştırılmasıdır. Karşılaştırılacak iki örneğin zaman eksenindeki değerleri,

$$A = [a_1, \ldots, a_i, \ldots, a_m]$$
$$B = [b_1, \cdots, b_j, \ldots, b_n] \quad (4.5)$$

bu örneklerin kesişen nokta çiftleri ise,

$$c(k) = [c(1), \ldots, c(k), \ldots, c(K)] \quad (4.6)$$

şeklinde yazılabilir. Her bir $c(k)$ için maliyet (cost) fonksiyonu,

$$d[c(k)] = \Delta[i(k), j(k)] \quad (4.7)$$

eşitliği ile bulunur. Eş.4.7'de d, örnek çiftler arasındaki zıtlıktır. *Warping* fonksiyonu tüm maliyet fonksiyonları enküçültülerek hesaplanır:

$$D(C) = \sum_{k=1}^{K} d[c(k)]. \quad (4.8)$$

Warping fonksiyonlarının hesaplanması (1,1) noktası ile (*M,N*) noktası arasında en az maliyetli yolu bulmayı gerektirir. Bu tür bir problem, devingen (dynamic) programlama problemidir. En düşük maliyetli yol,

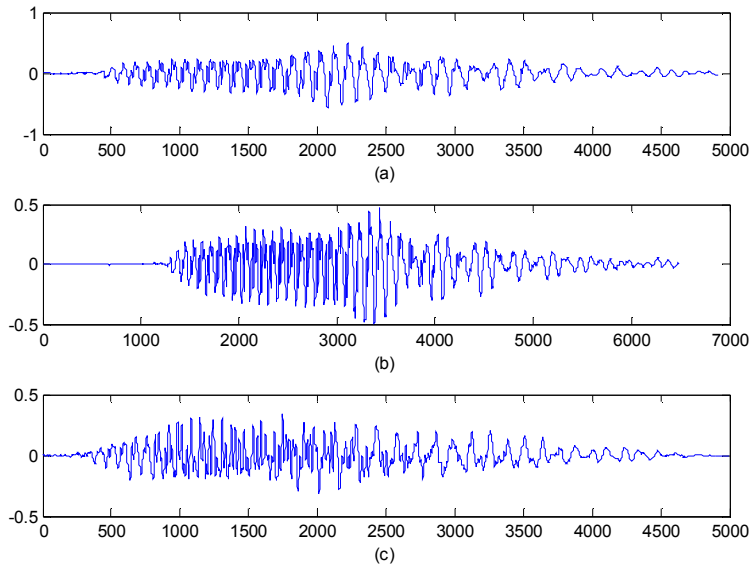
$$D(C_k) = d[c(k)] + \underset{\text{geçerli } c(k-1)}{MIN} [D(C_{k-1})] \quad (4.9)$$

eşitliği ile ifade edilir. Geçerli olan $c(k-1)$, tüm $c(k)$'lar üzerinde izlenebilir en kısa yoldur. Her nokta için üç olasılık vardır. Eğer $c(k) = (i, j)$ ise

$(i, j-1),\ (i-1, j),\ (i-1, j-1)$ olasılıkları bulunmaktadır. K adımdan oluşan çaktırma işleminde $w(t)$ ağırlık fonksiyonu olduğunda toplam maliyet,

$$L = \sum_{k=1}^{K} w(k) \tag{4.10}$$

eşitliği ile bulunur. Şekil 4.4'te farklı uzunluklara sahip iki /a/ foneminin *time warping* yöntemi ile eşit uzunluklara çekildiği gözlenebilir. Şekilde 4.4(b)'deki dalga biçimine sahip bir /a2/ fonemi, Şekilde 4.4(a)'daki bir /a1/ fonemi ile aynı uzunluğa çekilmiş ve /a2/ foneminin *time warping* sonucu dalga biçimi Şekilde 4.4(c)'de gösterilmiştir.

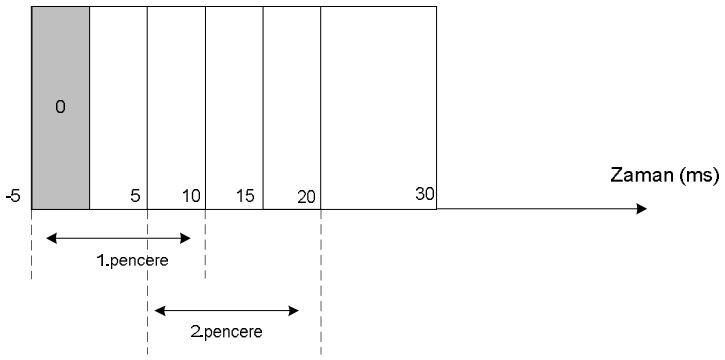


Şekil 4.4 *Time warping* yönteminin gösterimi

4.1.3 Pencereleme

Önişleme sürecinin sonraki aşamalarında ses sinyalinin *fourier* dönüşümü alınacağından, sesli ifadenin anlamlı uzunlukta parçalara bölünmesi gereklidir. bir seferde işleme alınacak veri kümesi, sesli ifadenin bir penceresi olarak alınır. Pencere uzunluğu ya da süresi; sıklık çözünürlüğünü kaybetmeyecek kadar uzun, pencerenin zamana göre durağanlığını kaybetmeyecek kadar kısa seçilmelidir.

Pencereleme yapılırken genellikle birbirinden kopuk veri kümeleri yerine, Şekil 4.5'teki gibi, üst üste binen kümeler üzerinde çalışılır [31]. Böylece pencerenin taşıyacağı bilgi miktarı daha fazla olacaktır.



Şekil 4.5 Üst üste binen pencereler

Öte yandan pencereleri dikdörtgenler halinde ayırmak, ses sinyalinin pencere dışında sıfır düzeyinde olmasına yol açar. Bu sorunu hafifletmek için ses sinyali uygun bir fonksiyonla evrişime (convolution) tabi tutulur:

$$y[k] = x[k] \otimes w[k] = \sum_i w[i] \cdot x[k-i]. \qquad (4.11)$$

Eş.4.11'deki $w[k]$ pencereleme fonksiyonu olarak adlandırılır. Bu çalışma kapsamında *Hamming* pencereleme fonksiyonu kullanılmıştır. Bu fonksiyon N pencere uzunluğu olmak üzere aşağıdaki gibi tanımlanır:

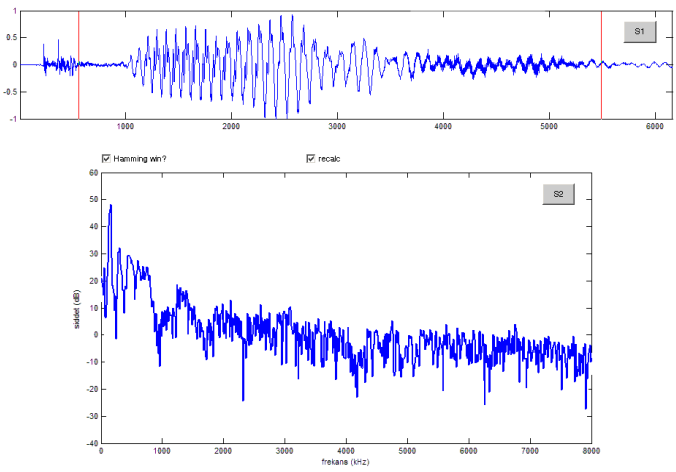
$$w[n] = 0.54 - 0.46\left(\frac{2\pi n}{N-1}\right),\ 0 \le n \le N-1 \qquad (4.12)$$

Türkçe fonem sınıflandırma sürecinde ünlü ve ünsüz fonemler birlikte sınıflandırılacağından pencere boyutu 15ms olarak belirlenmiş ve sesli ifade üzerindeki bir sonraki pencere, bir öncekinden 10ms uzaklıkta olmak üzere kaydırma uygulanmıştır. Bu şekilde, örneğin, 20ms uzunluğunda kesimlenen ünsüz bir fonem 2 pencere, 70ms uzunluğunda kesimlenen ünlü bir fonem 7 pencereden oluşur.

4.1.4 Sesli İfadelerin Modellenmesi

Sesli ifadeler için öznitelik vektörleri oluşturmakta en sık kullanılan yöntem spektral çözümlemedir. Sesli ifadelerin parametrik modellenmesinde çeşitli spektral çözümleme algoritmaları kullanılmaktadır. Modellemede ilk aşama sesli ifadenin spektrumunu çıkartmaktır (Bkz. Şekil 4.6). Bunun için kesikli *fourier* dönüşümü (discrete fourier transform – DFT) kullanılabilir. *N* adet örnek içeren bir pencere için alınan DFT, Eş. 4.13'teki gibi hesaplanır.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}} \quad k = 0,1,\ldots,N-1. \tag{4.13}$$

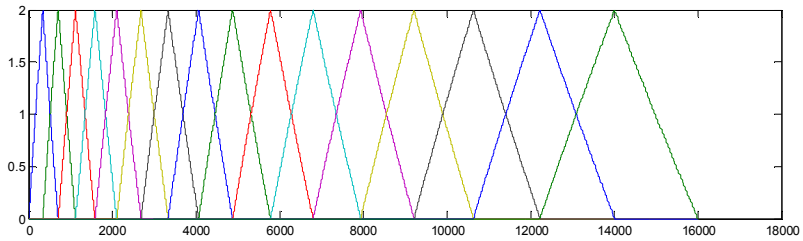


Şekil 4.6 "kar" sesli ifade sinyalinin dalga biçimi ve spektrumu (altta)

Filtre Dizisi Yöntemi

Yapısında insan kulağının çalışma ilkesi temel alınmıştır. Buna göre, sesli ifadeye ait sıklık bölgeleri belirli sayıda kanala ayrılır. Bu kanallar, birer bant geçişli filtredir. Her sıklık bölgesine ait değerler kümesinin bir merkez sıklık değeri vardır ve filtrelerin band genişlikleri, merkez sıklık değerinin %10 ila %20'si kadardır [4]. Filtrelerin merkez sıklıkları, akustik sıklık değerinden (f), algılanan sıklık değerine dönüşüm fonksiyonu ile bulunur. *Mel* sıklığına dönüşüm Eş 2.1'de verilmiştir. Filtre seçimi yapılırken kıstas, sesli ifadenin en uygun biçimde bir vektör tarafından

temsil edilmesidir. Bu yönden *Mel* filtre dizisinin iyi sonuçlar verildiği gözlenmiştir [32]. Bu filtre dizisinde, Şekil 4.7'de gösterildiği gibi, üçgen şeklinde filtreler vardır.



Şekil 4.7 Mel filtre bankası

Sesli ifadeye ait veriler *Mel* ölçeğine dönüştürüldükten sonra filtre dizisinden geçirilirler.

Cepstrum

Cepstrum, "spectrum" sözcüğünün ilk dört harfinin ters çevrilmesi ile türetilmiş olup, matematiksel olarak bir sinyalin fourier dönüşümünün (örneğin, DFT) logaritmasının, ters fourier (inverse discrete fourier transform – IDFT) dönüşümüdür. Bir tahrik sinyalinin ve o sinyale ait yansımaların üst üste bindirilmesi ile oluşan sinyallerde, bu, yansımaların bulunduğu konumlardaki tepe noktalarını gösterir [4]. Sesli ifadeler, bir tahrik titreşiminin ses yolu ile rezonansa uğraması ve duraklatılması ile elde edildiğinden, c*epstrum* fonksiyonu sesli ifadelerin parametrik gösterimi için kullanılabilir.

Bir sesli ifade sinyali, $g(n)$ tahrik sinyalini, $v(n)$ ise ses yolunun vuru tepkisini ifade etmek üzere,

$$x(n) = g(n) \otimes v(n) \tag{4.14}$$

şeklinde yazılabilir. Sesli ifadenin *fourier* dönüşümü alındığında,

$$X(f) = G(f) \cdot V(f), \tag{4.15}$$

bu dönüşümün logaritması alındığında ise,

$$\log(X(f)) = \log(G(f)) + \log(V(f)) \tag{4.16}$$

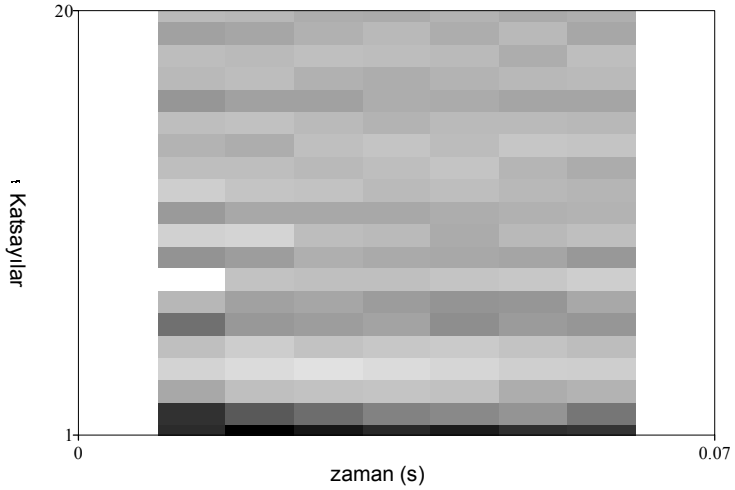
sonucunu verir. Yani logaritma uzayında tahrik sinyali ile ses yolunun vuru tepkisinin toplamı, sesli ifadeyi verir. Bu toplamdan öznitelik çıkarımı yoluna gidilebilir. *Cepstrum* katsayıları bulunurken Eş.4.16'daki toplamın ters *fourier* dönüşümü alınmalıdır. Ancak *log spektrum* simetrik ve gerçel bir fonksiyon olduğundan, *ceptrum* katsayıları kosinüs dönüşümü ile bulunabilir:

$$C(n) = \frac{2}{N}\sum_{k=0}^{N-1} X(k)\cos\left(\frac{2\pi}{N}kn\right). \tag{4.17}$$

Bu şekilde belirlenen katsayılara *fourier* dönüşümünden türetilmiş *cepstral* katsayılar denilir. *Fourier* dönüşümü yerine mel filtre dizisi kullanılırsa elde edilen katsayılara mel filtresi cepstrum katsayıları (MFCC) denilir [33]:

$$C_i = \sqrt{\frac{2}{N}}\sum_{k=0}^{N-1} o_k \cos\left(\frac{\pi i}{N}(k-0.5)\right) \tag{4.18}$$

Eş 4.18'de N filtre sayısını, o_k ise $k.$ filtre dizisinin çıkışını göstermektedir. Hesaplanan *cepstrum* parametreleri dizisindeki ilk değerler ses yoluna ait bilgileri, daha sonraki değerler, tahrik titreşimlerinin periyoduna ait bilgileri taşır [34]. Sesli ifadeler *cepstrum* katsayıları ile temsil edilirken genellikle ilk 20 parametre göz önüne alınır. Şekil 4.8'de yaklaşık 70ms uzunluğundaki bir /a/ fonemin önişleme aşaması sonrası 20 adet MFCC ile parametrik gösterimi verilmiştir. Bu şekilde 16KHz sıklığında 1120 örnek içeren fonem, 7 adet 20 boyutlu vektörle temsil edilmiş olmaktadır.



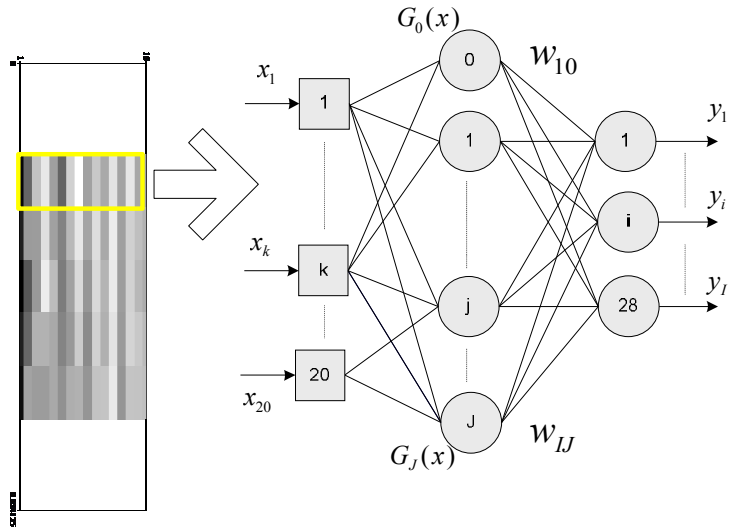
Şekil 4.8 70ms uzunluğundaki bir /a/ fonemi için Mel filtresi cepstrum katsayıları

4.2 Sınıflandırma

4.2.1 RTFA Tasarımı

Fonemler için özellik vektörleri çıkarıldıktan sonra RTFA kullanılarak sınıflandırma aşamasına geçilebilir. RTFA'nın donanım mimarisi tasarlama aşamasına geçilmeden önce, Matlab kullanılarak uygun benzetim sonuçlarını verecek model tasarlanmıştır.

Her pencere için 20 adet MFCC kullanıldığından RTFA'nın giriş uzayı 20 boyutludur. Katsayılar -1 ile 1 arasında normalize edildikten sonra sinir ağına aktarılmıştır. Çıkış katmanı 28 adet foneme karşılık 28 nöron içerir. Eğitim için kullanılan sette 240 fonem ve bunlara karşılık 778 pencere vardır. Örnek bir örüntü vektörünün içeriği Çizelge 4.1'de verilmiştir. Sınıflandırma pencere tabanlı yapılacağından, giriş uzayına uygulanan pencerenin hangi foneme ait olduğunun bulunması amaçlanmıştır.



Şekil 4.9 RTFA ile fonem sınıflandırma

4.2.2 Benzetim Sonuçları

Gizli katmanda kullanılacak radyal tabanlı fonksiyon sayısı, aynı zamanda 20-boyutlu 778 örüntü vektörünün ayrılacağı küme sayısıdır ve bu sayı deneme-yanılma yöntemi kullanılarak belirlenmiştir. Kullanılan Gauss fonksiyonu sayısına göre başarı yüzdeleri Çizelge 4-1'de özetlenmiştir.

Çizelge 4.1 Matlab ortamında başarı yüzdeleri

RTF Sayısı	Başarı Oranı (%)	
	Eğitim Seti	Test Seti
32	65	61
64	73	70
96	82	79
128	85	82
160	87	86
192	88	87

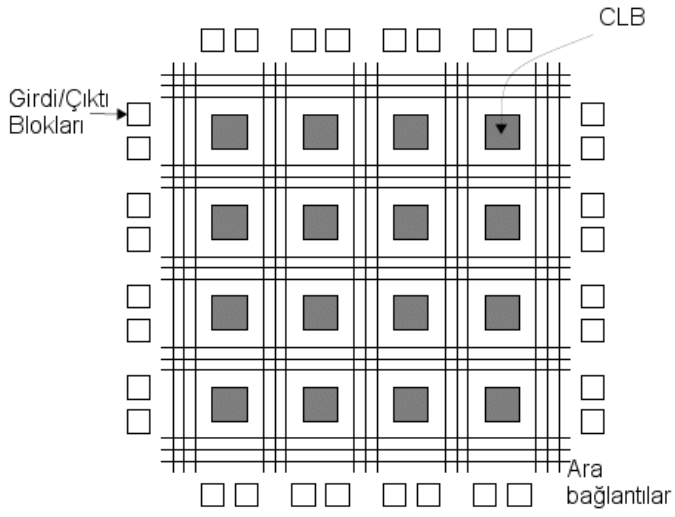
BÖLÜM 5

5 YAPAY SİNİR AĞLARININ DONANIM UYGULAMALARI

YSA'nın donanım tasarımlarında elektronik ve optik [36–40] yaklaşımlar önerilmiştir. Elektronikte analog, sayısal ve karma sinyal (mixed-signal) YSA tasarımlarının [41–44] zengin bir geçmişi vardır. YSA'nın FPGA (Field Programmable Gate Array) tasarımları ise görece yeni bir uygulama alanıdır. Bu bölümde FPGA mimarisi ve YSA'nın FPGA uygulamaları üzerine özet bilgiler verildikten sonra, tasarlanan donanım mimarisi üzerinde durulacaktır.

5.1 FPGA Yapısı

FPGA, "alan programlanabilir kapı dizileri" olarak Türkçe'ye çevrilebilir. FPGA, ASIC (Application Specific Integrated Circuit) çalışma hızına yakın performans sergilerken, yeniden programlanabilir olması nedeniyle tasarımda esneklik sunmaktadır. Geleneksel olarak FPGA, tasarlanan sistemin ASIC üretiminden önce test amaçlı kullanılırdı. Ancak CMOS teknolojisinin $0.18\,\mu$ ve altı sürece geçmesiyle birlikte çok yüksek yoğunluklu FPGA yapımı olanaklı olmuş ve ASIC teknolojisine alternatif bir uygulama platformu haline dönüşüştür.

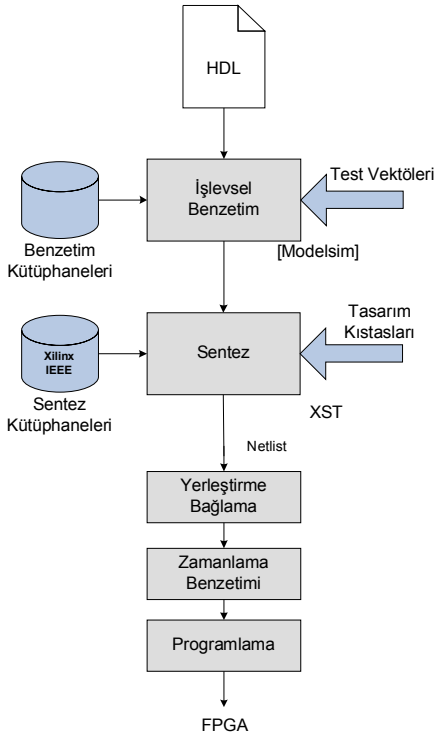


Şekil 5.1 FPGA genel yapısı

Şekil 5.1'de bir FPGA'nın genel yapısı görülmektedir. Her FPGA'da bulunan üç temel birim vardır. Bunlar; girdi/çıktı blokları, düzenlenebilir mantık blokları (configurable logic block – CLB) ve programlanabilir ara bağlantılardır. Girdi/çıktı blokları gerekli sinyallerin cihaz içine ve dışına aktarımı için kullanılır. Düzenlenebilir mantık blokları kapı ve yazmaçları içeren donanım bloklarıdır. Düzenlenebilir mantık bloklarının mimarisi ve ara bağlantı teknolojisi FPGA modeline göre değişiklikler arz etmektedir.

5.1.1 FPGA Tasarım Akışı

Donanım tasarımda izlenen FPGA tasarım akışı Şekil 5.2'de verilmiştir [45].



Şekil 5.2 FPGA tasarım akışı

RTFA'nın donanım mimarisi VHDL donanım tanımlama dili (hardware description language – HDL) kullanılarak oluşturulmuştur. Benzetim kütüphaneleri ve fonem test girdileri kullanılarak, *Modelsim* üzerinde uygun benzetim sonuçları alınamaya çalışılmıştır. Tasarımda hedef platform Xilinx Virtex-II FPGA ailesi olmuştur. Sentezleme sonucu oluşan *netlist* dosyası FPGA içinde yapılması gerekli bağlantıları tanımlar. Bu şekilde yerleştirme ve bağlama yapıldıktan sonra kapı ve

yazmaçlarda oluşan gecikmelerin de dahil edildiği zamanlama benzetimi yapılmış ve uygun sonuçlar alındıktan sonra tasarım FPGA içerisine gömülmüştür.

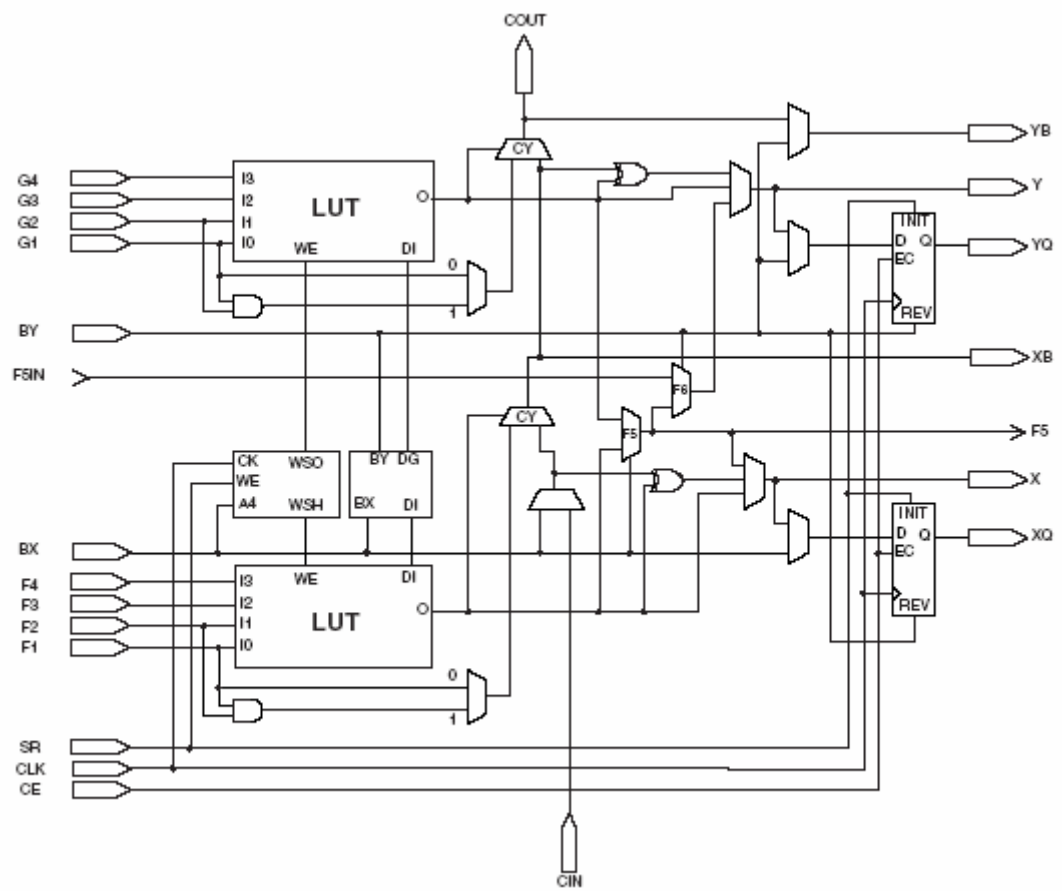
5.1.2 Xilinx Virtex-II Platformu

YSA, yoğun matris-vektör çarpma işlemleri ve doğrusal olmayan fonksiyonlar içeren matematiksel yapılardır. Bu yapıların FPGA üzerine eşlenebilmesi yüksek yoğunluklu mantık blokları gerekmektedir. Xilinx Virtex-II platformu 0.18 μ 5-metal katman süreci ile üretilmiş, özgül 18-bit çarpma blokları içeren yüksek yoğunluklu bir FPGA ailesidir. Xilinx Virtex-II platformunun genel özellikleri Çizelge 5.1'de özetlenmiştir.

Çizelge 5.1 Xilinx Virtex-II ailesi

FPGA	**Sistem Kapısı**	**CLB Dilimi**	**Çarpma Blokları**	**18Kbit BRAM**	**Girdi/Çıktı**
XC2V40	40K	256	4	4	88
XC2V80	80K	512	8	8	120
XC2V250	250K	1,536	24	24	200
XC2V500	500K	3,072	32	32	264
XC2V1000	1M	5,120	40	40	432
XC2V1500	1.5M	7,680	48	48	528
XC2V2000	2M	10,752	56	56	624
XC2V3000	3M	14,336	96	96	720
XC2V4000	4M	23,040	120	120	912
XC2V6000	6M	33,792	144	144	1,104
XC2V8000	8M	46,592	168	168	1,108

Virtex-II platformunda bir CLB dört dilimden oluşur ve her dilim, Şekil 5.3'te görüleceği gibi, iki adet 4 girişli LUT, aritmetik ve mantık kapıları, yol seçiciler (multiplexer – MUX), özgül *carry-lookahead* toplayıcılar ve 2 yazboz (flip-flop) içerir. LUT, *boolean* fonksiyonlarının gerçeklenmesinde, yazbozlar saklamada, diğer yapılar aritmetik ve mantık işlemlerinde kullanılır. Tez kapsamında yapılan çalışmalar için XC2V6000 [46] modeli kullanılmıştır.

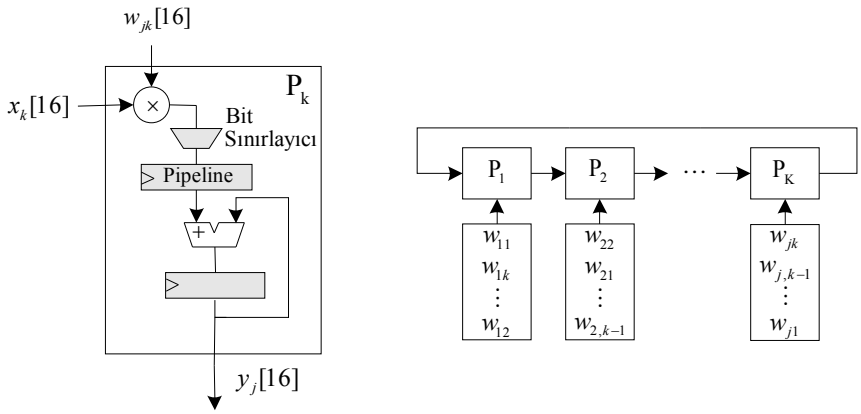


Şekil 5.3 Xilinx Virtex-II dilimi

Çarpma Blokları

Özgül çarpma blokları 18-bit x 18-bit sabit noktalı (fixed-point) çarpma yapabilen ve istenildiğinde *pipeline* kullanılabilen bloklardır ve YSA gibi hızlı ve yoğun çarpma işlemi gerektiren modellerde kullanılmaya elverişlidir.

Özgül çarpma bloklarının, Virtex-II platformu üzerinde gerçeklenebilecek en hızlı sabit-noktalı çarpıcı olduğu yapılan bir dizi ön çalışma ile doğrulanmıştır. Buna göre Şekil 5.4(a)'de çizimi verilen çarpıcı-toplayıcı devre (multiply and accumulate circuit – MAC) kullanılarak, Şekil 5.4(b)'de verilen sistolik diziler oluşturulmuş. Bu şekilde 16-bitlik sabit noktalı sayılar için vektör-matris çarpımı yapan modelin çarpma bloğu Virtex-II'nin özgül çarpma bloğunun yanısıra yaygın olarak kullanılan hızlı çarpma algoritmalarından *Booth–Wallace* [47], *Booth–4:2* [48] ve *Baugh–Wooley* [49] ile test edilmiştir. 9 adet sistolik dizi ile XC2V6000 FPGA modeli üzeride sentez sonuçları Çizelge 5.2'de özetlenmiştir.



Şekil 5.4 Sistolik vektör-matris çapma devresi

Çizelge 5.2 Sistolik vektör-matris çapma devresi için hız testi

Algoritma	**Saat Sıklığı (MHz)**
Özgül çarpma bloğu	145
Booth – 4:2	117
Baugh – Wooley	104
Booth – Wallace	95

Carry-Lookahead Toplayıcı

Carry-lookahead toplamanın temel ilkesi tüm eldeleri paralel olarak oluşturarak yayılım (propagation) süresini en aza indirmektir.

RAM

Virtex-II platformu iki tür RAM kullanır. Dağılmış (distributed) RAM, düzenlenebilir mantık bloklarına dağılmış ve esnek yapıda saklama birimleridir ancak kapasitesi sınırlıdır. *Block Select RAM* (BSRAM) ise esnek olmamasına rağmen hızlı ve yoğundur. Daha da önemlisi FPGA üzerine gömülü olarak geldiğinden, kullanıldığında FPGA üzerinde ek yer kaplamamaktadır. BSRAM birimleri Virtex-II üzerinde 18Kbit'lik bloklar şeklinde bulunurlar.

5.2 FPGA Uygulamalarında Kıstaslar

FPGA tabanlı YSA aşağıdaki kıstaslara göre değerlendirilebilirler [50]:

- Öğrenme algoritması gerçeklemesi
- Sinyal gösterimi
- Yürütüm süresinde yeniden düzenleme

5.2.1 Öğrenme Algoritması

Öğrenme algoritması FPGA üzerinde gerçeklenen sistemler *on-chip learning* özelliği taşırlar. Geri-yayılım öğrenme [19], FPGA üzerinde en çok tasarlanmış öğrenme algoritmasıdır [51–55]. İlk olarak Eldredge [56] Xilinx XC3090 FPGA modeli ile geri-yayılım algoritmasını gerçeklemiştir. Eldredge'in *RRANN* (run-time reconfiguration network) mimarisi bulanık kümelerin merkezlerini tahmin etmede %88'lik bir başarı oranı yakalamıştır. Ferucci ve Martin [57,58] *Adaptive Connectionist Model Emulator* (ACME) olarak adlandırdıkları mimariyi Xilinx XC4010 platformunda gerçeklemiş ve 3 girdi, 1 çıktı ve 3 gizli nöronu olan bir çok katmanlı sinir ağına XOR problemini öğretmeyi başarmıştır. Skrberk [59], ECX platformu adını verdiği modelde hem çok katmanlı sinir ağı hem de RTFA tasarımları yapmış ve bu tasarımları karakter ve sonar sinyali tanıma problemlerinde test etmiştir. Yang ve Paindavoive'in *MEMEC* kartı üzerinde çalışan RTFA [59], gerçek zamanlı yüz tanıma problemi için kullanılmıştır. Çizelge 5.3'te FPGA üzerinde gerçeklenen YSA uygulamalarının bir listesi verilmiştir [50].

Çizelge 5.3 Başlıca FPGA-tabanlı YSA uygulamaları

Mimari	Sinyal Gösterimi	YSA Modeli	CUPS	RTR
RRANN – 1994	Sabit noktalı (5–40 bit)	Geri-yayılım	722K	var
REMAP– 1995	Sabit noktalı (2–8 bit)	Sparse Dist. Memory	N/A	yok
ACME – 1994	Sabit noktalı (8-bit)	Geri-yayılım	1640	yok
ECX – 1999	Sabit noktalı (8–16 bit)	Geri-yayılım/RBF	3.5M	yok
FAST – 2000	Sabit noktalı (8-bit)	ART	4M	yok

Çizelge 5.3'te performans ölçüsü olan CUPS (connection updates per second), geri-yayılım algoritmasında öğrenme fazının ne kadar hızlı yapıldığının bir göstergesidir ve diğer YSA modellerinde bu ölçüt kullanılmaz.

5.2.2 Sinyal Gösterimi

YSA'nın donanım tasarımlarındaki performansı sinyal gösterimlerindeki duyarlılığa (precision) bağımlıdır. Aritmetik duyarlılığın sınırlanması nöronlar arasındaki bağlantıların nicemleme (quantization) hatalarını arttırır. Çok fazla niceleme hatası neticesinde öğrenme algoritması yakınsama özelliğini kaybedebilir. Öte yandan duyarlılık arttıkça kaplanan silikon alan da artacaktır. Duyarlılık/alan ödünleşimi (trade-off) donanım tasarımında önemli bir kıstastır. YSA'nın FPGA uygulamalarında görülen iki temel sinyal gösterimi aşağıda verilmiştir.

Kayan Noktalı Gösterim

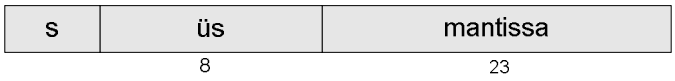
Bir kayan noktalı sayı (floating point number) $\pm d.dd\ldots d \times \beta^e$ şeklinde gösterilir. $\pm d_0.d_1 d_2 \ldots d_{p-1} \times \beta^e$,

$$\pm\left(d_0 + d_1\beta^{-1} + \ldots + d_{p-1} \times \beta^{-(p-1)}\right)\beta^e \qquad (5.1)$$

sayısı temsil eder. Eş 5.1'de β üs (base), e üs (exponent) ve p duyarlık (precision) olarak adlandırılır. IEEE 754-1985 standardına göre [62] tek duyarlı kayan noktalı sayı gösteriminde $\beta = 2,\ p = 24,\ m = 8$ ve $e = k - 127$ alınır. Üs'te bulunan bit sayısı m, üs'teki sayının işaretsiz (unsigned) tamsayı değeri k ile gösterilir. IEEE 754-1985 standardına göre kayan noktalı sayı düzeni Şekil 5.5'te verilmiştir. Buna göre Eş.5.1 kullanılarak kayan noktalı sayının değeri,

$$v = -1^s(1.f)2^e \qquad (5.2)$$

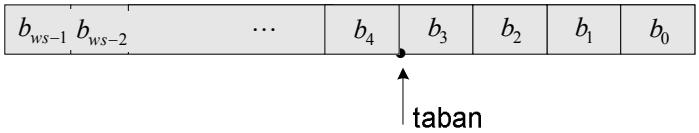
ile bulunur. Eş.5.2'de f *mantissa* olarak adlandırılır ve $p - 1 = 23$ bit uzunluğundadur. İşaret biti s ile gösterilir.



Şekil 5.5 IEEE 754-1985 standardına göre tek duyarlı kayan noktalı sayı düzeni

Sabit Noktalı Sayılar

Sabit noktalı sayı düzeni Şekil 5.6'da verilmiştir. Buna göre bir sabit noktalı sayı ki bölümden oluşur. Şekil 5.6'da b_{ws-1} ile b_4 arasındaki bölüm tamsayıları, b_3 ile b_0 arasındaki bölüm kesirli sayıları göstermektedir.



Şekil 5.6 Sabit noktalı sayı düzeni

Şekil 5.6'da gösterilen sabit noktalı sayıda üs (β) pozitif bir sayı ise değeri,

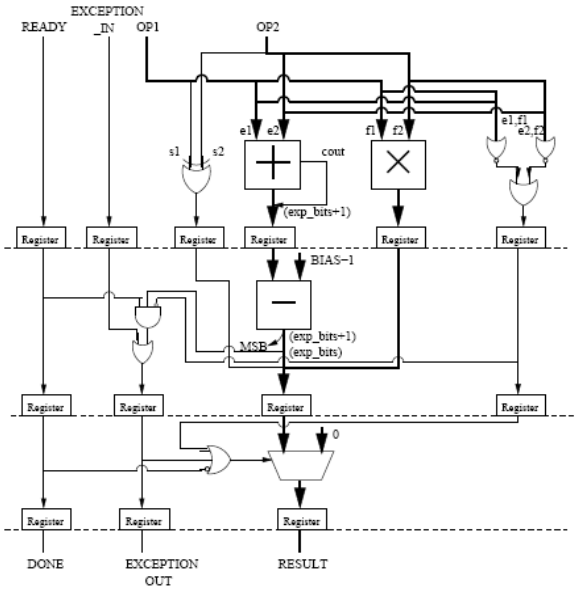
$$b_{ws-1}\beta^{ws-5} + \ldots + b_4 + b_3\beta^{-1} + b_2\beta^{-2} + b_1\beta^{-3} + b_0\beta^{-4} \qquad (5.3)$$

şeklinde hesaplanır.Üs 2 olarak alındığında genellikle 2'ye tümleyen gösterimi kullanılır.

Kayan noktalı sayıların geniş ve dinamik menzili ve değişken duyarlığı vardır. Öte yandan sabit noktalı sayılarda sınırlı bir menzili vardır ve duyarlık sayının gösteriminden bağımsızdır. Tamsayılar tasarlanan aritmetik devreler, sabit noktalı sayılarda da kullanılabilir. Kayan noktalı sayılarda ise özgül aritmetik devrelerin tasarlanması gereklidir. Şekil 5.7'de kayan noktalı sayılar için tasarlanan bir *pipeline* çarpma devresi gösterilmiştir [63].

YSA'nın FPGA uygulamalarında kayan noktalı sayılar kullanılmasının elverişli sonuçlar vermediği gözlenmiştir [63–64]. Li, Moussa ve Areibi'nin geri-yayılım algoritmasının FPGA uygulaması üzerine yaptığı testlerde 16-bit sabit noktalı sayı düzeni kullanılması ile oluşturulan tasarımın, 16-bit kayan noktalı düzeni kullanan

tasarımdan 12 kat hızlı çalıştığı ve 13 kat küçük silikon alan kapladığı ortaya gözlenmiştir [63]. Öte yandan Holt ve Baker [64], geri-yaylılım algoritması içi tolere edilebilir en küçük duyarlığın 16-bit olduğunu kanıtlamıştır.



Şekil 5.7 Kayan noktalı sayılar için *pipeline* çarpma devresi

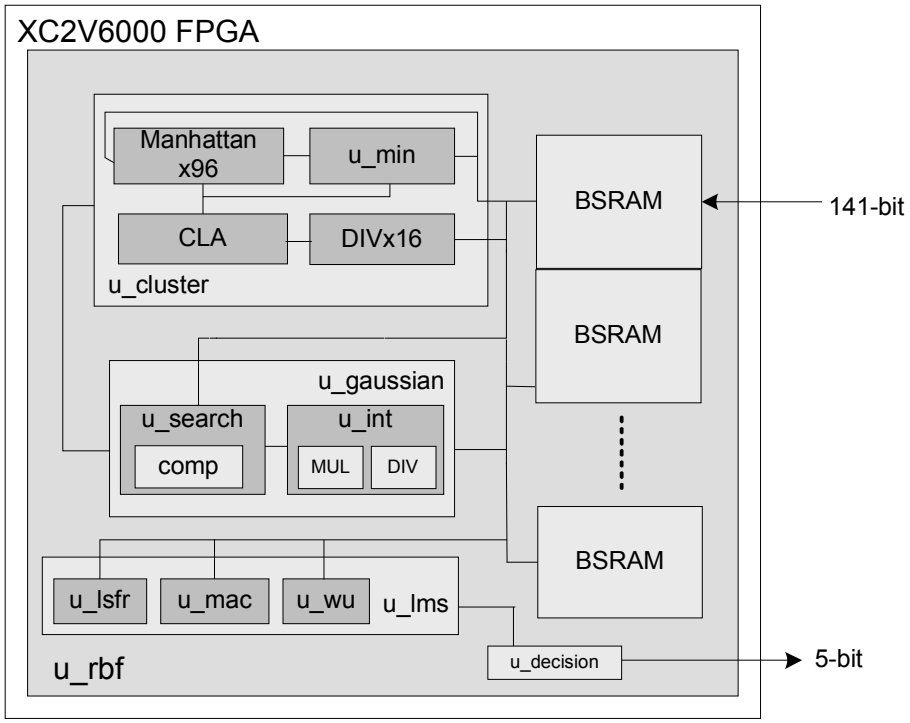
5.2.3 Yürütüm Sırasında Yeniden Düzenleme

Yürütüm sırasında yeniden düzenleme (run time reconfiguration – RTR) özelliği YSA algoritmaları gibi birden fazla aşamadan oluşan sistemlerde, tüm tasarımın bir seferde cihaza eşlenmesi yerine, farklı aşamaların farklı zamanlarda eşlenerek silikon alanı, güç ve işleme hızında eniyileştirmeyi hedefler. Ayrıca bu özellik sayesinde YSA topolojisi yürütüm sırasında dış dünyadan gelen tepkilere göre kendini uyarlayabilme özelliği de kazandığından sisteme esneklik sağlar. Dinamik RTR, her FPGA modelinde var olan bir özellik değildir.

BÖLÜM 6

6 RADYAL TABANLI FONKSİYON AĞININ DONANIM TASARIMI

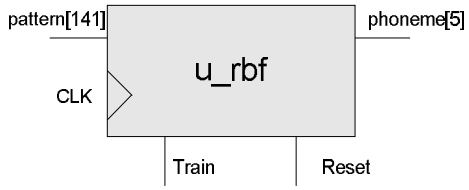
RTFA'nın donanım mimarisi VHDL ile tasarlanmış ve sentez sırasında standart IEEE kütüphanelerinin yanısıra *Xilinx Core* kütüphanesi de kullanılmıştır. Xilinx Virtex-II platformu üyesi XC2V6000 modeline eşlenen tasarım, dört temel birimden oluşmaktadır. Sistemin denetim birimi *u_rbf*, veri akışını düzenleyen ana birimdir ve diğer birimler, *u_rbf* biriminin denetiminde çalışır. Sistem, gelen öznitelik vektörlerini *u_cluster* birimi ile sayısı önceden belirli kümelere ayırır ve gerekli parametreleri *u_gauss* birimine gönderir. Burada radyal tabanlı fonksiyonların değerleri hesaplanır ve sonuç *u_lms* birimine aktarılır. En son olarak *u_decision*, *u_lms* birimine ait çıktıdan gelen pencerenin hangi foneme ait olduğu bulan basit bir karar devresidir. Sistemin hiyerarşik düzeni Şekil 6.1'de verilmiştir.



Şekil 6.1 Ana hatları ile donanım mimarisi

6.1 Denetim Birimi

Denetim biriminde kullanılan portlar aşağıda açıklanmıştır.



Şekil 6.2 *u_rbf* ana denetim birimi

- **pattern**. Örüntü vektörü, her elemanı 8-bit sabit noktalı sayı ile temsil edilen ve 16 mel cepstrum katsayısı içeren bir penceredir. Ayrıca her girdiye karşılık istenilen tepki için 13-bit gereklidir. İlk 8-bit, 28 boyutlu tepki vektörünün sıfırdan farklı bileşenini diğer 5-bit ise bileşenin konumunu gösterir. Örüntü vektörünün bir /a/ fonemi için içeriği Çizelge 6.1'de verilmiştir.
- **reset.** Yüksek olduğunda genel yeniden başlatma sinyali olarak kullanılır. Örüntüler işlenirken düşükte tutulmalıdır.
- **phoneme.** Girdi vektörüne ait fonem sonucunu belirten 5-bitlik yapıdır.
- **train.** Yüksek olduğunda girdi vektörü eğitim amacıyla kullanılır ve *phoneme* portundan çıktı alınmaz. Düşük olduğunda girdi vektörü test amaçlı kullanılır.
- **clk.** Eş zamanlı, artan kenar tetiklemeli sistem saatidir.

Tasarımda hesaplama yoğunluğunu azaltmak için 20 boyutlu mel cepstrum katsayıları yerine 16 boyutlu katsayılar ile çalışılmıştır. Öznitelik vektörünün her bir bileşeni de 8-bit sabit noktalı sayılarla temsil edildiğinden, her 15ms'lik pencere 128-bit veri içerir. Toplam 778 pencere içeren eğitim seti, yaklaşık 13 KB veriye karşılık gelmektedir.

Çizelge 6.1 Bir /a/ fonemine ait pencere için örüntü vektörünün içeriği

İstenilen Vektör	Girdi Vektörü	
	8-bit Sabit Nokta Gösterimi	Değeri
01111111	00100101	0.5649
00000	11111001	-0.1097
	00001010	0.1522
	11101110	-0.2949
	00000011	0.0408
	00001100	0.1842
	11111000	-0.1355
	00000000	-0.0143
	00000000	-0.0096
	00000000	-0.0134
	00000011	0.0375
	00000010	0.0229
	00000010	0.0204
	11111111	-0.0291
	00000000	-0.0116
	11111111	-0.0186

6.2 Kümeleme Birimi

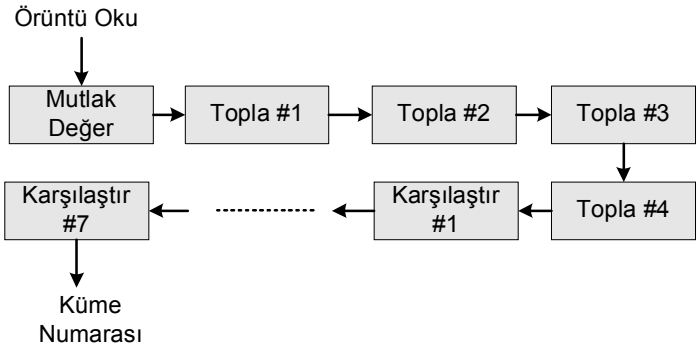
Bölüm 3.3.2'de anlatılan K-ortalama kümeleme algoritmasını gerçekleyen birimdir. Kümeleme biriminin donanıma uygulanmasında sistem kaynaklarını en çok tüketen hesaplama, uzaklık ölçütünün belirlenmesidir [65]. Bunun nedeni *Euclidean* uzaklığın karekök ve kare alma gibi işlemler içermesidir. Bu nedenle irim tasarlanırken, *Manhattan* uzaklığı kullanılmıştır. Manhattan metriğinde $(x_1, x_2, \ldots, x_n)$ noktası ile $(y_1, y_2, \ldots, y_n)$ noktası arasındaki uzaklık,

$$d_m = |x_1 - y_1| + |x_2 - y_2| + \ldots + |x_n - y_n| \tag{6.1}$$

şeklinde hesaplanır. Manhattan uzaklığı karekök ve kare alma işlemleri içermediğinden donanım uygulamalarında yaygın kullanılan bir uzaklık ölçütüdür.

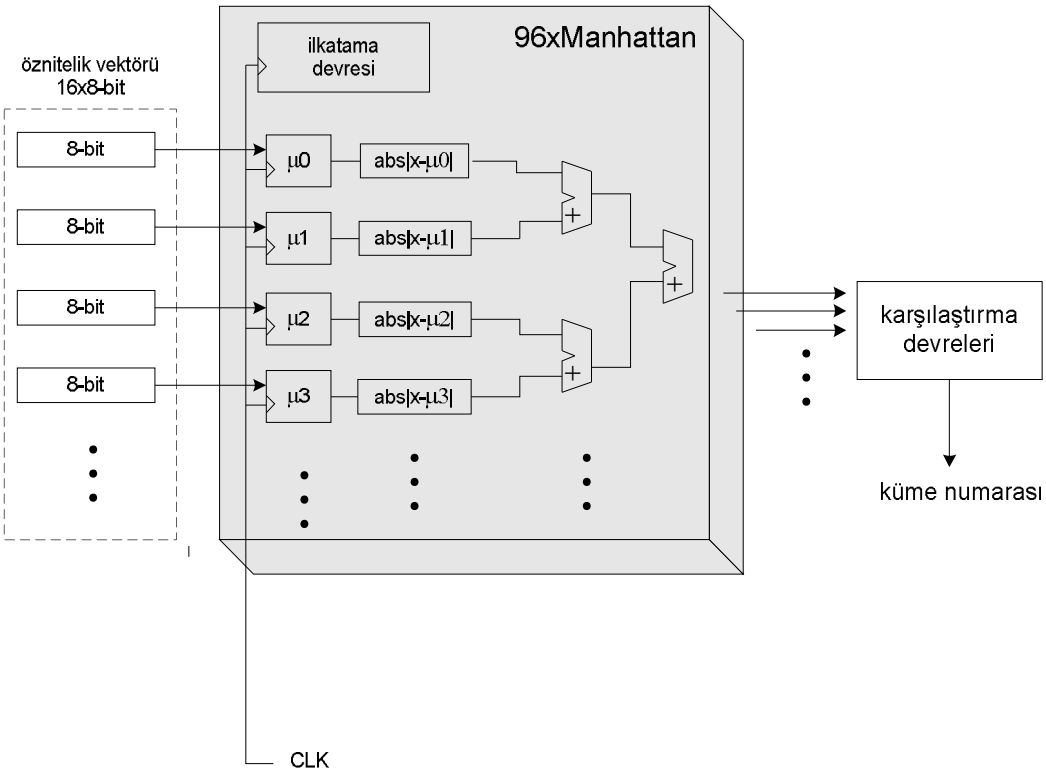
Kümeleme birimini tasarlamadan önce küme sayısını belirlemek gereklidir. Küme sayısı arttıkça gizli katmanda gerçeklenecek Gauss fonksiyonu ve dolayısıyla LMS algoritmasının işlem yoğunluğu artacaktır. Az miktarda küme sayısı ise yeterli başarı oranları vermemektedir. Çizelge 4.2'de görüleceği üzere küme sayısını 96 olarak seçmek uygundur. Buna göre 778 adet pencereyi 96 kümeye ayrılacaktır. Algoritma sonunda her pencerenin küme numarası belirlenmelidir.

128-bit genişliğe sahip BSRAM kullanıldığında her saat devrinde bir örüntüye yani pencereye erişmek mümkün olur. Tüm örüntü vektörleri 6 BSRAM'e sığacak boyuttadır. Küme numaralarını tutmak için ise 8-bit genişliğinde bir BSRAM kullanmak yeterli olacaktır. Tasarım 14 aşamalı bir *pipeline* kullanmaktadır. Bu aşamalar Şekil 6.3'te verilmiştir. Bu sayede her saat darbesinde bir pencerenin küme numarasını bulmak mümkündür. İlk aşamada öznitelik vektörü, ilkatanması yapılmış küme merkezlerinden çıkartılır.



Şekil 6.3 14 aşamalı *pipeline* kümeleme

İkinci aşamada tüm farkların mutlak değerleri alınır. Üçüncü ila altıncı aşamalar her küme için toplayıcı dizileri içerir. Toplama, Virtex-II'nin özgül toplayıcısı olan *carry-lookahead* yöntemi ile yapılmıştır. Altıncı aşama geçildikten sonra öznitelik vektörü ile 96 küme merkezi arasındaki uzaklık hesaplaması bitmiştir. Yedinci ila onüçüncü aşamalar en küçük uzaklığı seçmek için yapılan karşılaştırmalardan oluşur. Son adımda ise küme numarası ilgili BSRAM'e yazılır. Donanım mimarisi Şekil 6.4'te verilmiştir.



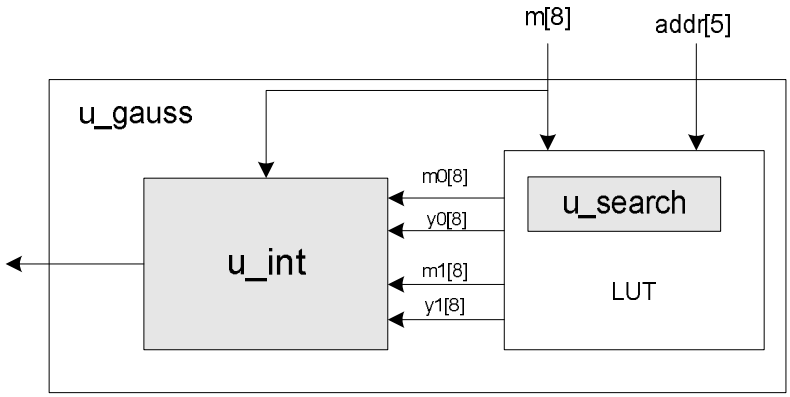
Şekil 6.4 *u_cluster* biriminin donanım mimarisi

Her özyinelemede, küme numaraları ilgili BSRAM'e yazıldıktan sonra, sonra her küme için toplanan değerler, *XilinxCoreGen* ile sentezlenen 16 bölme devresi kullanılarak, o kümedeki örüntü sayısına bölünerek kümelerin merkez noktaları bulunur. Bu şekilde merkez noktaları belirlenen 96 Gauss fonksiyonunun değişintileri, Bölüm 3.3.2'de anlatılan *k-en yakın komşu* (k-nearest neighbor) yöntemi kullanılarak basit bir aritmetik devre ile hesaplanır. Merkez vektörlerinin her bir bileşeni ve değişintiler 8-bit sabit noktalı sayılar ile temsil edilmiş ve son değerleri ilgili BSRAM'lere yazılmıştır.

6.3 Gauss Birimi

Merkez noktaları ve değişintileri bulunan Gauss fonksiyonlarının değerleri *u_gauss* birimi tarafından hesaplanır. Eş.3.1'de verilen Gauss fonksiyonları üstel değerler içerdiğinden, değerlerin donanım üzerinde bu şekilde hesaplanması oldukça maliyetli ve yavaş olmaktadır. Merkezleri ve değişintileri bilinen bu fonksiyonlar artık tek bağımsız değişkenli hale gelmiştir ve o değişken öznitelik vektörü olan $\bar{x}(p)$'dir. O halde, $G[\bar{x}(p)] = y$ şeklinde yazılabilen bu fonksiyonların değerleri bir başvuru çizelgesinde tutularak, ilgili değişken geldiğinde uygun değer atanabilir.

Ancak öznitelik vektörü 128-bit olduğundan, tüm olası $x(p)$ değerleri göz önüne alınırsa, 96×2^{128} değer içeren bir başvuru çizelgesine gereksinim vardır. Bu pratikte mümkün değildir. Ayrıca 16 boyutlu $x(p)$ vektörü yerine, tek boyutlu $\left\| \overline{x}(p) - \overline{\mu}_j \right\|$ skalarını örneklemek akıllıca olacaktır. Her bir Gauss fonksiyonu için 32 ve toplamda 96x32x16-bit (6KB) örnek için alınan değerler bir başvuru çizelgesinde saklanmıştır. Gauss birimi Şekil 6.5'deki gibi 3 alt birimden oluşur.



Şekil 6.5 *u_gauss* ve alt birimleri

6.3.1 Başvuru Çizelgesi

Başvuru çizelgesi, 16-bit genişliğinde 3x96 adet dRAM'den oluşur. BSRAM yerine dağılmış RAM (distributed RAM – dRAM) kullanılmıştır. Bunun nedeni, her biri 18 Kbit olan BSRAM'lerin az sayıda veri içeren yapılarda verimsiz şekilde kullanılmasını önlemektir. Başvuru çizelgesinin düzeni Çizelge 6.2'de verilmiştir.

Çizelge 6.2 Başvuru çizelgesi düzeni

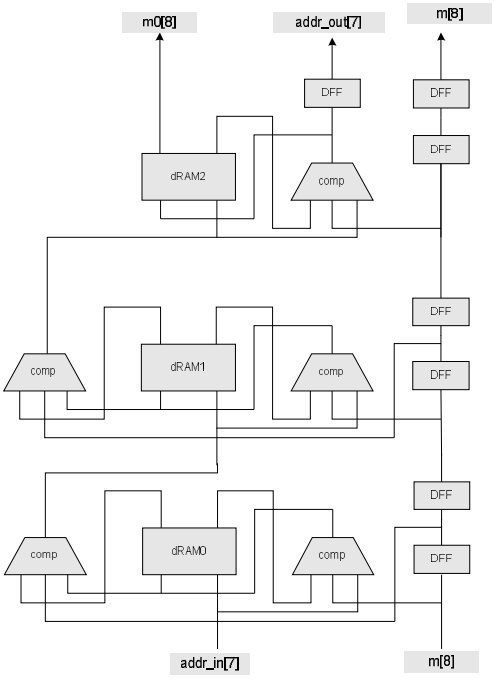
Değer	**Sinyal Gösterimi**	**Örnek**
$m = \left\| \overline{x}(p) - \overline{\mu}_j \right\|^2$	8-bit sabit nokta	32
$G_j\left[\overline{x}(p)\right]$	8-bit sabit nokta	32

6.3.2 Arama Birimi

Arama birimi olan *u_search* (Bkz. Şekil 6.6), 128-bit sabit nokta gösterimli öznitelik vektörünü ilgili BSRAM'den alır ve öncelikle bu vektörün merkeze olan uzaklığını Bölüm 6.2'de verilen Manhattan devresi ile hesaplar. Daha sonra LUT üzerinde bu vektöre eşit değerin adresini bulmaya çalışır. Bu, büyük bir olasılıkla gerçekleşmeyeceğinden, gelen uzaklık kriteri değerinden (m) küçük en büyük değerin adresi bulunur. Arama, ilgili 5-bit uzunluğundaki adresin ardışık olarak yaklaşıklanması yöntemiyle bulunur. Bu sayede arama, adres yolundaki bit sayısı kadar adımda tamamlanır. Her bir Gauss fonksiyonu için 3 dRAM ve 5 karşılaştırıcı kullanılmıştır.

Bu aşamalar şu şeklide özetlenebilir:

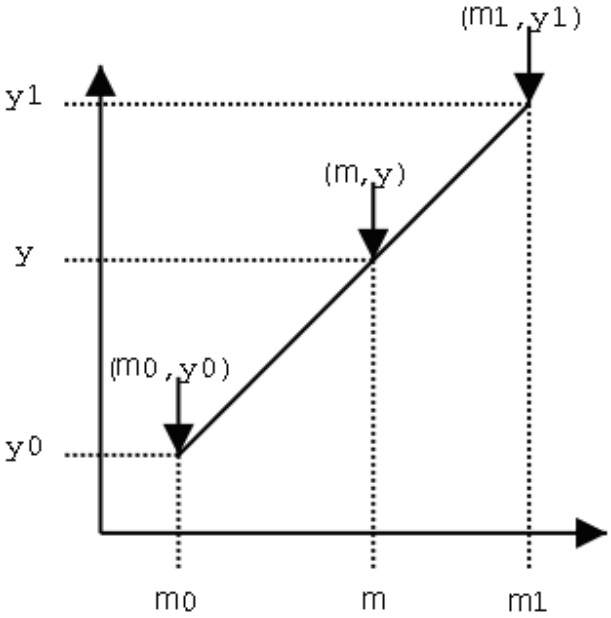
- Adresin en önemli biti (most significant bit – MSB) yükseğe çekilir ve öznitelik vektörü ile adresin karşılığı olan değer karşılaştırılır.
- Adreste okunan değer öznitelik vektöründen büyükse MSB düşüğe çekilir, aksi takdirde olduğu gibi bırakılır. (LUT biriminde değerler küçükten büyüğe doğru sıralanmıştır)
- Bu süreç sırasıyla tüm adres bitleri için tekrarlanır.



Şekil 6.6 *u_search* biriminin mimarisi

6.3.3 Aradeğerleme Birimi

Aradeğerleme (interpolation) birimi olan *u_int*, *u_search* tarafından bulunan değer ($m0$) ile başvuru çizelgesinde bir üstte yer alan değere ($m1$), doğrusal aradeğerleme uygular ve uzaklık kriterine (m) karşılık gelen Gauss fonksiyonunun yaklaşık sonucunu hesaplar. Doğrusal aradeğerleme Şekil 6.7'deki gibi gerçekleşir.



Şekil 6.7 Doğrusal aradeğerleme

Buna göre, bulunması istenen değer $y = G(m)$, $m0$'a karşı gelen değer $y0$ ve $m1$'e karşı gelen değer $y1$ ile gösterilmiştir. Doğrusal aradeğerleme Eş.6.2'de verildiği gibi yapılır:

$$y = y0 + \left(\frac{m - m0}{m1 - m0} \right) (y1 - y0). \qquad (6.2)$$

Aradeğerleme birimi *XilinxCoreGen* ile sentezlenen *pipeline* çarpma ve bölme devrelerinden oluşur.

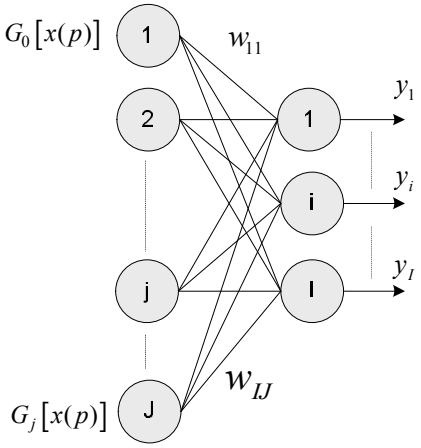
6.4 LMS Birimi

LMS birimi (*u_lms*), gizli katman ile çıkış katmanı arasındaki bağlantı değerlerinin bulunmasında kullanılır. RTFA'da gizli-çıkış katmanı modeli Şekil 6.8'de verilmiştir. LMS algoritması buna göre aşağıdaki eşitliklerde verildiği gibi iki aşamada gerçekleşir. Simgelemde kolaylık olması açısından Gauss fonksiyon değerini belirten $G[x(p)]$ yerine u kullanılacaktır.

$$\overline{y}(n) = \underline{w}(n)\overline{u}(n) \tag{6.3}$$

$$\begin{aligned} \overline{e}(n) &= \overline{d}(n) - \overline{y}(n) \\ \underline{w}(n+1) &= \underline{w}(n) + \eta\overline{u}(n)\overline{e}(n) \end{aligned} \tag{6.4}$$

Eşitliklerde n özyineleme numarasını göstermektedir. Eş.6.3'te verilen eşitlik bir vektör-matris çarpma işlemidir. Eş.6.4 ise bağlantıları güncellemek için kullanılmaktadır.

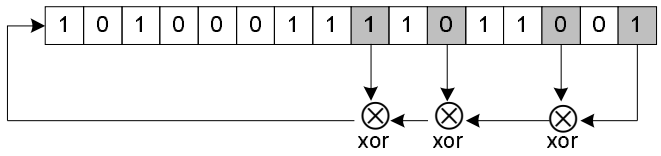


Şekil 6.8 RTFA'nın gizli ve çıkış katmanları

6.4.1 Doğrusal Kayan Geri-bildirimli Yazmaç

Doğrusal kayan geri-bildirimli yazmaç (linear shift feedback register – LSFR), girişi çıkışının bir önceki durumunun doğrusal bir fonksiyonu olan kayan yazmaçtır (shift register). LSFR'nin ilk girdi değeri *seed* olarak adlandırılır. Bir sonraki durumu

etkileyen bitlerin konumu *tap* dizisi olarak tanımlanır. Şekil 6.9'daki LSFR'nin tap çıkışları [16 14 11 9] olarak yazılır. Bu bitler gri renkle gösterilmiştir.



Şekil 6.9 Doğrusal kayan geri-bildirimli yazmaç

Beş tanesi paralel olarak gerçeklenen *u_lsfr* birimi gizli katman ile çıkış katmanı arasındaki bağlantılara ilk değer atamada kullanılır. Bu değerler küçük rasgele değerler olacağından, 8-bitlik bağlantı değerlerinin 6. ila 4. bitleri için rasgele değerler üretilmemiş, bu bitlere 0 atanmıştır. Bağlantılar, çift portlu BSRAM'lerde Gauss fonksiyon değerleri ile birlikte saklanmıştır.

6.4.2 Çarpma Toplama Devresi

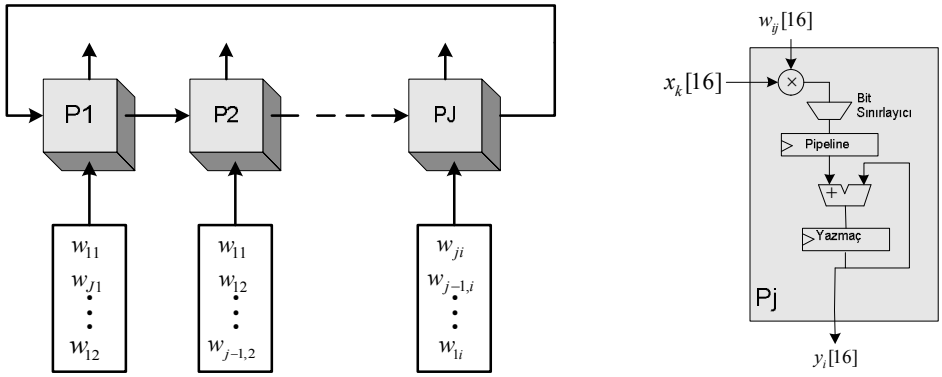
LMS algoritması yoğun ve yinelene çarpma işlemlerine gereksinim duyar. Bu nedenle verimli çarpma yapabilen çarpma bloklarına gereksinim vardır. Bu bloklar Bölüm 5.1.2'de değinilen nedenlerle *XilinxCoreGen* tarafından sentezlenen (Bkz. EK-2) *MULT18x18* birimlerinden oluşturulmuştur. Bu birimler, 18x18-bit *pipeline* çarpma yapabilen ve FPGA üzerinde gömülü olarak bulunan özgül birimlerdir. Pipeline derinliği kullanıcı tarafından düzenlenebilmektedir.

Eş.6.3 daha açık olarak aşağıdaki gibi yazılabilir.

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{6.5}$$

Eş.6.5'te tanımlı işlem bir sistolik çarpma-toplama devresi [66] (MAC) ile verimli bir şekilde yapılabilir. Sistolik çarpma ve toplama devresi Şekil 6.10'da verilmiştir. XC2V6000'de 144 adet *MULT18x18* birimi vardır. Bu nedenle gizli katmandaki 96 Gauss fonksiyonu için 96 işleme birimi (processing element – PE) kullanılabilir. Sistolik devrede her saat devrinde Gauss fonksiyon değerleri ve bağlantılar yatay

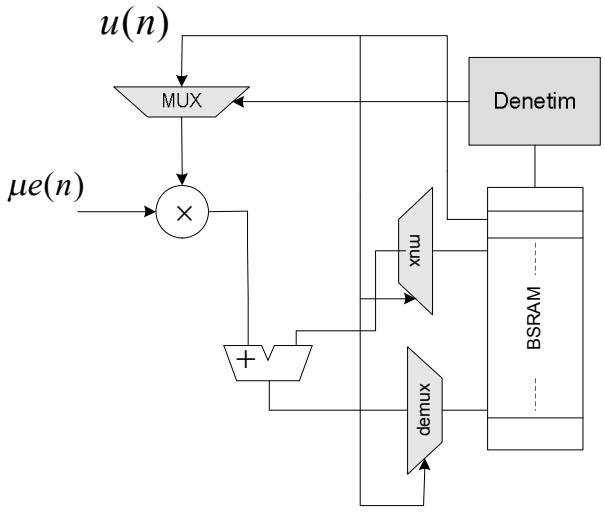
ve dikey kaydırıcılar ile kaydırılır ve bir kısmi toplam elde edilir. 96 işlem birim kullanıldığında 99 adımda çıkış vektörü bulunabilir. Devredeki tüm işlemler 8-bit sabit noktalı sayılar ile yapılmıştır.



Şekil 6.10 Sistolik çarpma ve toplama devresi

6.4.3 Bağlantı Güncelleme Birimi

Bağlantı güncellemek için kullanılan birimin mimarisi Şekil 6.11'de verilmiştir. Bağlantılar sadece eğitim sırasında güncelleneceğinden, paralel işlem yapılmamış, bir kerede bir bağlantı güncelleyen devre tasarlanmıştır.



Şekil 6.11 Bağlantı güncelleme birimi

6.5 Tasarım Sonuçları

RTFA için yüksek paralel işleme gücüne ve pipeline yapıya sahip bir donanım mimarisi tasarlanmıştır. Üç ana birim arası geçişler için verilerin yeniden düzenlenmesi ve diğer görece az saat devri içeren önişleme süreci ihmal edilirse sistemin performansı analizi, T_{CLK} bir saat devrini temsil etmek üzere, aşağıdaki gibi yapılabilir.

K-ortalama kümeleme birimi, her özyinelemede $16T_{CLK}$ zamanında bir örüntü vektörüne ait kümeyi tayin etmekte ve ilgili kümeye ait kısmi toplamları hesaplamaktadır. Bu, eğitim setindeki 778 örüntü için 12448 T_{CLK} yapar. 16 bölme devresi kullanıldığından, her özyineleme sonunda 96+2 T_{CLK} zaman merkez vektörlerini ve değişintiyi güncellemek için kullanılır. Bu durumda K-ortalama küme algoritması $12546 \times n_K$ saat devrinde sonlanır. Burada n_k algoritma için özyineleme sayısıdır.

Her örüntü vektörü için bir Gauss fonksiyonunun adresinin bulunması, en kötü olasılık durumunda, $5 \times 2T_{CLK}$ zamanda olur. $4T_{CLK}$ zaman da Gauss fonksiyonun değerinin bulunması ve ilgili BSRAM'e yazılması sırasında harcanır. Bu 96 Gauss fonksiyonu ve 778 örüntü vektörü için $1045632T_{CLK}$ zaman alır.

Gizli katman ile çıkış katmanı arasındaki bağlantılara rasgele ilk değer atanması $28 \times (5+1)$ T_{CLK} , her özyinelemede vektör-matris çarpımının bulunması $(28 \times 3)T_{CLK}$ zaman alır. Yine her özyinelemede bağlantıların güncellenmesi $96 \times 28 \times 4T_{CLK}$ zamanda gerçekleşir. Bu durumda LMS algoritması $11004 \times n_L$ saat devrinde sonlanır. Burada n_L algoritma için özyineleme sayısıdır.

Buna göre yapılan testlerde sistem 778 pencereyi $2.03\text{M}T_{CLK}$ zamanında ve *Modelsim'*de yapılan benzetimlere uyarınca Çizelge 6.3'deki başarı oranı ile sınıflar.

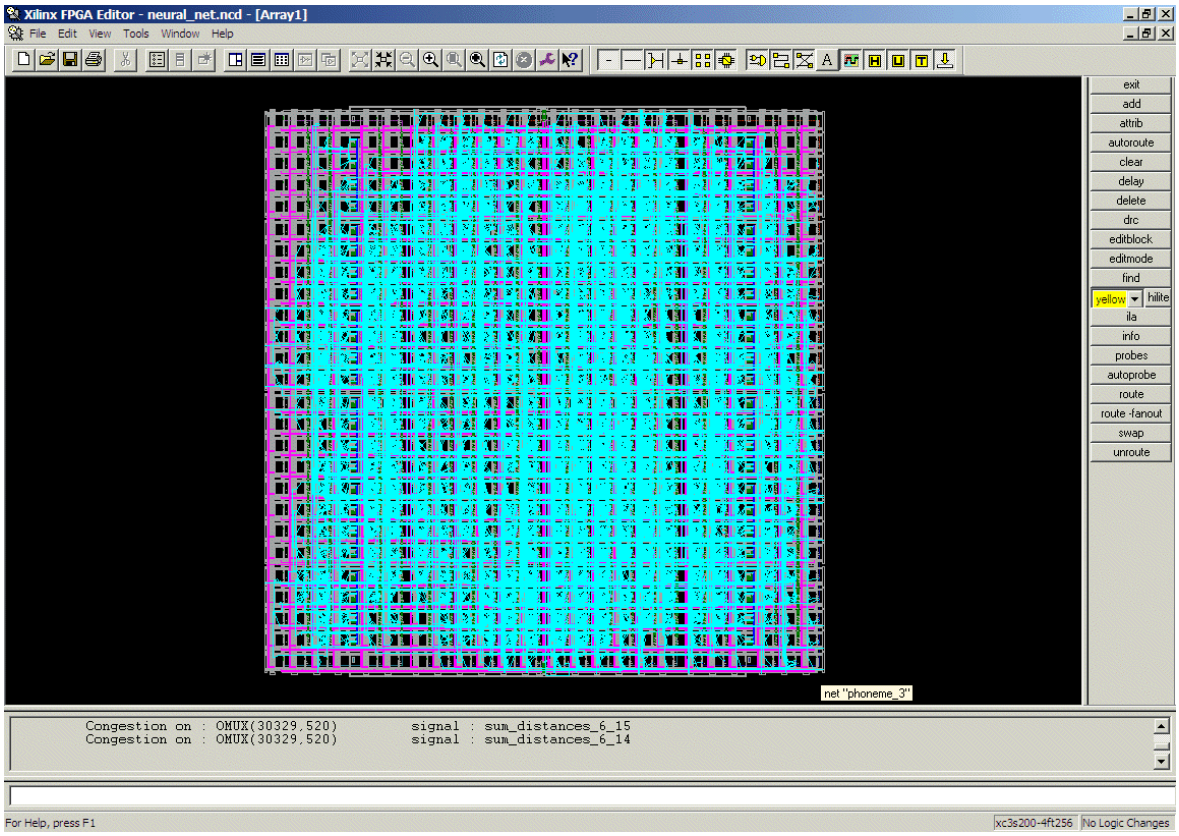
Çizelge 6.3 FPGA üzerinde fonem tanıma yüzdeleri

RTF Sayısı	Başarı Oranı (%)	
	Eğitim Seti	Test Seti
96	76	71

Sistemin XC2V6000 için XST ile yapılan sentez sonuçları Çizelge 6.4'te, FPGA serimi Şekil 6.12'de verilmiştir.

Çizelge 6.4 XC2V6000 Alan Raporu

Kaynak	Kullanılan	Toplam
I/O	149	824
CLB Dilimi	17958	33792
LUT	29119	67584
BSRAM	107	144
ÖZGÜL ÇARPICI	98	144



Şekil 6.12 FPGA Serimi

XST, tasarımı 14.27 MHz saat hızında sentezlemiştir. Bu durumda $T_{CLK} = 70$ ns'dir ve 778 pencere 0.14sn'de eğitilmiştir. Bu 5557 pencere/sn hızına karşılık gelmektedir. Bir sözcükte ortalama 30-35 pencere olduğu varsayılırsa tasarlanan donanımla saniyede ortalama 160 sözcük ile eğitilebilen bir ses tanıma sisteminin altyapısı oluşturulmuştur denilebilir.

BÖLÜM 7

7 SONUÇ

Tasarlanan donanımın yüksek paralel işleme gücü sayesinde oldukça hızlı sayılabilecek veri işleme yeteneği vardır. Bunda kullanılan FPGA'a özgü pipeline aritmetik işleyicilerin de rolü büyüktür. Ancak tartışılması gereken nokta bu hızın gerekliliği olabilir. Günümüz ASIC ve FPGA sistemlerinde güç ve enerji tüketimi performans kadar önemli sayılabilecek bir kriter olmuştur. Hızın artması, harcanan gücün ve silikon alanın artması sonucunu doğurmaktadır. Bu bakımdan tasarlanan donanımın güç yönünden bir analizinin yapılmaması eleştirilebilecek noktalardan biridir.

Diğer yandan sistemin Matlab ile yapılan başarı testlerinde tanıma oranı %87'lerde kalmıştır. Bunda en büyük pay, donanım mimarisi için harcanan zaman nedeniyle, sinyal işleme ve fonem kesimleme gibi konulara yeterince vakit ayrılamamasıdır. RTFA'da bulunan K-ortalama kümeleme algoritması, merkez değerlerine yapılan ilklendirilmeye oldukça duyarlıdır. Bu atamanın rasgele değerler yerine SFF (Subset Furthest-First) vb. bir yöntemle yapılması başarı oranını arttırabilecek önlemlerden biri olabilir. Alternatif olarak, K-ortalama / LMS ikilisi yerine DDA [67] (Dynamic Decay Adjustment) öğrenme algoritmasının daha başarılı sonuçlar verilip verilmediği araştırılabilir.

Sistemin donanım tasarımda tanıma oranının bir miktar düşmesinin nedenlerinden biri 8-bit sabit nokta kullanımından kaynaklanan niceleme hatasıdır. Ayrıca örneklenecek Gauss fonksiyon sayısının fazla olması nedeniyle az örnek kullanılması başarı oranını etkilemiştir. Bunun çözümü Gauss fonksiyonunun başvuru çizelgeleri yerine CORDIC [68] yöntemi ile modellenmesi olabilir.

Bütün bu özeleştirilere rağmen, Hacettepe Üniversitesi VLSI laboratuarı olarak YSA gibi uyarlanabilir sistemlerin donanım tasarımında oldukça tecrübe kazanılmıştır. Temmuz 2006'da geri-yayılım algoritmasının sistolik dizilerle gerçeklenmesi ile ilgili bir konferans makalesi [66] TAINN206[1]'da sunulmuştur.

[1] Turkish Artificial Intelligence & Neural Networks Symposium

Tezin kapsamında yapılan çalışmalar ile ilgili başka bir makale[1] de yayın kararı beklemektedir.

Tasarlanan donanım bir fonem sınıflandırma sistemidir ve Bölüm 4'te anlatılan sinyal işleme aşamalarını da kapsayacak şekilde mimarisinin genişletilmesi gelecekte yapılabilecek çalışmalardan biridir. Diğer yandan FPGA üzerinde gerçeklenen sistem, bir takım değişiklikler yapılarak ASIC olarak sentezlenebilir.

[1] IEEE Transactions on Consumer Electronics

KAYNAKLAR

[1] Tebelskis, J., Speech Recognition using Neural Networks, Carnegie Mellon University Tez, 1995.

[2] Mengüşoğlu, E., Bir Türkçe Fonem İfade Tanıma Sisteminin Kural Tabanlı Tasarımı ve Gerçekleştirimi, H.U. Tez, 1999.

[3] Halliday, D., Resnick, R., Fundamentals of Physics, John Wiley and Sons, 1981.

[4] Artuner, H., Bir Türkçe Fonem Kümeleme Sistemi Tasarımı ve Gerçekleştirimi, H.U. Tez, 1994.

[5] Stevens, S.S., Volkmann, J., Newman E.B., A Scale for the Measurement of the Psychological Magnitude Pitch, The Journal of the Acoustical Society of America, vol.8, issue.3, pp.185-190, 1937

[6] Savcı, F., Sesli İfade Tanıma için otomatik bir özellik çıkarımı dizgesinin tasarımı ve gerçekleştirimi, H.U. Tez, 1994.

[7] Arkın, E., Türkçe Sesbirimlerinin Sınıflandırılması İçin Bir Bulanık Sinir Ağının Tasarımı ve Gerçekleştirimi H.U. Tez, 2004.

[8] Demircan, Ö., Türkiye türkçesinin Ses Düzeni Türkiye Türkçesinde Sesler, Türk Dil Kurumu Yayınevi, 1979.

[9] Rabiner, L.R., A Tutorial on Hidde Markov Models and Selected Applications in Speech Recognition, Proceedings of IEEE77(2), vol.2, issue.77, pp.257, 1989.

[10] Lee, K.F., Automatic Speech Recognition, The Development of the SPHINX System, Kluwer Academic Publishers, 1989.

[11] Viterbi, A.J., Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Dlgorithm, IEEE Transactions on Information Theory, vol.13, issue.2, pp.260–269, 1967.

[12] Veeravalli, A.G., Pan, W.D., Adhami, R., Cox, P.G., A Tutorial on Using Hidden Markov Models for Phoneme Recognition, Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, pp. 154- 157, 2005.

[13] Haykin, S., Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999.

[14] Kangas, J., Torkkola, K., Kohonen, M., Using SOMs as Feature Extractors for Speech Recognition, IEEE International Conference on Acoustics, Speech, and Signal Processing, 1992.

[15] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K., Phoneme Recognition: Neural Networks vs. Hidden Markov, International Conference on Acoustics, Speech, and Signal Processing, 1988.

[16] Arriola, Y., Carrasco, R.A., Real-time automatic speech recognition using HMM and Neural Networks, SBT/IEEE International Telecommunications Symposium, 1990.

[17] Çankaya, İ., Gürgen, F., Güzeliş, C., Phoneme Recognition by Radial Basis Function Network Using Input-Output Clustering, Turkish Journal of Electrical Engineering and Computer Sciences, vol.4, 1996.

[18] Wan, E.A., Temporal backpropagation for FIR neural networks, IJCNN International Joint Conference on Neural Networks, 1990.

[19] Rumelhart, D.E.., Hinton, G.E., Learning Representations by Back-propagating Errors, Nature, 1/27, pp. 533, 1985.

[20] Park, J., Sandberg, I.W., Universal Approximation Using Radial Basis Function Networks, Neural Computation v.3, i.2, pp.246-257, 1991.

[21] Park, J., Sandberg, I.W., Approximation and Radial Basis Function Networks, Neural Computation, vol.5, issue.3, pp.305-316, 1993.

[22] Girosi, F., Poggio, T., Networks and Best Approximation Property, Technical Report AIM-1164, Massachusetts Institute of Technology, 1989.

[23] Moody, J., Darken, C., Learning with Localized Receptive Fields, Proceedings of the 1988 Connectionist Models Summer School, 1988.

[24] Moody, J., Darken, C., Fast Learning in Networks of Locally Tuned Processing Units, Neural Computation, v.1, pp. 281-294, 1989.

[25] MacQueen J.B., Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967.

[26] Kohonen, T., Oja, E., Simula, O., Visa, A., Kangas, J., Engineering applications of the self-organizing map, Proceedings of the IEEE, vol.84, issue.10, pp.1358-1384, 1996.

[27] Bors, A.G., Pitas, I., Median Radial Basis Function Neural Network, IEEE Transactions on Neural Networks, vol.7, issue.6, pp.1351-1364, 1996

[28] Christodoulu, C., Georgiopoulos M., Applications of Neural Networks in Electromagnetics, Artech House Inc., 2001.

[29] The CSLU Toolkit, http://www.cslu.ogi.edu/toolkit/ adresinden elde edilebilir (2007).
[30] Parson, T., Voice and Speech Processing, McGraw-Hill Inc., 1986.
[31] Morris, L.R., A PC Based Digital Speech Spectrograph, 1988
[32] Hyun, D., Lee, C., Optimization of mel-cepstrum for speech recognition, IEEE International Conference on Systems, Man, and Cybernetics, vol.1, pp.500-503, 1999.
[33] Davis, S., B., Mermelstein, P., Comparison of parametric representations for Monosyllabic Word Recognition in Continuously Spoken Sentence, IEEE Transactions on Acoustic, Speech and Signal Processing, vol.28, issue.4, pp.357-366, 1980.
[34] Picone, J., Signal Modeling Techniques in Speech Recognition, Proceedings of IEEE, vol.81, issue.9, 1215-1247, 1993.
[35] Kohonen, T., The Neural Phonetic Typewriter, IEEE Computer, vol.21, issue.3, pp.11-22, 1988.
[36] Blazer, W., Takahashi, M., Ohta, J., Kyuma, K., Optoelectronics Implementation of a Learning Boltzman Machine, Proceedings on International Conference on Fuzzy Logic and Neural Networks, pp.651, 1990.
[37] Fang, W.C., Sheu, B.J., Lee, J.C., Real Time Computing of Optical Flow Using Adaptive VLSI Neuroprocessors, IEEE Intl. Conference on Computer Design, pp.122-125, 1990.
[38] Farhat, N.H., Psalitis, D., Prate, A., Paek, E., Optical Implementation of Hopfield Model, Applied Optics, v.24, n.10, pp.1469-1475, 1985
[39] Otha, J., Nitta, Y., Kyuma, K., Dynamic Optical Neurochips, Proceedings of Intl. conference of Fuzzy Logic and Neural Networks, pp.661-664, 1990
[40] Psalitis, D., Yamamura, A., Hsu, K., Lin, X.G., Brady, D., Optoelectronic Implementations of Neural Networks, IEEE Communications Magazine, pp.37-40, 1989.
[41] Delcorso, D., Reyneri, L., Mixing Analog and Digital Techniques for Silicon Neural Networks, IEEE Symposium On Circuits and Systems, 1990.
[42] Dicataldo G., Palumbo, G., A Neural A/D Converter Utilizing Schmitt Trigger, IEEE Symposium On Circuits and Systems, 1990.

[43] Lee, B.W., Sheu, B.J., Design of a Neural Based A/D Converter Using Modified Hopfield Network, IEEE Journal on Solid State Circuits, vol.24, issue. pp.1129-1135, 1990.

[44] Moore, A., Corso, D.D., Tarassenko, L., Digital Analog Hybrid Synapse Chips for Electronic Neural Networks, Kaufmann Inc, pp.769-776, 1990.

[45] Smith, M.J.S., Application Specific Integrated Circuits, Addison-Wesley Inc., 1997.

[46] Virtex-II Complete Data Sheet, Xilinx Inc., http://direct.xilinx.com/ bvdocs/publications/ds031.pdf adresinden elde edilebilir (2007).

[47] Mori, J., Nagamatsu, M., Hirano, M., Tanaka, S., Noda, M., Toyoshima, Y., Hashimoto, K., A 10 ns 54×54 b parallel structured full array multiplier with 0.5 μm CMOS technology, IEEE Journal of Solid-State Circuits, v.26, i.4, pp. 600-606, 1991.

[48] Chang, K.C., Digital System Design with VHDL and Synthesis, IEEE Computer Society, 1999.

[49] Lakshminarayanan, G., Venkataramani, B., Senthil Kumar, J., Yousuf, A.K., Sriram, G., Design and FPGA implementation of image block encoders with 2D-DWT, TENCON Conference on Convergent Technologies for Asia-Pacific Region, vol.3, pp. 1015-1019, 2003.

[50] Nichols, K., A Reconfigurable Computing Architecture for Implementing Artificial Neural Networks on FPGAs, University of Guelph Tez, 2003.

[51] Beuchat, J.L., Haenni, J.O., Sanchez, E., Hardware Reconfigurable Neural Networks, 5th Reconfigurable Architectures Workshop, 1998.

[52] Eldredge, J.G., Hutchings, B.L., RRANN: The Run-Time Reconfiguration Artificial Neural Network, IEEE Custom Integrated Circuits Conference, 1997.

[53] Nordstrom, T., Davis, E.W., Svensson, B., Issues and Applications Driving Research in Non-conforming Massively Parallel Processors, Proceedings of the New Frontiers, a Workshop of Future Direction of Massively Parallel Processing, 1992.

[54] Andres, P.U., Structure-Adaptable Digital Neural Networks. Swiss Federal Institute of Technology-Lausanne Tez, 1999.

[55] Gadea, R., Cerda, J., Ballester, F., Macholi, A., Artificial Neural Network Implementation on a Single FPGA of a Pipelined On-line Backpropagation, The 13th International Symposium on System Synthesis, pp.225-230, 2000.

[56] Eldredge, J. G., FPGA Density Enhancement of a Neural Network Through Run-time Reconfiguration, Brigham Young University Tez, 1994.
[57] Ferrucci, A., Acme: A Field-programmable Gate Array Implementation of a Self Adapting and Scalable Connectionist Network, University of California Tez, 1994.
[58] Martin, M.H., A Reconfigurable Hardware Accelerator for Back-propagation Connectionist Classifiers, University of California Tez, 1994.
[59] Skrbek, M., Fast neural network implementation. Neural Network World, vol.9, issue.5, pp.375-391, 1999.
[60] Yang, F., Paindavoine, M., Implementation of an RBF Neural Network on Embedded Systems: Real-time Face Tracking and Identity Verification, IEEE Transactions on Neural Networks, vol.14, issue.5, pp.1162-1175, 2003.
[61] Garis, H., Korkin, M., The Cam-brain Machine, An FPGA-based Hardware Tool which Evolves a 1000 Neuron Net Circuit Module in Seconds and Updates a 75 Million Neuron Artificial Brain for Real Time Robot Control, Neurocomputing Journal, vol.42, pp.1-4, 2002.
[62] Stevenson, D., IEEE Task P754: A Proposed Standard for Binary Floating-point Arithmetic, IEEE Computer vol.14, issue.3, pp. 51–62, 1981.
[63] Li, X., Moussa, M., Areibi, S., Arithmetic Formats for Implementing Artificial Neural Networks on FPGAs, Canadian Journal of Electrical and Computer Engineering, vol.31, issue.1, pp.31-40, 2006.
[64] Holt, J.L., Baker, T.E., Backpropagation Simulations Using Limited Precision Calculations, International Joint Conference on Neural Networks (IJCNN-91), vol.2, pp.121-126, 1991.
[65] Gokhale, M., Frigo, J., McCabe, K., Theiler, J., Lavenier, D., Early experience with a Hybrid Processor: K-means Clustering, First International Conference on Engineering of Reconfigurable Systems and Algorithms, 2001.
[66] Ucar, A., Alkar, A.Z., HW/SW Codesign of FPGA-based Neural Networks, Proceedings of Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks, 2006.
[67] Aberbour, M., Mehrez, H., Architecture and design methodology of the RBF-DDA neural network, Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, vol.3, pp.199-202,1998.

[68] Meyer-Bäse, A., Watzel, R., Meyer-Bäse, U., Foo, S., A Parallel CORDIC Architecture Dedicated to Compute the Gaussian Potential Function in Neural Networks, Engineering Applications of Artificial Intelligence, vol.16, issue.7-8, pp.595-605, 2003.

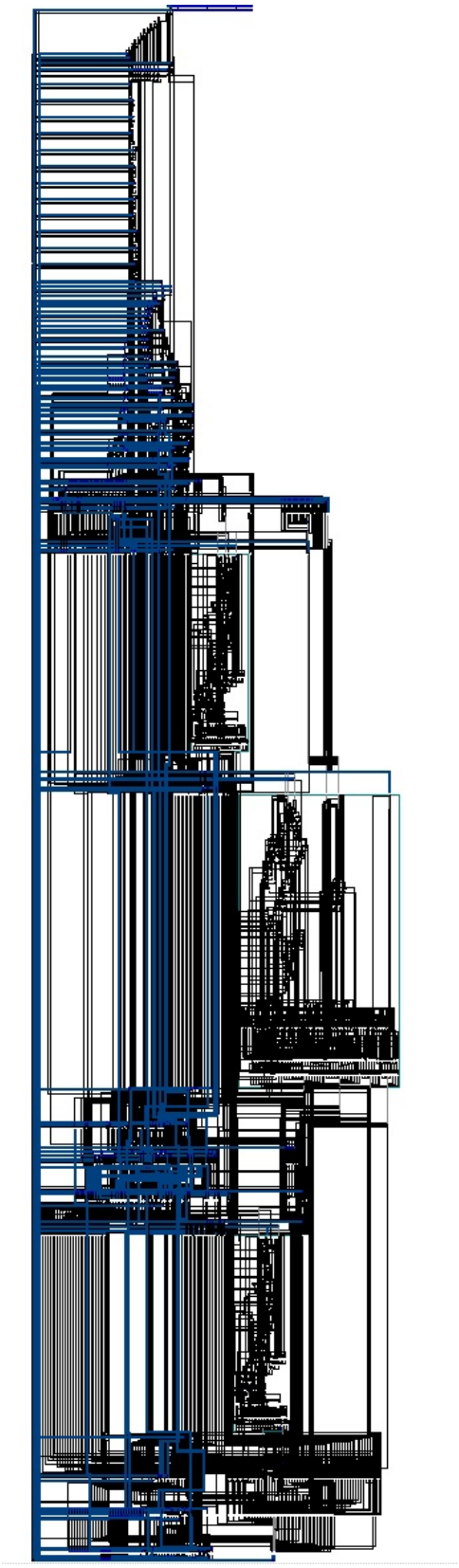
EKLER DİZİNİ

EK-1 Eğitim için Kullanılan Korpüs.

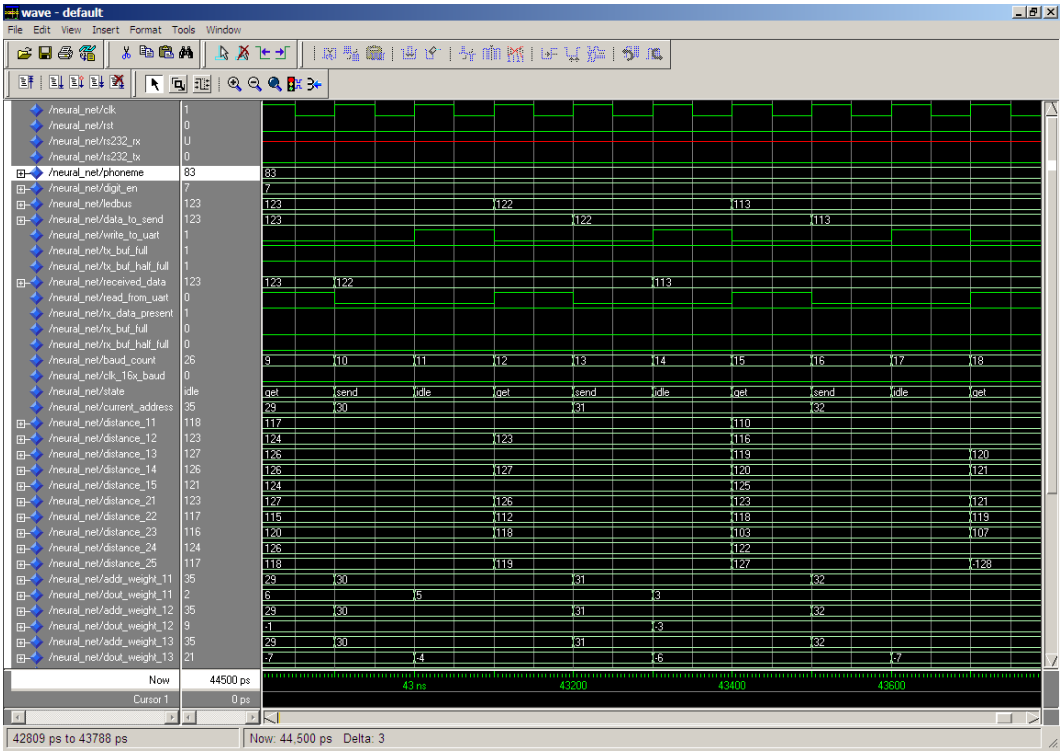
Çizelge EK-1.1 Eğitim için kullanılan korpüs

/a/	an kaplan baba savaş koca aç lamba mala ırmak fırça
/e/	erek sen resim emek dere ezmek ben sekiz evet perde
/ı/	ırmak fırsat kımız kalın cadı ısı kırbaç sıcak hayır altı
/i/	iki bir simit demir kirpi ilke bin viraj sekiz yedi
/o/	on kol yorum dekor büro okul zorba dokuz sifon kilo
/ö/	örtü yöntem kömür karagöz tuzgölü ölü törpü jöle amatör eminönü
/u/	umut dur su otur kurdu un kur kuru odun kulunuz
/ü/	üleşmek kül sürü ölçüm düşünür üryan kurdan müze kültür gücü
/b/	ben kablo hibe teşebbüs bilgi kobra kalbe lakabı
/c/	cin sac acı topacı ceviz secde kucak teveccüh
/ç/	çamur aç açık kıraç çoban üç fırça süreç
/d/	demir ad kadı üstadı düz gaddar zorda teyidi
/f/	fırsat kof sefil sedef fare köfte kefe zarif
/g/	güz agresif bilgi program git sigma sigara astrolog
/h/	hasta ah ahir siyah hiciv sıhhat sahil tarih
/j/	jale ajda proje baraj jilet hijyen enerji mesaj
/k/	kan aktif sıkı mekik kuru ekmek koku erkek
//	lale altın köle sakal lisan kıl sulu nasıl
/m/	mal şamdan çamur kilim meze şimdi umut sürüm
/n/	nice konser ana kalın nohut anne tanı lütfen
/p/	perde iplik yapı hesap pide çıplak süpürge geçip
/r/	rakip arpa kuru seher rekor serçe yorum ayar
/s/	sıcak hasta asalet takas sinir kas mesele ağustos
/ş/	şehit iş kasar savaş şahit baş kişi barış
/t/	tarak mat satır arnavut tazı tutmak sitem saat
/v/	virane gevsek sıv sınav ve havza dövülen otomotiv
/y/	yazı ayna ayar saray yedi kıyma kaya tümüyle
z/	zor ezme azeri dikiz zil izci kuzu kiraz

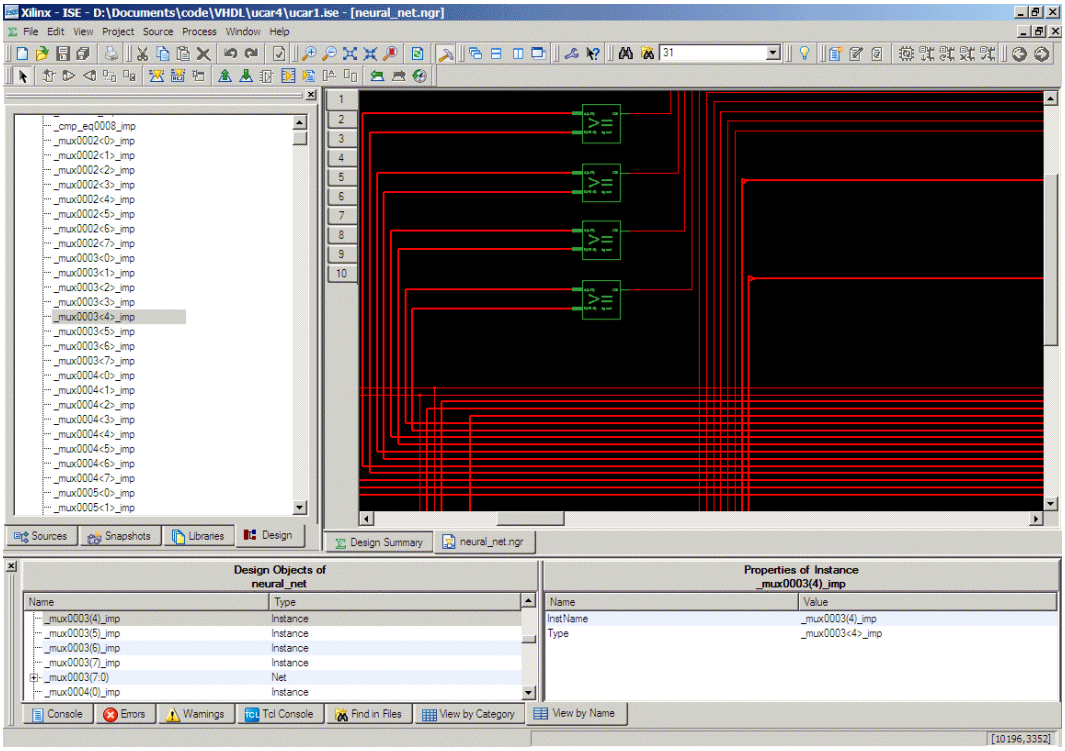
EK-2 Kullanılan Programların ve Devrelerin Görünümleri



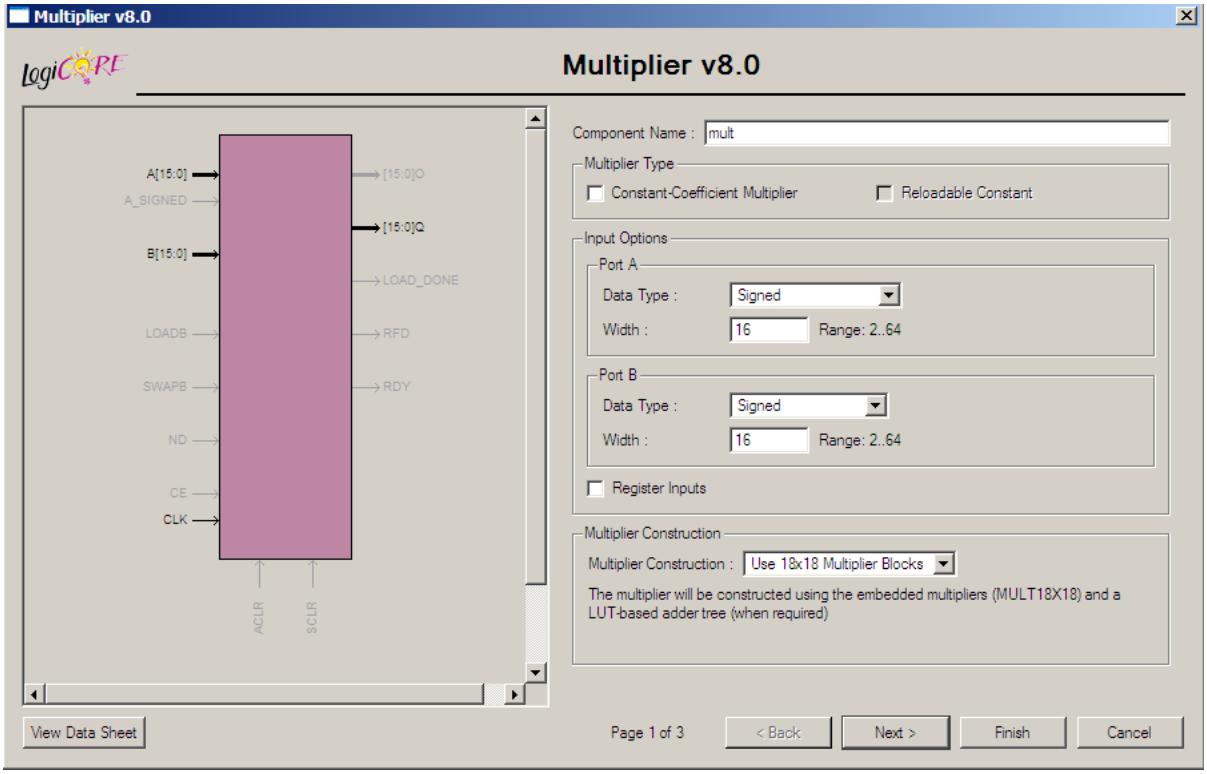
Şekil EK-2.1 Tasarımın RTL Şematiği



Şekil EK-2.2 Modelsim Benzetim Ortamı



Şekil EK-2.3 XST Sentez Ortamı



Şekil EK-2.4 *XilinxCoreGen* ile çarpma bloğu sentezleme

ÖZGEÇMİŞ

Adı Soyadı : Alper UÇAR

Doğum Yeri : Ankara

Doğum Yılı : 1979

Medeni Hali : Bekar

Eğitim ve Akademik Durumu:

Lise :	1993 – 1997	Atafen Fen Lisesi
Lisans:	1997 – 1999	İstanbul Üniversitesi Fizik Bölümü
	1999 – 2003	Yeditepe Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü

Yabancı Dil: İngilizce

İş Tecrübesi:

2004 - : Araştırma Görevlisi, Hacettepe Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü.